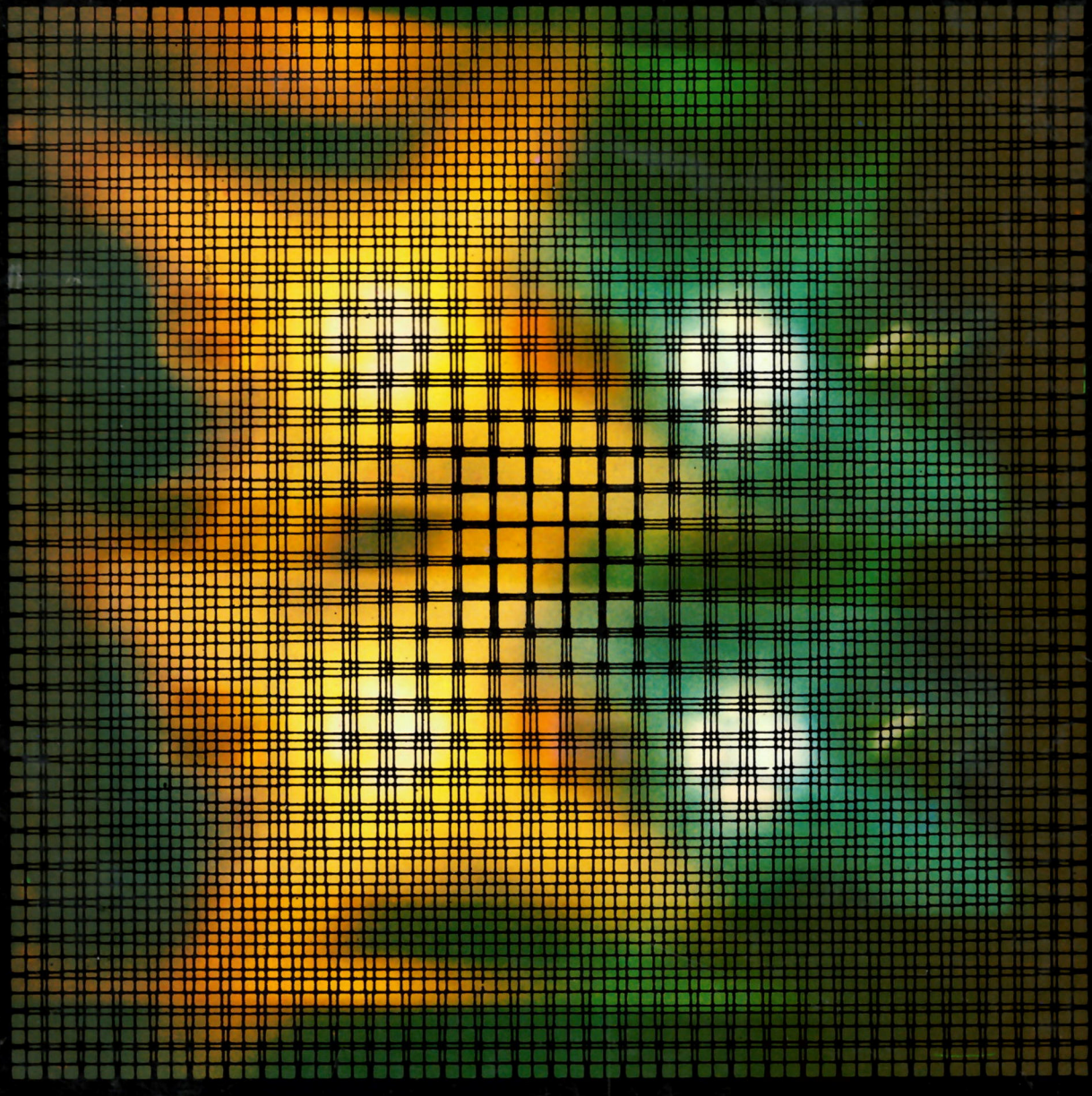
Fundamentos de Computadores Digitais

Thomas C. Bartee



4ª edição

Fundamentos de Computadores Digitais

Thomas C. Bartee

Harvard University

Traduzido por

Ivan José de Albuquerque

Engenheiro Eletrônico — IME. Engenheiro Nuclear — IME. Professor de Eletrônica e Eletrotécnica pela Faculdade de Educação da UERJ. Professor Assistente de Eletrônica de Telecomunicações — Escola de Engenharia da UFRJ

GUANABARA DOIS

Título do original em inglês
Digital Computers Fundamentals

Copyright © 1977, 1972, 1966, 1960 by
McGraw-Hill, Inc.

Direitos exclusivos para a língua portuguesa
Copyright © by
Editora Guanabara Dois S.A.
Rio de Janeiro — RJ

1980

Reservados todos os direitos. É proibida a duplicação ou reprodução deste volume, ou de partes do mesmo, sob quaisquer formas ou por quaisquer meios (eletrônico, mecânico, gravação, fotocópia, ou outros), sem permissão expressa da Editora.

Fotocomposição da Editora Guanabara Koogan S.A.

PREFÁCIO

O propósito deste livro é apresentar, da maneira mais explícita possível, os princípios dos modernos computadores digitais. Muitas questões foram incluídas em uma seção separada de cada capítulo, e as respostas às questões selecionadas de números ímpares são dadas no fim do livro.

Esta edição inclui os principais desenvolvimentos do computador nos dias de hoje, tais como os microprocessadores, as memórias de pequenos discos magnéticos substituíveis, os circuitos integrados em larga escala e a microprogramação. Em conseqüência, considerável parte do livro foi reescrita, e por isso há também muitas questões novas. Além disso, recebi inúmeras sugestões e comentários de professores, alunos e usuários do livro, e aproveitei todas estas colaborações para melhorar a apresentação e a área focalizada.

O Cap. 1 descreve os empregos do computador nos negócios, na indústria e na ciência, apresentando também uma breve introdução à programação. Também foi incluído um histórico, assim como alguns conceitos básicos relativos ao uso do computador.

Os Caps. 2 a 5 introduzem algumas idéias e princípios básicos usados em todos os computadores digitais e nos instrumentos digitais, sistemas de comunicações digitais, sistemas de controle digital e, na verdade, em todos os dispositivos digitais. Estes capítulos descrevem os sistemas de numeração, a álgebra booleana, o projeto lógico e as principais linhas de circuito digital. As seções sobre contadores e, em geral, o projeto lógico que usa os multivibradores biestáveis *(flip-flops)* foram ampliados em um capítulo completo. O material sobre circuitos digitais foi atualizado, e incluíram-se os principais desenvolvimentos em novos circuitos.

Os Caps. 6, 7 e 8 tratam das operações aritméticas, de memórias e dos dispositivos de entrada-saída. O material sobre operações aritméticas explica inicialmente como a aritmética é executada nos computadores, e a seguir como as unidades aritméticas-lógicas se organizam nos computadores. O capítulo sobre memórias apresenta memórias de núcleos magnéticos e de circuito integrado seguidas pelas memórias de tambor, de disco, de fita e de outros dispositivos de memória de massa. Incluíram-se nesta edição os últimos desenvolvimentos nesta área, tais como cartuchos ou estojos de fita, discos magnéticos pequenos e substituíveis, memórias dinâmicas com circuito integrado etc. O capítulo sobre entrada e saída descreve os principais dispositivos de entrada-saída, incluindo também matéria sobre conversores analógicos/digitais, conversores digitais/analógicos, terminais, modems (moduladores/demoduladores) e acopladores acústicos.

Os dois últimos capítulos, 9 e 10, discutem primeiro a unidade de controle nos computadores modernos, incluindo uma descrição tanto do projeto lógico convencional do conjunto de circuitos de controle quanto da microprogramação de controle do computador. A organização global dos

computadores é então discutida, como também o são os tópicos importantes na arquitetura do computador — técnicas de endereçamento, organização da barra coletora, consertos de interrupção etc. Os computadores existentes são usados como exemplos das idéias apresentadas. Deu-se ênfase aos micro e aos minicomputadores, mas as idéias básicas em grandes sistemas, tais como multiprocessadores, computadores de bombeamento, multiprogramação etc., são também discutidas.

O livro está organizado de modo que o assunto sobre circuitos eletrônicos não seja abordado. Há diversas razões para tal. Em algumas escolas, os circuitos são estudados em curso separado; outras tratam superficialmente o assunto de circuitos, uma vez que a compreensão dos detalhes de circuito não é essencial para o entendimento das funções executadas pelos circuitos. Além disso, o uso difundido de circuitos integrados torna possível estudar-se a operação de um computador a partir de um módulo ou bloco, como nos Caps. 3 e 4, de modo que a operação do computador possa ser compreendida considerando-se os biestáveis *(flip-flops)* e portas sob um ponto de vista funcional ou operacional.

Os símbolos dos diagramas-bloco neste livro são os adotados pelo American National Standards Institute e pelo Military Standards.

Tive considerável ajuda e consultoria durante a preparação das várias edições deste livro. A primeira edição foi escrita no MIT Lincoln Laboratory, e serei sempre grato ao MIT e aos membros do laboratório pelo considerável apoio que me deram.

Muitos amigos e usuários do livro contribuíram para as últimas edições. Lamento não poder citá-los todos. Certamente os Professores Glen Goff, W.W. Peterson, Robert Carrol, Irving Reed e Irwin Lebow merecem destaque. Devo também agradecer aos assistentes de ensino em meus cursos por suas inúmeras sugestões. A Sra. E. M. Vadeboncoeur preparou o manuscrito desta edição, e agradeço-lhe penhoradamente por seu trabalho nesta e nas edições anteriores.

Thomas C. Bartee

ÍNDICE

1 **OPERAÇÃO DO COMPUTADOR,** 1
1.1 Calculadoras e computadores, 1
1.2 Diferenças entre calculadoras de mesa e computadores, 3
1.3 Computadores eletrônicos digitais, 3
1.4 Aplicação de computadores aos problemas, 5
1.5 Aplicações de computadores no comércio e na indústria, 6
1.6 Aplicações científicas, 7
1.7 Alguns tipos diferentes de sistemas de computadores, 8
1.8 O uso de computadores digitais nos sistemas de controle, 11
1.9 Computadores de propósitos especiais e de propósitos gerais, 13
1.10 Sistemas de computador em tempo compartilhado, 13
1.11 Componentes básicos de um computador digital, 14
1.12 Construção da memória, 15
1.13 Instruções, 15
1.14 Instrução de multiplicação, 18
1.15 Instruções de desvio, ramificação, salto ou quebra de seqüência, 19
1.16 Leitura do programa no computador, 20
1.17 Sistemas de programação, 25
1.18 Linguagens de montador, 27
1.19 Linguagens de compilador, 29
1.20 Uma breve introdução à linguagem PL/1, 30
Exercícios, 33

2 **SISTEMAS DE NUMERAÇÃO,** 37
2.1 O sistema decimal, 37
2.2 O sistema binário, 38
2.3 Contagem no sistema binário, 39
2.4 Adição e subtração em binário, 40
2.5 Multiplicação e divisão em binário, 40
2.6 Conversão de números decimais em binários, 42
2.7 Números negativos, 43
2.8 O uso de complementos para representar números negativos, 44
2.9 Complementos em outros sistemas de numeração, 45
2.10 Complementos de número binário, 45
2.11 Representação de número decimal codificado em binário, 46
2.12 Sistemas de numeração octal e hexadecimal, 48
Exercícios, 52

3 **ÁLGEBRA BOOLEANA E CIRCUITOS PORTA,** 61
3.1 Conceitos fundamentais da álgebra booleana, 62
3.2 Multiplicação lógica, 62

3.3 Portas OU e portas E, 63
3.4 Complementação e inversores, 64
3.5 Cálculo de expressões lógicas, 65
3.6 Cálculo de uma expressão que contém parênteses, 67
3.7 Leis básicas da álgebra booleana, 68
3.8 Prova por indução perfeita, 70
3.9 Simplificação de expressões, 71
3.10 Teoremas de De Morgan, 71
3.11 Dualidade básica da álgebra booleana, 72
3.12 Dedução de uma expressão booleana, 72
3.13 Combinação de portas, 74
3.14 Soma-de-produtos e produto-de-somas, 75
3.15 Dedução de expressões de produto-de-somas, 77
3.16 Dedução de uma expressão com três variáveis de entrada, 78
3.17 Portas NÃO-E (NE) e portas NÃO-OU (NOU), 80
3.18 O método de mapeamento para simplificação de expressões, 83
3.19 Subcubos e cobertura, 86
3.20 Expressões de produto-de-somas — *Don't-cares* (saídas não-especificadas), 90
Exercícios, 92

4 **PROJETO LÓGICO,** 105
4.1 O multivibrador biestável, 105
4.2 A transferência, 107
4.3 O relógio, 108
4.4 O registro de deslocamento, 111
4.5 O contador binário, 112
4.6 Contadores BCD (Decimal Codificado em Binário), 115
4.7 Circuitos integrados, 117
4.8 Integração em média e larga escalas, 122
4.9 Projetos de biestáveis com circuitos lógicos, 129
4.10 O biestável com circuitos lógicos, 129
4.11 O biestável mestre-escravo, 131
4.12 Projeto de contador, 133
Exercícios, 135

5 **CIRCUITOS DIGITAIS,** 142
5.1 Circuitos de computador, 142
5.2 Características de diodos e transistores usados em circuitos de chaveamento, 143
5.3 O seguidor de emissor, 149
5.4 A porta E, 150
5.5 O circuito E com diodos, 151
5.6 A porta OU, 152
5.7 O circuito inversor, 153
5.8 Portas NOU, 155
5.9 O biestável a transistor, 157
5.10 Lógica resistor-transistor, 158
5.11 Lógica diodo-transistor, 159
5.12 Lógica transistor-transistor, 161
5.13 Lógica acoplada pelo emissor, 167
5.14 Circuitos a semicondutores metal-óxido, 170
5.15 Circuitos lógicos de CMOS, 173
Exercícios, 177

6 **UNIDADE ARITMÉTICA-LÓGICA,** 185
6.1 Construção da UAL, 185
6.2 Representação de inteiros, 186
6.3 Circuito meia-soma binário, 187
6.4 O circuito somador completo, 188
6.5 Um somador binário paralelo, 190
6.6 Números positivos e negativos, 191
6.7 Adição no sistema de complemento de 1, 192
6.8 Adição no sistema de complemento de 2, 193
6.9 Adição e subtração num elemento aritmético paralelo, 194
6.10 O somador completo, 197
6.11 O somador de decimal-codificado-em-binário (BCD), 199
6.12 Números BCD positivos e negativos, 200
6.13 Adição e subtração no sistema de complemento de 9, 201
6.14 A operação de deslocamento, 207
6.15 Operações básicas, 207
6.16 Multiplicação binária, 211
6.17 Multiplicação em decimal, 213
6.18 Divisão, 214
6.19 Operações lógicas, 218
6.20 Sistemas de numeração de ponto-flutuante, 220
6.21 Efetuando operações aritméticas com números de ponto-flutuante, 224
Exercícios, 225

7 **O ELEMENTO DE MEMÓRIA,** 230
7.1 Memórias de acesso aleatório, 231
7.2 Armazenamento em núcleos magnéticos, 233
7.3 Armazenamento de informação em núcleos magnéticos numa matriz bidimensional, 235
7.4 Montagem de planos de núcleos numa memória de núcleos, 238
7.5 Seqüência de sincronização, 240
7.6 Características do conjunto de memória de núcleos, 241
7.7 Endereço da memória e registros de memória intermediária, 242
7.8 Excitação das linhas de seleção X e Y, 243
7.9 Registros de memória intermediária e conjunto de circuitos associado, 244
7.10 Características da memória de núcleos, 246
7.11 Decodificadores, 246
7.12 Organização da memória de núcleos e esquemas da fiação, 249
7.13 Memórias de acesso aleatório a semicondutor, 255
7.14 Memórias bipolares de CI, 255
7.15 Memórias estáticas de MOS, 258
7.16 Memórias dinâmicas, 262
7.17 Memórias de leitura exclusiva, 264
7.18 Armazenamento em tambor magnético, 270
7.19 Operação em série e em paralelo de um tambor magnético, 271
7.20 Memórias de discos magnéticos, 274
7.21 Sistemas de armazenamento em disco flexível — o disco floppy, 279
7.22 Fita magnética, 280
7.23 Estojos e cassetes de fita, 285
7.24 Técnicas de gravação digital, 289
7.25 Técnicas de gravação de retorno a zero e de retorno à magnetização fundamental, 290
7.26 Técnicas de gravação de não-retorno a zero, 294
Exercícios, 296

8 **DISPOSITIVOS DE ENTRADA-SAÍDA,** 300
8.1 Diferenças entre as aplicações comerciais e científicas, 301
8.2 Fita perfurada, 301
8.3 Leitoras de fita, 304
8.4 Cartões perfurados, 306
8.5 Leitoras de cartões, 308
8.6 Códigos alfanuméricos, 310
8.7 Preparação de dados, 310
8.8 Reconhecimento de caracteres, 315
8.9 Equipamento de saída, 316
8.10 Impressoras, 317
8.11 Dispositivos de saída em tubos de raios catódicos, 320
8.12 Outros dispositivos de saída, 323
8.13 Códigos de correção de erro e de detecção de erro, 323
8.14 Terminais, 324
8.15 Dispositivos de entrada-saída de sistemas com componentes analógicos, 327
8.16 Conversores analógico/digital — Codificadores de posição angular, 328
8.17 Conversores digital/analógico, 329
8.18 Conversores eletrônicos analógico/digitais de alta velocidade, 331
Exercícios, 334

9 **A UNIDADE DE CONTROLE,** 339
9.1 Construção da palavra de instrução, 340
9.2 Organização do ciclo de instrução e do ciclo de execução dos registros de controle, 342
9.3 Seqüência de operação dos registros de controle, 347
9.4 Controle das operações aritméticas, 347
9.5 Seqüência típica das operações, 355
9.6 Instruções de salto, de desvio ou de transferência, 356
9.7 Instruções de deslocamento, 358
9.8 Microprogramação, 362
9.9 Variações em configurações de microprogramação, 364
Exercícios, 368

10 **ORGANIZAÇÃO DE COMPUTADORES,** 371
10.1 Formatos da palavra de instrução — número de endereços, 372
10.2 Representação de instruções e dados, 374
10.3 Técnicas de endereçamento, 375
10.4 Registros-índice, 376
10.5 Paginação e endereçamento indireto, 377
10.6 Interligação de componentes do sistema, 380
10.7 Dispositivos de interface — barramento ou vias, 387
10.8 Microprocessador e microcomputadores, 393
10.9 Organização de computador de endereço único, 404
10.10 Um microprocessador de endereço único, 409
10.11 Computadores de dois endereços, 422
10.12 Série de computadores IBM 360/370, 427
Exercícios, 432

BIBLIOGRAFIA, 438

RESPOSTAS AOS EXERCÍCIOS SELECIONADOS COM NÚMEROS ÍMPARES, 444

ÍNDICE, 469

Fundamentos de Computadores Digitais

1

OPERAÇÃO DO COMPUTADOR

A indústria de computadores continua a ser a importante indústria de crescimento mais rápido. As vendas de computadores de grande porte aumentam a cada ano, e o tamanho da indústria global de computadores foi, além disso, suplementado pelo mercado em rápido desenvolvimento dos minicomputadores e pela introdução de microcomputadores no mercado.

Os minicomputadores e os microcomputadores tornam possível estimular novas áreas de aplicação de computadores devido ao seu baixo custo, confiabilidade, reduzido tamanho e pouco peso. Estes pequenos computadores são capazes de calcular numa velocidade de centenas de milhares a milhões de operações por segundo e oferecem potência de computação que era encontrada em computadores de grande porte há 10 anos. Os lugares em que os pequenos computadores podem ser usados parecem intermináveis: áreas tais como a de controle de processos; monitoração médica; testes de produção; registros de instrumentos científicos; sistemas de verificação de armazenamento; e sistemas de avaliação e teste de automóveis estavam entre os primeiros a aparecer.

Mais de um milhão de pessoas trabalham atualmente na indústria de computadores, e isto não inclui os milhões que utilizam indiretamente os computadores, tais como funcionários de bancos que colocam todas as suas transações nos computadores, os empregados de motéis e de linhas aéreas que se valem dos computadores para fazer reservas e os mecânicos que usam máquinas-ferramentas controladas por computador.

De fato, os computadores agora encaminham nossas chamadas telefônicas a longa distância, processam e emitem os cheques em nossos bancos, tabelam nossos aviões e trens, fazem nossas previsões de tempo, predizem e processam nossas eleições, e figuram em tantas coisas que livros inteiros serão (e foram) escritos documentando apenas os tipos de aplicações.

Os computadores atualmente usam a importante participação dos componentes eletrônicos que estão sendo fabricados, e esta participação continuará a aumentar. A necessidade de pessoal especializado em computadores continua premente em todas as áreas: cerca de um quarto de milhão de novos programadores são requisitados a cada ano; o Ministério do Trabalho do governo federal continua a manter pessoal de serviço de máquinas comerciais e pessoal de operação de computadores eletrônicos em primeiro e segundo lugares numa lista de cinco "áreas de emprego de crescimento mais rápido".

1.1 CALCULADORAS E COMPUTADORES

A partir da época que os indivíduos começaram a usar aritmética, eles inventaram dispositivos para auxiliar na manipulação dos números. Um dos mais primitivos e mais engenhosos exemplos de auxílio à computação é o ábaco. Este primitivo antecessor (4000 a 3000 a.C.) dos modernos computadores consiste de uma armação retangular que transporta inúmeros fios paralelos. Cada fio sustenta várias contas que podem deslizar ao longo do comprimento do fio. Estas contas foram denominadas *calculi* (plural de *calculus*) pelos romanos, o que significava seixo. Esta raiz latina deu origem à palavra "calcular". Manipulando as contas, um operador habilidoso pode somar, subtrair, multiplicar e dividir com espantosa velocidade. Numa competição entre um proponente japonês desta antiga invenção e o operador treinado de uma máquina de calcular operada manualmente, o ábaco venceu com facili-

dade. A Ciência dirá a última palavra, no entanto, para os computadores descritos neste livro efetuarem operações vários milhões de vezes mais rápido que o melhor dos peritos em ábaco.

As máquinas calculadoras, inclusive os dispositivos familiares tais como as máquinas de somar, as calculadoras de mesa e as caixas registradoras, foram inventadas mais recentemente. A primeira calculadora mecânica bem-sucedida foi construída por Blaise Pascal no século dezessete. Ela foi concluída em 1642, e ainda existem modelos autênticos. A concepção primária que esta máquina introduziu foi a mecanização do transporte. A máquina consiste de uma série de rodas numeradas, ou diais, cada uma numerada de 0 a 9 e disposta para ser lida da esquerda para a direita. Quando um dos diais passa de 9 a 0, uma catraca faz com que a roda à sua esquerda seja deslocada uma unidade para a frente. A máquina soma e subtrai diretamente, mas a multiplicação e a divisão podem ser efetuadas por adições e subtrações repetidas.

Em 1671, Leibniz construiu uma máquina calculadora que não somente podia somar e subtrair mas também multiplicar. A adição e a subtração eram efetuadas de maneira idêntica à da máquina de Pascal, mas engrenagens adicionais foram incluídas a fim de permitir que a máquina multiplicasse diretamente.

Leibniz comentou que, embora o dispositivo não fosse inteiramente automático, o leve esforço envolvido na sua utilização era certamente preferível aos tediosos e muitas vezes errôneos processos requeridos na aritmética feita a mão. A calculadora projetada por Leibniz constituiu o modelo para a maioria das máquinas que surgiu mais tarde e incorporava quase todos os princípios atualmente usados no projeto de calculadoras.

À medida que a ciência e as técnicas comerciais progrediram, aumentou a necessidade da mecanização das operações aritméticas simples. Os cálculos aritméticos feitos pelas empresas comerciais aumentaram incrivelmente em volume e em complexidade, e os guarda-livros surgiram em maior número, assim como todos os tipos de pessoas que trabalham em escritórios. O trabalho de papel necessário para gerenciar eficientemente as funções de governo também alcançou proporções desconcertantes. Por exemplo, para processar os dados colhidos durante o censo do governo dos Estados Unidos, a demora foi tão grande que a informação não foi encontrada em forma resumida conveniente até depois de ter perdido sua validade. Os problemas do "Bureau do Censo" são tão críticos que ele se tornou o criador do cartão perfurado, utilizando-o em suas tabulações nos primórdios do ano de 1890.†

Nos campos da ciência e da engenharia, muitos problemas foram reduzidos a expressões matemáticas tão complexas que levaria um tempo enorme para efetuar os cálculos aritméticos necessários para avaliá-las com diversos conjuntos de parâmetros. Por exemplo, quando as fórmulas matemáticas que descrevem a rápida passagem de uma granada de artilharia ao longo de uma trajetória balística foram conhecidas pela primeira vez com grande exatidão, a solução destas fórmulas (envolvendo muitos ângulos de elevação do canhão diferentes, distâncias etc.) requeria os serviços de uma grande equipe de calculadores humanos durante longos períodos de tempo. Além disso, as tabelas tinham que ser recalculadas para novos tipos de granadas e armas. Isto freqüentemente introduzia um atraso a partir do instante em que as armas se tornavam disponíveis até que elas pudessem ser utilizadas com exatidão.

Felizmente, o calculador também foi progredindo. Os aperfeiçoamentos no projeto e na construção aumentaram a velocidade e o número e a variedade de operações que podiam ser executados por estes dispositivos, e ao mesmo tempo aprimorou as técnicas de fabricação que os tornou mais prontamente disponíveis. Nos primeiros modelos, as manivelas foram substituídas por motores elétricos, o que permitiu aumentar-lhes a velocidade de cálculo. Numerosos auxílios ao operador foram também introduzidos, aumentando assim a velocidade da computação e diminuindo as probabilidades de erro. Vários tipos de máquinas calculadoras de mesa eram comuns, e a maior parte da aritmética comercial era efetuada por estes dispositivos.

A etapa final no desenvolvimento da calculadora surgiu com a introdução das calculadoras eletrônicas. A velocidade destes dispositivos é eletrônica — e não mecânica — e, nos modelos maiores, há mesmo pequenas memórias eletrônicas ou, às vezes, tiras de fita ou cartões metálicos que podem armazenar pequenos "programas" contendo instruções para a calculadora. A calculadora pode então executar estes pequenos programas com velocidades eletrônicas à vontade do operador.

---

†Herman Hollerith, que desenvolveu o cartão perfurado para o Bureau do Censo, formou mais tarde sua própria companhia, que posteriormente foi incorporada à IBM.

O fato é que os dados (geralmente números) operados devem ser inseridos pelo elemento humano que opera a calculadora. As funções notáveis proporcionadas pelas calculadoras eletrônicas auxiliam grandemente, mas não eliminam a necessidade dos computadores.

Por exemplo, muitos computadores operam com grandes arquivos de dados que são armazenados nos dispositivos de memória do computador, tais como discos ou fitas magnéticos. Estes dados ocupariam muitos gabinetes de arquivo e consumiriam dos operadores humanos longos períodos de tempo para achar itens nestes arquivos. O computador, no entanto, pode pesquisar estes arquivos e achar e atualizar registros automaticamente. Por exemplo, os maiores computadores da rede bancária contêm milhões de registros referentes às contas bancárias dos clientes. Todas as transações feitas nas agências de um grande banco devem ser registradas, e os registros apropriados devem ser atualizados no fim de cada dia. O computador pode efetuar isto automaticamente, e as empresas bancárias e de seguros são grandes usuários de computadores assim como são as empresas comerciais que usam computadores para manter seus inventários, tais como as grandes cadeias de supermercados que utilizam os computadores para pedir ou encomendar automaticamente, para escriturar suas operações etc.

Os computadores são usados para controlar dispositivos de produção nas indústrias, para monitorar a operação de processos industriais e para auxiliar a estabelecer o tipo nos jornais e nos livros; eles executam automaticamente muitas funções e freqüentemente com velocidades e com exatidão e precisão impossíveis para os elementos humanos.

Nas aplicações científicas, alguns cálculos são extremamente complicados e longos, requerendo centenas de milhões de operações. A velocidade e a ausência de erros nos cálculos do computador tornam os computadores ideais para a execução destas mesmas operações. Muitas vezes, os cálculos científicos envolvem grandes quantidades de dados de teste (por exemplo, de satélites, de experiências nucleares ou de laboratórios médicos). Estes dados devem ser analisados com emprego de técnicas matemáticas que requerem muitos cálculos. Enquanto tais cálculos puderem ser executados em calculadoras eletrônicas, o esforço humano e o tempo requeridos devem tornar proibitivo o custo de tais cálculos.

1.2 DIFERENÇAS ENTRE CALCULADORAS DE MESA E COMPUTADORES

A diferença básica entre o computador digital automático e a calculadora eletrônica e eletromecânica é que o computador efetua longas seqüências de computações sem intervenção humana e geralmente tem uma memória de alguma capacidade. Além disso, os dados de entrada são, em geral, lidos automaticamente. O computador também pode tomar certas decisões lógicas e alterar suas atividades futuras em conseqüência destas decisões, mudando, às vezes, a maneira pela qual ele age através dos dados de entrada.

A seqüência de instruções que diz ao computador como resolver um determinado problema é chamada *programa*. O programa diz à máquina o que fazer, etapa por etapa, incluindo todas as decisões que devem ser tomadas. Disto se conclui que computador não planeja por si mesmo, mas que todo o planejamento deve ser feito antecipadamente. O crescimento da indústria de computadores criou a necessidade de pessoal treinado que nada faz além de preparar os programas, ou seqüências de instruções, que orientam o computador. A preparação da lista de instruções para o computador é chamada *programação,* e as pessoas que executam ésta função recebem o nome de *programadores.*

1.3 COMPUTADORES ELETRÔNICOS DIGITAIS

A história das tentativas para fabricar máquinas que executassem automaticamente seqüências extensas de cálculos é positivamente muito longa. A primeira tentativa mais bem conhecida foi feita no século XIX por Charles Babbage, um cientista e matemático inglês. Babbage tentou mecanizar seqüências de cálculos, eliminando o operador e projetando uma máquina de modo que ela executasse todas as operações necessárias numa seqüência predeterminada. A máquina projetada por Babbage usava cartões de cartolina, tendo buracos perfurados para introduzir tanto instruções quanto os dados necessários (números) na máquina. A máquina devia executar automaticamente as instruções ditadas pelos cartões, não parando até que uma seqüência inteira de instruções fosse concluída. Os cartões perfurados usados para controlar a máquina já tinham sido empregados para controlar a operação de máquinas de tecelagem. Bastante surpreendentemente, Babbage obteve algum dinheiro para o seu projeto graças ao governo inglês e iniciou a construção. Embora ele estivesse severamente limitado pela tecnologia de sua época e a máquina nunca fosse concluída, Babbage conseguiu estabelecer os princípios básicos sobre os quais são construídos os computadores modernos. Há mesmo alguma especulação de que se ele não tivesse

pouco dinheiro, ele poderia ter construído uma máquina bem-sucedida. Embora Babbage tivesse morrido sem realizar seu sonho, ele estabeleceu os conceitos fundamentais que foram usados para construir máquinas elaboradas mesmo além de suas expectativas.

Em 1930, os cartões perfurados foram muito usados nas grandes empresas comerciais, e vários tipos de máquinas de manipulação de cartões perfurados eram encontrados. Em 1937, Howard Aiken, em Harvard, propôs à IBM que podia ser construída uma máquina (usando algumas das partes e técnicas das máquinas de cartões perfurados) que colocasse automaticamente em seqüência as operações e os cálculos executados. Esta máquina usava uma combinação de dispositivos eletromecânicos, incluindo muitos relés. A máquina esteve em operação durante algum tempo, gerando muitas tabelas de funções matemáticas (particularmente as funções de Bessel), e foi usada nos cálculos de trajetórias na Segunda Guerra Mundial.

A máquina de Aiken foi notável em sua época, mas limitada na velocidade por seu uso de relés em vez de dispositivos eletrônicos e também pelo seu uso de cartões perfurados para colocar em seqüência as operações. Em 1943, S. P. Eckert e J. W. Mauchly, da Escola Moore de Engenharia da Universidade de Pensilvânia, iniciou o Eniac, que usava componentes eletrônicos (principalmente válvulas a vácuo) e era, portanto, mais rápido, mas que também usava chaves e vários fios em soquete para implementar a programação das operações. Mais tarde, Eckert e Mauchly construíram o Edvac, que tinha seu programa armazenado na memória do computador, não dependendo de colocação em seqüência externa. Esta foi uma importante inovação, e um computador que armazena internamente sua lista de operações, ou programa, é chamado *computador de programa armazenado*. Realmente, o Edsac, na Universidade de Manchester, iniciado mais tarde, porém concluído antes do Edvac, foi o primeiro computador operacional de programa armazenado.

Um ano depois, John Von Neumann, no Instituto de Estudos Avançados em Princeton, iniciou o IAS juntamente com a Escola Moore de Engenharia, e esta máquina incorporou a maioria dos conceitos gerais dos computadores paralelos de programa armazenado.

O Univac I foi o primeiro computador eletrônico digital disponível comercialmente e foi projetado por Eckert e Mauchly na companhia em que eles trabalhavam, que foi depois comprada pela Sperry Rand. O Bureau do Censo dos Estados Unidos comprou o primeiro Univac. (Mais tarde, o Univac e a metade da máquina de Aiken foram colocados na Instituição Smithsonian, onde eles podem ser vistos atualmente.) A IBM entrou na competição com o IBM 701, uma máquina grande, em 1953, e, em 1954, com o IBM 650, uma máquina muito menor que foi muito bem sucedida. O IBM 701 foi o precursor da série de máquinas IBM 704-709-7094, as primeiras "grandes vencedoras" na categoria de máquinas grandes.

Havia positivamente uns poucos computadores eletrônicos com válvulas a vácuo disponíveis e em uso no final de 1950, mas, nesta época, surgiu uma importante inovação na Eletrônica — o transistor. A substituição das grandes válvulas a vácuo dispendiosas (e quentes) pelos transistores pequenos, baratos, confiáveis e de dissipação de calor comparativamente baixa, conduziu aos chamados "computadores de segunda geração"; e o tamanho e a importância da indústria de computadores cresceu em ritmos assombrosos, enquanto os custos dos computadores individuais caíam substancialmente.

Em 1965, foi introduzida uma terceira geração de computadores. (A IBM Corporation, ao introduzir sua série 360, usou a expressão "terceira geração" como uma frase-chave em seus anúncios, e ela permanece como uma expressão vinculada à descrição de todas as máquinas desta era.) As máquinas deste período começaram a fazer uso intensivo de *circuitos integrados*, nos quais muitos transistores e outros componentes são fabricados e compactados juntos num único recipiente de pequenas dimensões. Os preços baixos e as altas densidades de integração destes circuitos mais as lições aprendidas nas máquinas anteriores conduziram a certas diferenças no projeto de sistema de computadores, e estas máquinas proliferaram e expandiram a indústria de computadores a sua atual dimensão de multibilhões de dólares.

As máquinas de quarta geração costumam ser menos facilmente distintas das primeiras gerações. No entanto, há algumas notáveis e importantes diferenças. A fabricação de circuitos integrados tornou-se tão avançada que permite incorporar milhares de componentes ativos em volumes de uma fração de polegada, levando ao que se chama integração em média escala (MSI) e integração em larga escala (LSI). Isto levou aos computadores de pequeno porte, baixo custo, grande memória e ultra-rápidos. Os grandes computadores têm-se tornado cada vez mais complexos; os computadores de porte médio trabalham atualmente como os computadores grandes do passado próximo trabalhavam; e há uma nova casta de computadores chamada microcomputadores e minicomputadores que são pequenos e baratos, são fabricados por muitas companhias diferentes e estão proliferando numa velocidade surpreendente.

1.4 APLICAÇÃO DOS COMPUTADORES AOS PROBLEMAS

Grande número de pessoal de administração tem sido empregado durante muitos anos nos departamentos de contabilidade de firmas comerciais. Os funcionários empregados por estas empresas gastam a maior parte de seu tempo efetuando computações aritméticas e introduzindo a seguir seus resultados em livros da companhia e nos cheques de pagamento, faturas, formulários de pedidos etc. A maioria dos cálculos aritméticos consiste de seqüências repetitivas de cálculos simples que os funcionários efetuam vezes sem conta sobre diferentes conjuntos de algarismos. Poucas decisões são necessárias, tendo as regras sido usualmente definidas de modo a cobrir quase todos os problemas que possam surgir.

Uma tarefa típica num escritório de folhas de pagamento é o processamento de cheques de pagamento dos empregados da companhia que trabalham por salário-hora.† Este serviço envolve o cálculo do salário total multiplicando-se cada salário-hora do empregado pelo número de horas que ele trabalhou, levando em consideração qualquer extraordinário, calculando e a seguir deduzindo impostos, seguro, contribuições de caridade etc.; emitindo a seguir o cheque necessário e introduzindo num registro de todos os algarismos. A Fig. 1.1 representa um fluxograma de um possível procedimento. Fluxogramas, tais como estes, constituem ferramentas padronizadas de comércio e, muitas vezes, são usadas pela indústria de computação. Tais fluxogramas são muito úteis quando na redução de problemas às etapas necessárias e representam um auxílio inestimável no campo da programação. O exemplo dado deliberadamente omite os salários extraordinários, impostos irregulares tais como FICA e outros aspectos complicados. O procedimento seguido por um funcionário na execução desta seqüência de computações pode ser como segue:

1. O funcionário verifica o registro de trabalho diário do empregado e adiciona o número de horas de trabalho por dia, obtendo o número total de horas trabalhadas durante a semana.
2. O número total de horas trabalhadas é multiplicado pela taxa de pagamento, e o salário total da semana é obtido.

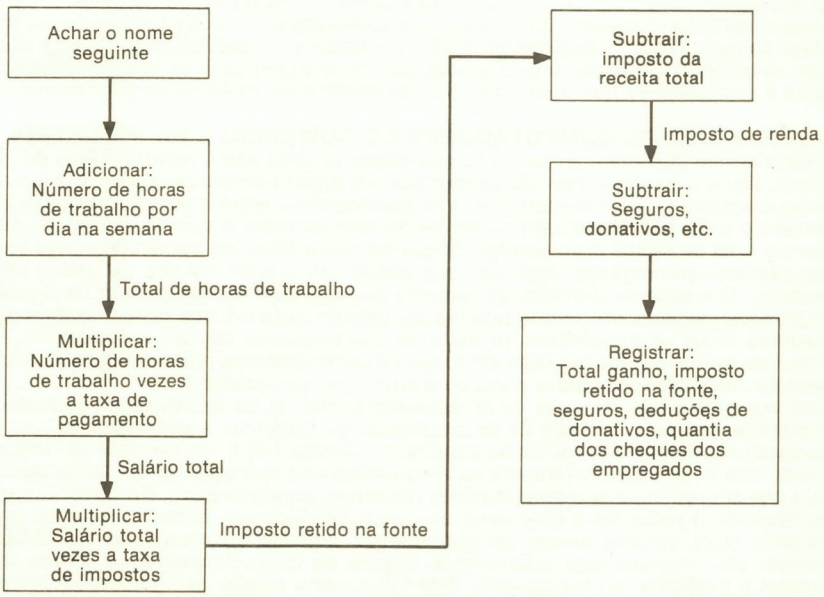


Fig. 1.1 Fluxograma de cálculo de cheque de pagamento.

†É interessante notar que 95% dos cheques emitidos pelo Governo Federal são redigidos por computadores.

3. O salário total é multiplicado pela taxa de imposto do empregado, e é achado o valor do imposto retido na fonte.
4. O imposto retido na fonte é subtraído do salário total.
5. Quaisquer deduções regulares, tais como seguros, são subtraídas.
6. Um registro de cada uma das operações acima é introduzido nos livros da companhia, e é emitido um cheque na quantia correta.

Pode ser visto que quase tudo do procedimento acima tem possibilidade de ser mecanizado por uma máquina que permite somar, multiplicar e subtrair na seqüência correta. A máquina deve possuir também as seguintes características menos óbvias:

1. A capacidade de lembrar os resultados intermediários que foram obtidos. Por exemplo, o salário total ganho deve ser lembrado enquanto o imposto estiver sendo calculado. É também conveniente manter na máquina a taxa de pagamento do empregado, a taxa de imposto retido na fonte, as taxas de seguro e a quantia regularmente dada às instituições de caridade.
2. A capacidade de aceitar informações. Os registros de tempo de trabalho, mudanças nas taxas de pagamento, quantias de dedução etc. devem ser introduzidos na máquina.
3. A capacidade de imprimir os resultados obtidos.

A ampla aceitação das máquinas de computação digital nos escritórios de folhas de pagamento é principalmente devida ao tipo repetitivo de trabalho que normalmente é realizado em tais escritórios. A mecanização de tais serviços é direta, embora muitas vezes complicada; mas a precisão e a velocidade inerentes, bem como os custos operacionais mais baixos, possibilitados pelas máquinas eletrônicas comerciais, justificam o uso especialmente popular neste campo. A maior parte das grandes empresas utiliza relógios de ponto que registram a hora em que cada empregado chega e sai, perfurando cartões de ponto. Estes cartões de ponto são "lidos" num computador que interpreta as perfurações e a seguir efetua automaticamente todas as operações aritméticas necessárias e imprime um registro de todos os totais, subtotais etc. para os livros da companhia. O computador também mantém um registro da quantia total paga por semana e por ano ao empregado, bem como um registro das diversas deduções, tais como o imposto retido na fonte. O computador pode determinar o número total de horas trabalhadas por todos os empregados durante qualquer semana dada, o pagamento médio de todos os trabalhadores durante qualquer período selecionado, o número de pessoas que contribuem para as várias instituições de caridade e praticamente qualquer outro tipo de informação da folha de pagamento.

1.5 APLICAÇÕES DE COMPUTADORES NO COMÉRCIO E NA INDÚSTRIA

Pode ser visto imediatamente que o uso dos computadores não é restrito à área do cálculo científico. De fato, as máquinas são produzidas em maior número para os negócios do que para uso científico ou de laboratório. A principal diferença entre o uso de máquinas digitais no comércio e no trabalho científico reside na relação entre o número de operações efetuadas e o total de dados processados. Enquanto que a máquina comercial efetua somente alguns cálculos empregando cada um dos dados, um grande volume de dados deve ser processado. O problema científico geralmente começa com menor número de dados, mas são efetuados cálculos em grande quantidade usando cada um dos dados. Ambos os tipos de máquina ainda se enquadram no título de computadores digitais automáticos, e qualquer tipo de trabalho pode ser feito em todos os computadores, embora algumas máquinas possam ser mais bem adaptadas a um ou a outro tipo de problema.

Um dos primeiros empregos do computador digital foi na mecanização da maioria das rotinas e dos aspectos clericais da administração ou gerência. A descrição do uso de um computador no cálculo das folhas de pagamento (Seção 1.4) é um exemplo de uma aplicação comercial e ilustra a similaridade na programação da operação de um computador e no cálculo dos procedimentos administrativos referentes ao empregado. Primeiro, o problema a ser resolvido é reduzido a uma série de operações simples: achar o nome do próximo empregado cujos salários devem ser computados, calcular quantas horas ele trabalhou e multiplicar este número pelo salário-hora. Depois do procedimento a ser usado ter sido executado e explicado ao funcionário, este funcionário recebe as informações numéricas necessárias, tais como taxas de pagamento, de seguro etc. Se as operações forem mais simplificadas, cada etapa na Fig. 1.1 poderá ser executada por um funcionário diferente. Por exemplo, o primeiro funcionário pode achar o registro do empregado e mandá-lo ao segundo funcionário, que computa o número total de horas trabalhadas e apresenta isto ao funcionário seguinte, que multiplica pela taxa de salário; e assim sucessivamente, até que

todas as operações tenham sido executadas. De tudo isto se conclui que a divisão dos procedimentos comerciais em etapas básicas constitui deveras uma prática muito antiga.

O procedimento para a preparação de uma lista de instruções para um computador digital é basicamente o mesmo. Todas as operações que o computador deve executar são escritas em forma de fluxograma (Fig. 1.1). O problema é então dividido numa lista de instruções para a máquina que especificam exatamente como deve ser obtida a solução. Depois do problema ter sido programado, a lista de instruções é transferida para algum meio que a máquina possa "ler" (cartões perfurados, por exemplo) e é a seguir lida na máquina. A máquina efetua automaticamente as etapas requeridas. Observemos que uma vez que o procedimento tenha sido estabelecido e as etapas programadas tenham sido lidas, a programação está terminada até que seja desejada uma mudança no procedimento. As mudanças nas taxas, por exemplo, podem ser inseridas simplesmente lendo as novas taxas de pagamento na máquina. Isto não afeta o procedimento.

1.6 APLICAÇÕES CIENTÍFICAS

A ciência e a engenharia modernas usam a matemática como uma linguagem para expressar as leis físicas em termos precisos. O computador eletrônico digital é uma valiosa ferramenta para estudar as conseqüências destas leis. Muitas vezes, o procedimento exato para resolver um problema foi encontrado, mas o tempo empregado para efetuar manualmente os cálculos necessários é exagerado. Às vezes é preciso resolver o mesmo problema repetidamente, usando diferentes conjuntos de parâmetros, e o computador é especialmente útil na solução dos problemas deste tipo. O computador não é apenas capaz de calcular tipos de expressões matemáticas em altas velocidades, mas, se um conjunto de cálculos for efetuado repetidamente nos diferentes conjuntos de valores numéricos, o computador será capaz de comparar resultados e determinar os valores ótimos que foram empregados.

Uma fórmula algébrica é uma expressão de uma relação matemática. Muitas das leis da

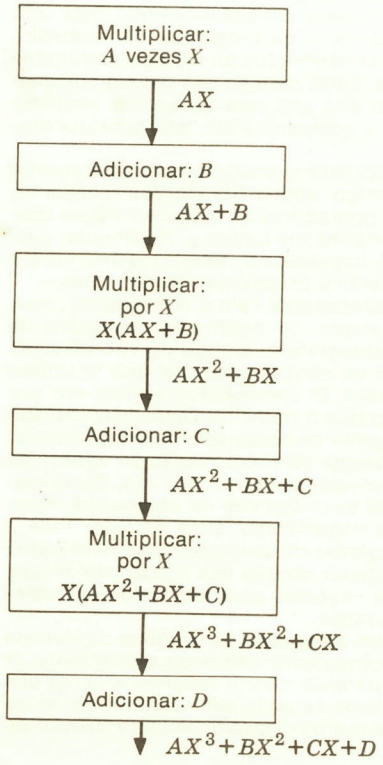


Fig. 1.2 Fluxograma da avaliação de expressão.

Física, da Eletrônica, da Química etc. são expressas nesta fórmula, em cujos casos os computadores digitais podem ser facilmente usados, porque as fórmulas algébricas podem ser diretamente mudadas para as etapas básicas que elas representam. A Fig. 1.2 representa um fluxograma que ilustra as etapas necessárias para calcular a expressão $ax^3 + bx^2 + cx + d$, dados valores numéricos para a, b, c, d e x. As etapas requeridas são como segue:

1. Multiplicar a por x, obtendo ax.
2. Adicionar b, obtendo $ax + b$.
3. Multiplicar este resultado por x, formando $ax^2 + bx$.
4. Adicionar c, obtendo $ax^2 + bx + c$.
5. Multiplicar este resultado por x; $x(ax^2 + bx + c)$ ou $ax^3 + bx^2 + cx$.
6. Adicionar d, obtendo $ax^3 + bx^2 + cx + d$.

Usando calculadoras operadas manualmente, seriam necessários vários minutos para efetuar os cálculos destinados a avaliar esta expressão algébrica para um só conjunto de valores, mas praticamente qualquer computador poderia efetuar esta série de operações vários milhares de vezes por segundo. Enquanto que a expressão algébrica mostrada é certamente muito mais simples do que muitas fórmulas encontradas por profissionais dos ramos científico e de engenharia, percebe-se imediatamente o valor de usar um computador em certos tipos de problemas.

1.7 ALGUNS TIPOS DIFERENTES DE SISTEMAS DE COMPUTADORES

O uso mais comum dos computadores é nos programas de operação perfurados em cartões (ou talvez registrados em fita de papel ou em fita magnética) e passados num computador que então prepara os impressos, cheques ou alguma forma de apresentação de dados registrando os resultados. Este é um exemplo do que se denomina *processamento em lotes*. Quando um computador é usado deste modo, os dados de entrada (e, muitas vezes, o programa) são introduzidos no computador e processados automaticamente, em geral sem a intervenção do operador. Freqüentemente, muitos serviços (ou conjuntos de dados) diferentes são processados, um logo após o outro, ou mesmo simultaneamente, mas sem qualquer interação do usuário do sistema durante a operação do programa. Por exemplo, os operadores de perfuradoras podem perfurar muitos condensados de cartões perfurados que contêm dados sobre clientes e outras informações. Estes cartões são depois empilhados juntos e transportados para um grande computador que processa os cartões, emitindo cheques, remetendo contas, imprimindo registros para a companhia etc. Isto constitui processamento em lotes.

Similarmente, o usuário de um computador num laboratório científico pode apresentar cartões de programa e cartões de dados como um *serviço*, com a fita elástica familiar ao redor destes junto com quaisquer observações para os operadores da máquina. Vários destes condensados de cartões são coletados e depois empilhados juntos e, finalmente, passados na máquina. Posteriormente, os resultados são impressos e, talvez mesmo no dia seguinte, fornecidos aos usuários individuais. Isso também é processamento em lotes.

Noutros tipos de sistemas, os usuários interagem diretamente com o computador, inserindo e recebendo dados conforme desejado. Por exemplo, um agente de passagens de linhas aéreas deseja fazer uma reserva. Este agente datilografa o número do vôo de avião desejado e a identidade do passageiro numa máquina de escrever especial que se comunica, através de linhas telefônicas, com um computador. O computador verifica em sua memória se o vôo está completo e, se não estiver, introduz o nome do passageiro em sua lista do vôo e depois comunica este fato de volta ao agente de passagens de linhas aéreas. Se não houver lugares disponíveis, o computador transmite esta informação ao agente de passagens, que tenta convencer o passageiro a se interessar por um outro vôo. Desta maneira, uma linha aérea põe em comunicação todos os seus agentes de passagens, mantendo um registro constante dos vôos, passageiros e pagamentos, além de fazer toda a escrituração. Os terminais onde estão localizados os agentes de passagens são espalhados pelo mundo inteiro, mas se comunicam com o computador através das linhas telefônicas. Os motéis, hotéis, corretores da Bolsa e muitos outros negócios possuem sistemas similares de reservas, transferência de informações e escrituração.

Todos estes são chamados sistemas de *interação em-linha*, pois os usuários do sistema se comunicam diretamente com o computador e o computador responde diretamente. O desenvolvimento destes sistemas tem progredido em paralelo com o desenvolvimento dos dispositivos de entrada por teclado, bem como dispositivos de saída para usuários de vários tipos, incluindo exposições de televisão, impressoras e outros dispositivos de exposição de dados.

Os sistemas em-linha são também amplamente usados em aplicações científicas onde os usuários operam seus programas num terminal conectado ao computador, talvez por linhas telefônicas, tentando mudanças e variações à vontade. Os experimentadores podem tentar um conjunto de entradas e estudar o resultado, depois tentar outras entradas e estudar estes resultados. A técnica da computação interativa em-linha é usada pelos projetistas de circuito, arquitetos e químicos em quase todas as áreas, incluindo os sistemas médicos para uso em hospitais.

Um dispositivo de entrada/saída amplamente usado é o *teletipo*, mostrado na Fig. 1.3. Este é um exemplo de um teclado "semelhante à máquina de escrever", que gera um registro impresso ao ser utilizado, mas que também gera sinais elétricos que podem ser usados como entrada do computador. Similarmente, os sinais eletrônicos provenientes de um computador podem ser usados para controlar o teletipo, e o teletipo datilografará, sob con-



Fig. 1.3 Teletipo Modelo 33 KSR. (Cortesia da AT&T.)

trole do computador, os resultados dos cálculos.

O teletipo é, muitas vezes, usado em sistemas nos quais o painel ou console (neste caso o teletipo) está a certa distância do computador. Um acessório especial chamado *modem* (modulador-demodulador) é usado para tornar possível a transmissão dos sinais elétricos gerados pelo teletipo ao computador e receber de volta a resposta do computador sobre linhas telefônicas. No computador está localizado um outro modem que pode também transmitir ou receber, e este par de modems permite comunicação bilateral sobre linhas telefônicas. O usuário do teletipo simplesmente disca o número em que o computador está localizado, estabelece uma ligação e a identidade do usuário e o direito ao uso do computador (o computador geralmente testa uma palavra de referência ou senha), e depois continua a usar o computador.

Uma vez que os teletipos são absolutamente bem padronizados, o mesmo console (painel) de teletipo pode usar vários computadores diferentes, quando disponíveis. Atualmente existem muitas companhias que fornecem serviços de computador aos usuários que possuem teletipos à sua disposição. Os usuários discam simplesmente para o computador que eles preferem.

O teletipo é apenas uma das muitas espécies de *terminais de computador* correntemente em uso. (Há muitos competidores para o teletipo da Fig. 1.3.) A Fig. 1.4 mostra um teclado idêntico a uma máquina de escrever de entrada de dados, mas as respostas do computador são transmitidas a um demonstrador de osciloscópio similar a uma televisão, fornecendo, assim, uma exposição temporária (em vez de um "impresso", que é a página impressa). (Às vezes, é fornecido algum meio para fazer uma cópia do que está sendo exigido, na conveniência do usuário.) Este tipo de dispositivo de saída é chamado *dispositivo de exposição visual* e permite que o computador desenhe imagens ou faça gráficos bem como use caracteres impressos.

O terminal de computador na Fig. 1.4 é portátil. Um acessório que prende um aparelho



Fig. 1.4 Terminal de computador portátil com indicador de tubo de raios catódicos. (Cortesia da Logitron, Inc.)

telefônico é fornecido, de modo que, quando o aparelho telefônico estiver colocado neste acessório, pode ser feita uma discagem para um computador, e, uma vez estabelecida a conexão, o terminal gera então tons de áudio no aparelho quando as teclas forem abaixadas no teclado. O receptor do terminal também decodifica os tons codificados que representam caracteres gerados pelo computador, apresentando a informação recebida no dispositivo de indicação visual do osciloscópio. Deste modo, o usuário de um terminal pode "conversar", ou se comunicar, com qualquer computador compatível conectado ao sistema telefônico.

Quando vários usuários compartilham de um computador, usando o computador, ao mesmo tempo, muitas vezes através de linhas telefônicas, este computador é dito ser de *tempo compartilhado.* Por esta expressão entende-se que o computador é capaz de alternar e de interdirecionar a passagem de seus programas de modo que vários serviços ou usuários possam estar funcionando ao mesmo tempo. Isso contribui para um uso mais eficiente, e os sistemas de reserva de linhas aéreas, de motéis e de hotéis e a maioria dos sistemas em-linha usam o tempo compartilhado. Retornaremos a estes sistemas na Seção 1.10.

1.8 O USO DE COMPUTADORES DIGITAIS NOS SISTEMAS DE CONTROLE

A capacidade dos computadores digitais em fazer cálculos precisos e tomar decisões em altas velocidades possibilitou usá-los como partes de sistemas de controle. O primeiro exemplo do emprego de um computador num sistema de controle de grande porte foi o Sistema de Defesa Aérea SAGE. Neste sistema, os dados originários de uma rede de estações de radar, que são usados para detectar as posições de todos os aviões na área, são introduzidos através de enlaces de comunicações num computador de alta velocidade. O computador armazena toda informação posicional que chega das estações de radar e, a partir daí, calcula as posições futuras do avião, sua velocidade e altitude, e todas as outras informações pertinentes. Vários outros tipos de informação também são transmitidos ao computador, inclusive informação oriunda de navios a pique, avião EW, observadores aéreos do Corpo de Observadores Terrestres *(Ground Observer Corps),* planos de vôo tanto para aviação militar quanto para a civil e informações sobre previsão do tempo.

Um só computador recebe toda esta informação e, a partir dela, calcula uma imagem composta da situação aérea completa. O computador então gera exibições sobre osciloscópios especiais que são usados por membros do alto comando militar para tomar decisões táticas. O computador ainda ajuda estes operadores calculando o uso mais eficiente do avião interceptador, dos canhões antiaéreos e também dos mísseis antiaéreos. Por meio de enlaces de rádio, o computador guia automaticamente tanto o avião interceptador quanto os mísseis a seus alvos.

É evidente que o computador neste sistema é verdadeiramente uma máquina de processamento de dados em alta velocidade. O computador recebe os dados de muitas fontes, processa estes dados e depois "controla" o poder de defesa aérea.

Um sistema desta espécie é chamado *sistema de controle em tempo real* porque a informação deve ser processada e as decisões devem ser tomadas em tempo real. Quando um computador é usado para processar dados comerciais ou para executar a maioria de cálculos científicos, o tempo não constitui um fator crítico. Nos sistemas de tempo real, o computador deve "conservar-se", processando todos os dados em altas velocidades a fim de ser eficiente.†

Outros exemplos de aplicações de controle em tempo real incluem as refinarias de petróleo e outras áreas de fabricação que usam o computador para controlar automaticamente os processos de fabricação. Os computadores digitais são também usados para guiar as máquinas-ferramentas que realizam operações de usinagem com precisão e automaticamente, conforme indicado na Fig. 1.5. Além disso, tanto os veículos espaciais tripulados como os não tripulados transportam computadores digitais que realizam as funções necessárias de orientação, enquanto que uma rede de computadores em terra monitora e orienta o progresso do vôo.

A maioria dos sistemas de controle em tempo real requer um dispositivo importante conhecido por *conversor analógico/digital.* Em muitos casos, as entradas nestes sistemas estão sob a forma de *grandezas analógicas ou análogas* tais como deslocamentos mecânicos (por exemplo, posições de eixo) ou temperaturas, tensões, pressões etc. Uma vez que o computador digital opera com dados digitais em vez de dados analógicos, surge um problema fundamental de "linguagem" que requer a conversão das grandezas ou quantidades

†A maioria dos sistemas de reserva seria considerada sistemas de tempo real por seus operadores, uma vez que os atrasos de quaisquer conseqüências seriam prejudiciais aos negócios.

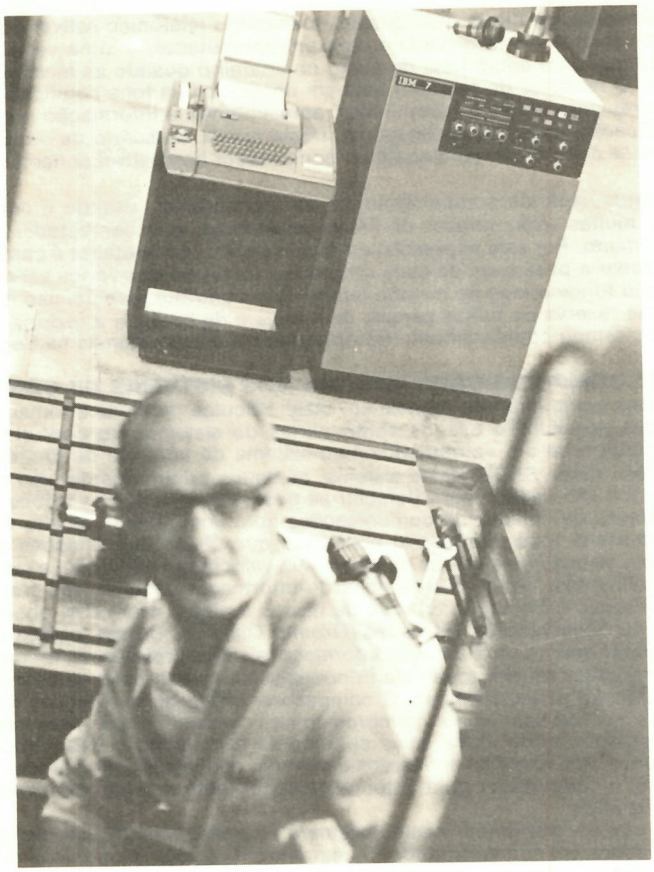


Fig. 1.5 Computador usado no sistema de controle industrial em-linha.

analógicas em representações digitais. Esta é a função do conversor analógico/digital.

O mesmo problema ocorre na saída do computador, onde é muitas vezes necessário converter os dados numéricos de saída provenientes do computador em deslocamentos mecânicos ou sinais elétricos do tipo analógico. Por exemplo, uma saída em "número" do computador pode ser usada para girar um eixo através do número de rotações indicado pelo número de saída. Um dispositivo que converte informação do tipo digital em grandezas analógicas é chamado *conversor digital/analógico*. No Cap. 8 será encontrada uma descrição dos dois tipos de conversores: ADC e DAC.

Os elementos básicos num sistema de controle que usa um computador digital consistem de (1) dispositivos de coleta de dados que executam medições no meio físico externo e, se necessário, também executam conversão analógica/digital nos dados do sistema que deve ser controlado; (2) o próprio computador digital, que efetua cálculos com os dados fornecidos e toma as necessárias decisões; (3) os meios de comunicação com, ou controlados por, certos elementos no meio físico externo. Se ninguém auxilia o computador em seus cálculos ou decisões, o sistema é considerado totalmente automático; se um elemento humano também entra no circuito de controle, o sistema é definido como semi-automático.

A Fig. 1.5 mostra um computador sendo usado numa aplicação fabril. Devido à sua alta velocidade, os computadores podem medir, provar, analisar e controlar as funções de fabricação quando elas ocorrem, muitas vezes em operação que não é assistida. Os computadores podem manejar o controle do piso das lojas, testar controle da qualidade, manipu-

lar materiais e monitorar produção. A estação idêntica a uma máquina de escrever à esquerda do computador na Fig. 1.5 é usada para estabelecer comunicação com o sistema.

1.9 COMPUTADORES COM PROPÓSITOS ESPECIAIS E PROPÓSITOS GERAIS

Há, em geral, dois tipos de computadores digitais. O primeiro é o *computador digital com propósitos especiais,* que executa uma seqüência fixa e preestabelecida de cálculos. Este tipo de computador pode ser construído mais eficientemente considerando-se que ele será mais leve e menor e consumirá menos potência etc. do que o computador de propósitos gerais. Por causa das vantagens da construção, os pequenos computadores de propósitos especiais são usados onde fatores tais como o peso, o consumo de potência etc. são críticos, como nos sistemas de controle de aviões, sistemas de controle de mísseis, equipamento especial para ensaio geral em dispositivos militares usados no campo etc.

O segundo tipo de computador é definido como um *computador digital de propósitos gerais.* A seqüência de instruções que a máquina segue é geralmente lida neste tipo de máquina e armazenada na memória da mesma. Podemos fazer com que a máquina siga uma outra seqüência de instruções simplesmente lendo no conjunto desejado de instruções. Uma vez que a seqüência de operações efetuadas pelo computador digital de propósitos gerais pode ser facilmente modificada, a máquina possui grande flexibilidade, e este é o tipo de máquina geralmente usado no comércio e nas computações científicas. O computador de propósitos gerais pode processar uma pilha de cartões de folha de pagamento e, depois, após ter sido lido nele um outro programa, pode executar um inventário de um estoque da companhia. Nas aplicações científicas, um computador de propósitos gerais pode calcular a órbita de um satélite e, então, após ter sido lido um novo programa, projetar um sistema de lentes para um projetor de cinema. O computador de propósitos gerais pode ser usado para resolver uma ampla variedade de problemas, cujos detalhes podem ter sido desconhecidos quando a máquina foi projetada. O computador de propósitos especiais é geralmente capaz apenas de resolver um tipo especial de problema.

1.10 SISTEMA DE COMPUTADOR EM TEMPO COMPARTILHADO

Dentre os mais importantes sistemas de computador em tempo real estão os projetados para auxiliar as operações de pesquisa, os negócios e as tomadas de decisão nos meios militares. Estes sistemas dotam os usuários de um grande arquivo de informações e de uma instalação computacional, pelo uso dos quais eles podem comunicar-se de maneira direta. Conforme foi mencionado, o usuário do sistema é capaz de introduzir instruções no computador utilizando dispositivos tais como um teclado de teletipo ou um conjunto de botões de pressão num painel ou console. O computador responde ou imprimindo os resultados ou exibindo-os num osciloscópio. A interação entre o usuário e o computador é direta; faz-se com que o computador responda imediatamente. Em geral, o tempo que decorre desde quando é dado um comando a um computador até que os resultados sejam obtidos é chamado *tempo de retorno.* O projetista de um sistema de tempo compartilhado em-linha tenta minimizar o tempo de retorno.

Estes sistemas são usados de várias maneiras. Os sistemas de projeto auxiliado pelo computador têm consoles e indicadores visuais tais que um projetista possa achar os detalhes de um projeto usando o computador para efetuar todos os cálculos e para exibir os efeitos das mudanças no projeto. O projetista então procura otimizar o projeto mudando os parâmetros e observando os efeitos destas mudanças no projeto.

O gerente comercial ou o chefe militar usa grandes sistemas digitais de maneira que confie fortemente no enorme número de registros que pode ser armazenado na memória de uma máquina digital. Os sistemas comerciais são chamados *sistemas de controle de administração,* e os sistemas militares são chamados *sistemas de controle e comando.* Os sistemas comerciais contêm os arquivos e registros dos negócios (comércio ou indústria), bem como quaisquer outros dados pertinentes, e os sistemas militares armazenam as informações militares requeridas pelos oficiais para tomar decisões críticas. Em qualquer caso, a informação é continuamente atualizada, usando as instalações de comunicações. Quando for solicitada uma categoria de dados, o computador a localiza em sua memória e fornece aspectos ou características selecionadas sobre o pedido. Por exemplo, o computador pode armazenar os tipos de aviões disponíveis em cada aeroporto no mundo, junto com o alcance e a capacidade de cada avião para transportar bombas. Se o avião fosse pilotado de um lugar a outro, ou se aviões estivessem no ar ou se perdessem, esta informação seria introduzida no computador. Se o pessoal da Força Aérea então quisesse projetar um ataque aéreo, pediria os dados ao computador e depois tentaria usar aviões que estivessem localizados favoravelmente e fossem adequados a essas missões. O computador pode também

fornecer dados referentes ao combustível e munição disponíveis em cada lugar, para facilitar ainda a tomada de decisões.

Similarmente, um executivo numa empresa comercial teria acesso a todos os registros de vendas da companhia, salários, níveis de estoque corrente, cálculo das despesas gerais e quaisquer outros dados que ele ou ela possam desejar calcular, usando os registros da companhia.

Nos sistemas acima, o computador está sujeito a ser *compartilhado em tempo,* uma vez que ele tem muitos canais de informação de entrada e saída, cada um com usuários diferentes, e estes usuários compartilham do computador principal.

1.11 COMPONENTES BÁSICOS DE UM COMPUTADOR DIGITAL

O diagrama em blocos na Fig. 1.6 ilustra as cinco principais divisões operacionais de um computador eletrônico digital. Embora presentemente as máquinas disponíveis variem enormemente nos detalhes de construção de vários componentes, os conceitos do sistema global permanecem sensivelmente os mesmos. É interessante notar que as máquinas nos dias de hoje são basicamente projetadas de acordo com os preceitos que Charles Babbage propôs em seu "engenho analítico". Embora a máquina de Babbage fosse de construção mecânica enquanto que os atuais computadores são eletrônicos, as funções que as várias seções das máquinas executam são exatamente as mesmas.

É provável que a importante diferença conceitual resida na capacidade de a memória do moderno computador armazenar tanto as instruções quanto os dados a serem usados no mesmo local. Isto permite uso mais eficiente da memória e também possibilita ao computador alterar as instruções que constituem o programa enquanto o computador estiver operando.

Um computador digital pode ser dividido nas seguintes unidades fundamentais:

1. *Entrada.* Os dispositivos de entrada lêem os dados necessários na máquina. Na maioria dos computadores de propósitos gerais, as instruções que constituem o programa devem ser lidas na máquina junto com todos os dados a serem usados nas computações. Alguns dos dispositivos de entrada mais comuns são as leitoras de cartão perfurado e de fita de papel perfurada, as leitoras de fita magnética e os vários dispositivos de entrada manual tais como chaves articuladas e botões de pressão.
2. *Controle.* A seção de controle de um computador põe em seqüência a operação do computador, controlando as atividades de todas as outras unidades. O conjunto de cir-

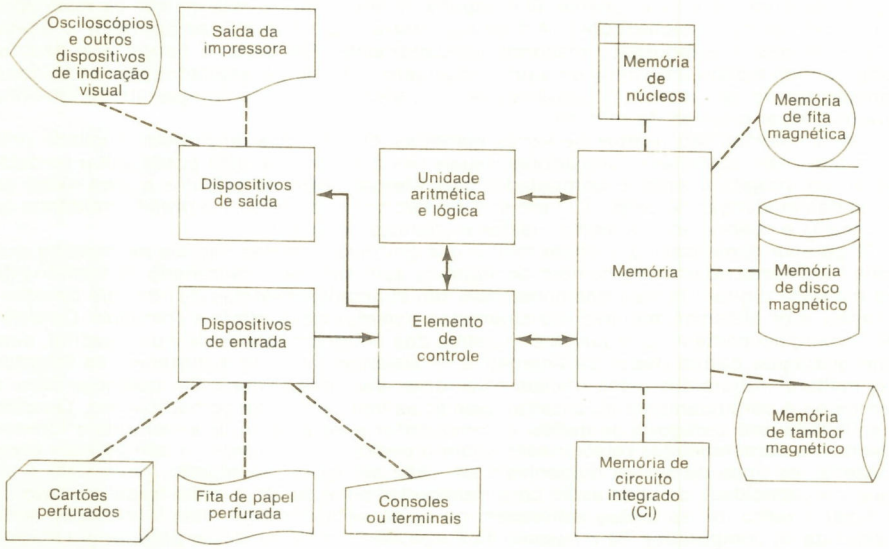


Fig. 1.6 Diagrama-bloco de um computador digital típico.

cuitos de controle interpreta as instruções que constituem o programa e então orienta o restante da máquina em sua operação.

3. *Memória.* A memória, ou seção de armazenamento do computador, consiste dos dispositivos usados para armazenar a informação que será usada durante as computações. A seção de memória do computador é também usada para reter tanto os resultados intermediários quanto os resultados finais enquanto o computador prossegue através do programa. Os dispositivos de memória principal ou de alta velocidade para um computador são divididos em peças de igual dimensão, cada uma das quais é então identificada com o que se chama *endereço* ou *local na memória.* Se a unidade de controle estiver procurando uma peça específica de informação ou uma instrução localizada na seção de memória, ela vai buscá-la por meio de seu endereço. Os dispositivos de memória são construídos de modo que seja possível por controle obter a informação contida em qualquer endereço. A duração de tempo requerida para obter informação pode variar um tanto, contudo, e é determinada pelo tipo de dispositivo usado para armazenar a informação. Os dispositivos de armazenamento comuns são os núcleos magnéticos, as memórias de circuito integrado, os tambores magnéticos, a fita magnética e os discos magnéticos.
4. *Unidade aritmética-lógica.* As unidades aritméticas-lógicas da maioria dos computadores são capazes de efetuar as operações de adição, subtração, divisão e multiplicação, bem como algumas "operações lógicas" que serão descritas. A unidade de controle diz à unidade aritmética-lógica qual destas operações deve efetuar e então vê se os números necessários foram fornecidos. O elemento (unidade) aritmético pode ser comparado às máquinas de calcular descritas anteriormente, em que os números a serem usados são inseridos e ele é então orientado para efetuar as operações necessárias.
5. *Saída.* Os dispositivos de saída são usados para registrar os resultados obtidos pelo computador e apresentá-los ao "mundo exterior." Os dispositivos de saída, em sua maioria, são dirigidos pelo elemento de controle, que também faz com que a informação necessária seja fornecida a eles. Os dispositivos comuns de saída são as máquinas perfuradoras de cartões, máquinas de fita magnética, máquinas de escrever eletromecânicas especiais, tubos de raios catódicos e dispositivos de impressão de alta velocidade. Há também muitos tipos incomuns de dispositivos de saída, tais como lâmpadas, campainhas elétricas (cigarras) e alto-falantes.

1.12 CONSTRUÇÃO DA MEMÓRIA

Foi mencionado que a memória interna ou de alta velocidade está dividida em vários endereços ou locais. Em cada um destes endereços, um grupo de dígitos é armazenado e manipulado pelo computador como uma unidade. Denomina-se geralmente "palavra"† o grupo de dígitos armazenado em cada endereço na memória. A cada endereço na memória é atribuído um número, e o endereço é então denominado por aquele número. Dizemos que o endereço 100 contém o valor 300, ou que o endereço 50 contém uma palavra de instrução. Na maioria dos computadores, as palavras de instrução podem ser armazenadas nos mesmos locais como palavras de números ou de dados, o que torna a memória mais flexível. Observe também que as palavras da instrução consistem de um grupo de dígitos quando armazenadas no computador.

O tempo que decorre para se obter uma palavra de um dispositivo de armazenamento é chamado *tempo de acesso.* O tempo de acesso dos dispositivos de armazenamento usados numa máquina tem um efeito marcante sobre a velocidade do computador. Um dos fatores que durante longo tempo impediram a construção dos computadores de alta velocidade foi a falta de dispositivos de armazenamento confiáveis com pequenos tempos de acesso. O desenvolvimento de dispositivos de armazenamento (tais como os núcleos magnéticos) com tempos de acesso muito curtos e mais a capacidade de armazenar informação por um tempo indefinido constituiu um grande passo à frente.

1.13 INSTRUÇÕES

Foi mencionado que os computadores podem tomar decisões lógicas simples. Muitas destas decisões se baseiam em números e, portanto, são decisões quantitativas e não qualitati-

†Há realmente duas maneiras de organizar uma memória agora em uso geral. Uma delas consiste em armazenar um caráter em cada endereço (um caráter é um 1 ou 2, um *A* ou *B* etc.). Estes sistemas são chamados *sistemas de caracteres endereçáveis.* A segunda maneira consiste em armazenar um *operando* completo ou palavra de instrução em cada endereço. Estes sistemas são chamados *sistemas de palavra endereçável.* Como pode ser imaginado, os sistemas de caracteres endereçáveis têm menores quantidades de informação em cada endereço do que os sistemas de palavra endereçável. Há, muito naturalmente, prós e contras para cada tipo de sistema. Simplificaremos a descrição considerando que cada endereço sucessivo contém uma palavra, como nos sistemas de palavra endereçável, mas descreveremos outros sistemas posteriormente.

vas. A espécie de decisão que o computador pode tomar é quanto a saber se um número é maior do que outro, ou se o resultado de alguma série de cálculos é positivo ou negativo. No entanto, a maioria das decisões tomadas por funcionários clericais e cientistas é também baseada em cálculos. Por exemplo, uma vez que os fenômenos físicos tenham sido expressos em fórmulas, as soluções para os problemas específicos são expressas por meio de números. Talvez seja relevante a seguinte observação feita por Lord Kelvin: "Quando você pode medir aquilo de que você está falando e exprime sua medida em números, você sabe alguma coisa sobre aquilo. Mas quando você não pode medir aquilo ou expressá-lo em números, seu conhecimento é de natureza escassa ou insatisfatória."

O computador digital não calcula sua própria solução aos problemas, porém deve ser dito exatamente como resolver qualquer problema dado e também como tomar todas as decisões. Conforme foi mencionado, a preparação de uma lista de instruções que diz ao computador como efetuar seus cálculos é chamada programação. O procedimento para a programação de um problema consiste geralmente de duas etapas separadas. A primeira etapa, o planejamento do programa, envolve a determinação da seqüência de operações requerida para resolver o problema. Ela consiste, em geral, na divisão do problema em diagramas de fluxo (fluxogramas) tais como os ilustrados nos tópicos precedentes. Uma vez que o problema tenha sido reduzido a esta forma, ele está pronto para ser *codificado;* esta é a segunda etapa. A *codificação* consiste em escrever as etapas resumidas no fluxograma numa linguagem especial que pode ser interpretada pela máquina. O programa final codificado consiste de uma lista de instruções para o computador, escritas num formato especial que detalha as operações que o computador deve executar.

As palavras de instrução que orientam o computador são armazenadas na máquina em forma numérica. No entanto, o programador raramente escreve instruções em forma numérica; em vez disto, cada instrução para o computador é escrita usando-se um código de letras para designar a operação a ser executada, além do endereço na memória do número a ser usado nesta etapa do cálculo.† Mais tarde, a seção alfabética da palavra de instrução é convertida em forma numérica por um programa de computador chamado *montador.* Este será descrito numa outra seção.

Uma palavra de instrução como é escrita pelo programador consiste de duas partes: (1) a parte do *código de operação* que designa a operação (adição, subtração, multiplicação etc.) a ser efetuada e (2) o *endereço* do número a ser usado. Uma palavra de instrução típica escrita pelo programador é

ADD 535

Esta palavra de instrução é dividida em duas partes principais: primeiro, a parte do código de operação, consistindo das letras ADD, que orienta o computador a executar a operação aritmética de adição; e segundo, a parte de endereço, que diz ao computador o endereço em armazenamento do número a ser usado.

É importante observar que a segunda seção da palavra de instrução dá somente o local (endereço) em armazenamento do número a ser adicionado. O número 535 na instrução mostrada não é o número que na realidade deva ser adicionado, mas apenas diz ao computador onde achar o número desejado. A que é somado o número no endereço 535? Ele deve ser somado ao número que já está na unidade aritmética-lógica num registro ou dispositivo de armazenamento chamado *acumulador.* Se o acumulador contiver zero antes de a instrução ser executada, o acumulador conterá o número que está armazenado no endereço 535 após a instrução ter sido executada. Se o acumulador contiver o número 500 antes de a instrução ser executada, e o número armazenado no endereço 535 for 200, o número armazenado no acumulador após a operação ADD (SOMAR) será 700. Para ilustrar este princípio com maior clareza, várias outras instruções são explicadas no Quadro 1.1.

Um pequeno programa que adiciona três números com emprego destas instruções é mostrado no Quadro 1.2.

O programa opera como segue: A seção de controle começa com a palavra de instrução no endereço 1, que limpa o acumulador e depois adiciona o número no endereço 6 neste acumulador. A instrução no endereço 2 adiciona o número no endereço 7 ao número

†As instruções descritas são típicas de um *computador de um só endereço.* Também são construídos computadores que usam dois, três e quatro endereços em cada palavra de instrução de computador. Estes computadores são chamados computadores de *múltiplos endereços* e serão descritos no Cap. 9. O tipo de instrução de um só endereço é muito direto e será usado nas ilustrações deste capítulo. O computador de dois endereços é igualmente popular e igualmente fácil de aprender e será discutido posteriormente no livro. O aprendizado da codificação de dois endereços, depois da codificação de um só endereço ter sido conhecida a fundo, é realmente muito simples.

QUADRO 1.1

Palavra de instrução		
Código de operação	Parte de endereço	Função executada pela instrução
CLA	430	O acumulador é esvaziado de todos os números anteriores, e o número no endereço 430 é acrescentado nele. Depois de a instrução ser executada, o acumulador contém o número em armazenamento no endereço 430. CLA é um código mnemônico para *clear and add* (limpar e acrescentar).
ADD	530	O número localizado no endereço 530 é acrescentado ao número no acumulador. Depois da instrução, o acumulador contém a soma do número que ele continha anteriormente com o número no endereço 530.
SUB	235	Subtrai o número localizado no endereço 235 na memória do número do acumulador e coloca a diferença no acumulador.
STO	433	O número no acumulador é armazenado no endereço 433. Qualquer informação previamente neste endereço é destruída. O número que esteve no acumulador antes de a instrução ser executada permanece no acumulador. Isto é geralmente designado por instrução de STORE (ARMAZENAMENTO).
HLT	000	A máquina recebe a ordem de parar. O número no acumulador permanece.

QUADRO 1.2

Endereço na memória	Palavra de instrução		Conteúdo de elemento aritmético após a instrução ser executada
	Código de operação	Parte de endereço	
1	CLA	6	200
2	ADD	7	500
3	ADD	8	900
4	STO	9	900
5	HLT	0	900
6 contém o número 200			
7 contém o número 300			
8 contém o número 400			

QUADRO 1.3

Palavra de instrução		
Código de operação	Parte de endereço	Função
MUL	400	O número no endereço 400 é multiplicado pelo número já no acumulador e pelo produto colocado no acumulador.

já no acumulador. Isto produz a soma de 200 + 300, ou 500. A terceira instrução adiciona o conteúdo do endereço 8 a esta soma, dando 900 no acumulador. Este número é então armazenado na memória no local 9 na memória. A máquina recebe então a ordem de parar. Observemos que a máquina é parada antes de alcançar os dados. Isto é para evitar que o elemento de controle apanhe os dados, por exemplo, o número 200, que está no endereço 6, e tente usá-lo como uma instrução.

Não há diferença entre um número e uma instrução no que concerne ao armazenamento. Ambos são armazenados na mesma forma básica. Daqui se conclui que as instruções são geralmente colocadas numa seção da memória diferente da dos dados a serem usados. O computador progride através das instruções e é parado antes de alcançar os dados. O fato de que quaisquer instruções ou dados possam ser armazenados em todos os endereços torna a máquina mais flexível. Pode ser visto que tanto uma grande quantidade de dados quanto poucas instruções, ou muitas instruções e poucos dados, podem ser usados enquanto a quantidade total de armazenamento disponível não for excedida.

1.14 INSTRUÇÃO DE MULTIPLICAÇÃO

Adicionando-se uma outra instrução, a de multiplicação, será possível escrever programas bem sofisticados (ver Quadro 1.3).

QUADRO 1.4

Endereço na memória	Palavra de instrução		Conteúdo do acumulador após a instrução ser executada
	Código de operação	Parte de endereço	
1	CLA	22	a
2	MUL	26	ax
3	ADD	23	$ax + b$
4	MUL	26	$x(ax + b)$ ou $ax^2 + bx$
5	ADD	24	$ax^2 + bx + c$
6	MUL	26	$x(ax^2 + bx + c)$ ou $ax^3 + bx^2 + cx$
7	ADD	25	$ax^3 + bx^2 + cx + d$
8	STO	27	$ax^3 + bx^2 + cx + d$
9	HLT	000	$ax^3 + bx^2 + cx + d$
.....			
22 contém a			
23 contém b			
24 contém c			
25 contém d			
26 contém x			
27 contém 0			

O programa no Quadro 1.4 calcula a expressão $ax^3 + bx^2 + cx + d$. As quantidades reais de a, b, c, d e x são armazenadas na memória nos locais 22, 23, 24, 25 e 26, respectivamente. É importante observar que o programa mostrado avalia a expressão para quaisquer valores que possam ser lidos nestes locais. A expressão pode ser avaliada para qualquer número de valores de x passando-se o programa para um valor de x, em seguida substituindo-se os valores sucessivos de x no registro 26 e repassando-se o programa para cada valor. Na prática real é possível ter-se o programa repetindo-o automaticamente por meio de instruções especiais. Todos os diversos valores desejados de x podem então ser armazenados e a equação resolvida para cada x sem parar o computador.

Pode ser visto, do Quadro 1.4, que funções algébricas muito complicadas podem ser calculadas usando-se apenas pouquíssimas instruções. O programa mostrado pode ser executado por um computador de alta velocidade em menos de 1/10.000 s. O valor de tal velocidade na solução dos mais complexos problemas encontrados na engenharia e na ciência pode ser visto prontamente. Os computadores estão tornando possíveis técnicas de engenharia que não foram anteriormente utilizáveis devido aos custos elevados decorrentes de tempo e de dinheiro em computações demoradas.

1.15 INSTRUÇÕES DE DESVIO, RAMIFICAÇÃO, SALTO OU QUEBRA DE SEQÜÊNCIA

Todas as intruções explicadas até agora foram usadas para executar problemas com cálculos aritméticos simples. No entanto, foi assinalado que o computador é capaz de repetir a mesma seqüência de instruções sem ser interrompido e reiniciado. Esta facilidade é proporcionada por um grupo de instruções denominadas instruções de *desvio* ou de *salto*. Estas instruções não dizem ao computador para efetuar a instrução no endereço seguinte ao da instrução que está sendo executada, mas para saltar para alguma outra instrução. Algumas instruções de desvio são *incondicionais* por natureza e fazem com que o computador salte independente de qual das condições possa estar. Outras instruções de desvio são *condicionais* e dizem ao computador para saltar somente se certas coisas forem verdadeiras. As instruções de desvio possibilitam ao computador fazer escolhas lógicas que alteram suas ações futuras.

Duas instruções típicas são mostradas no Quadro 1.5.

O pequeno programa mostrado no Quadro 1.6 ilustra vários princípios muito importantes. A finalidade do programa é adicionar todos os inteiros pares de 2 a 100. O programa é da categoria repetitiva onde algumas pequenas ordens são usadas para gerar um programa que passa durante algum tempo repetindo a mesma seqüência de instruções. O programa ilustra como a capacidade de desvio num número negativo pode ser usada para formar um

QUADRO 1.5

Código de operação	Parte de endereço	Função
BRA	420	Esta instrução diz ao computador para executar a instrução no endereço 420 seguinte. O computador saltará ou se desviará da instrução que ele deveria ter executado e executa, em vez disso, a instrução 420. O computador executará depois a instrução 421, seguida pela 422 etc.
BRM	420	O computador se desviará para a instrução 420 somente se o número no acumulador for negativo. Se o número for positivo ou zero, o computador não se desviará, mas executará a instrução armazenada no próximo endereço na memória. BRM é um código mnemônico para *branch on minus* (desviar no menos).

QUADRO 1.6

Endereço	Palavra de instrução		Índice do acumulador		
	Código de operação	Parte de endereço	1.ª vez	2.ª vez	Última vez
1	CLA	39	0	2	98
2	ADD	41	2	4	100
3	STO	39	2	4	100
4	ADD	43	2	6	2 + 4 + 6 + ... + 100
5	STO	43	2	6	2 + 4 + 6 + ... + 100
6	CLA	40	−50	−49	−1
7	ADD	42	−49	−48	+0
8	STO	40	−49	−48	+0
9	BRM	1	−49	−48	+0
10	HLT	000			

39 contém 000
40 contém −50
41 contém 2
42 contém 1
43 contém 0

contador que determinará quantas vezes uma parte de um programa é repetida.

O número armazenado no local 39 aumenta de 2 cada vez que o programa passa pela máquina. O total destes números é armazenado no endereço 43, que, após o programa ter parado, contém o total de todos os números que estiverem no local 39. O número armazenado no endereço 40 diminui em magnitude de 1 cada vez que o programa é executado, até que o número armazenado em 40 não é mais negativo. Quando a instrução BRANCH WHEN MINUS (BRM) — DESVIAR QUANDO MENOS — "falha", o programa então executa a próxima instrução em seqüência, que é uma instrução de PARADA. O zero é considerado um número positivo, embora isso varie nas diferentes máquinas. Observemos que a primeira seção do programa se repetirá um número de vezes igual ao número negativo armazenado no registro 40. Um contador simulado é formado pelo −50 armazenado no endereço 40, pelo 1 armazenado no local 41 e pelas instruções nos endereços 6 a 9. Qualquer seqüência de instruções que precede um contador desta espécie será executada um número de vezes determinado pelo contador. Este é um dispositivo especialmente útil aos esquemas iterativos quando se conhece o número de iterações requeridas.

1.16 LEITURA DO PROGRAMA NO COMPUTADOR

Após o programador ter escrito a lista de instruções que o computador deve executar, uma outra etapa deve ser conduzida antes do programa escrito poder ser lido na máquina. Esta etapa geralmente consiste em registrar o programa em algum meio, tal como cartão perfurado ou fita de papel perfurada, que os dispositivos de entrada do computador possam ler. Se o meio de entrada for fita de papel perfurada, o programa escrito será dado a um operador de uma máquina de perfuração de fita de papel. A máquina de perfuração de fita possui um teclado que lembra o de uma máquina de escrever que contém caracteres tanto alfabéticos quanto numéricos. Uma longa tira de fita de papel é introduzida na seção de perfuração da máquina quando o operador da máquina maneja o teclado. Cada vez que o operador abaixa uma tecla do teclado, são feitas várias perfurações na fita de papel. O mecanismo do teclado é arranjado de modo que seja perfurada na fita uma combinação diferente de perfurações para cada tecla que é abaixada. O operador da máquina de perfuração de fita datilografa o programa tal como uma secretária datilografa uma carta. O resultado é uma tira de fita de papel que foi perfurada com grupos codificados de perfurações.

Uma leitora de fita de papel conectada ao computador digital é usada para ler esta fita

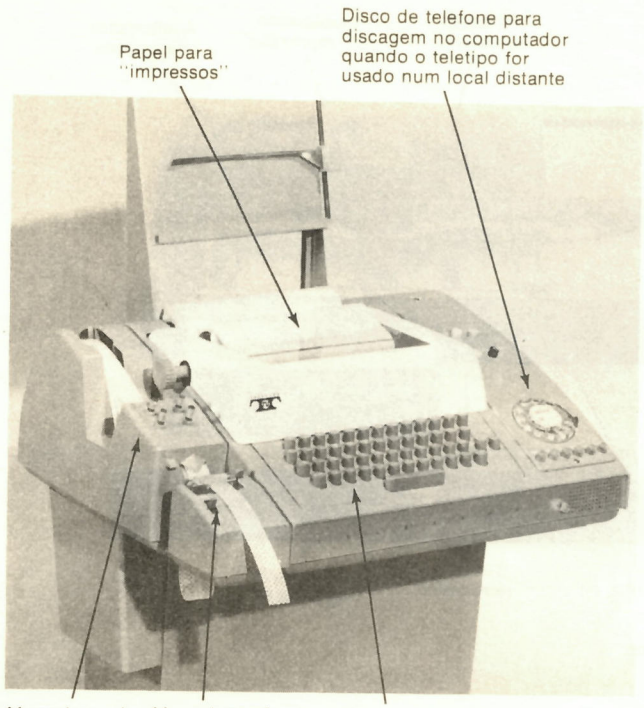


Fig. 1.7 Console ou painel de teletipo com perfuradora de fita. (Cortesia da AT & T.)

no computador. A leitora de fita de papel "percebe" as perfurações feitas na fita e fornece os locais destas perfurações na fita para o computador sob a forma de sinais elétricos. O computador então armazena esta informação. Uma descrição das leitoras e das perfuradoras de fita de papel, e da maneira pela qual a fita de papel é codificada, será apresentada no Cap. 8.

A Fig. 1.7 mostra um teletipo que pode perfurar ou ler fita de papel com um teclado que pode também ser incorporado às linhas telefônicas (observar o disco no canto inferior direito). Uma pequena seção de fita pode ser vista estendendo-se desde o dispositivo de perfuração de fita. Depois de uma fita ser perfurada, ela pode ser lida por uma leitora de fita de computador de alta velocidade. No entanto, muitas vezes, um teletipo é conectado diretamente a um computador, e sua leitora de fita é usada para ler a fita no computador.

A preparação de cartões perfurados é semelhante. O operador de uma perfuradora de cartões (ver Fig. 1.8) comprime as teclas de um teclado que faz perfurações nas colunas sucessivas de um "cartão perfurado" retangular. O conjunto de cartões assim perfurados é então montado em ordem numa "pilha" de cartões que pode ser lida por uma leitora de cartão perfurado.

O procedimento geral para cartões perfurados consiste em que o programador escreva o programa num formulário especial de codificação tal como o mostrado na Fig. 1.9. Um teclado é então usado para datilografar (ou imprimir) o programa escrito no formulário. A Fig. 1.9 também mostra um cartão perfurado contendo a primeira linha da página 2 do programa. Em geral, uma linha de escrita num formulário de codificação é introduzida num único cartão.†

†O programa no formulário de codificação da Fig. 1.9 é uma seção de um programa de processamento de dados numa empresa escrito numa linguagem de programação de "alto nível" chamada PL/1 (ver Seção 1.20). Observamos que a linguagem é semelhante à utilizada pelos contadores ou contabilistas. (Quanto a isto, a PL/1 se assemelha a uma primitiva linguagem comercial chamada Cobol.)

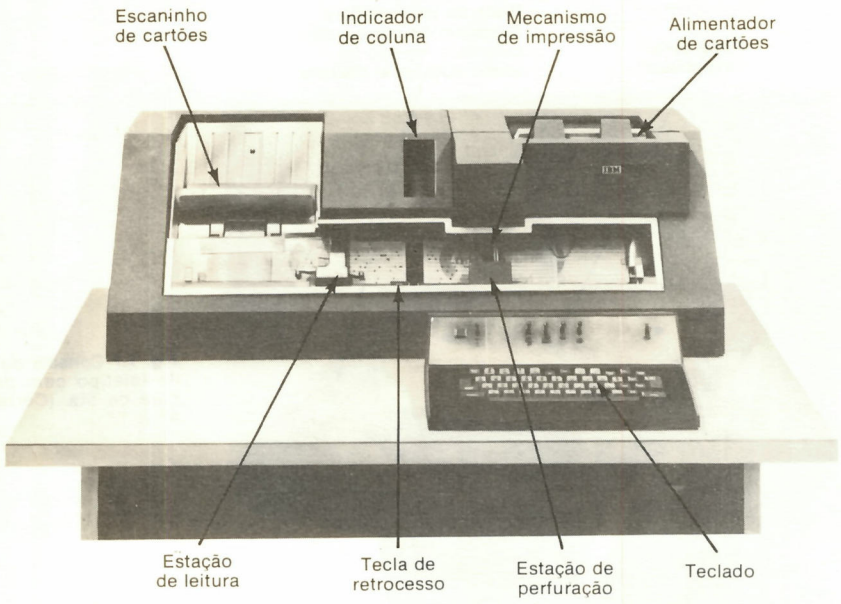


Fig. 1.8 Perfuradora de cartões.

Existem inúmeras outras técnicas para entrada de dados e de programa nos computadores. A Fig. 1.10 mostra um cartão especial no qual os dados podem ser introduzidos marcando-se simplesmente as caixas no cartão com um lápis macio. Um dispositivo especial de leitura de "marca sensível" pode perceber a posição das caixas que foram marcadas e transmitir sua localização no cartão para um computador.†

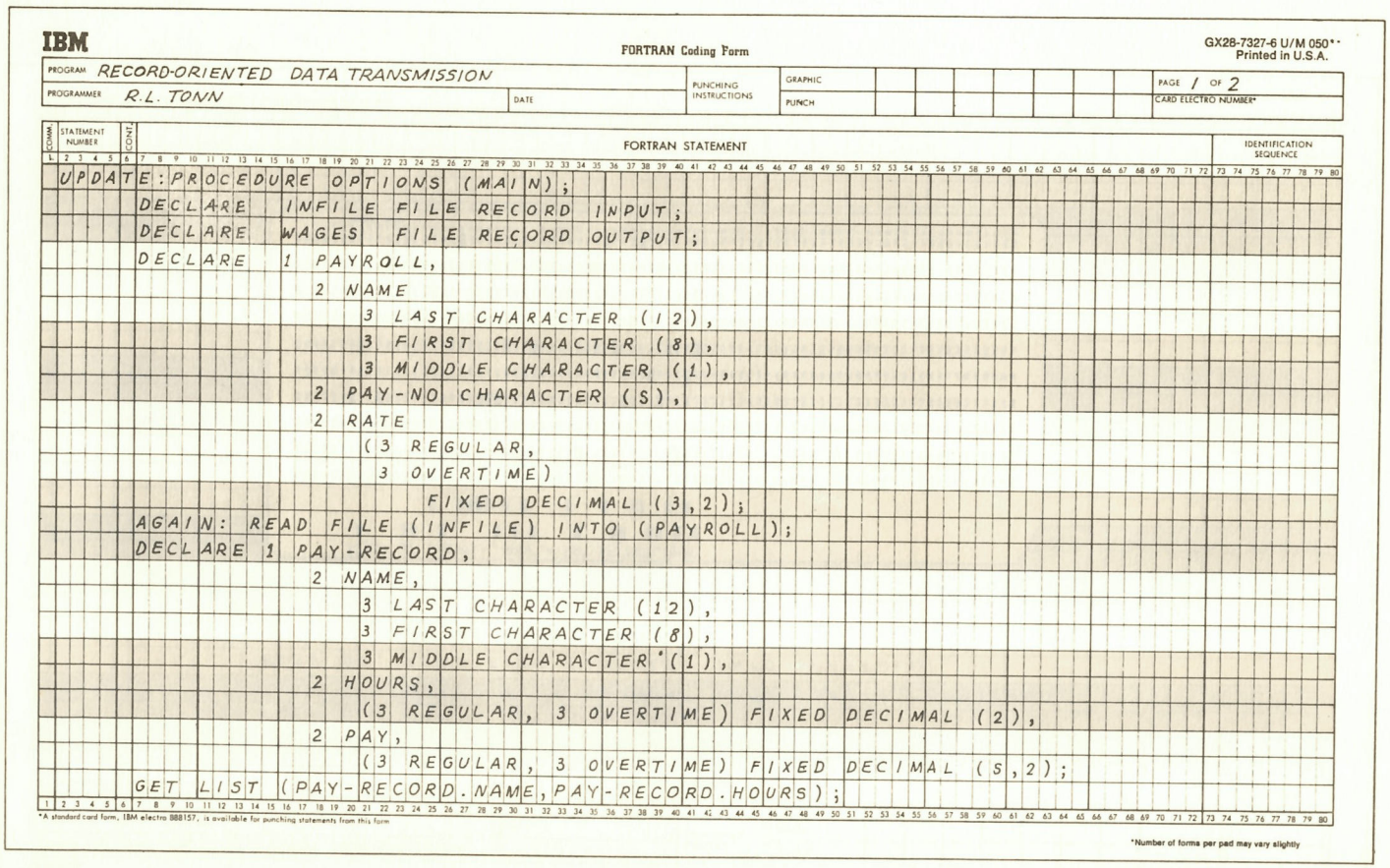
Um outro modo de entrada de programa é nos sistemas *em-linha*, onde o operador de um teclado datilografa o programa diretamente no computador. Neste caso, o teclado contém um conjunto de circuitos elétricos que gera sinais para o computador cada vez que uma chave é abaixada. Geralmente, um mecanismo de impressão também datilografa ou imprime os caracteres que o operador imprime no papel. O computador geralmente possui a capacidade de escrever as respostas usando o mesmo mecanismo de impressão que foi usado para registrar o programa e os dados introduzidos, ou, às vezes, o computador dá as respostas por meio de um indicador de osciloscópio ou de algum outro dispositivo gráfico. Tais sistemas permitem que o computador e seu usuário se comuniquem diretamente em vez de usar meios intermediários tais como fita ou cartões.

A Fig. 1.11 mostra um teclado e um indicador de osciloscópio num sistema em que a informação manipulada por teclado no console é registrada no papel (observar o rolo de papel na impressora acima do teclado) bem como exibida no osciloscópio. Tanto a impressora quanto o osciloscópio são também usados para imprimir ou exibir a informação de saída do computador.

A maioria dos dados introduzidos nos computadores é ainda sob a forma de cartões perfurados. (Há cerca de 500.000 perfuradoras de cartões atualmente em uso ativo.) O uso de sistemas em-linha e de outros meios de entrada está crescendo, no entanto, e representa uma porção substancial do uso global dos computadores.

A descrição precedente das etapas na preparação de um programa deve tornar evidente que nem sempre é econômico usar um computador de grande porte para efetuar seqüências de cálculos muito simples que necessitem ser executadas apenas uma vez ou

†O cartão que foi mostrado possui uma única declaração proveniente de uma linguagem de programação chamada Basic escrita nele.



IBM

FORTRAN Coding Form

GX28-7327-6 U/M 050**
Printed in U.S.A.

PROGRAM: RECORD-ORIENTED DATA TRANSMISSION
PROGRAMMER: R.L. TONN
DATE:
PUNCHING INSTRUCTIONS — GRAPHIC / PUNCH
PAGE 1 OF 2
CARD ELECTRO NUMBER*

C	STATEMENT NUMBER	CONT.	FORTRAN STATEMENT	IDENTIFICATION SEQUENCE
			UPDATE:PROCEDURE OPTIONS (MAIN);	
			DECLARE INFILE FILE RECORD INPUT;	
			DECLARE WAGES FILE RECORD OUTPUT;	
			DECLARE 1 PAYROLL,	
			2 NAME	
			3 LAST CHARACTER (12),	
			3 FIRST CHARACTER (8),	
			3 MIDDLE CHARACTER (1),	
			2 PAY-NO CHARACTER (5),	
			2 RATE	
			(3 REGULAR,	
			3 OVERTIME)	
			FIXED DECIMAL (3,2);	
			AGAIN: READ FILE (INFILE) INTO (PAYROLL);	
			DECLARE 1 PAY-RECORD,	
			2 NAME,	
			3 LAST CHARACTER (12),	
			3 FIRST CHARACTER (8),	
			3 MIDDLE CHARACTER (1),	
			2 HOURS,	
			(3 REGULAR, 3 OVERTIME) FIXED DECIMAL (2),	
			2 PAY,	
			(3 REGULAR, 3 OVERTIME) FIXED DECIMAL (5,2);	
			GET LIST (PAY-RECORD.NAME,PAY-RECORD.HOURS);	

*A standard card form, IBM electro 888157, is available for punching statements from this form

*Number of forms per pad may vary slightly

Fig. 1.9 Formulários de codificação e um cartão perfurado.

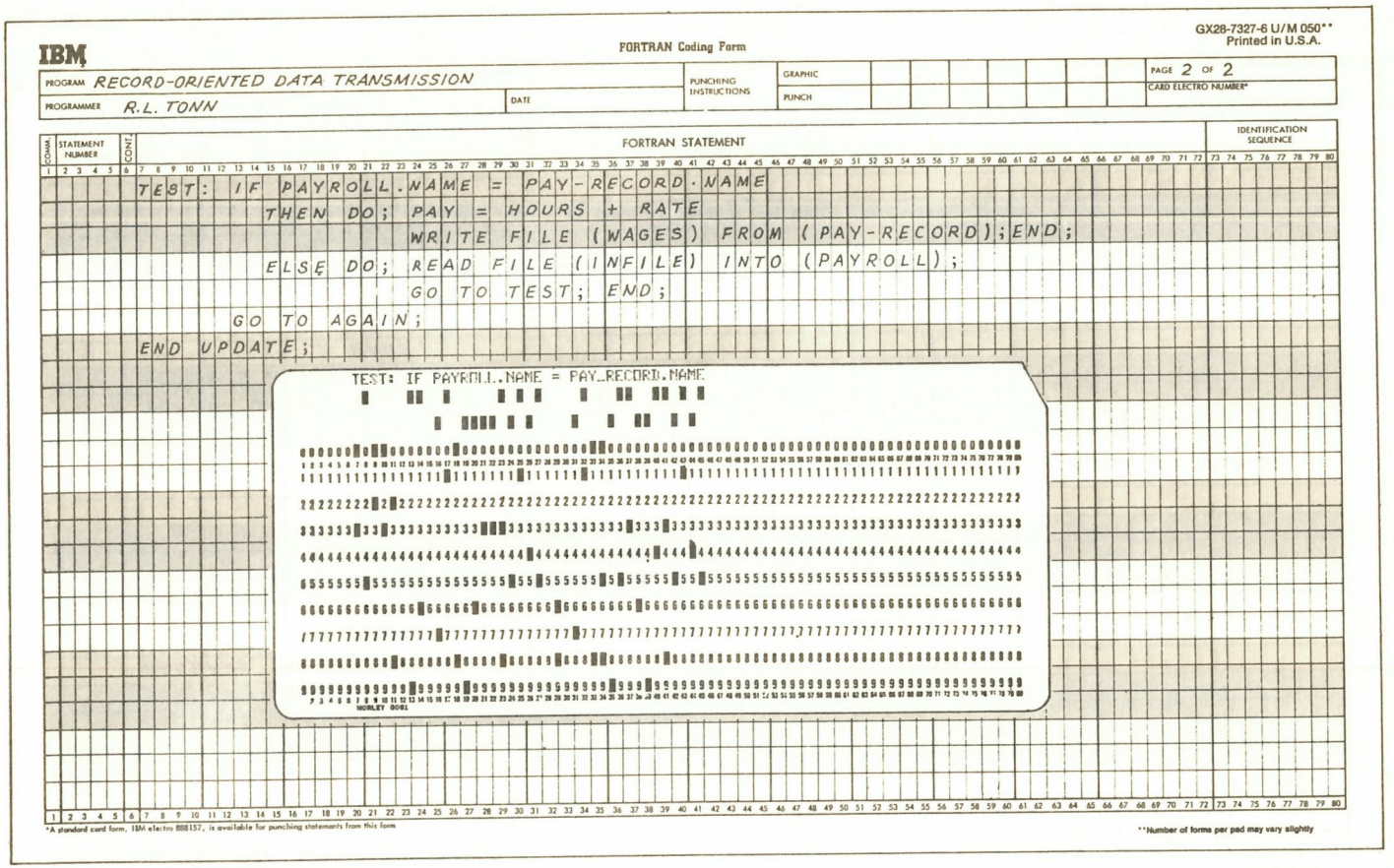


Fig. 1.9 *(Continuação.)*

Observações: As declarações da linguagem BASIC possuem três elementos: número da declaração, tipo da declaração e operandos. O cartão abaixo está marcado para a declaração: 150 LET A = B + C − 5

O número da declaração (150) é indicado no campo NÚMERO DA DECLARAÇÃO por uma única marca de lápis nas caixas rotuladas, 1, 5 e 0.

O tipo da declaração (LET) é indicado no campo DECLARAÇÃO por uma única marca na caixa rotulada LET.

O operando (A = B + C − 5) é indicado no campo FÓRMULA.

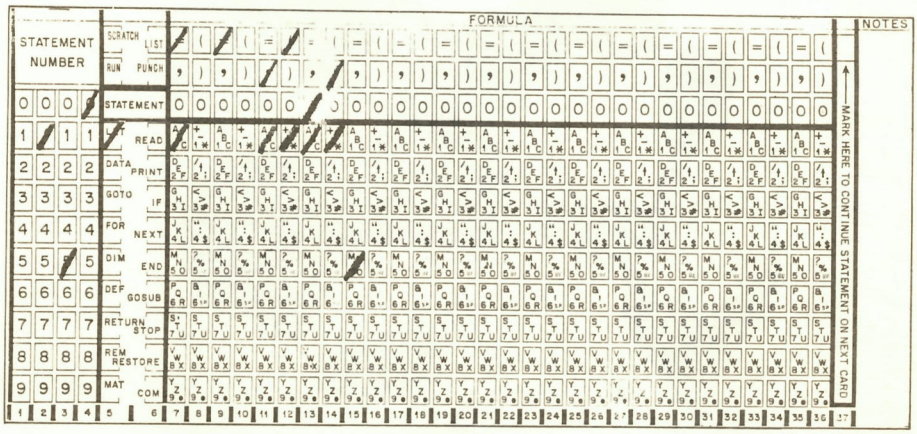


Fig. 1.10 Cartão com marcas de lápis que podem ser lidas por uma leitora sensível a marcas.

duas vezes. O computador é valioso quando muitos conjuntos repetitivos de cálculos devem ser efetuados.

Consideremos o procedimento da contabilidade de folha de pagamento descrito inicialmente. Uma vez que tenha sido escrito um programa que calculará a renda bruta semanal, impostos, descontos etc. dos empregados de uma grande fábrica, o mesmo programa pode ser usado para computar cada um dos milhares de cheques de empregados. Se um dispositivo combinado de relógio e perfuradora de cartão for usado para perfurar a hora que um empregado entra numa fábrica e a hora que o empregado sai num cartão de ponto que também registra, em forma perfurada, um número de identificação do empregado, toda a operação pode ser feita automaticamente. Os cartões de ponto podem então ser "lidos" pelo computador, que computará o tempo de trabalho, os salários totais por semana etc. e imprimirá realmente o cheque de pagamento e, a seguir, registrará em formulário impresso todos os dados requeridos pelo sistema de escrituração.

A Fig. 1.12 mostra um sistema em que fichas de razão (livro de contabilidade) são usadas para registrar dados impressos provenientes do computador (e dados manipulados em teclado, se desejado). As fichas de razão costumam ser amplamente usadas nos sistemas de contabilidade em faturamento, escrituração e manipulação de registros em geral. As fichas de razão são introduzidas na abertura do alimentador no centro e logo acima do teclado. Podem ser usados diferentes tamanhos de fichas de razão nas diversas aplicações. Este tipo de impressora é típico das usadas em muitas aplicações comerciais.

1.17 SISTEMAS DE PROGRAMAÇÃO

A discussão precedente mostrou um procedimento básico para escrever um programa introduzindo-o no computador. Há, no entanto, vários tipos de *linguagens de programação* que facilitam enormemente a escrita real de programas. Uma das primeiras coisas que a profissão de programação descobriu foi que o maior auxílio à programação foi o próprio



Fig. 1.11 Teclado com indicador de osciloscópio e impressora anexos.

computador; ele foi considerado capaz de traduzir programas escritos de uma linguagem que era direta e natural para o programador em linguagem de máquina ou de computador.

Em conseqüência, foram escritos programas cuja finalidade era ler outros programas escritos numa linguagem natural para o programador e traduzi-los na linguagem de máquina. Os sistemas de programação atualmente em uso são principalmente de dois tipos: *montadores* e *compiladores*.† O montador e o compilador são destinados à mesma finalidade básica; cada um deles é um programa projetado para ler um programa escrito numa linguagem de programação e traduzi-lo. O montador ou o compilador é lido primeiro na máquina e é então seguido pelo programa a ser traduzido. Após a tradução, o montador ou o compilador armazena o programa em linguagem de máquina nos cartões perfurados ou em fita magnética ou de papel ou ainda em alguma memória grande, de modo que ele pode ser executado quando se desejar.

O propósito deste procedimento é permitir que os programadores escrevam as operações que eles querem que o computador execute de uma maneira mais simples do que a linguagem de máquina. A linguagem que o programador escreve é chamada *linguagem de programação*, e um programa escrito em tal linguagem é chamado *programa fonte*. O programa traduzido em linguagem de máquina (ou em alguma intermediária) é chamado *programa objeto*.

---

†Há também um tipo de programa tradutor que traduz uma linha ou declaração de uma vez, chamado *interpretador* e usado com linguagens de programação tais como Basic.



Fig. 1.12 Teclado e impressora de fichas de razão (livro de contabilidade).

Para retornar ao assunto dos programas tradutores, uma linguagem de montador difere de uma linguagem de compilador em que a maior parte da linguagem de montador assemelha-se rigorosamente à linguagem de máquina, principalmente porque cada instrução para o computador em linguagem de montador é traduzida numa única palavra de computador. Em sistemas de compilador, uma única instrução para o computador pode ser convertida em muitas palavras de computador.

1.18 LINGUAGENS DE MONTADOR

Cada instrução para o computador numa linguagem de programação artificial é chamada *declaração*. A característica básica que mais distingue uma linguagem de montador é que cada declaração é traduzida pelo programa montador numa única palavra de instrução de máquina.† Em conseqüência, uma linguagem de montador assemelha-se um tanto à linguagem de máquina. As facilidades oferecidas ao programador são substanciais, no entanto, e geralmente incluem:

1. *Códigos mnemônicos de operação.* O programador pode escrever instruções para o computador usando letras em vez de números binários, e as letras que designam uma

†Naturalmente, este não é estritamente o caso (às vezes, uma única declaração pode ser traduzida em várias palavras), mas é, em geral, verdadeiro.

dada operação podem ser arranjadas num código mnemônico que transmite o "sentido" da instrução. Em nosso exemplo precedente de codificação foram usados os códigos mnemônicos ADD, MUL, CLA etc. O montador deve traduzir estes códigos mnemônicos em binário de máquina correto ou em números decimais codificados em binário e "empacotar" estes nas palavras de instrução que constituem o programa objeto.

2. *Referência simbólica de endereços de armazenamento.* Uma das maiores facilidades oferecidas ao programador é a capacidade de designar (atribuir um nome) as diferentes peças de dados usadas no programa e ter o montador endereços atribuídos automaticamente a cada nome.

   Se desejarmos avaliar a expressão algébrica $y = ax^3 + bx^2 + cx + d$ como na Seção 1.14, o programa pode aparecer conforme indicado no Quadro 1.7.

   Observemos que ao endereço da primeira instrução foi simplesmente dado um nome FST, que consiste de três letras, e que não foram especificados outros endereços na memória. Se dissermos ao montador que FST = 1, o montador então verá que as instruções estão colocadas na memória como no programa na Seção 1.14. Observemos também que aos operandos foram simplesmente dados os nomes de variáveis X, A, B, C e D, como na equação, em vez de atribuir a eles endereços na memória. O programa montador atribuirá endereços a esses nomes de variáveis, e se ele atribui A a 22, B a 23, C a 24 etc., o programa final parecerá o da Seção 1.14.

   O montador verá também que os valores aritméticos reais de X, A, B, C e D são colocados nos locais corretos na memória quando os dados são lidos no computador.
3. *Representação conveniente dos dados.* Isto significa simplesmente que o programador pode escrever os dados de entrada como, por exemplo, números decimais ou letras, ou talvez em alguma forma específica para o problema, e que o programa montador converterá os dados desta forma na forma requerida para computação de máquina.
4. *Listagens de programa.* Uma característica importante da maioria dos montadores é sua capacidade de imprimir para o programador uma listagem do programa fonte e também uma listagem do programa objeto que é em linguagem de máquina. Um estudo destas listagens auxiliará grandemente o programador na pesquisa de quaisquer erros cometidos na escrita do programa e também na modificação do programa quando isto for exigido.
5. *Detecção de erro.* Um programa montador também notificará ao programador se for cometido um erro na utilização da linguagem do montador. Por exemplo, o programador pode usar o mesmo nome da variável, por exemplo X, duas vezes, e então dar a X dois valores diferentes, ou o programador pode escrever códigos de operação ilegais etc. Esta espécie de diagnose de um erro de programa é muito útil durante o ensaio definitivo de um novo programa.

Os montadores proporcionam muitas outras facilidades que auxiliam o programador, tal como a capacidade de usar programas que já foram escritos como parte de um novo programa e a capacidade de usar rotinas oriundas destes programas como parte de um novo programa. Freqüentemente, os programadores têm um conjunto de programas diver-

QUADRO 1.7

	Palavra de instrução	
Endereço na memória	Operação	Operando
FST	CLA	A
	MUL	X
	ADD	B
	MUL	X
	ADD	C
	MUL	X
	ADD	D
	STO	Y
	HLT	

QUADRO 1.8

Endereço	Palavras de instrução		Comentários
	Operação	Operando	
A	DEC	0	
B	DEC	−50	
C	DEC	2	
D	DEC	1	
E	DEC	0	
N	CLA	A	último valor de inteiro
	ADD	C	
	STO	A	armazena a soma para o tempo seguinte
	ADD	E	
	STO	E	
	CLA	B	
	ADD	D	
	STO	B	
	BRM	N	
	HLT		

sos que eles passarão juntos na máquina em diferentes combinações. Isto se torna possível simplesmente especificando-se ao montador os nomes de variáveis nos diferentes programas as quais devem ser as mesmas variáveis, os pontos de entrada e de saída para os programas etc. Isto significa que os programas escritos numa linguagem de montador podem ser encadeados de várias maneiras.

Consideremos o pequeno programa no Quadro 1.6 que soma os inteiros pares de 0 a 100. Isto ilustrará o uso de nomes simbólicos para endereços quando for usada uma instrução de desvio. O programa de linguagem do montador é mostrado no Quadro 1.8.

Observemos que os valores das variáveis foram especificados antes que o programa fosse iniciado; os DEC indicam que os valores dados a A, B, C, D e E estão em decimal. Isto permite ao montador localizar as variáveis na memória e atribuir valores a elas.† Além disso, note que a instrução BRM de transferência foi para o endereço simbólico N.

Se ao montador fosse dito iniciar o programa no endereço 1 na memória, a conversão em linguagem objeto ou de máquina faria com que ele se assemelhasse ao montador da Seção 1.15, contanto que o montador decidisse armazenar A, B, C, D e E nos locais 39 a 43. Além disso, os códigos de operação de CLA, MUL etc. seriam convertidos à forma numérica ou de máquina, se isto fosse necessário.

1.19 LINGUAGENS DE COMPILADOR

Os tipos mais avançados de linguagens de programação chamam-se *linguagens de compilador, linguagens de alto nível* ou *linguagens orientadas para o problema.* Estas são as linguagens mais simples de se usar e também as mais simples de se aprender. Estas linguagens revelam, no entanto, muito pouco sobre as máquinas digitais nas quais elas são passadas. O projetista da linguagem geralmente se concentra na especificação de uma linguagem de programação que seja bem simples para o usuário casual de um computador digital e ainda que tenha facilidades suficientes para tornar a linguagem e seu compilador associado valiosos para os programadores profissionais. De fato, muitas linguagens são quase que completamente independentes do computador, e os programas escritos numa destas linguagens podem ser passados em qualquer computador que tenha um compilador ou tradutor da linguagem em sua biblioteca de programas.

Certas linguagens foram muito bem-sucedidas e têm encontrado uso extremamente

†De certo modo, o código DEC de operação diz "atribuir o valor decimal na coluna operacional ao nome da variável na coluna de endereços".

amplo na indústria de computadores. A linguagem mais famosa é o Fortran, que é a mais primitiva das linguagens e foi regularmente atualizada. Um programa escrito em Fortran pode ser passado na maioria dos computadores comerciais que tenham dimensões de memória suficientemente grandes para acomodar um compilador Fortran, porque a maioria dos fabricantes preparará um compilador Fortran para seus computadores.

Há diversas outras linguagens em uso ativo. O Algol, uma linguagem que foi projetada para uso internacional, é amplamente usada na descrição de algoritmos em revistas técnicas e tem tido sucesso moderado na prática real; o Cobol é uma linguagem especialmente projetada para sistemas comerciais; o Mad e diversas outras linguagens foram desenvolvidas pelas universidades com propósitos educacionais; o Formac faz manipulação simbólica; Jovial é uma linguagem para sistemas em tempo real; o Basic é uma linguagem muito simples, fácil de usar, e que é freqüentemente usada nos sistemas em-linha e que se tornou muito popular etc. Tudo que pode ser dito é que não há verdadeiramente uma linguagem de programação universal, mas que há diversas boas linguagens. Felizmente, não há uma grande diferença entre essas linguagens de um ponto de vista conceitual e, quando se aprende uma das linguagens, o aprendizado de uma outra não constitui grande problema.

Uma linguagem de programação popular é denominada Linguagem de Programação Um (abreviada PL/1). Algumas das características desta linguagem bastarão para ilustrar o aspecto geral das linguagens do compilador. O PL/1 foi escolhido devido à sua simplicidade nas equações algébricas e por causa do formato muito elementar para leitura dos dados e impressão dos resultados. Além disso, o PL/1 neste nível quase que se assemelha ao Fortran, exceto que não necessitamos nos preocupar com a mistura de variáveis reais com as variáveis inteiras, nem com as declarações de formato para impressão ou leitura, contanto que estejamos querendo usar o formato padrão do PL/1 (neste, adotamos a atitude do Basic, e nossos programas serão muito semelhantes). Finalmente, foram desenvolvidas versões simplificadas de PL/1 pelos fabricantes de microprocessadores e de computadores microprogramados.

Uma vez que a seção seguinte apresenta uma introdução a uma linguagem de compilador específica, aqueles que estão familiarizados com as linguagens de compilador tais como Fortran ou Cobol podem omitir esta próxima seção. Outros podem desejar deixar este assunto para mais tarde. O texto é arranjado de modo que o assunto subseqüente no livro não dependa da seção seguinte, e esta seção pode ser omitida ou estudada posteriormente. A matéria é introduzida neste ponto porque muitos acreditam ser uma boa idéia estudar uma linguagem de compilador bem antes de estudar os computadores (ou mesmo pode haver uma falsa concepção de que os programadores de computador devem realmente escrever seus programas em linguagem de máquina ou de computador, o que limitaria grandemente o uso dos computadores). Em muitos casos, apenas uma leitura desta matéria será suficiente. Àqueles que desejam estudá-la com maior profundidade, foram preparadas algumas questões no fim do capítulo que exigirão um estudo de algumas características adicionais de PL/1, e estas poderão ser usadas como um suplemento.

*1.20 UMA BREVE INTRODUÇÃO À LINGUAGEM PL/1

Conforme foi mencionado, a característica distintiva dos compiladores e suas linguagens de programação é que uma única declaração que o programador escreve pode ser convertida pelo compilador em várias instruções em linguagem de máquina. Em PL/1, por exemplo, uma única declaração para o computador pode gerar inúmeras instruções em linguagem objeto ou de máquina. Em conseqüência, a linguagem não depende particularmente da estrutura do computador no qual o programa é passado, e os programas seguintes podem ser passados em qualquer computador que tenha um compilador de PL/1.

O compilador de PL/1 é escrito em linguagem de máquina, certamente, e um programa de PL/1 é fundamentalmente passado em linguagem de máquina. Por esta razão, o conhecimento da máquina e de sua organização pode ser de grande utilidade para aqueles que descrevem e ensaiam programas. Além disso, para os programadores de sistemas (aqueles que mantêm, modificam e preparam os compiladores, montadores, programas de carga etc.), um conhecimento da máquina na qual o programa é operado torna-se indispensável. O fato de que os compiladores são reforçados por um exército de pessoal técnico, desde programadores de sistemas até engenheiros de manutenção e de projeto, técnicos e operadores de computador, é muitas vezes tolerado pelo usuário da máquina cujo programa é prodigiosamente depurado e operado por profissionais no campo dos computadores. Como a maioria dos dispositivos eletrônicos, os computadores digitais não são tão independentes e automantidos como parecem aos usuários casuais.

Assim prevenidos, examinaremos a estrutura do PL/1 em poucos detalhes, deixando uma exposição mais completa para uma das referências.

A primeira coisa a ser aprendida ao escrever-se declarações é o uso do símbolo =. Isto é tratado como um comando para o computador, para quando escrevemos Y = A + B, o que se entende por "substituir o valor corrente de Y pelo de A + B." Assim, portanto, se escrevemos

```
A = 50;
B = 20;
Y = A + B;
END;
```

achamos que depois disto ser passado, A será igual a 50, B a 20 e Y a 70. Observemos que cada declaração é seguida por um ponto-e-vírgula. Como um outro exemplo, podemos aumentar, diminuir, ou de outro modo, variar o valor corrente de Y adicionando ou subtraindo dele. Consideremos

```
Y = 30;
A = 40;
Y = Y + A;
END;
```

Depois destas quatro declarações serem operadas, Y terá o valor de 70; isto significa que o local na memória que foi usado para armazenar Y terá nele o valor 70. Observe que as declarações acima tomaram a forma de equações algébricas. Aqui está um outro exemplo:

```
Y = 20;
Z = 50;
W = Y + Z;
M = W - 30;
END;
```

Depois disto ser passado, W terá o valor 70 e M, o valor 40.

Em PL/1, o símbolo de adição é o familiar + e o símbolo de subtração é o usual −, mas a multiplicação é indicada por um asterisco (*), e a exponenciação (elevação a uma potência) por dois asteriscos. Assim, A*B significa "multiplicar A por B" e A**B significa A^B ou "elevar A à B-ésima potência". Portanto, as declarações de programa

```
A = 20;
B = 30;
C = A * B;
END;
```

dão um valor de 600 para C, e o programa

```
A = 20;
B = 2;
C = A ** B;
END;
```

dá um valor de 400 a C.

A divisão é indicada por um símbolo /. Assim,

```
A = 20;
B = 2;
Y = 20/2;
END;
```

colocará o valor 10 no local na memória confiado a Y.

Examinemos uma simples maneira de formar um laço no programa, isto é, repetir uma seqüência de instruções até que desejemos parar. Aqui estão duas declarações de PL/1:

```
LOOP. DO WHILE X LE Y;
```

e

```
END LOOP;
```

A primeira destas diz "faça o seguinte conjunto de declarações até a END LOOP, enquanto X for menor do que ou igual a Y." Em outras palavras, "repita as seguintes instruções até que Y seja maior do que X." Se escrevermos as seguintes declarações:

```
T = 0;
M = 4;
N = 2;
P = 6;
LOOP. DO WHILE N LE M;
S = P * N;
T = T + S;
N = N + 1;
END LOOP;
Y = T * 2;
END;
```

então as declarações entre o LOOP e END LOOP serão repetidas até que N seja maior do que M. Uma vez que N inicia com o valor 2 (e como 1 é adicionado de cada vez) enquanto M começa com o valor 4, N assumirá os valores 2, 3, 4, 5. Mas, quando N é igual a 5, ele será maior do que M, e o programa prosseguirá com a instrução Y = T*2; e depois END. As declarações LOOP serão, portanto, repetidas três vezes. A primeira vez, S será igual a P vezes N, ou 12, T será igual a 0 + S, ou 12, e N será então aumentado de 2 a 3. A segunda vez, S será igual a P vezes N, ou 6 vezes 3, que são 18; T será igual a 12 + 18, ou 30; e N será aumentado para 4. A terceira e última vez através do laço, S será igual a 6 vezes 4, ou 24, T assumirá o valor 30 + 24, ou 54, e N será aumentado para 5. N será então maior do que M, e a declaração END LOOP será operada. T, que tem o valor 54, será então multiplicado por 2, dando 108, e o programa irá parar com Y igual a 108.

Examinemos duas outras características. Para ler dados, simplesmente escrevemos

```
READ (X, Y, Z);
```

e isto dirá ao computador para arranjar para leitura o valor de X, Y e Z dos cartões e continuar com os valores lidos como valores correntes de X, Y e Z. Devemos, portanto, fornecer os valores de X, Y e Z em cartões perfurados. A vantagem disto é que podemos mudar os valores de X, Y e Z substituindo simplesmente nossos cartões de dados por cartões que contêm novos valores. Se escrevermos

```
READ (X, Y, Z);
M = X + (Y * Z);
END;
```

e fixarmos cartões com os valores X = 20, Y = 30 e Z = 2, teremos M = 80. Se mudarmos nossos cartões de dados para ler X = 5, Y = 3 e Z = 4, teremos então M = 17 após passarmos o programa.

Para imprimir dados, escrevemos a declaração

```
WRITE (X, Y, Z, A),
```

e o computador imprimirá os valores correntes de X, Y, Z e A.

Deve ser observado que as declarações READ e WRITE admitem que o programador estará satisfeito com o formato padronizado das declarações de imprimir e de dados de entrada. Admitindo que este seja o caso, podemos escrever o seguinte programa, que avaliará primeiro a equação $Y = AX^3 + BX^2 + CX + D$ para valores de A, B, D e X que são lidos nos cartões de dados ou em fita e para C = 1. Se o valor de Y para estes valores particulares for maior do que 2.000, o programa imprimirá o valor de Y como calculado e também o valor C = 1. Se, no entanto, Y for menor do que ou igual a 2.000, o programa calculará o menor inteiro positivo que, quando substituído em C, fará $AX^3 + BX^2 + CX + D$ maior que 2.000. O programa imprimirá então este valor de C e o valor de $AX^3 + BX^2 + CX + D$ associado ao valor de C.

```
READ (A, B, D, X);
     C = 0;
     Y = 0;
LOOP. DO WHILE Y LE 2000;
     C = C + 1;
     Y = A * X ** 3 + B * X ** 2 + C * X + D;
END LOOP;
PRINT (C, Y);
END;
```

Neste tópico, apenas falamos do poder desta linguagem de programação. Não é possível numa curta exposição fazer mais do que mostrar várias declarações e dar uma idéia geral de como opera tal linguagem. Entretanto, um programador habilidoso poderá fazer muito valendo-se somente do vocabulário restrito que introduzimos.

Um fato que deve ser destacado nesta breve introdução é que uma vez aprendidos os tipos de declaração e os detalhes da linguagem, a tarefa de programação de um determinado problema se torna grandemente facilitada.

EXERCÍCIOS

1. Discutir processamento em lotes e dar diversos exemplos de empresas comerciais e industriais que possam usar processamento em lotes.
2. Discutir os sistemas de processamento em-linha e dar exemplos de empresas comerciais e industriais que possam usar sistemas em-linha.
3. Dar exemplos de indústrias que possam usar sistemas de controle em tempo real em fabricação.
4. Às vezes, o mesmo computador é usado por diversas companhias diferentes durante o dia, e o computador é compartilhado em tempo entre estas companhias. Discutir os problemas que possam surgir no faturamento das companhias pelos serviços do computador.
5. Os valores de X, Y e Z são armazenados nos endereços da memória 40, 41 e 42, respectivamente. Usando as instruções para o computador generalizado de um só endereço descrito nas Seções 1.13 a 1.15, escrever um programa que formará a soma X + Y + Z e armazená-lo no endereço 43 da memória.
6. Explicar a diferença entre o *endereço* de uma palavra na memória e a própria *palavra*.
7. Sabendo-se que os valores de X, Y e Z são armazenados nos locais 20, 21 e 22, respectivamente, usar as instruções para o computador dado nas Seções 1.13 a 1.15 para escrever um programa que formará $X^2 + Y^2 + Z^2$ e armazenar isto no endereço 40 da memória.
8. O programa no Quadro 1.4 é passado com os seguintes valores quando o programa é iniciado: 4 no endereço 22, 3 no endereço 23, 4 no endereço 24, 5 no endereço 25 e 2 no endereço 26. Que valor estará no endereço 27 após o programa ter sido passado na máquina?
9. Converter o fluxograma da Fig. 1.2 num programa de computador em linguagem de montador, usando as instruções descritas no capítulo.
10. Se o programa no Quadro 1.4 for iniciado com o local 22 contendo 5, o local 23 contendo 4, o local 24 contendo 3, o local 25 contendo 1 e o local 26 contendo 2, que

número será armazenado no local 27 após o programa ser passado na máquina?

11. Escrever um programa que armazenará $X^5 + X$ no registro 40, usando a linguagem de montador nas Seções 1.13 a 1.15, sabendo-se que X está no registro 20. Usar menos de 20 instruções. Reescrever agora este programa usando a linguagem de montador da Seção 1.18.
12. Se o programa no Quadro 1.6 fosse operado, mas o número do endereço 40 fosse −20 em vez de −50 quando o programa foi iniciado, o número no endereço 43 seria a soma de todos os inteiros pares desde 2 em vez de −2 até 100.
13. Desenhar um fluxograma para o programa da Seção 1.15.
14. Sabendo-se que um valor de X é armazenado no endereço 39, escrever um programa que formará X^4 e armazená-lo no endereço 42, usando a linguagem de montador nas Seções 1.13 até 1.15.
15. Desenhar um fluxograma que mostra como achar o maior número num conjunto de cinco números armazenados nos locais 20, 21, 22, 23 e 24 na memória.
16. Valores de X e de Y são armazenados nos endereços 30 e 31. Escrever um programa que armazenará o maior dos dois valores no endereço 50, usando a linguagem de montador das Seções 1.13 a 1.15.
17. Dados três números diferentes, determinar se eles estão em ordem crescente ou em ordem decrescente. Desenhar um fluxograma para o problema e escrever um programa usando a linguagem de montador das Seções 1.13 a 1.15. Admitir que os números estejam armazenados nos locais 30, 31 e 32 na memória.
18. Um valor de Y é armazenado no local 55 e um valor de A, no local 57. Escrever um programa em linguagem de montador para armazenar AY^2 no local 30.
19. Escrever um programa para achar $AX^2 + BY + CZ^2$, com A no local 20, B no local 21, C no local 22, X no local 23, Y no local 24 e Z no local 25. Armazenar seu resultado no local 30.
20. Um valor de X é armazenado no endereço 40. Escrever um programa que armazenará X^8 no endereço 45, usando menos de 10 palavras de instrução.
21. Usando a linguagem de montador descrita nas Seções 1.13 a 1.15, escrever um programa que se desviará para o local 300 se o número armazenado no registro 25 na memória for maior do que o número armazenado no registro 26 e que se transferirá ou se desviará para o local 400 se o número no endereço 26 for igual a ou maior do que o número no endereço 25.
22. Escrever um programa que produzirá o valor X − Y ou X, qualquer que seja maior, tanto em linguagem de montador das Seções 1.13 a 1.15 quanto em linguagem de compilador na Seção 1.20. Armazenar este valor no local 300 para o programa de montador e atribuir a variável B a este valor para o programa escrito na linguagem do compilador. Para o programa do montador, admitir que X esteja no local 100 e Y, no 101.
23. Calcular o maior dos três números A, B e C e atribuir o maior dos números à variável X, usando a linguagem de montador na Seção 1.18 para escrever o programa.
24. Escrever um programa em PL/1 que atribuirá o valor 50 à variável B e o valor 60 à variável C.
25. Escrever um programa em PL/1 que atribuirá o valor 10 a A, 20 a B e 5 a C e que depois calculará o valor $(A \times B)^C$ e atribuirá este valor a uma variável D.
26. Em PL/1, qual será o valor de D se executarmos a declaração D = X + (Y**Z) quando X tem valor 5, Y tem valor 2 e Z tem valor 3?
27. Se o programa em PL/1 que foi usado para ilustrar o comando LOOP for passado na máquina com P tendo valor 4 mas com T, M e N como no programa no texto, qual será o valor de Y depois de o programa ser passado na máquina?
28. Suponhamos que o programa final em PL/1 no texto seja passado na máquina com valores de 4 para A, 3 para B, 2 para C e 2 para X. Que valores de C e de Y farão com que o programa seja impresso?
29. Usando a linguagem de compilador da Seção 1.20, escrever um programa que determinará qual dos números X − Y ou Y − X é positivo ou igual a 0 e qual deles imprimirá o valor deste número positivo ou zero, atribuindo a variável Z a este número.
30. Modificar o programa na Seção 1.15 de modo que ele irá somar todos os números ímpares desde 1 até 77.
31. Temos 30 números armazenados nos registros sucessivos em nossa memória, iniciando no local 300. Escrever um programa que converterá quaisquer números negativos no 30 em forma positiva. Isto significa escrever um programa que considerará os 30 números, converterá cada número em seu valor positivo sem alterar sua magnitude e irá rearmazená-lo no mesmo local. Usar a linguagem de montador.
32. Escrever um programa, usando a linguagem de montador das Seções 1.13 a 1.15, que

rearranjará cinco números armazenados nos endereços 300 a 304 de modo que eles estejam em ordem decrescente (por exemplo, 10, 3, 0, −5, −7).

Questões adicionais sobre PL/1

33. Numa seção sem parênteses de uma determinada declaração, a linguagem PL/1, como as linguagens Fortran e Algol, primeiro executa a exponenciação, ou elevação a uma potência, depois a multiplicação e a divisão e depois a adição e a subtração. Dentro desta hierarquia, as declarações são avaliadas da esquerda para a direita. Assim, a declaração A = ((B/C) − (D*E))/(C**D) pode ser escrita como A = (B/C − D*E) / (C**D). Os seguintes são outros quatro exemplos do modo pelo qual o compilador avalia declarações aritméticas:

(a) $A^B + C \cdot D + E/F$ pode ser escrita A**B + C*D + E/F.
(b) $(A \cdot B \cdot C)E^F + (G/H)$ pode ser escrita A*B*C*E**F + G/H.
(c) $\frac{A \cdot B \cdot C}{D \cdot E}$ deve ser escrita (A*B*C)/(D*E).
(d) $A \cdot B + (C/D) + EF$ é escrita A*B + C/D + E * F.

Achar o valor de X nas seguintes declarações se A = 5, B = 2, C = 20 e D = 15:

(a) X = A**B +C
(b) X = A * C ** B
(c) X = C + D * B + A
(d) X = C + D ** B/A + A

34. Usando a linguagem de compilador da Seção 1.20, admitir que temos lido os valores das variáveis X, Y e Z. Atribuir o maior valor destas variáveis à variável A, o segundo maior valor à variável B e o terceiro maior valor à variável C. Por exemplo, se X for lido como 29, Y como 31 e Z como 23, depois de o programa operar, desejaríamos os valores A = 33, B = 29 e C = 21.
35. Escrever as declarações em PL/1 que correspondam às seguintes fórmulas algébricas escritas em notação matemática convencional:

(a) $\frac{a^c}{b^d}$ *(b)* $a^2 + b + cd$ *(c)* $\left(\frac{a}{b}\right) + \left(\frac{c}{d}\right)^e$ *(d)* $(a \cdot b)^r + \frac{bd}{e}$

36. Em PL/1 é possível escrever declarações matemáticas com variáveis subscritas, que são chamadas variáveis de conjunto. Isto significa que, se tivermos um conjunto de variáveis X_1, X_2, X_3 e X_4, poderemos escrever estas variáveis como em PL/1, ou seja, X(1), X(2), X(3) e X(4). Observemos que os parênteses não indicam multiplicação, mas o fato de que a variável é uma variável de conjunto ou uma variável subscrita. Agora, se escrevermos num programa X(K), se isto indica X(1) ou X(2) ou X(3) ou X(4), por exemplo, depende do valor da variável K naquele instante particular de tempo. Quais são os valores de X em cada uma das seguintes declarações se A(1) = 3, A(2) = 4, A(3) = 5 e K = 1:

(a) X = A(1) + A(2)
(b) Y = A(2) ** A(1) + A(3)
(c) X = A(K) + A(2) * A(3)
(d) Y = A(K + 2) + A(K)

37. Há uma classe de declarações em PL/1 chamada *declarações de controle*. Estas declarações tornam possível iterar muitas vezes através de um dado conjunto de instruções, fornecendo facilidades de indexação durante estas iterações. A forma geral da mais importante iteração destas declarações, que é a declaração DO, é DO N K =P, Q. Aqui, N é o número ou localização da declaração de alguma declaração executável, K é uma simples variável e P e Q são ou constantes ou variáveis sem conjunto. Uma declaração DO típica pode, portanto, aparecer como DO 30 K = 1, 25. Esta declaração num programa fará com que as instruções ou declarações que seguem a declaração DO até a declaração 30 sejam repetidas, com a variável K igual a 1 até 25. Isto significa que o laço desde a declaração DO até a declaração 30 no programa é repetido 25 vezes, a primeira vez com a variável K igual a 1, a segunda vez com a variável K igual a 2, a

terceira vez com a variável K igual a 3 etc. até K = 25. Neste instante, o programa passará da declaração 30 à declaração seguinte. Se, portanto, escrevermos

```
      X = 2
      B(1) = 5
      B(2) = 6
      B(3) = 7
      B(4) = 8
      DO 19 K = 1, 4
      A(K) = B(K) + 6
19    X = X ** 2
```

Depois de este programa ter sido passado, A(1) será igual a 11, A(2) será igual a 12, A(3) será igual a 13 e A(4) será igual a 14, enquanto X = 65.536.

Usando a declaração DO em PL/1, escrever um programa que rearranjará um conjunto de 20 números que são armazenados em registros sucessivos. Estas variáveis são A(1) até A(20), onde cada valor é suposto positivo. Atribuir estes valores a um conjunto de variáveis B(1) até B(20) de modo que B(1) seja igual ao menor dos valores de A, B(2) seja igual ao segundo menor dos valores de A e, finalmente, B(20) seja igual ao maior valor das variáveis A.

38. Usando a declaração DO explicada no Exercício 37, escrever um programa que examinará e formará uma média dos valores de um conjunto de variáveis A(1) até A(10), onde A(1) até A(10) são supostos já estarem armazenados na memória do computador quando começa o programa. Chamar esta média dos 10 valores de valor *médio* destes valores e dar este valor à variável denominada M quando o programa tiver operado.

2

SISTEMAS DE NUMERAÇÃO

Foi a Índia que nos deu o método engenhoso de exprimir todos os números por meio de dez símbolos, cada símbolo recebendo um valor de posição bem como um valor absoluto; uma idéia consistente e importante que agora nos parece tão simples a ponto de ignorarmos seu verdadeiro mérito.

MARQUÊS DE LAPLACE

Como um matemático, Laplace teve oportunidade de apreciar muito bem o sistema de numeração decimal. Ele estava a par dos séculos de esforço mental e de simples casualidade que tinham decorrido para o desenvolvimento do sistema de numeração que usamos, e estava em posição de apreciar suas vantagens. Nosso atual sistema de numeração proporciona aos modernos matemáticos e cientistas uma grande vantagem em relação aos das civilizações anteriores e constitui um fator importante em nosso rápido progresso.

Já que as mãos representam as ferramentas mais convenientes que a natureza concedeu, os seres humanos sempre tenderam a usá-las na contagem. É ao mesmo tempo natural e auspicioso o fato de que nosso sistema de numeração seja baseado no número de dígitos que possuímos. No entanto, foi absolutamente algum tempo depois de aprendermos a contar que tentamos representar os números graficamente. Os primeiros algarismos que foram achados consistem de marcas verticais ou horizontais. Nosso 1 é um exemplo desta espécie de símbolo, e é interessante notar que o símbolo do 2 consiste de duas marcas horizontais com uma linha de conexão, e o 3 consiste de três linhas horizontais com conexões. Os algarismos romanos constituem bons exemplos de linhas usadas como base para os algarismos.

O sistema decimal de contagem tem sido tão amplamente adotado por toda a nossa atual civilização que dificilmente consideramos as possibilidades de outros sistemas de numeração. Apesar disso, não é razoável esperar que um sistema baseado no número de dedos que possuímos seja o mais eficiente sistema de numeração para a construção de máquinas. O fato é que um sistema pouco utilizado, porém muito simples, o sistema de numeração binário, é comprovadamente o mais natural e eficiente sistema para uso em máquinas.

2.1 O SISTEMA DECIMAL

Nosso atual sistema de números possui 10 símbolos separados: 0, 1, 2, 3, ..., 9, que são chamados algarismos arábicos. Seríamos forçados a parar em 9 ou a inventar outros símbolos se não fosse o uso da *notação posicional*. Um exemplo dos primeiros tipos de notação pode ser achado nos algarismos romanos, que são essencialmente aditivos: III = I + I + I, XXV = X + X + V. Novos símbolos (X, C, M etc.) foram usados quando os números aumentavam de valor. Assim, V em vez de IIIII = 5. A única importância da posição nos algarismos romanos reside em saber se um determinado símbolo precede ou se segue a um outro símbolo (IV = 4, enquanto que VI = 6). A deselegância deste sistema pode ser facilmente vista se tentarmos multiplicar XII por XIV. O cálculo com emprego de algarismos romanos era tão difícil que os primeiros matemáticos foram forçados a efetuar operações aritméticas

quase que inteiramente em ábacos, ou quadros de contagem, que traduzem seus resultados novamente em forma de números romanos. As computações a papel e lápis são inacreditavelmente intrincadas e difíceis em tais sistemas. De fato, a capacidade de efetuar operações tais como adição e multiplicação foi considerada uma grande proeza nas primitivas civilizações.

Agora podem ser vistas a grande beleza e a simplicidade do nosso sistema de numeração. É necessário aprender apenas os 10 algarismos básicos e o *sistema de notação posicional* a fim de contar cálculo desejado. Depois de memorizar as tabuadas de adição e de multiplicação e de aprender algumas regras simples, é possível efetuar todas as operações aritméticas. Observemos a simplicidade da multiplicação de 12 × 14 usando o presente sistema:

$$\begin{array}{r} 14 \\ \times\ 12 \\ \hline 28 \\ 14 \\ \hline 168 \end{array}$$

O significado real do número 168 pode ser visto mais claramente se observarmos que ele é pronunciado como "um cento e sessenta e oito". Basicamente, o número é uma contração de (1 × 100) + (6 × 10) + 8. O ponto importante é que o valor de cada dígito é determinado por sua posição. Por exemplo, o 2 em 2.000 tem um valor diferente do 2 em 20. Mostramos isto verbalmente dizendo "dois milhares" e "vinte". Diferentes representações verbais foram inventadas para números de 10 a 20 (onze, doze, ...), mas, de 20 para cima, dividimos apenas em potências de 10 (centenas, milhares, milhões, bilhões). Os números escritos, no entanto, são sempre contraídos e somente os 10 algarismos básicos são usados independente do tamanho do inteiro escrito. A regra geral para representação de números no sistema decimal, usando notação posicional, é como segue: $a_1 10^{n-1} + a_2 10^{n-2} + \ldots + a_n$ é expressa como $a_1 a_2 \ldots a_n$, onde n é o número de dígitos à esquerda do ponto decimal.

A *base*, ou *raiz*, de um sistema de numeração é definida como o número de dígitos diferentes que podem ocorrer em cada posição no sistema de numeração. O sistema de numeração decimal tem uma base, ou raiz, igual a 10. Isto significa que o sistema tem 10 dígitos diferentes (0, 1, 2, . . ., 9), qualquer um dos quais pode ser usado em cada posição num número. A História registra o uso de diversos outros sistemas de numeração. O sistema quinário, que tem como base 5, prevaleceu entre os esquimós e os índios da América do Norte. Exemplos do sistema duodecimal (base 12) podem ser vistos em relógios, polegadas e pés, e em dúzias ou grosas.

2.2 O SISTEMA BINÁRIO

Um matemático alemão do século dezessete, Gottfried Wilhelm von Leibniz, foi um advogado do sistema binário de numeração que tem por base 2, usando apenas os símbolos 0 e 1. Se parece estranho que um eminente matemático advogue tal sistema de numeração muito simples, deve ser notado que ele foi também um filósofo. As razões de Leibniz para advogar o sistema binário parecem ter sido místicas. Ele sentiu que havia grande beleza na analogia entre zero, representando o vazio, e um, representando a Divindade.

Independente de quão boas fossem as razões de Leibniz para advogá-lo, o sistema binário tornou-se muito popular na última década. Os computadores digitais da atualidade são construídos para operar em sistemas binários ou de números codificados em binário, e as presentes indicações são que as futuras máquinas também serão construídas para operar nestes sistemas.

Os elementos básicos nos primitivos computadores foram os relés e as chaves. A operação de uma chave, ou de um relé, pode ser vista como de natureza essencialmente binária. Isto significa que a chave ou está ligada (1) ou desligada (0). Os principais elementos de circuito nos mais modernos computadores são os transistores similares aos usados em aparelhos de rádio e de televisão. O desejo da confiabilidade levou os projetistas a usar estes dispositivos de modo que eles estivessem essencialmente num de dois estados, plenamente conduzindo ou não conduzindo. Uma simples analogia pode ser feita entre este tipo de circuito e uma lâmpada elétrica. Em qualquer instante de tempo determinado, a lâmpada (ou o transistor) está ou no estado *on* (conduzindo) ou no estado *off* (não conduzindo). Mesmo após uma válvula eletrônica ficar fraca, geralmente é muito fácil dizer se

ela está conduzindo ou cortada (não conduzindo). O mesmo pode ser visto nos receptores de rádio. Quando um desses receptores envelhece, geralmente o volume diminui, e procuramos compensar a perda girando o controle de volume para o máximo. Mesmo quando o receptor de rádio se torna muito fraco, no entanto, ainda é possível dizer com facilidade se ele está ligado ou desligado.

Em virtude do grande número de partes eletrônicas usadas nos computadores, é altamente desejável utilizá-los de tal maneira que ligeiras variações em suas características não irão afetar seus desempenhos. A melhor maneira de se conseguir isto é a utilização de circuitos que são basicamente *biestáveis* (que têm dois estados possíveis).

2.3 CONTAGEM NO SISTEMA BINÁRIO

O mesmo tipo de notação posicional é usada nos sistemas de numeração binário e no sistema decimal. O Quadro 2.1 lista os primeiros 20 números binários.

Enquanto o mesmo sistema de notação posicional é usado, o sistema decimal usa potências de 10 e o sistema binário potências de 2. Como foi explicado anteriormente, o número 125 significa realmente $(1 \times 10^2) + (2 \times 10^1) + (5 \times 10^0)$. No sistema binário, o mesmo número (125) é representado por 1111101, significando $(1 \times 2^6) + (1 \times 2^5) + (1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$.

Para exprimir o valor de um número binário, portanto, $a_1 2^{n-1} + a_2 2^{n-2} + \ldots + a_n$ é representado por $a_1 a_2 \ldots a_n$, onde a é 1 ou 0 e n é o número de dígitos à esquerda do ponto de base (raiz) binária.

Os exemplos seguintes ilustram a conversão de números binários para o sistema decimal:

$$\begin{aligned} 101 &= (1 \times 2^{3-1}) + (0 \times 2^{3-2}) + (1 \times 2^{3-3}) \\ &= (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) \\ &= 4 + 1 = 5 \end{aligned}$$

$$\begin{aligned} 1001 &= (1 \times 2^{4-1}) + (0 \times 2^{4-2}) + (0 \times 2^{4-3}) + (1 \times 2^{4-4}) \\ &= (1 \times 2^3) + (0 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) \\ &= 8 + 1 = 9 \end{aligned}$$

$$\begin{aligned} 11{,}011 &= (1 \times 2^{2-1}) + (1 \times 2^{2-2}) + (0 \times 2^{2-3}) + (1 \times 2^{2-4}) + (1 \times 2^{2-5}) \\ &= (1 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2}) + (1 \times 2^{-3}) \\ &= 2 + 1 + \tfrac{1}{4} + \tfrac{1}{8} \\ &= 3\tfrac{3}{8} \end{aligned}$$

Observe que os números fracionários são formados pelo mesmo princípio geral que no sistema decimal. Quando

$$0{,}123 = (1 \times 10^{-1}) + (2 \times 10^{-2}) + (3 \times 10^{-3})$$

QUADRO 2.1

Decimal		Binário	Decimal		Binário
1	=	1	11	=	1011
2	=	10	12	=	1100
3	=	11	13	=	1101
4	=	100	14	=	1110
5	=	101	15	=	1111
6	=	110	16	=	10000
7	=	111	17	=	10001
8	=	1000	18	=	10010
9	=	1001	19	=	10011
10	=	1010	20	=	10100

no sistema decimal,

$$0{,}101 = (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3})$$

no sistema binário.

2.4 ADIÇÃO E SUBTRAÇÃO EM BINÁRIO

A adição em binário é efetuada exatamente como a adição em decimal. Realmente, a aritmética binária é muito mais simples de se aprender. A tabela completa para a adição em binário é como segue:

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 0$ mais um transporte de 1

"Os transportes" são efetuados de maneira idêntica à da aritmética decimal. Uma vez que 1 é o maior dígito no sistema binário, qualquer soma superior a 1 exige que seja transportado um dígito. Por exemplo, 100 mais 100 em binário requer a adição dos dois números 1 na terceira posição à esquerda, com um transporte. Uma vez que 1 + 1 = 0 mais um transporte de 1, a soma de 100 e 100 é 1.000. Temos, a seguir, mais três exemplos da adição em binário:

DECIMAL	BINÁRIO	DECIMAL	BINÁRIO	DECIMAL	BINÁRIO
5	101	15	1111	$3\frac{1}{4}$	11,01
6	110	20	10100	$5\frac{3}{4}$	101,11
11	1011	35	100011	9	1001,00

A subtração é a operação inversa da adição. Para subtrair, é necessário estabelecer um procedimento para diminuir um dígito menor do dígito maior. O único caso em que isto ocorre usando números binários é quando 0 é subtraído de 1. O resto é 1, mas é necessário tomar 1 por empréstimo da coluna seguinte à esquerda. Esta é a tabela de subtração em binário:

$0 - 0 = 0$
$1 - 0 = 1$
$1 - 1 = 0$
$0 - 1 = 1$ com um empréstimo de 1

Alguns exemplos tornam mais claro o procedimento da subtração em binário.

DECIMAL	BINÁRIO	DECIMAL	BINÁRIO	DECIMAL	BINÁRIO
9	1001	16	10000	$6\frac{1}{4}$	110,01
−5	−101	−3	−11	$-4\frac{1}{2}$	−100,1
4	100	13	1101	$1\frac{3}{4}$	1,11

2.5 MULTIPLICAÇÃO E DIVISÃO EM BINÁRIO

A tabela para multiplicação em binário é muito pequena, com apenas quatro entradas em vez das 100 necessárias na multiplicação em decimal. A tabela para a multiplicação em binário é

$0 \times 0 = 0$
$1 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 1 = 1$

Os três exemplos seguintes de multiplicação em binário ilustram a simplicidade de cada operação. É necessário apenas copiar o multiplicando se o dígito no multiplicador for 1 e copiar todos os zeros se o dígito no multiplicador for um 0. É evidente a facilidade com que cada etapa da operação é efetuada.

DECIMAL	BINÁRIO	DECIMAL	BINÁRIO	DECIMAL	BINÁRIO
12	1100	102	1100110	1,25	1,01
×10	×1010	×8	×1000	×2,5	×10,1
120	0000	816	1100110000	625	101
	1100			250	1010
	0000			3,125	11,001
	1100				
	1111000				

A divisão em binário é, novamente, muito simples. Como no sistema decimal (ou em qualquer outro), a divisão por zero não tem significado. A tabela completa é

$$0 \div 1 = 0$$
$$1 \div 1 = 1$$

Aqui estão dois exemplos de divisão:

DECIMAL

```
   5
5 ) 25
```

BINÁRIO

```
    101
101 ) 11001
      101
       101
       101
```

DECIMAL

```
    2,416···
12 ) 29,0000
     24
     50
     48
     20
     12
     80
     72
      8
```

BINÁRIO

```
      10,011010101···
1100 ) 11101,00
       1100
       10100
        1100
        10000
         1100
         10000
          1100
          .....
```

Convertendo-se de binário a decimal o quociente obtido no segundo exemplo, prossegui-

ríamos como segue:

$$
\begin{aligned}
10{,}0110101 = 1 \times 2^{1} &= 2{,}0 \\
0 \times 2^{0} &= 0{,}0 \\
0 \times 2^{-1} &= 0{,}0 \\
1 \times 2^{-2} &= 0{,}25 \\
1 \times 2^{-3} &= 0{,}125 \\
0 \times 2^{-4} &= 0{,}0 \\
1 \times 2^{-5} &= 0{,}03125 \\
0 \times 2^{-6} &= 0{,}0 \\
1 \times 2^{-7} &= 0{,}0078125 \\
0 \times 2^{-8} &= 0{,}0 \\
1 \times 2^{-9} &= 0{,}001953125 \\
&\overline{\quad 2{,}416015625}
\end{aligned}
$$

Portanto o binário 10,0110101 é aproximadamente igual ao decimal 2,416.

2.6 CONVERSÃO DE NÚMEROS DECIMAIS EM BINÁRIOS

Existem vários métodos para converter um número decimal em número binário. O primeiro método é o mais óbvio e consiste simplesmente em subtrair todas as potências de 2 que podem ser subtraídas do número decimal até que não haja resto. A mais alta potência de 2 é subtraída primeiro, depois a segunda mais alta etc. Para converter o inteiro decimal 25 ao sistema de numeração binário, acha-se inicialmente a mais alta potência de 2 que pode ser subtraída de 25. Esta é $2^4 = 16$. Então, $25 - 16 = 9$. A mais alta potência de 2 que pode ser subtraída de 9 é 2^3, ou 8. O resto após a subtração é 1, ou 2^0. A representação binária de 25 é, portanto, 11001.

Este é um método laborioso de conversão de números. É conveniente usá-lo em números pequenos, quando ele puder ser efetuado mentalmente, porém é menos usado em números maiores. Em vez disso, o número decimal é repetidamente dividido por 2, e o resto depois de cada divisão é usado para indicar os coeficientes do número binário a ser formado. Observemos que o número binário derivado é escrito de baixo para cima.

$$
\begin{aligned}
125 \div 2 &= 62 + \text{resto } 1 \\
62 \div 2 &= 31 + \text{resto } 0 \\
31 \div 2 &= 15 + \text{resto } 1 \\
15 \div 2 &= 7 + \text{resto } 1 \\
7 \div 2 &= 3 + \text{resto } 1 \\
3 \div 2 &= 1 + \text{resto } 1 \\
1 \div 2 &= 0 + \text{resto } 1
\end{aligned}
$$

A representação binária de 125 é, portanto, 1111101. Verificando este resultado, obtemos

$$
\begin{aligned}
1 \times 2^{6} &= 64 \\
1 \times 2^{5} &= 32 \\
1 \times 2^{4} &= 16 \\
1 \times 2^{3} &= 8 \\
1 \times 2^{2} &= 4 \\
0 \times 2^{1} &= 0 \\
1 \times 2^{0} &= 1 \\
&\overline{\quad 125}
\end{aligned}
$$

Este método não funcionará para números mistos. Se métodos similares são usados, é necessário primeiro decompor o número em sua parte inteira e sua parte fracionária. Assim, 102,247 seria decomposto em 102 e 0,247, e a representação binária de cada parte achada e, depois, as duas partes adicionadas.

A conversão de frações decimais em frações binárias pode ser efetuada usando-se várias técnicas. Novamente, o método mais óbvio consiste em subtrair a mais alta potência negativa de 2 que pode ser subtraída da fração decimal. A próxima potência negativa de 2 mais alta é então subtraída do resto da primeira subtração, e este processo continua até que não haja mais resto, ou até a precisão desejada.

$$0{,}875 - (1 \times 2^{-1}) = 0{,}875 - 0{,}5 = 0{,}375$$
$$0{,}375 - (1 \times 2^{-2}) = 0{,}375 - 0{,}25 = 0{,}125$$
$$0{,}125 - (1 \times 2^{-3}) = 0{,}125 - 0{,}125 = 0$$

Portanto, o decimal 0,875 é representado pelo binário 0,111. Um método muito mais simples para frações mais longas consiste em "dobrar" repetidamente a fração decimal. Se aparecer um 1 à esquerda da vírgula depois da multiplicação por 2 ser efetuada, um 1 é acrescentado à direita da fração binária que está sendo formada. Se depois de uma multiplicação por 2 permanecer um 0 à esquerda da vírgula do número decimal, um 0 é acrescentado à direita do número binário. O seguinte exemplo ilustra o uso desta técnica de conversão do decimal 0,4375 para o sistema binário:

	REPRESENTAÇÃO BINÁRIA
$2 \times 0{,}4375 = 0{,}8750$	0,0
$2 \times 0{,}875 = 1{,}750$	0,01
$2 \times 0{,}75 = 1{,}50$	0,011
$2 \times 0{,}5 = 1{,}0$	0,0111

A representação binária de 0,4375 é, portanto, 0,0111.

2.7 NÚMEROS NEGATIVOS

Uma conversão padronizada que foi adotada para escrever números negativos consiste na colocação de um "símbolo decimal" antes de um número que seja negativo. Por exemplo, o número negativo 39 é escrito −39. Se −39 deve ser adicionado a +70, escrevemos

$$+70 + (-39) = 31$$

Quando o número negativo for subtraído de um número positivo, escrevemos $+70 - (-39) = +70 + 39 = 109$. As regras do manuseio de números negativos são bem conhecidas e não serão repetidas aqui, mas, uma vez que os números negativos constituem uma parte importante de nosso sistema de numeração, as técnicas usadas para representar números negativos nas máquinas digitais serão descritas.

Nas máquinas binárias, os números são representados por um conjunto de dispositivos de armazenamento biestável, cada um dos quais representa um dígito binário. Por exemplo, dado um conjunto de cinco chaves, qualquer número de 00000 a 11111 pode ser representado pelas chaves, se definirmos uma chave com seus contatos fechados representando um 1 e uma chave com contatos abertos representando um 0. Se desejarmos aumentar a faixa total de números que podemos representar de modo que ela possa incluir os números negativos desde 00000 até −11111, será necessário um outro bit (ou chave). A seguir, tratamos este bit como um *bit de sinal* e o colocamos antes da magnitude do número a ser representado.

Geralmente, a convenção adotada é a seguinte: quando o bit de sinal for um 0, o número representado é positivo, e, quando o bit de sinal for um 1, o número é negativo. Se a situação anterior, onde cinco chaves são usadas para armazenar a magnitude de um número, for estendida de modo que ambos, positivos e negativos, possam ser armazenados, uma sexta chave será necessária. Quando os contatos desta chave forem abertos, o número será um número positivo igual à magnitude do número armazenado nas outras cinco chaves, e, se a chave do bit de sinal estiver fechada, o número representado pelas seis chaves será um número negativo com uma magnitude determinada pelas outras cinco chaves. Um exemplo é mostrado na Fig. 2.1.

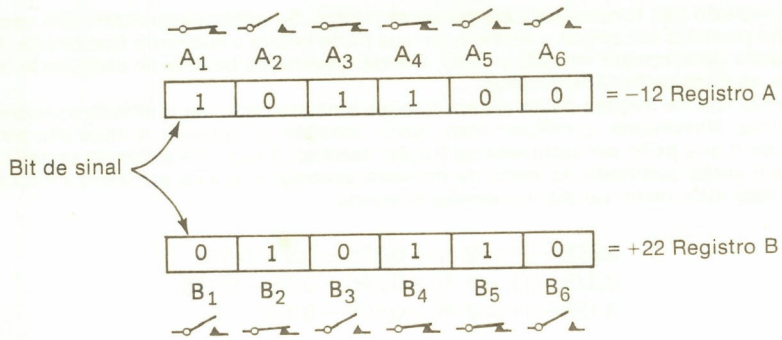


Fig. 2.1 Exemplo de representação de número negativo.

Os conjuntos de dispositivos de armazenamento que representam um número ou são manipulados como uma entidade são denominados *registros*, e a eles são atribuídos nomes tais como registro A, registro B, registro C etc. Podemos então escrever que o registro A contém −12 e o registro B contém +22. Na escrita de um número com sinal em forma binária, o bit de sinal é geralmente colocado afastado da magnitude do número por meio de um período, de modo que 0,0111 representa 0111 positivo, ou o decimal 7 positivo, e 1,0111 representa −0111, ou o decimal 7 negativo.

A confusão que surge da separação do bit de sinal da magnitude do número com um período que também pode ser um ponto de base (binário ou decimal) é compensada quando os números negativos forem manipulados no que é conhecido por "forma complementada", conforme veremos na seção seguinte. Deve ser enfatizado que um símbolo diferente do período pode ser usado para separar os bits de sinal e de magnitude, tais como um hífen ou uma estrela. Então, −1011 (decimal 11 negativo) pode ser escrito 1-1011 ou 1*1011, e + 1100 como 0-1100 ou 0*1100. Não há nenhuma objeção particular a estas convenções, mas o uso de um período para separar o bit de sinal de magnitude é absolutamente bem estabelecido e aparecerá neste texto.

2.8 O USO DE COMPLEMENTOS PARA REPRESENTAR NÚMEROS NEGATIVOS

Foi descrita a convenção de usar um bit de sinal para indicar se um número armazenado é negativo ou positivo. A magnitude do número armazenado, no entanto, nem sempre é representada em forma normal, mas muito freqüentemente os números negativos são armazenados em "forma complementada". Usando-se esta técnica, uma máquina pode ser fabricada tanto para somar quanto para subtrair, utilizando apenas um conjunto de circuitos de somar. A técnica real envolvida será descrita no Cap. 6.

Há dois tipos básicos de complementos que são úteis nos sistemas binários e também nos decimais. No sistema decimal, os dois tipos são denominados *complemento* de 10 e *complemento* de 9.

O complemento de 10 de qualquer número pode ser formado subtraindo-se cada dígito do número 9 e em seguida adicionando-se 1 ao dígito menos significativo do número assim formado. Por exemplo, o complemento de 10 de 87 é 13 e o complemento de 10 de 23 é 77. A subtração pode ser efetuada simplesmente adicionando-se o complemento de 10 do subtraendo ao minuendo e abandonando o transporte final, se houver. Por exemplo:

SUBTRAÇÃO NORMAL	SUBTRAÇÃO COM COMPLEMENTO DE 10
89	89 89
−23	−23 = +77
66	166
	↳ o transporte é abandonado
98	98 98
−87	−87 = +13
11	111
	↳ o transporte é abandonado

O complemento de 9 de um número decimal é formado subtraindo-se de 9 cada dígito do número. Por exemplo, o complemento de 9 do número 23 é 76 e o complemento de 9 de 87 é 12. Quando a subtração é efetuada usando o complemento de 9, o complemento do subtraendo é adicionado como antes, mas qualquer transporte gerado deve ser adicionado ao dígito mais à direita do resultado.

SUBTRAÇÃO NORMAL	SUBTRAÇÃO COM COMPLEMENTO DE 9
$\begin{array}{r} 89 \\ -23 \\ \hline 66 \end{array}$	$\begin{array}{r} 89 \\ -23 \\ \hline \end{array} = \begin{array}{r} 89 \\ 76 \\ \hline 165 \\ 1 \\ \hline 66 \end{array}$
$\begin{array}{r} 98 \\ -87 \\ \hline 11 \end{array}$	$\begin{array}{r} 98 \\ -87 \\ \hline \end{array} = \begin{array}{r} 98 \\ 12 \\ \hline 110 \\ 1 \\ \hline 11 \end{array}$

As regras para manuseio de sinais durante o processo da subtração e para a manipulação de todas as combinações de números positivos e negativos será explicada no Cap. 6. Se, de início, isto parecer uma técnica difícil de manejar, deve ser observado que a maioria das máquinas que estão sendo fabricadas atualmente subtrai usando um número complementado.

2.9 COMPLEMENTOS EM OUTROS SISTEMAS DE NUMERAÇÃO

Há dois tipos de complementos para cada sistema de numeração. Uma vez que somente as máquinas binárias e decimais codificadas em binário estão sendo agora construídas em quantidade, apenas estes sistemas de numeração serão explicados em detalhes. Os dois tipos de complementos e as regras para obtê-los são:

1. *Complemento verdadeiro.* Este é formado subtraindo-se cada dígito do número da base-menos-um do sistema de numeração e em seguida adicionando-se 1 ao dígito menos significativo do número formado. O complemento verdadeiro de um número no sistema decimal é denominado complemento de 10 e, no sistema binário, é denominado complemento de 2.
2. *Complemento de base-menos-um.* A base-menos-um é 9 para o sistema decimal e 1 para o sistema binário. O complemento em cada sistema é formado subtraindo-se cada dígito do número da base-menos-um. Por exemplo, o complemento de base-menos-um do decimal 72 é 27.

2.10 COMPLEMENTOS DE NÚMERO BINÁRIO

De acordo com a regra na seção precedente, o complemento de 2 de um número binário é formado simplesmente subtraindo-se cada dígito (bit) do número da base-menos-um e acrescentando um 1 ao bit menos significativo. Uma vez que a base no sistema de numeração binário é 2, cada bit do número binário é subtraído de 1. A aplicação desta regra é realmente muito simples; cada 1 no número é mudado para um 0 e cada 0 para um 1. Um 1 é então adicionado ao bit menos significativo do número formado. Por exemplo, o complemento de 2 de 10110 é 01010 e o complemento de 2 de 11010 é 00110. A subtração usando o sistema de complemento de 2 envolve a formação do complemento de 2 do subtraendo e depois a adição deste "complemento verdadeiro" ao minuendo. Por exemplo:

$$\begin{array}{r} 11011 \\ -10100 \\ \hline 00111 \end{array} = \begin{array}{r} 11011 \\ +01100 \\ \hline 1\ 00111 \end{array} \quad \text{e} \quad \begin{array}{r} 11100 \\ 00100 \\ \hline 11000 \end{array} = \begin{array}{r} 11100 \\ 11100 \\ \hline 1\ 11000 \end{array}$$

o transporte é abandonado — abandonado

A subtração usando o sistema de complemento de 1 é também direta. O complemento de 1 de um número binário é formado mudando-se cada 1 no número para um 0 e cada 0 no número para um 1. Por exemplo, o complemento de 1 de 10111 é 01000 e o complemento de 1 de 11000 é 00111.

Quando a subtração é efetuada no sistema de complemento de 1, qualquer realimentação ou retorno de transporte é adicionado ao bit menos significativo. Por exemplo:

```
 11001          11001          11110          11110
-10110   =    +01001    e     -01101   =     10010
-------        -------        -------        -------
 00011         10010          10001         -10000
       \_____________1                \_____________1
               00011                          10001
```

2.11 REPRESENTAÇÃO DE NÚMERO DECIMAL CODIFICADO EM BINÁRIO

Uma vez que a maioria dos elementos do circuito eletrônico usados para construir computadores digitais é inerentemente binária quanto à operação, o sistema de numeração binário é o sistema de numeração mais natural para um computador. Além disso, os computadores construídos utilizando o sistema de numeração binário requerem uma menor quantidade de conjuntos de circuito e são, portanto, mais eficientes do que as máquinas que operam em outros sistemas de numeração. Por outro lado, o sistema decimal foi usado durante um longo tempo, e há uma reação natural para efetuar cálculos num sistema de numeração binário. E também, uma vez que os cheques, as contas, as taxas de impostos, os preços etc. são todos escritos no sistema decimal, os valores da maioria das grandezas devem ser convertidos de decimal a binário antes que as computações possam começar. Por estas e outras razões, a maioria das primeiras máquinas operava em sistemas de numeração decimal codificado em binário. Em tais sistemas, um grupo codificado de bits binários é usado para representar cada um dos 10 dígitos decimais. Por exemplo, um código óbvio e natural é um simples "código binário ponderado", conforme mostrado no Quadro 2.2.

Este é conhecido como um código *8, 4, 2, 1 de decimal codificado em binário* ou simplesmente código BCD.

QUADRO 2.2

Código binário	Dígito decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

Observe que 4 bits binários são necessários para cada dígito decimal, e a cada bit é atribuído um peso; por exemplo, o bit mais à direita tem um peso 1 e o bit mais à esquerda em cada grupo de código tem peso 8. Adicionando os pesos das posições em que aparecem os 1, o dígito decimal representado por um grupo de código pode ser obtido. Isso é um tanto antieconômico já que $2^4 = 16$, e, portanto, os 4 bits podiam realmente representar 15 valores diferentes; mas a menor escolha seguinte, 3 bits, dá somente 2^3, ou 8, valores, que são insuficientes. Se o número decimal 214 deve ser representado neste tipo de código, são necessários 12 bits binários como segue: 00 10 0001 0100. Para representar o número decimal 1246, são necessários 16 bits: 0001 0010 0100 0110.

Este código é muito útil e tem sido muito usado. Uma dificuldade relativa a este código,

QUADRO 2.3

Código excesso de 3	
Decimal	Código binário
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

no entanto, reside na formação dos complementos dos números neste sistema. É prática comum efetuar a subtração num computador adicionando o complemento do subtraendo; no entanto, quando o sistema BCD 8, 4, 2, 1 é utilizado, o complemento mais natural do número armazenado não é útil porque a maneira mais direta de um computador complementar um número consiste simplesmente em mudar cada 0 para um 1 e cada 1 para um 0. No entanto, o complemento natural de 0010 (decimal 2) é 1101, que representa 13, e não um caráter BCD aceitável neste sistema. Para contornar esta dificuldade, diversos outros códigos foram usados. Um dos primeiros, um código que foi empregado nas primeiras máquinas Mark construídas em Harvard e tem sido muito usado desde então, é conhecido como código *excesso de 3* e é formado pela adição de 3 ao número decimal e depois formando-se o número codificado em binário no código binário ponderado normal. Por exemplo, para formar a representação em excesso de 3 do número 4, primeiro, o 3 é adicionado produzindo 7 e, depois, o BCD "normal" é usado, que é 0111. Portanto, 0111 é o código excesso de 3 para o dígito decimal 4. O Quadro 2.3 mostra todos os 10 dígitos decimais e o código para cada um.

Complementando cada dígito do código binário que representa um dígito decimal, o complemento de 9 daquele dígito pode ser formado. Por exemplo, o complemento de 0100 (decimal 1) é 1011, que representa o decimal 8.

O número decimal 243 codificado no sistema excesso de 3 seria 0101 0111 0110 e o número decimal 347 seria 0110 0111 1010. O complemento de 9 de 243 é o decimal 756, ou o binário 1010 1000 1001. O Quadro 2.4 lista a representação em código excesso de 3 para

QUADRO 2.4

Dígito decimal	Código excesso de 3	Complemento de 9
0	0011	1100
1	0100	1011
2	0101	1010
3	0110	1001
4	0111	1000
5	1000	0111
6	1001	0110
7	1010	0101
8	1011	0100
9	1100	0011

QUADRO 2.5

Código 2, 4, 2, 1				
	Binário codificado			
	Peso de bit			
Decimal	2	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	0	1
8	1	1	1	0
9	1	1	1	1

cada um dos 10 dígitos decimais junto com os complementos de 9 listados abaixo. Pode ser visto que o complemento de 9 de cada grupo de código pode ser formado mudando-se cada 0 para um 1 e cada 1 para um 0 no grupo de código.

O código excesso de 3 não é um código ponderado, no entanto, porque não podem ser atribuídos pesos aos bits de modo que sua soma iguale os dígitos decimais representados.

Um código ponderado em que o complemento de 9 possa ser formado complementando cada dígito binário é o código 2, 4, 2, 1 (ver Quadro 2.5). Se cada bit de um grupo de código for complementado, o complemento de 9 do dígito decimal representado está formado. Por exemplo, 0010 (decimal 2) complementado é 1101 (decimal 7), e 1011 (decimal 5) complementado é 0100 (decimal 4). Este código é amplamente usado em instrumentos e nas calculadoras eletrônicas.

A seguinte convenção é geralmente adotada para distinguir binário de decimal. Um número binário é identificado por um índice 2 colocado no fim do número (00110_2), e um número decimal é identificado pelo índice 10 (por exemplo, o decimal 948 pode ser escrito 948_{10}). Podemos então escrever 0111_2 como 7_{10}. Esta convenção será usada sempre que necessário.

2.12 SISTEMAS DE NUMERAÇÃO OCTAL E HEXADECIMAL

Há dois outros sistemas de numeração que são muito úteis na indústria de computadores: o sistema de numeração octal e o sistema de numeração hexadecimal.

QUADRO 2.6

Octal	Decimal	Octal	Decimal
0	0	11	9
1	1	12	10
2	2	13	11
3	3	14	12
4	4	15	13
5	5	16	14
6	6	17	15
7	7	20	16
10	8	21	17

QUADRO 2.7

Três dígitos binários	Dígito octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

O sistema de numeração octal tem uma base, ou raiz, igual a 8; oito símbolos diferentes são usados para representar números. Estes são comumente 0, 1, 2, 3, 4, 5, 6 e 7. Usando notação posicional, os primeiros poucos números octais e seus equivalentes decimais são como indicado no Quadro 2.6.

Para converter um número octal em um número decimal, usamos a mesma espécie de polinômio que foi usada no caso binário, exceto que agora temos uma base 8 em vez de base 2. Portanto, 1213 em octal é $(1 \times 8^3) + (2 \times 8^2) + (1 \times 8^1) + (3 \times 8^0) = 512 + 128 + 8 + 3 = 651$ em decimal. Além disso, 1.123 em octal é $(1 \times 8^0) + (1 \times 8^{-1}) + (2 - 8^{-2}) + (3 \times 8^{-3})$, ou $1 + {}^1/_8 + {}^2/_{64} + {}^3/_{512} = 1{}^{83}/_{512}$ em decimal.

Há um simples artifício para a conversão de um número binário em um número octal. Basta grupar os dígitos binários em grupos de três, começando no ponto octal, e ler cada conjunto de três dígitos binários de acordo com o Quadro 2.7.

Convertamos o número binário 011101. Primeiro, separamo-lo em grupos de 3 dígitos (assim, 011 101) e, depois, convertendo cada grupo de três dígitos binários, obtemos 35 em octal. Portanto, o binário 011101 = octal 35. Aqui estão outros exemplos:

$$111110111_2 = 767_8$$
$$110110101_2 = 665_8$$
$$11011_2 = 33_8$$
$$1001_2 = 11_8$$
$$10101{,}11_2 = 25{,}6_8$$
$$1100{,}111_2 = 14{,}7_8$$
$$1011{,}1111_2 = 13{,}74_8$$

A conversão de decimal em octal pode ser efetuada dividindo-se repetidamente o número decimal por 8 e usando-se cada resto como um dígito no número octal que está sendo formado. Por exemplo, para converter 200_{10} numa representação octal, dividimos como segue:

$200 \div 8 = 25$ o resto é 0
$25 \div 8 = 3$ o resto é 1
$3 \div 8 = 0$ o resto é 3

Portanto, $200_{10} = 310_8$.

Observemos que quando o número a ser dividido é menor que 8, usamos 0 como quociente e o número como resto. Verifiquemos isto:

$$310_8 = (3_{10} \times 8^2_{10}) + (1_{10} \times 8^1_{10}) + (0_{10} \times 8^0_{10}) = 192_{10} + 8_{10} = 200_{10}$$

Aqui está um outro exemplo. Desejamos converter 3964_{10} em octal.

$$\begin{aligned} 3964 \div 8 &= 495 \quad \text{com um resto } 4 \\ 495 \div 8 &= 61 \quad \text{com um resto } 7 \\ 61 \div 8 &= 7 \quad \text{com um resto } 5 \\ 7 \div 8 &= 0 \quad \text{com um resto } 7 \end{aligned}$$

Portanto, $7574_8 = 3964_{10}$. Verificando,

$$\begin{aligned} 7574_8 &= (7_{10} \times 8^3_{10}) + (5_{10} \times 8^2_{10}) + (7_{10} \times 8_{10}) + 4_{10} \\ &= (7_{10} \times 512_{10}) + (5_{10} \times 64_{10}) + (7_{10} \times 8_{10}) + (4_{10} \times 1_{10}) \\ &= 3584_{10} + 320_{10} + 56_{10} + 4_{10} \\ &= 3964_{10} \end{aligned}$$

Há diversas outras técnicas para conversão de octal em decimal e de decimal em octal, mas elas não são usadas manualmente com muita freqüência, e foi comprovado que as tabelas tinham tanto valor quanto qualquer uma das técnicas adotadas neste procedimento. As tabelas de conversão de octal/decimal e de decimal/octal são prontamente encontradas em inúmeros locais, inclusive nos manuais distribuídos pelos fabricantes de máquinas binárias.

Um importante emprego do sistema octal é nas listagens de programas e nas "listagens" da memória de máquinas binárias, tornando assim as impressões mais compactas. Os manuais de vários dos maiores fabricantes usam números octais para representar números binários devido à facilidade na conversão e na compactação.

O sistema de numeração hexadecimal é útil por uma razão idêntica. Toda a série de computadores IBM 360, o Spectra 70 da RCA, o NOVA, o PDP-II e muitos outros computadores, a maioria dos minicomputadores em particular, têm suas memórias organizadas em conjuntos de *bytes*, cada byte consistindo de oito dígitos binários. Cada byte ou é usado como uma única entidade para representar um só caráter alfanumérico ou é separado em dois pedaços de 4 bits. (Examinaremos a codificação dos caracteres alfanuméricos usando bytes no Cap. 8.) Quando os bytes são manipulados em dois pedaços de 4 bits, ao programador é dada a opção de declarar cada caráter de 4 bits como um pedaço de um número

QUADRO 2.8

Binário	Hexadecimal	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

binário ou como dois números decimais codificados em binário. Por exemplo, o byte 00011000 pode ser declarado um número binário, em cujo caso ele é igual ao decimal 24, ou como dois caracteres decimais codificados em binário, em cujo caso ele representa o número decimal 18.

Quando a máquina estiver manipulando números em binário, mas em grupos de quatro dígitos, é conveniente ter um código para representação de cada um destes conjuntos de quatro dígitos. Uma vez que há 16 números diferentes possíveis que podem ser representados, os dígitos 0 a 9 não serão suficientes; assim, as letras A, B, C, D, E e F são também

QUADRO 2.9 QUADRO DE CONVERSÃO DE HAXADECIMAL PARA DECIMAL

A. CONVERSÃO DE INTEIROS

EXEMPLO: 2322_{16} é $8192_{10} + 768_{10} + 32_{10} + 2_{10} = 8994,0$

HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC
0	0	0	0	0	0	0	0
1	4.096	1	256	1	16	1	1
2	8.192	2	512	2	32	2	2
3	12.288	3	768	3	48	3	3
4	16.384	4	1.024	4	64	4	4
5	20.480	5	1.280	5	80	5	5
6	24.576	6	1.536	6	96	6	6
7	28.672	7	1.792	7	112	7	7
8	32.768	8	2.048	8	128	8	8
9	36.864	9	2.304	9	144	9	9
A	40.960	A	2.560	A	160	A	10
B	45.056	B	2.816	B	176	B	11
C	49.152	C	3.072	C	192	C	12
D	53.248	D	3.328	D	208	D	13
E	57.344	E	3.584	E	224	E	14
F	61.440	F	3.840	F	240	F	15
Posições de hexadecimais	4		3		2		1

B. CONVERSÃO DE FRAÇÕES

HEX	0123 DEC	HEX	4567 DECIMAL	HEX	0123 DECIMAL	HEX	4567 EQUIVALENTE DECIMAL
0,0	0,0000	0,00	0,0000 0000	0,000	0,0000 0000 0000	0,0000	0,0000 0000 0000 0000
0,1	0,0625	0,01	0,0039 0625	0,001	0,0002 4414 0625	0,0001	0,0000 1525 8789 0625
0,2	0,1250	0,02	0,0078 1250	0,002	0,0004 8828 1250	0,0002	0,0000 3051 7578 1250
0,3	0,1875	0,03	0,0117 1875	0,003	0,0007 3242 1875	0,0003	0,0000 4577 6367 1875
0,4	0,2500	0,04	0,0156 2500	0,004	0,0009 7656 2500	0,0004	0,0000 6103 5156 2500
0,5	0,3125	0,05	0,0195 3125	0,005	0,0012 2070 3125	0,0005	0,0000 7629 3945 3125
0,6	0,3750	0,06	0,0234 3750	0,006	0,0014 6484 3750	0,0006	0,0000 9155 2734 3750
0,7	0,4375	0,07	0,0273 4375	0,007	0,0017 0898 4375	0,0007	0,0001 0681 1523 4375
0,8	0,5000	0,08	0,0312 5000	0,008	0,0019 5312 5000	0,0008	0,0001 2207 0312 5000
0,9	0,5625	0,09	0,0351 5625	0,009	0,0021 9726 5625	0,0009	0,0001 3732 9101 5625
0,A	0,6250	0,0A	0,0390 6250	0,00A	0,0024 4140 6250	0,000A	0,0001 5258 7890 6250
0,B	0,6875	0,0B	0,0429 6875	0,00B	0,0026 8554 6875	0,000B	0,0001 6784 6679 6875
0,C	0,7500	0,0C	0,0468 7500	0,00C	0,0029 2968 7500	0,000C	0,0001 8310 5468 7500
0,D	0,8125	0,0D	0,0507 8125	0,00D	0,0031 7382 8125	0,000D	0,0001 9836 4257 8125
0,E	0,8750	0,0E	0,0546 8750	0,00E	0,0034 1796 8750	0,000E	0,0002 1362 3046 8750
0,F	0,9375	0,0F	0,0585 9375	0,00F	0,0036 6210 9375	0,000F	0,0002 2888 1835 9375
Posições de hexadecimais	1		2		3		4

usadas (ver Quadro 2.8).

Para converter binário em hexadecimal, simplesmente separamos um número binário em grupos de quatro dígitos e convertemos cada grupo de quatro dígitos de acordo com o código precedente. Assim, $10111011_2 = BB_{16}$, $10010101_2 = 95_{16}$, $11000111_2 = C7_{16}$ e $10001011_2 = 8B_{16}$. A mistura de letras e dígitos decimais pode parecer estranha de início, mas estes são simplesmente símbolos convenientes, tal como são os dígitos decimais.

A conversão de hexadecimal em decimal é direta, mas consome tempo. Por exemplo, BB representa $(B \times 16^1) + (B \times 16^0) = (11 \times 16) + (11 \times 1) = 176 + 11 = 187$. Similarmente,

$$\begin{aligned} AB6_{16} &= (10_{10} \times 16^2_{10}) + (11_{10} \times 16_{10}) + 6_{10} \\ &= (10_{10} \times 256_{10}) + 176_{10} + 6_{10} \\ &= 2560_{10} + 176_{10} + 6_{10} \\ &= 2742_{10} \end{aligned}$$

Para converter, por exemplo, $3A6_{16}$ em decimal:

$$\begin{aligned} 3A6_{16} &= (3_{10} \times 16^2_{10}) + (10_{10} \times 16_{10}) + 6_{10} \\ &= (3_{10} \times 256_{10}) + (10_{10} \times 16_{10}) + 6_{10} \\ &= 768_{10} + 160_{10} + 6_{10} \\ &= 934_{10} \end{aligned}$$

Novamente, as tabelas são convenientes para a conversão de hexadecimal em decimal e de decimal em hexadecimal. O Quadro 2.9 é útil para a conversão em qualquer sentido.

O principal emprego do sistema hexadecimal é no que concerne às máquinas organizadas em bytes. Os usuários destes computadores se tornam espantosamente adeptos, com experiência, no manuseio do sistema hexadecimal.

Um número absolutamente grande de exercícios foi incluído no fim deste capítulo. Para os que desejarem estudar os sistemas de numeração octal e hexadecimal com mais profundidade, os Exercícios 60 a 69 contêm informação a respeito da adição e da multiplicação em octal, e os Exercícios 70 a 74 podem ser usados para suplementar o estudo do sistema hexadecimal.

EXERCÍCIOS

1. Converter os seguintes números decimais em números binários equivalentes:

   (a) 39 *(b)* 59 *(c)* 512
   (d) 0,4475 *(e)* $\frac{25}{32}$ *(f)* 0,796875
   (g) 256,75 *(h)* 129,5625 *(i)* 4096,90625

2. Converter os seguintes números decimais em números binários equivalentes:

   (a) 14 *(b)* 13 *(c)* 15
   (d) 6 *(e)* 21 *(f)* 19
   (g) 63 *(h)* 103 *(i)* 256

3. Converter os seguintes números decimais em números binários equivalentes:

   (a) 43 *(b)* 64 *(c)* 4096
   (d) 0,375 *(e)* $\frac{27}{32}$ *(f)* 0,4375
   (g) 512,5 *(h)* 131,5625 *(i)* 2048,0625

4. Converter os seguintes números em números binários equivalentes:

   (a) 11 *(b)* 0,125 *(c)* $2\frac{7}{16}$
   (d) 4,62 *(e)* $3\frac{5}{8}$ *(f)* 0,775

5. Converter os seguintes números binários em números decimais equivalentes:

(a) 1101 (b) 11011 (c) 1011
(d) 0,1011 (e) 0,001101 (f) 0,001101101
(g) 111011,1011 (h) 1011011,001101 (i) 10110,0101011101

6. Converter os seguintes números binários em números decimais equivalentes:

(a) 1101 (b) 1001 (c) 10111
(d) 1011 (e) 11011 (f) 10101

7. Converter os seguintes números binários em números decimais equivalentes:

(a) 1011 (b) 100100 (c) 10011
(d) 0,1101 (e) 0,1001 (f) 0,0101
(g) 1011,0011 (h) 1001,1001 (i) 101,011

8. Converter os seguintes números binários em números decimais equivalentes:

(a) 0,01 (b) 0,0111 (c) 1,110
(d) 111,101 (e) 0110,011 (f) 1101,0011

9. Efetuar as seguintes adições e verificar convertendo os números binários em decimais:

(a) 1001,1 + 1011,01 (b) 100101 + 100101
(c) 0,1011 + 0,1101 (d) 1011,01 + 1001,11

10. Efetuar as seguintes adições e verificar convertendo os números binários em decimais e adicionando.

(a) 1011 + 1101 (b) 1110 + 1111 (c) 11,11 + 10,011
(d) 1101,11 + 1,01 (e) 11101,1 + 10010,1 (f) 111,1 + 111,11

11. Efetuar as seguintes adições e verificar convertendo os números binários em decimais:

(a) 1101,1 + 1011,1 (b) 101101 + 1101101
(c) 0,0011 + 0,1110 (d) 1100,011 + 1011,011

12. Efetuar as seguintes subtrações em binário e verificar convertendo os números em decimal e subtraindo.

(a) 1111 − 1000 (b) 1101 − 1011 (c) 1011,1 − 111,1
(d) 1111,01 − 1011,1 (e) 111,11 − 111,1 (f) 1101,1 − 1110,01

13. Efetuar as seguintes subtrações no sistema de numeração binário.

(a) 64 − 32 (b) 127 − 63
(c) 93,5 − 42,75 (d) $84\frac{9}{32} - 48\frac{5}{16}$

14. Efetuar as seguintes subtrações no sistema de numeração binário.

(a) 128 − 32 (b) $\frac{1}{8} - \frac{1}{16}$ (c) $2\frac{1}{8} - 4\frac{3}{32}$
(d) $31 - \frac{5}{8}$ (e) $62 - 31\frac{1}{16}$ (f) 129 − 35

15. Efetuar as seguintes subtrações no sistema de numeração binário.

(a) $37 - 35$ *(b)* $128 - 64$ *(c)* $94{,}5 - 43{,}75$ *(d)* $255 - 127$

16. Efetuar as seguintes multiplicações e divisões no sistema de numeração binário.

(a) 16×6 *(b)* 31×8 *(c)* $23 \times 2{,}525$
(d) $15 \times 7{,}625$ *(e)* $6 \div 3$ *(f)* $16 \div 6$

17. Efetuar as seguintes multiplicações e divisões no sistema de numeração binário.

(a) 24×12 *(b)* 18×14 *(c)* $32 \div 8$
(d) $27 \div 18$ *(e)* $49{,}5 \times 51{,}75$ *(f)* $58{,}75 \div 23{,}5$

18. Efetuar as seguintes multiplicações e divisões no sistema de numeração binário.

(a) $15 \times 2{,}75$ *(b)* $18 \div 6$ *(c)* $256\frac{1}{2} \div 128\frac{1}{4}$
(d) $31{,}5 \div 15{,}75$ *(e)* $3 \div \frac{3}{8}$ *(f)* $2\frac{2}{8}$

19. Efetuar as seguintes multiplicações e divisões no sistema de numeração binário.

(a) 15×13 *(b)* 10×15 *(c)* $44 \div 11$
(d) $42 \div 12$ *(e)* $7{,}75 \times 2{,}5$ *(f)* $22{,}5 \div 4{,}75$

20. Converter os seguintes números decimais tanto em seus complementos de 9 quanto em seus complementos de 10.

(a) 8 *(b)* 18 *(c)* 7
(d) 23 *(e)* 24 *(f)* 98

21. Converter os seguintes números decimais tanto em seus complementos de 9 quanto em seus complementos de 10.

(a) 5436 *(b)* 1932 *(c)* 45,15 *(d)* 18,293

22. Converter os seguintes números decimais tanto em seus complementos de 9, quanto em seus complementos de 10.

(a) $9\dot{8}$ *(b)* 78 *(c)* 0,81
(d) 0,15 *(e)* 297,64 *(f)* 332,51

23. Converter os seguintes números decimais tanto em seus complementos de 9 quanto em seus complementos de 10.

(a) 3654 *(b)* 2122 *(c)* 54,19 *(d)* 37,263

24. Converter os seguintes números binários tanto em seus complementos de 1 quanto em seus complementos de 2.

(a) 1101 *(b)* 1010 *(c)* 1111
(d) 1110 *(e)* 1011 *(f)* 1001

25. Converter os seguintes números binários tanto em seus complementos de 1 quanto em seus complementos de 2.

(a) 1011 *(b)* 11011 *(c)* 1011,01 *(d)* 11011,01

26. Converter os seguintes números binários tanto em seus complementos de 1 quanto em seus complementos de 2.

(a) 1011 (b) 1101 (c) 0,0111
(d) 0,101 (e) 11,101 (f) 101,011

27. Converter os seguintes números binários tanto em seus complementos de 21 quanto em seus complementos de 2.

(a) 101111 (b) 100100 (c) 10111,10 (d) 10011,11

28. Efetuar as seguintes subtrações usando tanto os complementos de 9 quanto os complementos de 10.

(a) 8 − 4 (b) 16 − 8 (c) 198 − 124
(d) 28,5 − 23,4 (e) 27,6 − 23,4 (f) 0,55 − 0,42

29. Efetuar as seguintes subtrações usando tanto os complementos de 9 quanto os complementos de 10.

(a) 948 − 234 (b) 347 − 263 (c) 349,5 − 245,3 (d) 412,7 − 409,2

30. Efetuar as seguintes subtrações usando tanto os complementos de 9 quanto os complementos de 10.

(a) 14 − 8 (b) 15 − 8 (c) 0,5 − 25
(d) 0,41 − 0,2 (e) 0,434 − 0,23 (f) 1,2 − 0,35

31. Efetuar as seguintes subtrações usando tanto os complementos de 9 quanto os complementos de 10.

(a) 1024 − 913 (b) 249 − 137 (c) 24,1 − 13,4 (d) 239,3 − 119,4

32. Efetuar as seguintes subtrações de números binários usando tanto os complementos de 1 quanto os complementos de 2.

(a) 1010 − 1001 (b) 110 − 11 (c) 110 − 0,11
(d) 0,111 − 0,101 (e) 0,1111 − 0,11 (f) 11,11 − 10,11

33. Efetuar as seguintes subtrações usando os complementos de 1 e de 2:

(a) 1011 − 101 (b) 11011 − 11001
(c) 10111,1 − 10011,1 (d) 11011 − 10011,11

34. Quantos números diferentes podem ser armazenados num conjunto de quatro chaves, cada uma tendo três posições diferentes (4 chaves de três posições)?
35. Quantos números binários diferentes podem ser armazenados num registro que consiste de seis chaves?
36. Quantos números BCD diferentes podem ser armazenados em 12 chaves? (Admitir chaves de 2 posições ou chaves LIGAR-DESLIGAR.)
37. Quantos números BCD diferentes podem ser armazenados num registro que contém 12 chaves usando um código 8, 4, 2, 1? E usando um código em excesso de 3?
38. Escrever os 12 primeiros números no sistema de numeração de base 5 (ou quinário).
39. Escrever os 10 primeiros números no sistema de numeração quaternário, que tem uma base, ou raiz, igual a 4. Usar os dígitos 0, 1, 2 e 3 para expressar estes números.
40. Escrever os 20 primeiros números no sistema de numeração de base 12 (ou *duodecimal*). Usar A para o 10 e B para o 11.
41. Escrever os 25 primeiros números num sistema de numeração de base 11, usando os

dígitos 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 e A para expressar os 25 números que se escrever. (Decimal 10 = A, por exemplo.)

42. Efetuar as seguintes subtrações no sistema de numeração binário usando complementos de 1.

(a) 1011 − 1001 *(b)* 1110 − 1001
(c) 101,11 − 101,01 *(d)* 111,1 − 110,1

43. Usando o sistema de numeração com complemento de 1, efetuar as seguintes subtrações:

(a) 0,1001 − 0,0110 *(b)* 0,1110 − 0,0110 *(c)* 0,01111 − 0,01001
(d) 11011 − 11001 *(e)* 1110101 − 1010010

44. Efetuar as seguintes subtrações no sistema de numeração binário usando complementos de 2.

(a) 1111 − 1101 *(b)* 1110 − 1101
(c) 1011,11 − 101,01 *(d)* 111,1 − 110,1

45. Usando o sistema de numeração com complementos de 2, efetuar as seguintes subtrações e representar também as respostas como frações decimais.

(a) 0,101010 − 0,010101 *(b)* 0,11001 − 0,00100
(c) 0,111000 − 0,000111 *(d)* 0,101100 − 0,010011

46. Converter os seguintes números hexadecimais em números decimais.

(a) 15 *(b)* B9 *(c)* AA4
(d) 9,A *(e)* 9,1A

47. Converter os seguintes números hexadecimais em decimais.

(a) B6C7 *(b)* 64AC *(c)* A492 *(d)* D2763

48. Converter os seguintes números octais em decimais.

(a) 14 *(b)* 124 *(c)* 105
(d) 123 *(e)* 156 *(f)* 15,5

49. Converter os seguintes números octais em decimais.

(a) 2376 *(b)* 2483 *(c)* 286431 *(d)* 22632

50. Converter os seguintes números binários em octais.

(a) 111 *(b)* 111011 *(c)* 101,111
(d) 0,11101 *(e)* 11,11 *(f)* 1101,1101

51. Converter os seguintes números binários em octais.

(a) 101101 *(b)* 101101110 *(c)* 10110111
(d) 110110,011 *(e)* 011,1011011

52. Converter os seguintes números octais em binários.

(a) 56 *(b)* 43 *(c)* 231,2
(d) 231,4 *(e)* 454,45 *(f)* 32,234

53. Converter os seguintes números octais em binários.

(a) 7423 *(b)* 3364 *(c)* 33962 *(d)* 3232,14 *(e)* 3146,52

54. Converter os seguintes números decimais em octais.

(a) 15 *(b)* 9 *(c)* 19
(d) 0,54 *(e)* 0,625 *(f)* 2,125

55. Converter os seguintes números decimais em octais.

(a) 932 *(b)* 332 *(c)* 545,375 *(d)* 632,97 *(e)* 4429,625

56. Converter os seguintes números hexadecimais em binários.

(a) 9 *(b)* 1A *(c)* 0,A1
(d) 0,AB *(e)* A,A *(f)* 12,A

57. Converter os seguintes números hexadecimais em binários.

(a) CD *(b)* 649 *(c)* A13 *(d)* AA,1A *(e)* AB2,234

58. Converter os seguintes números binários em hexadecimais.

(a) 1101,0110 *(b)* 11011110 *(c)* 1111
(d) 11101 *(e)* 11110,01011 *(f)* 1011,11010

59. Converter os seguintes números binários em hexadecimais.

(a) 10110111 *(b)* 10011100 *(c)* 1011111
(d) 0,01111110 *(e)* 0,10110111101

60. Uma regra simples para se multiplicar dois dígitos em qualquer base consiste apenas em multiplicar os dois dígitos em decimal. Se o produto for menor do que a base, ou raiz, considerá-lo; se for maior, dividir (em decimal) pela base e usar o resto como a primeira, ou menos significativa, posição e o quociente como o transporte, ou mais significativo, dígito. Assim, na base 6, $2 \times 2 = 4$, $3 \times 1 = 3$ etc.; no entanto, $2 \times 4 = 8$ e

$$6\overline{)8}\ 1$$

E, assim, $2_6 \times 4_6 = 12_6$. Similarmente, na base 7, $3 \times 4 = 12$ e

$$\begin{array}{r} 1 \\ 7\overline{)12} \\ \underline{7} \\ 5 \end{array}$$

Portanto, $3_7 \times 4_7 = 15_7$. Usando esta regra, efetuar, na base 7:

(a) $2_7 \times 3_7 =$ (b) $2_7 \times 2_7 =$
(c) $4_7 \times 4_7 =$ (d) $4_7 \times 3_7 =$

61. Utilizando a regra do Exercício 60, efetuar:

(a) $3_6 \times 4_6$ (b) $3_6 \times 3_6$ (c) $3_9 \times 4_9$
(d) $4_9 \times 5_9$ (e) $5_9 \times 15_9$

62. Uma tabela de adição em octal é como segue:

+	0	1	2	3	5	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

Usando a tabela acima, podemos adicionar em octal

```
  1 1 <-- transportes
  1 2 6
+ 3 5 7
-------
  5 0 5
```

Efetuar as seguintes adições:

(a) $7_8 + 6_8 =$ (b) $6_8 + 6_8 =$ (c) $6_8 + 16_8 =$
(d) $5_8 + 4_8 =$ (e) $5_8 + 14_8 =$

63. Usando a tabela do Exercício 62, efetuar:

(a) $15_8 + 14_8 =$ (b) $24_8 + 36_8 =$ (c) $126_8 + 347_8 =$
(d) $67_8 + 45_8 =$ (e) $136_8 + 636_8 =$

64. Construir uma tabela de adição em hexadecimal.
65. Usando a tabela do Exercício 64, efetuar:

(a) $6_{16} + A1_{16}$ (b) $7_{16} + 17_{16}$ (c) $8_{16} + 28_{16}$
(d) $A16A_{16} + B16A_{16}$ (e) $A84_{16} + A83_{16}$

66. Efetuar as adições no Exercício 65 em binário e converter novamente a hexadecimal.
67. Efetuar as adições no Exercício 63 em binário e converter novamente a octal.
68. Para multiplicar dois números em octal, usamos a regra dada no Exercício 60 e depois continuamos como segue:

$$6 \times 27 = 6 \times 20 = (6 \times 20) + (6 \times 7)$$
$$= 140 + 52 = 212$$

Multiplicar o seguinte em octal.

(a) $6 \times 7 =$ *(b)* $6 \times 10 =$ *(c)* $5 \times 14 =$

69. Para multiplicar números que têm mais de um dígito, proceder como no Exercício 68, depois somar em octal.

$$\begin{array}{r} 23 \\ \underline{23} \\ 70 \\ \underline{46} \\ 550 \end{array}$$

Multiplicar os seguintes números octais.

(a) 3×14 *(b)* 23×12 *(c)* 11×22
(d) 22×44 *(e)* 13×13 *(f)* 14×15

70. Efetuar as seguintes multiplicações de números hexadecimais

(a) $A \times 7$ *(b)* 9×15 *(c)* $A1 \times 8$
(d) $A11 \times 7$ *(e)* $A12 \times 7$ *(f)* $A13 \times 2B$

71. Usando a regra do Exercício 60, efetuar as seguintes multiplicações de números hexadecimais:

(a) $15 \times B$ *(b)* $14 \times B$ *(c)* $14 \times B$
(d) $142 \times A$ *(e)* 13×14

72. Efetuar as multiplicações no Exercício 70 em binário, depois converter novamente para hexadecimal.
73. Efetuar as multiplicações no Exercício 71 em binário, depois converter novamente para hexadecimal.
74. Na conversão de números decimais em hexadecimal, é conveniente passar primeiro para octal, depois para binário e em seguida para hexadecimal. Por exemplo, para converter 412_{10} em hexadecimal, primeiro passamos para octal.

$$\begin{array}{r} 51 \\ 8\overline{)412} \\ \underline{40} \\ 12 \\ \underline{8} \\ 4 \end{array} \qquad \begin{array}{r} 6 \\ 8\overline{)51} \\ \underline{48} \\ 3 \end{array}$$

3.º dígito ← 2.º dígito ← 1.º dígito

Agora, $412_{10} = 634_8$; convertendo este resultado em binário, $634_8 = \underbrace{110}_{6}\ \underbrace{011}_{3}\ \underbrace{100}_{4}$;

depois, reagrupando. $\underbrace{1}_{1}\ \underbrace{1001}_{9}\ \underbrace{1100}_{C}$; portanto, $412_{10} = 19C$.

Converter os seguintes números decimais em hexadecimais.

(a) 25 *(b)* 396 *(c)* 1342
(d) 512 *(e)* 261

3

ÁLGEBRA BOOLEANA E CIRCUITOS PORTA

Os modernos computadores digitais são projetados, mantidos, e têm sua operação analisada com emprego de técnicas e da simbologia originárias de um campo da Matemática chamado *Álgebra Moderna.* Os algebristas estudaram durante um período de cerca de cem anos os sistemas matemáticos chamados *Álgebras Booleanas.* Nada podia ser mais simples e normal para o raciocínio humano do que as regras de uma álgebra Booleana, pois estas se originaram nos estudos de como raciocinamos, que linhas de raciocínio são válidas, o que constitui uma experiência e outros assuntos relacionados.

O nome Álgebra Booleana homenageia um brilhante† matemático inglês, George Boole, que, em 1854, publicou um livro clássico, *Uma investigação das leis do pensamento, sobre as quais são baseadas as teorias matemáticas da Lógica e das Probabilidades.* O propósito estabelecido por Boole era o de realizar uma análise matemática da Lógica. O trabalho de Boole é talvez mais bem apresentado pela seguinte citação de seu primeiro capítulo:

> As idéias básicas do presente tratado são: investigar as leis fundamentais daquelas operações da mente através das quais é realizado o raciocínio; expressá-las na linguagem simbólica de um Cálculo e, a partir disto, estabelecer a ciência da Lógica e construir seu método; fazer daquele próprio método a base de um método geral para a aplicação da doutrina matemática das Probabilidades; e, finalmente, coletar dos vários elementos de verdade assinalados no curso destas indagações alguns prováveis indícios concernentes à natureza e à constituição da mente humana.

Iniciando por esta investigação das leis do pensamento, Boole instituiu uma "Álgebra Lógica". Esta investigação na natureza da Lógica e fundamentalmente da Matemática levou os matemáticos e os estudiosos de Lógica que surgiram mais tarde a diversos novos campos da Matemática. Dois destes campos, conhecidos por "Cálculo de proposições" e "Álgebra de conjuntos", foram principalmente baseados no trabalho de Boole. Este livro designará a Álgebra atualmente usada no projeto e na manutenção do conjunto de circuitos lógicos por *Álgebra Booleana.*††

A álgebra booleana surgiu inicialmente por ter relação com os problemas que apareceram no projeto de circuitos de chaveamento com relés, em 1938, de Claude E. Shannon, que era assistente de pesquisa no Departamento de Engenharia Elétrica no Instituto de Tecnologia de Massachusetts. Uma versão da tese de Shannon, escrita no MIT para o grau de Mestre de Ciências, foi publicada sob o título *A Symbolic Analysis of Relay and Switching Circuits.* Este artigo técnico apresentava um método para representação de qualquer circuito consistindo de combinações de chaves e relés por um conjunto de expressões matemáticas, e foi desenvolvido um cálculo para manipular estas expressões. O cálculo usado baseava-se comprovadamente nas regras da álgebra booleana. As técnicas fundamentais descritas por Shannon foram adotadas quase que universalmente no projeto e na análise dos circuitos de chaveamento. Em virtude da relação analógica existente entre as

---

†George Boole era filho de um fabricante de calçados. Sua educação formal terminou no terceiro grau. A despeito disto, ele foi um aluno brilhante, ensinando Grego e Latim em sua própria escola, e um conceituado matemático que deixou contribuições valiosas nas áreas das equações diferenciais e das equações de diferenças bem como na álgebra.

††Esta álgebra é, às vezes, chamada *álgebra de comutação*. Ela é, de fato, apenas uma das várias realizações do que os modernos algebristas denominam álgebra booleana.

ações dos relés e dos modernos circuitos eletrônicos, as mesmas técnicas que foram desenvolvidas para o projeto de circuitos de relé estão ainda sendo usadas no projeto de modernos computadores de alta velocidade. Há diversas vantagens em se ter uma técnica matemática para descrição das operações internas de um computador. Para uma finalidade, muitas vezes é bem mais conveniente fazer cálculos com expressões usadas para representar os circuitos de chaveamento do que usar diagramas esquemáticos ou mesmo lógicos. Além disso, assim como uma expressão algébrica ordinária pode ser simplificada por meio dos teoremas básicos, a expressão que descreve uma dada estrutura com circuitos de chaveamento pode também ser reduzida ou simplificada. Isso permite que o projetista de lógica simplifique o conjunto de circuitos usado, conseguindo economia na construção e confiabilidade de operação. A álgebra booleana também proporciona um modo econômico e direto de descrição do conjunto de circuitos usado nos computadores. Em suma, um conhecimento de álgebra booleana torna-se indispensável no campo da computação.

3.1 CONCEITOS FUNDAMENTAIS DE ÁLGEBRA BOOLEANA

Quando uma variável é usada numa fórmula algébrica, supõe-se, em geral, que a variável pode assumir qualquer valor numérico. Por exemplo, na fórmula $2X \times 5Y = Z$, admitimos que X, Y e Z possam variar através de todo o campo dos números reais.

As variáveis usadas nas equações booleanas possuem, contudo, uma característica única; elas podem assumir apenas um de dois valores possíveis. Estes dois valores podem ser representados pelos símbolos 0 e 1.† Se uma equação que descreve o conjunto de circuitos lógicos tiver inúmeras variáveis, compreende-se ainda que cada uma das variáveis poderá assumir somente o valor 0 ou 1. Por exemplo, na equação $X + Y = Z$, cada uma das variáveis X, Y e Z pode ter somente os valores 0 ou 1.

Este conceito se tornará mais claro se for definido um outro símbolo, o +, ou símbolo de *adição lógica*. Quando o símbolo + é colocado entre duas variáveis, sejam X e Y, já que tanto X quanto Y podem assumir apenas o papel de 0 ou 1, podemos definir o símbolo + listando todas as combinações possíveis para X e para Y e o valor resultante de $X + Y$.

As possíveis combinações de entrada e de saída podem ser dispostas como segue:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 1$$

Esta é uma tabela de adição lógica e poderia representar uma tabela padrão de adição em binário, exceto quanto à última entrada. Quando tanto X quanto Y representam dígitos 1, o valor de $X + Y$ é 1. O símbolo + não tem, portanto, o significado "normal", mas é um símbolo de adição lógica. A equação $X + Y = Z$ pode ser lida "X ou Y é igual a Z" ou "X mais Y é igual a Z". Este conceito pode ser estendido a qualquer número de variáveis; por exemplo, na equação $A + B + C + D = E$, mesmo se A, B, C e D tivessem todas o valor 1, a soma dos valores, ou E, representaria apenas 1.

Para evitar ambigüidade, foram recomendados vários outros símbolos como substituições para o sinal +. Alguns destes são U,⁺ v e V. A maioria do pessoal especializado em computadores, no entanto, ainda usa o sinal +, que foi o símbolo originalmente proposto por Boole.

3.2 MULTIPLICAÇÃO LÓGICA

Uma segunda operação importante em álgebra booleana é a operação que denominaremos *multiplicação lógica*. As regras da multiplicação lógica podem ser novamente dadas pela simples listagem de todos os valores que possam ocorrer. Estes são

$$0 \cdot 0 = 0$$
$$0 \cdot 1 = 0$$
$$1 \cdot 0 = 0$$
$$1 \cdot 1 = 1$$

---

†Ou T e F ou + e − etc. 0 e 1, no entanto, são quase universalmente usados no trabalho de computadores.

⁺A equação precedente podia então ser escrita $A \cup B \cup C \cup D = E$.

Assim, por exemplo, se escrevermos $Z = X \cdot Y$ e acharmos $X = 0$ e $Y = 1$, então $Z = 0$. Somente quando X e Y forem ambos 1 é que deveremos ter para Z um 1.

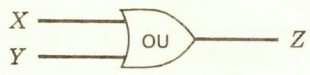
Tanto o + quanto o · obedecem a uma regra matemática chamada *lei associativa*. Esta lei diz que, para +, $(X + Y) + Z = X + (Y + Z)$ e para · , $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$. O significado disto é que, por exemplo, podemos escrever $X + Y + Z$ sem ambigüidade, pois não importa em que ordem a soma lógica é efetuada, porque o resultado é o mesmo; assim, adicionando X a Y e depois adicionando Z, obtém-se o mesmo resultado que quando se adiciona Y a Z e depois se adiciona este resultado a X. Podemos testar esta lei tanto para + quanto para ·, experimentando novamente todas as combinações.

Note que enquanto os + e os · podem ser usados livremente, os dois não podem ser "misturados" sem ambigüidade na ausência de outras regras. Por exemplo, $A \cdot B + C$ significa $(A \cdot B) + C$ ou $A \cdot (B + C)$? As duas expressões formam valores diferentes para $A = 0$, $B = 0$ e $C = 1$, pois então temos $(0 \cdot 0) + 1 = 1$ e $0 \cdot (0 + 1) = 0$, que são diferentes. (A operação sempre da esquerda para a direita aliviará isto e é usada em algumas linguagens de programação, mas não comumente por algebristas ou projetistas de computador ou pessoal de manutenção.) A regra que é utilizada é que · é sempre efetuada antes da +. Assim, $X \cdot Y + Z$ é $(X \cdot Y) + Z$, e $X \cdot Y + X \cdot Z$ significa $(X \cdot Y) + (X \cdot Z)$.

3.3 PORTAS OU & PORTAS E

As operações + e · são fisicamente realizadas por dois tipos de circuitos eletrônicos chamados portas OU e portas E. Trataremos destes circuitos como "caixas pretas", adiando para mais tarde qualquer discussão a respeito de como opera o conjunto de circuitos real.

Uma *porta* é simplesmente um circuito eletrônico que opera com um ou mais sinais de entrada para produzir um sinal de saída. Um dos circuitos porta mais simples e mais freqüentemente usados é chamado porta OU, e o símbolo de diagrama-bloco que o representa é mostrado na Fig. 3.1, assim como a tabela de combinações das entradas e saídas correspondentes. Uma vez que as entradas X e Y são sinais com valores 0 ou 1 em qualquer instante de tempo determinado, o sinal de saída Z pode ser descrito listando-se simplesmente todos os valores de X e de Y e o valor resultante para Z. Um estudo da tabela na Fig. 3.1 indica que a porta OU efetua adição lógica em suas entradas.



ENTRADA		SAÍDA
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Fig. 3.1 A porta OU.

Similarmente, a porta E, na Fig. 3.2, efetua multiplicação lógica com os valores da entrada, fornecendo uma saída Z com valor $X \cdot Y$, de modo que Z é um 1 somente quando tanto X quanto Y forem dígitos 1.

Assim como as operações + e · podiam ser estendidas a diversas variáveis usando a lei associativa, as portas OU e as portas E podem ter mais do que duas entradas. A Fig. 3.3 mostra portas E e portas OU de três entradas e a tabela de todas as combinações da entrada para cada um dos circuitos. Como devia ser esperado, as portas OU com entradas X, Y e Z têm uma saída 1 se X OU Y OU Z for um 1, de modo que podemos escrever $X + Y + Z$ para sua saída. Além disso, a saída da porta E com entradas X, Y e Z é um 1 somente quando todas as três entradas forem dígitos 1, de modo que podemos escrever a saída como $X \cdot Y \cdot Z$.

O argumento acima pode ser generalizado. Uma porta OU de quatro entradas tem uma

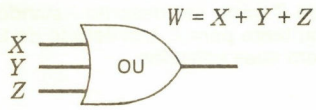


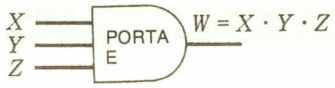
ENTRADA		SAÍDA	
X	Y	Z	
0	0	0	$0 \cdot 0 = 0$
0	1	0	$0 \cdot 1 = 0$
1	0	0	$1 \cdot 0 = 0$
1	1	1	$1 \cdot 1 = 1$

Fig. 3.2 A porta E.

saída 1 quando qualquer de suas entradas for um 1, e uma porta E de quatro entradas tem uma saída 1 somente quando todas as quatro entradas forem dígitos 1.

É muitas vezes conveniente abreviar $X \cdot Y \cdot Z$ para XYZ, e, às vezes, usaremos esta convenção.





ENTRADA			SAÍDA
X	Y	Z	W
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

ENTRADA			SAÍDA
X	Y	Z	W
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Fig. 3.3 Portas OU e E com três entradas.

3.4 COMPLEMENTAÇÃO E INVERSORES

As duas operações definidas até agora foram as que os algebristas chamariam *operações binárias* no sentido de que elas definem uma operação com duas variáveis. Há também *operações singulares ou unárias ou unitárias*, que definem uma operação com uma única variável. Um exemplo familiar de operação unitária é $-$, pois podemos escrever -5 ou -10 ou $-X$, significando que devemos considerar o negativo destes valores. ($-$ é também usado como um símbolo de operação binária de subtração, o que o torna um exemplo familiar, porém ambíguo.)

Na álgebra booleana temos uma operação chamada *complementação*, e o símbolo que usaremos é $\bar{}$. Assim, escreveremos $\bar{X}$, significando "achar o complemento de X", ou

$\overline{(X + Y)}$, significando "achar o complemento de $X + Y$". A operação de complementação pode ser definida bem simplesmente:

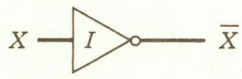
$$\overline{0} = 1$$
$$\overline{1} = 0$$

O complemento de um valor pode ser considerado repetidamente. Por exemplo, podemos achar $\overline{\overline{X}}$; *para* $X = 0$, ele é $\overline{\overline{0}} = \overline{1} = \overline{0} = 1$. Para $X = 1$, ele é $\overline{\overline{1}} = \overline{0} = \overline{1} = 0$.

Uma regra útil é baseada no fato de que $\overline{\overline{X}} = X$; verificando, achamos que $\overline{\overline{0}} = \overline{1} = 0$ e $\overline{\overline{1}} = \overline{0} = 1$. [Esta regra, de que a complementação dupla dá o valor original, é uma característica importante de uma álgebra booleana que geralmente não se aplica para a maioria das operações unitárias ou unárias. Por exemplo, observemos que, para a operação de elevar ao quadrado um número real, a regra não se aplica: $(3^2)^2 = 81$, e não 3.]

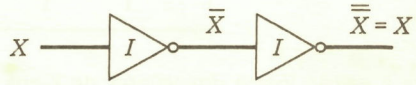
A operação de complementação é fisicamente realizada por um circuito porta chamado *inversor*. A Fig. 3.4(*a*) mostra um inversor e a tabela de combinações de suas entrada e saída. A Fig. 3.4(*b*) mostra também que a ligação de dois inversores em série dá uma saída igual à entrada, e isto constitui o correspondente de circuito lógico à lei da complementação dupla, $\overline{\overline{X}} = X$.

Vários outros símbolos foram usados como símbolo da complementação. Por exemplo, o ~ é freqüentemente usado pelos estudiosos da lógica, que escrevem $\sim X$ e lêem isto como "a negação de X". O símbolo ′ tem sido usado pelos matemáticos e pelos técnicos de computadores; portanto, X' é o complemento de X nestes sistemas. O símbolo com a barra acima é agora usado pelo American National Standards Institute (Instituto de Padrões Nacionais Americanos) e pelo Military Standards, bem como pela maioria dos periódicos e fabricantes, e também o usaremos.



ENTRADA X	SAÍDA $\overline{X}$
0	1
1	0

$\overline{0} = 1$
$\overline{1} = 0$
(*a*)



(*b*)

Fig. 3.4 *(a)* Diagramas-bloco de inversor; *(b)* dois inversores em série.

3.5 CÁLCULO DE EXPRESSÕES LÓGICAS

As tabelas de valores das três operações que foram explicadas são, às vezes, chamadas *tabelas da verdade* ou *tabelas de combinações*. Para estudar uma expressão lógica, é muito útil construir uma tabela de valores para as variáveis e em seguida avaliar a expressão para cada uma das combinações possíveis das variáveis alternativamente. Consideremos a expressão $X + Y\overline{Z}$. Há três variáveis nesta expressão, X, Y e Z, cada uma das quais pode assumir os valores 0 ou 1. As possíveis combinações de valores podem ser dispostas em ordem crescente,† como no Quadro 3.1.

†Observemos que as variáveis em cada fila desta tabela podem ser combinadas num número binário. Os números binários contarão então de 000 a 111 em binário ou de 0 a 7 em decimal. Às vezes, cada fila é numerada em decimal de acordo com o número representado. Pode ser feita referência então à fila usando-se o número decimal. Por exemplo, a fila 0 tem valores de 0, 0, 0 para X, Y, e Z, a fila 6 tem valores 1, 1, 0 e a fila 7 tem valores de 1, 1, 1.

QUADRO 3.1

X	Y	Z
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

QUADRO 3.2

X	Y	Z	$\bar{Z}$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Uma das variáveis, Z, é complementada na expressão $X + Y\bar{Z}$; portanto, uma coluna é agora acrescentada à tabela, listando os valores de $\bar{Z}$ (ver Quadro 3.2).

Uma coluna é agora acrescentada, listando os valores que $Y\bar{Z}$ assume para cada valor de X, Y e Z. Esta coluna conterá o valor 1 somente quando tanto Y quanto $\bar{Z}$ forem um 1 (ver Quadro 3.3).

QUADRO 3.3

X	Y	Z	$\bar{Z}$	$Y\bar{Z}$
0	0	0	1	0
0	0	1	0	0
0	1	0	1	1
0	1	1	0	0
1	0	0	1	0
1	0	1	0	0
1	1	0	1	1
1	1	1	0	0

Agora a adição lógica dos valores de X aos valores que foram calculados para $Y\bar{Z}$ é efetuada numa coluna final (ver Quadro 3.4).

QUADRO 3.4

X	Y	Z	$\bar{Z}$	$Y\bar{Z}$	$X + Y\bar{Z}$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	0	0	1
1	1	0	1	1	1
1	1	1	0	0	1

A coluna final contém o valor de $X + Y\bar{Z}$ para cada conjunto de valores de entrada que X, Y e Z podem assumir. Por exemplo, quando $X = 1$, $Y = 0$ e $Z = 1$, a expressão tem o valor de 1.

3.6 CÁLCULO DE UMA EXPRESSÃO QUE CONTÉM PARÊNTESES

O seguinte exemplo ilustra o processo de construção de uma tabela da verdade para a expressão $X + Y(\bar{X} + \bar{Y})$. Há somente duas variáveis na expressão, X e Y. Primeiro é construída uma tabela dos valores que X e Y podem assumir (ver Quadro 3.5).

QUADRO 3.5

X	Y
0	0
0	1
1	0
1	1

Agora, já que a expressão contém tanto $\bar{X}$ quanto $\bar{Y}$, duas colunas são acrescentadas, listando os complementos dos valores originais das variáveis (ver Quadro 3.6).

QUADRO 3.6

X	Y	$\bar{X}$	$\bar{Y}$
0	0	1	1
0	1	1	0
1	0	0	1
1	1	0	0

Os diversos valores que o termo dentro do parênteses, $(\bar{X} + \bar{Y})$, assume são calculados agora (ver Quadro 3.7).

QUADRO 3.7

X	Y	$\bar{X}$	$\bar{Y}$	$(\bar{X} + \bar{Y})$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

Os valores de $\bar{X} + \bar{Y}$ são agora multiplicados pelos valores de Y na tabela, formando uma outra coluna que representa $Y(\bar{X} + \bar{Y})$ (ver Quadro 3.8).

QUADRO 3.8

X	Y	$\bar{X}$	$\bar{Y}$	$\bar{X} + \bar{Y}$	$Y(\bar{X} + \bar{Y})$
0	0	1	1	1	0
0	1	1	0	1	1
1	0	0	1	1	0
1	1	0	0	0	0

Finalmente, os valores de $Y(\bar{X} + \bar{Y})$ são acrescentados aos valores de X que estão listados, formando a coluna final e completando a tabela (ver Quadro 3.9).

QUADRO 3.9

X	Y	$\bar{X}$	$\bar{Y}$	$\bar{X} + \bar{Y}$	$Y(\bar{X} + \bar{Y})$	$X + Y(\bar{X} + \bar{Y})$
0	0	1	1	1	0	0
0	1	1	0	1	1	1
1	0	0	1	1	0	1
1	1	0	0	0	0	1

A inspeção da coluna final da tabela indica que os valores assumidos pela função $X + Y(\bar{X} + \bar{Y})$ são idênticos aos valores achados na tabela de adição lógica. Isso indica que a função $X + Y(\bar{X} + \bar{Y})$ é equivalente à função $X + Y$. Esta equivalência foi estabelecida pela *prova por indução perfeita*. Se um circuito lógico fosse construído para cada uma das duas expressões, um circuito requereria dois inversores, dois circuitos OU e um circuito E, enquanto que o outro circuito requereria apenas um circuito OU. Ambos os circuitos realizariam a mesma função, produzindo saídas idênticas para cada combinação de entradas.

3.7 LEIS BÁSICAS DA ÁLGEBRA BOOLEANA

Foram apresentadas algumas relações fundamentais de álgebra booleana. Um conjunto completo das operações básicas é listado abaixo.† Embora de aspecto simples, estas regras podem ser usadas para construir uma álgebra booleana,+ determinando todas as relações que se seguem:

$$\text{Se } X \neq 0, \text{ então } X = 1$$

e

$$\text{Se } X \neq 1, \text{ então } X = 0$$

ADIÇÃO LÓGICA	MULTIPLICAÇÃO LÓGICA	REGRAS DE COMPLEMENTO
$0 + 0 = 0$	$0 \cdot 0 = 0$	$\bar{0} = 1$
$0 + 1 = 1$	$0 \cdot 0 = 0$	$\bar{1} = 0$
$1 + 0 = 1$	$1 \cdot 0 = 0$	
$1 + 1 = 1$	$1 \cdot 1 = 1$	

Uma lista de relações úteis é apresentada no Quadro 3.10. A maioria das regras básicas pelas quais as expressões da álgebra booleana podem ser manipuladas está contida nesta última tabela. Cada uma destas regras pode ser demonstrada usando-se a "prova por indução perfeita". Um exemplo desta prova para a regra 3 no Quadro 3.10 é como segue: A variável X pode ter somente o valor 0 ou 1; se X tiver o valor 0, então $0 + 0 = 0$; se X tiver o valor 1, então $1 + 1 = 1$. Portanto, $X + X = X$.

A mesma técnica básica pode ser usada para demonstrar o restante das regras. A regra 9 estabelece que a complementação dupla de uma variável resulta na variável original. Se X for igual a 0, então o primeiro complemento será 1 e o segundo será 0, o valor original. Se o valor original de X for 1, então o primeiro complemento será 0 e o segundo será 1, o valor original. Portanto, $\bar{\bar{X}} = X$.

---

†Há, realmente, diversos possíveis conjuntos de postulados que podem ser usados para definir a álgebra. O tratamento particular de álgebra booleana utilizado aqui é deduzido do de E. V. Huntington e M. H. Stone. O autor também gostaria de admitir a influência de I. S. Reed e S. H. Caldwell neste desenvolvimento dos conceitos da álgebra.

+Observemos que estas regras são usadas para construir um *exemplo*, ou *realização*, de uma álgebra booleana. Notamos que, estritamente falando, esta álgebra booleana consiste de um conjunto B de dois elementos que denominamos 0 e 1, uma operação de adição +, uma operação de multiplicação · e uma operação de complementação $^-$. Há outras álgebras booleanas (um número infinito), mas esta era a álgebra de Boole original. Esta álgebra é, às vezes, chamada *álgebra de comutação* para identificá-la mais rigorosamente, mas é a mesma que o "cálculo proposicional", por exemplo.

QUADRO 3.10 REGRAS DE ÁLGEBRA BOOLEANA

1. $0 + X = X$
2. $1 + X = 1$
3. $X + X = X$
4. $X + \overline{X} = 1$
5. $0 \cdot X = 0$
6. $1 \cdot X = X$
7. $X \cdot X = X$
8. $X \cdot \overline{X} = 0$
9. $\overline{\overline{X}} = X$
10. $X + Y = Y + X$
11. $X \cdot Y = Y \cdot X$
12. $X + (Y + Z) = (X + Y) + Z$
13. $X(YZ) = (XY)Z$
14. $X(Y + Z) = XY + XZ$
15. $X + XZ = X$
16. $X(X + Y) = X$
17. $(X + Y)(X + Z) = X + YZ$
18. $X + \overline{X}Y = X + Y$
19. $XY + YZ + \overline{Y}Z = XY + Z$

As regras 10 e 11, que são conhecidas por *leis comutativas,* expressam o fato de que a ordem em que uma combinação de termos é efetuada não afeta o resultado da combinação. A regra 10 é a lei comutativa da adição, que estabelece que a ordem da adição não afeta a soma ($X + Y = Y + X$). A regra 11 é a lei comutativa da multiplicação ($XY = YX$), que estabelece que a ordem da multiplicação não afeta o produto.

As regras 12 e 13 constituem as *leis associativas.* A regra 12 estabelece que, na adição de vários termos, a soma que será obtida se o primeiro termo for adicionado ao segundo e o resultado adicionado será a mesma que a soma obtida se o segundo termo for adicionado ao terceiro e o resultado adicionado ao primeiro termo, ou: $X + (Y + Z) = (X + Y) + Z$. A regra 13 é a lei associativa da multiplicação, estabelecendo que, num produto com três fatores, *quaisquer* dois podem ser multiplicados juntos, seguidos pelo terceiro, $X(YZ) = (XY)Z$.

A regra 14, a *lei distributiva,* estabelece que o produto de um monômio (X) por um polinômio $(Y + Z)$ é igual à soma dos produtos do monômio por cada termo do polinômio, $X(Y + Z) = XY + XZ$.

As três leis, comutativa, associativa e distributiva, podem ser estendidas de modo a incluir qualquer número de termos. Por exemplo, a lei comutativa da adição estabelece que $X + Y = Y + X$. Isso pode ser estendido para

$$X + Y + Z + A = A + Y + Z + X$$

A lei comutativa para a multiplicação também pode ser estendida: $XYZ = YZX$.
Estas duas leis são úteis para se rearrumar os termos de uma equação.

As leis podem também ser combinadas:

$$(X + Y) + (Z + A) = (A + Y) + (X + Z)$$

e $(XY)(ZA) = (XA)(ZY)$. Estas duas leis são úteis para o reagrupamento dos termos de uma equação.

A lei distributiva pode ser estendida de diversas maneiras:

$$X(Y + Z + A) = XY + XZ + XA$$

Se dois polinômios, tais como $(W + X)$ e $(Y + Z)$, devem ser multiplicados, um dos polinômios é tratado como um monômio e multiplicado pelos termos individuais do outro polinômio. Os resultados são depois multiplicados de acordo com a lei distributiva. Por exemplo,

$$(W + X)(Y + Z) = W(Y + Z) + X(Y + Z) = WY + WZ + XY + XZ$$

3.8 PROVA POR INDUÇÃO PERFEITA

Observemos que, entre outras, a regra 17 não se aplica à álgebra "normal". A regra pode ser obtida das regras precedentes como segue:

$$\begin{aligned}
(X + Y)(X + Z) &= XX + XZ + XY + YZ && \text{onde } XX = X, \text{ regra 7, assim} \\
&= X + XZ + XY + YZ \\
&= X + XY + XZ + YZ \\
&= X(1 + Y) + Z(X + Y) && \text{onde } (1 + Y) = 1, \text{ regra 2, assim} \\
&= X + Z(X + Y) \\
&= X + XZ + YZ \\
&= X(1 + Z) + YZ && \text{onde } (1 + Z) = 1, \text{ regra 2, assim} \\
&= X + YZ
\end{aligned}$$

Portanto,

$$(X + Y)(X + Z) = X + YZ$$

Uma vez que a regra 17 não se aplica à álgebra normal é interessante testar a regra usando a prova pela indução perfeita. Será portanto necessário construir tabelas da verdade para o segundo membro $(X + YZ)$ e para o primeiro membro $(X + Y)(X + Z)$ da equação e comparar os resultados (ver Quadros 3.11 e 3.12).

QUADRO 3.11

X	Y	Z	ZY	$X + YZ$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

QUADRO 3.12

X	Y	Z	$X + Y$	$X + Z$	$(X + Y)(X + Z)$
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

A última coluna da tabela para a função $X + YZ$ é idêntica à última coluna da tabela para $(X + Y)(X + Z)$. Isto demonstra (por meio da prova da indução perfeita) que as expressões são equivalentes.

As regras 15 e 16 também não são regras na álgebra "normal". A seguir, temos uma demonstração da regra 15 usando as regras precedentes:

$X + XZ = X(1 + Z)$ lei distributiva, e já que
$(1 + Z) = 1$ pela regra 2
$X + XZ = X(1)$ e $X(1) = X$ pela regra 6

Portanto,

$$X + XZ = X$$

É proveitoso experimentar e provar a regra 15 usando a prova por indução perfeita neste ponto. Aqui está uma prova da regra 16 usando as regras que a precedem.

$$\begin{aligned} X(X + Y) &= XX + XY \quad \text{lei distributiva, e já que } XX = X \\ &= X + XY \\ &= X(1 + Y) \quad \text{onde } 1 + Y = 1, \text{ regra 2, assim} \\ &= X \end{aligned}$$

É instrutivo provar esta regra também pela indução perfeita neste ponto.

3.9 SIMPLIFICAÇÕES DE EXPRESSÕES

As regras dadas podem ser usadas para simplificar expressões booleanas, tal como as regras da álgebra normal podem ser usadas para simplificar expressões. Consideremos a expressão

$$(X + Y)(X + \bar{Y})(\bar{X} + Z)$$

Os dois primeiros termos consistem de $(X + Y)(X + \bar{Y})$; estes termos podem ser multiplicados e, uma vez que $X + X\bar{Y} + XY = X$ e $Y\bar{Y} = 0$, reduzidos a X.

A expressão foi agora reduzida a $X(\bar{X} + Z)$, que pode ser expressa como $X\bar{X} + XZ$ (regra 14), e, já que $X\bar{X}$ é igual a 0, a expressão inteira $(X + Y)(X + \bar{Y})(\bar{X} + Z)$ pode ser reduzida a XZ.

Uma outra expressão que pode ser simplificada é $XYZ + X\bar{Y}Z + XY\bar{Z}$. Inicialmente, os três termos $XYZ + X\bar{Y}Z + XY\bar{Z}$ podem ser escritos $X(YZ + \bar{Y}Z + Y\bar{Z})$, usando a regra 14. Então, usando a regra 14 novamente: $X[Y(Z + \bar{Z}) + Y\bar{Z}]$; e já que $Z + \bar{Z}$ é igual a 1, temos $X(Y + \bar{Y}Z)$.

A expressão $X(Y + \bar{Y}Z)$ pode ser ainda reduzida para $X(Y + Z)$ usando-se a regra 18. A expressão final pode ser escrita de duas maneiras: $X(Y + Z)$ ou $XY + XZ$. A primeira expressão é, em geral, preferível se a equação é construída como um circuito eletrônico, porque ela requer apenas um circuito E e um circuito OU.

3.10 TEOREMAS DE De MORGAN

As duas regras seguintes são conhecidas por teoremas de De Morgan.

$$\overline{(X + Y)} = \bar{X} \cdot \bar{Y}$$
$$\overline{(X \cdot Y)} = \bar{X} + \bar{Y}$$

O complemento de qualquer expressão booleana, ou uma parte de qualquer expressão, pode ser achado por meio desses teoremas. Nestas regras, duas etapas são usadas para formar um complemento:

1. Os símbolos de adição devem ser substituídos pelos símbolos de multiplicação ou os símbolos de multiplicação pelos de adição.
2. Cada um dos termos na expressão é complementado.

O uso do teorema de De Morgan pode ser demonstrado achando-se o complemento da expressão $(X + YZ)$. Primeiro, é importante observar que um sinal de multiplicação foi omitido e que a expressão podia ser escrita $X + (Y \cdot Z)$. Para complementar isto, o símbolo de adição é substituído por um símbolo de multiplicação, e os dois termos são complementados, dando $\bar{X} \cdot (\overline{Y \cdot Z})$; depois, o termo restante é complementado, $\bar{X}(\bar{Y} + \bar{Z})$. A seguinte equivalência foi achada: $\overline{(X + YZ)} = \bar{X}(\bar{Y} + \bar{Z})$.

O complemento de $(\bar{W}X + YZ)$ pode ser formado de duas etapas:

1. O símbolo de adição é mudado.
2. O complemento de cada termo é formado.

$$\overline{(\bar{W} \cdot X)}\,\overline{(Y \cdot \bar{Z})}$$

se torna $(W + \bar{X})(\bar{Y} + Z)$. Observemos que, uma vez que W e Z já foram complementados, eles se tornam não-complementados pelo teorema $\bar{\bar{X}} = X$.

Às vezes, é necessário complementar ambos os membros de uma equação. Isto pode ser feito exatamente como antes:

Complementando ambos os membros,

$$\begin{aligned} WX + YZ &= 0 \\ \overline{(WX + YZ)} &= \overline{0} \\ (\bar{W} + \bar{X})(\bar{Y} + \bar{Z}) &= 1 \end{aligned}$$

3.11 DUALIDADE BÁSICA DA ÁLGEBRA BOOLEANA

O teorema de De Morgan expressa uma dualidade básica que sustenta toda a álgebra booleana. Os postulados e teoremas que foram apresentados podem todos ser divididos em pares. Por exemplo, $(X + Y) + Z = X + (Y + Z)$ é o "dual" de $(XY)Z = X(YZ)$, e $X + 0 = X$ é o dual de $X \cdot 1 = X$.

Muitas vezes, as regras ou teoremas são listados numa ordem que ilustra a dualidade da álgebra. Na demonstração dos teoremas ou regras da álgebra, é então necessário somente provar um teorema, e o "dual" do teorema fica necessariamente demonstrado. Por exemplo, se for provado que $X + XY = X$, poder-se-á imediatamente acrescentar o teorema $X(X + Y) = X$ à lista de teoremas como o "dual" da primeira expressão.† Com efeito, toda a álgebra booleana é afirmada nesta base de "dois-para-um".

3.12 DEDUÇÃO DE UMA EXPRESSÃO BOOLEANA

Ao projetar um circuito lógico, o projetista lógico trabalha a partir de dois conjuntos de valores conhecidos: (1) os vários estados que as entradas para a estrutura lógica podem assumir e (2) as saídas desejadas para cada condição da entrada. A expressão lógica é deduzida destes conjuntos de valores.

Consideremos um problema específico. Uma estrutura ou circuito lógico tem duas entradas, X e Y, e uma saída Z. A relação entre as entradas e saídas deve ser como segue:

1. Quando tanto X quanto Y forem 0, a saída Z deve ser 1.
2. Quando X for 0 e Y for 1, a saída Z deve ser 0.
3. Quando X for 1 e Y for 0, a saída Z deve ser 1.
4. Quando X for 1 e Y for 1, a saída Z deve ser 1.

Estas relações podem ser expressas em forma tabular como indicado no Quadro 3.13.

Agora é necessário acrescentar uma outra coluna à tabela. Esta coluna consistirá de uma lista de *termos produto* obtida dos valores das variáveis de entrada. A nova coluna conterá cada uma das variáveis de entrada listadas em cada fila da tabela, com a letra representando a respectiva entrada complementada quando o valor da entrada para esta variável for 0, e não-complementada quando o valor da entrada for 1. Os termos obtidos desta maneira são designados por termos produto. Com duas variáveis X e Y de entrada, cada fila da tabela conterá um termo produto que consiste de X e de Y, com X ou Y complementados ou não, dependendo dos valores de entrada para aquela fila (ver Quadro 3.14).

---

†Quando a primeira expressão, $X + XY = X$, foi complementada, $\bar{X}(\bar{X} + \bar{Y}) = \bar{X}$ é obtida. As variáveis não-complementadas podem então ser substituídas em ambos os membros da equação sem alterar a equivalência básica da expressão.

Sempre que Z for igual a 1, o termo produto de X e Y proveniente da mesma fila é removido e formado numa *soma-de-produtos*. Portanto, os termos produto das primeira, terceira e quarta filas são selecionados. Estes são $\bar{X}\bar{Y}$, $X\bar{Y}$ e XY.

Existem agora três termos, cada um sendo o produto de duas variáveis. A soma destes produtos é igual à expressão desejada. Este tipo de expressão é muitas vezes denominado *expansão canônica* para a função. A expressão completa na forma normal é

$$\bar{X}\bar{Y} + X\bar{Y} + XY = Z$$

O primeiro membro desta expressão pode ser simplificado como segue:

$$\begin{aligned} \bar{X}\bar{Y} + X\bar{Y} + XY &= Z \\ \bar{X}\bar{Y} + X(\bar{Y} + Y) &= Z \\ \bar{X}\bar{Y} + X &= Z \\ X + \bar{Y} &= Z \end{aligned}$$

e, finalmente, pela regra 18 no Quadro 3.10,

A tabela da verdade pode então ser construída para verificar a função que foi deduzida (ver Quadro 3.15).

A última fila desta coluna concorda com a última fila da tabela da verdade da função desejada, mostrando que as expressões são equivalentes.

QUADRO 3.13

Entradas		Saída
X	Y	Z
0	0	1
0	1	0
1	0	1
1	1	1

QUADRO 3.14

Entradas		Saída	Termos produto
X	Y	Z	
0	0	1	$\bar{X}\bar{Y}$
0	1	0	$\bar{X}Y$
1	0	1	$X\bar{Y}$
1	1	1	XY

QUADRO 3.15

X	Y	$\bar{Y}$	$X + \bar{Y}$
0	0	1	1
0	1	0	0
1	0	1	1
1	1	0	1

A expressão $X + \bar{Y}$ pode ser construída segundo uma entre duas maneiras. Se somente as entradas X e Y estivessem disponíveis, como poderia ser o caso se as entradas para o circuito fossem originárias de um outro circuito lógico ou de certos tipos de dispositivos de armazenamento, seria necessário um inversor para formar $\bar{Y}$. O circuito então exigiria um inversor mais uma porta OU. Geralmente, o complemento da entrada Y seria disponível, e apenas uma porta OU seria requerida para a segunda maneira pela qual a expressão deve ser construída.

Será deduzida uma outra expressão com três entradas (designadas por X, Y e Z). Admitamos que as relações desejadas entre as entradas e a saída foram determinadas, conforme indicado no Quadro 3.16.

QUADRO 3.16

Entradas			Saída
Quando: $X = 0$,	$Y = 0$,	$Z = 0$	1
$X = 0$,	$Y = 0$,	$Z = 1$	0
$X = 0$,	$Y = 1$,	$Z = 0$	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

1. É formada uma tabela da verdade (ver Quadro 3.17).
2. Uma coluna é acrescentada, listando as entradas *X*, *Y* e *Z* de acordo com seus valores nas colunas de entrada (ver Quadro 3.18).
3. Os termos produto originários de cada fila em que a saída é um 1 são coletados ($\bar{X}\bar{Y}\bar{Z}$, $\bar{X}Y\bar{Z}$, $X\bar{Y}\bar{Z}$ e $XY\bar{Z}$), e a expressão desejada é a soma destes produtos ($\bar{X}\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XY\bar{Z}$). Portanto, a expressão completa na forma padrão para a estrutura ou circuito desejado é

$$\bar{X}\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XY\bar{Z} = A$$

QUADRO 3.17

Entradas			Saída
X	*Y*	*Z*	*A*
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

QUADRO 3.18

Entradas			Saída	Termos produto
X	*Y*	*Z*	*A*	
0	0	0	1	$\bar{X}\bar{Y}\bar{Z}$
0	0	1	0	$\bar{X}\bar{Y}Z$
0	1	0	1	$\bar{X}Y\bar{Z}$
0	1	1	0	$\bar{X}YZ$
1	0	0	1	$X\bar{Y}\bar{Z}$
1	0	1	0	$X\bar{Y}Z$
1	1	0	1	$XY\bar{Z}$
1	1	1	0	XYZ

Esta expressão pode ser simplificada conforme mostrado abaixo:

$$\begin{aligned}
\bar{X}\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XY\bar{Z} &= A \\
\bar{X}(\bar{Y}\bar{Z} + Y\bar{Z}) + X(\bar{Y}\bar{Z} + Y\bar{Z}) &= A \\
\bar{X}[\bar{Z}(\bar{Y} + Y)] + X[\bar{Z}(\bar{Y} + Y)] &= A \\
\bar{X}\bar{Z} + X\bar{Z} &= A \\
\bar{Z} &= A
\end{aligned}$$

A função pode, portanto, ser realizada por um único inversor conectado à entrada *Z*. A inspeção da tabela da verdade indicará que a saída *A* é sempre igual ao complemento da variável *Z* de entrada.

3.13 COMBINAÇÃO DE PORTAS

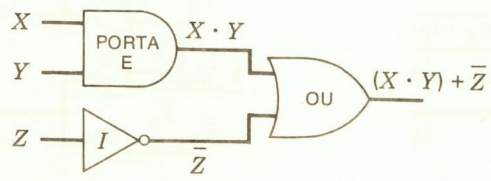
As portas OU, as portas E e os inversores descritos nas Seções 3.3 e 3.4 podem ser interligados para formar *circuitos porta* ou *circuitos lógicos*. (Aqueles que estudam a teoria do

chaveamento denominarão também estas estruturas *circuitos combinacionais.*) A expressão de álgebra booleana correspondente a um determinado circuito lógico pode ser deduzida progredindo-se sistematicamente desde a entrada até a saída com as portas. A Fig. 3.5*(a)* mostra um circuito lógico com três entradas X, Y e Z e uma expressão de saída $(X \cdot Y) + \bar{Z}$. Na Fig. 3.5*(b)* e *(c)* são mostrados um circuito que forma $(X \cdot Y) + (\bar{X} \cdot \bar{Y})$ e uma outra estrutura que forma $(X + Y) \cdot (\bar{X} + \bar{Y})$.

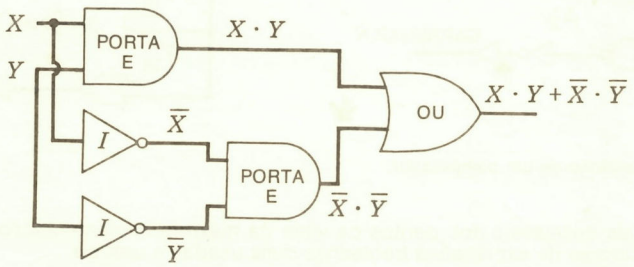
Podemos analisar a operação destes circuitos lógicos usando as expressões da álgebra booleana e podemos, por exemplo, na pesquisa de defeitos de um computador, determinar quais as portas defeituosas pelo exame das entradas no circuito lógico e das saídas, e verificando se as operações booleanas estão sendo efetuadas adequadamente. A escrituração do conjunto de circuitos do computador é feita por meio de diagramas-bloco, como na Fig. 3.6, que mostra uma impressão típica. O uso de álgebra booleana está completamente difundido por toda a indústria de computadores.

3.14 SOMA-DE-PRODUTOS E PRODUTO-DE-SOMAS

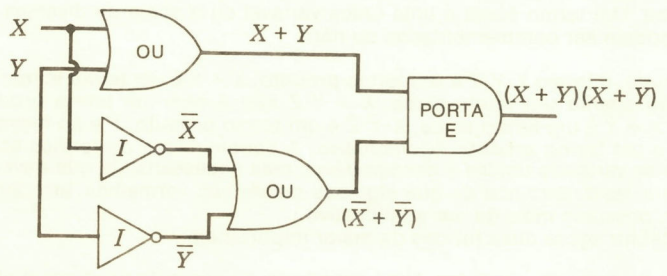
Uma consideração importante no procedimento com circuitos lógicos e seus correspondentes algébricos é a *forma* da expressão da álgebra booleana e a forma resultante do circuito lógico. Certos tipos de expressões da álgebra booleana conduzem a circuitos lógi-



(a)



(b)



(c)

Fig. 3.5 Três redes ou circuitos lógicos.

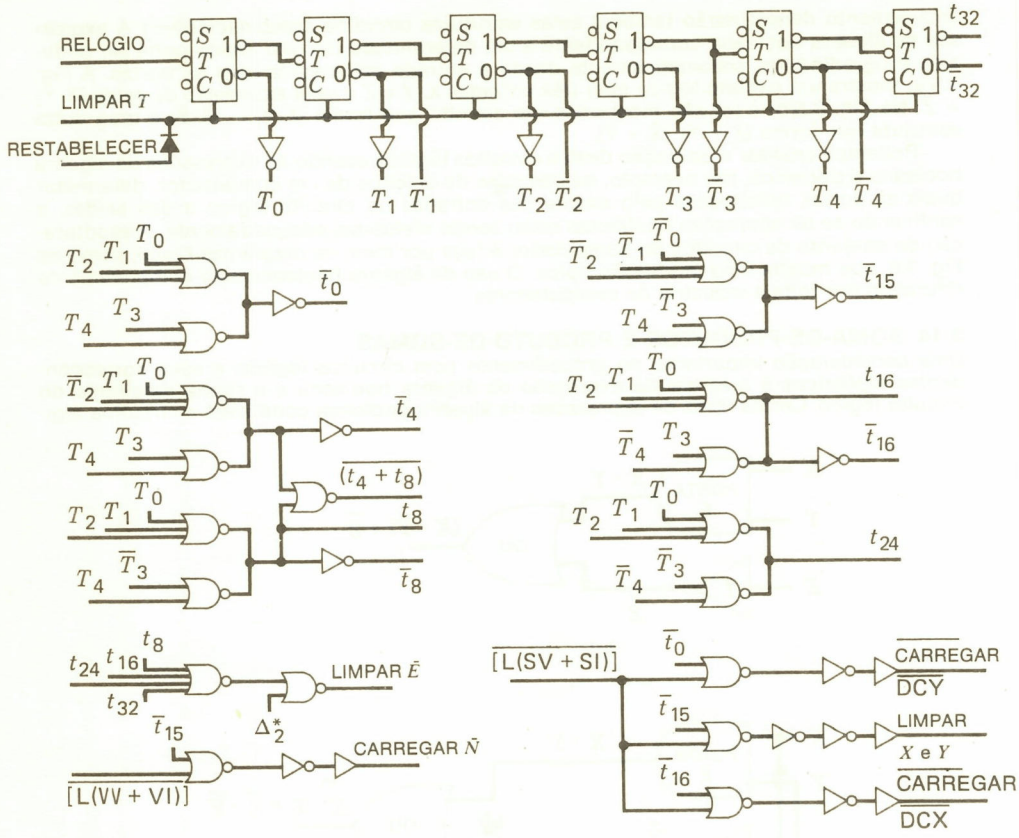


Fig. 3.6 Diagrama-bloco de um computador.

cos que são mais desejáveis dos pontos de vista da máxima implementação. Definiremos agora as duas formas de expressões booleanas mais usadas e usáveis.

Definamos primeiro os termos:

1. *Termo produto.* Um termo produto é uma única variável ou o produto lógico de diversas variáveis. As variáveis podem ser complementadas ou não.
2. *Termo soma.* Um termo soma é uma única variável ou a soma de diversas variáveis. As variáveis podem ser complementadas ou não.

Por exemplo, o termo $X \cdot Y \cdot Z$ é um termo produto; $X + Y$ é um termo soma; X é tanto um termo produto quanto um termo soma: $X + Y \cdot Z$ não é nem um termo produto nem um termo soma; $X + \bar{Y}$ é um termo soma; $X \cdot \bar{Y} \cdot \bar{Z}$ é um termo produto; $\bar{Y}$ é ao mesmo tempo um termo soma e um termo produto. (*Comentário:* A denominação de termos soma e termos produto para as variáveis únicas é desagradável, mas necessária. Já que devemos ser tolerantes com ela, lembremo-nos de que algumas maçãs são vermelhas, arredondadas e lustrosas, isto é, possuem mais de um qualificativo.)

Vamos definir agora duas formas da maior importância:

1. *Expressão de soma-de-produtos.* Uma expressão de soma-de-produtos é um termo produto ou vários termos produto logicamente adicionados.
2. *Expressão de produto-de-somas.* Uma expressão de produto-de-somas é um termo de soma ou vários termos de soma logicamente multiplicados.

Por exemplo, a expressão $\bar{X}\cdot Y + X\cdot\bar{Y}$ é uma expressão de soma-de-produtos; $(X + Y)(\bar{X} + \bar{Y})$ é uma expressão de produto-de-somas. As seguintes são todas expressões de soma-de-produtos:

$$X$$
$$X\cdot Y + Z$$
$$\bar{X}\cdot\bar{Y} + \bar{X}\cdot\bar{Y}\cdot\bar{Z}$$
$$X + Y$$

As seguintes são expressões de produto-de-somas:

$$(X + Y)\cdot(X + \bar{Y})\cdot(\bar{X} + \bar{Y})$$
$$(\bar{X} + Y + Z)\cdot(X + \bar{Y})\cdot(\bar{X} + \bar{Y})$$
$$(\bar{X} + Z)$$
$$\bar{X}$$
$$(X + Y)X$$

Uma importante razão da preferência das expressões de soma-de-produtos ou de produto-de-somas é sua conversão direta em circuitos lógicos de muita precisão.

Em sua mais simples e precisa forma, elas constituem as *estruturas de dois níveis*, que são circuitos para os quais o mais longo trajeto através de que um sinal deve passar da entrada à saída é o de duas portas.

Nota: Na discussão seguinte será suposto que, quando um sinal estiver disponível, seu complemento $\bar{X}$ também está disponível. Em outras palavras, não são requeridos inversores para complementar as entradas. Isto é muito importante e bem realístico, uma vez que a maioria dos sinais vem dos multivibradores biestáveis, que estudaremos mais tarde, e que fornecem tanto uma saída quanto seu complemento.

A Fig. 3.7 mostra diversos circuitos lógicos. A Seção *(a)* da figura mostra as estruturas de soma-de-produtos, e a Seção *(b)* mostra as estruturas de produto-de-somas. Os circuitos lógicos para as expressões de soma-de-produtos em forma "convencional", isto é, expressões com pelo menos dois termos produto e com pelo menos duas variáveis em cada termo produto, entram diretamente num circuito porta E/OU, enquanto as expressões "convencionais" de produto-de-somas entram diretamente nos circuitos porta OU/E, conforme mostrado na figura.

3.15 DEDUÇÃO DAS EXPRESSÕES DE PRODUTO-DE-SOMAS

A seqüência de etapas, descrita na Seção 3.12, permitiu deduzir uma expressão da soma-de-produtos para um determinado circuito. Uma outra técnica, realmente um dual da primeira, forma a expressão requerida como um produto-de-somas. A expressão deduzida desta maneira é constituída, antes da simplificação, de termos consistindo cada um de somas de variáveis tais como $(X + Y + Z)\ldots$. A expressão final é o produto destes termos soma e tem a forma $(X + Y + Z)\,(X + Y + \bar{Z})\ldots(\bar{X} + \bar{Y} + \bar{Z})$.

O método para se chegar à expressão desejada é como segue:

1. Construir uma tabela dos valores de entrada e de saída.
2. Construir uma coluna adicional de termos soma que contenha variáveis complementadas e não-complementadas (dependendo dos valores nas colunas da entrada) para cada fila da tabela. Em cada fila da tabela é formado um termo soma; contudo, neste caso, se o valor da entrada para uma certa variável for 1, a variável será complementada e, se 0, não-complementada.
3. A expressão desejada é o produto dos termos soma provenientes das filas em que a saída é 0.

O uso destas regras será ilustrado por exemplos resolvidos nesta e nas seções seguintes.

O Quadro 3.19 contém os valores de entrada e de saída que descrevem uma função a ser realizada por um circuito lógico.

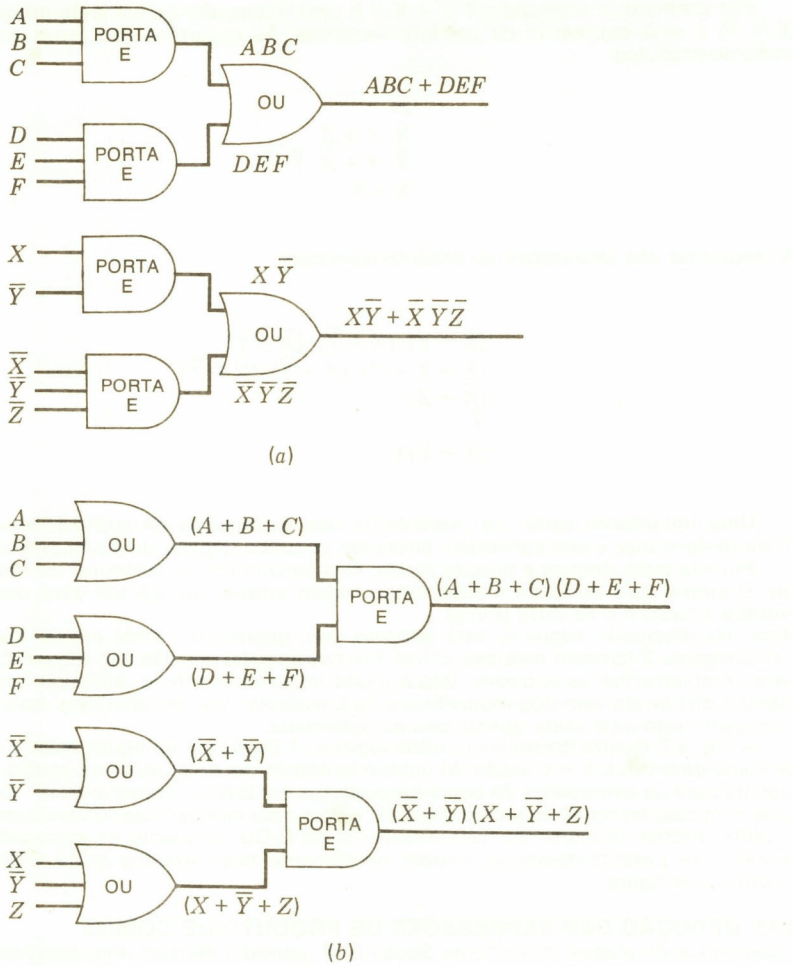


Fig. 3.7 *(a)* Circuitos porta E/OU; *(b)* Circuitos porta OU/E.

Uma coluna que contém as variáveis de entrada em forma de termo soma é agora acrescentada a cada fila. Uma determinada variável é complementada se o valor de entrada da variável for 1 na mesma fila, e não-complementada se o valor for 0 (ver Quadro 3.20). Cada termo soma é, portanto, simplesmente o complemento do termo produto que ocorre na mesma fila na tabela anterior das expressões de soma-de-produtos. Observemos que o termo soma $(\bar{X} + Y)$ na terceira fila do Quadro 3.20 é o complemento do termo produto $X\bar{Y}$ usado na dedução da soma-de-produtos.

Uma expressão de produto-de-somas é agora formada selecionando-se os termos soma para os quais a saída é 0 e multiplicando-os. Neste caso, os 0 aparecem nas segunda e terceira filas, mostrando que a expressão desejada é $(X + \bar{Y})\,(\bar{X} + Y)$. Uma expressão de soma-de-produtos pode ser achada multiplicando-se os dois termos desta expressão, resultando $XY + \bar{X}\bar{Y}$. Neste caso, o mesmo número de portas deve ser requerido para construir circuitos que correspondam tanto à soma-de-produtos quanto ao produto-de-somas.

3.16 DEDUÇÃO DE UMA EXPRESSÃO COM TRÊS VARIÁVEIS DE ENTRADA

Consideremos o Quadro 3.21, que exprime uma relação de entrada/saída para a qual deve ser deduzida uma expressão.

Duas colunas serão acrescentadas desta vez, uma que contém os termos de soma-de-produtos e a outra, os termos de produto-de-somas (ver Quadro 3.22).

As duas expressões podem agora ser escritas da seguinte maneira:

Soma-de-produtos: $(\overline{X}Y\overline{Z}) + (\overline{X}YZ) + (XY\overline{Z}) = A$

Produto-de-somas: $(X + Y + Z)(X + Y + \overline{Z})(\overline{X} + Y + Z)$
$(\overline{X} + Y + \overline{Z})(\overline{X} + \overline{Y} + \overline{Z}) = A$

QUADRO 3.19

Entradas		Saída
X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

QUADRO 3.20

Entradas		Saída	Termos soma
X	Y	Z	
0	0	1	$(X + Y)$
0	1	0	$(X + \overline{Y})$
1	0	0	$(\overline{X} + Y)$
1	1	1	$(\overline{X} + \overline{Y})$

QUADRO 3.21

Entradas			Saída
X	Y	Z	A
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

QUADRO 3.22

Entradas			Saída	Termos produto	Termos soma
X	Y	Z	A		
0	0	0	0	$\overline{X}\overline{Y}\overline{Z}$	$X + Y + Z$
0	0	1	0	$\overline{X}\overline{Y}Z$	$X + Y + \overline{Z}$
0	1	0	1	$\overline{X}Y\overline{Z}$	$X + \overline{Y} + Z$
0	1	1	1	$\overline{X}YZ$	$X + \overline{Y} + \overline{Z}$
1	0	0	0	$X\overline{Y}\overline{Z}$	$\overline{X} + Y + Z$
1	0	1	0	$X\overline{Y}Z$	$\overline{X} + Y + \overline{Z}$
1	1	0	1	$XY\overline{Z}$	$\overline{X} + \overline{Y} + Z$
1	1	1	0	XYZ	$\overline{X} + \overline{Y} + \overline{Z}$

As duas expressões podem ser simplificadas conforme indicado abaixo:

SOMA-DE-PRODUTOS

$$(\bar{X}Y\bar{Z}) + (\bar{X}YZ) + (XY\bar{Z}) = A$$
$$\bar{X}(Y\bar{Z} + YZ) + (XY\bar{Z}) = A$$
$$\bar{X}Y + XY\bar{Z} = A$$
$$Y(\bar{X} + X\bar{Z}) = A$$
$$\bar{X}Y + Y\bar{Z} = A$$

PRODUTO-DE-SOMAS

$$(X + Y + Z)(X + Y + \bar{Z})(\bar{X} + Y + Z)(\bar{X} + Y + \bar{Z})(\bar{X} + \bar{Y} + \bar{Z}) = A$$
$$(X + Y)(\bar{X} + Y)(\bar{X} + \bar{Z}) = A$$
$$Y(\bar{X} + \bar{Z}) = A$$

As duas expressões finais são nitidamente vistas como equivalentes. Observemos, no entanto, que a mais curta expressão de soma-de-produtos, que é $\bar{X}Y + Y\bar{Z}$, requer duas portas E e uma porta OU, como é mostrado na Fig. 3.8, enquanto que a mais curta expressão de produto-de-somas, $Y(\bar{X} + \bar{Z})$, requer somente uma única porta E e uma única porta OU. Em alguns casos, a mínima expressão da soma-de-produtos irá requerer menos elementos lógicos para ser construída e, em outros casos, a construção da mínima expressão de produto-de-somas irá requerer menor número de elementos. Se o critério único é o número de elementos lógicos, torna-se necessário obter tanto uma expressão mínima de soma-de-produtos quanto também uma expressão mínima de produto-de-somas, a fim de comparar as duas. É possível deduzir a expressão da expansão canônica da estrutura ou circuito a ser projetado numa das formas — por exemplo, produto-de-somas — para simplificar a expressão, e então converter a expressão simplificada para a outra forma, usando as leis distributivas. Qualquer simplificação adicional que for exigida pode então ser efetuada. Deste modo, as expressões mínimas em cada forma podem ser obtidas sem a dedução de ambas as expansões canônicas, embora isto possa ser desejável.

As técnicas de simplificação que foram descritas são algébricas e dependem do emprego judicioso dos teoremas que foram apresentados. O problema de se simplificar as expressões booleanas, de modo que seja sempre achada a expressão mais curta, é bastante complexo; no entanto, é possível, por meio do uso repetido de certos algoritmos, deduzir as expressões mínimas de soma-de-produtos e de produto-de-somas. Examinaremos este problema nas seções seguintes.

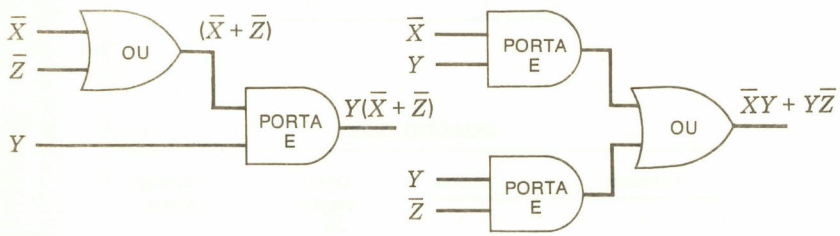


Fig. 3.8 Circuitos para $\bar{X}Y + Y\bar{Z}$ e $Y(\bar{X} + \bar{Z})$.

3.17 PORTAS NÃO-E (NE) E PORTAS NÃO-OU (NOU)

Dois outros tipos de portas, portas NE e portas NOU, são freqüentemente usadas em computadores. Felizmente, a álgebra booleana que foi descrita pode fazer com facilidade a análise da operação destas portas.

Uma porta NE é mostrada na Fig. 3.9. As entradas são A, B e C, e a saída da porta é escrita $\bar{A} + \bar{B} + \bar{C}$. A saída será portanto um 1 se A for um 0 ou B for um 0 ou C for um 0, e a saída será um 0 somente se A E B E C forem todos 1.

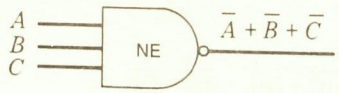
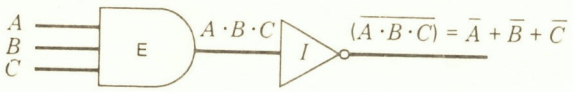


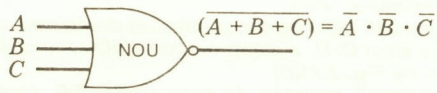
Fig. 3.9 A porta NE.



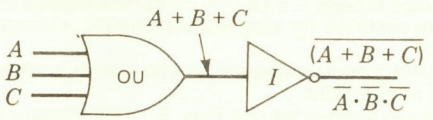
A operação da porta pode ser analisada usando-se o circuito equivalente de diagrama-bloco mostrado na Fig. 3.9, que tem uma porta E seguida de um inversor. Se as entradas forem A, B e C, a saída da porta E será $A \cdot B \cdot C$, e o complemento desta saída é $\overline{(A \cdot B \cdot C)} = \bar{A} + \bar{B} + \bar{C}$, conforme indicado na figura.

A porta NOU pode ser analisada de maneira idêntica. A Fig. 3.10 mostra o símbolo de diagrama-bloco da porta NOU com as entradas A, B, C e a saída $\bar{A}\bar{B}\bar{C}$. Isto significa que a saída da porta NOU será um 1 somente quando todas as três entradas forem 0. Se qualquer entrada representa um 1, a saída de uma porta NOU será portanto um 0.

Abaixo do símbolo de diagrama-bloco da porta NOU na Fig. 3.10 está um circuito equivalente mostrando uma porta OU e um inversor.† As entradas A, B e C são "trabalhadas" pela porta OU, dando $A + B + C$, que é complementada pelo inversor, dando $\overline{(A + B + C)} = \bar{A}\bar{B}\bar{C}$.



(*a*)



(*b*)

ENTRADA			SAÍDA
A	B	C	
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Fig. 3.10 *(a)* Símbolo do diagrama-bloco da porta NOU; *(b)* porta ∀ OU e circuito equivalente de inversor para a porta NOU.

As portas NE de entradas múltiplas podem ser analisadas similarmente. Uma porta NE de quatro entradas, com entradas A, B, C e D, tem uma saída $\bar{A} + \bar{B} + \bar{C} + \bar{D}$, que diz que a saída será um 1 se qualquer uma das entradas for um 0 e será um 0 somente quando todas as quatro entradas forem 1.

Um raciocínio idêntico mostrará que a saída de uma porta NOU de quatro entradas, com entradas A, B, C e D, pode ser representada pela expressão de álgebra booleana $\bar{A}\bar{B}\bar{C}\bar{D}$, que será igual a 1 somente quando A, B, C e D forem todos 0.

Se uma das duas linhas de entrada para uma porta NE de duas entradas contivesse as entradas $A + B$ e a outra contivesse $C + D$, conforme indicado na Fig. 3.11*(a)*, a saída da porta NE seria

$$\overline{[(A + B)(C + D)]} = \bar{A}\bar{B} + \bar{C}\bar{D}$$

†A "bolha", ou pequeno círculo, na saída da porta NE e NOU representa complementação. A NE pode então ser vista como um símbolo de E seguido de um complementador, e a NOU pode ser analisada similarmente.

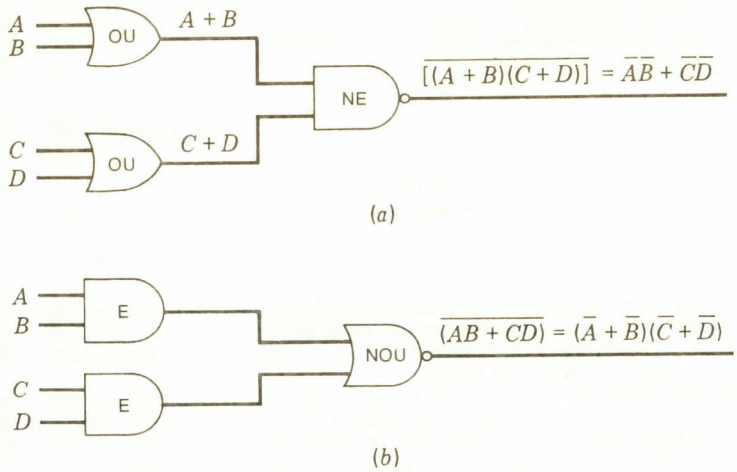


Fig. 3.11 Dois tipos de circuitos lógicos. *(a)* Circuitos porta OU/NE; *(b)* Circuitos porta E/OU.

Podemos comprovar isto notando que a porta NE primeiro "trabalha" as entradas (neste caso, $A + B$ e $C + D$) e, em seguida, complementa.

Se uma das linhas de entrada para uma porta NOU de duas entradas contivesse o sinal $A \cdot B$ e a outra linha de entrada contivesse o sinal $C \cdot D$, a saída da porta NOU seria $(\overline{A \cdot B} + \overline{C \cdot D}) = (\bar{A} + \bar{B})(\bar{C} + \bar{D})$, conforme indicado na Fig. 3.11*(b)*.

Observemos que podemos fazer uma porta E partindo de duas portas NE, usando o artifício mostrado na Fig. 3.12, e fazer uma porta OU de duas entradas partindo de três portas NE, como também é mostrado na mesma figura. Um conjunto de portas NE pode, portanto, ser usado para fazer qualquer estrutura combinacional, substituindo-se os diagramas-bloco, mostrados na Fig. 3.12, pelos blocos E e OU. (A complementação de uma variável, quando necessária, pode ser obtida partindo de uma única porta NE, conectando-se a variável a todas as entradas.)

A porta NOU também pode ser usada para formar qualquer função booleana desejada, e os artifícios fundamentais são mostrados na Fig. 3.13.

Realmente, não é necessário usar as caixas mostradas nas Figs. 3.12 e 3.13 para substi-

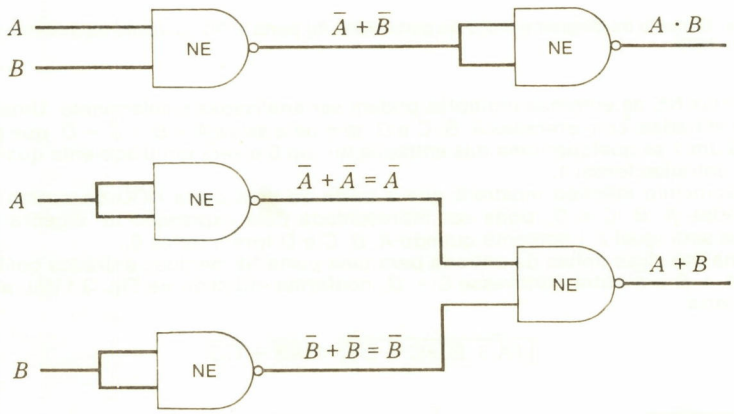


Fig. 3.12 Operações E e OU originárias de portas NE.

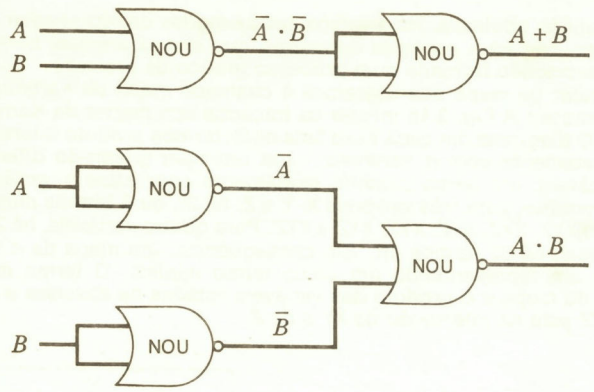


Fig. 3.13 Portas E e OU originárias de portas NOU.

tuir simplesmente as portas E e OU, pois um circuito porta NE de dois níveis produz a mesma função que um circuito porta E/OU de dois níveis, e um circuito porta NOU de dois níveis produz a mesma função que um circuito porta OU/E de dois níveis. Isso é indicado na Fig. 3.14. Comparemos a saída do circuito porta NE com o da Fig. 3.7, por exemplo.

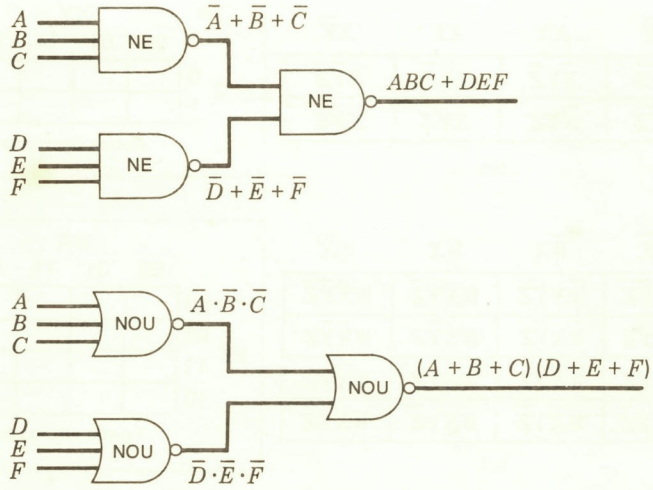


Fig. 3.14 Portas NE e NOU em redes de dois níveis.

*3.18 O MÉTODO DO MAPEAMENTO PARA SIMPLIFICAÇÃO DE EXPRESSÕES

Nas seções anteriores examinamos a dedução de uma expressão de álgebra booleana para uma determinada função com emprego de uma tabela de combinações para listar os valores da função desejada. Para deduzir uma expressão de soma-de-produtos para a função, foi listado um conjunto de termos produto, e aqueles para os quais a função tivesse que ter valor 1 foram selecionados e adicionados logicamente a fim de formar a expressão desejada.

A tabela de combinações proporciona um modo natural e preciso de listar todos os

*Esta seção e as seguintes são opcionais. Os capítulos subseqüentes podem ser lidos sem este material.

valores de uma função booleana. Há diversos outros modos de representar ou listar valores da função, e o uso de certas espécies de mapas, que examinaremos, permitirá também a minimização da expressão formada num processo gráfico de precisão.

O tipo particular de mapa que usaremos é chamado *mapa de Karnaugh* em homenagem a seu idealizador.† A Fig. 3.15 mostra os traçados dos mapas de Karnaugh de duas a quatro variáveis. O diagrama em cada caso lista os 2^n termos produto diferentes que podem ser formados exatamente com n variáveis, cada um num quadrado diferente. Para uma função de n variáveis, um termo produto exatamente com estas n variáveis é chamado *termo mínimo*. Portanto, para três variáveis X, Y e Z, há 2^3, ou 8 termos mínimos diferentes, que são $\bar{X}\bar{Y}\bar{Z}$, $\bar{X}\bar{Y}Z$, $\bar{X}Y\bar{Z}$, $\bar{X}YZ$, $X\bar{Y}\bar{Z}$, $X\bar{Y}Z$, $XY\bar{Z}$ e XYZ. Para quatro variáveis, há 2^4, ou 16 termos, para cinco variáveis, há 32 termos etc. Em conseqüência, um mapa de n variáveis terá 2^n quadrados, cada um representando um único termo mínimo. O termo mínimo em cada caixa, ou célula, do mapa é o produto das variáveis listadas na abscissa e na ordenada da célula. Assim, $\bar{X}YZ$ está na interseção de $\bar{X}Y$ e de Z.

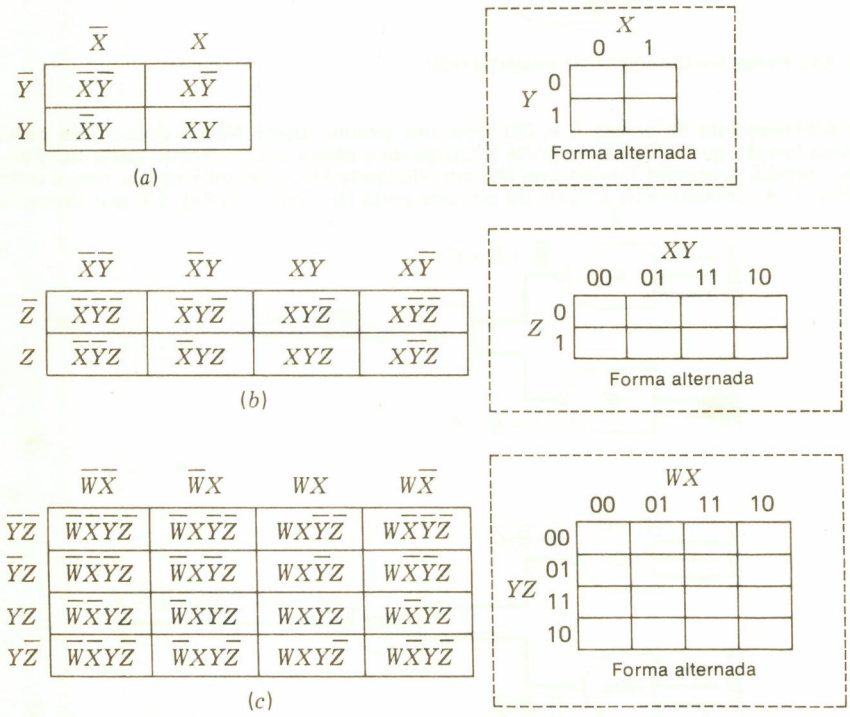


Fig. 3.15 Mapas de Karnaugh para *(a)* duas variáveis, *(b)* três variáveis, e *(c)* quatro variáveis.

Dado um formulário de mapa de Karnaugh, o mapa é preenchido colocando-se 1 nos quadrados, ou células, de cada termo que conduz a uma saída 1.

Como exemplo, consideremos uma função de três variáveis que para os seguintes valores da entrada deve ser 1:

$$X = 0,\ Y = 1,\ Z = 0$$
$$X = 0,\ Y = 1,\ Z = 1$$
$$X = 1,\ Y = 1,\ Z = 0$$
$$X = 1,\ Y = 1,\ Z = 1$$

†Mapas similares são, às vezes, chamados *diagramas de Veitch*.

Esta função é mostrada na Fig. 3.16*(a)* tanto para tabela de combinações quanto para o formulário do mapa de Karnaugh. Uma outra função de quatro variáveis é mostrada na Fig. 3.16*(b)*.

Como um meio de exibição dos valores de uma função, o mapa de Karnaugh é conveniente e proporciona algum "sentimento" da função devido à sua apresentação gráfica. Seu principal emprego, no entanto, é devido ao arranjo das células. Cada célula difere de sua adjacente por ter exatamente uma variável complementada no termo mínimo de uma célula que não é complementada no termo mínimo na célula adjacente.

Como um exemplo disto, consideremos o mapa de quatro variáveis na Fig. 3.16 e o

X	Y	Z	VALORES DA FUNÇÃO
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

	$\bar{X}\bar{Y}$	$\bar{X}Y$	XY	$X\bar{Y}$
$\bar{Z}$	0	1	1	0
Z	0	1	1	0

(a)

W	X	Y	Z	VALORES DA FUNÇÃO
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

	$\bar{W}\bar{X}$	$\bar{W}X$	WX	$W\bar{X}$
$\bar{Y}\bar{Z}$	1	0	0	0
$\bar{Y}Z$	1	1	0	1
YZ	0	0	1	0
$Y\bar{Z}$	0	1	0	1

(b)

Fig. 3.16 Dois mapas de Karnaugh. *(a)* Mapa da expressão booleana $\bar{X}Y\bar{Z} + \bar{X}YZ + XY\bar{Z} + XYZ$; *(b)* mapa de função de quatro variáveis.

termo mínimo $\bar{W}X\bar{Y}Z$. Há quatro células adjacentes à célula que contém $\bar{W}X\bar{Y}Z$. Estas contêm (1) $WX\bar{Y}Z$, que difere na variável W; (2) $\bar{W}\bar{X}\bar{Y}Z$ em X; (3) $\bar{W}XYZ$, que difere de $\bar{W}X\bar{Y}Z$ em Y; e (4) $\bar{W}X\bar{Y}\bar{Z}$, que difere de $\bar{W}X\bar{Y}Z$ em Z.

Um artifício deve ser observado neste ponto. Os mapas devem ser considerados "enrolados", ou contínuos, de modo que os bordos superior e inferior ou os bordos laterais direito e esquerdo estão tocando-se. Para o mapa de três variáveis, consideremos que os bordos lateral esquerdo e lateral direito estejam tocando-se, de modo que o mapa é visto estar enrolado como um arco horizontalmente na página. Isso coloca a célula que contém $\bar{X}\bar{Y}\bar{Z}$ próxima a $X\bar{Y}\bar{Z}$, bem como a $\bar{X}\bar{Y}Z$ e $\bar{X}Y\bar{Z}$. Além disso, para este mapa, ele coloca $\bar{X}\bar{Y}Z$ próximo a $X\bar{Y}Z$, o qual toca por causa do enrolamento, bem como a $\bar{X}YZ$ e $\bar{X}\bar{Y}\bar{Z}$.

Para o mapa de quatro variáveis, o mapa é enrolado de modo que o bordo superior toque o bordo inferior, e o lateral esquerdo toque o lateral direito. O toque do superior e do inferior coloca $\bar{W}X\bar{Y}\bar{Z}$ próximo a $\bar{W}XY\bar{Z}$, e os bordos laterais esquerdo e lateral direito ao se tocarem fazem com que $W\bar{X}YZ$ se coloque próximo a $\bar{W}\bar{X}YZ$.

Uma boa regra para lembrarmos é que há dois termos mínimos adjacentes a um determinado termo mínimo num mapa de duas variáveis; há três termos mínimos próximos a um determinado termo mínimo num mapa de três variáveis; há quatro termos mínimos próximo a um certo termo mínimo num mapa de quatro variáveis etc.

3.19 SUBCUBOS E COBERTURA

Um *subcubo* é definido como um conjunto de exatamente 2^m células adjacentes contendo 1. Para $m = 0$, o subcubo consiste de uma única célula (e, portanto, de um único termo mínimo). Para $m = 1$, um subcubo consiste de duas células adjacentes; por exemplo, as células que contêm $\bar{X}\bar{Y}Z$ e $\bar{X}YZ$ formam um subcubo conforme indicado na Fig. 3.17*(a)*, assim como $X\bar{Y}\bar{Z}$ e $\bar{X}\bar{Y}\bar{Z}$ (já que o mapa é enrolado).

Para $m = 2$, o subcubo tem quatro células adjacentes, e vários de tais subcubos são mostrados na Fig. 3.17*(c)*. Observemos que omitimos aqui os 0 por clareza e preenchemos somente os 1 da função. Este procedimento será mantido.

Finalmente, um subcubo que contém oito células (para $m = 3$) é mostrado na Fig. 3.17*(d)*.

(Às vezes, é conveniente designar um subcubo que contém duas células por cubo-2, um subcubo de quatro células por cubo-4, um subcubo de oito células por cubo-8 etc., e isto será freqüentemente usado.)

Para demonstrar o uso dos mapas e subcubos na minimização das expressões de álgebra booleana, necessitamos examinar uma regra de álgebra booleana:

$$AX + A\bar{X} = A$$

Na equação acima, a variável A pode permanecer para mais do que uma variável. Por exemplo, seja $A = WY$; temos então

$$(WY)X + (WY)\bar{X} = (WY)$$

Ou seja, $A = W\bar{Y}\bar{Z}$; temos então

$$W\bar{Y}\bar{Z}X + W\bar{Y}\bar{Z}\bar{X} = W\bar{Y}\bar{Z}$$

A regra básica pode ser provada por fatoração

$$AX + A\bar{X} = A(X + \bar{X})$$

Então, já que $X + \bar{X} = 1$, temos

$$AX + A\bar{X} = A(X + \bar{X}) = A \cdot 1 = A$$

Cada um dos exemplos dados pode ser verificado similarmente; por exemplo,

$$W\bar{Y}\bar{Z}\bar{X} + W\bar{Y}\bar{Z}X = W\bar{Y}\bar{Z}(\bar{X} + X) = W\bar{Y}\bar{Z} \cdot 1 = W\bar{Y}\bar{Z}$$

Esta regra pode ser estendida. Consideremos

$$WX\bar{Y}\bar{Z} + WX\bar{Y}Z + WXY\bar{Z} + WXYZ$$

Aqui há quatro termos, cada um com duas variáveis WX inalteráveis, enquanto que as outras duas variáveis Y e Z assumem todos os valores possíveis. O termo WX é então igual à soma dos outros termos, pois

$$\begin{aligned} W\bar{X}\bar{Y}\bar{Z} + W\bar{X}\bar{Y}Z + WX\bar{Y}\bar{Z} + WXYZ &= W\bar{X}\bar{Y}(\bar{Z} + Z) + WXY(Z + \bar{Z}) \\ &= W\bar{X}\bar{Y} + WXY \\ &= WX(\bar{Y} + Y) \\ &= WX \end{aligned}$$

Portanto, WX pode ser substituído pelos outros quatro termos numa expressão sem modificar os valores que a expressão assume para quaisquer valores de entrada das variáveis, isto é, $WX = W\bar{X}\bar{Y}\bar{Z} + W\bar{X}\bar{Y}Z + WXYZ$.

Num mapa, as operações algébricas acima podem ser facilmente efetuadas. Já que um subcubo de duas células tem ambas as células diferindo de uma única variável, um termo produto exatamente com aquelas variáveis que não diferirem cobrirá (pode ser substituído por) os dois termos mínimos nas duas células.

Consideremos o subcubo de duas células para $\bar{X}\bar{Y}Z$ e $\bar{X}YZ$ no mapa de três variáveis na Fig. 3.17*(a)*. O único termo produto $\bar{X}Z$ é igual à soma desses dois termos mínimos; isto é,

$$\bar{X}\bar{Y}Z + \bar{X}YZ = \bar{X}Z$$

Similarmente, as duas células que contêm termos mínimos $\bar{X}\bar{Y}\bar{Z}$ e $X\bar{Y}\bar{Z}$ formam um subcubo de duas células, conforme indicado na Fig. 3.17*(a)*, a partir do qual formamos $\bar{Y}\bar{Z}$, que pode ser substituído por $\bar{X}\bar{Y}\bar{Z} + X\bar{Y}\bar{Z}$ numa expressão.

Similarmente, o subcubo de quatro células num mapa de três variáveis [Fig. 3.17*(b)*] com termos $\bar{X}\bar{Y}\bar{Z}$, $\bar{X}\bar{Y}Z$, $X\bar{Y}\bar{Z}$, $X\bar{Y}Z$ tem uma única variável inalterada Y; e achamos $\bar{Y} = \bar{X}\bar{Y}\bar{Z} + \bar{X}\bar{Y}Z + X\bar{Y}\bar{Z} + X\bar{Y}Z$.

Em geral, um subcubo com 2^m células num mapa de n variáveis terá $n - m$ variáveis, que são as mesmas em todos os termos mínimos, e m variáveis, que assumem todas as combinações possíveis de serem complementadas ou não-complementadas. Assim, para um mapa de quatro variáveis em que $m = 3$, quaisquer oito células adjacentes que formem um subcubo terão $4 - 3 = 1$ variável inalterada e três variáveis que mudam a complementação de célula para célula. Então, um subcubo de oito células num mapa de quatro variáveis pode ser usado para determinar uma única variável que pode ser substituída pela soma do termo mínimo em todas as oito células.

Por exemplo, na Fig. 3.17*(d)*, achamos um subcubo de oito células com os termos mínimos $\bar{W}\bar{X}\bar{Y}Z$, $\bar{W}\bar{X}YZ$, $\bar{W}X\bar{Y}Z$, $\bar{W}XYZ$, $W\bar{X}\bar{Y}Z$, $W\bar{X}YZ$, $WX\bar{Y}Z$ e $WXYZ$, e a soma destes será achada equivalente a Z.

O conjunto de termos mínimos numa expressão, no entanto, não forma necessariamente um único subcubo, e há dois casos a serem tratados. Denominamos *subcubo máximo* o maior subcubo que pode ser achado ao redor de um certo termo mínimo. Então, os dois casos são:

1. Todos os subcubos máximos se interseccionam; isto é, nenhuma célula num subcubo máximo constitui parte de um outro subcubo máximo. Diversos exemplos são mostrados na Fig. 3.18.
2. Os subcubos máximos se interseccionam; isto é, as células num subcubo máximo estão também em outros subcubos máximos. A Fig. 3.19 mostra exemplos deste caso.

O caso 1 é o que se trata com mais facilidade. Nele, os termos produto correspondentes aos subcubos máximos são selecionados, e a soma destes termos forma uma expressão mínima de soma-de-produtos.

(Na teoria da comutação, o termo produto correspondente a um subcubo máximo é chamado *prime implicant*.)

A Fig. 3.18 mostra um exemplo com quatro variáveis; há um subcubo de duas células contendo $WXYZ$ e $WX\bar{Y}Z$ que podem ser cobertas pelo termo produto WXZ. Há também um subcubo de quatro células contendo $\bar{W}\bar{X}\bar{Y}Z$, $\bar{W}\bar{X}YZ$, $W\bar{X}\bar{Y}Z$ e $W\bar{X}YZ$ que podem ser cobertos por $\bar{W}\bar{X}$. A expressão mínima é, portanto, $\bar{W}\bar{X} + WXZ$.

Dois outros exemplos são mostrados nas Fig. 3.18*(b)* e *(c)*. Em cada caso, os subcubos não interseccionam ou dividem as células, e assim o termo produto *(prime implicant)* que

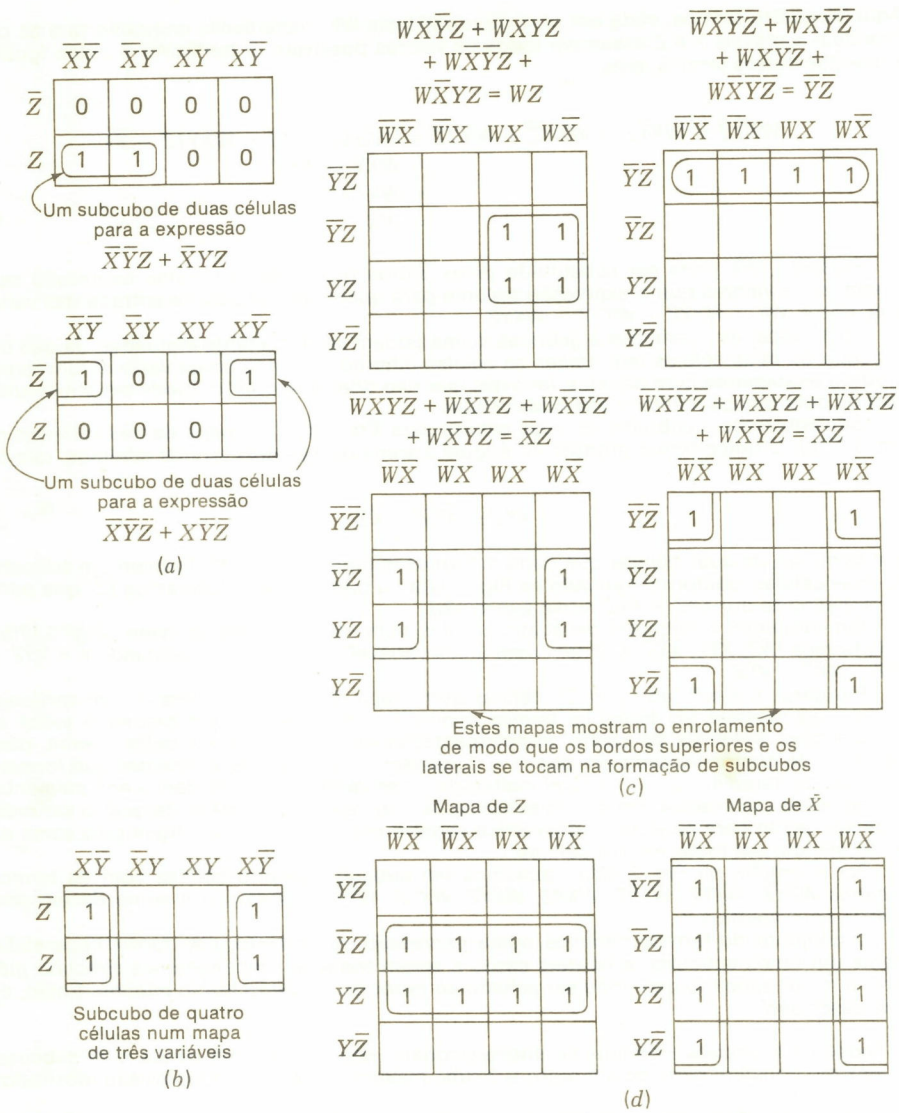


Fig. 3.17 Subcubos com dois, quatro e oito células. Células vazias são supostas conter 0.

corresponde a um determinado subcubo máximo pode ser deduzido prontamente, e a soma destes termos produto para um certo mapa forma a expressão mínima.

Quando os subcubos se interseccionam, a situação pode ser mais complicada. O primeiro princípio a observar é: *Cada célula que contém um 1 (isto é, cada célula-1) deve estar contida em algum subcubo que é selecionado.*

A Fig. 3.19*(a)* mostra um mapa com um par de subcubos que se interseccionam e mais um outro subcubo. A expressão mínima é, neste caso, formada simplesmente pela adição dos três termos produto associados aos três subcubos máximos. Observemos que um único termo, $\bar{W}XYZ$, é dividido entre dois subcubos e, por causa disso, está efetivamente duas vezes na expressão mínima. Isto é permissível em virtude da regra idempotente da

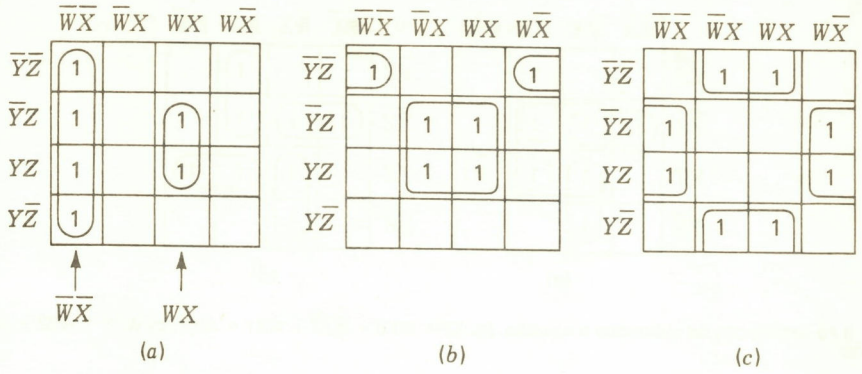


Fig. 3.18 Mapas com subcubos deslocados. *(a)* Mapa de $\bar{W}\bar{X} + WXZ$; *(b)* mapa de $XZ + \bar{X}\bar{Y}\bar{Z}$; *(c)* mapa de $X\bar{Z} + \bar{X}Z$.

álgebra booleana, $A + A = A$, que estabelece que a repetição de termos não muda a equivalência funcional.

Dois outros exemplos de subcubos máximos que se interseccionam são mostrados na Fig. 3.19*(b)* e *(c)*.

Enquanto os subcubos máximos puderem ser prontamente achados e não houver opções na seleção do subcubo, o problema da minimização é direto. Em alguns casos, o problema é mais complicado. A Fig. 3.20 mostra uma expressão com um subcubo de quatro células no centro do mapa, que é máximo. A seleção deste subcubo máximo não conduz a uma expressão mínima, no entanto, porque as quatro células com 1 ao redor deste subcubo também devem ser cobertas. Em cada caso, estas células-1 podem ter uma única célula adjacente e, portanto, constituírem parte de subcubos máximos que consistem de células-2. Na Fig. 3.20*(a)*, $\bar{W}\bar{X}\bar{Y}Z$ está numa célula adjacente apenas a $\bar{W}X\bar{Y}Z$ e, assim, forma parte de uma célula-2. A Fig. 3.20*(b)* mostra uma outra maneira de formar subcubos para o mapa, e isto conduz à expressão mínima $W\bar{X}\bar{Y} + WYZ + \bar{W}XY + \bar{W}\bar{Y}Z$.

A pesquisa de expressões mínimas para tais mapas não é direta. As regras a serem seguidas são:

1. Começar pelas células que não são adjacentes a outras células. Os termos mínimos nestas células não podem ser encurtados e devem ser usados como estiverem.
2. Achar todas as células que sejam adjacentes apenas a uma outra célula. Estas formam subcubos de duas células cada um.
3. Achar as células que conduzem a subcubos máximos de quatro células. Achar então subcubos de oito células etc.
4. A expressão mínima é formada a partir de uma coleção de tão poucos cubos quanto possível, cada um dos quais é tão grande quanto possível, isto é, cada um dos quais é um subcubo máximo.

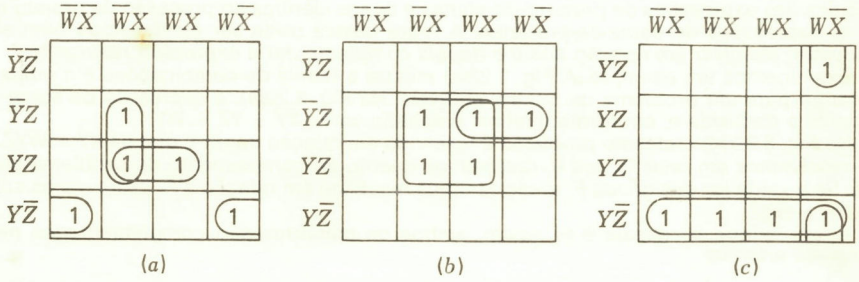


Fig. 3.19 Interseção de subcubos. *(a)* $\bar{W}XZ + XYZ + \bar{X}Y\bar{Z}$; *(b)* $XZ + W\bar{Y}Z$; *(c)* $\bar{Y}\bar{Z} + W\bar{X}\bar{Z}$.

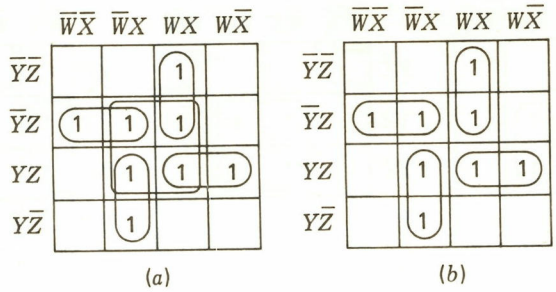


Fig. 3.20 Interseção de subcubos e solução. *(a)* $XZ + WYZ + \bar{W}\bar{Y}Z + \bar{W}XY + WX\bar{Y}$; *(b)* $WX\bar{Y} + WYZ + \bar{W}XY + \bar{W}\bar{Y}Z$.

A Fig. 3.21 mostra um exemplo de um mapa difícil. Os subcubos máximos podem ser selecionados de diversas maneiras de modo que todas as células sejam cobertas. A figura mostra três mapas, dos quais somente um conduz a uma expressão mínima. A prática com vários mapas conduzirá à habilidade em pesquisar expressões mínimas. Nos casos difíceis, há técnicas algébricas que garantem uma expressão mínima, e estas podem ser achadas em Bartee, Lebow e Reed, Birkhoff e Bartee, McCluskey e outros na Bibliografia.

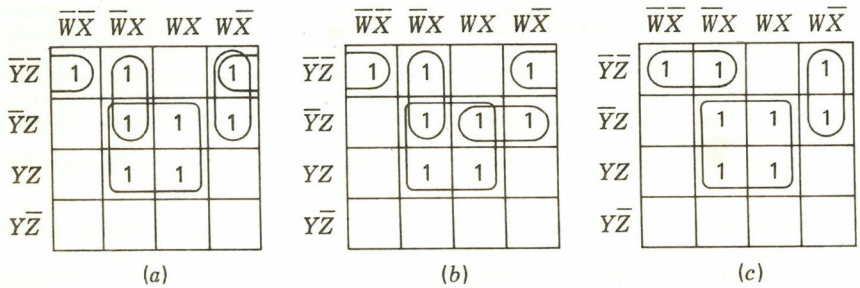


Fig. 3.21 Três coberturas do mesmo mapa. *(a)* $XZ + X\bar{Y}\bar{Z} + W\bar{X}\bar{Y} + \bar{W}X\bar{Y}$; *(b)* $XZ + X\bar{Y}\bar{Z} + \bar{W}X\bar{Y} + WY\bar{Z}$; *(c)* $XZ + \bar{W}\bar{Y}\bar{Z} + W\bar{X}\bar{Y}$.

3.20 EXPRESSÕES DE PRODUTO-DE-SOMAS — *DON'T-CARES* (SAÍDAS NÃO-ESPECIFICADAS)

A técnica das expressões de produto-de-somas é quase idêntica ao processo de projeto que usa as expressões de soma-de-produtos. A regra básica pode ser estabelecida bem simplesmente: *Resolver em relação aos 0 e depois complementar a expressão resultante.*

Examinemos um exemplo. A Fig. 3.22*(a)* mostra a tabela de combinações e o mapa de Karnaugh para um problema de quatro variáveis. Na Fig. 3.22*(a)*, a expressão de soma-de-produtos é deduzida e, em forma mínima, é achada como $\bar{X}\bar{Y} + YZ + WY$.

Na Fig. 3.22*(b)*, o mesmo problema é resolvido em relação aos 0, o que dá $X\bar{Y} + \bar{W}YZ$. Já que resolvemos em relação aos 0, resolvemos quanto ao complemento do problema desejado. Se a saída for designada F, então teremos resolvido em relação a $\bar{F}$. Escrevemos então $\bar{F} = X\bar{Y} + \bar{W}YZ$.

Agora, o que se deseja é F; assim, ambos os membros são complementados nesta expressão e temos

$$F = (\bar{X} + Y)(W + \bar{Y} + Z)$$

Esta expressão está em forma de produto-de-somas e é um tanto mais simples do que a expressão de soma-de-produtos.

Se as expressões de soma-de-produtos e de produto-de-somas forem igualmente fáceis de implementar, então um determinado problema deve ser resolvido em ambas as formas e escolhida a solução mais simples. Não há nenhum modo de se determinar qual será a mais simples, a não ser um domínio completo do problema.

Há uma outra situação freqüentemente encontrada em que certas saídas não são especificadas no problema. Tais saídas são chamadas saídas *don't-care* (saídas não-especificadas), pois o projetista não se preocupa em saber quais as saídas correspondentes a essas entradas particulares.

A Fig. 3.23*(a)* mostra tal problema com 6 dos 16 valores possíveis de saída listados como saídas não-especificadas (*don't-cares*). Isto constitui uma parte de um tradutor de BCD, e assim estas seis combinações de entrada particulares nunca são usadas.

Uma vez que os valores de saída não-especificada são irrelevantes, eles podem ser preenchidos com 1 e 0 de qualquer maneira que seja vantajosa. A Fig. 3.23*(a)* mostra um mapa de Karnaugh da tabela de combinações na figura, com saídas não-especificadas colocadas nos lugares apropriados. Ao resolver esta tabela, uma saída não-especificada pode ser usada ou como um 1 ou como um 0; assim, as saídas não-especificadas são usadas para ampliar ou completar um subcubo sempre que possível, mas, do contrário, são ignoradas (isto é, feitas 0). *As saídas não-especificadas não necessitam ser cobertas pelos subcubos selecionados, mas são usadas somente para ampliar os subcubos que contêm 1, que devem ser cobertos.*

Na Fig. 3.23*(a)*, a fileira vertical de quatro saídas não-especificadas na coluna *WX* é para uso em duas vezes, uma vez para preencher, ou para completar, a fila superior de 1 e uma vez para completar a terceira fila. Estes subcubos dão os termos $\bar{Y}\bar{Z}$ e YZ; assim, a mínima expressão de soma-de-produtos é $\bar{Y}\bar{Z} + YZ$. Observemos que, se todas as saídas não-especificadas fossem feitas 0, a solução requereria mais termos.

Um outro problema é desenvolvido na Fig. 3.23*(b)*. Para este problema, a solução é $W\bar{Z} + W\bar{Y}$. Observemos que duas das saídas não-especificadas são feitas 0. Com efeito, as saídas não-especificadas são escolhidas de modo que elas conduzam à melhor solução.

ENTRADAS				VALORES DA FUNÇÃO
W	X	Y	Z	
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

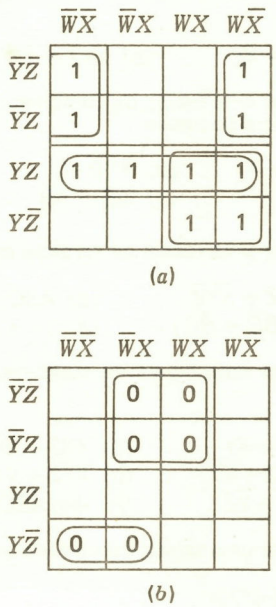
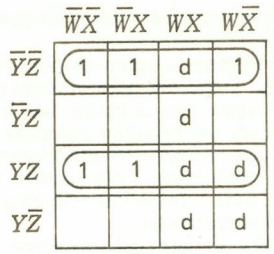
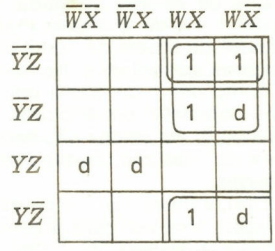


Fig. 3.22 Resolução para produto de somas. *(a)* $\bar{X}\bar{Y} + YZ + WY$; *(b)* $(\bar{X} + Y)(W + \bar{Y} + Z)$.

W	X	Y	Z	
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	d
1	0	0	1	d
1	1	0	0	d
1	1	0	1	d
1	1	1	0	d
1	1	1	1	d



(a) $\overline{Y}\overline{Z} + YZ$



(b) $W\overline{Z} + W\overline{Y}$

Fig. 3.23 Condições com saídas não-especificadas. *(a)* Mapa de tabela com saídas não-especificadas; *(b)* resolução de um outro mapa com saídas não-especificadas.

EXERCÍCIOS

1. Preparar uma tabela de combinações para as seguintes expressões de álgebra booleana:

   $X\overline{Y} + XY$ (b) $X\overline{Y}Z + \overline{Y}\overline{Z}$

2. Preparar uma tabela de combinações (tabela da verdade) para as seguintes expressões de álgebra booleana:

   (a) $\overline{X} + Y$ (b) $A \cdot \overline{B} \cdot C$ (c) $A + \overline{B} + \overline{C}$
   (d) $\overline{A} \cdot \overline{B}$ (e) $\overline{A}B + A\overline{B}$ (f) $A + \overline{A}B$

3. Preparar uma tabela da verdade para as seguintes expressões booleanas:

   (a) $XYZ + \overline{X}\overline{Y}\overline{Z}$ (b) $ABC + A\overline{B}\overline{C} + \overline{A}\overline{B}\overline{C}$
   (c) $A(B\overline{C} + \overline{B}C)$ (d) $(A + B)(A + C)(\overline{A} + \overline{B})$

4. Preparar uma tabela de combinações para as seguintes expressões de álgebra booleana:

   (a) $\overline{X}\overline{Y} + XY$ (b) $XYZ + \overline{X}\overline{Y}\overline{Z}$
   (c) $\overline{X}YZ + \overline{X}\overline{Y}$ (d) $\overline{X}\overline{Y}\overline{Z} + XYZ + \overline{X}YZ$
   (e) $\overline{X}\overline{Z} + \overline{Y}\overline{Z}$ (f) $AB(\overline{A}\overline{B}C + \overline{B}C)$

5. Preparar uma tabela da verdade para as seguintes expressões booleanas:

   (a) $A\overline{B} + \overline{A}B$ (b) $A\overline{B} + B\overline{C}$
   (c) $A\overline{C} + AC$ (d) $\overline{A}\overline{B}C + AB\overline{C} + \overline{A}BC$
   (e) $AB(A\overline{B}C + A\overline{B}\overline{C} + AB\overline{C})$

6. Preparar uma tabela de combinações para as seguintes expressões de álgebra booleana:

(a) $X(\overline{Y}+\overline{Z})+XY$ (b) $X\overline{Y}(Z+YZ)+\overline{Z}$
(c) $[X(Y+\overline{Y})+\overline{X}(Y+\overline{Y})]\cdot Z$ (d) $AB(\overline{A}B+\overline{A}\overline{B})$
(e) $A((B+C)+\overline{C})$ (f) $\overline{A}\overline{B}\overline{C}(\overline{A}\overline{B}\overline{C}+\overline{A}\overline{B}C)$

7. Simplificar as seguintes expressões e desenhar um diagrama-bloco do circuito para cada expressão simplificada, usando portas E e OU. Admitir que as entradas sejam provenientes de multivibradores biestáveis *(flip-flops)*.

(a) $A\overline{B}\overline{C}+\overline{A}\overline{B}\overline{C}+\overline{A}B\overline{C}+\overline{A}\overline{B}C$
(b) $ABC+\overline{A}BC+A\overline{B}C+AB\overline{C}+A\overline{B}\overline{C}+\overline{A}B\overline{C}+\overline{A}\overline{B}\overline{C}$
(c) $A(A+B+C)(\overline{A}+B+C)(A+\overline{B}+C)(A+B+\overline{C})$
(d) $(A+B+C)(A+\overline{B}+\overline{C})(A+B+\overline{C})(A+\overline{B}+C)$

8. Simplificar as expressões no Exercício 4 e desenhar diagramas-bloco das redes ou circuitos lógicos para suas expressões simplificadas, usando portas E, portas OU e inversores.
9. Simplificar as seguintes expressões:

(a) $ABC(AB\overline{C}+A\overline{B}\overline{C}+\overline{A}BC)$ (b) $AB+A\overline{B}+\overline{A}C+\overline{A}\overline{C}$
(c) $XY+XYZ+XY\overline{Z}+\overline{X}YZ$ (d) $XY(\overline{X}YZ+X\overline{Y}\overline{Z}+\overline{X}\overline{Y}\overline{Z})$

10. Simplificar as expressões no Exercício 6 e desenhar diagramas-bloco das redes ou circuitos lógicos para suas expressões simplificadas, usando portas E, portas OU e inversores.
11. Formar os complementos das seguintes expressões. Por exemplo, o complemento de $(XY + XZ)$ é igual a

$$(\overline{X}\overline{Y}+\overline{X}\overline{Z})=(\overline{X}+\overline{Y})(\overline{X}+\overline{Z})=\overline{X}+\overline{Y}\overline{Z}$$

(a) $(A+BC+AB)$ (b) $(A+B)(B+C)(A+C)$
(c) $AB+\overline{B}C+C\overline{D}$ (d) $AB(\overline{C}D+\overline{B}C)$
(e) $A(B+C)(\overline{C}+\overline{D})$

12. Complementar as seguintes expressões (como no Exercício 11).

(a) $\overline{X}\overline{Y}+XY$ (b) $XYZ+\overline{X}\overline{Y}$
(c) $X(\overline{Y}+\overline{Z})$ (d) $X(YZ+\overline{Y}\overline{Z})$
(e) $XY(\overline{Y}Z+XZ)$ (f) $XY+\overline{X}\overline{Y}(YZ+\overline{X}\overline{Y})$

13. Provar os dois teoremas básicos de De Morgan usando a prova pela indução perfeita.
14. Provar as seguintes regras usando a prova por indução perfeita.

(a) $X\overline{Y}+XY=X$
(b) $X+\overline{X}Y=X+Y$

15. Converter as seguintes expressões à forma de soma-de-produtos.

(a) $(A+B)(\overline{B}+C)(\overline{A}+C)$
(b) $(\overline{A}+C)(\overline{A}+\overline{B}+\overline{C})(A+\overline{B})$
(c) $(A+C)(A\overline{B}+AC)(\overline{A}\overline{C}+\overline{B})$

16. Converter as seguintes expressões à forma de soma-de-produtos.

(a) $(\overline{A}+B)(\overline{C}+\overline{B})$ (b) $\overline{A}B(\overline{B}C+\overline{B}\overline{C})$
(c) $(A+B\overline{C})(\overline{A}B+\overline{A}\overline{B})$ (d) $A\overline{B}(A\overline{B}\overline{C}+\overline{A}C)$
(e) $(\overline{A}+B)(A\overline{C}+(B+C))$ (f) $(\overline{A}+C)(AB+AB+AC)$

17. Que regra é o dual da regra 12 no Quadro 3.10?
18. Dar um dual da regra $X+\overline{X}Y=X+Y$.
19. Multiplicar os seguintes termos soma, formando uma expressão de soma-de-produtos em cada caso. Simplificar enquanto multiplica quando for possível.

(a) $(A+C)(B+D)$
(b) $(A+C+D)(B+D+C)$
(c) $(A\overline{B}+C+DC)(AC+\overline{B}C+D)$
(d) $(A\overline{B}+\overline{A}B+A\overline{C})(\overline{A}\overline{B}+A\overline{B}+A\overline{C})$

20. Converter as seguintes expressões à forma de produto-de-somas.

(a) $A+AB+\overline{A}C$ (b) $BC+\overline{A}\overline{B}$
(c) $AB(\overline{B}+\overline{C})$ (d) $\overline{A}\overline{B}(\overline{B}\overline{C}+B\overline{C})$
(e) $(A+B+C)(AB+AC)$ (f) $(\overline{A}+\overline{B})A\overline{B}C$

21. Escrever a expressão booleana (sob a forma de soma-de-produtos) para um circuito lógico que terá uma saída 1 quando $X=0, Y=0, Z=1$, e $X=1, Y=1, Z=0$; e uma saída 0 para todos os outros estados da entrada. Desenhar o diagrama-bloco para este circuito, admitindo que as entradas sejam originárias de multivibradores biestáveis *(flip-flops)*.
22. Deduzir uma expressão de álgebra booleana e desenhar um diagrama-bloco para uma rede lógica que terá saída 1 somente quando $X=1, Y=1, Z=0$; $X=0, Y=0, Z=0$.
23. Provar a regra 17 no Quadro 3.10 multiplicando o membro à esquerda e usando a regra 15 para simplificar.
24. Provar o seguinte teorema usando as regras no Quadro 3.10.

$$(X+Y)(X+\overline{Y})=X$$

25. Escrever a expressão booleana (em forma de soma-de-produtos) para uma rede lógica que terá uma saída 1 quando $X=0, Y=1, Z=1$; $X=1, Y=1, Z=0$; $X=1, Y=1, Z=0$; e $X=1, Y=1, Z=1$. O circuito terá uma saída 0 para todos os outros conjuntos de valores da entrada. Simplificar a expressão deduzida e desenhar um diagrama-bloco para a expressão simplificada.
26. Deduzir a expressão de álgebra booleana para uma rede lógica que terá saídas 0 somente quando $X=1, Y=1, Z=0$; $X=0, Y=0, Z=0$; $X=1, Y=0, Z=0$. As saídas devem ser 1 para todos os outros casos.
27. Provar a regra 18 no Quadro 3.10 usando a prova pela indução perfeita.

QUADRO 3.23

Entradas			Saída	Saída	Saída
X	Y	Z	F_1	F_2	F_3
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	1	1
1	1	1	1	0	1

28. Desenvolver expressões de soma-de-produtos e de produto-de-somas para F_1, F_2 e F_3 no Quadro 3.23.
29. Desenvolver tanto expressões de soma-de-produtos quanto de produto-de-somas que descrevem o Quadro 3.24 e depois simplificar ambas as expressões. Desenhar um diagrama-bloco para o conjunto de circuitos lógicos que corresponde às expressões simplificadas, usando apenas portas NE para a soma-de-produtos e portas NOU para o produto-de-somas.
30. Desenhar diagramas-bloco para F_1, F_2 e F_3 no Exercício 28 usando somente portas NE.

QUADRO 3.24

Entradas			Saída
X	*Y*	*Z*	*A*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

31. Escrever expressões de álgebra booleana para os Quadros 3.25 a 3.27, mostrando combinações em forma de soma-de-produtos; depois simplificar as expressões e desenhar um diagrama-bloco do circuito correspondente a cada expressão.

QUADRO 3.25

Entradas			Saída
A	*B*	*C*	*Z*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

QUADRO 3.26

Entradas			Saída
A	*B*	*C*	*Z*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

QUADRO 3.27

Entradas			Saída
X	*Y*	*Z*	*P*
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

32. Desenhar diagramas-bloco para os F_1, F_2 e F_3 no Exercício 28, usando apenas portas NOU.
33. Desenhar um diagrama-bloco para F_1, F_2 e F_3 no Exercício 28, usando circuitos OU/NE.
34. Desenhar diagramas-bloco para F_1, F_2 e F_3 no Exercício 28, usando circuitos porta E/NOU.
35. Desenhar mapas de Karnaugh para a expressão no Exercício 4.
36. Desenhar mapas de Karnaugh para a expressão no Exercício 2.
37. Para um mapa de quatro variáveis em W, X, Y e Z, desenhar os subcubos para:

    (a) $W\bar{X}\bar{Y}$ (b) $W\bar{X}$ (c) $\bar{X}Y\bar{Z}$ (d) $\bar{Y}$

38. Para um mapa de quatro variáveis em W, X, Y e Z, desenhar os subcubos para:

    (a) $W\bar{X}\bar{Y}\bar{Z}$ (b) $W\bar{Z}$ (c) $\bar{W}Z$ (d) Y

39. Desenhar mapas das expressões no Exercício 40 e depois desenhar os subcubos para os termos encurtados que forem achados.
40. Aplicar a regra $AY + A\bar{Y} = A$ onde for possível nas seguintes expressões:

    (a) $\bar{X}\bar{Y} + \bar{X}Y$ (b) $\bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C}$
    (c) $A\bar{B}C + ABC$ (d) $ABC + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C}$
    (e) $ABC + \bar{A}\bar{B}\bar{C} + A\bar{B}C$ (f) $ABC + \bar{A}BC + \bar{A}B\bar{C}$

    Nota: Há uma técnica para escrever os termos mínimos que é amplamente usada. Consiste em se escrever a letra m (para representar *termo mínimo* ou *minterm*) junto com o valor do número binário determinado pela seta da tabela de combinações em que está o termo mínimo. Por exemplo, na variável X, Y, Z, temos a seguinte tabela de combinações inacabada:

Entrada			Saída	Termos produto	Designação como m_i
X	Y	Z			
0	0	0		$\bar{X}\bar{Y}\bar{Z}$	m_0
0	0	1		$\bar{X}\bar{Y}Z$	m_1
0	1	0		$\bar{X}Y\bar{Z}$	m_2
0	1	1		$\bar{X}YZ$	m_3
1	0	0		$X\bar{Y}\bar{Z}$	m_4
1	0	1		$X\bar{Y}Z$	m_5
1	1	0		$XY\bar{Z}$	m_6
1	1	1		XYZ	m_7

Na tabela acima, temos: $m_0 = \bar{X}\bar{Y}\bar{Z}$, $m_1 = \bar{X}\bar{Y}Z$, $m_2 = \bar{X}Y\bar{Z}$, $m_3 = \bar{X}YZ$ e assim até $m_7 = XYZ$. Agora, podemos substituir m_i por termos reais e encurtar a escrita de expressões. Por exemplo, $m_1 + m_2 + m_4$ significa $\bar{X}\bar{Y}Z + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z}$. Similarmente, $m_0 + m_3 + m_5 + m_7$ significa $\bar{X}\bar{Y}\bar{Z} + \bar{X}YZ + X\bar{Y}Z + XYZ$.

Isto pode ser estendido a quatro ou mais variáveis. Uma expressão em W, X, Y, Z pode ser escrita como $m_0 + m_{13} + m_{15} = \bar{W}\bar{X}\bar{Y}\bar{Z} + WX\bar{Y}Z + WXYZ$. Ou $m_2 + m_5 + m_9 = \bar{W}\bar{X}Y\bar{Z} + \bar{W}XY\bar{Z} + W\bar{X}\bar{Y}Z$. Como pode ser visto, para mudar um termo mínimo para seu m_i, fazer simplesmente as variáveis não-complementadas 1 e as variáveis complementadas 0. Assim, $WXY\bar{Z}$ seria 0110, ou o decimal 6, $W\bar{X}Y\bar{Z}$ seria 0010, ou o decimal 2. Estes dois termos seriam então escritos m_6 e m_2. (Observar que devemos saber em quantas variáveis está um termo mínimo.)

41. Desenhar os mapas de Karnaugh em X, Y, Z para:

    (a) $m_0 + m_1 + m_5 + m_7$ (b) $m_1 + m_3 + m_5 + m_4$
    (c) $m_1 + m_2 + m_3 + m_5$ (d) $m_0 + m_5 + m_7$

42. Desenhar os subcubos para um mapa de três variáveis em X, Y, Z para:

(a) $m_1 + m_3 + m_5 + m_0$ (b) $m_5 + m_7$ (c) $m_0 + m_2$

43. Achar os subcubos máximos para os mapas desenhados no Exercício 42.
44. Achar expressões mínimas para os mapas desenhados no Exercício 42.
45. Usando mapas, simplificar as seguintes expressões nas quatro variáveis W, X, Y e Z.

(a) $m_2 + m_3 + m_5 + m_6 + m_7 + m_9 + m_{11} + m_{13}$
(b) $m_0 + m_2 + m_4 + m_8 + m_9 + m_{10} + m_{11} + m_{12} + m_{13}$

46. Usando mapas, simplificar as seguintes expressões nas quatro variáveis W, X, Y e Z.

(a) $m_1 + m_3 + m_5 + m_7 + m_{12} + m_{13} + m_8 + m_9$
(b) $m_0 + m_5 + m_7 + m_8 + m_{11} + m_{13} + m_{15}$

47. Usando mapas, deduzir expressões mínimas de produto-de-somas para as funções dadas no Exercício 46.
48. Usando mapas, deduzir expressões mínimas de produto-de-somas para as funções dadas no Exercício 42.
49. Usando mapas, simplificar as seguintes expressões, usando forma de soma-de-produtos.

(a) $\overline{A}\overline{B}C + A\overline{B}\overline{C} + \overbrace{ABC + \overline{A}BC + \overline{A}B\overline{C}}^{\text{saídas-não-especificadas}}$

(b) $ABC + \overline{A}\overline{B}\overline{C} + \overbrace{A\overline{B}C + A\overline{B}\overline{C}}^{\text{saídas-não-especificadas}}$

(c) $ABCD + \overline{A}\overline{B}\overline{C}D + \overline{A}BCD + \overbrace{A\overline{B}CD + \overline{A}\overline{B}CD + AB\overline{C}\overline{D}}^{\text{saídas-não-especificadas}}$

50. Usando mapas, deduzir expressões mínimas de produto-de-somas para as funções dadas no Exercício 46.
51. Usando mapas, simplificar as seguintes expressões, usando a forma de soma-de-produtos.

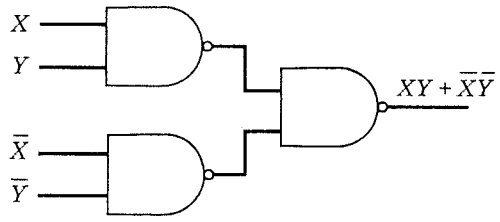
(a) $ABC + \overline{A}\overline{B}C + \overbrace{\overline{A}BC + A\overline{B}C + AB\overline{C}}^{\text{saídas-não-especificadas}}$

(b) $ABCD + \overline{A}\overline{B}CD + \overbrace{\overline{A}BCD + \overline{A}\overline{B}CD + \overline{A}\overline{B}\overline{C}\overline{D}}^{\text{saídas-não-especificadas}}$

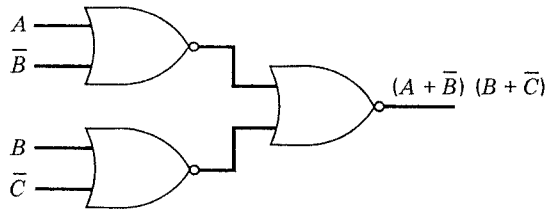
(c) $A\overline{B}C\overline{D} + A\overline{B}CD + \overbrace{\overline{A}\overline{B}CD + \overline{A}B\overline{C}\overline{D}}^{\text{saídas-não-especificadas}}$

Nota: As seguintes regras são úteis ao projeto que usa portas NE e portas NOU:

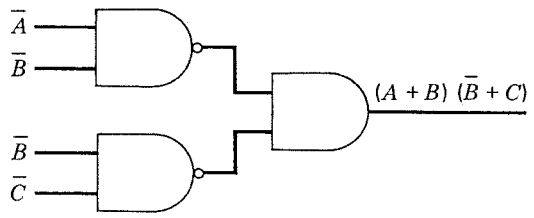
(1) Para projetar um circuito porta NE/NE, usar a forma de soma-de-produtos. Por exemplo, no projeto de $XY + \overline{X}\overline{Y}$:



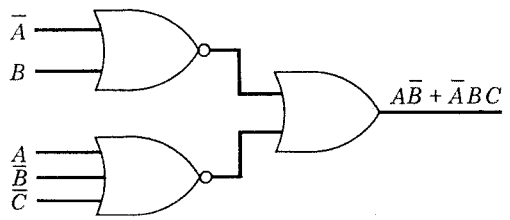
(2) Para projetar um circuito porta NOU/NOU, usar a forma de produto-de-somas. Por exemplo, no projeto de $(A + \bar{B})\ (B + \bar{C})$:



(3) Para projetar um circuito porta NE/E, usar a forma de produto-de-somas, mas complementar cada variável de entrada. Por exemplo, no projeto de $(A + B)\ (\bar{B} + C)$:



(4) Para projetar um circuito porta NOU/OU, usar a forma de soma-de-produtos, mas complementar cada variável de entrada. Por exemplo, projetar $A\bar{B} + \bar{A}BC$:



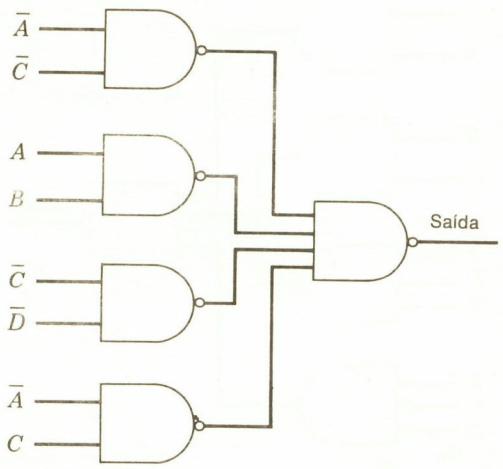
52. *(a)* Projetar um circuito porta combinacional E/OU para a seguinte expressão de álgebra booleana:

$$ABCD + AB\overline{C}\overline{D} + \overline{A}\overline{B}CD + AB\overline{C}D + ABC\overline{D} + \overline{A}BCD$$

Usar o menor número possível de portas.

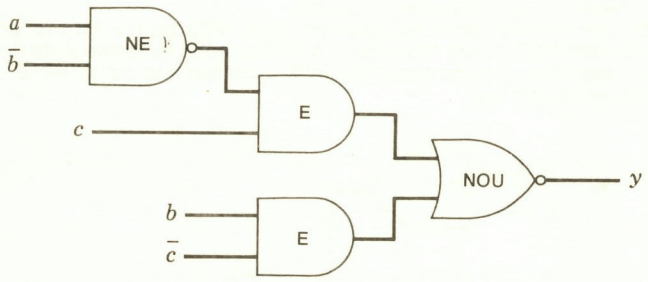
(b) Projetar um circuito porta combinacional NOU/NOU para a função de álgebra booleana do item *(a)*, usando novamente o menor número possível de portas.

53. O seguinte circuito é um circuito porta NE/NE. Desenhar um diagrama-bloco para um circuito porta NOU/NOU que realize a mesma função, usando o menor número possível de portas.



54. *(a)* Deduzir uma expressão de álgebra booleana para a saída y da rede mostrada abaixo.

(b) Converter a expressão de y deduzida em *(a)* à forma de produto-de-somas.



55. *(a)* Projetar um circuito porta combinacional OU/E para a expressão booleana

$$ABCD + \overline{A}B\overline{C}\overline{D} + \overline{A}B\overline{C}D + \overline{A}BCD + (\overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}CD)$$

Os dois termos entre parênteses são termos de saída não-especificada *(don't-care)*.

(b) Usando apenas portas NE, projetar um circuito combinacional para a função de álgebra booleana dada no item *(a)*.

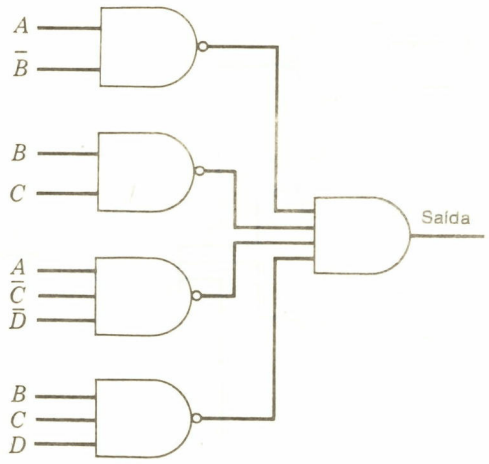
56. *(a)* Projetar um circuito porta combinacional do tipo OU/E para a expressão de álgebra booleana

$$ABCD + \overline{A}BC\overline{D} + \overline{A}B\overline{C}D + \overline{A}B\overline{C}\overline{D} + (\overline{A}\overline{B}\overline{C}\overline{D} + \overline{A}\overline{B}CD)$$

Os dois termos entre parênteses são termos de saída não-especificada *(don't-care)*.

(b) Usando apenas portas NOU, projetar um circuito combinacional para a função de álgebra booleana dada no item *(a)*.

57. O seguinte circuito porta NE/E deve ser reprojetado usando-se uma configuração de porta NOU/OU. Fazer a alteração usando o menor número possível de portas.



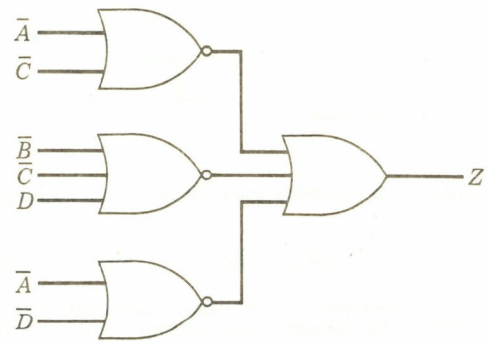
58. *(a)* Projetar um circuito porta combinacional do tipo E/OU para a expressão de álgebra booleana

$$ABCD + AB\overline{C}D + \overline{A}\overline{B}CD + AB\overline{C}\overline{D} + ABC\overline{D} + \overline{A}B\overline{C}\overline{D}$$

Usar o menor número possível de portas.

(b) Projetar um circuito combinacional de porta NOU para a função de álgebra booleana do item *(a)*, usando novamente o menor número possível de portas.

59. Converter o seguinte circuito porta NOU/OU num circuito porta NE/E. Usar o menor número possível de portas.



60. *(a)* Projetar um circuito porta combinacional do tipo E/OU para a expressão de álgebra booleana

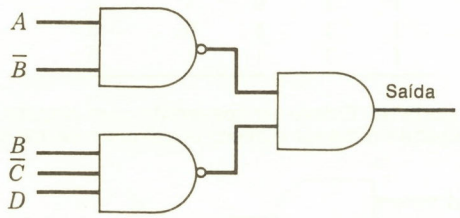
$$ABCD + AB\overline{C}D + \overline{A}\overline{B}CD + AB\overline{C}\overline{D} + AB\overline{C}\overline{D} + \overline{A}BCD$$

Usar o menor número possível de portas.

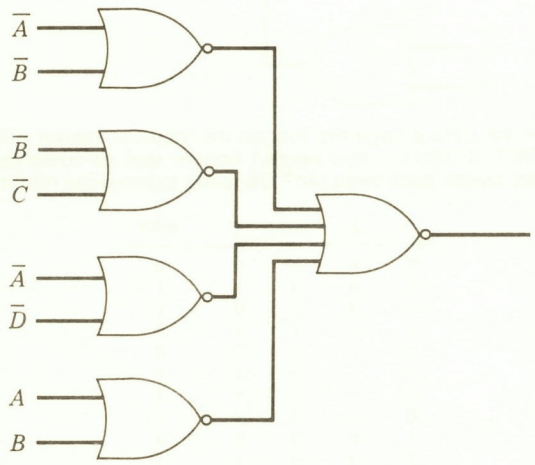
(b) Projetar um circuito porta combinacional do tipo NOU/NOU para a função de álgebra booleana do item *(a)*, usando novamente o menor número de portas possível.

61. Um circuito combinacional tem três entradas de controle, c_1, c_2 e c_3; três entradas de dados a_1, a_2 e a_3; e uma única saída z. ($\bar{a}_1$, $\bar{a}_2$, $\bar{a}_3$, $\bar{c}_1$, $\bar{c}_2$ e $\bar{c}_3$ são também disponíveis como entradas.) Cada entrada é um sinal avaliado em binário. Somente uma das entradas de controle pode ser um 1 em qualquer instante de tempo dado, e todas as três podem ser simultaneamente 0. Quando c_1 for um 1, o valor de z deve ser o valor de a_1; quando c_2 for um 1, o valor de z deve ser o valor de a_2; e quando c_3 for um 1, o valor da saída deve ser o valor de a_3. Se c_1, c_2 e c_3 forem 0, a saída z deverá ter valor 0. Projetar esta rede usando apenas portas NOU. Fazer com que a rede tenha dois níveis e usar o menor número possível de portas.

62. O seguinte circuito porta NE/E deve ser reprojetado usando uma configuração de porta NOU/OU. Fazer a alteração usando o menor número possível de portas.



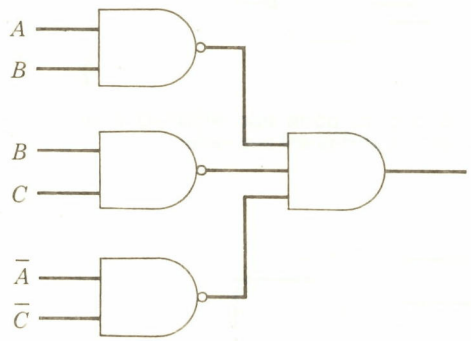
63. Converter o seguinte circuito porta tipo NOU/OU num circuito porta NE/E. Usar o menor número possível de portas em sua rede.



64. A expressão mínima da seguinte função irá requerer menor número de portas NE ou de portas NOU? (d significa *don't-care*.) Supor que os complementos sejam disponíveis. Quantas portas para cada um? Dar suas expressões mínimas.

X_1	X_2	X_3	X_4	Saída
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	d
1	0	0	0	d
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	d
1	1	0	1	0
1	1	1	0	d
1	1	1	1	0

65. O seguinte circuito porta NE/E deve ser convertido num circuito porta do tipo NOU/OU. Fazer a conversão usando o menor número possível de portas em seu projeto final.



66. A expressão mínima para a seguinte função irá requerer menor número de portas NE ou de portas NOU? (d significa *don't-care*.) Admitir que os complementos estejam disponíveis. Quantas portas para cada um? Dar suas expressões mínimas.

X_1	X_2	X_3	X_4	Saída
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	d
1	0	0	0	d
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	d
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

67. Simplificar

(a) $(\overline{W}+\overline{X}+Y+Z)(\overline{W}+X+\overline{Y}+Z)(\overline{W}+X+Y+Z)$

$$\overbrace{(W+X+\overline{Y}+Z)(W+\overline{X}+Y+Z)(\overline{W}+\overline{X}+Y+\overline{Z})(W+X+Y+Z)}^{\text{saídas-não-especificadas}}$$

(b) $\overline{a}\overline{b}\overline{c}d+\overline{a}b\overline{c}d+ab\overline{c}d+\overbrace{\overline{a}\overline{b}\overline{c}\overline{d}+\overline{a}bcd+abcd+ab\overline{c}\overline{d}+\overline{a}b\overline{c}\overline{d}}^{\text{saídas-não-especificadas}}$

(c) Para os itens *(a)* e *(b)*, projetar diagramas-bloco para o conjunto de circuitos lógicos das expressões simplificadas usando ou portas NE somente ou portas NOU somente. Admitir que os complementos das entradas sejam disponíveis. Tanto em *(a)* quanto em *(b)* não devem ser usados os mesmos tipos de portas.

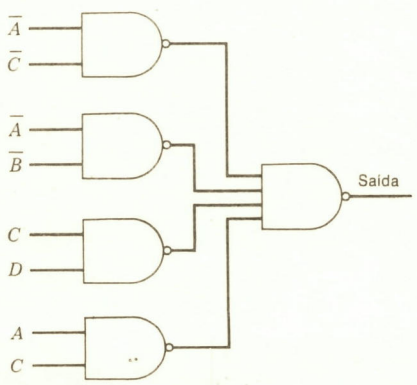
68. Escrever uma expressão de álgebra booleana sob a forma de soma-de-produtos para um circuito lógico com três entradas A, B e C (e seus complementos $\bar{A}$, $\bar{B}$ e $\bar{C}$) que deve ter uma saída 1 somente quando duas ou três das entradas tiverem um valor 1.

69. Desenhar um diagrama-bloco de um circuito porta, usando apenas portas NOU, com três entradas A, B e C (e seus complementos), que tenha uma saída 1 somente quando duas ou três das entradas tiverem um valor 1 como no Exercício 1.

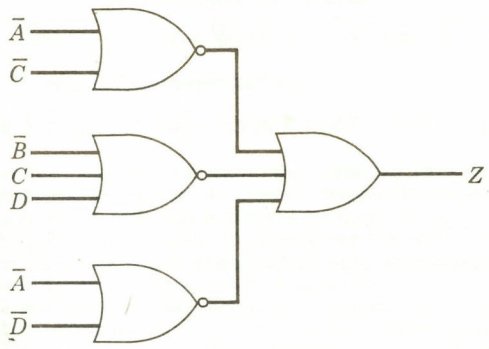
70. A expressão mínima da seguinte função irá requerer menor número de portas NE ou de portas NOU? (d significa *don't-care*.) Admitir que os complementos estejam disponíveis. Quantas portas para cada um? Dar suas expressões mínimas.

x_1	x_2	x_3	x_4	Saída
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	d
1	0	0	0	d
1	0	0	1	1
1	0	1	0	1
1	0	1	1	d
1	1	0	0	d
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

71. Temos abaixo um circuito porta do tipo NE/NE. Desenhar um diagrama-bloco de um circuito porta do tipo NOU/NOU que realize a mesma função, usando o menor número possível de portas.



72. Converter o seguinte circuito porta tipo NOU/OU num circuito porta do tipo NE/NE. Usar o menor número possível de portas em seu circuito.



73. *(a)* Projetar um circuito porta combinacional do tipo E/OU para a função de álgebra booleana

$$f = \overline{wxyz} + \overline{wxyz} + \overline{wxyz} + \overline{wxyz} + \overline{wxyz}$$

Usar o menor número possível de portas.

(b) Projetar um circuito porta combinacional tipo NOU para a função do item *(a)*, usando novamente o menor número de portas possível.

4

PROJETO LÓGICO

O capítulo precedente descreveu portas e a análise de circuitos ou redes lógicas que usam a álgebra booleana. Os dispositivos básicos usados nas seções de cálculo ou seções operacionais dos computadores digitais consistem de portas e dispositivos chamados *multivibradores biestáveis* (*flip-flops*). É extraordinário que mesmo os maiores computadores sejam primariamente construídos destes dispositivos. Por conseguinte, este capítulo descreve primeiro os multivibradores biestáveis e suas características. De um ponto de vista intuitivo, os multivibradores biestáveis proporcionam memória, e as portas proporcionam operações sobre (ou funções de) os valores armazenados nestes dispositivos de memória.

Em seqüência à introdução aos multivibradores biestáveis, é apresentado o emprego de multivibradores biestáveis e portas para realizar diversas das mais úteis funções nos computadores. As funções particulares descritas incluem: contagem em binário e em decimal-codificado-em-binário, transferência de valores e deslocamento ou escalamento de valores armazenados nos multivibradores biestáveis.

Há, casualmente, vários outros nomes que têm sido usados em vez de multivibrador biestável (*flip-flop*). Estes incluem *circuito de trava, binário* e *disparador biestável*, mas multivibrador biestável (*flip-flop*) tem sido o mais freqüentemente usado. Além disso, há diversos outros tipos de dispositivos de memória em computadores, e estes serão estudados no Cap. 7. Nas operações reais, no entanto, os multivibradores biestáveis permanecem predominantes devido às suas altas velocidades, à facilidade com que eles podem ser montados ou interpretados e à maneira natural pela qual as portas e os multivibradores biestáveis podem ser interligados.

Este capítulo também contém uma seção sobre relógios em computadores digitais. Os computadores não executam os programas, considerando etapas em tempos escolhidos ao acaso, mas prossegue de etapa em etapa a intervalos precisamente controlados por um relógio que proporciona uma base de tempo cuidadosamente regulada em todas as operações. Algum conhecimento dos empregos de relógios em computadores é indispensável, e o assunto é introduzido aqui.

Todo o material acima é apresentado sem recurso ao estudo de teoria de circuitos eletrônicos digitais. Em muitos casos, os estudantes já possuíam conhecimentos de circuitos digitais antes de estudarem esta matéria; em outros casos, onde os cursos e os usuários do livro estiverem com premência de tempo, decide-se prosseguir sem preocupação com detalhes de circuito relativos às portas e aos multivibradores biestáveis que forem usados. Como tal, este livro foi construído de modo que seja possível utilizá-lo omitindo-se a matéria sobre circuitos, que está no Cap. 5. (A critério do Professor, a matéria também pode ser selecionada do Cap. 5.)

É fato que os computadores vieram a ser construídos de circuitos que operam de maneira modular ou em blocos, e isto permite tratar o assunto de circuitos deste modo. Como os circuitos integrados têm-se tornado cada vez mais complexos, surgiu a necessidade de examiná-los de um ponto de vista funcional em vez de um ponto de vista de componentes, tanto no projeto de computadores quanto na manutenção. Perguntamos de início, "Que faz o circuito?", e depois, somente mais tarde, "Como ele trabalha?"

4.1 O MULTIVIBRADOR BIESTÁVEL

O circuito básico para armazenamento de informação em máquinas digitais é chamado *multivibrador biestável* (*flip-flop*). Há diversos tipos fundamentais de multivibradores biestáveis (também conhecidos por "circuitos basculadores") e muitos projetos de circuito; no entanto, há duas características compartilhadas por todos os multivibradores biestáveis.

1. O multivibrador biestável, como o próprio nome indica, é um dispositivo biestável, isto é, um circuito com apenas dois estados estáveis, os quais serão designados estado 0 e

estado 1.

O circuito multivibrador biestável pode lembrar, ou armazenar, um bit binário de informação por causa de sua característica biestável. O multivibrador biestável responde a entradas. Se uma entrada faz com que ele vá para seu estado 1, ele permanecerá ali e "lembrará" um 1 até que algum sinal faça com que ele vá para o estado 0. Similarmente, uma vez colocado no estado 0, ele ali permanecerá até que se diga que vá para o estado 1. Esta simples característica, a capacidade do multivibrador biestável reter seu estado, constitui a base do armazenamento de informação nas seções de operação ou de cálculo de um computador digital.

2. O multivibrador biestável tem dois sinais de saída, um dos quais é o complemento do outro.

A Fig. 4.1 mostra o diagrama-bloco de um tipo particular de multivibrador biestável, *o biestável RS*. Há duas entradas, designadas por S e R, e duas saídas, assinaladas X e $\bar{X}$. Para descrever e analisar a operação do biestável, há diversas convenções consideradas padrões na indústria de computadores.

1. A cada biestável é atribuído um "nome". Os nomes convenientes são geralmente letras tais como X ou Y ou A ou B, ou combinações de letras-números tais como A_1 ou B_2, ou, às vezes, devido à dificuldade em colocar índices com máquinas de escrever ou impressoras, simplesmente $A1$ ou $B2$. O biestável na Fig. 4.1 é chamado X. Ele possui duas saídas, a saída X e a saída $\bar{X}$.

   As linhas de saída X e $\bar{X}$ são sempre complementos; isto é, se a linha de saída $\bar{X}$ tem um sinal 1, a linha de saída X tem um sinal 0; e se a linha de saída X tem um sinal 0, a linha de saída $\bar{X}$ tem um sinal 1.
2. O estado do biestável é considerado o estado da saída X. Assim, se a linha X de saída tiver um sinal 1 sobre ela, diremos que o "biestável X está no estado 1". Similarmente, se a linha X contiver um sinal 0, diremos que o "biestável X está no estado 0".

Estes dois conjuntos de convenções são muito importantes e convenientes. Observemos que, quando o biestável X está no estado 1, a linha X de saída tem um 0 sobre ela e, quando o biestável X está no estado 0, a linha X de saída tem um 1 sobre ela.

Há duas linhas de entrada para o biestável RS. Estas são usadas para controlar o estado do biestável. As regras são:

1. Enquanto ambas as linhas de entrada S e R transportarem sinais 0, o biestável permanecerá no mesmo estado, isto é, não mudará de estado.
2. Um sinal 1 na linha S (a linha SET) e um sinal 0 na linha R fazem com que o biestável se "ajuste" (*set*) ao estado 1.
3. Um sinal 1 na linha R (a linha RESET) e um sinal 0 na linha S fazem com que o biestável se "reajuste" (*reset*) ao estado 0.
4. A colocação de um 1 sobre a linha S e de um 1 sobre a linha R ao mesmo tempo é proibida. Se isto ocorrer, o biestável pode ir para qualquer estado. (Esta é, com efeito, uma entrada ambígua em que se está dizendo ao biestável para se ajustar (SET) e se reajustar (RESET) simultaneamente.)

Um exemplo de uma possível seqüência de sinais de entrada e o estado resultante do biestável é como segue:

S	R	X	
1	0	1	X é o estado do biestável depois de as entradas S e R serem aplicadas.
0	0	1	O biestável permanece no mesmo estado.
0	0	1	
0	1	0	O biestável é reajustado.
0	0	0	
0	0	0	
0	1	0	Ao biestável é dito para se reajustar, mas já está reajustado.
0	0	0	
1	0	1	O biestável está ajustado.
0	0	1	

Enquanto que as convenções acima podem parecer formidáveis de início, elas podem ser simplesmente resumidas pela observação de que um 1 sobre a linha *S* faz com que o biestável se ajuste (SET) (isto é, assuma o estado 1) e um 1 na linha *R* faz com que o biestável se reajuste (RESET) (isto é, assuma o estado 0). O biestável nada faz na ausência de entradas 1 e estaria irremediavelmente perturbado pelos 1 em ambas as entradas *S* e *R*.

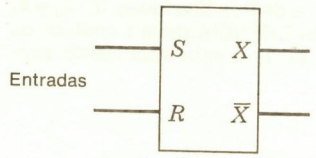


Fig. 4.1 Biestável tipo *RS*.

É muito conveniente que sejamos capazes de desenhar gráficos das entradas e saídas dos circuitos do computador para mostrar como eles agem quando as entradas variam. Admitiremos a convenção de que um sinal 1 é um sinal positivo e um sinal 0 é um sinal de 0 V, ou sinal de terra. Isto é convencional na maioria dos circuitos dos dias atuais e é o que se chama *lógica positiva*. A Fig. 4.2 mostra diversos sinais quando eles progridem em tempo, com os valores binários comuns de cada sinal escritos acima dele. Os sinais na Fig. 4.2

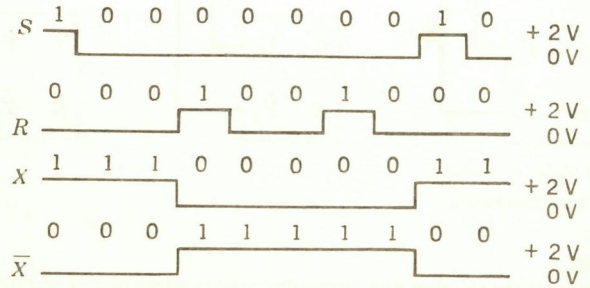


Fig. 4.2 Formas-de-onda do biestável *RS*.

representam a seqüência de sinais dada na lista acima junto com ambos os sinais *X* e $\overline{X}$ da linha de saída do biestável. Escolhemos arbitrariamente +2 V para o estado 1 dos sinais e 0 V para o estado 0; contudo, estes são níveis usados muito freqüentemente. Observemos que o biestável muda apenas quando os níveis de entrada o comandarem, e que ele muda imediatamente (na realidade deve haver um ligeiro retardo desde quando é dito ao biestável para mudar os estados até o instante em que ele muda, já que nenhum dispositivo físico pode responder instantaneamente; portanto, admitimos que o retardo do biestável na resposta é bem pequeno, talvez uma pequena fração de microssegundo).

4.2 A TRANSFERÊNCIA

O biestável *RS*, embora de operação simples, é adequado a todas as finalidades e é um circuito básico de multivibrador biestável. Examinemos a operação deste biestável numa configuração chamada *circuito de transferência*. A Fig. 4.3 mostra dois conjuntos de biestáveis denominados X_1, X_2, e X_3 e Y_1, Y_2 e Y_3. A função desta configuração é transferir os estados ou *conteúdos* de Y_1 para X_1, de Y_2 para X_2 e de Y_3 para X_3 ao comando de "TRANSFERÊNCIA" (*TRANSFER*), que consiste de um 1 na linha de TRANSFERÊNCIA.

Admitamos que Y_1, Y_2 e Y_3 tenham sido ajustados em alguns estados que queremos lembrar ou armazenar em X_1, X_2 e X_3, enquanto que os biestáveis *Y* são usados em cálculos posteriores. A colocação de um 1 na linha de TRANSFERÊNCIA produzirá esta desejada transferência de informação. A compreensão da transferência do estado de Y_1 para X_1 depende de ver que, se Y_1 está no estado 0, a linha de saída de Y_1 tem sobre ela um 0, e, portanto, a linha de entrada conectada à porta E será um 0 e a porta E colocará um 0 na linha de entrada *S* de X_1, enquanto que a saída $\overline{Y}_1$ de Y_1 será um 1, produzindo, na presença de um 1 na linha de TRANSFERÊNCIA, um 1 na entrada *R* de X_1. Um raciocínio idêntico

mostrará que um 1 em Y_1 produzirá um 1 a ser colocado em X_1 na presença de um 1 na linha de TRANSFERÊNCIA. Enquanto a linha de TRANSFERÊNCIA for um 0, ambas as entradas para os biestáveis X serão 0, e eles permanecerão no último estado que eles assumiram.

A operação simples acima, a chamada *operação de transferência*, é muito importante. Conjuntos relacionados de biestável num computador são chamados *registros*, e podemos chamar simplesmente *registro Y* aos três biestáveis Y_1, Y_2 e Y_3, e os três biestáveis X_1, X_2 e X_3 seriam chamados *registro X*. Então, um 1 na linha de TRANSFERÊNCIA deve transferir os conteúdos do registro Y para o registro X. Voltaremos mais tarde a este importante conceito.

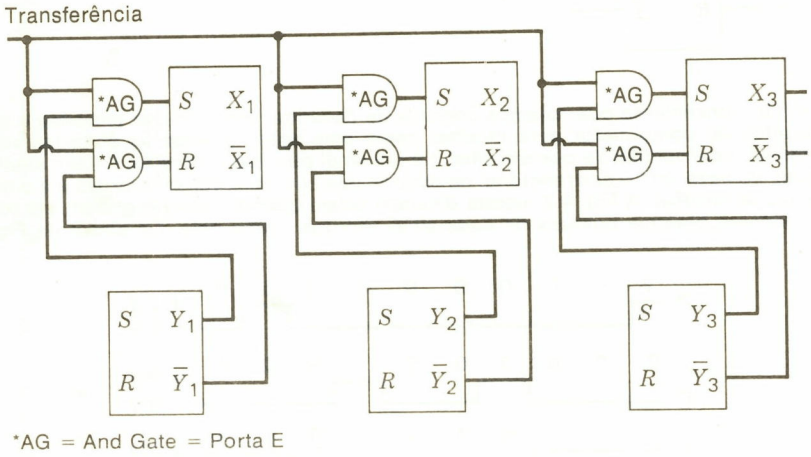


Fig. 4.3 Circuito de transferência.

4.3 O RELÓGIO

Um fato muito importante relativo aos computadores digitais (e aos multivibradores biestáveis), como os que concebemos agora, é que eles são quase invariavelmente sincronizados. Isto significa que há algum "relógio mestre" em algum lugar transmitindo sinais que são cuidadosamente regulados em tempo.

Há excelentes razões pelas quais as máquinas são projetadas desta maneira. A maneira alternativa, com as operações comandando outras operações como elas ocorrem, é chamada *operação assíncrona* (a maneira sincronizada é chamada *operação síncrona*) e conduz a uma considerável dificuldade no projeto e na manutenção. Em conseqüência, a operação genuinamente assíncrona é raramente usada.

O relógio é, portanto, o movedor ou motor do computador no sentido de que ele mede cuidadosamente o tempo e transmite sinais de relógio regularmente espaçados que fazem com que aconteçam certos fatos. Podemos então examinar a operação dos biestáveis e portas antes e após o relógio "iniciar uma ação." Os sinais de iniciação, por motivos históricos, são freqüentemente chamados *pulsos de relógio*.*

A Fig. 4.4 mostra um relógio. A forma-de-onda do relógio na Fig. 4.4(*a*) e (*b*) é chamada *onda quadrada*. A figura mostra duas porções importantes de uma onda quadrada, o *bordo dianteiro*, ou *bordo de subida* ou, às vezes, *bordo crescente positivo*, e o *bordo de queda*, ou *bordo crescente negativo*. Estas são particularmente importantes uma vez que a maioria dos biestáveis agora em uso responde ou a um bordo de queda ou a um bordo de subida (mas não a ambos). Com efeito, um sistema que responda a bordos de subida do relógio "descansa" entre tais bordos e muda de estado somente quando ocorrem tais bordos crescentes positivos. (A razão para os períodos de descanso é dar aos circuitos tempo para assumir seus novos estados e dar a todos os transitórios tempo para desaparecer. A freqüência em que ocorrem tais bordos é geralmente determinada pela velocidade com que os

*O termo *pulso de relógio* tem uma origem histórica. Os primitivos computadores usaram pulsos elétricos de curta duração para iniciar as operações, e estes foram naturalmente chamados *pulsos de relógio*. Alguns circuitos ainda usam estes pulsos estreitos, mas a maioria dos circuitos responde agora aos bordos de ondas quadradas como na Fig. 4.4.

circuitos podem passar para seus novos estados; o retardo sincroniza as portas, já que elas devem processar os novos sinais etc.)

Uma vez que os sinais de relógio são usados para iniciar as ações do biestável, uma entrada de relógio é incluída na maioria dos biestáveis. Esta entrada é assinalada com um CL, conforme mostrado na Fig. 4.5. Um biestável sincronizado pode responder tanto ao bordo crescente positivo do sinal de relógio quanto ao bordo crescente negativo.* Se um determinado biestável responde ao bordo crescente positivo do sinal, não há "bolha" na entrada CL no diagrama-bloco, como na Fig. 4.5(*a*). Se o biestável responde a um bordo crescente negativo, ou sinal, uma bolha é colocada na entrada CL, como na Fig. 4.5(*b*).

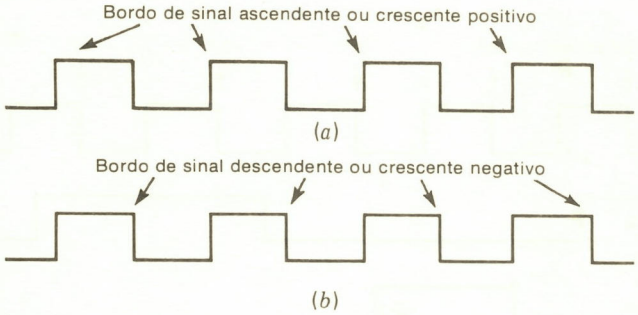


Fig. 4.4 Formas-de-onda de relógio.

É importante compreender a convenção acima porque os biestáveis sincronizados realmente respondem a uma *variação* no nível da entrada de relógio, e não ao nível em si.

Isto é mostrado na Fig. 4.5(*c*) e (*d*). O biestável na Fig. 4.5(*a*) responde aos bordos de relógios crescentes positivos (deslocamentos positivos), e um conjunto típico de sinais para o *biestável RS sincronizado* na Fig. 4.5(*a*) é mostrado na Fig. 4.5(*c*).

O biestável é operado de acordo com estas regras:

1. Se as entradas *S* e *R* são 0 quando o bordo (pulso) de relógio ocorre, o biestável não muda de estado, mas permanece em seu estado presente.
2. Se a entrada *S* é um 1 e a entrada *R* é um 0 quando o pulso de relógio (bordo crescente positivo) ocorre, o biestável passa para o estado 1.
3. Se a entrada *S* for um 0 e a entrada *R* for um 1 quando o pulso de relógio ocorre, o biestável está limpo para o estado 0.
4. Ambas as entradas *S* e *R* não devem ser 1 quando ocorrer um bordo crescente positivo do sinal de relógio.

Naturalmente, nada acontece ao estado do biestável nos intervalos das ocorrências da iniciação do sinal de relógio crescente positivo.**

A Fig. 4.5(*c*) mostra este fato com um sinal de relógio de onda quadrada. O biestável é ajustado em um 1 pelo primeiro bordo crescente positivo de relógio; um 0 na ocorrência do segundo sinal de relógio; nenhuma mudança ocorre no terceiro bordo crescente positivo de relógio; o biestável é ajustado para 1 novamente no quarto bordo; e permanece um 1 até que ocorra o sexto bordo de relógio. Observemos que as entradas *S* e *R* podem ser qualquer coisa entre os bordos de relógio sem afetar a operação do biestável. (Eles podem até mesmo ser 1 sem efeito, exceto quando ocorre o bordo crescente positivo.)

A Fig. 4.5(*d*) mostra as formas-de-onda típicas do biestável na Fig. 4.5(*b*). Este biestável está *disparando pelo bordo negativo* porque ele responde a deslocamentos no nível do

*Um biestável que responde a um sinal de relógio crescente ou decrescente (como em oposição à resposta a um nível de cc) é chamado biestável *mestre-escravo* ou *de comando pelo bordo* por motivos que serão explicados.

**Há um certo retardo a partir do instante em que o bordo crescente positivo do sinal de relógio diz ao biestável para "ir" até que as saídas dos biestáveis sejam capazes de mudar de valores. O próprio sinal de relógio também requererá algum pequeno intervalo de tempo para crescer, devido a razões físicas. Nos sistemas atuais, o tempo de crescimento nos sinais de relógio, isto é, o tempo necessário a um aumento de 90% do crescimento total, varia de cerca de 1×10^{-9} a cerca de 50×10^{-9} s. O retardo desde a mudança do sinal de relógio até que a saída do biestável mude a 90%, que é chamado *tempo de retardo*, varia desde $10{,}5 \times 10^{-9}$ a 50×10^{-9} s na maioria dos circuitos.

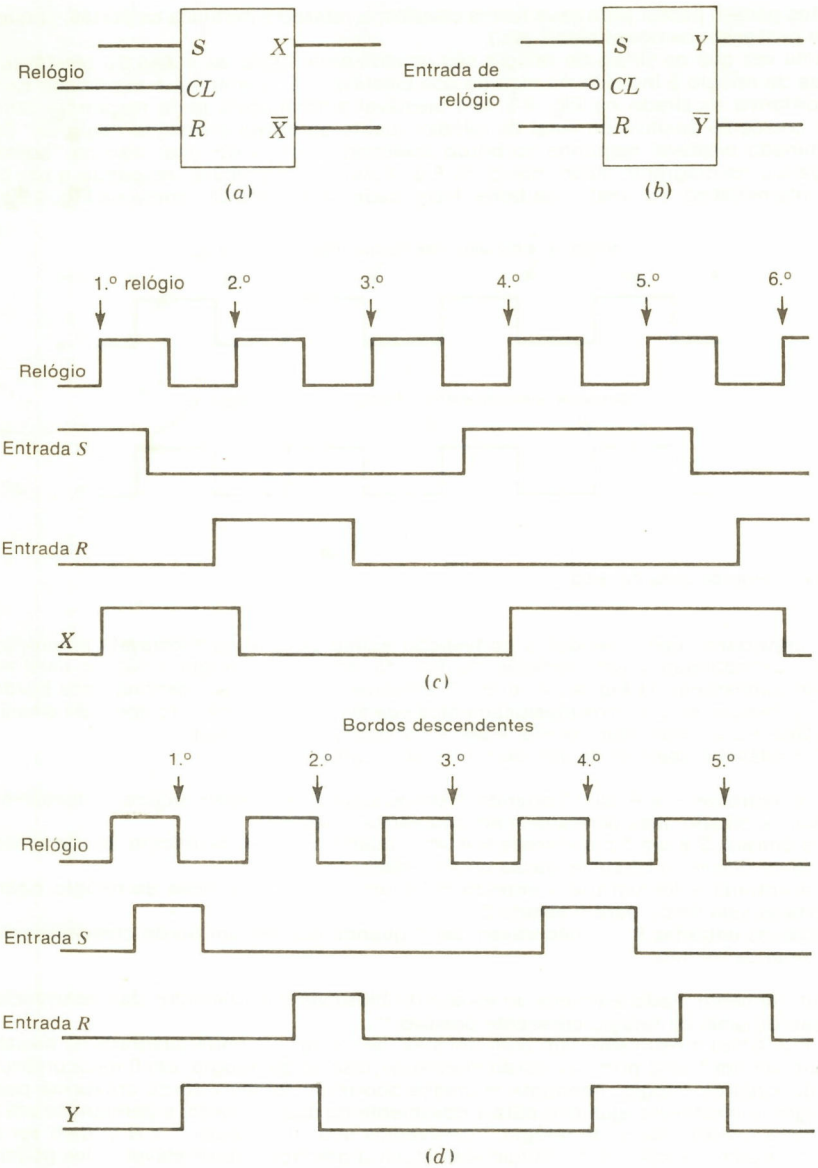


Fig. 4.5 Biestáveis sincronizados e formas-de-onda. (*a*) Biestável de comando pelo bordo positivo; (*b*) biestável de comando pelo bordo negativo; (*c*) formas-de-onda do biestável em (*a*) de comando pelo bordo positivo; (*d*) formas-de-onda do biestável em *(b)*.

relógio que são crescentes negativos. As regras de operação se mantêm como antes: 0 em *S* e *R* não conduz a nenhuma mudança; um 1 em *S* ajusta o biestável; e um 1 em *R* limpa (ou reajusta) o biestável. O biestável responde às entradas *S* e *R* somente no instante exato em que o relógio, ou a entrada CL, se torna negativo.

4.4 O REGISTRO DE DESLOCAMENTO

A Fig. 4.6 mostra um *registro de deslocamento*. Este circuito aceita informação proveniente de alguma fonte de entrada e depois desloca esta informação ao longo da cadeia de multivibradores biestáveis, movendo a informação um biestável cada vez que ocorre um sinal de relógio crescente positivo.

A Fig. 4.6 também mostra uma seqüência típica de sinais de entrada e sinais de biestável no registro de deslocamento. O valor da entrada é tirado por X_1 quando chega o primeiro sinal de relógio crescente positivo. Qualquer coisa neste biestável e nos restantes é deslocada corretamente neste instante de tempo. Admitimos que todos os biestáveis estejam inicialmente em seus estados 0. Na figura, a forma-de-onda de entrada está em 1 quando ocorre o primeiro relógio; portanto, X_1 vai para o estado 1.

Quando chega o segundo sinal de relógio crescente positivo, a entrada está em 0; portanto, X_1 vai para o estado 0, mas o 1 em X_1 é deslocado para X_2. Quando aparece o terceiro bordo de relógio, a entrada é um 1; portanto, X_1 adquire um 1, o 0 anteriormente em X_1 é deslocado para X_2 e o 1 em X_2 vai para X_3. Este processo continua. Os valores em X_3 são simplesmente abandonados no fim do registro.

Observemos que cada biestável tira o valor no biestável à sua esquerda quando o registro de deslocamento é escalonado. O raciocínio é como segue: Se, por exemplo, X_1 estiver no estado 1, sua linha de saída *X* será um 1 e, assim, a entrada *S* para X_2 será um 1; e a saída $\bar{X}$ de X_1 será um 0 e, assim, a entrada *R de* X_2 será um 0. Isto faz com que X_2 assuma seu estado 1 quando ocorrer o pulso de relógio. Um 0 em X_1 fará com que X_2 vá para 0 quando ocorrer o pulso de relógio, e a razão deste fato deve ser analisada.

Há um problema que poderá ocorrer se certas precauções de projeto não forem tomadas com os biestáveis. Se as saídas do biestável mudarem demasiado rápido, um estado

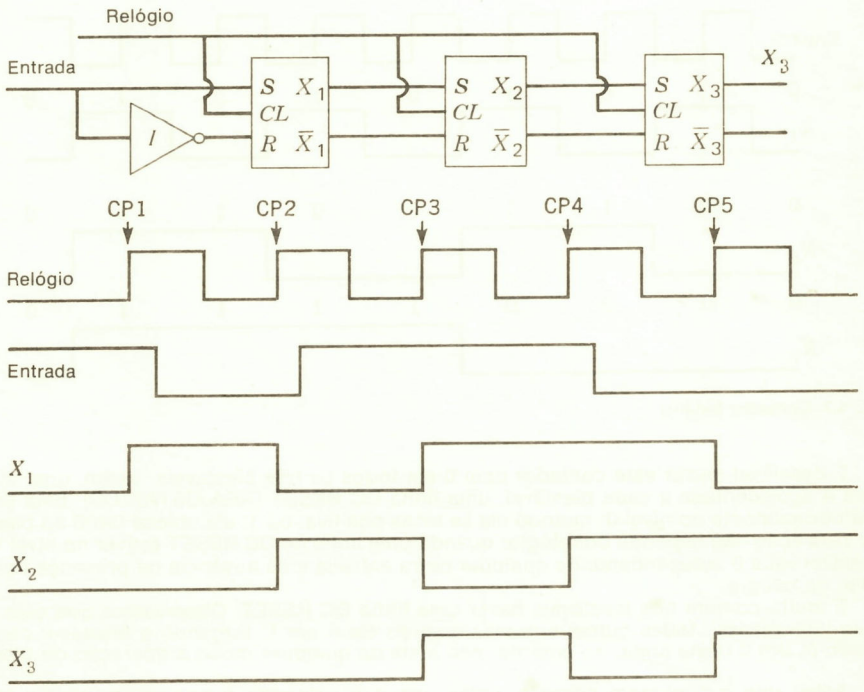


Fig. 4.6 Registro de deslocamento com formas-de-onda.

poderá agitar ou acelerar a cadeia. Este é o chamado *problema da corrida* e é manipulado projetando-se os biestáveis de modo que eles assumam o valor em suas entradas exatamente quando ocorrer o bordo crescente positivo do relógio e não ligeiramente após ou durante o tempo de subida do relógio. Isso conduz a uma certa complexidade no projeto do biestável, que não consideraremos agora. O projetista e o fabricante de circuitos nos proporcionam esta proteção, e a aceitaremos presentemente, mas uma explicação detalhada está nas seções finais deste capítulo e pode ser lida agora, se desejado.

4.5 O CONTADOR BINÁRIO

Visto que o contador binário é um dos mais úteis dentre os circuitos lógicos, há muitas espécies de contadores binários. O propósito fundamental do contador binário é registrar o número de ocorrências de alguma entrada. Esta é uma função básica, a de contagem, e é usada repetidamente.

O primeiro tipo de contador binário a ser explicado é mostrado na Fig. 4.7. Este contador registra o número de ocorrências de um bordo (ou pulso) crescente positivo na entrada.

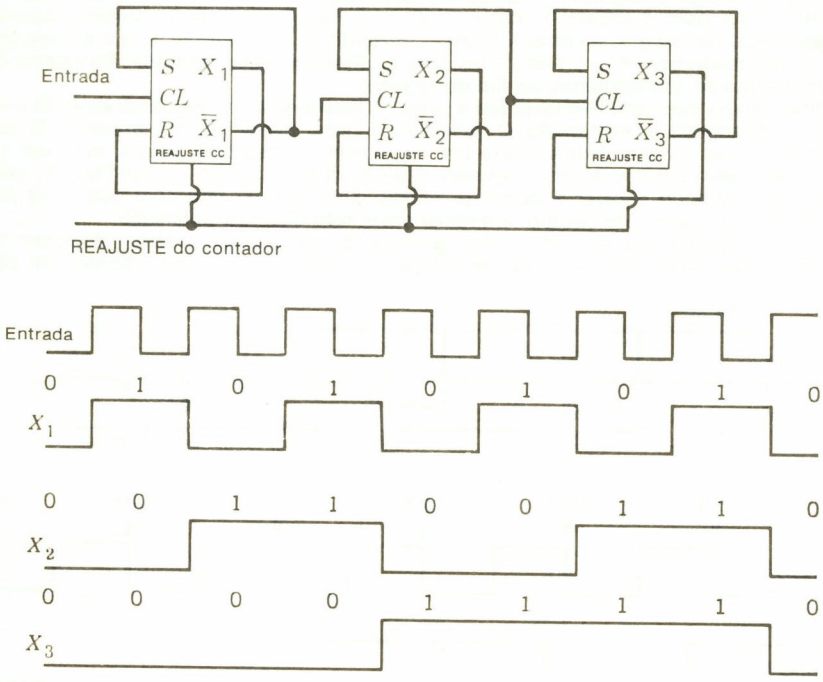


Fig. 4.7 Contador binário.

É desejável iniciar este contador com 0 em todos os três biestáveis; assim, uma outra linha é acrescentada a cada biestável, uma linha DC RESET (REAJUSTAR CC). Esta linha está normalmente no nível 0; quando ela se torna positiva, ou 1, ela coloca um 0 no biestável. Esta ação não depende do relógio; quando uma linha de DC RESET estiver no nível 1, o biestável vai a 0 independente de qualquer outra entrada e na ausência ou presença de um pulso de relógio.

É muito comum nos biestáveis haver uma linha DC RESET. Observemos que esta entrada "ultrapassa" todas outras entradas quando ela é um 1, forçando o biestável para o estado 0. Um 0 nesta linha, no entanto, não afeta de qualquer modo a operação do biestável.

Antes que a contagem comece, então, um 1 é colocado temporariamente na linha RESET COUNTER (REAJUSTAR CONTADOR), e os três biestáveis são limpos (reajustados)

para 0. A linha RESET COUNTER é então retornada a 0.

Quando ocorrer o primeiro bordo positivo de relógio, o biestável X_1 passa a seu estado 1. Isso é porque quando o biestável X_1 está no estado 0, a saída $\bar{X}_1$ é alta, ou 1, colocando um 1 na entrada S (ver Fig. 4.7), e a saída X_1 é baixa, ou 0, colocando um 0 na entrada R; assim, um 1 vai para o biestável X_1.

Os biestáveis X_2 e X_3 não são afetados por esta mudança, pois, embora a saída $\bar{X}_1$ esteja conectada à entrada CL de X_2, o sinal foi de 1 a 0. Este é um deslocamento negativo, que nada faz em X_2.

O contador agora tem $X_3 = 0$, $X_2 = 0$, $X_1 = 1$, ou o binário 001; assim, o primeiro bordo de relógio de entrada escalonou o contador de 000 a 001.*

A ocorrência do segundo bordo de relógio crescente positivo faz com que o biestável X_1 vá do estado 1 ao estado 0. A explicação é a seguinte: Quando X_1 for um 1, a saída X_1 será um 1 e estará conectada à entrada R, e a saída $\bar{X}_1$ será um 0 e estará conectada à entrada S. Isto diz ao biestável para "ir a 0", e, quando ocorre o segundo pulso de relógio, ele vai.

Isto é importante. Quando um biestável é de acoplamento cruzado, isto é, quando sua saída não-complementada está conectada à sua entrada R e sua saída complementada à sua entrada S, a ocorrência de um bordo de relógio fará sempre com que ele "complemente" ou mude de valores.

A mudança de valor de 1 para 0 do biestável X_1 faz com que X_2 mude de 0 para 1. Isso é porque a saída de $\bar{X}_1$ está conectada à entrada CL de X_2 e foi de 0 a 1, um deslocamento positivo, e, já que X_2 é de acoplamento cruzado, ele complementará (muda de valores) e irá de 0 a 1. Isto não afeta X_3, porque a entrada CL de X_3 foi de 1 a 0, um deslocamento negativo.

O contador agora prosseguiu para $X_1 = 0$, $X_2 = 1$ e $X_3 = 0$; assim, a seqüência de estados foi 000, 001, 010.

Um raciocínio deste tipo mostrará que a progressão de estados pelo contador será como segue:

X_3	X_2	X_1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1
0	0	0
0	0	1
0	1	0

Esta é uma lista de números binários de 0 a 7, que se repete vezes sem conta. Assim, depois de cinco pulsos de entrada, o contador contém 101, ou 5 em binário; depois de sete pulsos, o contador contém 111, ou 7 em binário. O número máximo de pulsos que este contador pode manipular, sem ambigüidade, é 7. Após oito pulsos, o contador contém 0; após nove pulsos, 1 etc. No comércio, isso é chamado *contador de módulo 8* ou *contador de três estágios*.

O contador pode ser ampliado por um outro biestável, X_4, que é de acoplamento cruzado e que tem sua entrada CL conectada à saída do biestável $\bar{X}_3$. Isto forma um *contador de módulo 15* ou *de quatro estágios*, que pode manipular até 15 contagens. Um quinto biestável formaria um contador que contaria até 31, um sexto biestável até 63 etc.

Consideremos agora um *contador binário ativado a níveis e sinais de relógio*. Este é um contador excessivamente popular nos modernos computadores com circuitos integrados, e

*Observar que os números binários estão escritos no sentido oposto ao do traçado do diagrama-bloco, que tem o bit menos significativo à esquerda. Isso faz um diagrama-bloco mais nítido e é freqüentemente usado. Os padrões, de fato, requerem um fluxo de sinal da esquerda para a direita.

ele demonstra o fato de que as operações em sua maioria são *permitidas* pelos níveis lógicos e ativadas pelos sinais de relógio. O contador precedente é chamado *contador de ondulação* porque muda a ondulação abaixo da cadeia de biestáveis.

Na Fig. 4.8, a entrada ENABLE (ATIVAR) para o primeiro biestável na cadeia X_1 vai para as duas portas E, que também têm a saída do biestável como entradas. Observemos, se o sinal ENABLE é um 0, que as duas portas E terão saída 0, e X_1 permanecerá no mesmo estado independente de quantos pulsos de relógio possam ocorrer.

Quando o sinal ENABLE for um 1, no entanto, as saídas do biestável X_1 farão com que o biestável sempre mude de valores quando ocorrer um pulso de relógio. Assim, o contador registra o número de pulsos de relógio que ocorrem enquanto o ENABLE estiver ligado. Então, o biestável X_2 mudará somente quando X_1 for um 1, o sinal ENABLE for um 1 e ocorrer um sinal de relógio crescente positivo. Similarmente, X_3 mudará de estados somente quando X_1 e X_2 forem 1, o ENABLE for um 1 e ocorrer um bordo positivo de relógio.

As duas portas E, combinadas com um biestável *RS* na Fig. 4.8, são tão úteis que a maioria das linhas populares de biestáveis contém, num único receptáculo de circuito integrado, o biestável e suas duas portas E, conforme mostrado na Fig. 4.9(*a*).

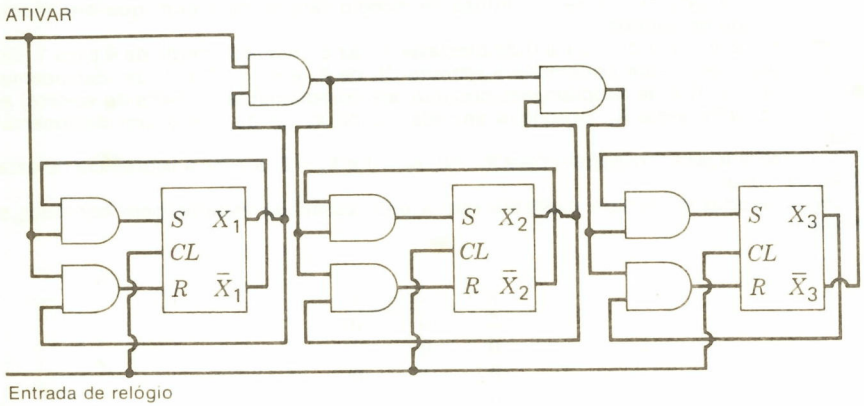


Fig. 4.8 Contador binário comandado a níveis lógicos e sinais de relógio.

A Fig. 4.9(*b*) mostra um outro biestável muito popular e útil, que consiste do biestável *RS* e de suas duas portas E, mas com as portas E tendo o acoplamento cruzado já permanentemente feito dentro do receptáculo. Nesta forma, as duas linhas tiradas para fora são chamadas *J* e *K*, e o biestável é denominado *biestável JK*. Este é mais útil, porque a análise indica que as entradas *J* e *K* atuam exatamente como entradas *RS* para duas entradas 0 — neste caso, o biestável nunca muda os estados. No entanto, com um 0 em *J* e um 1 em *K*, o biestável passa para o estado 0 quando aparecer um bordo positivo de relógio, e, com um 1 em *J* e um 0 em *K*, o biestável passa para 1 quando aparecer um bordo positivo de relógio. O fato significativo é que, quando ambas as linhas *J* e *K* forem 1, o biestável sempre mudará os estados quando aparecer um bordo positivo de relógio.

Os biestáveis na Fig. 4.9(*a*) e (*b*) têm ambos as entradas DC RESET e DC SET. As bolhas na entrada do diagrama-bloco indicam que estas são ativadas por entradas 0 e são normalmente mantidas num nível 1. Quando, por exemplo, um 0 é colocado na entrada DC SET, o biestável vai a um nível 1 independente do relógio ou de outras entradas. DC SET e DC RESET não devem ser 0 ao mesmo tempo, porque isto é proibido e conduz a um estado seguinte indeterminado.

Constitui uma regra geral o fato de que as bolhas, ou pequenos círculos, nas entradas DC SET e DC RESET significam que essas entradas são ativadas por níveis 0. A ausência destas bolhas significaria que as entradas são ativadas por níveis 1 e estão normalmente em 0.

Há um outro tipo de circuito, atualmente em uso geral, o chamado biestável *D*. Este biestável simplesmente toma o valor em sua entrada quando um pulso de relógio aparece e permanece em seu mesmo estado até que apareça o próximo pulso de relógio. Conforme mostrado na Fig. 4.9(*c*), o biestável *D* pode ser feito partindo de um biestável *RS* e de um

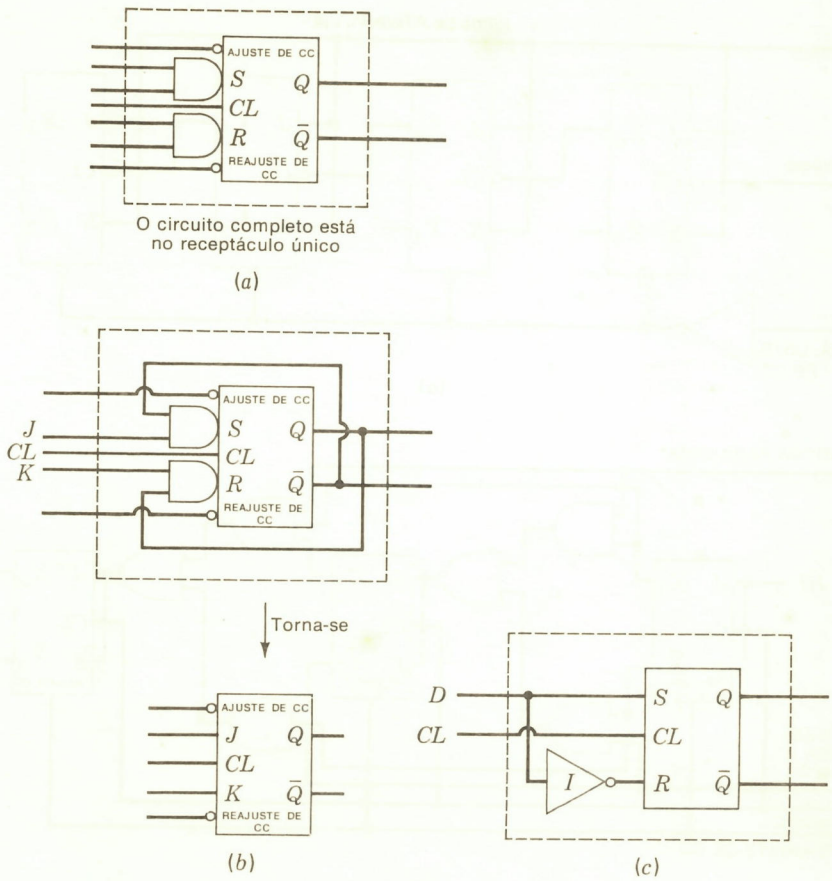


Fig. 4.9 Biestáveis tipos *JK* e *D*. (*a*) Biestável tipo *RS* com portas E; (*b*) como se faz um biestável *JK* a partir de biestáveis *RS*; (*c*) biestável tipo *D*.

inversor.

O biestável *D* é muito útil porque, quando sincronizado, ele assume o estado em sua entrada e o mantém até que seja sincronizado novamente. Somente uma única linha de entrada é necessária para uma transferência, enquanto que os biestáveis *RS* ou *JK* requerem duas linhas de entrada.

Um exemplo do uso dos biestáveis *JK* é mostrado na Fig. 4.10(*a*) e (*b*). A Fig. 4.10(*a*) mostra a simplicidade de um contador binário a portas e com biestáveis *JK*. A Fig. 4.10(*b*) mostra um diagrama-bloco de um *contador binário para cima e para baixo*. Quando a linha UP ENABLE (ATIVAR PARA CIMA) for alta ou um 1, o contador contará para cima, isto é, 0, 1, 2, 3, 4, ... Quando a linha DOWN ENABLE (ATIVAR PARA BAIXO) for um 1, o contador contará para baixo, isto é, 6, 5, 4, ... Em geral, o contador aumentará seu valor de 1 se a linha UP ENABLE for um 1 e chegar um pulso de relógio, ou diminuirá seu valor de 1 se a entrada DOWN ENABLE for um 1 e ocorrer um pulso de relógio.

Uma linha RESET é fornecida, a qual é usada, para DC RESET, o contador para 0. Isto é ativado por um 1 na linha RESET.

4.6 CONTADORES DE BCD (DECIMAL-CODIFICADO-EM-BINÁRIO)

Os contadores binários considerados até agora contam até seus limites antes do reajuste a todos os 0. Muitas vezes se deseja ter contadores que contêm em decimal-codificado-em-

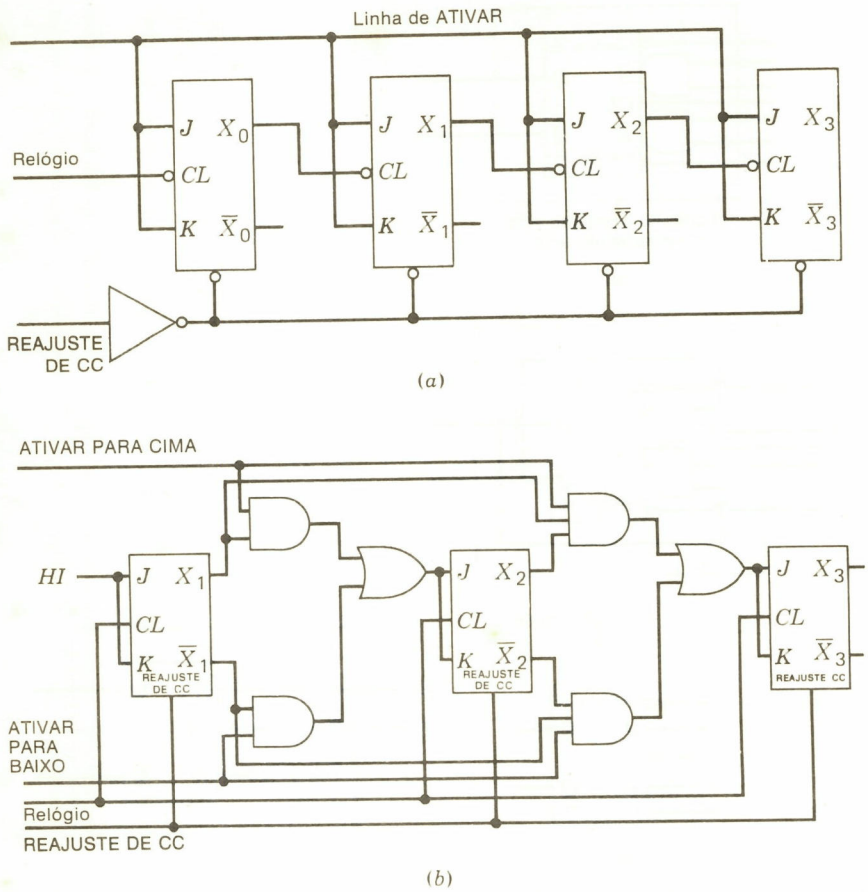


Fig. 4.10 Contadores binários com biestáveis *JK*. (*a*) Contador de ondulação comandado a portas; (*b*) contador para cima/para baixo.

binário (BCD). A Fig. 4.11(*a*) mostra um contador típico de BCD. O exame deste contador mostra que ele conta normalmente até que alcance 1001; isto quer dizer que a seqüência até aquele tempo é como segue:

X_4	X_3	X_2	X_1
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1

Quando o próximo bordo crescente negativo ocorre na entrada, no entanto, o contador de BCD retorna a todos os 0. Ao mesmo tempo (isto é, durante o intervalo em que o contador vai de 9 a 0), um bordo de sinal crescente negativo ocorre na saída CARRY. Esta saída CARRY (TRANSPORTE) pode estar conectada à ENTRADA de um outro contador de BCD, que será então escalonado de 1 quando o primeiro estágio BCD vai de 9 a 0. Isto é mostrado na Fig. 4.11(*b*), onde vários estágios de BCD com quatro biestáveis são combinados para fazer um grande contador.

Se considerarmos apenas duas das "caixas BCD", achamos que a seqüência é como segue:

8	4	2	1	8	4	2	1 valor de bits
Y_1	Y_1	Y_2	Y_1	X_1	X_1	X_2	X_1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	1
0	0	0	0	0	1	1	0
0	0	0	0	0	1	1	1
0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	1
.	.	.	.	.	.	.	.
0	0	0	1	1	0	0	1
0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	1
.	.	.	.	.	.	.	.

Aqui contamos até 21; isto continuaria até que o contador atingisse 99, quando a parte *Y* poria fora um sinal que poderia ser usado para comandar um outro estágio e formar um contador que pudesse contar até 999.

Esta espécie de repetição de várias "caixas" ou "módulos", tal como um contador de BCD, é facilitada pelos fabricantes colocando-se um contador completo de BCD de quatro estágios num único receptáculo de circuito integrado, como é muitas vezes o caso.

Um fato deve ser observado a respeito do diagrama-bloco na Fig. 4.11(*b*). Os biestáveis disparam (isto é, são ativados por) deslocamentos crescentes negativos nos níveis de entrada na entrada. Isto é indicado pelos pequenos círculos de "bolhas" nas entradas. Em conseqüência, um biestável como X_3 é ativado quando X_2 vai de 1 a 0, isto é, quando a saída 1 faz uma transição negativa.

Além disso, note que as entradas não-conectadas, tais como as entradas *K* de todos os biestáveis e as entradas *J* de X_1 e X_3, são supostas estar nos níveis 1. Isto é devido à construção do circuito.

4.7 CIRCUITOS INTEGRADOS

Os multivibradores biestáveis e portas usados nas modernas máquinas de computação — as quais abrangem desde as calculadoras e microcomputadores até os grandes computadores de alta velocidade — são construídos e compactados utilizando-se a chamada *tecnologia de circuitos integrados*. Quando são usados os circuitos integrados, um ou mais biestáveis ou portas completos são compactados num só receptáculo de circuito integrado (CI). Estes receptáculos de CI possuem pinos de entrada e de saída ou conexões que são depois interligados por tiras chapeadas (metalizadas) sobre painéis de circuito, fios ou meios diversos para formar dispositivos completos de computação.

Nos primitivos computadores, os circuitos porta e os biestáveis foram construídos empregando-se componentes elétricos discretos tais como resistores, capacitores, transistores e, antes desses, válvulas a vácuo e relés. Os componentes individuais serviam para

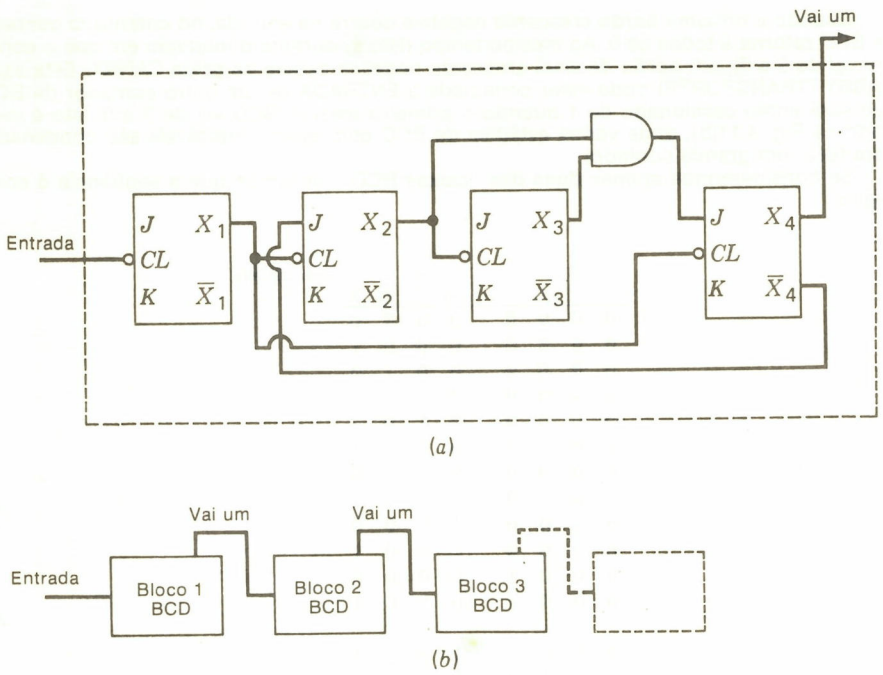


Fig. 4.11 Contadores de BCD. (*a*) Contador de décadas, ou de BCD; (*b*) blocos de contadores de BCD em cascata.

formar biestáveis e portas que depois de interligados constituíam os computadores. Com a tecnologia de CI dos dias de hoje, os biestáveis e as portas são fabricados em receptáculos e somente os receptáculos de CI (ou "caixas") necessitam ser interligados.

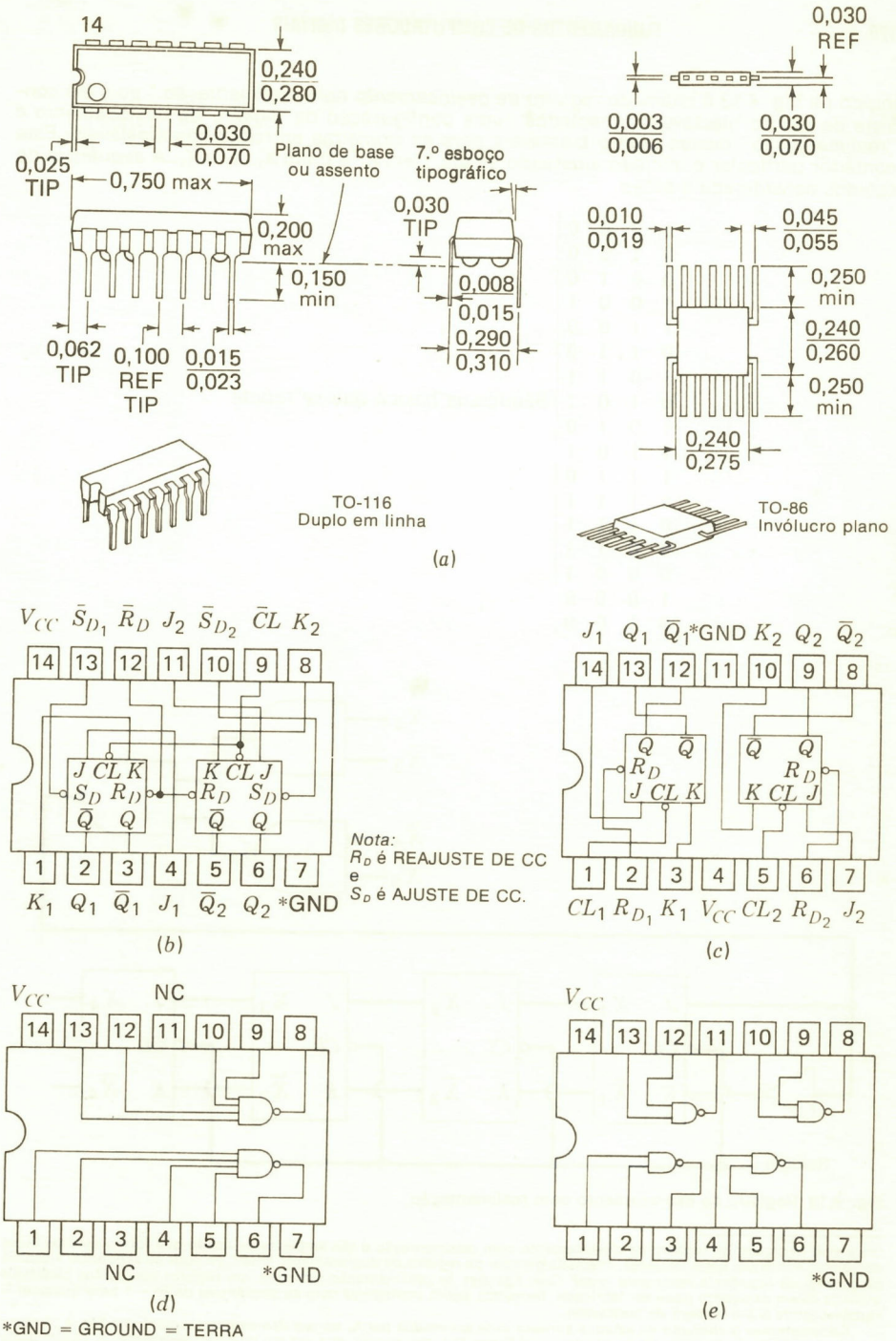
Dois receptáculos típicos de CI são mostrados na Fig. 4.12(*a*). Um é chamado *embalagem em linha dupla* (no comércio, ele é denominado *coffin* ou DIP), e esta embalagem particular tem 14 pinos que proporcionam as ligações externas. Por muito tempo, esta embalagem de 14 pinos constituiu um padrão na indústria, e DIP plásticos e cerâmicos desta espécie foram as embalagens de CI das mais vendidas durante alguns anos.

No entanto, quando se procurou aperfeiçoar a tecnologia dos CI, houve uma tendência para aumentar o número de pinos por embalagem. Assim, as embalagens com 16 e 20 pinos estão tornando-se populares, e até 70 pinos por embalagem podem atualmente ser encontradados em alguns produtos de fabricantes de CI.

A Fig. 4.12(*b*) até (*e*) mostra como vários biestáveis e portas são compactados num único receptáculo. As entradas e saídas são numeradas, e cada número se refere a um pino externo sobre o receptáculo de CI. Uma ligação de terra e uma tensão positiva de alimentação para cada receptáculo são requeridas de modo que somente 12 pinos fiquem para ser usados como entradas e saídas reais das portas e biestáveis. (Para estes circuitos, cada pino V_{CC} é conectado a uma fonte de alimentação de 5,5 V, e o pino GND, a OV ou terra do sistema.)

Os circuitos particulares na Fig. 4.12 são chamados circuitos lógicos transistor-transistor (TTL) e constituem circuitos de alta velocidade amplamente usados (ver Cap. 5). Estas configurações especiais com ligações de pino idênticas são fabricadas por quase todos os grandes fabricantes de CI, e as embalagens de um fabricante podem ser facilmente substituídas pelas embalagens de um outro fabricante (contanto que os requisitos de velocidade ou as capacidades de carga não sejam violados). Há muitas outras embalagens com, por exemplo, 3 portas NE de três entradas, 2 biestáveis do tipo *RS*, portas OU exclusivo etc.

A fim de ilustrar o uso das embalagens de CI no projeto lógico, examinaremos agora uma implementação da Fig. 4.13, usando as embalagens mostradas na Fig. 4.12. O circuito



*GND = GROUND = TERRA

Fig. 4.12 Receptáculos de CI e circuitos porta e de biestáveis. (*a*) Receptáculos de CI com invólucro plano e em linha dupla; (*b*) biestáveis *JK* duplos com relógio e reajustes comuns e ajustes separados; (*c*) biestáveis *JK* duplos com reajustes e relógios separados; (*d*) portas NE duplas de quatro entradas; (*e*) portas NE quádruplas de duas entradas.

lógico na Fig. 4.13 é chamado *registro de deslocamento com realimentação*,* pois ele consiste de quatro biestáveis conectados numa configuração de registro de deslocamento e "realimentação" destes quatro biestáveis para as primeiras entradas dos biestáveis. Este contador particular é iniciado ajustando-se um 1 em X_1 e 0 em X_2, X_3 e X_4. A seqüência de estados considerada é então

1	0	0	0	
0	1	0	0	
0	0	1	0	
1	0	0	1	
1	1	0	0	
0	1	1	0	
1	0	1	1	
0	1	0	1	Seqüência básica que se repete
1	0	1	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	
0	1	1	1	
0	0	1	1	
0	0	0	1	
1	0	0	0	
0	1	0	0	

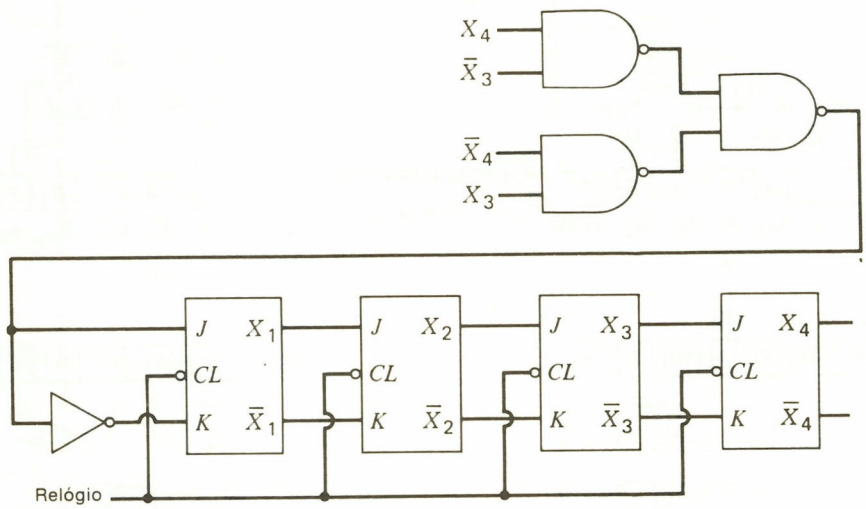


Fig. 4.13 Registro de deslocamento com realimentação.

*Este tipo particular de registro de deslocamento com realimentação é tão amplamente usado que foram escritos livros completos sobre ele. Ele é, às vezes, chamado *gerador de registro de deslocamento linear, gerador de seqüência aleatória ou gerador de seqüência recorrente linear*. Com ligações de realimentação similares, um registro com tantos biestáveis quantos forem desejados pode ser fabricado, formando assim contadores com as seqüências de $2^N - 1$ para qualquer N razoável (onde N é o número de biestáveis).

Consideremos o conjunto de estados consecutivos assumidos por X_4 no registro de deslocamento na Fig. 4.13 como sendo sua seqüência de saída. Cada não-zero quádruplo (4-tuple) ocorre uma vez em qualquer segmento de 15 bits desta seqüência, cada não-zero triplo (3-tuple) ocorre duas vezes etc. Acrescentando-se um segmento de 15 bits desta seqüência a um outro segmento de 15 bits, bit-por-bit mod 2 (OU exclusivo), obter-se-á ainda um outro segmento de 15 bits. Estas seqüências são usadas nos instrumentos para formar geradores de números aleatórios e para gerar ruído de faixa de passagem; em radar, para observações interplanetárias; nos sistemas de comunicações, para gerar ruído e codificar ou cifrar; e para muitas outras finalidades. Ver Birkhoff e Bartee para outras informações e referências.

Observemos que esta seqüência contém 15 dos 16 possíveis números de 4 bits que podem ser tirados por este circuito. (Somente a combinação de todos os 0 é excluída.) Esta é uma seqüência amplamente usada que ocorre em muitos instrumentos e tem muitos empregos em sistemas de radar, sistemas de sonar, caixas para cifragem de códigos etc.

Muito freqüentemente, a seqüência de estados assumida por um circuito lógico é escrita numa *tabela de contador*. A tabela de contador para a seqüência considerada acima é

X_1	X_2	X_3	X_4
1	0	0	0
0	1	0	0
0	0	1	0
1	0	0	1
1	1	0	0
0	1	1	0
1	0	1	1
0	1	0	1
1	0	1	0
1	1	0	1
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1

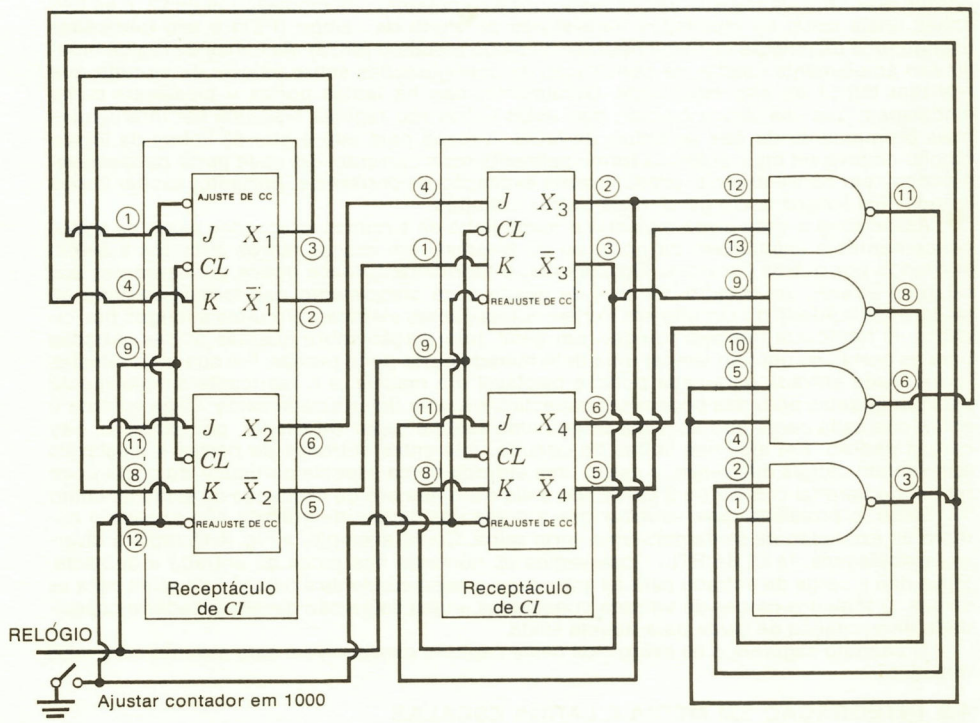


Fig. 4.14 Projeto de registro de deslocamento na Fig. 4.13 usando TTL. Os números nos círculos são números de pinos dos receptáculos de CI.

Na tabela de contador, os nomes dos biestáveis são listados primeiro, seguidos pelo estado inicial. Então, os estados sucessivos considerados são listados em ordem, e a linha final contém o estado que precede o estado inicial.

Há uma técnica direta de projetar um circuito lógico para realizar uma tabela de contador; esta técnica é desenvolvida na Seção 4.12. Por ora, retornemos à implementação do contador na Fig. 4.13.

A fim de implementar este contador, necessitamos de quatro biestáveis e de um circuito porta que produzirão a função $\bar{X}_3X_4 + X_3\bar{X}_4$. Conforme mostrado, isso pode ser feito com um circuito porta NE/NE com 3 portas NE de duas entradas. Também é necessário um inversor. (Uma porta NE pode ser usada para isto, conectando-se ambas as entradas.)

Resta ainda um problema: Necessitamos iniciar o contador com X_1 no estado 1 e os outros três biestáveis no estado 0. Já que as entradas DC RESET são conectadas juntas nos biestáveis [ver Fig. 4.12(*b*)], é necessário usar um artifício para o biestável X_1. Isto simplesmente implica dar outro nome às entradas J e K e às duas saídas, de modo que J se torna K, K se torna J, e os nomes das duas saídas ficam invertidos. A entrada DC RESET então se torna uma entrada DC SET no novo biestável (que recebeu outro nome).

A Fig. 4.14 mostra o circuito como finalmente projetado. Observemos como X_1 difere de X_2 e X_3 quanto às ligações.

O circuito lógico na Fig. 4.14 pode ser incrementado usando um painel de circuito impresso para fazer as ligações entre os receptáculos de CI, ou as ligações podem ser feitas por fios individuais usando-se qualquer um dos diversos painéis de interconexão fabricados pelas várias companhias. A colocação de um 0 (terra) na entrada DC RESET ajusta os biestáveis nas condições iniciais desejadas, e o circuito então escalonará através dos estados desejados.

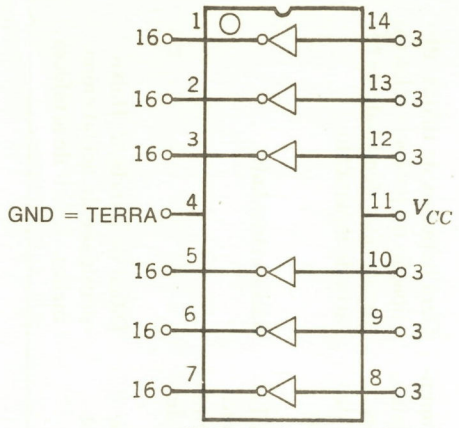
Há nove linhas principais de circuitos integrados atualmente sendo produzidas em quantidades substanciais. O Quadro 4.1 lista estas linhas e apresenta algumas das características de cada linha. As seis primeiras linhas de CI na tabela são chamadas *lógica bipolar* porque elas utilizam transistores convencionais (bipolares) nas embalagens de CI, e as três linhas finais usam os chamados transistores de efeito de campo (FET) e são fabricadas usando uma tecnologia de semicondutor de óxido metálico (MOS). As linhas de lógica bipolar são amplamente usadas na construção de configurações sobre painéis de circuito que realizam lógica de alta velocidade. Geralmente, não há tantas portas e biestáveis numa embalagem que usa lógica bipolar, mas estas linhas são rápidas e podem ser interligadas mais prontamente do que as linhas de MOS. A razão para isto é que as linhas da lógica bipolar empregam mais potência (principalmente mais corrente) em cada porta ou biestável e podem, em conseqüência, produzir mais excitação de corrente e, portanto, excitar cabos longos, fios longos e, em geral, mais outros circuitos.

Associados a cada porta e biestável numa linha de circuitos integrados estão os dados concernentes à capacidade da porta ou do biestável em excitar outros circuitos e serem excitados por outros circuitos. Tipicamente, o fabricante fornece dados concernentes aos retardos através do circuito, tempos de queda e de crescimento nas formas-de-onda de saída, a capacidade do circuito em excitar outras cargas elétricas, circuitos e longos fios ou cabos. O fabricante também fornece, em geral, informação sobre quantas outras entradas para as portas de um tipo similar uma determinada porta pode excitar. Em sua mais simples forma, cada entrada para cada porta e biestável é a mesma, e o fabricante simplesmente estipula quantas entradas podem ser conectadas a uma determinada saída. Cada entrada é então chamada *carga padrão* e diz-se que uma saída é capaz de excitar, por exemplo, oito cargas padrão. Em algumas linhas de circuito, diferentes entradas de portas e biestáveis apresentam cargas diferentes, e assim uma entrada numa espécie particular de porta pode ter um número tal como 2 ou 3 associado a ela e um número de excitação de saída tal como 12. Então, o projetista deve verificar que a soma das cargas de entrada não exceda o número de excitação de saída para uma certa saída. Como exemplo, a Fig. 4.15 mostra diversas embalagens de CI de RTL. Observemos os números nos pinos de entrada e de saída. Estes dão a carga de entrada para as entradas e a capacidade de excitação de saída para as saídas. O total das cargas de entrada conectadas a uma saída não devem exceder a capacidade de excitação de saída para aquela saída.

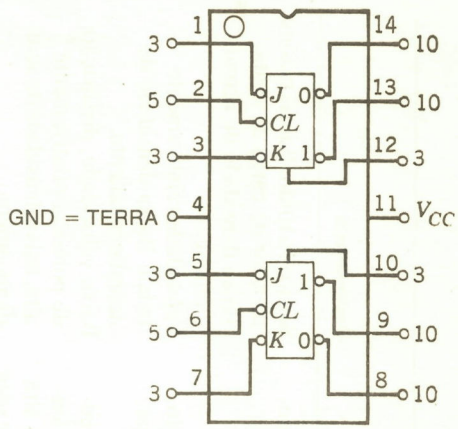
O capítulo seguinte e os exercícios neste capítulo desenvolvem este assunto com mais detalhes.

4.8 INTEGRAÇÃO EM MÉDIA E LARGA ESCALAS

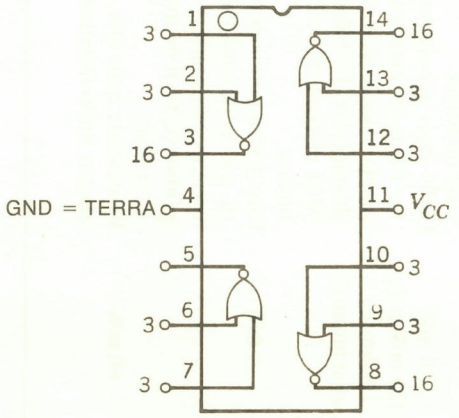
Conforme foi explicado, a maioria dos circuitos atualmente fabricados emprega a tecnologia geral de conjunto de circuitos integrados. Neste caso, os transistores, diodos, resistores e quaisquer outros componentes são fabricados em conjunto, usando as técnicas da Física



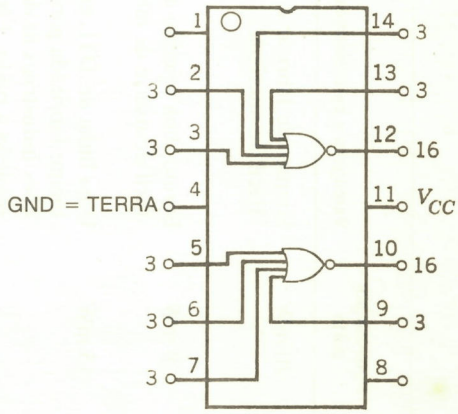
MC789P– Inversor sêxtuplo



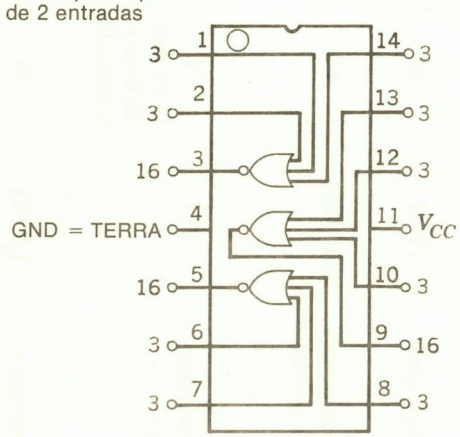
MC790P– Biestável *JK* duplo



MC724P– Porta quádrupla de 2 entradas



MC725P– Porta dupla de 4 entradas



MC792P– Porta tripla de 3 entradas

Fig. 4.15 Embalagens de CI de RTL.

QUADRO 4.1 LINHAS DE CIRCUITOS INTEGRADOS

Nome da linha de circuito	Acrônimo	Velocidade (retardo por porta), ns	Potência por porta	Vantagens e características	Desvantagens
Lógica resistor-transistor	RTL	40	20 mW	Simples de fabricar e barato. Fácil de interligar.	Sensível a ruído. Baixas razões de fator de carga de saída. Baixa densidade de acondicionamento.
Lógica diodo-transistor	DTL	30	8 mW	Barato, boa rejeição a ruído, fácil de usar e de interligar.	Velocidade relativamente baixa. Baixa densidade de acondicionamento.
Lógica diodo-transistor de alto nível	HLDTL	40	15 mW	Uma linha de DTL especialmente fabricada para aplicações industriais devido à alta rejeição a ruído. Fácil de usar e de interligar.	Baixa velocidade, dissipação de potência relativamente alta, baixa densidade de acondicionamento.
Lógica transistor-transistor	TTL	7 (3 ns para Schottky restaurado)	10 mW (20 mW para Schottky restaurado)	Linha mais popular no momento. Fácil de interligar; rápida, ampla seleção de circuitos disponíveis. Embalagens de MSI disponíveis, a custos baixos.	Gera impulsos de ruídos, dissipação de potência relativamente alta, densidade de acondicionamento.
Lógica transistor-transistor de baixa potência	LPTTL	30	1 mW	Uma baixa potência por TTL de portas desenvolvida para aplicações espaciais e outras portáteis, fácil de usar e de interligar.	Baixa velocidade.
Lógica acoplada por emissor	ECL	0,5	60 mW	Velocidade mais alta, gera pouco ruído internamente.	Difícil de interligar. Baixa densidade de acondicionamento. Difícil de arrefecer.

QUADRO 4.1 (CONT.) LINHAS DE CIRCUITOS INTEGRADOS

Nome da linha de circuito	Acrônimo	Velocidade (retardo por porta), ns	Potência por porta	Vantagens e características	Desvantagens
Silício metal-óxido do canal *p*	PMOS	50	0,1 mW	Baixa potência, boa densidade de acondicionamento, fácil de fabricar, barato.	Lento e frágil, e tem capacidade limitada para excitar linhas e servir de interface com outras linhas de circuitos.
Silício metal-óxido do canal *n*	NMOS	20	0,1 mW	Mais rápido do que o PMOS, potência relativamente baixa, boa densidade de acondicionamento, barato e relativamente fácil de fabricar.	Tem capacidade limitada para excitar linhas para servir de interface com outros circuitos.
Silício metal-óxido complementar	CMOS	10	10 nW	Potência sobressalente requerida muito baixa. Velocidade e densidade de acondicionamento moderadas, preços razoáveis.	Difícil de fabricar. Consumo de potência aumenta quando for comutado em altas velocidades.

Nota: Na tabela acima, W é para watts; n é para nano = 10^{-9}; m é para mili = 10^{-3}; e s é para segundos.

do estado sólido, num único receptáculo. Na tecnologia mais comum, chamada *conjunto de circuitos integrados monolíticos*, uma pastilha de semicondutor singelo é processada por fotomascaramento, gravação, difusões e outras etapas, produzindo assim um conjunto completo de diodos, transistores e resistores já interligados para formar uma ou mais portas lógicas de biestáveis.

Dois termos acham amplo uso na discussão de conjunto de circuitos integrados (CI). Quando mais de um ou dois biestáveis ou algumas portas são compactados num único receptáculo, o processo é chamado *integração em média escala* (MSI). Observemos que a integração em média escala ainda se refere a circuitos integrados, exceto que mesmo mais circuitos sejam abrigados num único receptáculo. Não há regras específicas fixadas, mas, geralmente, se mais de 10, porém menos de 100, portas ou biestáveis estiverem numa única embalagem, o fabricante se referirá a ela como MSI.

Quando mais de 100 portas ou biestáveis são fabricados num único e pequeno receptáculo, o processo é chamado *integração em larga escala* (LSI). Algumas idéias sobre a complexidade dos conjuntos desta espécie serão achadas em capítulos posteriores onde memórias e unidades aritméticas e lógicas que consistem de milhares de biestáveis e portas numa única embalagem serão estudadas.

A despeito dos vários níveis de integração, os circuitos são surpreendentemente similares em princípio, exceto que a LSI tende a usar uma tecnologia baseada em MOS, enquanto a MSI e os circuitos integrados "convencionais" usam transistores *npn* e, às vezes, *pnp* "convencionais" fabricados em pastilhas de silício. Há boas razões para este fato. Os circuitos de MOS requerem áreas muito pequenas numa pastilha e usam muito pouca potência, o que é muito importante, considerando o fator volume/complexidade. Por outro lado, como tinha sido mencionado, os circuitos bipolares convencionais são mais rápidos e mais prontamente interligados. Em conseqüência, a tecnologia MOS é mais freqüentemente usada nos conjuntos maiores que podem ser tratados somente como unidades singelas completas em vez de numa base de circuito-por-circuito.

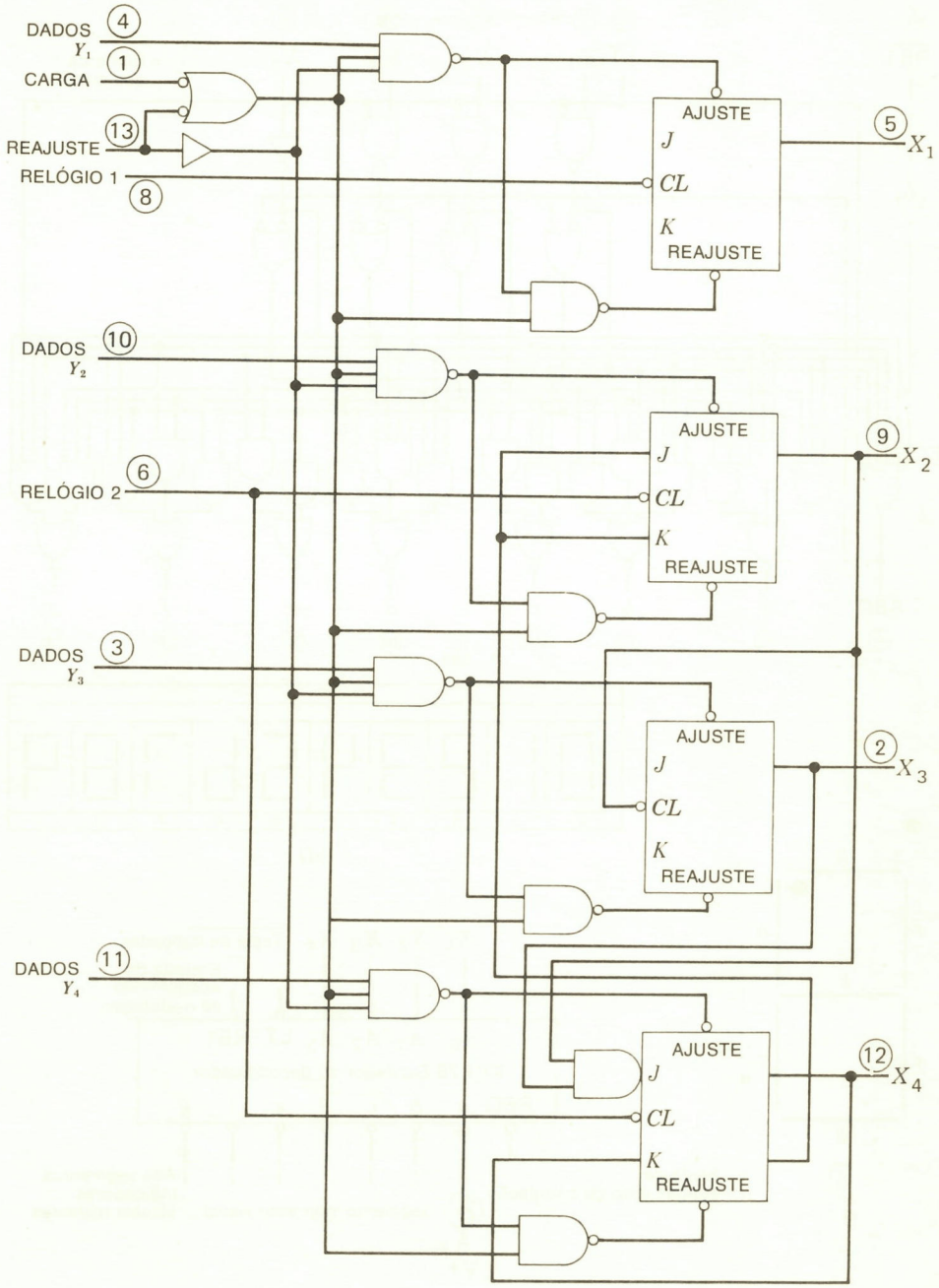
A Fig. 4.16(*a*) mostra uma embalagem típica de MSI contendo um contador de BCD completo. Este contador escalona de 0 a 9 e depois reajusta para 0 quando X_1 (que é o pino 5) está conectado ao relógio 2 (que é o pino 6). O contador é escalonado cada vez que uma forma-de-onda de relógio de entrada conectada ao relógio 1 (pino 8) se torna negativa (nos bordos negativos). O contador pode ser reajustado a todos os 0 conectando-se um 0 na linha de reajuste (pino 13). Os dados provenientes dos quatro fios de entrada conectados a Y_1, Y_2, Y_3 e Y_4 serão carregados nos biestáveis X_1, X_2, X_3 e X_4, respectivamente, se a entrada de LOAD (CARGA) for puxada para baixo até um 0. (Normalmente ela é um 1.)

Um exemplo de circuitos porta em embalagens MSI é mostrado na Fig. 4.16(*b*), que ilustra um *decodificador de sete segmentos*. Quando tiverem de ser lidos números decimais provenientes de uma calculadora digital, de um instrumento, de um microcomputador etc., são freqüentemente usados dispositivos de indicação visual que empregam diodos de emissão de luz (LED) ou cristais líquidos. Cada dígito do indicador visual é formado de sete segmentos, cada um consistindo de um diodo emissor de luz ou de um cristal que pode ser ligado ou desligado. Um arranjo típico é mostrado na Fig. 4.16(*c*) que designa as letras de *a* até *g* para os segmentos. Para formar o dígito 5, por exemplo, são ligados os segmentos *a*, *f*, *g*, *c* e *d*. O conjunto de dígitos assim formado por estes segmentos é mostrado na Fig. 4.16(*d*).

O decodificador de sete segmentos na Fig. 4.16(*b*) pode ser conectado às saídas dos quatro biestáveis no contador de BCD na Fig. 4.16(*a*), conectando-se as saídas X_1, X_2, X_3 e X_4 da Fig. 4.16(*a*) às entradas X_1, X_2, X_3 e X_4 da Fig. 4.16(*b*). Se as sete saídas *a* até *f* da Fig. 4.16(*b*) forem então conectadas a um dispositivo de indicação visual digital decimal, será formado um contador com um indicador de dígitos decimais, tal como os existentes na calculadora familiar.

O contador de BCD na Fig. 4.16(*a*) pode ser ampliado de diversos dígitos, conectando-se a saída X_4 de um dígito à entrada 1 de relógio do dígito seguinte de ordem mais alta no contador.

O decodificador de sete segmentos na Fig. 4.16(*b*) tem a capacidade de apagar os zeros dianteiros num indicador visual de múltiplos dígitos, o que é comumente feito nas calculadoras. Consideremos que um contador de BCD de múltiplos estágios foi conectado a diversos decodificadores de sete segmentos com um decodificador por estágio contador de BCD. Se a saída de apagamento de ondulação (RBO) de cada decodificador de sete segmentos for conectada à entrada de apagamento de ondulação (RBI) do decodificador de sete segmentos do próximo dígito de ordem mais alta no contador e a entrada de apagamento de ondulação do decodificador de sete segmentos do dígito mais significativo for conectada a uma entrada 0, será formado um circuito de apagamento. Então, por exemplo,



(*a*)

Nota: As entradas desligadas são 1.

Fig. 4.16 Ver legenda na página seguinte.

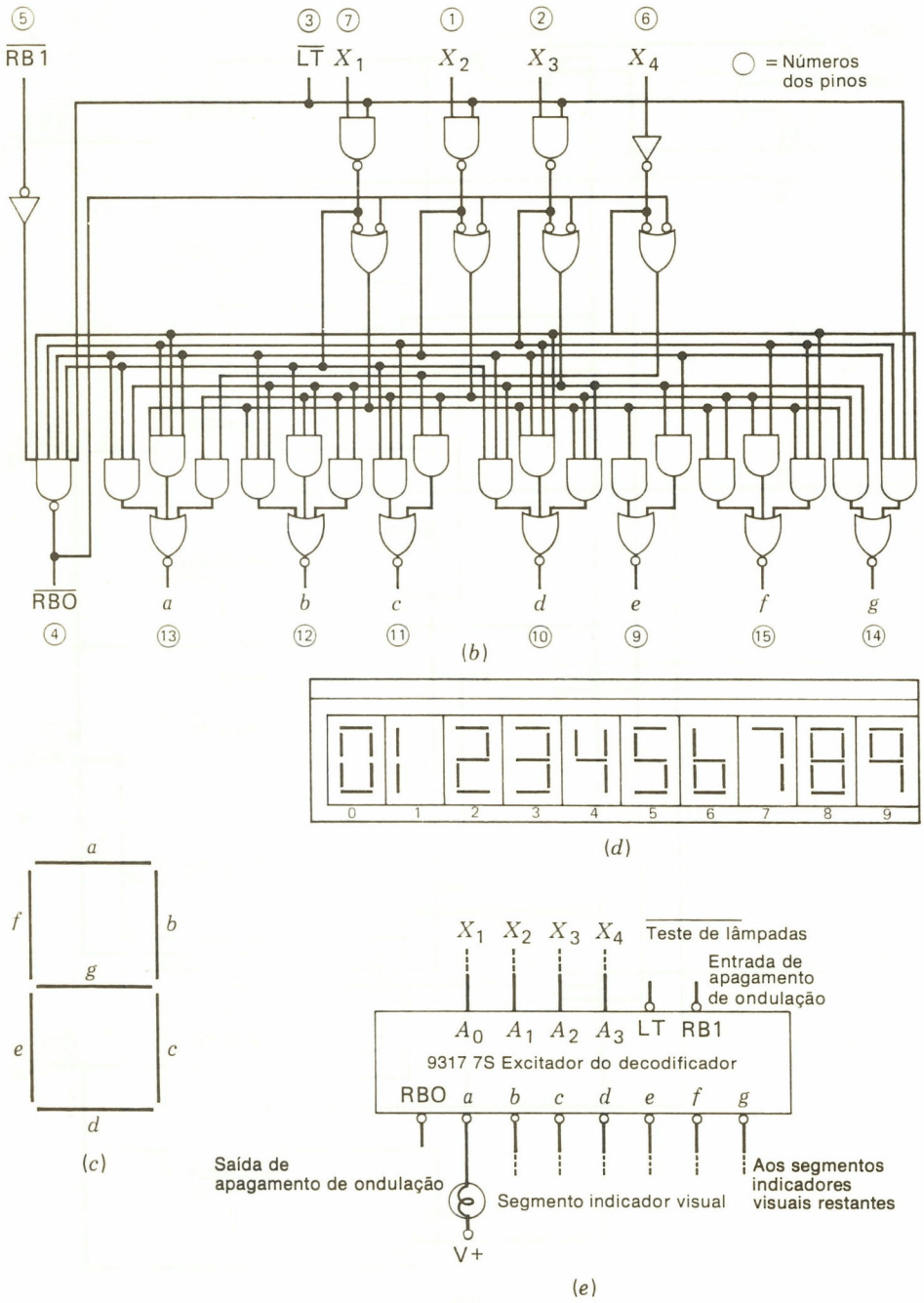


Fig. 4.16 (*a*) Contador de BCD em embalagem de CI; (*b*) diagrama lógico de decodificador de sete segmentos; (*c*) designação dos sete segmentos; (*d*) números formados pelos sete segmentos; (*e*) disposição geral de conexão do decodificador de sete segmentos. (Cortesia de Fairchild Semiconductor.)

num contador de quatro estágios, o número 0014 terá os dois 0 dianteiros desligados ou "apagados"; o número 0005 será exibido simplesmente como 5, com os indicadores de 0 não ligados etc. Com efeito, o circuito testa um valor 0 em sua entrada e, se o valor for 0 e todos os dígitos à sua esquerda forem 0, então ele desliga todos os sete segmentos e gera um sinal de apagamento para o excitador de luz dos dígitos seguintes mais à direita. A entrada de teste de luz (LT) pode ser usada para testar todos os sete segmentos simultaneamente. Observemos que fazendo-se a entrada do teste de luz um 0, faremos com que todos os sete segmentos fiquem ligados.

Nos produtos de MSI de alguns fabricantes, um contador de BCD de quatro estágios e um decodificador de sete segmentos são ambos colocados na mesma embalagem dotada de uma saída e de uma entrada de apagamento para cada dígito. Esta descrição dá alguma noção do que sejam as embalagens de MSI mais complexas.

4.9 PROJETOS DE BIESTÁVEIS COM CIRCUITOS LÓGICOS

É importante observar que os biestáveis podem ser feitos de circuitos porta e, de fato, que isto constitui uma prática comum. A Fig. 4.17 mostra duas portas NOU com acoplamento cruzado para formar um biestável tipo *RS*. As portas NOU de acoplamento cruzado na Fig. 4.17(*a*) têm duas entradas, *S* e *R*, e duas saídas, *Q* e $\bar{Q}$. Esta configuração realiza o biestável tipo *RS* na Fig. 4.17(*b*).

A operação das portas NOU é como segue: Consideremos que tanto *S* quanto *R* sejam 0. Se *Q* for um 1, então a porta NOU mais à direita tem um 1 e uma entrada 0, e sua saída será um 0. Isto coloca um 0 na saída $\bar{Q}$ e dois 0 no ponto para a porta NOU mais à esquerda que terá uma saída 1, e a configuração será estável. Um raciocínio idêntico mostrará que a configuração será estável com um 1 em $\bar{Q}$ e um 0 em *Q*.

As entradas *S* e *R* trabalham como segue: Se um **1** for colocado na entrada *R* e um 0 na entrada *S*, isto forçará a porta NOU mais à esquerda para uma saída 0, e isto fará com que a porta NOU mais à direita tenha dois 0 como entradas e uma saída 1. O biestável foi agora limpo com um 0 na saída *Q* e um 1 na saída $\bar{Q}$. Raciocínio idêntico mostrará que um 1 na entrada *S* e um 0 na entrada *R* forçarão a porta NOU do biestável para o estado 1 com *Q* um 1 e $\bar{Q}$ um 0.

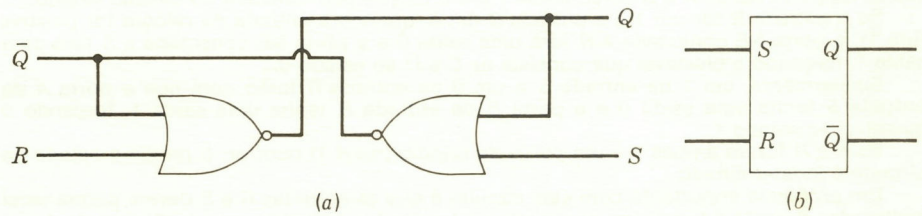


Fig. 4.17 Biestável *RS* formado por portas NOU com acoplamento cruzado. (*a*) Portas NOU de acoplamento cruzado; (*b*) biestável *RS* correspondente a (*a*).

4.10 O BIESTÁVEL COM CIRCUITOS LÓGICOS

Tal como a Fig. 4.17 mostrou que duas portas NOU podem ser usadas para formar um biestável *RS*, a Fig. 4.18 mostra que duas portas NE podem ser usadas para formar um biestável *RS*. Neste caso, as entradas operam como segue. Quando *S* e *R* forem ambas 1, o biestável permanecerá em seu estado presente, isto é, não mudará de estados. Se, no entanto, a entrada *R* for a 0, a porta NE conectada a *R* terá uma saída 1 independente da outra entrada realimentada para a porta NE, e isto forçará o biestável ao estado 0 (contanto que a entrada *S* seja mantida alta ou um 1).

Um raciocínio idêntico mostra que, fazendo a entrada *S* um 0, faremos com que a porta NE na entrada *S* tenha uma saída 0, forçando o biestável ao estado 1 (novamente, contanto que a entrada *R* seja mantida alta ou 1).

Se ambas as entradas *R* e *S* forem feitas 0, o estado seguinte dependerá de qual entrada é primeiro retornada a 1 e, se ambas forem retornadas a 1 simultaneamente, o estado resultante do biestável será indeterminado. Em conseqüência, esta é uma combinação de entrada "proibida" ou "restrita".

O diagrama-bloco na Fig. 4.18 mostra o biestável que seria um biestável *RS* convencional, caso as duas entradas não estivessem invertidas. Isto é mostrado pelas duas "bolhas"

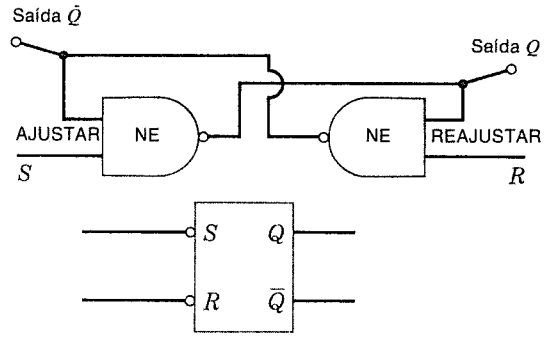


Fig. 4.18 Duas portas NE usadas para formar o biestável *RS*.

nas entradas *R* e *S*. O circuito é, portanto, ativado pelos 0, e as entradas estão normalmente em 1.

Uma forma limitada de biestável sincronizado pode ser formada usando-se quatro portas NE, conforme mostrado na Fig. 4.19. O circuito tem uma entrada *R* e uma entrada *S* e também uma entrada *CL* de relógio. Este biestável é ativado por um nível positivo na entrada de relógio, e não numa transição positiva. Assim, o biestável "assume" seus níveis de entrada durante a porção positiva dos sinais de relógio, não muda em níveis de relógio. Para verificar como o circuito trabalha, se o sinal de relógio estiver no nível 0, ambas as portas NE *A* e *B* terão saídas 1, e, portanto, as entradas da porta NE para *C* e *D* serão 1 e, como antes, o biestável permanecerá em seu estado presente com ou *C* ou *D* ligado. (Ambos não podem estar ligados por causa do acoplamento cruzado.)

Se o sinal de relógio for ao nível 1 e ambas as entradas *R* e *S* estiverem no nível 0, as saídas da porta NE de *A* e *B* serão ainda 1 e o biestável permanecerá no mesmo estado.

Se a entrada *R* for um 1 e a entrada *S* um 0, quando a entrada de relógio for positiva (um 1), a porta NE conectada a *R* terá uma saída 0 e a porta NE conectada a *S* terá uma saída 1, forçando o biestável que consiste de *C* e *D* ao estado 0.

Similarmente, um 1 na entrada *S* e um 0 na entrada *R* farão com que a porta *A* da entrada *S* tenha uma saída 0 e a porta *B* da entrada *R* tenha uma saída 1, forçando o biestável ao estado 1.

Se *S* e *R* forem ambas 1 e um pulso de relógio (nível 1) ocorrer, o próximo estado do circuito é indeterminado.

Um problema importante com este circuito é que as entradas *R* e *S* devem permanecer inalteradas durante o tempo em que o relógio for um 1. Isto limita consideravelmente o uso

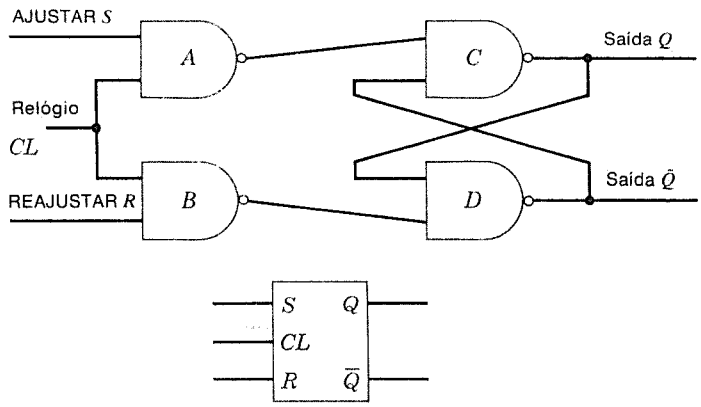


Fig. 4.19 Forma limitada de biestável sincronizado constituído de quatro portas NE.

do circuito, conduzindo a circuitos mais complicados para dar mais flexibilidade ao projetista.

É melhor que o biestável assuma o estado indicado pelas entradas *R* e *S* precisamente no instante em que ocorre o bordo crescente positivo (ou dianteiro) do relógio e para que o próximo estado do biestável não seja afetado pelas mudanças que ocorrem nas entradas depois deste instante. Em conseqüência, a maioria dos biestáveis nas linhas bipolares é comandada pelo bordo, não ativado pelo nível. Estudaremos os biestáveis desta categoria na próxima seção.

Quando são usados biestáveis de circuitos lógicos da espécie mostrada na Fig. 4.19, é prática comum ter vários sinais de relógio diferentes, cada um consistindo de pulsos positivos, mas com as seções positivas ocorrendo em tempos diferentes. Um sistema de relógio deste tipo é chamado *relógio de múltiplas fases.*

4.11 O BIESTÁVEL MESTRE-ESCRAVO

Num computador, os biestáveis estão ligados, ou diretamente ou através de portas, a outros biestáveis, e, quando ocorrem bordos de relógio, as mudanças resultantes numa saída de um certo biestável podem conduzir a uma mudança na entrada de um outro biestável. Este problema da corrida é geralmente atenuado usando-se biestáveis que tiram a informação de entrada aplicada no instante da mudança de valor do relógio ou no instante pouco antes da mudança de valor do relógio. Naturalmente, estes biestáveis são logicamente mais complicados, mas a eliminação da preocupação quanto aos efeitos da interação entre os circuitos provocou um uso difundido de tais biestáveis.

Para incorporar este "fator de segurança" na operação do biestável, circuitos de componentes discretos usam capacitores para armazenar os sinais de entrada e assim "atenuar" e retardar as mudanças nos valores da entrada.

Os circuitos integrados usam lógica pura para realizar o mesmo efeito. O projeto mais utilizado em biestáveis de CI é o indicado na Fig. 4.20. Esta figura mostra um biestável tipo *JK* que realmente consiste de dois biestáveis mais algum circuito lógico. Os dois biestáveis são chamados "mestre" e "escravo". Na Fig. 4.20(*a*), o biestável mestre é constituído pelas portas NE mais à esquerda, e o biestável escravo, pelas portas NE mais à direita.

A saída do biestável muda no bordo crescente negativo do pulso de relógio. A sincronização básica é mostrada na Fig. 4.20(*a*). Primeiro, no bordo crescente positivo e durante a seção positiva do pulso de relógio, o biestável mestre é carregado pelas duas portas NE mais à esquerda. Em seguida, durante o bordo crescente negativo do sinal de relógio, as duas portas NE mais à direita carregam os conteúdos do biestável mestre no biestável escravo depois de as duas portas NE de entrada serem desativadas. Isto significa que o biestável mestre não mudará de valor enquanto o relógio estiver baixo (0); portanto, o escravo permanece ligado a um biestável estável com um valor tomado durante a seção positiva do pulso de relógio.

Uma explicação mais detalhada da ação do biestável é a seguinte: Se o sinal de relógio for baixo, as duas portas NE de entrada terão ambas saídas 1; portanto, o biestável mestre não muda de estados, já que ele é um biestável de porta NE e pode ser ajustado ou limpo (reajustado) somente por entradas 0.

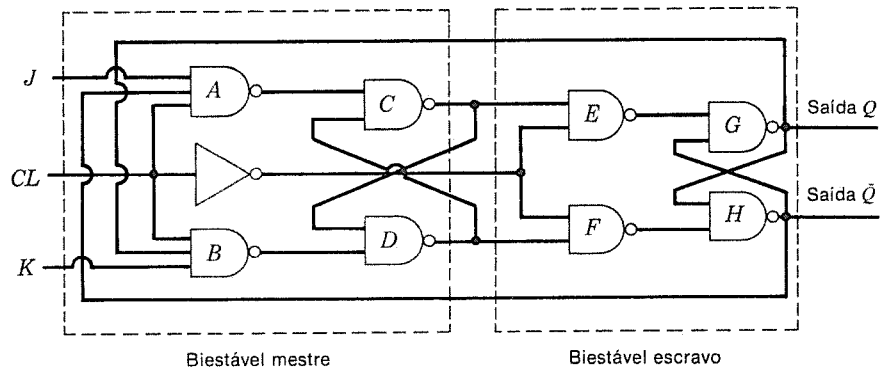
Ao mesmo tempo, enquanto o sinal de relógio for baixo (um 0), o inversor faz com que as entradas para as portas NE *E* e *F* forcem o valor do biestável mestre para dentro do biestável escravo. A situação é estável. O mestre não pode mudar, e o biestável de saída está "escravizado" ao mestre.

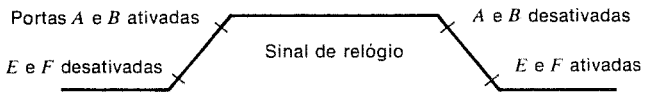
Quando o relógio inicia positivo, no entanto, os limites do circuito são arranjados de modo que primeiro as portas NE *E* e *F* sejam desativadas e as portas NE *A* e *B* para o biestável mestre sejam em seguida ativadas. Isto significa que o declive natural do sinal de entrada mais os retardos e valores do circuito asseguram que as duas portas NE *E* e *F* tenham ambas 1 em suas saídas antes que qualquer mudança no biestável mestre possa afetar o biestável escravo.

Quando o sinal de entrada for um 1, então, o biestável mestre aceitará informação proveniente das entradas *J* e *K*, e o biestável escravo está agora isolado do mestre e não mudará os estados, independente de mudanças no mestre.

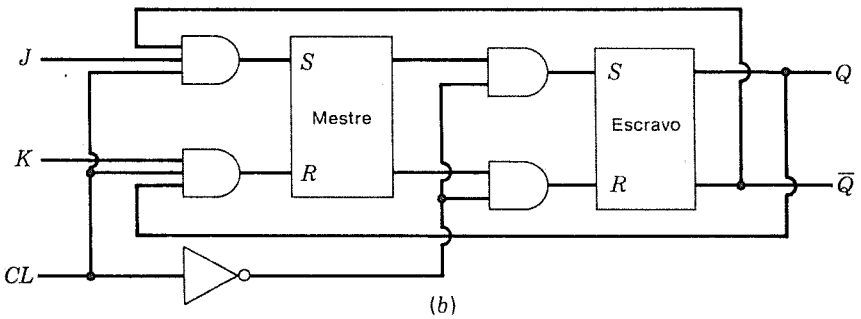
A operação do biestável mestre está de acordo com as seguintes regras quando o nível do relógio for um 1:

1. Se *J* e *K* estiverem ambas em níveis 0, as duas portas NE de entrada terão saídas 1 e o biestável mestre não mudará de valores.
2. Se a entrada *J* for um 1 e a entrada *K* um 0, o biestável mestre irá para seu estado 1, com a porta NE superior do mestre tendo uma saída 1.



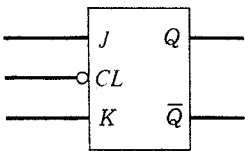


(a)



(b)

ENTRADAS		ESTADO DO BIESTÁVEL APÓS RELÓGIO
J	K	
0	0	Inalterado
0	1	0
1	0	1
1	1	Complementos



(c)

Fig. 4.20 (*a*) Arranjo de portas para biestável mestre-escravo; (*b*) biestável *RS* equivalente a (*a*); (*c*) tabela de estados e símbolo do biestável mestre-escravo.

3. Se a entrada *J* for um 0 e a entrada *K* um 1, o biestável mestre irá para seu estado 0, com a porta NE superior tendo uma saída 0.
4. Se *J* e *K* forem ambos 1, o biestável mestre tomará o valor do complemento do biestável escravo (isto é, um 0 se o escravo for 1 ou um 1 se o escravo for um 0).

Quando o sinal de relógio se torna negativo em seu nível 0, primeiro as portas NE de entrada para o biestável mestre são desativadas, isto é, cada saída vai para um 1, em seguida as portas NE *E* e *F* são ativadas (pelo sinal de relógio invertido). Isto faz com que o estado do biestável mestre seja transferido para dentro do biestável escravo.

Os efeitos de tudo isto são mostrados na tabela de estado seguinte na Fig. 4.17(*d*), que indica que o biestável é do tipo *JK* ativado por um sinal de relógio crescente negativo.

*4.12 PROJETO DE CONTADOR

O projeto de um contador em seqüência através de um determinado conjunto de estados é direto, usando a técnica que vamos expor.

Primeiro, é construída uma tabela de contador, listando os estados a serem assumidos. Admitamos que nosso objetivo seja um contador na seguinte seqüência:

estado inicial →	0	0	0	
	1	1	1	
	1	0	1	isto se repete
	1	1	0	
	0	0	1	
	0	1	0	
	0	0	0	
	1	1	1	
	1	0	1	

Isto requererá três biestáveis. Primeiro, usamos biestáveis do tipo *RS* e denominamos os três biestáveis C_1, C_2 e C_3. Agora, cada biestável tem duas entradas, uma entrada *R* e uma entrada *S*. Portanto, damos à entrada *R* em C_1 o nome C_1^R, à entrada *S* em C_1 o nome C_1^S, à entrada *R* em C_2 o nome C_2^R e etc. até C_3^S.

O problema agora consiste em deduzir expressões de álgebra booleana para cada uma das seis entradas nos biestáveis. Para isso, colocamos a tabela de estados numa *tabela de projeto de contador*, listando os três biestáveis e seus estados e listando também as seis entradas nos biestáveis. Isto é mostrado na Fig. 4.21(*a*).

Os valores de C_1^S, C_1^R, C_2^S, C_2^R, C_3^S, C_3^R são então preenchidos com emprego da seguinte regra:

Regra de projeto

Consideremos uma fila na tabela e um biestável específico:

1. Se o estado do biestável for um 0 na fila e um 0 na fila seguinte, colocar um 0 na coluna da entrada *S* e um d na coluna da entrada *R* para as entradas do biestável.
2. Se o biestável for um 1 na fila e um 1 na fila seguinte, colocar um 0 na coluna da entrada *R* e um d na coluna da entrada *S*.
3. Se o biestável for um 0 na fila e mudar para um 1 na fila seguinte, colocar um 1 na coluna *S* e um 0 na coluna *R*.
4. Se o biestável for um 1 na fila e mudar para um 0 na fila seguinte, colocar um 1 na coluna *R* e um 0 na coluna *S*.

Como exemplo, consideremos o biestável C_1 na Fig. 4.21(*a*). O biestável tem valor 0 na primeira fila e muda para um 1 na segunda fila. Portanto, colocamos um 1 em C_1^S na fila 1 e um 0 em C_1^R na fila 1. Na fila 2, C_1 tem valor 1 e permanece um 1 na fila 3. Portanto, preenchemos um d em C_1^S e um 0 em C_1^R.

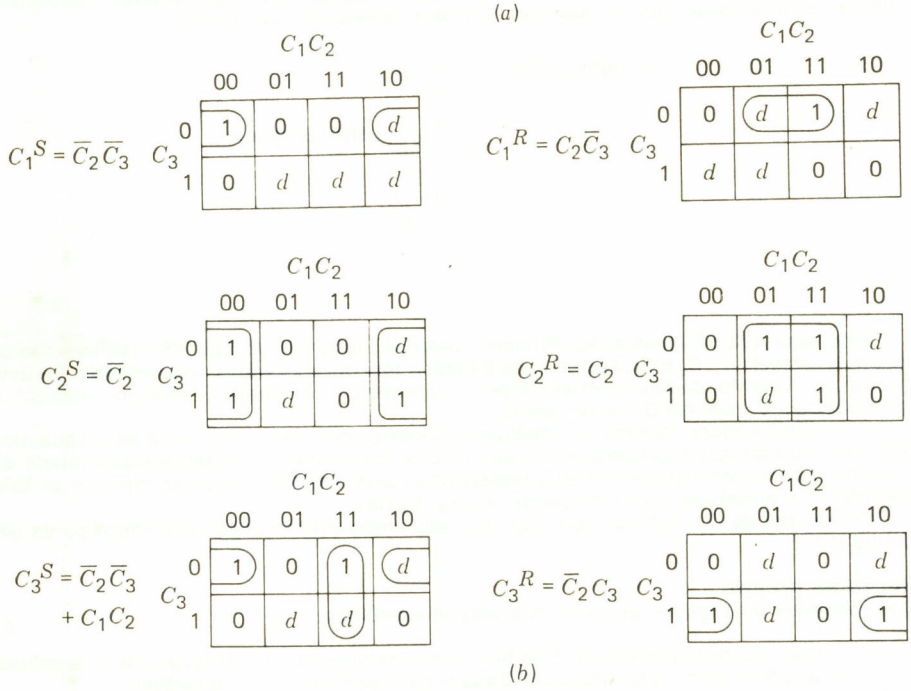
Uma verificação da Fig. 4.21(*a*) mostrará como esta regra funciona.

O projeto do contador agora prossegue formando-se uma expressão de algébra boo-

*Esta seção pode ser omitida numa primeira leitura.

C_1	C_2	C_3	$C_1{}^S$	$C_1{}^R$	$C_2{}^S$	$C_2{}^R$	$C_3{}^S$	$C_3{}^R$
0	0	0	1	0	1	0	1	0
1	1	1	d	0	0	1	d	0
1	0	1	d	0	1	0	0	1
1	1	0	0	1	0	1	1	0
0	0	1	0	d	1	0	0	1
0	1	0	0	d	0	1	0	d

(*a*)



(*b*)

Fig. 4.21 Projeto de um contador que usa biestáveis tipo *RS*.

leana a partir desta tabela para $C_1{}^S$, $C_1{}^R$, $C_2{}^S$, $C_2{}^R$, $C_3{}^S$, $C_3{}^R$ e depois minimizando cada expressão. Isto é mostrado na Fig. 4.21(*b*). Observemos que quaisquer estados não-utilizados do contador podem ser incluídos como d no mapa, pois eles são saídas não-especificadas (*don't-cares*) porque o contador nunca os utiliza.

A etapa final consiste em desenhar o diagrama-bloco do contador usando as expressões mínimas. O projeto final deste contador é indicado na Fig. 4.22.

Quando são usados os biestáveis tipo *JK*, a técnica é muito semelhante, exceto que as entradas sejam $C_1{}^J$, $C_1{}^K$, $C_2{}^J$, $C_2{}^K$, $C_3{}^J$, $C_3{}^K$. A regra do projeto é então

Regra do projeto

Para um determinado biestável numa fila selecionada, as entradas *J* e *K* no biestável são preenchidas como segue:

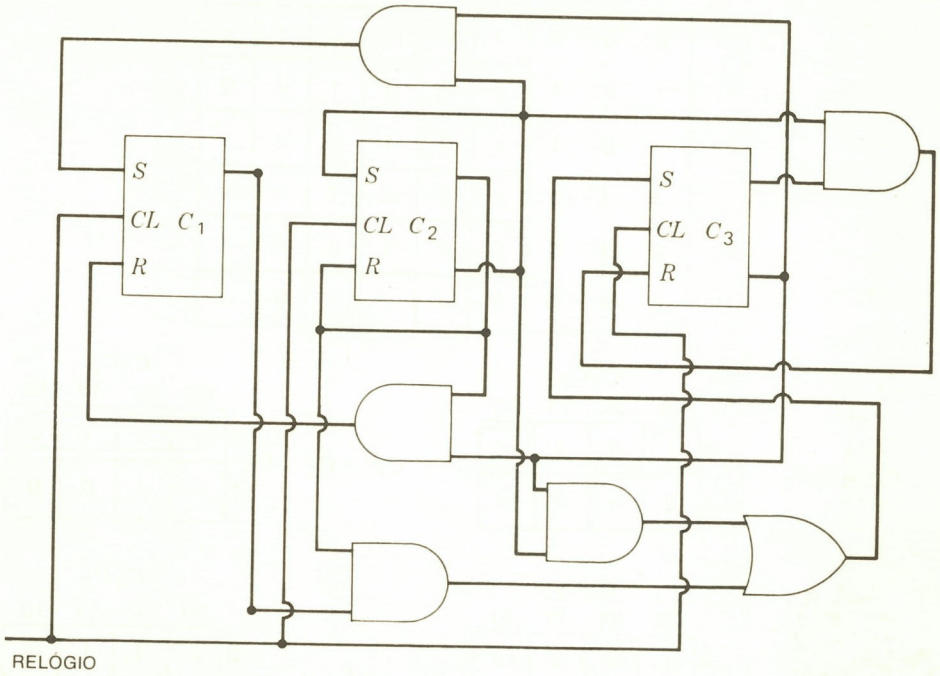


Fig. 4.22 Contador com biestáveis do tipo *RS*.

1. Se o biestável for um 0 na fila e permanecer um 0 na fila seguinte, colocar um 0 na coluna de entrada *J* e um d na coluna de entrada *K*.
2. Se o biestável for um 1 na fila e permanecer um 1 na fila seguinte, colocar um 0 na coluna de entrada *K* e um d na coluna de entrada *J*.
3. Se o biestável for um 0 na fila e mudar para um 1 na fila seguinte, colocar um 1 na coluna de entrada *J* e um d na coluna de entrada *K*.
4. Se o biestável for um 1 na fila e mudar para um 0 na fila seguinte, colocar um d na coluna de entrada *J* e um 1 na coluna de entrada *K*.

O procedimento do projeto prossegue novamente preenchendo-se a tabela de projeto, formando-se as expressões de álgebra booleana e minimizando-as e, finalmente, desenhando o diagrama-bloco do contador. A quaisquer combinações não-utilizadas podemos dar valores d no mapa para auxiliar a minimização. Um projeto completo do contador é mostrado na Fig. 4.23. Observemos que este biestável *JK* requer menos portas do que o projeto de *RS*, e isto geralmente é o que ocorrerá, embora, às vezes, o diagrama deva ser o mesmo.

EXERCÍCIOS

1. Desenhar um conjunto de formas-de-onda para S e R e X e $\bar{X}$ (como na Fig. 4.2) de modo que o biestável na Fig. 4.1 deva ter os sinais de saída 0011010 na linha de saída 1.
2. Se a saída $\bar{X}$ do biestável X for conectada a um inversor, a saída do inversor será sempre a mesma que a saída X do biestável. Verdadeiro ou falso. Por quê?
3. Desenhar um conjunto de formas-de-onda para S e R (como na Fig. 4.2) de modo que o biestável na Fig. 4.1 deva ter os sinais de saída 101110001 na linha de saída.
4. Se a porta E conectada à entrada R de X_1 na Fig. 4.3 apresentar defeito de modo que sua saída seja sempre 0, deveremos esperar, após algumas transferências, que X_1 deva estar sempre no estado _____. Por quê?
5. Desenhar um conjunto de formas-de-onda como na Fig. 4.5(*d*) para o biestável na Fig. 4.5(*b*) de modo que o biestável deva ter os sinais de saída 0010110 em sua linha de saída.

C_1	C_2	C_3	C_1^J	C_1^K	C_2^J	C_2^K	C_3^J	C_3^K
0	0	0	1	*d*	1	*d*	1	*d*
1	1	1	*d*	0	*d*	1	*d*	0
1	0	1	*d*	0	1	*d*	*d*	1
1	1	0	*d*	1	*d*	1	1	*d*
0	0	1	0	*d*	1	*d*	*d*	1
0	1	0	0	*d*	*d*	1	0	*d*

$C_1^J = \overline{C}_2\overline{C}_3$

C_3 \ C_1C_2	00	01	11	10
0	1	0	*d*	*d*
1	0	*d*	*d*	*d*

$C_1^K = \overline{C}_3$

C_3 \ C_1C_2	00	01	11	10
0	*d*	*d*	1	*d*
1	*d*	*d*	0	0

$C_2^J = 1$

C_3 \ C_1C_2	00	01	11	10
0	1	*d*	*d*	*d*
1	1	*d*	*d*	1

$C_2^K = 1$

C_3 \ C_1C_2	00	01	11	10
0	*d*	1	1	*d*
1	*d*	*d*	1	*d*

$C_3^J = C_1 + \overline{C}_2$

C_3 \ C_1C_2	00	01	11	10
0	1	0	1	*d*
1	*d*	*d*	*d*	*d*

$C_3^K = \overline{C}_2$

C_3 \ C_1C_2	00	01	11	10
0	*d*	*d*	*d*	*d*
1	1	*d*	0	1

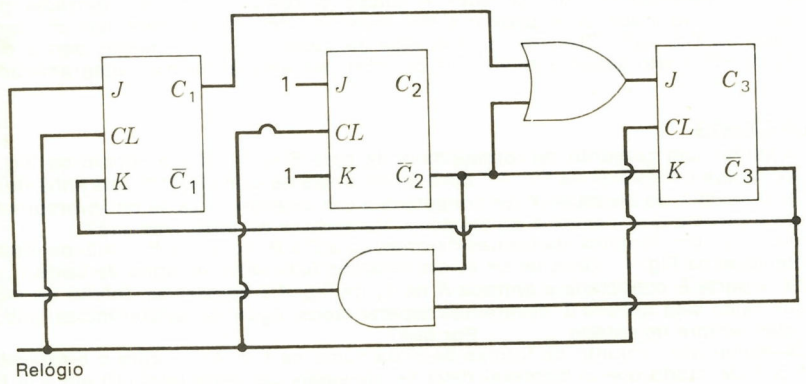


Fig. 4.23 Projeto de um contador com biestáveis *JK*.

6. Desenhar um conjunto de formas-de-onda como na Fig. 4.5(*c*) para o biestável na Fig. 4.5(*a*) de modo que o biestável deva ter os sinais 10111001 em sua linha de saída 1.
7. Desenhar uma forma-de-onda de entrada como na Fig. 4.6 de modo que X_3 deverá ter o sinal de saída 000011010 se X_1, X_2 e X_3 forem iniciados no estado 0.
8. Na Fig. 4.6, se o biestável X_2 "crava" (isto é, falha) em seu estado 1, X_3 terá uma saída 1 após (*a*) 1 pulso de relógio, (*b*) 2 pulsos de relógio, (*c*) 3 pulsos de relógio e para cada pulso de relógio posteriormente.
9. O contador binário na Fig. 4.7 usa biestáveis que atuam nas transições positivas. Desenhar um diagrama-bloco de um contador binário com "bolhas" nas entradas *CL* (isto é, com biestáveis que atuam nas entradas *CL* ou de relógio crescente *negativo*).
10. Se a linha X_3 (a saída de X_3) estiver conectada à linha INPUT (ENTRADA) na Fig. 4.6, estará formado um *contador em anel*. Se este circuito for iniciado com $X_1 = 0$, $X_2 = 1$ e $X_3 = 1$, desenhar a forma-de-onda em X_1, X_2 e X_3 para seis pulsos de relógio.
11. Redesenhar a Fig. 4.7 como ela está, mas colocar bolhas em cada entrada *CL* de X_1, X_2 e X_3 [isto é, fazer o mesmo desenho, mas usar o biestável na Fig. 4.5(*b*) em vez do da Fig. 4.5(*a*)]. Redesenhar as formas-de-onda na Fig. 4.7 para este circuito.
12. O contador no Exercício 11 conta para cima ou para baixo?
13. Fazer uma simples mudança na Fig. 4.7, conectando a saída do biestável X_2 à entrada *CL* de X_3 em vez da saída $\bar{X}_2$ de X_2. Redesenhar agora as formas-de-onda na Fig. 4.7 para esta configuração mudada.
14. Depois de responder ao Exercício 13, usando o biestável na Fig. 4.5(*a*), projetar um contador que conte como segue:

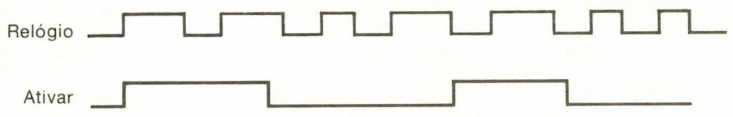
X_3	X_2	X_1
0	0	0
0	1	1
0	1	0
1	0	1
1	0	0
1	1	1
1	1	0
0	0	1
0	0	0
0	1	1
0	1	0
1	0	1
.	.	.

15. Depois de responder ao Exercício 13, usando apenas o biestável na Fig. 4.5(*a*), projetar um contador que conte como segue:

X_3	X_2	X_1
0	0	0
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0
1	1	1

Este contador conta *para baixo*. A Fig. 4.7 conta *para cima*.

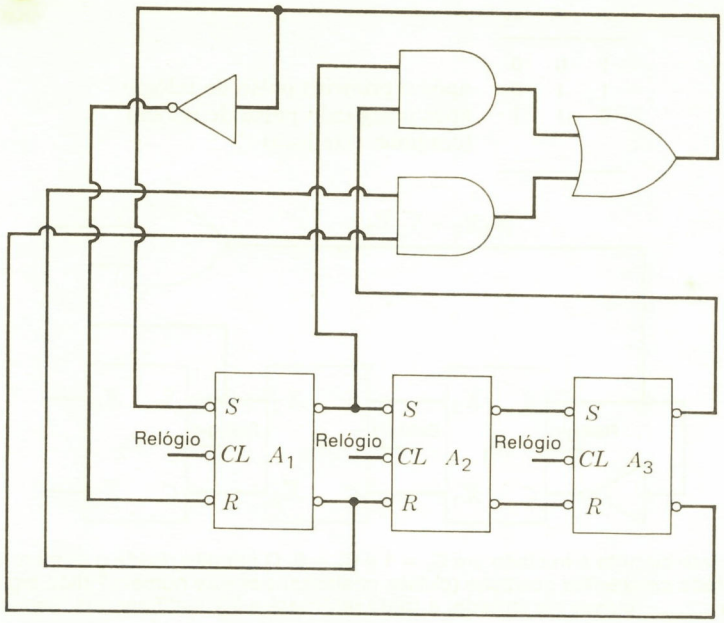
16. Desenhar uma forma-de-onda de relógio e as formas-de-onda de saída da porta E conectada à entrada S de X_1, à entrada R de X_2 e à saída 1 de X_1, X_2 e X_3 na Fig. 4.8.
17. Na Fig. 4.10(*a*), quando a linha ENABLE (ATIVAR) for um 1, o contador conta na ocorrência de um bordo de relógio crescente negativo. Desenhar as formas-de-onda de saída de FF0, FF1 e FF2 para estas formas-de-onda. Iniciar os biestáveis em 0.



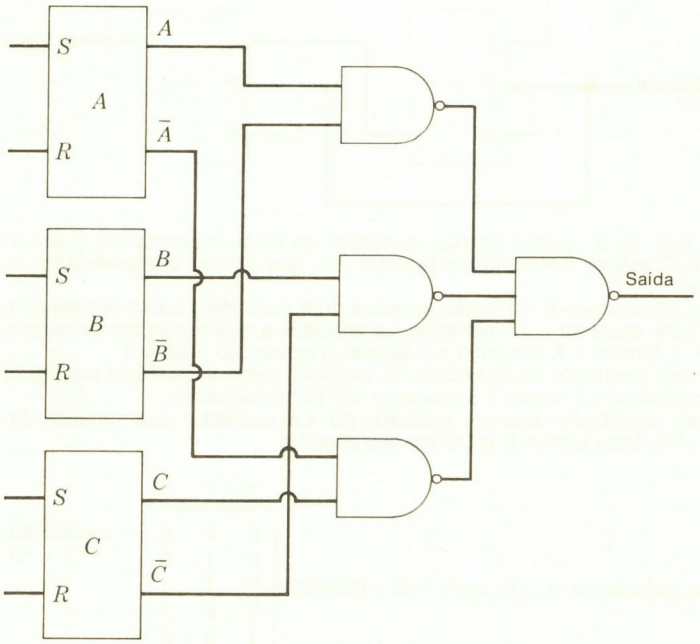
18. Para a Fig. 4.10(*b*), desenhar as formas-de-onda de ativar para cima, ativar para baixo e relógio de modo que o contador inicie em 000 e conte como segue:

X_3	X_2	X_1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
0	1	1
0	1	0
0	1	1

19. Suponhamos que a porta E na Fig. 4.11(*a*) falhe de modo que sua saída seja sempre um 1. Escrever a seqüência ou mostrar a forma-de-onda através da qual este contador irá em resposta aos sinais de relógio.
20. Suponhamos que a saída da porta E na Fig. 4.11(*a*) esteja conectada à entrada K de X_4 em vez de à entrada J. Como contará o computador?
21. Na Fig. 4.11(*b*), o "vai um" do bloco 1 ao bloco 2 vai de 1 a 0 a cada _____ pulsos de relógio. O "vai um" do bloco 3 ao bloco 4 vai de 1 a 0 a cada _____ pulsos de relógio.
22. *(a)* Se substituirmos o bloco 2 na Fig. 4.11*(b)* pelo contador de ondulação de quatro estágios na Fig. 4.10(*a*), usando a saída 1 de FF3 naquele contador como a linha de "vai um", a linha de "vai um" do bloco 2 irá de 1 a 0 após quantos pulsos de relógio? *(b)* Para a configuração no Exercício 22*(a)*, depois de quantos pulsos de relógio o "vai um" do bloco 2 irá de 0 a 1?
23. Usando os circuitos na Fig. 4.12, projetar um contador binário sincronizado e comandado.
24. Usando os circuitos na Fig. 4.12, projetar um contador de BCD.
25. Usando os circuitos na Fig. 4.12, projetar um circuito porta com entradas A, B e C que terão saída 1 quando $\bar{A}B\bar{C}$ ou $A\bar{B}\bar{C}$ forem 1.
26. Projetar um contador de BCD usando os biestáveis na Fig. 4.12.
27. As entradas para as portas NOU na Fig. 4.15 têm 0 sobre elas quando não conectadas. A conexão de apenas duas entradas de uma porta NOU de quatro entradas dá uma porta que atua como uma porta de duas entradas? Explicar sua resposta. *Nota*: As entradas para portas TTL, tais como na Fig. 4.12, que são deixadas desconectadas, atuam automaticamente como se elas tivessem 1 sobre elas.
28. Uma porta NE de três entradas, como na Fig. 4.12, atua como uma porta de duas entradas atuaria se somente fossem usadas duas entradas? Explicar sua resposta.
29. Projetar o contador na Fig. 4.13 usando os blocos na Fig. 4.15.
30. Dar os estados dos biestáveis no seguinte circuito após cada um dos cinco primeiros sinais de relógio (pulsos) serem aplicados. O circuito é iniciado no estado $A_1 = 0$, $A_2 = 0$, $A_3 = 1$.

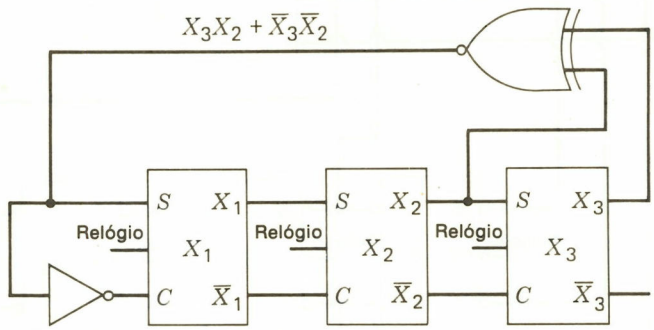


31. Reprojetar o seguinte circuito usando apenas biestáveis *RS* e portas NOU.

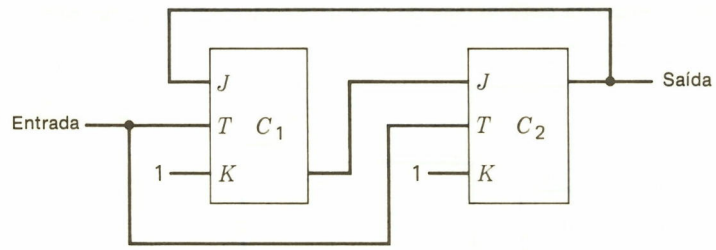


32. O bloco de portas no seguinte circuito constitui uma rede ou circuito combinacional "igual a" que realiza a função booleana $\bar{X}_3\bar{X}_2 + X_3X_2$. Se este conjunto de três biestáveis for iniciado em $X_1 = 1$, $X_2 = 0$ e $X_3 = 0$, qual será a seqüência de estados internos? Como um início, os três primeiros estados podem ser listados como segue:

X_1	X_2	X_3	
1	0	0	
1	1	0	após o primeiro pulso de relógio
0	1	1	após o segundo pulso de relógio
.	.	.	(continue esta lista)



33. O seguinte circuito é iniciado em $C_2 = 1$ e $C_2 = 0$. O circuito divide o número de bordos de entrada crescentes positivos (pulsos positivos) por que número? (Isto significa que, a cada _____ pulsos de entrada, a saída retornará a 0.) Justificar sua resposta.



34. Na Fig. 4.20, se a ligação de realimentação da saída da porta NE *C* até a entrada da porta NE *D* estiver interrompida (aberta), em que estado provavelmente acharemos o biestável?
35. Na Fig. 4.20, por que *E* e *F* estão ativados num nível mais baixo do que *A* e *B*?
36. Na Fig. 4.20, escrever 0 e 1 em todas as entradas e saídas das portas quando a entrada *CL* for 1, *J* for um 1, *K* for um 1 e o biestável estiver no estado 1.
37. Usando seu resultado do Exercício 36, explicar como o biestável na Fig. 4.19 complementa quando *J* e *K* forem 1 e aparecer um bordo negativo.
38. A seguinte seqüência deve ser realizada por um contador que consiste de três biestáveis tipo *RS*. Usar portas E e OU em seu projeto.

A_1	A_2	A_3	
0	0	0	← estado inicial
0	1	0	
0	1	1	
0	0	1	
1	0	0	
1	1	0	
0	0	0	
.	.	.	

A seqüência se repete após este segmento

39. Projetar um contador, usando somente biestáveis *JK*, portas E e portas OU, que conte na seguinte seqüência.

0	0	0	
0	1	0	isto se repete
0	1	1	
1	0	0	
0	0	0	
0	1	0	
0	1	1	
1	0	0	
0	0	0	

40. Projetar um contador que use três biestáveis *JK* designados X_1, X_2 e X_3 e quaisquer portas à sua escolha, que conte como segue:

X_1	X_2	X_3	
0	0	0	estado inicial
0	1	1	após o primeiro pulso de relógio
0	1	0	após o segundo pulso de relógio
1	1	1	após o terceiro pulso de relógio
1	0	1	após o quarto pulso de relógio
0	0	0	após o quinto pulso de relógio
0	1	1	
0	1	0	

41. A seguinte seqüência deve ser realizada por um contador que consiste de três biestáveis *JK*. Usar portas E e OU em seu projeto.

	A_1	A_2	A_3	
	0	0	0	estado inicial
	0	1	1	
A seqüência se repete após este segmento	0	1	0	
	0	0	1	
	1	0	1	
	1	1	0	
	0	0	0	
	.	.	.	

42. Projetar um contador usando três biestáveis *JK* designados X_1, X_2 e X_3 e quaisquer portas à sua escolha, que conte como segue:

X_1	X_2	X_3	
0	0	0	estado inicial
0	1	1	após o primeiro pulso de relógio
0	1	0	após o segundo pulso de relógio
1	1	1	após o terceiro pulso de relógio
1	0	1	após o quarto pulso de relógio
0	0	0	após o quinto pulso de relógio
0	1	1	
0	1	0	

43. O biestável de porta NE na Fig. 4.18 terá que saídas nas linhas 0 e 1 se ambas as SET e RESET forem feitas 0?
44. Se as entradas *S* e *R* na Fig. 4.18 forem ambas feitas 0 e a *S* for feita um 1 seguida por *R*, qual será o estado resultante do biestável?

5

CIRCUITOS DIGITAIS

O conjunto de circuitos eletrônicos e os dispositivos eletrônicos têm sofrido inúmeros aprimoramentos desde que o computador digital surgiu pela primeira vez. Os grandes sistemas de válvulas a vácuo com seus invólucros de vidro e filamentos aquecidos foram comprimidos em pequenos receptáculos plásticos de circuito integrado (CI) por diligentes físicos de estado sólido e projetistas de circuitos eletrônicos. A última década trouxe mesmo uma significativa mudança, e dispositivos e circuitos menores, mais rápidos e a custos mais baixos, estão sendo continuamente apresentados. Mudanças básicas, no entanto, não surgem tão rapidamente, e a maioria dos aprimoramentos é devida agora aos refinamentos da tecnologia existente ao invés de novas idéias radicais. Deve ser notado, porém, que a indústria de circuitos eletrônicos integrados digitais constitui a maior parte da indústria eletrônica, e descrições de novos projetos e aperfeiçoamentos dos antigos projetos continuam a encher as páginas das revistas técnicas sobre computadores.

O estudo e a análise dos circuitos reais usados nos computadores eletrônicos — desde os microcomputadores até os sistemas — de grande porte — estão concentrados na tecnologia de CI. Além disso, os circuitos integrados atualmente em uso geral são, via de regra, variações de diversos projetos básicos de circuito. De fato, é possível dizer-se com segurança que cerca de 95% de todos os computadores contêm agora circuitos que usam as estruturas básicas do Quadro 4.1.

O propósito deste capítulo é de início explicar alguns dos princípios de circuito que se relacionam a todos os circuitos digitais e depois examinar, uma de cada vez, as principais tecnologias de circuito. São dados fatos básicos sobre as propriedades gerais de cada tecnologia de circuito, e são assinaladas algumas das vantagens e desvantagens de cada tipo de circuito. Nenhuma tentativa é feita para ser de grande alcance ou extensão; somente foram selecionados e explicados circuitos representativos.

É possível omitir este capítulo numa primeira leitura sem perda de continuidade, e é também possível amostrar seções selecionadas e examinar algumas, mas nem todas, das linhas de circuito. No entanto, o autor sente que uma inspeção deste assunto vale bem o esforço envolvido, e, tendo gasto muitas horas projetando circuitos, possui uma grande vivência no que concerne ao valor desta matéria. Ele também compreende, no entanto, que este assunto seja muitas vezes ensinado em outros cursos e que as limitações do tempo não permita a inclusão da matéria.

5.1 CIRCUITOS DE COMPUTADOR

Embora um moderno computador eletrônico seja composto de um grande número de componentes eletrônicos, a máquina completa conterá usualmente apenas alguns tipos de circuitos básicos. A teoria atual da construção de computadores dita o emprego de um pequeno número de "blocos" básicos que são utilizados repetidamente. A construção desta espécie simplifica grandemente o projeto das máquinas, aumenta a confiabilidade — uma vez que somente são usados alguns circuitos bem testados — e torna a manutenção das máquinas mais fácil e mais rápida. Os circuitos usados nestes blocos são simples em princípio. Eles são mais simples, por exemplo, do que a maioria dos circuitos usados nos modernos aparelhos de rádio e de televisão. Há diversos tipos de circuitos: circuitos que efetuam operações lógicas em sinais de entrada; elementos de armazenamento, tais como os biestáveis, que armazenam bits de informação; e circuitos acessórios, tais como os excitadores de linha coaxial, circuitos indicadores a lâmpada neon etc.

Os circuitos que serão descritos foram classificados de acordo com as funções que eles executam. Por exemplo, seguindo-se à descrição das características de diodos e de

transistores, o primeiro tipo de circuito a ser descrito é a porta E a diodo. Este é seguido por uma porta E a diodo-e-transistor. Depois, finalmente, é descrita uma porta E toda a transistor. Aqui, a finalidade é, portanto, descrever a função que cada circuito realiza e a maneira pela qual um circuito difere de outro e, simultaneamente, explicar o processo de funcionamento dos circuitos.

Não foi feita nenhuma tentativa para descrever cada configuração de circuito atualmente em uso; somente os circuitos representativos de cada tipo são incluídos na discussão.

5.2 CARACTERÍSTICAS DE DIODOS E TRANSISTORES USADOS EM CIRCUITOS DE CHAVEAMENTO

Antes de analisar os circuitos lógicos, descreveremos sucintamente as características gerais dos diodos e transistores de junção quando usados como dispositivos de comutação. Isto não implica que as características destes dispositivos nos circuitos de chaveamento sejam radicalmente diferentes das usadas nos circuitos de áudio e noutros circuitos; no entanto, em sua maioria, os estudos introdutórios genéricos dos transistores referem-se a eles como dispositivos de amplificação linear. Quando os transistores são usados em circuitos de comutação, eles são geralmente operados nos extremos de seus pontos de operação ou próximo deles. Isto significa que eles são muito freqüentemente operados quer numa condição de "corte" quer numa condição de "saturação". Portanto, algumas das características que são mais importantes para o circuito do computador são de menor importância em outros campos.

Além disso, nenhuma tentativa será feita para explicar a Física dos dispositivos semicondutores de junção. A Física dos semicondutores e a análise detalhada das características de operação dos dispositivos de junção costumam estar de tal maneira interligada às técnicas de fabricação, à geometria das junções e a muitas outras considerações que o assunto tem-se desenvolvido num campo altamente especializado (e muito fascinante), que é tratado com detalhes em outros livros. Para nossos propósitos, e para os da maioria dos usuários de dispositivos de circuito, as características operacionais destes dispositivos é que são importantes, e restringiremos nossa discussão a estas características operacionais.

Em primeiro lugar, serão observadas as características dos diodos semicondutores de modo resumido. Um diodo semicondutor é constituído de duas regiões de material semicondutor† de tipos diferentes ligados por processos químicos elaborados. Um tipo de material semicondutor é chamado material *tipo p*, e o outro, material *tipo n* [refira-se à Fig. 5.1*(a)*].* Quando dois tipos diferentes de material semicondutor são ligados por processos químicos, fica formada uma *junção de semicondutor*, e uma única junção também é chamada *diodo*. Na Fig. 5.1*(a)*, o material tipo *p* é denominado *anodo* do diodo e o material tipo *n* é chamado *catodo* do diodo. A Fig. 5.1*(a)* também mostra o símbolo esquemático do diodo.

Um estudo dos dispositivos semicondutores, ou do estado sólido, explicaria o funcionamento físico interno do diodo. No entanto, nosso único interesse aqui é visualizar o diodo como um componente dos circuitos eletrônicos de chaveamento; portanto, examinaremos o diodo apenas do ponto de vista de suas características elétricas. Quando aplicamos uma tensão elétrica, possivelmente através de um resistor como na Fig. 5.1*(b)*, de modo que o anodo do diodo seja positivo em relação ao catodo, dizemos que o diodo está *polarizado diretamente*. Um diodo que está polarizado diretamente conduzirá corrente um tanto livremente. (Em nossa análise será usada *corrente convencional;* esta corrente flui do pólo positivo para o negativo.)

A Fig. 5.1*(c)* mostra uma curva característica típica de um diodo semicondutor de germânio e uma do diodo de silício. A maioria dos diodos em uso nos computadores é constituída de diodos de silício. Observemos que, quando polarizado diretamente, o diodo de silício apresenta uma queda de tensão da ordem de 0,7 V.** Observemos também que a região de polarização direta no gráfico permanece à direita das ordenadas.

Quando o catodo de um diodo for positivo em relação ao anodo, diz-se que o diodo está *polarizado inversamente*, e ele apresentará uma resistência muito alta ao fluxo da cor-

---

†Um semicondutor tem uma condutividade aproximadamente compreendida entre a dos metais, que são "bons" condutores, e a dos isolantes, que conduzem deficientemente.

*O material do tipo *n* tem um excesso de portadores (elétrons) de corrente negativa, enquanto que o material do tipo *p* tem um excesso de portadores (buracos) de corrente positiva. Assim, a corrente flui prontamente quando o material do tipo *p* é positivo e o material do tipo *n* é negativo. Alternativamente, flui muito pouca corrente quando o material do tipo *p* é negativo e o material do tipo *n* é positivo.

**Contanto que flua 1 mA ou mais de corrente (1 mA = 1 miliampère, ou 1×10^{-3}A).

rente [ver Fig. 5.1*(b)* e *(c)*]. Esta resistência varia de dezenas a centenas de megohms.*

Um diodo pode, portanto, ser imaginado como uma espécie de chave eletrônica que está fechada (deixa passar livremente a corrente) quando polarizado diretamente e abre (quase não deixa passar corrente) quando polarizado inversamente.

Os transistores mais comumente usados em circuitos de computadores são os *transistores de junção*, construídos ou de uma região de material *n* entre duas regiões de material *p* ou de uma região de material *p* entre duas regiões de material *n*. O primeiro destes é chamado transistor *pnp*, e o segundo, transistor *npn*. A Fig. 5.2*(a)* e *(b)* mostra os dois tipos de transistores. As regiões de material *n* e *p* recebem nomes; para o transistor *pnp*, a região média é chamada *base*, enquanto que as duas regiões do tipo *p* são chamadas *coletor* e *emissor* e, mais adiante, identificadas pelos símbolos esquemáticos mostrados na Fig. 5.2*(a)*. Para o transistor *npn*, o material tipo *p* constitui a base, e as duas regiões de material tipo *n* constituem o emissor e o coletor.

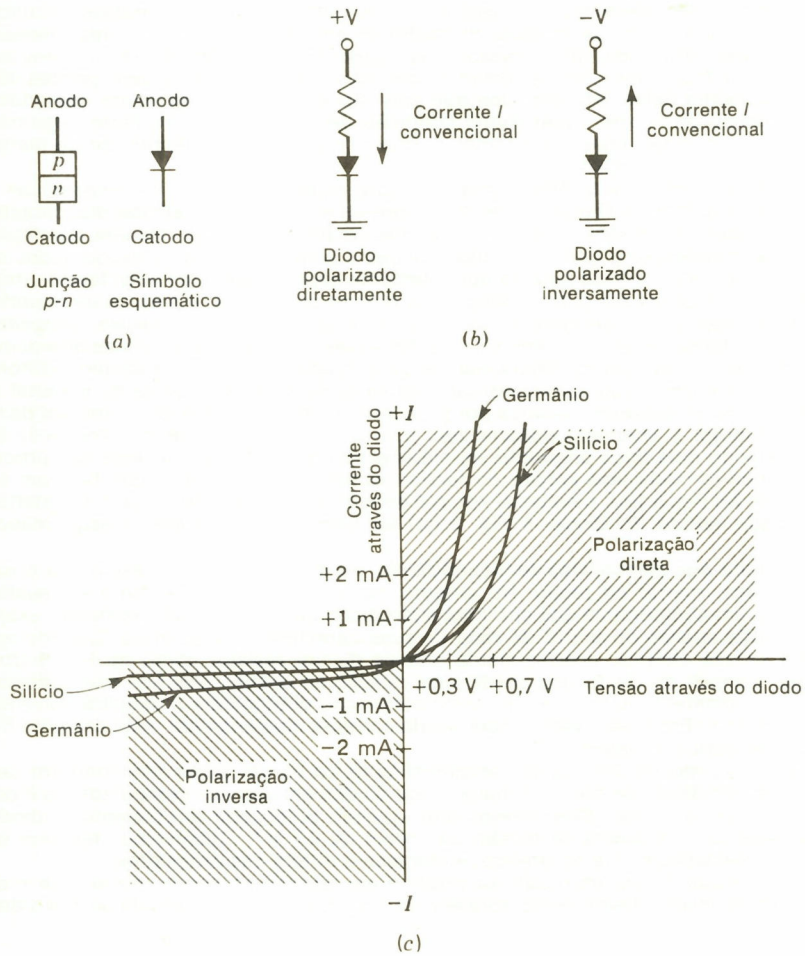


Fig. 5.1 Características e símbolo de diodo.

*A parte da curva dos diodos de silício que está abaixo do nível 0 de corrente está um tanto distorcida, uma vez que um diodo deixa passar menos do que poucas centenas de nanoampères quando polarizado inversamente (1 N = 10^{-9} A).

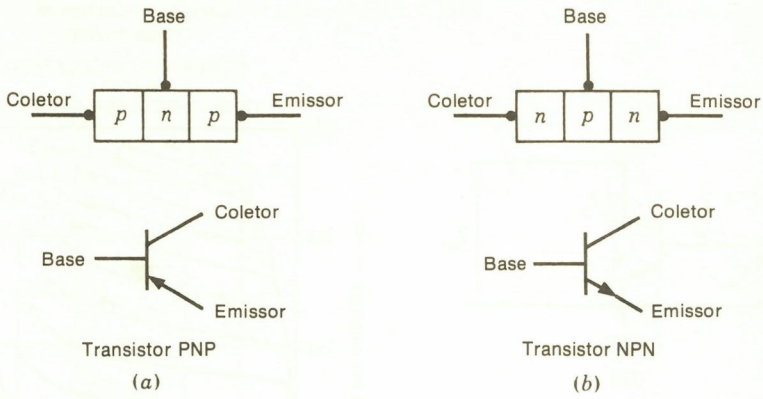


Fig. 5.2 Designações e símbolos de transistor.

As características de operação de um transistor são geralmente estudadas conectando-se o transistor num circuito, variando-se as correntes e tensões e plotando-se as tensões e correntes de saída resultantes.

A Fig. 5.3 mostra um transistor *npn* num circuito com emissor em terra (ou emissor à massa). As correntes e tensões no circuito são identificadas como segue: A corrente *para dentro da* base do transistor é chamada I_b; a corrente *para dentro do* coletor é chamada I_c; a corrente *para dentro do* emissor é chamada I_e; a tensão do coletor é chamada V_{ce} (esta é a tensão entre o coletor e o emissor); a tensão na base é chamada V_{be} (tensão entre base e emissor).

Nesta configuração, o transistor pode ser operado de três modos: (1) *região ativa,* (2) *região saturada* e (3) *região de corte.*

Em primeiro lugar, vamos definir estes termos e, em seguida, examinemos as características do transistor em cada região de operação.

1. *Região ativa.* O transistor *npn* na Fig. 5.3*(a)* está na *região ativa* quando a corrente flui para dentro da base, isto é, quando I_b é positiva e quando V_{ce} é mais positiva do que V_{be} (o coletor é mais positivo do que a base).
2. *Região de saturação* (ou saturada). O transistor *npn* na Fig. 5.3*(a)* está na *região saturada* quando flui corrente positiva para dentro da base (I_b é positiva) e quando V_{ce} é igual a ou menor do que V_{be} (o coletor é menos positivo do que a base).
3. *Região de corte.* O transistor *npn* é dito estar *cortado* quando não flui nenhuma corrente para dentro da base ou quando flui corrente para fora da base (quando I_b é ou 0 ou negativa).

Ativa Quando são usados transistores nos circuitos de televisão ou de rádio, eles são mais freqüentemente operados na região ativa, porque uma pequena variação na corrente da base causa então uma grande variação na corrente de coletor, tornando possível, portanto, amplificar a entrada. Para examinar este fato, é apresentada uma família de curvas na Fig. 5.3*(b)* para um transistor amplamente usado, o 2N706.

Para examinarmos um exemplo do ganho de corrente, observemos que, se, na Fig. 5.3*(b)*, a tensão V_{ce} for mantida em 8 V, e I_b, a corrente da base, estiver em 0,45 mA, então I_c, a corrente de coletor, será de aproximadamente 15 mA. Se I_b for variada para 0,85 mA, uma variação de 0,4 mA, I_c variará para 30 mA, uma variação de 10 mA. Portanto, a corrente de coletor variou mais do que a corrente da base por um fator 30.

Na região ativa, a junção base-emissor está polarizada diretamente e a junção base-coletor está polarizada inversamente. Isto significa que a corrente flui facilmente da base para o emissor, mas que a corrente base-para-coletor é pequena. Quando a corrente é aumentada através da junção base-emissor, no entanto, isto estimula o fluxo da corrente através da base desde o coletor até o emissor. O fato de que uma pequena variação na corrente da base produz uma grande variação na corrente do coletor ao emissor constitui o "segredo" da utilidade do transistor como um amplificador. A base de um transistor é fabri-

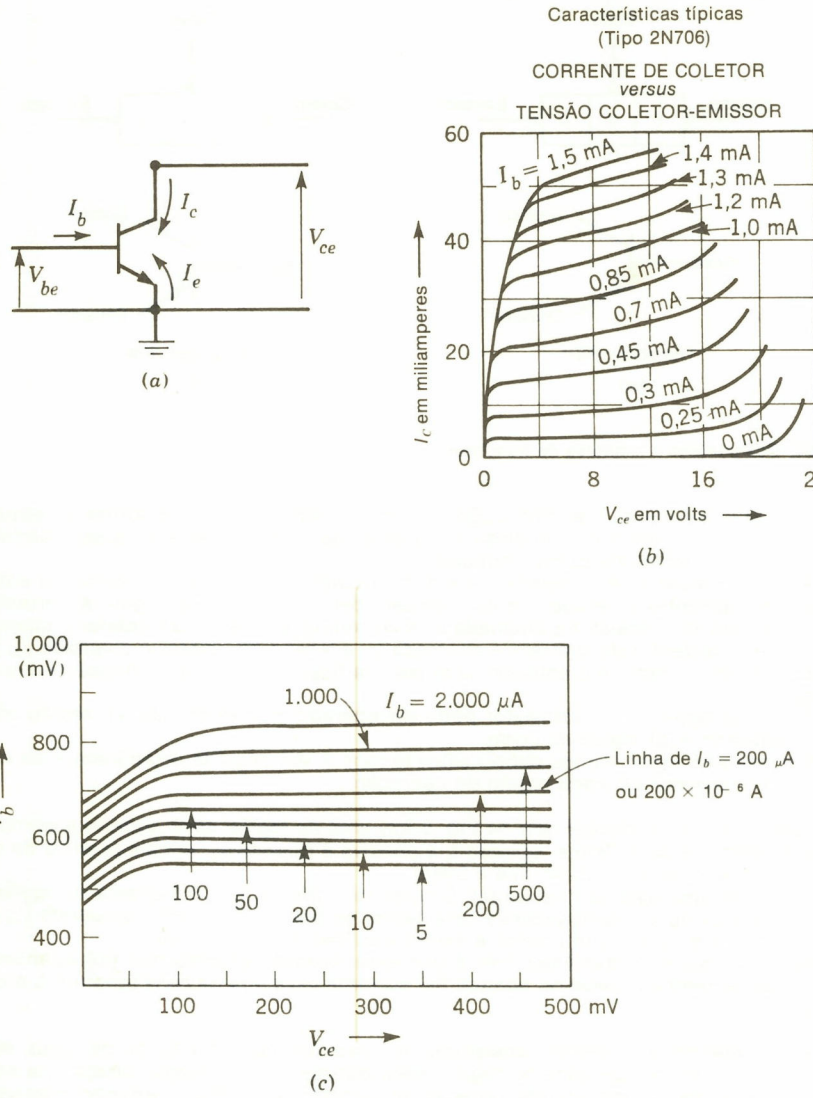


Fig. 5.3 Configuração de emissor-comum de transistor *npn* e sua operação.

cada com pequeníssima espessura, o que estimula o fluxo de corrente pela base (do coletor ao emissor), uma vez que a área da base começa a "romper-se" (isto é, a permitir o fluxo de corrente) devido à corrente da base ao emissor. Observemos que esta corrente de base-emissor também controla a corrente de coletor-emissor.

Saturada Quando a corrente I_b é positiva, mas V_{ce} é menor do que V_{be}, então as junções base-coletor e base-emissor estarão ambas polarizadas diretamente e o transistor "parecerá" dois diodos polarizados diretamente. A corrente fluirá livremente em ambas as jun-

ções da base e da base-coletor.†

Corte Quando flui para dentro da base uma corrente negativa ou quando não flui corrente (isto é, V_{be} é negativa), o transistor está cortado e virtualmente não flui corrente no coletor ($I_c = 0$). Isto pode ser visto na Fig. 5.3*(b)*, pois, com 0,25 mA na base, fluirá uma corrente menor que 3 mA no coletor e, para I_b igual a 0, virtualmente não flui corrente no coletor ($I_c = 0$).* Se V_{be} for negativa (a base é negativa em relação ao emissor), ambas as junções do transistor serão polarizadas inversamente e virtualmente não fluirá corrente no circuito (I_c e I_b serão 0).

Examinemos agora a operação do circuito na Fig. 5.4*(a)*. Um resistor de carga de 500 Ω foi acrescentado ao circuito de coletor e um resistor de entrada de 2.000 Ω, ao circuito da base. O gráfico de um sinal de entrada-saída para este circuito é mostrado na Fig. 5.4*(b)*.

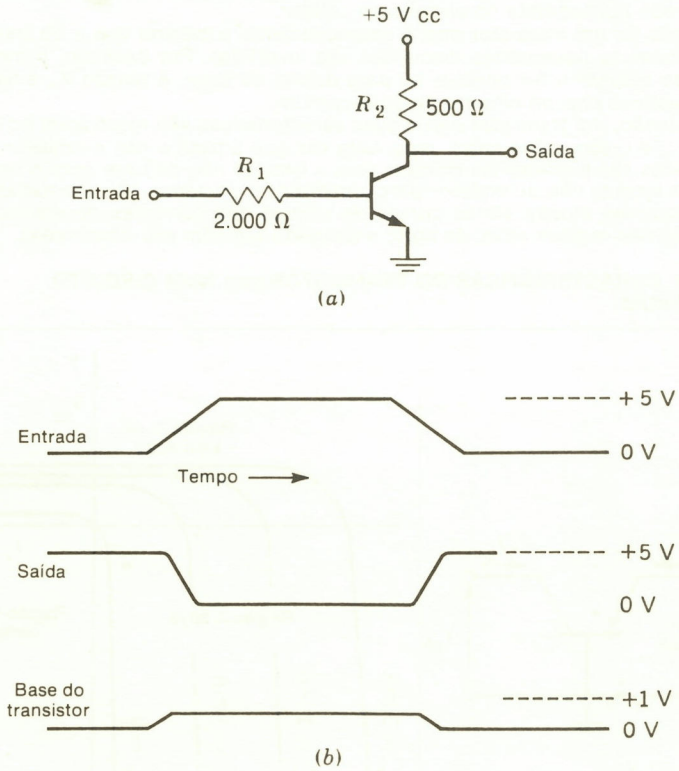


Fig. 5.4 O inversor a transistor. *(a)* Circuito inversor; *(b)* formas-de-onda para o circuito inversor.

Na Fig. 5.4*(b)*, a entrada começa em 0 V, de modo que não flui corrente de base no transistor e ele está em corte. Em conseqüência, a saída está em + 5 V, já que não flui corrente no coletor e não há queda de tensão em R_2.

Como a entrada começa positiva, começa a fluir corrente para dentro da base, e o transistor passa para a região ativa. Isto faz com que flua corrente no circuito do coletor, e esta corrente atravessa o resistor R_2, e, portanto, a saída começa a cair. Em algum ponto, a

†Na Fig. 5.3*(c)* são mostradas algumas curvas de um transistor saturado. Observemos que as tensões de coletor são menores do que as tensões da base. Observemos também que as tensões da base são muito "achatadas" para amplas faixas de tensão de coletor. Por exemplo, com uma corrente I_b da base de 1.000 μA (microampère) (1 μA = 10^{-6} A), a tensão da base será cerca de 0,75 V quando o coletor for de 0,1 a 0,5 V.

A ligeira "dobra" nas extremidades das curvas na Fig. 5.3(b)* é devida à ruptura na junção nas tensões mais elevadas. O transistor deve ser operado com tensão de coletor menor do que +10 V.

saída cairá abaixo da tensão na base, que será mantida em aproximadamente 0,8 V — a junção base-emissor está bem polarizada diretamente pela corrente através do resistor R_1 — e o transistor estará saturado.

Quando a tensão na entrada retornar a 0 V, o processo é invertido; o transistor passa da saturação até a região ativa e é novamente cortado.

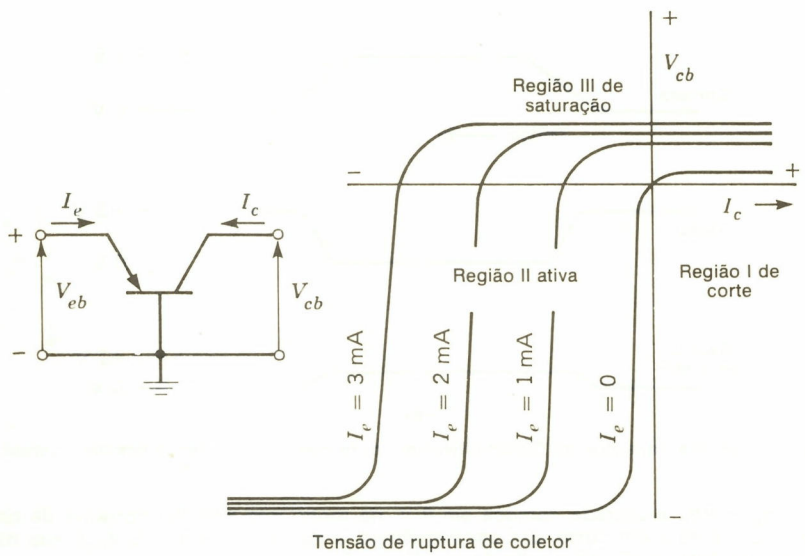
Observemos que a saída do transistor é uma réplica aproximada "invertida" da entrada; assim, o circuito é geralmente chamado um *inversor.*

Nota: Deve ser feito um comentário geral. Um transistor *npn* está, estritamente falando, "cortado" quando a corrente da base for 0 ou negativa, o que significa, para nossos circuitos, que a tensão V_{be} da base é 0 ou negativa. Na prática geral, um transistor é dito estar cortado quando flui corrente "desprezível". Certamente, isto não é preciso, nem pode ser feito assim. O que se quer dizer é que flui tão pouca corrente para dentro da base que virtualmente não flui corrente no circuito do coletor.

A operação de um transistor *pnp* é essencialmente a mesma que a do transistor *npn*, exceto que todas as polaridades discutidas são invertidas. Por exemplo, o transistor *pnp* estará cortado quando I_b for positiva ou para dentro da base. A tensão V_{ce} é feita negativa para os transistores *pnp* no circuito de base-comum.

Em conclusão, um transistor *pnp* e suas características são mostrados no Quadro 5.1. As correntes I_e e I_c são como antes, mas, uma vez que a base e não o emissor está polarizada, as tensões são medidas do emissor para a base (e não da base para o emissor) e do coletor para a base (e não do emissor para o coletor), e, portanto, são chamadas V_{eb} e V_{cb}. O Quadro 5.1 também mostra várias curvas de valores das correntes de emissor. Algumas características das regiões ativa, de corte e saturada também são observadas.

QUADRO 5.1 CARACTERÍSTICAS DO TRANSISTOR *pnp* NUM CIRCUITO DE BASE-COMUM



DEFINIÇÕES

1. *Corte* O transistor *pnp* é definido como estando "cortado" quando a corrente de entrada que flui do emissor (I_e) do transistor é 0, ou quando flui corrente do emissor, e quando o potencial do coletor é negativo em relação à base. A região de corte na figura acima permanece à direita da curva traçada para uma corrente de emissor igual a 0. Esta região de corte é designada como Região I na figura e se estende verticalmente para cima do eixo dos *x* do gráfico. Uma observação importante a

fazer é que a junção emissor-base está polarizada inversamente. Para o transistor *pnp*, isto significa que a base é positiva em relação ao emissor. (Para o transistor *npn*, a base era negativa.)

2 *Ativa* Nesta região, o transistor *pnp* tem uma polarização emissor-base direta ou normal (o emissor é positivo em relação à base), e o coletor é negativo em relação à base. A região ativa é designada como Região II na figura. Esta é a região normal de operação do transistor, e a maioria dos amplificadores a transistor opera nesta região.

3 *Saturação* Quando o coletor for positivo em relação à base, diz-se que o transistor está *saturado*. Esta é a Região III na figura anterior.

CARACTERÍSTICAS DOS TRÊS MODOS DE OPERAÇÃO

O transistor apresenta características de entrada e de saída diferentes em cada uma das três regiões. Observando-se novamente a figura, as três regiões de operação têm as seguintes características:

1 Quando o transistor está cortado, o emissor e o coletor são ambos negativos em relação à base. Em conseqüência, tanto o emissor quanto o coletor aparecem como grandes resistências, e flui pequena corrente tanto do coletor quanto do emissor. No entanto, fluirá alguma corrente inversa por causa da corrente de fuga através da superfície e corrente inversa através da junção.* A magnitude desta corrente varia com a tensão através da junção e com a temperatura da junção, mas geralmente permanece em algum lugar entre 1 e 100 nA para os transistores de silício.

2 Na região ativa, o transistor apresenta todas as características comuns aos circuitos amplificadores de sinal. O transistor possui um ganho de potência desde a entrada até a saída nesta região, e a corrente que flui no circuito de coletor é uma função da corrente para dentro do emissor. O ganho de corrente do emissor ao coletor é designado por α, a relação entre a pequena variação na corrente do emissor e a variação resultante na corrente de coletor $\Delta I_c/\Delta I_e$, com a tensão de coletor mantida constante. Este valor geralmente permanece entre 0,9 a 0,99 para os transistores de junção. Para um transistor particular, o valor de α varia um tanto em toda a região ativa.†

Se a tensão positiva de coletor for aumentada além de um certo ponto, conhecido como *tensão de ruptura*, a corrente de coletor aumentará rapidamente sem que virtualmente haja variação na tensão de coletor. Isto pode ser causado ou por avalanche dos portadores de minoria ou por um outro fenômeno, conhecido como *transposição*, que ocorre quando a região de carga espacial do coletor se estende ao emissor. Devem ser tomadas precauções para verificar se os transistores não são operados com tensões que causarão ruptura.

3 Quando o transistor *pnp* está saturado, tanto o emissor quanto o coletor são positivos (polarizados diretamente) em relação à base, e ambas as junções emissor-base e coletor-base apresentam uma resistência pequena à passagem ou fluxo da corrente.

5.3 O SEGUIDOR DE EMISSOR

Há três configurações de transistor que produzem um ganho de potência da entrada à saída. As configurações de base-comum e emissor-comum são mostradas na Fig. 5.3 e no Quadro 5.1; a terceira configuração é a de *coletor-comum* mostrada na Fig. 5.5. Na prática real, o coletor-comum é ligado e usado como na Fig. 5.5*(a)* e *(b)*, que mostra um só resistor no emissor, uma entrada na base e uma saída do emissor. Nesta forma, o circuito é chamado seguidor de emissor, pois, enquanto o nível da entrada não exceder +V ou −V, a saída no emissor será uma réplica exata da entrada. Portanto, o emissor *segue* a base, exceto para um ligeiro deslocamento no nível. (Para o circuito *npn*, o deslocamento está *abaixo* de aproximadamente 0,5 V e, para o circuito *pnp*, está *acima* de aproximadamente 0,5 V.) Devido à sua capacidade em simplesmente seguir o nível do sinal de entrada, o seguidor de emissor é geralmente usado para fornecer ganho de corrente. O seguidor de emissor é operado na região ativa. As características do seguidor de emissor são:

*A corrente de fuga é muitas vezes designada por "corrente de corte" e identificada como I_{co}.
†Observar que $1/1 - \alpha$ dá o ganho de corrente da base ao coletor.

1. O ganho de tensão do circuito é pouco menor que a unidade. Uma variação de 3 V na tensão de entrada produzirá uma variação de aproximadamente 3 V na saída.
2. O nível de cc de saída do seguidor de emissor é aproximadamente igual ao nível de cc de entrada. Se a entrada na base de um seguidor de emissor for um nível de cc de +3 V e o transistor estiver operando na região ativa, o nível da saída no emissor também estará em aproximadamente +3 V cc. Para o circuito na Fig. 5.5*(a)*, o nível de saída no emissor será ligeiramente negativa em relação à base por causa da queda de tensão na junção base-emissor, e o nível de cc de saída "seguirá" efetivamente o nível da entrada.
3. O ganho de corrente do seguidor de emissor é maior do que a unidade. Na maioria dos transistores, o ganho de corrente através do seguidor de emissor será maior do que 25. Uma vez que o ganho de tensão do circuito é pouco menor do que a unidade, e o ganho de corrente é maior do que a unidade, o seguidor de emissor tem ganho de potência.
4. A saída do seguidor de emissor está *em fase* ou segue a entrada. Isto se aplica tanto às correntes de saída e de entrada quanto às tensões de saída e de entrada.
5. O seguidor de emissor possui uma alta resistência de entrada e uma baixa resistência de saída. (A resistência de entrada é usualmente, pelo menos, de uma ordem de grandeza maior do que a resistência de carga no emissor.)

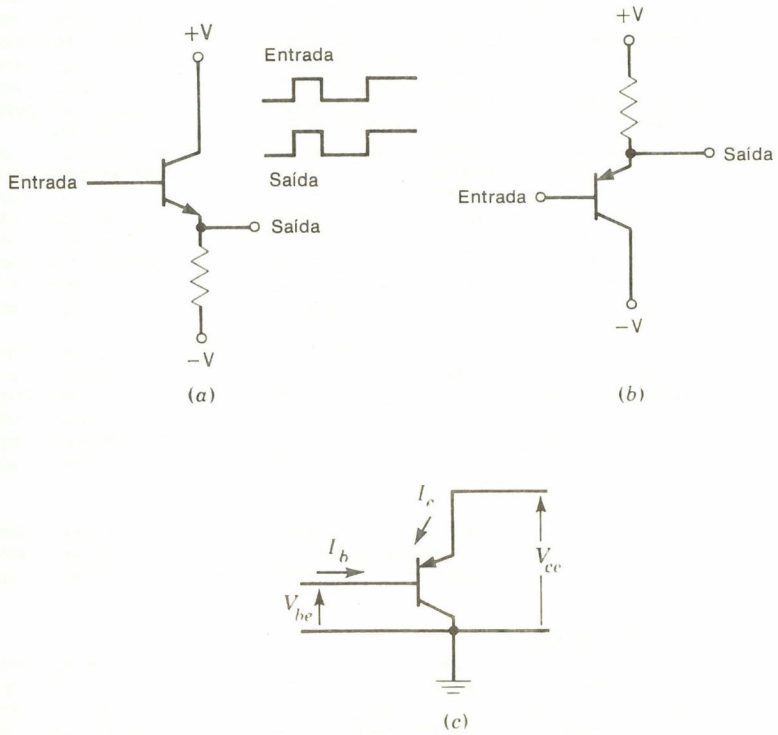


Fig. 5.5 O seguidor de emissor. *(a)* Seguidor de emissor *npn*; *(b)* seguidor de emissor *pnp*; *(c)* configuração coletor-comum.

5.4 A PORTA E

Uma vez que a informação processada por um computador digital é representada por conjuntos de sinais elétricos que podem ser considerados representar valores binários, um determinado sinal pode ser imaginado representar ou um 0 ou um 1. Admitamos que uma tensão contínua (cc) positiva represente um 1, e uma tensão negativa represente um 0. Neste caso, a função de uma porta E com duas entradas é a de produzir um sinal positivo de saída (um 1) somente quando ambas as entradas são positivas (1). Se a porta E tiver três

entradas, a saída será positiva somente quando todas as três entradas forem positivas (em todos os casos, o nível de saída será negativo, representando um 0).

Na maioria dos sistemas, 0 e 1 são representados por um sinal de cc positivo (representando um 1) ou um sinal de cc de 0 V (representando um 0). A função do circuito porta E é então produzir um sinal de cc positivo em sua saída somente quando um sinal de cc positivo for aplicado a todas as entradas do circuito simultaneamente. Se as entradas do circuito forem designadas ou rotuladas *X*, *Y* e *Z*, o circuito produzirá um sinal de cc positivo somente quando um sinal de cc positivo for simultaneamente aplicado a *X* E *Y* E *Z*.

5.5 O CIRCUITO E A DIODO

A Fig. 5.6 ilustra uma porta E a diodo com duas entradas. No circuito ilustrado, um nível de cc de 0 V representa um binário 0, e um sinal de +3 V representa um binário 1. Se ambas as entradas estiverem em 0 V, ambos os diodos estarão polarizados diretamente e a saída será mantida em 0 V pelos diodos. A queda de tensão total através do resistor será de 5 V.†

Se a linha de entrada *X* for a 3 V e a entrada *Y* permanecer em 0 V, o diodo conectado à entrada *Y* ainda estará polarizado diretamente e a saída permanecerá em 0 volts de cc. (Observemos que, neste caso, o diodo de entrada *X* estará polarizado inversamente e haverá uma queda de tensão de 3 V no diodo.) Se a entrada em *Y* for a +3 V e *X* permanecer em 0 V, a saída estará em 0 V de cc.

Quando as entradas *X* e *Y* forem ambas ao nível de +3 V, o nível da saída então crescerá para +3 V e a saída representará um 1.

As respostas da porta E a diodo aos diversos tipos de entradas estão ilustradas na Fig. 5.6. O circuito na Fig. 5.6 podia ter mais do que duas entradas. Neste caso, seria necessário um diodo para cada entrada, e cada diodo adicional deve ser ligado exatamente como os dois diodos na figura. Se quatro diodos estiverem conectados conforme indica a Fig. 5.6, a saída aumentará até o nível de sinal de +3 V somente quando os sinais de entrada para

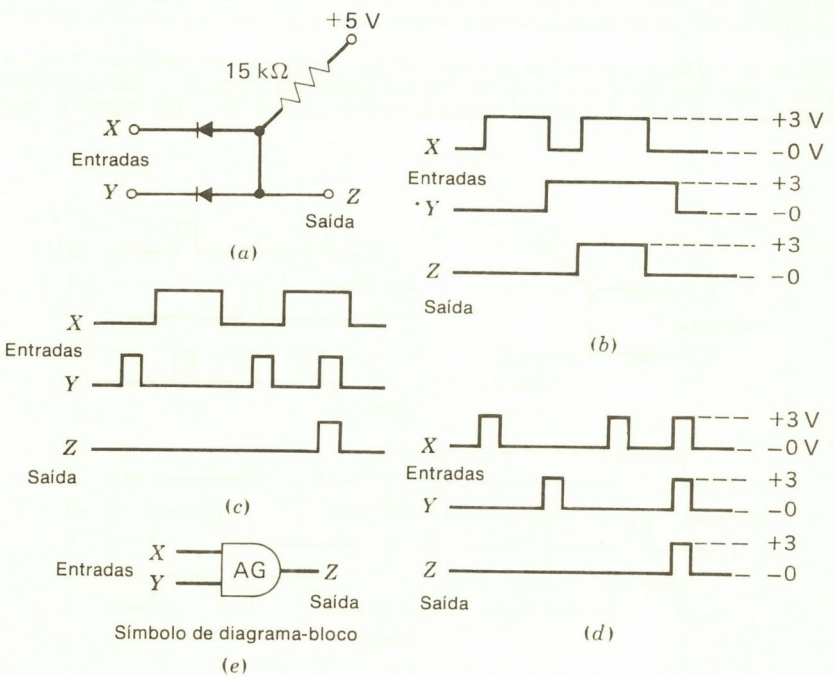


Fig. 5.6 Porta E a diodo.

†Admitimos, para simplificar a explicação, que não há queda de tensão num diodo polarizado diretamente. De fato, nos diodos de silício, a saída deve ser aproximadamente igual a +0,7 V e a queda no resistor de cerca de 4,3 V.

todos os quatro diodos forem positivos. Há um limite prático para o número de diodos que podem ser conectados desta maneira, no entanto, devido ao fato de que os diodos não possuem na realidade uma resistência inversa infinita ou resistência direta zero. Com um grande número de entradas, as resistências finitas diretas e inversas dos diodos causarão níveis de saída variáveis, dependendo do estado das entradas.

5.6 A PORTA OU

A porta OU tem a propriedade de que um sinal que representa um 1 aparecerá na saída se qualquer uma das entradas representar um 1. A Fig. 5.7 ilustra um circuito porta OU a diodo. Há duas entradas (*X* e *Y*) no circuito e uma saída. Os sinais de entrada no circuito consistem de sinais de 0 V representando os 0, e sinais de +3 V representando os 1. Se ambas as entradas do circuito estiverem em 0 V de cc, ambos os diodos estarão polarizados diretamente e a saída do circuito estará em 0 V de cc, representando um 0. Se qualquer uma das entradas do circuito aumentar para +3 V de cc, o diodo nesta entrada estará polarizado diretamente e a saída aumentará para +3 V, representando um 1. O diodo na entrada que permanece em 0 V estará então polarizado inversamente pelo sinal de +3 V na saída. Este circuito tem a propriedade de que o nível de saída estará no nível da entrada mais positiva.

Se ambas as entradas do circuito aumentarem para +3 V, a saída estará novamente em +3 V. (Este circuito é, às vezes, denominado circuito OU *inclusive*, porque a saída é um 1 quando ambas as entradas forem 1.)

Podem ser acrescentadas mais entradas ao circuito ilustrado na Fig. 5.7. Então, é necessário um diodo para cada entrada. Se qualquer uma ou qualquer combinação das entradas aumenta até o nível de +3 V, a saída estará em +3 V.

Como no caso da porta E a diodo, não é prático ter um número excessivo de entradas no circuito porque as resistências diretas e inversas dos diodos são finitas, e diferentes combinações de sinais de entrada produzirão então diferentes níveis de sinal nas saídas.

O símbolo lógico do diagrama-bloco usado para a porta OU também é ilustrado na Fig. 5.7.

Se for invertida a convenção de que um sinal positivo representa um 1 e um sinal negativo representa um 0, o circuito da Fig. 5.7 executará então a função E. Observemos também que, se a porta E que está ilustrada na Fig. 5.6 tiver entradas negativas represen-

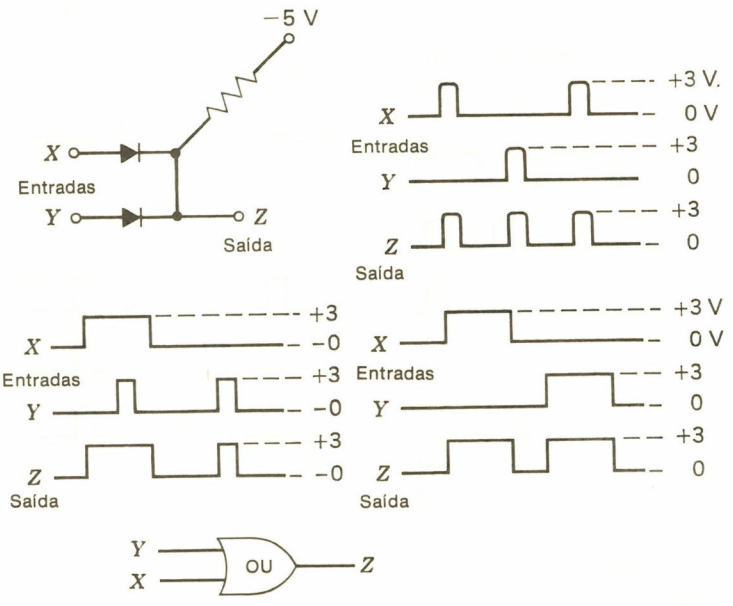


Fig. 5.7 Porta OU a diodo.

tando 1 e sinais de entrada positivos representando 0, o circuito executará então a função OU. Para evitar confusão, o circuito na Fig. 5.7 é, às vezes, denominado porta OU *positiva* ou porta E *negativa*. O circuito na Fig. 5.6 seria então denominado porta E *positiva* ou porta OU *negativa*. A menos que se observe o contrário, consideraremos sinais relativamente positivos para representar 1 e sinais relativamente negativos para representar 0. O termo porta E designará então uma porta E positiva, tal como a da Fig. 5.6, e uma porta OU significará porta OU positiva, tal como a da Fig. 5.7.

5.7 O CIRCUITO INVERSOR

O circuito mostrado na Fig. 5.8 é o de uma linha de circuitos integrados primitiva e outrora muito popular chamada circuitos RTL (lógica resistor-transistor). Estes circuitos são fabricados por quase todos os principais fabricantes de CI e têm a vantagem de serem simples e bem baratos. Os 0 binários são representados por sinais de 0 V, e os 1 binários, por sinais que variam de cerca de +0,8 a +3,5 V. (O motivo da variação nos sinais do 1 será explicado.) A lógica de computadores completos pode ser construída a partir deste circuito simples; portanto, ele é talvez o circuito a transistor mais fundamental.

O X de entrada na Fig. 5.8 pode ser um 0 em binário que é 0 V, caso em que, desde que a base estará no mesmo potencial que o emissor, o transistor estará cortado e não fluirá corrente no circuito do coletor; assim, a saída aumentará até um 1 em binário ou o nível de +3,5 V.

Se a entrada estiver em +3 V, um 1 em binário, o transistor será acionado e conduzirá na saturação, e o coletor estará então em 0 V, mais a queda emissor-coletor, representando um 0 em binário.

Este circuito pode ser usado para ilustrar um ponto importante. Se uma única saída de inversor for conectada a um único inversor como na Fig. 5.9*(a)*, ou a dois inversores como na Fig. 5.9*(b)*, ou a três inversores como na Fig. 5.9*(c)*, tensões de saída que diferem ocorrem no nível 1 na saída do primeiro inversor.

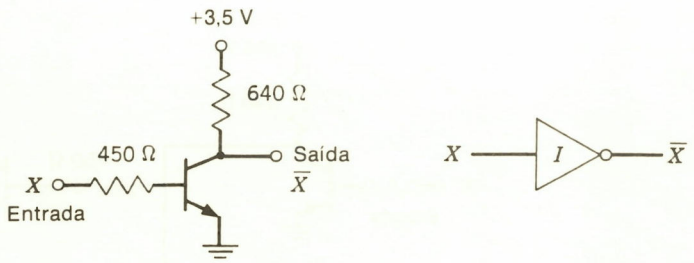


Fig. 5.8 Inversor tipo RTL.

Se a entrada do primeiro inversor for um 0, o transistor estará desligado, e, para a carga do inversor único na Fig. 5.9*(a)*, o resistor na saída do primeiro estágio e a entrada do segundo estágio formam um divisor de tensão em série com a junção base-emissor do transistor do segundo estágio. Admitindo que esta queda seja de 0,5 V, nos resistores de 640 Ω e 450 Ω haverá uma queda de 3 V, e o ponto Z estará em +1,7 V.

Com dois inversores conectados na saída do primeiro inversor, como na Fig. 5.9*(b)*, o ponto Z estará em +1,3 V, enquanto que, com três inversores na saída, como na Fig. 5.9*(c)*, o ponto Z estará em + 1,1 V.

Uma vez que todas estas três tensões (1,7 1,3 e 1,1 V) representam dígitos 1, e já que podem mesmo ser acrescentados mais circuitos, é necessário (1) limitar o número de circuitos conectados a uma saída e (2) representar os 0 e os 1 por tensões situadas nos intervalos.

Como exemplo do primeiro caso acima, a Fig. 4.15 mostra várias embalagens de CI da linha Motorola. Nela está incluída uma embalagem de seis inversores contendo estes inversores num único receptáculo de CI da espécie mostrada nas Figs. 5.8 e 5.9. Os números no interior do MC 789P indicam os números dos pinos; assim, o inversor mais acima possui 14 pinos de entrada e um pino de saída. Os outros números são destinados ao *carregamento*, ou *fator de carga de entrada* e *fator de carga de saída*. Para o inversor mais acima, vemos que a entrada no pino 14 tem um fator de carga igual a 3, e a saída no pino 1 pode excitar

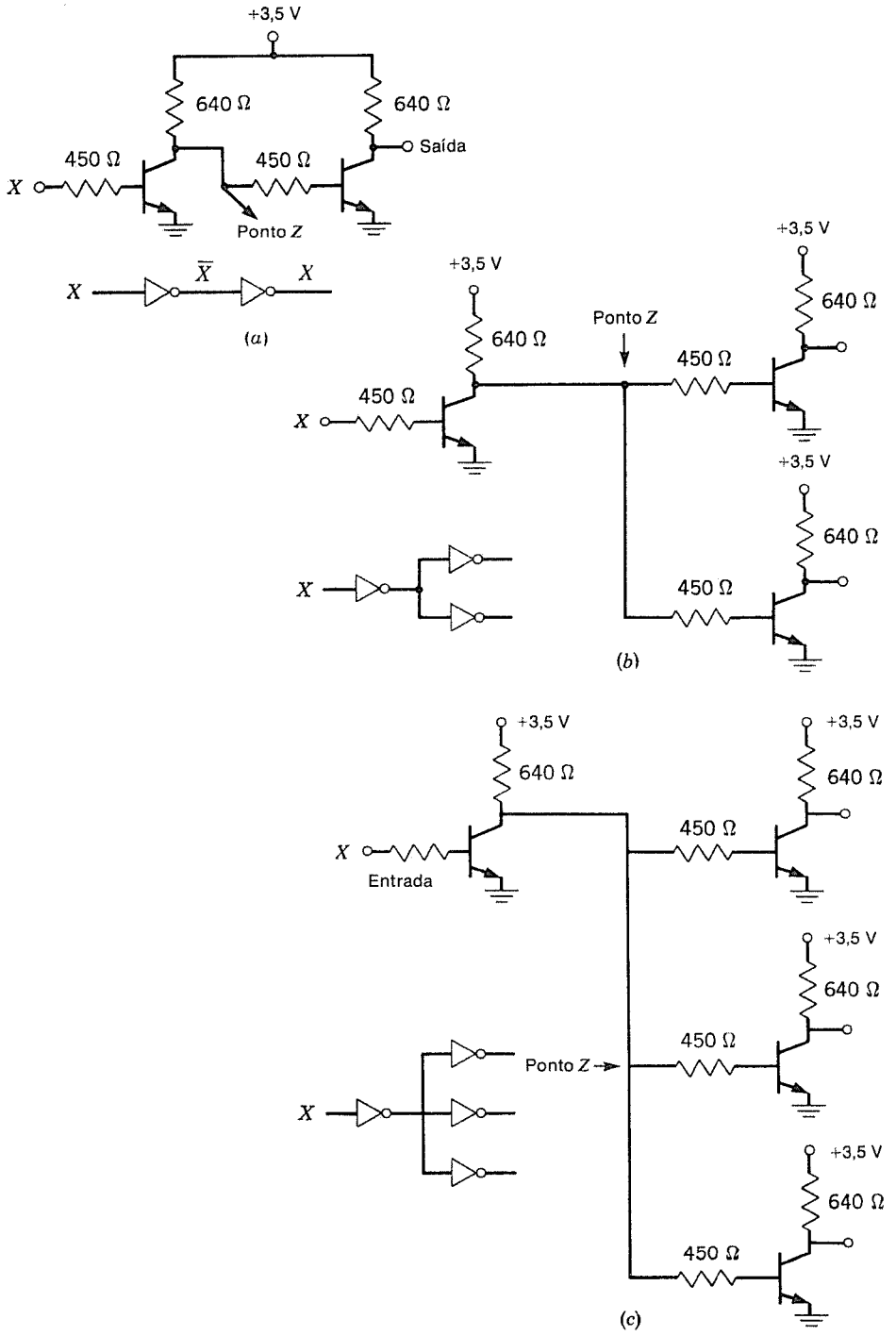


Fig. 5.9 Inversores interligados. *(a)* Dois inversores em série; *(b)* dois inversores para uma única saída; *(c)* três inversores para uma única saída.

16 cargas. Portanto, podemos conectar cinco entradas do inversor a uma única saída do inversor (se 5 × 3 = 15, que é menor do que 16), mas não seis inversores, porque sobrecarregaria uma única saída. Similarmente, a entrada de pino 2 da embalagem do biestável *JK* duplo é uma entrada *CL* para um biestável *JK* com uma carga de cinco; assim, apenas três das entradas *CL* podem ser conectadas a uma saída de inversor. Similarmente, três inversores podem ser conectados ao pino 14 da embalagem do biestável *JK* duplo, já que isto daria um fator de carga igual a 9, que é menor do que a carga 10, a saída 0 do biestável pode suportar.

Quanto aos níveis de tensão, os circuitos são projetados de modo que os níveis inferiores a +0,5 V representam os 0 em binário, enquanto que os sinais superiores a 0,8 V representam os 1. Há, portanto, um intervalo† indeterminado de 0,3 V que oferece alguma proteção contra ruído e uma certa imunidade contra as variações nos circuitos. (O projeto adequado pode ainda proporcionar proteção contra ruído. No entanto, o sistema RTL é geralmente considerado mais sensível a ruído do que alguns outros sistemas.) O nível da fonte de alimentação, V_{cc}, deve ser 3,5 V cc ± 10%. Cada um dos circuitos na Fig. 4.15 pode ser obtido nas embalagens de CI descritas no Cap. 4, tal como o RTL fabricado por qualquer um dos grandes fabricantes.

5.8 PORTAS NOU

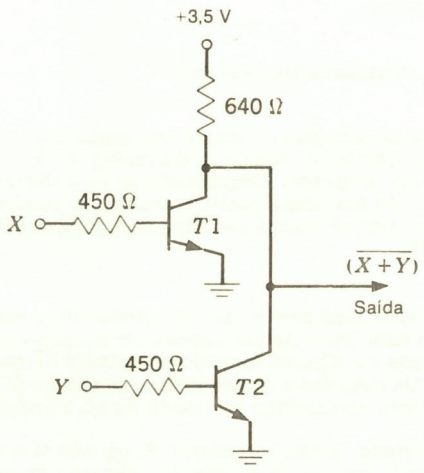
Combinando-se dois dos circuitos na Fig. 5.8, é possível fazer uma porta NOU de duas entradas. A Fig. 5.10 mostra que: os dois coletores estão ligados e compartilham de um só resistor de carga.

Se qualquer uma ou ambas as entradas *X* e *Y* no circuito da Fig. 5.10 forem altas (ou 1), o transistor conectado àquela entrada estará conduzindo e saturado. Isto significa que o coletor estará essencialmente em terra, ou no potencial 0, e a saída será um 0. Por exemplo, se a entrada em *X* estiver em +2 V, T_1 será posto a conduzir e saturado e a saída estará em 0 volts. Se a entrada *Y* estiver por acaso em 0 volts, ela não afetará a saída, e a junção coletor-base de T_2 estará aproximadamente no mesmo potencial.

Isso também se aplica se ambas as entradas estiverem em um 1 ou num alto nível de entrada: ambos T_1 e T_2 estarão saturados, e a saída será um 0.

Se ambas as entradas *X* e *Y* forem por acaso 0, então ambos T_1 e T_2 estarão desligados e a saída estará em +3,5 V.

Uma tabela da verdade para o circuito é mostrada na Fig. 5.10. Isto indica que a pre-



X	Y	Saída
0	0	1
0	1	0
1	0	0
1	1	0

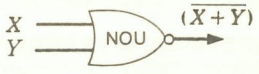


Fig. 5.10 Portas NOU de duas entradas.

†Esta é, às vezes, chamada "região de ambigüidade".

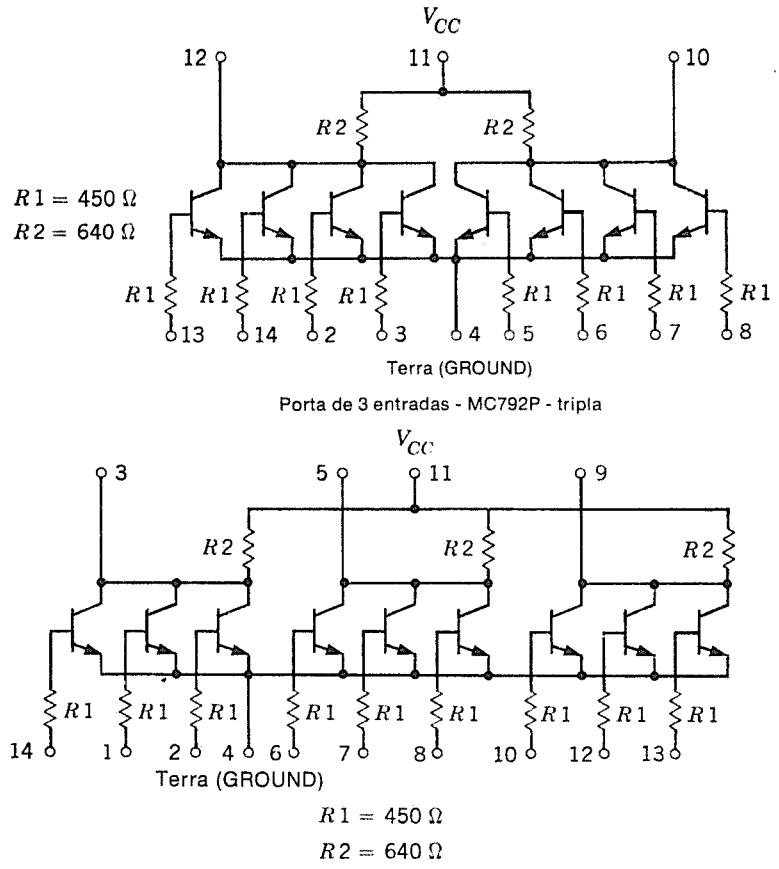


Fig. 5.11 Portas NOU de três e de quatro entradas. (Cortesia de Motorola Corp.)

sença de um 1 em qualquer uma ou em ambas as entradas conduz a uma saída 0; portanto, o circuito é uma porta NOU de duas entradas. Este é um dos circuitos na Fig. 4.15.

A Fig. 5.11 mostra que o circuito pode ser ampliado. Conectando-se os coletores de três inversores, forma-se uma porta NOU de três entradas; quatro transistores podem formar uma porta NOU de quatro entradas; etc. Ambos destes circuitos são circuitos RTL e correspondem ao diagrama-bloco na Fig. 4.15.

5.9 O BIESTÁVEL A TRANSISTOR

A característica básica do circuito multivibrador biestável é que ele possui dois estados estáveis. Esta capacidade de ser estável num dado instante em apenas um de dois estados ou condições possíveis é quase sempre baseada em alguma variação do circuito fundamental a dois transistores mostrado na Fig. 5.12. Os circuitos a dois transistores são arranjados de modo que um dos transistores esteja sempre conduzindo e o outro esteja sempre cortado.

Admitamos que o transistor T_2 esteja cortado. Então, o resistor R_4 de 640 Ω em seu coletor "puxa para cima" a base do transistor T_1 através do resistor R_2. Isto faz o transistor T_1 conduzir e saturar. A saída no coletor de T_1 está então essencialmente em terra, o que aplica cerca de 0 V à base de T_2 através de R_3, mantendo-o desligado. Neste instante, a saída Q está em aproximadamente +1,7 V, dependendo da queda entre base e emissor de T_1, e a saída $\bar{Q}$ está provavelmente em +0,3 V. Este é então o estado 1 do biestável.

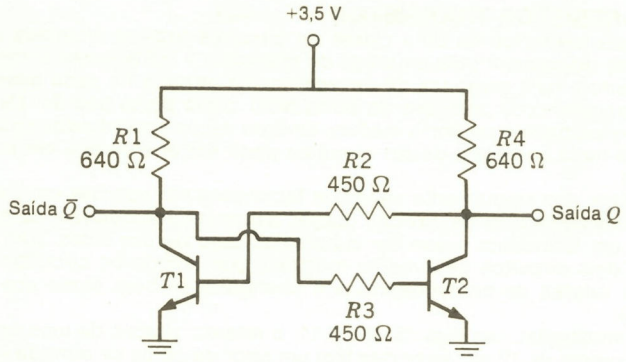


Fig. 5.12 Multivibrador biestável a dois transistores.

Se T_1 estiver cortado, então R_1 e R_2 põem T_2 a conduzir, e o coletor de T_2 então está próximo do potencial de terra, mantendo T_1 cortado. A saída Q está, portanto, próxima do potencial de terra (talvez +0,3 V), e a saída $\bar{Q}$ estará em aproximadamente +1,7 V, e assim o biestável está no estado 0.

Um exame mostrará que o circuito não será estável em qualquer condição e sim apenas no estado 1 ou no estado 0 delineados acima.

Este biestável possui saídas, mas não possui entradas. A Fig. 5.13 mostra um biestável do tipo *RS* com uma entrada SET (AJUSTE) e uma entrada RESET (REAJUSTE). As entradas SET e RESET estão normalmente em 0. Quando uma tensão positiva é aplicada à entrada RESET, o transistor T_4 conduz, e o coletor de T_4 fica num potencial próximo da terra; isto corta o transistor T_1, e a saída de T_1 aumenta, levando T_2 a conduzir. Se a tensão positiva for agora retirada da entrada RESET, o biestável estará no ponto 0, com T_1 cortado e a saída $\bar{Q}$ alta e com T_2 conduzindo e a saída Q baixa.

Uma tensão positiva aplicada à entrada SET levará T_3 a conduzir, pondo em terra a base de T_2 e cortando T_2. Isto levará T_1 a conduzir e, removendo a tensão de entrada positiva da entrada SET, o biestável será deixado no estado 1, com a saída Q alta e a saída $\bar{Q}$ baixa.

A Fig. 5.13 também mostra que este circuito consiste de duas portas NOU com acoplamento cruzado, que é a maneira pela qual se faz um biestável básico do tipo *RS* em lógica RTL.

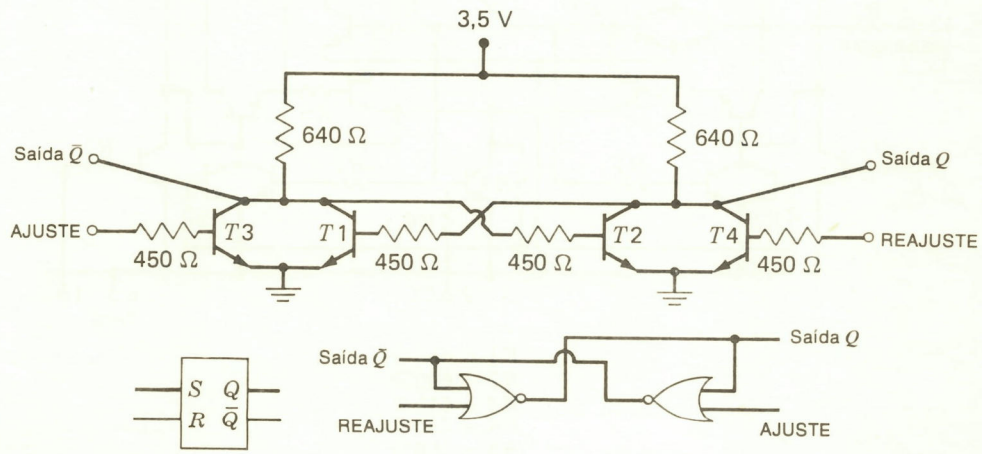


Fig. 5.13 Multivibrador biestável do tipo SET-RESET.

5.10 LÓGICA RESISTOR-TRANSISTOR

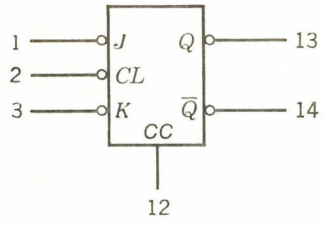
Já vimos diversos exemplos de uma classe de circuitos lógicos chamada circuitos RTL. Estes circuitos se distinguem pelo emprego de resistores e transistores somente como elementos de circuito e pela presença de um resistor na entrada de cada base. Os circuitos são também caracterizados pelo uso da porta NOU como porta padrão. Os circuitos são geralmente de velocidade e potência médias, embora sejam encontrados circuitos de baixa potência, e uma linha completa destes circuitos pode ser encontrada em todos os principais fabricantes.

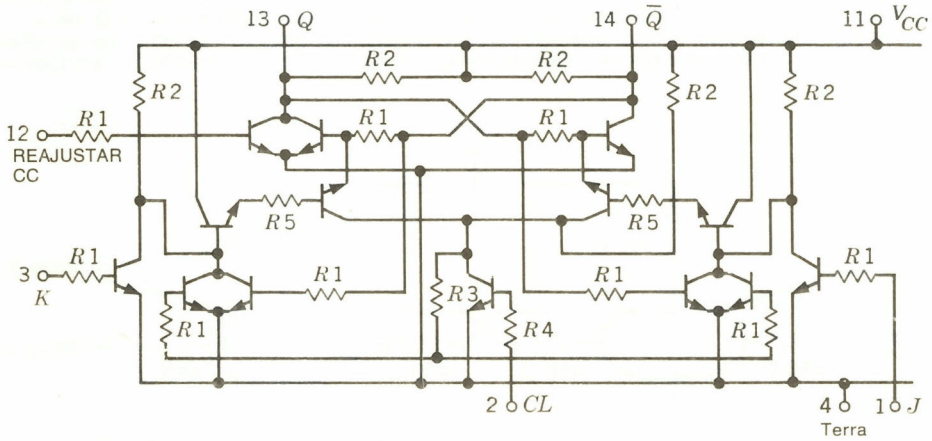
Estes circuitos são amplamente usados e fabricados em padrões em toda a indústria, de modo que as embalagens de circuito usadas sejam intercambiáveis no sentido de que um circuito de um fabricante possa ser substituído pelo de um outro. (Um exame da esquemática dos dois circuitos geralmente mostrará que ambos os circuitos são idênticos com respeito a valores de componentes e à configuração bem como aos números dos pinos externos.)

Nas portas mostradas nas Figs. 5.10 e 5.11, o retardo através de uma única porta descarregada é da ordem de 12 ns. As portas têm um fator de carga de entrada igual a 3 e uma capacidade de saída de 16; portanto, cinco entradas podem ser conectadas a uma única saída.

OPERAÇÃO SINCRONIZADA

J	K	Saída Q
0	0	Mudanças
0	1	Vai a 0
1	0	Porta 1
1	1	Sem Mudanças





$R1 = 450\ \Omega$
$R2 = 640\ \Omega$
$R3 = 510\ \Omega$
$R4 = 225\ \Omega$
$R5 = 300\ \Omega$

Fig. 5.14 Multivibrador biestável de RTL.

A Fig. 5.14 mostra a esquemática de um biestável padrão do tipo *JK* sincronizado. Este biestável tem as entradas e saídas invertidas (observar as bolhas no diagrama-bloco) porque ele é construído de portas NOU que levam à lógica invertida. A Fig. 5.14 mostra uma tabela dos valores da operação sincronizada de entrada. Isto mostra que, se ambas as entradas *J* e *K* forem 1 quando o bordo crescente negativo do pulso de relógio chegar, o biestável permanecerá no mesmo estado. Se, no entanto, ambas as entradas forem 0 quando chegar o bordo negativo, o biestável mudará de estados. Se a entrada no pino 3 for um 1 e a entrada no pino 1 for um 0 quando chegar o pulso de relógio, o biestável passará para o estado 0 (já que as entradas estão invertidas), com o pino 13 alto (uma vez que a saída 1 está invertida) e o pino 14 baixo (já que a saída 0 está invertida). Similarmente, se o pino 3 for um 1 e o pino 1 for um 0 quando o pulso de relógio for negativo, posteriormente o pino 13 estará baixo (um 0) e o pino 14, alto (um 1). Esta entrada invertida/saída invertida requer alguma reflexão, mas é comum em todas as linhas da lógica RTL.

O pino 12 é usado para reajustar o biestável em seu estado 0, e um mais ou sinal 1 é usado para ativar a entrada. Durante a operação sincronizada normal, esta entrada está em terra, ou no potencial 0.

5.11 LÓGICA DIODO-TRANSISTOR

A lógica diodo-transistor (DTL) é novamente uma das linhas de circuito mais freqüentemente usadas. Os circuitos são caracterizados por portas a diodo seguidas por inversores a transistor, o que conduz a uma porta NE como a porta básica. A família DTL tem mais resistência ao ruído que a família RTL, e a lógica é mais "natural", mas os circuitos são mais complicados. A DTL foi a linha de maior vendagem durante alguns anos e é ainda amplamente usada, particularmente nas aplicações industriais em que a proteção contra o ruído é um fator importante na avaliação do desempenho do circuito lógico. A DTL apresenta um desempenho razoavelmente bom nesta área, e há uma linha de circuito chamada HLDTL (ou DTL de alto nível) em que a tensão da fonte de alimentação é de +25 V em vez de +5 V usados na DTL básica. Esta linha de circuitos tem excelente proteção contra ruído e é amplamente usada nas aplicações industriais onde, por exemplo, a maquinaria produz ruído elétrico e há grandes transitórios na linha de alimentação.

A porta DTL básica é mostrada na Fig. 5.15*(a)*. Este circuito particular é padronizado virtualmente em todos os fabricantes. O circuito consiste basicamente de uma porta E a diodo seguida de um inversor a transistor. Há também um transistor adicional (um seguidor-de-emissor) entre a porta E a diodo e a entrada do inversor a transistor. Isto significa que há mais ganho através do circuito de entrada, e, portanto, é requerida menor excitação de entrada. O transistor T_1 extra também significa que um nível de entrada mais alto é necessário para fazer T_2 conduzir, uma vez que existem agora três junções (um diodo e as junções base-emissor de T_1 e T_2) entre a saída da porta E a terra, e, portanto, é requerida uma tensão mais alta para começar a fluir uma corrente significativa. Isto aumenta a proteção contra o ruído.

Para analisar a operação do circuito na Fig. 5.15*(a)*, se qualquer uma das três entradas estiver no potencial de terra, a base de T_1 estará apenas ligeiramente positiva (o valor da queda através de um diodo), e as três junções em série, a junção base-emissor de T_1, o diodo D_1 e a junção base-emissor de T_2, não conduzirão, particularmente se R_3 estiver também fixando a base em terra. Em conseqüência, a saída tenderá a aumentar até a tensão de saída de +5 V.

Se todas as entradas estiverem num nível alto ou nível 1 (3 V ou mais), a saída da porta E a diodo será alta e o emissor de T_1 estará aproximadamente no nível de entrada (observemos que, na junção base-emissor de T_1, há uma queda de tensão aproximadamente igual ao aumento através dos diodos), de modo que T_2 passa a conduzir fortemente e satura. Isto proporciona uma saída baixa ou saída 0. O circuito opera, portanto, como uma porta NE.

Um diagrama-bloco e a esquemática do biestável padrão do tipo *RS* em DTL é mostrado na Fig. 5.15*(b)*. O biestável padrão tem duas portas E de duas entradas que ficam conectadas às entradas *S* e *R*. Em DTL, as entradas não-utilizadas são 1; portanto, usando simplesmente uma entrada *R* e uma entrada *S*, teremos um biestável convencional tipo *RS*. Uma tabela dos valores do estado seguinte para este biestável é mostrada na figura. O biestável é iniciado por bordos crescentes negativos no relógio. As linhas DC RESET e DC SET estão incluídas, e estas estão normalmente num nível alto ou nível 1 (desconectadas, elas serão 1) e são ativadas quando "ligadas à terra" ou fazendo a entrada um 0.

Há um artifício freqüentemente usado com este biestável. Conectando-se a saída Q a uma entrada *R*, e a saída $\bar{Q}$ a uma entrada *S*, tem-se um biestável tipo *JK* (Q complementa com duas entradas 1).

O circuito na Fig. 5.15*(b)* pode ser ainda analisado usando-se o diagrama lógico da Fig.

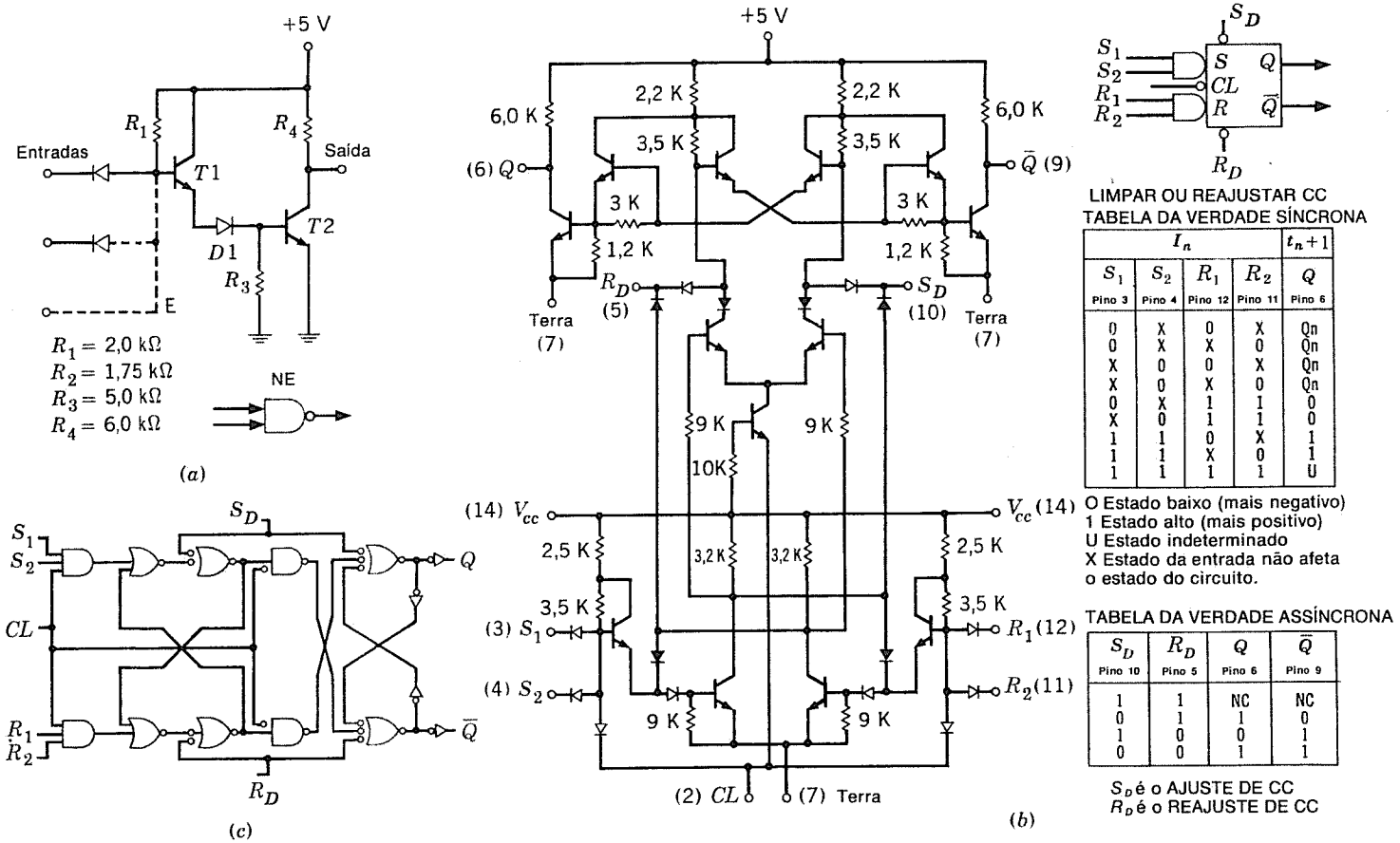


Fig. 5.15 *(a)* Porta DTL básica; *(b)* diagrama-bloco e esquema do biestável padrão do tipo *RS* em DTL; *(c)* diagrama lógico.

5.15*(c)*, que mostra o circuito reduzido a sua forma de bloco. Isto demonstra que o biestável é um circuito do tipo mestre-escravo, conforme descrito anteriormente.

5.12 LÓGICA TRANSISTOR-TRANSISTOR

A linha de circuitos chamada lógica (TTL) transistor-transistor ou família TTL é atualmente a linha de circuito mais usada por causa da alta velocidade dos circuitos TTL. Estes circuitos são caracterizados por um transistor com múltiplos emissores nas entradas. Por outro lado, os circuitos são muito similares à DTL, com uma porta NE padrão, exceto que o estágio de saída contém freqüentemente dois transistores.

A característica que distingue os circuitos TTL de outras linhas de circuito é um transistor com múltiplos emissores no circuito de entrada. O diagrama esquemático para esta espécie de transistor é mostrado como T_1 na Fig. 5.16*(a)*. O transistor com múltiplos emissores tem simplesmente uma área de coletor maior do que o normal e várias junções base-emissor. O transistor de dois emissores na Fig. 5.16*(a)* funciona como o circuito de dois transistores porque o transistor de dois emissores é efetivamente representado por dois transistores com coletor e base comuns. A Fig. 5.16*(b)* e *(c)* mostra o circuito de entrada de TTL com transistor separado e em forma de transistor com múltiplos emissores. Os transistores T_1A, T_1B e T_1C são combinados para formar um único transistor T_1 de múltiplos emissores em *(c)*. Esse circuito opera como segue. Se uma das três entradas *X*, *Y* e *Z* estiver em 0 V, a corrente fluirá através de R_1 e da junção base-emissor de T_1 para a entrada. Isto fixará a base de T_1 em cerca de 0,5 V. A junção base-coletor de T_1 e a junção base-emissor de T_2 formam duas junções em série com o resistor R_3 para terra, e, portanto, a base de T_2 não será positiva em mais do que 0,2 V que é insuficiente para que T_2 passe a conduzir; e uma corrente pequena (fuga somente) fluirá no circuito de coletor de T_1, e T_2 estará cortado. Em conseqüência, não fluirá corrente pelos resistores R_2 e R_3, e o emissor de T_2 estará em terra e o coletor em +5 V.

Se todas as três entradas estiverem num nível de +3,5 V (o nível 1), então a corrente fluirá através de R_1 e da junção base-coletor de T_1 (que está polarizada diretamente) para dentro da base de T_2 levando-o a conduzir. A corrente fluirá através de R_1 e de R_2 até que o transistor T_2 esteja saturado com seu emissor e coletor, ambos em cerca de 2,5 V. Neste instante, observemos que as três junções emissor-base do transistor com múltiplos emissores estarão polarizadas inversamente.

As duas saídas do coletor e do emissor de T_2 operam como tesouras que se fecham quando T_2 está conduzindo e se abrem quando T_2 está cortado. Assim, o emissor e o coletor de T_2 estarão aproximadamente na mesma tensão (tesoura fechada) quando T_2 estiver conduzindo, e o emissor estará em terra e o coletor em +5 quando T_2 estiver cortado (tesoura aberta).

Observemos que T_2 está conduzindo quando as três entradas estiverem altas (1) e T_2 está cortado quando qualquer uma das três entradas estiverem baixas (um 0), e assim a operação do circuito é basicamente a de uma porta NE.

Uma porta NE de TTL completa é mostrada na Fig. 5.17. Este é um circuito TTL de "segunda geração" e é típico dos circuitos oferecidos pelos principais fabricantes de TTL em suas linhas de velocidade média.

Examinemos a operação do circuito na Fig. 5.17, admitindo que a entrada *Y* está em +3,5 V e *X* está em 0 V e, portanto, T_1 tem sua junção base-emissor conectada a *Y* polarizada inversamente e que a corrente total proveniente de R_1 flui através da junção base-emissor conectada a *X*. Neste instante, (1) T_2 estará cortado; (2) T_4 estará cortado porque na ausência da corrente através de R_4 sua base estará essencialmente em terra; (3) T_3 e T_5 serão postos em condução pela corrente que flui através de R_2. Os transistores T_5 e T_3 estão conectados numa *configuração Darlington* que produz grande ganho de corrente. Em conseqüência, será necessária uma pequena corrente através de R_2 para levar T_3 a conduzir, e a saída estará em +5 V menos as quedas entre base-emissor de T_5 e T_3 e, portanto, em cerca de +4 V.

Consideremos que a tensão de entrada em *X* vai lentamente aumentando positiva. Em conseqüência, o coletor de T_1 se torna positivo e T_2 começa a conduzir. Quando o emissor de T_2 se torna cada vez mais positivo, T_4 começa a conduzir e, quando o coletor de T_2 se torna cada vez mais negativo, a combinação de T_5 a T_3 começa a cortar. Finalmente, (1) T_2 saturará; (2) T_4 estará saturado; e, portanto, cerca de +0,5 V ou menos.

O circuito na Fig. 5.17 terá cerca de 7 ns de retardos para entrar em condução e cerca de 11 ns de retardos para entrar em corte e uma dissipação de potência de cerca de 10 mW.

A razão para as altas velocidades reside em a construção do TTL ser toda a transistores e no fato de que o estágio final excita corrente em ambas as direções. O inversor padrão a transistor (ver Figs. 5.8 ou 5.15 dos exemplos em RTL e DTL) tem uma saída que é "excitada

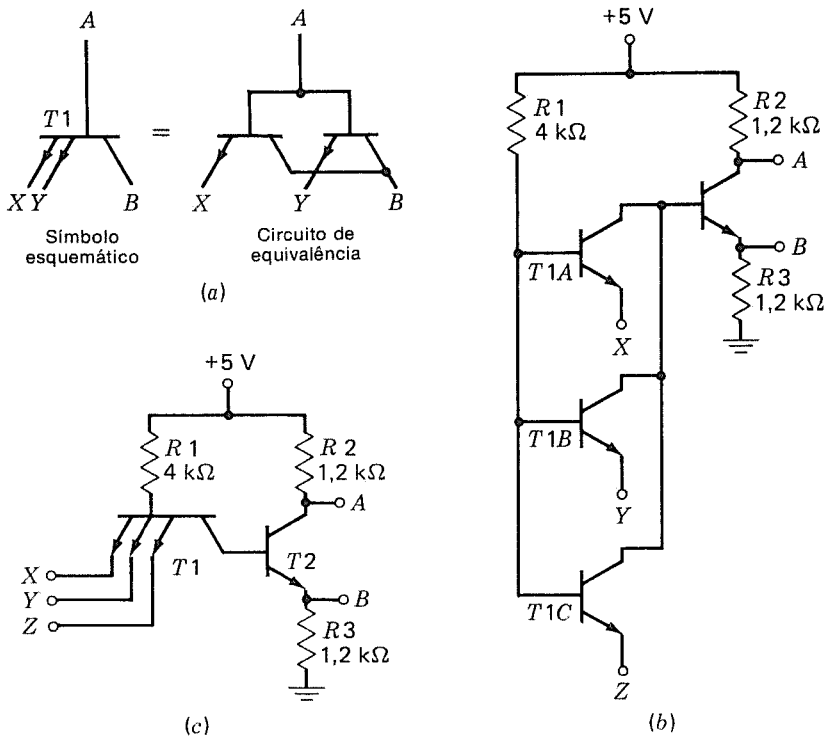


Fig. 5.16 Circuito porta TTL. *(a)* Símbolo esquemático e circuito equivalente; *(b)* versão discreta de porta *TTL; (c)* porta TTL com múltiplos emissores.

para baixo" quando o transistor de saída entra em condução. Quando o transistor é cortado, no entanto, o bordo crescente é formado pelo resistor na saída do circuito que fornece corrente a todas as capacitâncias espúrias neste circuito e a quaisquer circuitos conectados à saída. Portanto, o tempo de crescimento é exponencial. Com a saída a dois transistores da TTL (o circuito é chamado *circuito de pólo totem* ou circuito com repartição de alimentação), o bordo crescente é "excitado para cima" pelo transistor de cima que está conduzindo e pelo transistor de baixo que está cortado. Isto proporciona um bordo abrupto ou acentuado, e o bordo de descida é similarmente abrupto.

Em muitas aplicações aeroespaciais, militares e industriais é desejável ter uma dissipação de potência mais baixa. Se os valores dos resistores no projeto do circuito básico mostrado na Fig. 5.17 forem aumentados, o circuito consumirá potência muito menor, e os fabricantes de circuito oferecem circuitos de TTL a baixa potência. Valores típicos de resistores são: $R_1 = 40$ K, $R_2 = 20$ K, $R_3 = 500\ \Omega$, $R_4 = 12$ K e $R_5 = 5$ K. Naturalmente, haverá um decréscimo na velocidade, mas as portas de baixa potência são ainda capazes de produzir retardos de 33 ns com apenas 1 mW de dissipação de potência.

Há um problema famoso com circuitos TTL que pode ser ilustrado usando a Fig. 5.17. Quando os circuitos TTL são comutados, geram grandes "transitórios" em suas saídas, e as interligações devem ser cuidadosamente observadas para evitar oscilação ou mesmo danos ao circuito. Geralmente, são até colocados capacitores em paralelo com a fonte de alimentação de +5 V em relação à terra de cada receptáculo de circuito a fim de evitar transitórios na tensão da fonte de alimentação.

O problema se desenvolve porque é possível que T_3 e T_4 estejam simultaneamente conduzindo quando o circuito for comutado. Por exemplo, se a saída estiver comutando desde um nível 0 até um nível 1, então T_4 deve cortar e T_3 deve conduzir. Mas se T_3 conduzir antes que T_4 esteja completamente cortado, então ambos os transistores estarão conduzindo si-

multaneamente, quase pondo em curto os +5 V para terra.

O problema do transitório da corrente no TTL tem importunado a família TTL e está em evidência no circuito da Fig. 5.17. Enquanto T_3 e T_4 podem ambos estar conduzindo durante a ativação e a desativação, o caso da desativação é usualmente pior, uma vez que o tempo de armazenamento de T_4 faz com que ambos os transistores estejam conduzindo durante um tempo maior. O transitório da corrente de ativação pode ser reduzido quando se aumenta a relação entre R_2 e R_4. Desta maneira, o coletor de T_2 alcança uma tensão mais baixa antes que T_4 comece a conduzir. No entanto, há uma troca envolvida, já que o aumento de R_2 diminui a excitação para levar T_4 a conduzir, reduzindo seu tempo de ativação. O valor aumentado de R_2 também resulta em diminuir a imunidade ao ruído. Até agora, não foi encontrada nenhuma resposta "ideal" ao transitório da corrente.

Embora os circuitos TTL padronizados sejam rápidos, há sempre um desejo de acelerar as linhas de circuito. A fim de aumentar ainda mais as velocidades da lógica TTL, deve ser considerado um problema básico. Quando um transistor estiver saturado e deve ser cortado, antes de o transistor começar a passar para o corte existe um retardo (causados pelos portadores minoritários) chamado *retardo de tempo de armazenamento*. Este retardo é mostrado na Fig. 5.18*(a)* onde um inversor a transistor é cortado a partir da saturação. Enquanto este retardo for da ordem de nanossegundos, é ainda significativo já que vários dos transistores do circuito TTL se tornam saturados em tempos diferentes.†

A fim de minorar este problema, os circuitos de TTL mais recentes usam um grampeador a diodo entre a base e o coletor. Este diodo é mostrado na Fig. 5.18*(b)*. O diodo usado, no entanto, não é um diodo convencional, mas sim um diodo especial (chamado diodo Schottky) formado pela junção de um metal com um semicondutor.

O diodo Schottky é mais rápido que os diodos convencionais porque os elétrons que cruzam a junção e entram no metal quando a corrente está fluindo não são distintos dos elétrons de condução do metal. Uma vez que estes elétrons são portadores majoritários, não há retardo associado à recombinação de portadores minoritários como nos diodos a semicondutor. Em conseqüência, os tempos de recuperação reversa para os diodos

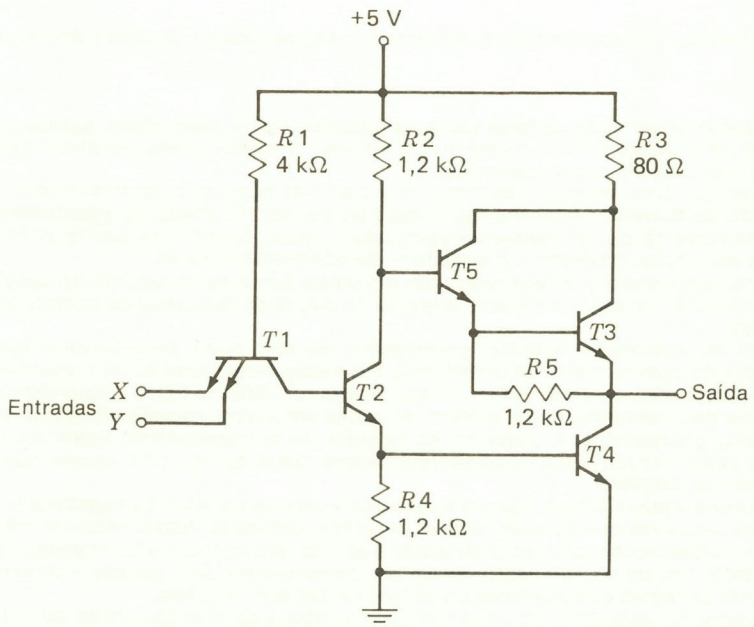


Fig. 5.17 Porta NE de TTL clássica.

†A dopagem do ouro é muitas vezes usada para reduzir os tempos de armazenamento dos portadores de minoria nos transistores de comutação.

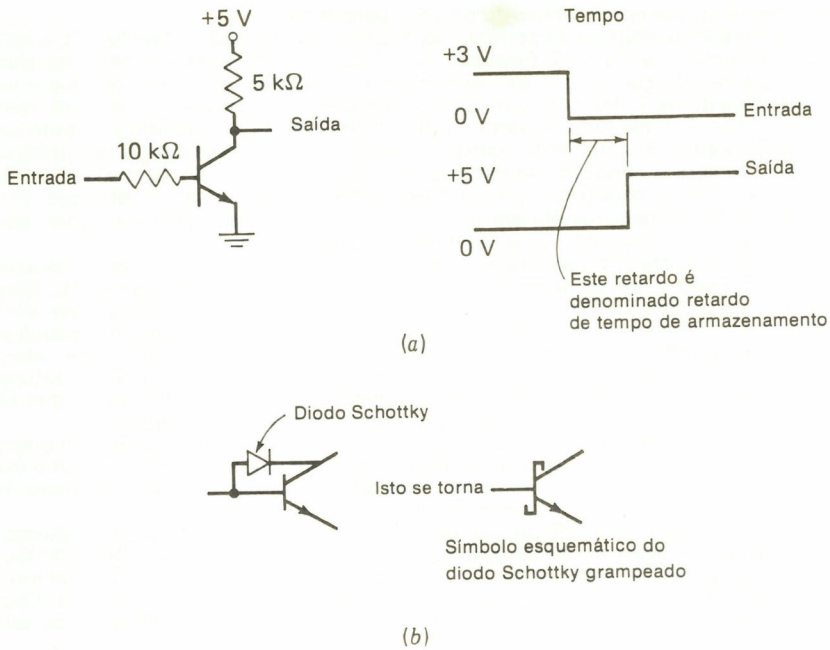


Fig. 5.18 Transistor Schottky grampeado. *(a)* Retardo de armazenamento; *(b)* diodos Schottky e transistor.

Schottky estão geralmente na faixa baixa de picossegundos. Além disso, devido à escolha dos materiais (silicietos de alumínio ou de platina), a queda direta no diodo Schottky é menor do que a de um diodo convencional.

O transistor com um diodo Schottky conectado da base ao coletor comutará mais rápido devido ao transistor ficar impossibilitado de saturar. O tempo de armazenamento de portadores minoritários, normalmente associado à operação do transistor quando ele está saindo da saturação, é evitado e o circuito pode operar mais rápido.

Quando um transistor é fabricado com um diodo Schottky conectado da base ao coletor, a combinação recebe um símbolo especial de diagrama esquemático conforme indica a Fig. 5.18*(b)*.

O uso de transistores Schottky grampeados resultou numa série de circuitos TTL de alta velocidade com retardos da ordem de 2 nanossegundos (tempos de crescimento e de queda são da ordem de 3 ns). A Fig. 5.19*(b)* mostra um circuito típico. Observemos que os transistores que normalmente se tornam saturados em algum instante são agora transistores Schottky grampeados, e, portanto, os retardos de armazenamento ficam reduzidos. A operação deste circuito é essencialmente a mesma que a da Fig. 5.17, exceto que os transistores não se saturam.

Os diodos ligados à terra nas entradas amortecem os transitórios negativos e os sinais crescentes negativos que possam ocorrer durante a oscilação. Estes circuitos, no entanto, devem ser cuidadosamente interconectados, e as ligações com circuito impresso ou de fios de mais de 8 pol. de comprimento devem ser consideradas com cautela e terminadas de acordo com as regras estabelecidas pelos fabricantes dos circuitos.

O Quadro 5.2 apresenta algumas das características de diversas linhas de TTL. Observemos que a velocidade está associada ao consumo de potência.

Os circuitos TTL são também encontrados em MSI. O decodificador de sete segmentos com MSI e os contadores de BCD no Cap. 4 usavam TTL. Uma porta de TTL simplificada é empregada internamente em tais circuitos de MSI, e esta porta é mostrada na Fig. 5.19*(a)*.

Um biestável de TTL é mostrado na Fig. 5.20, o Fairchild 9022. Este biestável opera como qualquer biestável tipo *JK*, exceto que ele fornece portas E nas entradas *J* e *K* e uma

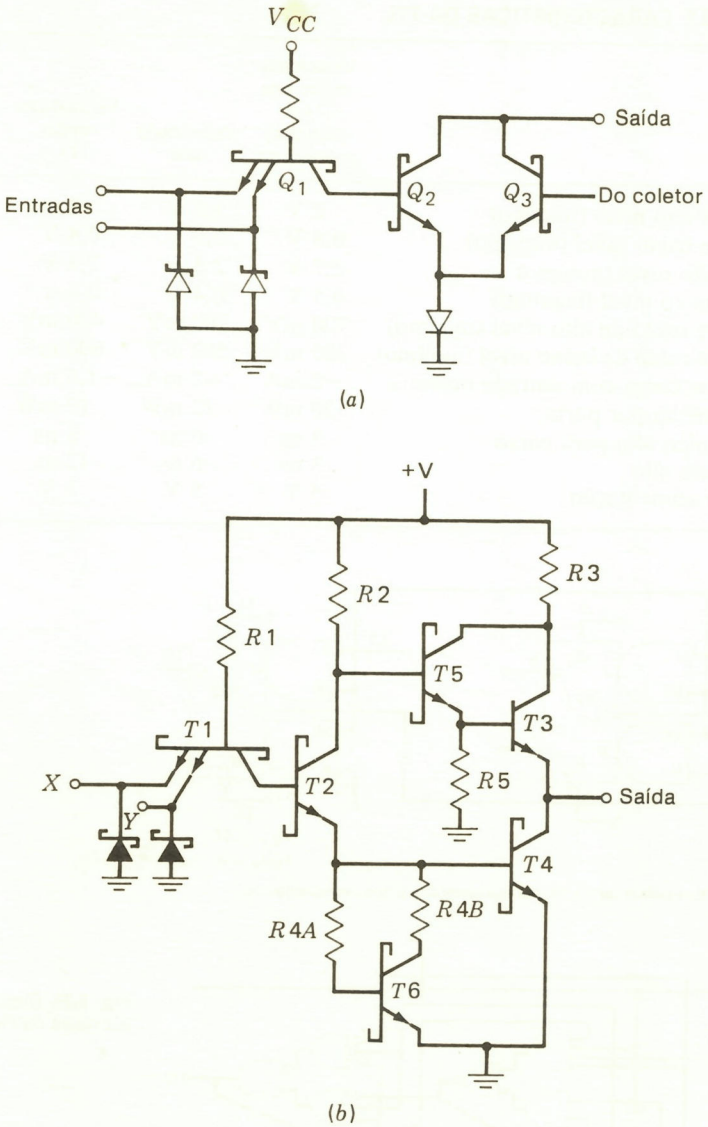
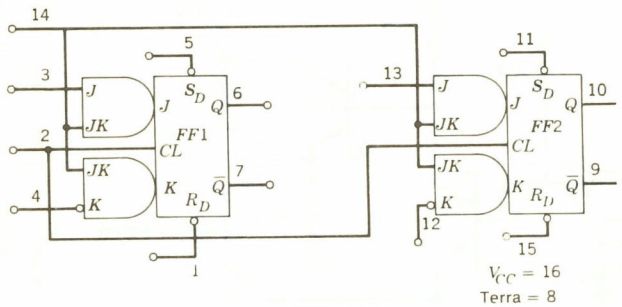


Fig. 5.19 *(a)* Circuito Schottky de MSI; *(b)* porta lógica de TTL Schottky grampeada.

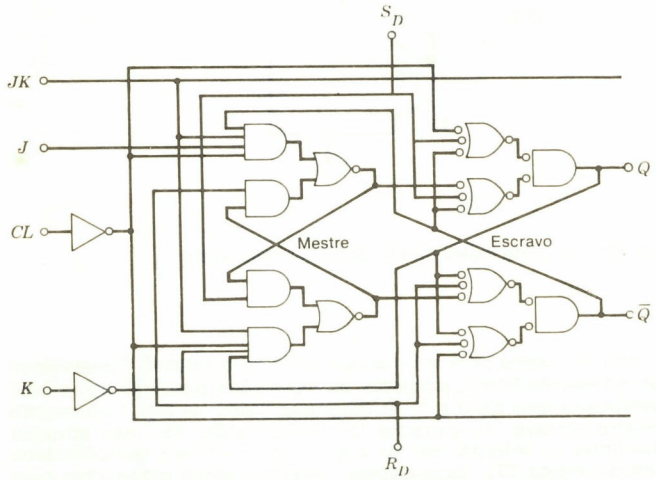
entrada *JK* ou "porta", que pode ser usada para inibir a operação do biestável. Observemos que a entrada de relógio não apresenta bolha; portanto, as transições positivas do relógio ativam o biestável (isto é, deslocamentos de 0 a 1). O diagrama-bloco na Fig. 5.20 mostra que o biestável é do tipo mestre-escravo. As entradas DC RESET e DC SET são ativadas ligando-se à terra a respectiva linha de entrada. Isso dá alguma idéia sobre a complexidade dos circuitos nas embalagens da lógica TTL. Dois destes biestáveis estão embalados num receptáculo de 14 pinos conforme indica a Fig. 4.12.

QUADRO 5.2 CARACTERÍSTICAS DA TTL

	Velocidade muito alta 74 S (Schottky grampeada)	Velocidade alta	Velocidade média 74	Velocidade baixa 74 L
Entrada de alto nível (mínimo)	2 V	1,8 V	2 V	2 V
Entrada de baixo nível (máximo)	0,8 V	0,9 V	0,8 V	0,7 V
Saída de alto nível (mínimo)	2,7 V	2,5 V	2,4 V	2,4 V
Saída de baixo nível (máximo)	0,5 V	0,4 V	0,4 V	0,3 V
Margem de ruído de alto nível (mínimo)	700 mV	700 mV	400 mV	400 mV
Margem de ruído de baixo nível (mínimo)	300 mV	500 mV	400 mV	400 mV
Corrente de carga com entrada máxima	−2 mA	−2 mA	−1,6 mA	−1,8 mA
Potência média por porta	20 mW	22 mW	10 mW	1 mW
Retardo típico alto para baixo	3 ns	6 ns	8 ns	31 ns
baixo para alto	3 ns	6 ns	12 ns	35 ns
Tensão de alimentação	5 V	5 V	5 V	5 V



Os números de 1 a 14 são números dos pinos no receptáculo.



O acima é repetido duas vezes num único receptáculo.

Fig. 5.20 Biestável de TTL. (Cortesia da Fairchild Corp.)

5.13 LÓGICA ACOPLADA PELO EMISSOR

A lógica acoplada pelo emissor (ECL) tem diversos outros nomes, a saber: *lógica de modo corrente (CML), lógica de orientação de corrente* e *lógica não-saturada.* O último termo constitui a chave deste tipo de circuito: quando os transistores são operados numa condição de saturação, eles entram em corte lentamente devido ao retardo causado por uma carga armazenada na região do coletor e da base.† Este retardo no "tempo de corte" pode ser eliminado operando-se os transistores somente em qualquer das regiões ativa ou de corte. Conforme veremos, nestes circuitos a corrente é "dirigida" em vez de haver tensões ou níveis distribuídos.

A linha de lógica ECL é a mais rápida comumente encontrada. Os fabricantes de ECL têm inúmeros circuitos básicos cada um com diferentes características e inconvenientes. A ECL não se mostrou tão popular quanto a TTL, principalmente porque ela é mais cara, difícil de arrefecer e mais difícil de interconectar, e é considerada como tendo menor imunidade ao ruído (isto é discutível). Além disso, a ECL pode ser mais rápida do que é necessário em muitas aplicações. Por outro lado, inúmeros computadores "super-rápidos" usam a ECL assim como vários dos computadores especiais das mais altas velocidades.

A configuração básica da ECL pode ser mais bem descrita pelo exame de um inversor particular. A Fig. 5.21 mostra um inversor em ECL com uma entrada X. Os níveis lógicos neste sistema são como segue: Binário 0 é representado por $-1{,}55$ V, e binário 1 é representado por $-0{,}75$ V. Observemos que esta é "lógica positiva", uma vez que, embora ambos os níveis sejam negativos, o nível mais positivo, $-0{,}75$, é o binário 1. Além disso, observemos a pequena diferença de sinal entre 0 e 1.

A operação do circuito é baseada num *amplificador diferencial* que consiste de T_4 e T_3. Quando a entrada em T_3 estiver em $-1{,}55$ V, T_3 estará cortado e a corrente fluirá através de R_3 e R_2. O cálculo indicará uma queda de cerca de 0,8 V em R_2; assim, admitindo uma queda entre base-e-emissor de T_1 igual a 0,75 V, a saída X estará em $-1{,}55$ V.

Uma vez que T_2 é cortado pela entrada de $-1{,}55$ V, fluirá uma corrente muito pequena através de R_1, e a saída $\bar{X}$ estará na queda de tensão entre base-e-emissor de T_2; portanto, a saída estará em $-0{,}75$ V.

Um exame mostrará que, se a entrada estiver em $-0{,}75$ V, o transistor T_3 estará conduzindo, T_4 estará cortado, a saída X estará em $-0{,}75$ V e a saída $\bar{X}$ em $-1{,}55$ V. (A chave para

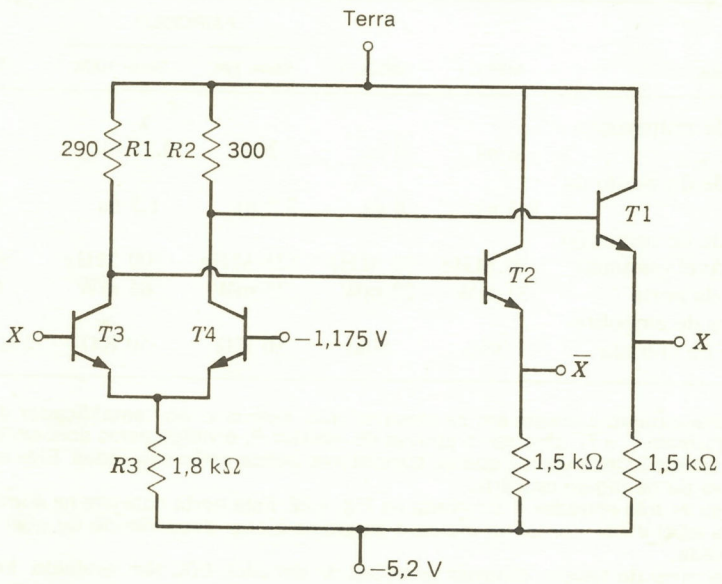


Fig. 5.21 Circuito básico de porta ECL. (Cortesia de Motorola Corp.; Motorola tem o nome comercial MECL para a ECL da Motorola.)

†Isto foi chamado retardo de armazenamento na seção precedente.

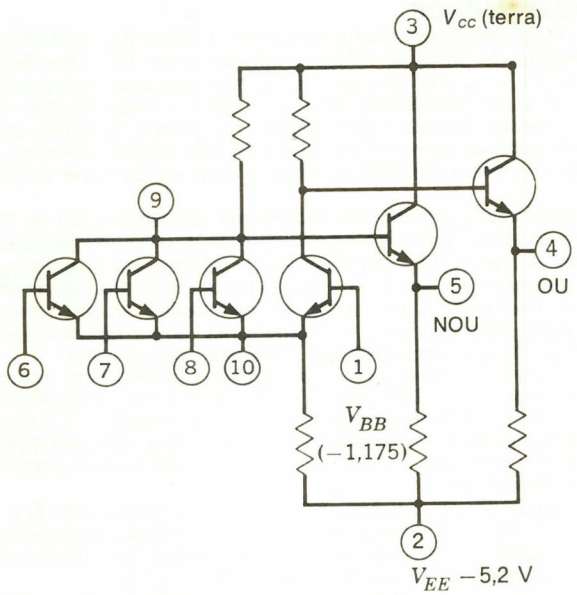


TABELA DA VERDADE

Entradas			Saídas	
8	7	6	5	4
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

Fig. 5.22 Porta ECL de três entradas.

QUADRO 5.3 CARACTERÍSTICAS GERAIS DE CIRCUITOS ECL

			FAIRCHILD		
Característica	MECL I	MECL II	Série 10K	Série 100K	MECL III
Retardo de propagação de porta	8 ns	4 ns	2 ns	0,75 ns	1 ns
Velocidade do bordo da porta	8,5 ns	4 ns	3,5 ns	1,5 ns	1 ns
Velocidade de oscilação do biestável (mínima)	40 MHz	70 MHz	125 MHz	500 MHz	500 MHz
Potência da porta	31 mW	22 mW	25 mW	65 mW	60 mW
Resistores de empobrecimento de entrada	Não	Não	50 KΩ	50 KΩ	2 KΩ, 50 KΩ

analisar este circuito consiste em se observar que, num circuito "amplificador diferencial" tal como formam T_3 a T_4, a corrente através do resistor R_3 dividida pelos dois emissores será quase constante.) Observemos que os transistores nunca estão saturados. Eles estão ou na região ativa ou na região de corte.

A porta de três entradas é mostrada na Fig. 5.22. Esta porta consiste na combinação de uma porta NOU e OU, conforme indica o diagrama-bloco, dependendo de qual ligação de saída é usada.

No decorrer do tempo, diversas gerações de circuitos ECL têm evoluído. Em geral, os circuitos se tornaram mais rápidos e requerem mais potência em cada geração. Maior número de facilidades e lógica mais complicada por pastilha são também encontrados nas linhas novas, inclusive pastilhas de MSI. Observe que os circuitos na Fig. 5.21 requerem três tensões: terra, −1,175 V e −5,2 V. Os circuitos posteriores fornecem um circuito em cada pastilha para gerar a tensão intermediária (neste caso, −1,175 V), de modo que é necessária

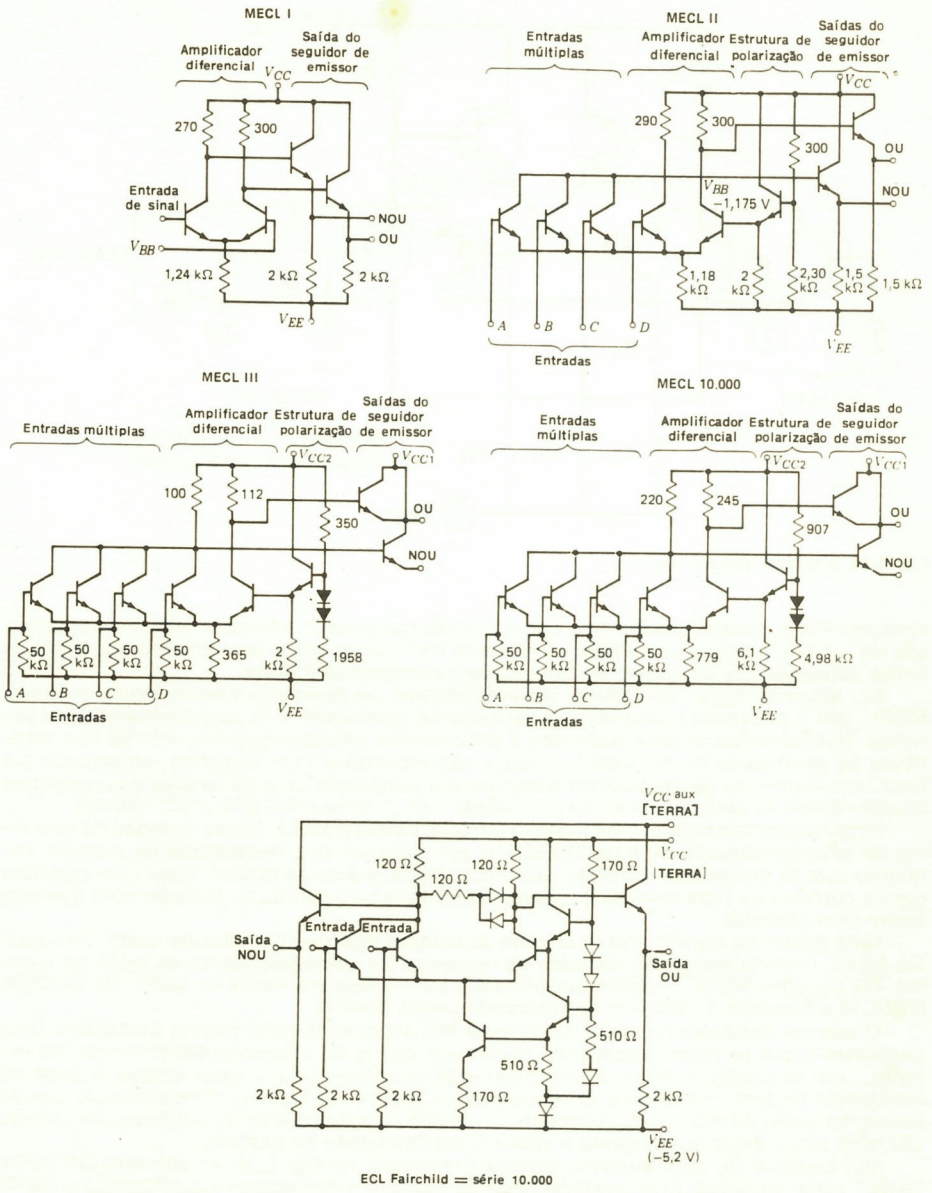


Fig. 5.23 Diversas gerações de circuitos ECL.

apenas uma única fonte de alimentação.

A Fig. 5.23 mostra as quatro gerações através das quais a Motorola tem atravessado com sua lógica ECL que é chamada ECL ou MECL da Motorola, e também a série da Fairchild n.º 10.000. O Quadro 5.3 esboça algumas das características destes circuitos, e a velocidade *versus* potência e as características gerais de ruído podem ser deduzidas a partir da tabela. Da Fig. 5.23, é evidente que variações foram empregadas quando a tecnologia

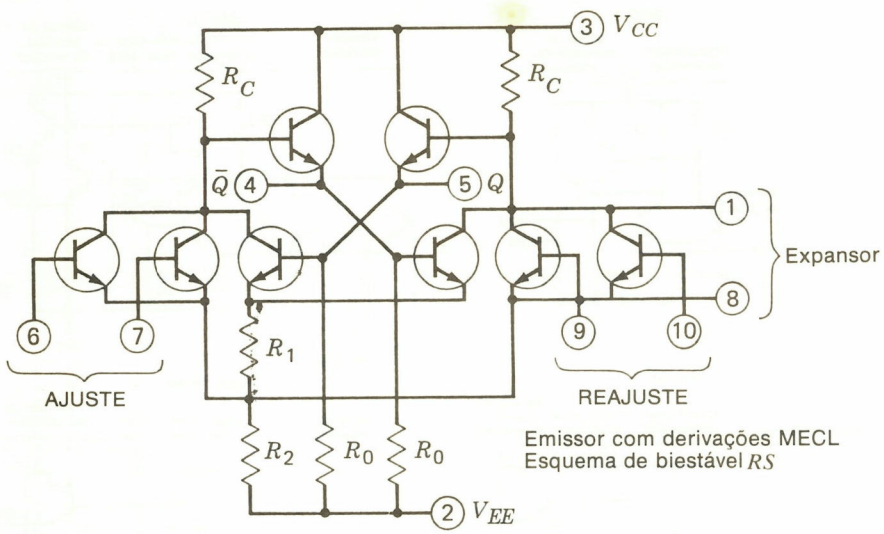


Fig. 5.24 Biestável tipo *RS* de ECL.

avançou. Por exemplo, observemos que o circuito que produz a terceira tensão de polarização de −1,175 V para a Linha MECL I não está incluído na pastilha, enquanto que todas as linhas subseqüentes possuem isto como uma característica interna.

Em segundo lugar, os valores correspondentes de resistores diferem entre as linhas MECL. Isto é necessário para obter a variação da velocidade e os aprimoramentos da potência. Naturalmente, a velocidade não é determinada unicamente pelos valores dos resistores; as geometrias do transistor, embora não mostradas num esquema, constituem um fator importante. As geometrias do transistor em conjunção com os valores dos resistores proporcionam as características de velocidade e de potência nas diferentes famílias.

Observemos também que as portas Fairchild 10.000 e MECL III são dotadas de resistores de empobrecimento de base (50.000 Ω) em cada um dos transistores de entrada, enquanto que as outras duas famílias não o são. Estes resistores proporcionam um caminho para a corrente de fuga da base em bases de entrada não-utilizadas, fazendo com que elas sejam bem cortadas.

Uma diferença significativa final entre as famílias está nos circuitos de saída. Os circuitos MECL I normalmente são dotados de resistores de empobrecimento de saída na pastilha. Os circuitos MECL II podem ser obtidos com ou sem resistores de saída. Os circuitos MECL III e Fairchild 10.000 têm normalmente saídas abertas.

O uso de resistores de saída na pastilha tem tanto vantagens quanto limitações. Uma vantagem é que se torna necessário um número menor de componentes externos. No entanto, com as saídas abertas, o projetista pode escolher tanto o valor quanto o local da resistência da terminação para satisfazer aos requisitos do sistema. Finalmente, o uso de resistores externos reduz o aquecimento na pastilha e a dissipação de potência, permitindo LSI mais complexa e aumentando a vida e a confiabilidade da pastilha.

Um biestável do tipo *RS* neste sistema é mostrado na Fig. 5.24. As entradas DC SET e RESET estão incluídas. Este biestável pode ser usado em contadores em velocidades até 30 milhões de contagens por segundo. Biestáveis mais complicados operam em velocidades até mais altas.

5.14 CIRCUITOS A SEMICONDUTOR METAL-ÓXIDO

Os circuitos até agora descritos são todos denominados circuitos *bipolares* e usam transistores "convencionais". Para integração em larga escala (LSI), muito freqüentemente é usado um outro tipo de transistor chamado *transistor de efeito de campo* (FET). Enquanto que as características destes FET não se mostraram desejáveis para algumas aplicações (por causa de sua lentidão, sensibilidade e falta de características de excitação), a facilidade de fabricação, as pequenas dimensões e a pequena dissipação de potência afastaram

os fatores negativos e conduziram aos FET construídos de MOS (semicondutor metal-óxido) como a tecnologia primária para uso em grandes sistemas. Os FET construídos de MOS são chamados MOSFET.

A fabricação de circuitos MOS usando FET envolve diferentes técnicas em diferentes companhias. A maioria dos circuitos MOSFET agora fabricados usa uma estrutura chamada *transistores de efeito de campo de porta isolada de modo realçado no canal p* ou *transistores de efeito de campo de porta isolada de modo realçado no canal n*. Trataremos principalmente destas estruturas.

A Fig. 5.25*(a)* mostra uma seção transversal de um FET do tipo de canal *p*. Conforme indicado na Fig. 5.25*(b)*, um substrato de material (silício) do tipo *n* é formado inicialmente, e duas regiões separadas do tipo *p* de baixa resistividade são difundidas neste substrato. Então, a superfície desta estrutura é coberta com uma camada de óxido isolante. São cortados orifícios dentro do óxido, e dois contatos metálicos são feitos nas duas regiões de material *p*, e um pedaço fino de metal chamado porta *(G)* é colocado sobre o que é chamado *canal*.

Em ausência de tensões aplicadas, a estrutura acima (ver Fig. 5.25) forma dois diodos com acoplamento antiparalelo, e se tentarmos forçar a corrente desde a fonte até o dreno, a junção *pn* alternativa seguida de uma junção *np* não permitirá o fluxo da corrente (em qualquer direção).

A porta é usada para causar e controlar o fluxo da corrente da seguinte maneira. Consideremos a fonte a ser ligada à terra e o dreno conectado a uma tensão negativa através de um resistor. A Fig. 5.26 mostra este caso [o símbolo esquemático do FET é mostrado na Fig. 5.25*(d)*]. A área de metal da porta, junto com a camada de óxido isolante e o canal semicondutor, formam um capacitor, com a porta de metal sendo a placa de cima e o substrato de material *n* sendo a placa de baixo. Tornando negativo o potencial da porta, produz-se uma carga positiva correspondente no substrato de semicondutor do tipo *n* ao longo do canal, conforme indicado na Fig. 5.25*(c)*. Colocando-se um potencial negativo suficiente na porta, a carga positiva induzida no canal faz com que finalmente esta seção de material se torne do tipo *p*, e a corrente começa a fluir da fonte para o dreno — e daí, a expressão

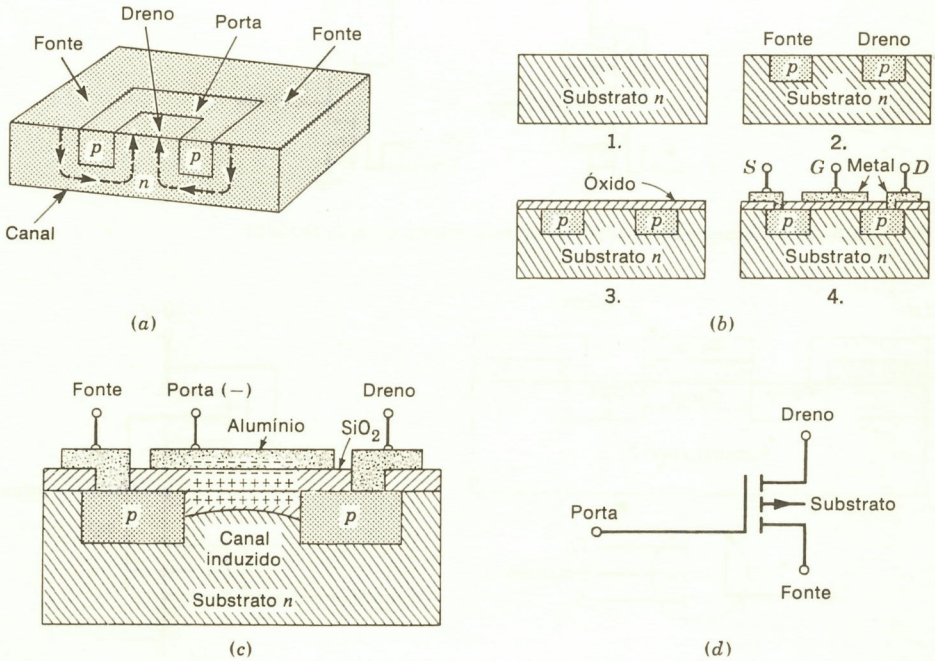


Fig. 5.25 Estrutura do MOSFET. *(a)* Configuração geral; *(b)* etapas da fabricação; *(c)* seção transversal de porta polarizada negativamente; *(d)* símbolo esquemático. (Cortesia da Motorola Corp.)

modo de realce de corrente.

Quanto mais negativa se torna a porta, mais do "tipo *p*" se torna o canal de semicondutor e flui mais corrente. Em conseqüência, este tipo de MOS é também chamado PMOS.

Como um circuito de comutação, um FET pode ser usado para formar um inversor. Com uma entrada 0, ou terra, a saída do circuito mostrado na Fig. 5.26 terá −10 V e, com uma entrada negativa de −7 V ou mais, a saída irá cerca de 0 V.

Duas observações devem ser feitas sobre este circuito. Primeiro, as tensões usadas são maiores do que as dos circuitos nas seções precedentes. Segundo, o circuito a FET tem uma alta impedância de entrada (talvez 10^{12} Ω) porque a entrada da porta é, efetivamente, um capacitor.

Um outro fato deve ser notado. Em vez de formar resistores reais para estes circuitos, um outro FET é usado, simplificando portanto a fabricação. Isto é mostrado na Fig. 5.26. As áreas da porta do resistor FET são controladas, de modo que o FET representa uma alta resistência (talvez 10 a 100 kΩ) quando a porta está no potencial do dreno. Alguns fabricantes mostram este FET usando o símbolo regular, e alguns mostram o símbolo de barra-mais-resistor, também visto na figura.

Quando um substrato do tipo *p* com dopagem *n* na fonte e no dreno for usado, como na Fig. 5.27, então o transistor de canal *n* é formado conforme indica a figura. O símbolo do diagrama esquemático do transistor de canal *n* e um circuito inversor são mostrados na Fig. 5.27*(a)* e *(b)*. Observemos que o circuito usa uma tensão positiva e se comporta similarmente a um circuito inversor a transistor *npn*. Este tipo de MOS é chamado NMOS.

Uma porta NOU pode ser formada em NMOS conforme indicado na Fig. 5.28*(a)*. Os níveis lógicos deste circuito são + 2 a 0 V para um 0 binário e > +8 V para um 1 binário. (Esta é lógica positiva.) Se qualquer uma das entradas *A*, *B* ou *C* for um 1, o FET correspondente conduzirá, fazendo com que a saída vá cerca de +2 V ou menos. Se todas as entradas estiverem em +2 V ou menos, os FET estarão todos cortados e a saída estará em +8 V (ou mais).

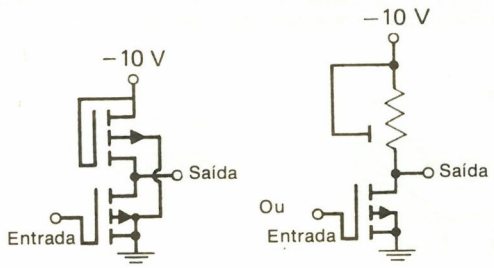


Fig. 5.26 Símbolos esquemáticos alternativos do elemento resistivo MOSFET.

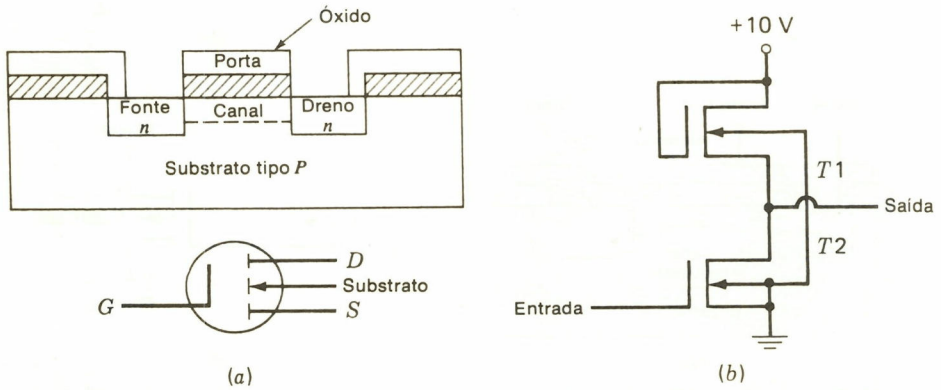
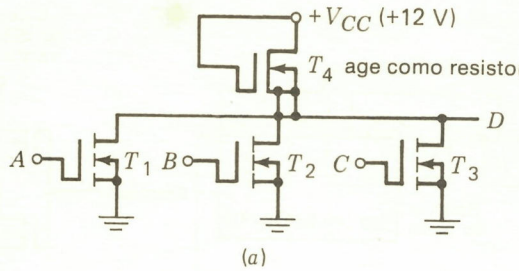
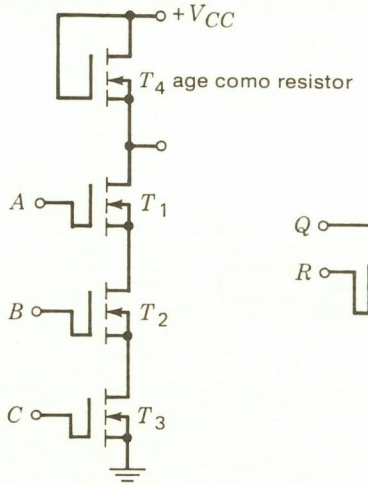


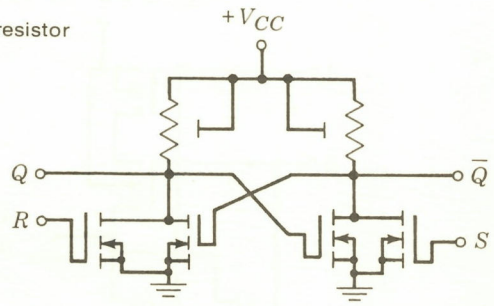
Fig. 5.27 *(a)* Transistor MOSFET de canal *n* (NMOS); *(b)* inversor de canal *n*.



(a)



(b)



(c)

Fig. 5.28 Circuito lógico a FET. *(a)* Porta OU de três entradas; *(b)* porta NE de três entradas; *(c)* biestável tipo *RS*.

Na Fig. 5.28 são mostrados diversos biestáveis e portas que usam NMOS. (PMOS é o mesmo, exceto para tensões V_{cc} negativas e lógica negativa.) As resistências elevadas usadas nestes circuitos significam baixa dissipação de potência. Isto, combinando com as pequenas áreas necessárias para fabricar um FET, torna possível fabricar grande número de circuitos numa única pequena pastilha. Comumente, um ou dois registros de deslocamento de 1.024 bits são incluídos num único receptáculo com 14 fios terminais. Os sistemas lógicos com milhares de portas e biestáveis são também fabricados em receptáculos padronizados, e estes são muitas vezes usados em memórias de CI que serão estudadas no Cap. 7, e em pastilhas de microprocessadores, conforme se irá descrever no Cap. 10.

5.15 CIRCUITOS LÓGICOS DE CMOS

Uma série de circuitos que usa transistores MOSFET chamada MOS complementar (CMOS) foi originalmente desenvolvida para as indústrias oceanográficas e aeroespaciais. Estes circuitos têm um consumo de potência muito baixo e considerável resistência ao ruído. No entanto, estes circuitos são lentos em comparação às linhas lógicas de alta velocidade, mas grande número de circuitos pode ser colocado numa única pastilha, a tensão da fonte de alimentação pode variar sobre uma larga faixa e os circuitos são relativamente econômicos de fabricar. Os circuitos CMOS mais novos têm-se tornado relativamente rápidos e são

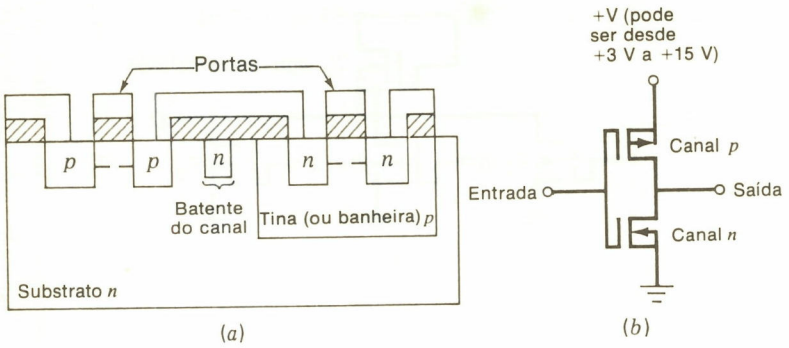


Fig. 5.29 Inversor a CMOS. *(a)* Elementos de CMOS; *(b)* circuito que é o inversor a CMOS básico.

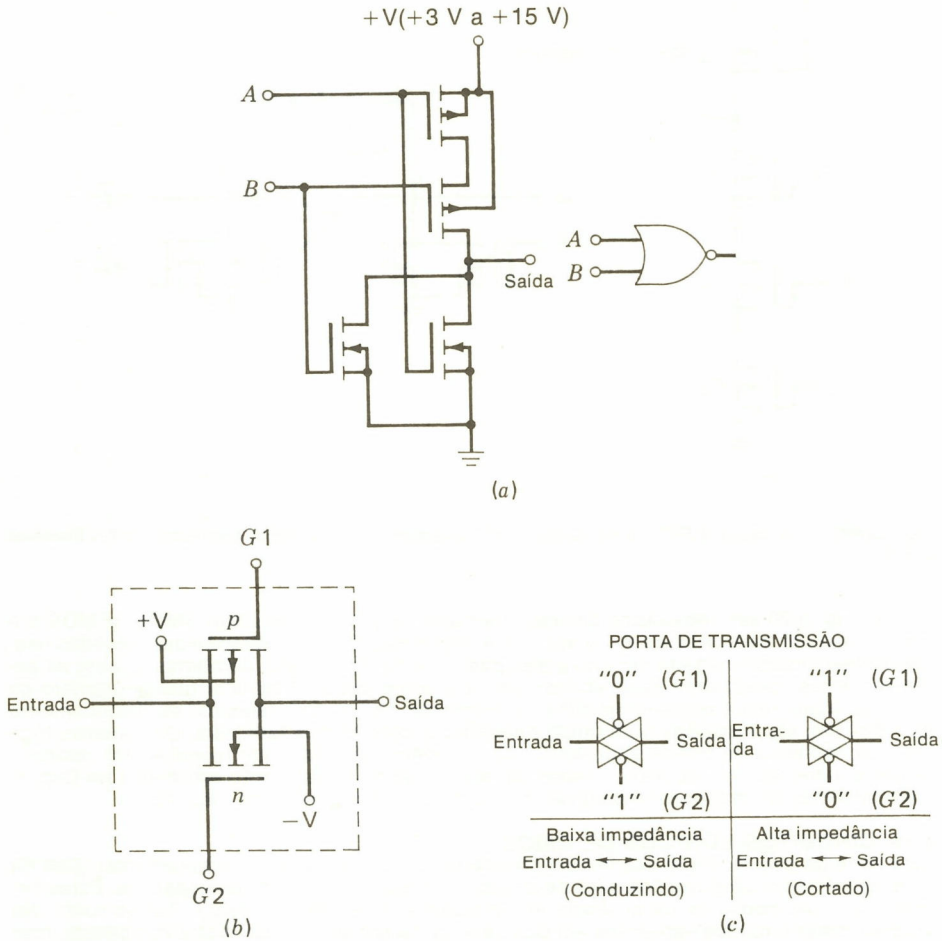


Fig. 5.30 Porta NOU a CMOS e porta de transmissão. *(a)* Porta NOU em que um par adicional de transistores é necessário em cada perna de porta acrescentada; *(b)* esquema da porta de transmissão; *(c)* símbolo esquemático.

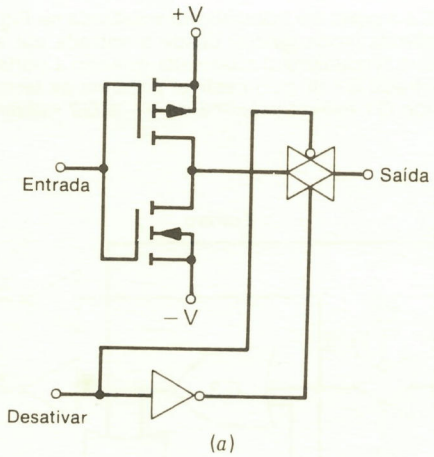
amplamente usados em muitas aplicações desde relógios e calculadoras eletrônicos até microprocessadores.

Os circuitos CMOS são fabricados como indicado na Fig. 5.29*(a)*, que mostra que ambos os transistores de canal *n* e de canal *p* podem ser fabricados sobre o mesmo substrato.

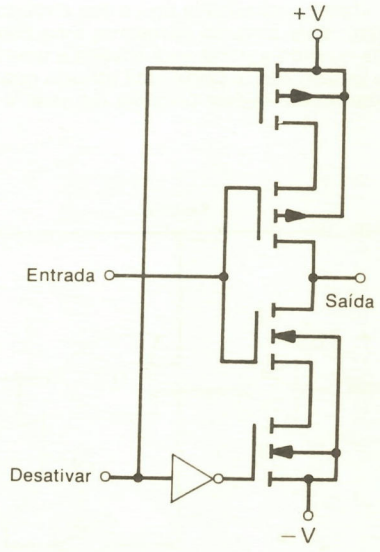
A mais simples forma do circuito integrado CMOS consiste de um transistor MOS do canal *n* e de um canal *p*, com ambos os contatos da porta unidos para formar a entrada e ambos os contatos do dreno unidos para formar a saída. Este circuito constitui o inversor CMOS básico [Fig. 5.29*(b)*]. Quando a tensão na entrada estiver próxima do nível da terra, a tensão porta/fonte do transistor do canal *p* se aproxima do valor da tensão de alimentação, +V, e o canal *p* está ativado. Um caminho de baixa resistência é criado entre a tensão de alimentação e a saída, enquanto que um caminho de alta resistência está entre a saída e a terra, devido ao transistor do canal *n* estar cortado. A tensão de saída se aproximará da tensão de alimentação V+. Quando a tensão de entrada estiver próxima de V+, o canal *p* fica desativado e o canal *n* fica ativado, fazendo com que a tensão de saída se aproxime da terra.

Observemos que, em qualquer estado, o consumo de potência do circuito é extremamente baixo porque um transistor está sempre cortado e porque os transistores de canal *n* e de canal *p* apresentam resistência muito elevada quando cortados, permitindo que uma corrente de fuga muito baixa passe pelo transistor que está na condição de corte.

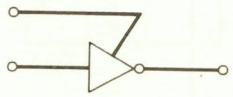
Quando forem usadas as tecnologias de porta de silício e de metal, são fornecidos batentes protetores do canal para minimizar a corrente de fuga entre os transistores separados conforme mostrado na Fig. 5.29*(a)*. Todos os dispositivos de canal *p* devem estar



(a)



(b)



ENTRADA	DESATIVAR	SAÍDA
1	0	0
0	0	1
X	1	Alta impedância

X = Estado de *Don't-care* (saída não-especificada)

(C)

Fig. 5.31 Circuito CMOS de três estados.

QUADRO 5.4 CARACTERÍSTICAS DA LINHA DE CIRCUITO

	TTL padrão 74	TTL de baixa potência	DTL	TTL Schottky gram-peado	Fonte de 5 V de CMOS	Fonte de 10 V de CMOS	ECL
Potência quiescente	10 mW	1 mW	8,5 mW	20 mW	10 nW	10 nW	60 mW
Retardo de propagação	7 ns	30 ns	30 ns	3 ns	15 ns	10 ns	0,5 ns
Freqüência de oscilação do biestável	35 MHz	4 MHz	5 MHz	80 MHz	5 MHz	10 MHz	500 MHz
Imunidade a ruído	1 V	1 V	1 V	0,8 V	2 V	4 V	0,5 V
Fator de carga de saída	10	10	8	10	50	50	5

circundados por um batente de canal *n* contínuo, que também pode atuar como um caminho de condução para a fonte de alimentação externa em locais apropriados. Similarmente, os batentes de canal *p* circundam todos os dispositivos de canal *n* e proporcionam um caminho de condução entre os dispositivos de canal *n* que estão eletricamente conectados ao mais baixo potencial e ao contato de terra externo.

Uma porta NOU de duas entradas pode ser construída conforme indicado na Fig. 5.30. Cada entrada adicional requer um par adicional de transistores MOS de canal *n* e de canal *p*.

Uma característica única dos circuitos CMOS é a *porta de transmissão* mostrada na Fig. 5.30*(b)*. Este circuito apresenta uma baixa resistência (impedância) desde a entrada até a saída quando a porta está ativada e uma resistência (impedância) muito alta quando a porta está desativada. A porta está ativada quando a entrada G1 da porta estiver próxima da terra e quando G2 estiver próxima do nível V+. Quando G1 estiver próxima de V+ e G2 estiver

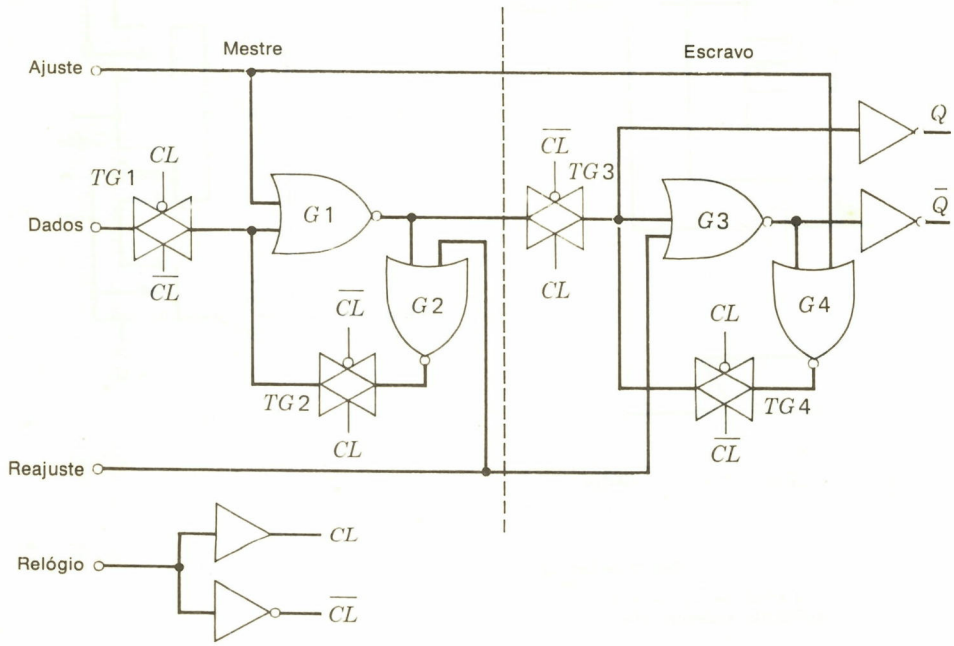


Fig. 5.32 Biestável do tipo *D* em CMOS.

próxima da terra, a porta está desativada e tem uma resistência desde a entrada até a saída que é geralmente maior que $10^9\ \Omega$. A Fig. 5.28*(b)* também mostra o símbolo de diagrama lógico para a porta de transmissão.

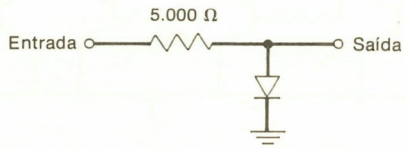
As portas de transmissão costumam ser amplamente usadas nos projetos de CMOS. Freqüentemente é usado um circuito de três estados, e este será descrito em capítulos posteriores. A Fig. 5.31 mostra este circuito e seus três estados.

A Fig. 5.32 mostra um biestável do tipo *D* na tecnologia de CMOS que usa portas de transmissão para simplificar o projeto. Observemos a simplicidade desta configuração *versus* circuitos TTL e DTL mostrados inicialmente.

O Quadro 5.4 dá alguns detalhes da operação de CMOS *versus* outras linhas de circuito. Observemos o baixo consumo de potência (nanowatts quando não comutado), velocidades competitivas e proteção contra ruído.

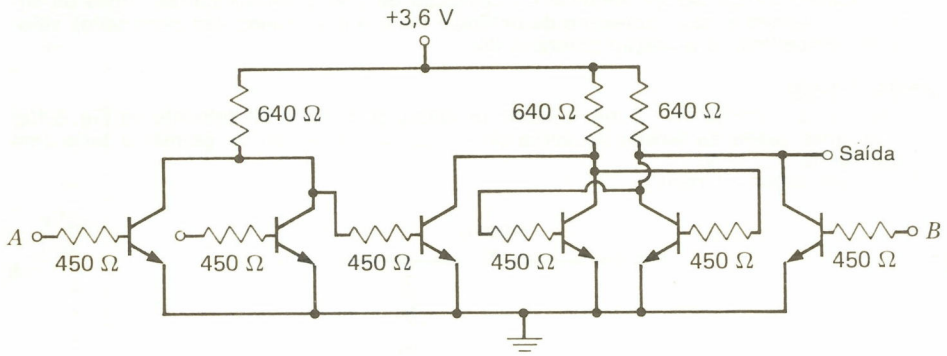
EXERCÍCIOS

1. Para uma corrente de +2 mA, o diodo de silício polarizado diretamente na Fig. 5.1*(c)* tem uma queda de tensão de cerca de _____ V. Um diodo de germânio teria uma queda de _____ V.
2. Consideremos o circuito

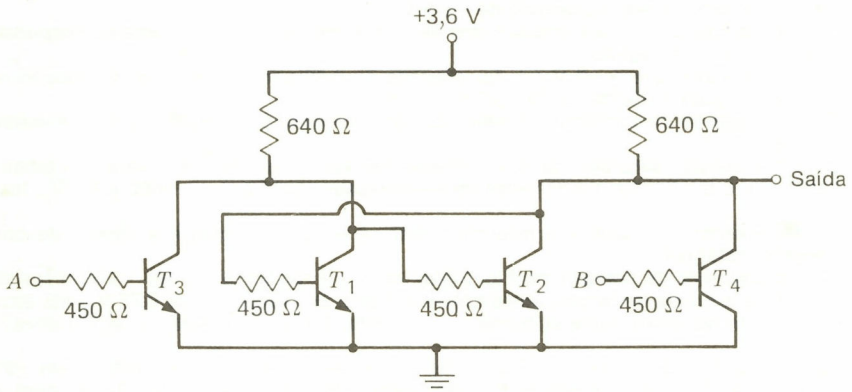


   Se tivermos uma tensão de entrada de +10 V, a saída estará em cerca de _____ volts, e aproximadamente _____ ampères fluirão através do resistor. (Resposta para ambos os diodos de germânio e de silício.)
3. Para que valor de corrente o diodo de silício, cujas curvas características estão na Fig. 5.1*(c)*, tem uma queda de tensão de +0,5 V?
4. Se, para o Exercício 2, a entrada fosse de −10 V, dar a tensão de saída correspondente e as correntes no resistor.
5. Se V_{ce} for mantido em +8 V e I_b for alterada de 0,85 para 1 mA, qual a variação resultante em I_c para o 2N706 na Fig. 5.3?
6. Qual é o ganho de corrente no transistor do Exercício 5 (a relação entre as variações em I_b e em I_c)?
7. Para o transistor saturado cujas características estão na Fig. 5.3*(c)*, qual é a tensão da base, que é também a queda entre base-e-emissor, quando I_b = 1.000 μA e V_{ce} for 300 mV?
8. Para o Exercício 7, qual é a diferença entre a tensão da base e a tensão de coletor naquele instante?
9. Para o seguidor de emissor na Fig. 5.5*(a)*, se o +V for 0 V e −V for −5 V, o resistor é 1.000 Ω, e a queda de tensão entre base-e-emissor do transistor for 0,75 V, uma entrada de −2 volts resultará numa saída de _____ volts. O transistor está na região ativa? Por quê?
10. Para o seguidor de emissor na Fig. 5.5*(a)*, se −V estiver em terra, +V estiver em +5 V, o resistor for 2 kΩ, e a queda entre base-e-emissor do transistor for 0,5 V, para uma entrada de +1 V dar a tensão de saída e a corrente através do resistor. O transistor está na região ativa? Por quê?
11. Se a entrada *X* na Fig. 5.6 estiver em +3,5 V e a entrada *Y* em +3 V, qual será o nível da tensão de saída em *Z*? Admitir um diodo de silício com uma queda direta de 0,6 V.
12. Se a entrada *X* no circuito da Fig. 5.6 estiver em +0,5 V e a entrada *Y* em 0,25 V, para um diodo de silício com uma queda direta de 0,6 V, qual será a tensão de saída em *Z*?
13. Desenhar o esquema de uma porta E a diodo com quatro entradas.
14. Se a entrada *X* na porta OU da Fig. 5.7 estiver em +2 V e a entrada *Y* em 0,5 V, qual será a tensão de saída? Admitir um diodo de silício com uma queda direta de 0,6 V.
15. Para o circuito descrito no exercício precedente, qual será o nível da saída se *X* estiver em 0,1 V e *Y* em 0,7 V?
16. Desenhar o esquema de uma porta OU a diodo com quatro entradas.
17. Desenhar o esquema de um circuito porta E/OU a diodo que realize a função de álgebra booleana $AB + CD$.

18. Analisar os níveis de saída para o circuito que foi projetado no exercício precedente com níveis de entrada de +3 V para uma lógica 1 e de +0,5 V para uma lógica 0.
19. Desenhar o diagrama-bloco e o esquema de um simples inversor RTL, como na Fig. 5.8, mas com sua saída conectada a quatro outros inversores.
20. Para o desenho que foi feito no Exercício 19, calcular a tensão de saída do primeiro inversor quando ele tiver uma saída 1.
21. Desenhar um diagrama-bloco do seguinte circuito. As entradas são *A*, *B* e *C* e cada uma transporta um sinal de 0 ou +2 V, representando 0 ou 1, respectivamente. Quais serão os níveis de saída se a *saída* não estiver conectada a nada?



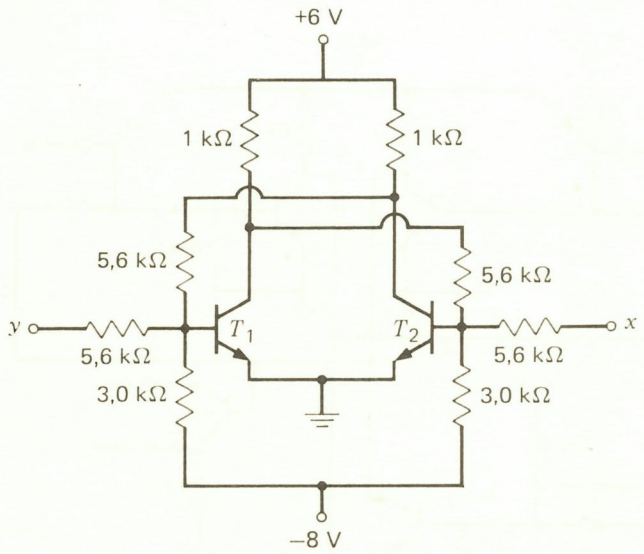
22. Admitir que as entradas *A* e *B* estão ambas no potencial de terra (0 V) e que o transistor T_2 está cortado. Se a entrada *B* for aumentada para +2 V e depois novamente abaixada para 0 V, descrever a operação do circuito.



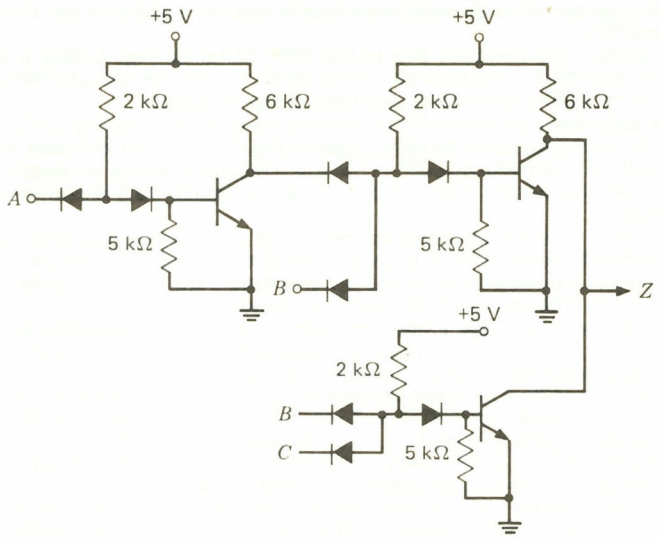
23. Discutir o problema da proteção contra ruído nos circuitos RTL e DTL.
24. No circuito TTL na Fig. 5.15*(c)*, se todas as entradas estiverem em +3 V, quais serão as tensões aproximadas no emissor e no coletor de T_2?
25. Que transistores no circuito da Fig. 5.17 estarão saturados quando ambas as entradas estiverem baixas?
26. Que transistores no circuito da Fig. 5.17 estarão saturados quando ambas as entradas estiverem altas?
27. Usando a informação obtida nos dois exercícios precedentes, discutir como o uso de transistores Schottky grampeados pode acelerar o circuito na Fig. 5.17.
28. Comparar as vantagens e os inconvenientes dos circuitos TTL no Quadro 5.2.
29. Tecer comentários sobre a simplicidade e o baixo consumo de potência do circuito na Fig. 5.19*(a)*.
30. A conexão da saída de uma porta DTL como na Fig. 5.15*(a)* a diversas entradas da porta

DTL tende a causar níveis de saída mais baixos quando a saída for um 1? Comparar isto com a porta RTL.

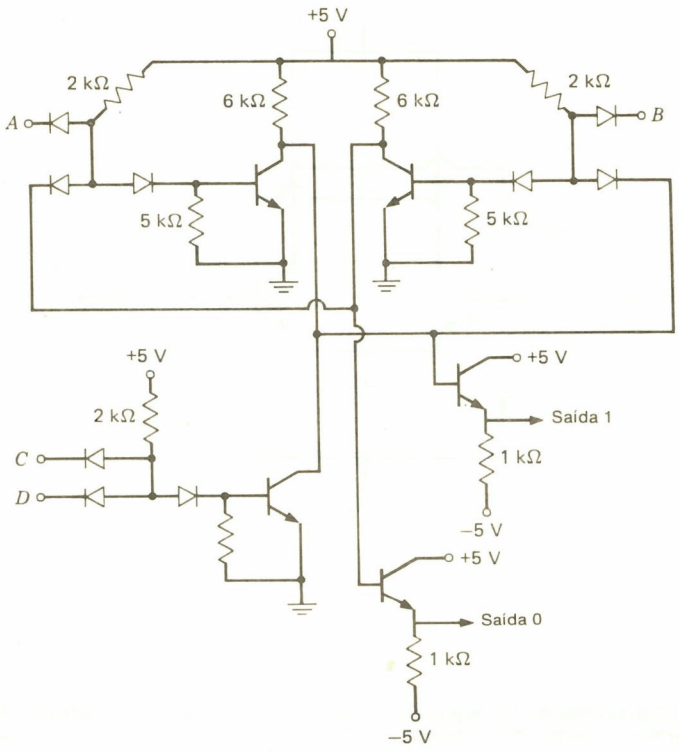
31. Se conectarmos três entradas das portas DTL como na Fig. 5.15*(a)* à saída de uma única porta, qual será a corrente de coletor de T_2 quando ele estiver conduzindo? Admitir que T_1 tenha ganho apreciável, sua base drena corrente 0.
32. Explicar a ação dos dois transistores de saída na porta TTL na Fig. 5.17.
33. Escrever os valores na saída de cada porta na Fig. 5.15*(c)* para entradas de $S_1 = 1$, $S_2 = 1$, $R_1 = 0$ e $R_2 = 1$. Admitir que $CL = 1$ e que o biestável esteja no estado 1.
34. Refazer o exercício acima, mas com $S_1 = 0$. A que estado irá o biestável quando CL for a 0? Explicar.
35. Se, na Fig. 5.21, T_4 entra em curto de modo que o emissor e o coletor estejam ambos permanentemente conectados à base, qual será a saída em X? Explicar.
36. Dado um único transistor *npn*, como o conectaria para fazer da porta MECL de três entradas na Fig. 5.22 uma porta de quatro entradas?
37. T_4, na Fig. 5.28*(a)* e *(b)*, age como um resistor. Analisar a operação de ambas essas portas se $A = 0$, $B = 1$ e $C = 1$. Fazer V_{cc} igual a 30 V, 25 V um 1 e 0 V um 0.
38. Desenhar diagramas-bloco para o biestável na Fig. 5.28*(c)*.
39. Na Fig. 5.28*(c)*, a saída 0 se torna positiva ou vai a 0 se $R = 10$ V e $S = 0$ V? Explicar.
40. Na Fig. 5.12, se o resistor de 640 Ω de T_1 até +3,5 V estiver aberto (ou desconectado), o biestável sempre estará no estado _____. Explicar sua resposta.
41. Se as entradas RESET e SET para o biestável de porta NOU na Fig. 5.13 estiverem ambas conectadas a +3,5 V, as saídas 0 e 1 estarão em que níveis?
42. Admitir que as entradas x e y no seguinte circuito estejam no potencial de terra e que o transistor T_1 esteja cortado. Explicar a operação do circuito se a entrada x for feita 5 V positivos e em seguida retornada à terra.



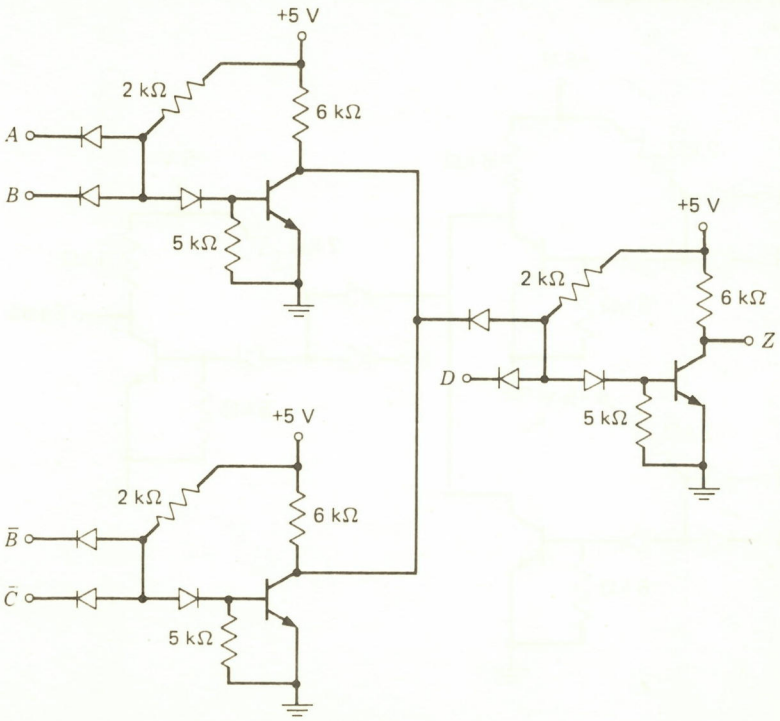
43. Escrever uma expressão de álgebra booleana relacionando as entradas A, B e C e seus complementos à saída Z no seguinte circuito.



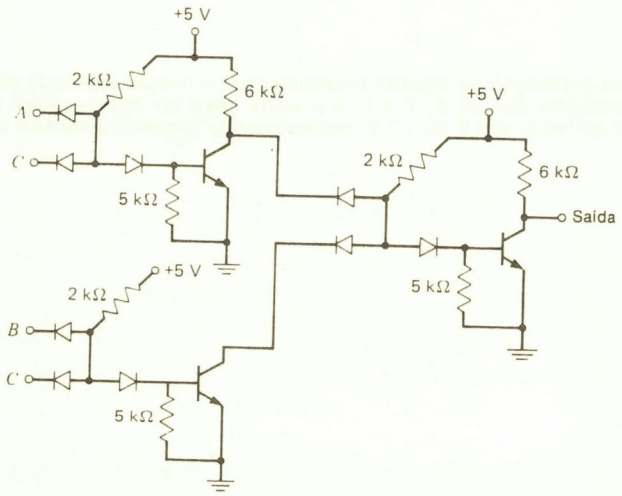
44. As entradas do seguinte circuito são *A*, *B*, *C*, *D* e *E*, cada uma das quais é um sinal de 0 ou +3 V, representando 0 e 1, respectivamente. Desenhar um diagrama-bloco para este circuito usando portas E, OU, NE, NOU, ou quaisquer portas a sua escolha.



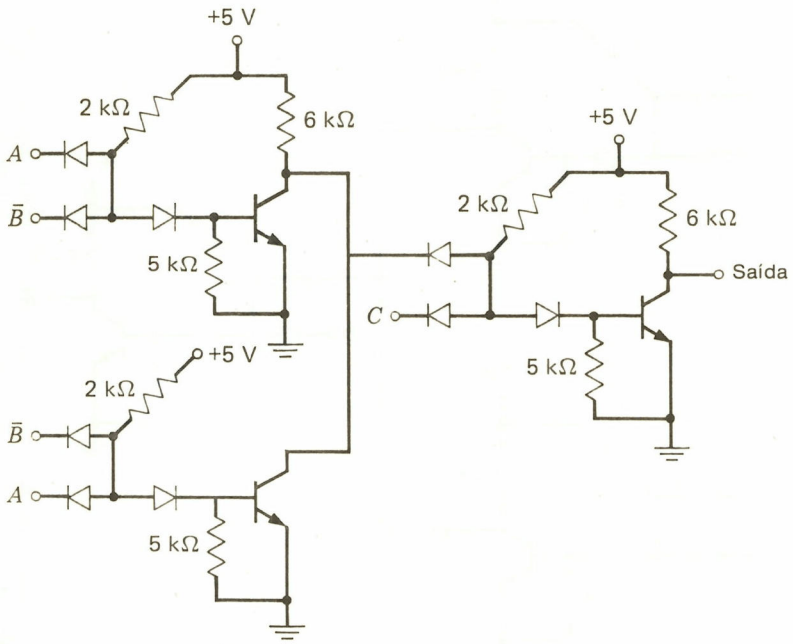
45. O seguinte circuito DTL tem entradas *A*, *B*, *C* e *D* e uma única saída *Z*. Desenhar um diagrama-bloco deste circuito usando portas E, OU, NE, NOU, inversores ou biestáveis.



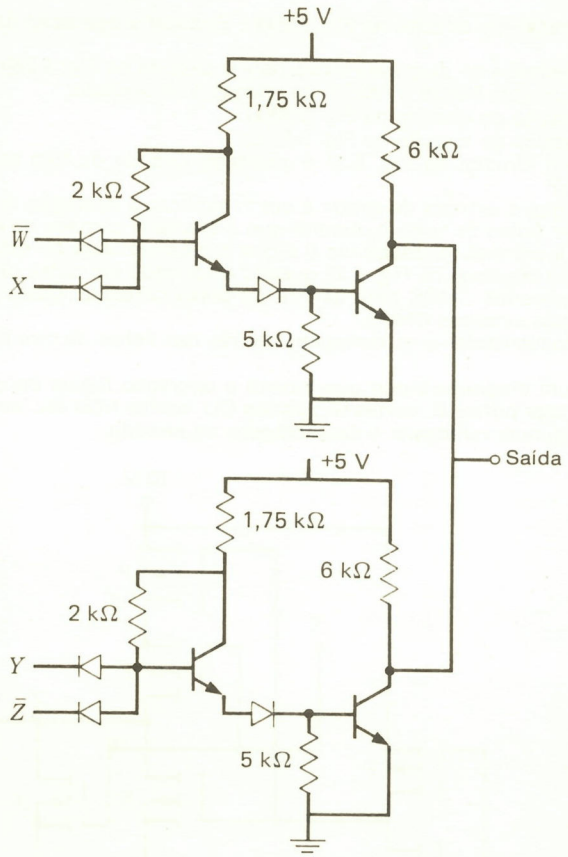
46. Desenhar um diagrama-bloco para o seguinte circuito usando quaisquer portas ou biestáveis a sua escolha. As entradas *A*, *B* e *C* são níveis de sinal de 0 ou de +3 V representando 0 e 1, respectivamente.



47. Desenhar um diagrama-bloco do seguinte circuito usando quaisquer portas ou biestáveis a sua escolha. As entradas *A*, *B* e *C* são níveis de sinal de 0 ou +3 V, representando 0 e 1, respectivamente.

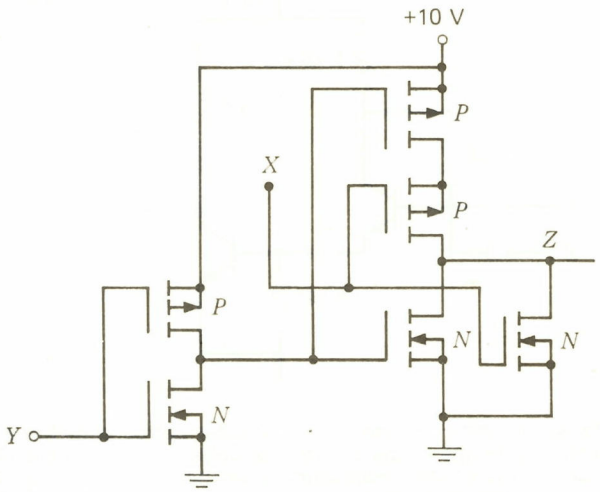


48. Escrever uma expressão de álgebra booleana para a função realizada pelo circuito seguinte. As entradas são *W*, *X*, *Y* e *Z*, e a saída deve ser expressa em termos dessas variáveis. As entradas são 0 ou +3 V, representando respectivamente 0 e 1 em binário.



49. A capacidade de um certo circuito TTL em excitar outros circuitos é freqüentemente expressa em termos do que se denomina uma *carga unitária*. Para o TTL da série 7400 da Fairchild, uma carga unitária representa 40 μA (para terra), quando o nível estiver no estado alto, e é a capacidade de aceitar 1,6 mA no estado baixo. Então, uma entrada para uma porta NE na série 7400 requer que uma carga unitária seja excitada baixa ou alta. Uma saída de porta NE na série 7400 pode excitar 20 cargas unitárias quando estiver alta e 10 cargas unitárias quando estiver baixa. Se uma porta NE TTL for conectada a cinco entradas em outras portas NE e a saída for alta, quantos ampères a saída pode fornecer para terra?
50. Usando os dados do exercício precedente, quantas entradas de porta podem ser conectadas em segurança a uma determinada saída de porta? (Isto é denominado *fator de carga de saída* da porta).
51. Comparar as razões de *fator de carga de saída* das portas TTL 7400 com as portas RTL na Fig. 4.15.
52. Por que a carga unitária é maior no nível baixo do que no nível alto? Examinar o circuito TTL básico para obter sua resposta.
53. Explicar por que as saídas TTL vão do nível baixo ao nível alto mais rapidamente do que as DTL quando deve ser excitado um fio longo ou uma outra carga capacitiva.
54. Quais são as vantagens dos circuitos MOS em relação aos circuitos bipolares convencionais?
55. Por que as portas CMOS requerem tão pouca potência quando elas não estão mudando de estados?
56. Desenhar um gráfico comparando velocidade e consumo de potência nos circuitos ECL da Motorola 10.000 e MECL III. Este gráfico indica alguma relação?

57. Explicar a operação do circuito na Fig. 5.28*(a)*. Explicar a operação do circuito na Fig. 5.28*(b)*.
58. Desenhar os símbolos do diagrama-bloco para o circuito na Fig. 5.28*(c)*.
59. Comparar os circuitos NMOS e PMOS em relação à velocidade.
60. Explicar a operação do circuito na Fig. 5.30*(a)*.
61. Explicar a operação do circuito na Fig. 5.30*(b)*.
62. Por que razão o circuito da Fig. 5.31 é chamado circuito de três estados? Qual é o terceiro estado?
63. Considerando que a entrada de *dados* é um 1, explicar a operação do circuito na Fig. 5.32 durante um pulso de relógio. Admitir que o biestável comece no estado 0.
64. Considerando a entrada de dados um 0 e que o biestável está no estado 1, mostrar o que acontece ao biestável na Fig. 5.32 quando é aplicado um pulso de relógio.
65. Comparar os circuitos CMOS com os TTL de baixa potência. Quais poderiam ser as desvantagens dos circuitos CMOS?
66. Discutir os inconvenientes e as vantagens de três das linhas de circuito constantes do Quadro 5.4.
67. *(a)* Desenhar um diagrama-bloco que mostre a operação lógica do circuito seguinte. Podem-se usar portas E, portas NE, portas OU, portas NOU etc. em seu diagrama.
    (b) Explicar algumas vantagens e desvantagens do circuito.



68. Calcular a potência dissipada no resistor R_2 quando o circuito ECL na Fig. 5.21 estiver em ambos de seus estados de saída.
69. Comparar as séries de circuitos MECL III e ECL 10.000 da Fairchild no que diz respeito à dissipação de potência. Estes circuitos são muito semelhantes?

6

UNIDADE ARITMÉTICA-LÓGICA

A unidade aritmética-lógica (UAL) é a seção do computador que efetua as operações aritméticas e lógicas com os dados processados pelo computador. Esta seção da máquina pode ser relativamente pequena, consistindo provavelmente de uma ou mais pastilhas de integração em larga escala (LSI), ou, nos grandes "trituradores de números" (computadores orientados para os problemas científicos), ela pode consistir de um conjunto considerável de componentes lógicos de alta velocidade. A despeito das variações no tamanho e na complexidade, as pequenas máquinas geralmente efetuam suas operações aritméticas e lógicas usando os mesmos princípios que as máquinas grandes. O que varia é a velocidade das portas lógicas e dos biestáveis usados; além disso, são usadas técnicas especiais para acelerar as operações e efetuar várias operações em paralelo.

Embora muitas funções possam ser executadas pela UAL das máquinas dos dias presentes, as operações aritméticas básicas — adição, subtração, multiplicação e divisão — continuam a ser operações de "meio de vida". Até a literatura dá evidência da natureza fundamental destas operações, pois quando uma nova máquina é descrita, os tempos necessários à adição e à multiplicação costumam sempre ser incluídos como características relevantes. De acordo com isto, este capítulo descreve de início os meios pelos quais um computador soma, subtrai, multiplica e divide. Outras operações básicas, tais como deslocamento, multiplicação lógica e adição lógica, são a seguir descritas.

Deve ser lembrado que a unidade de controle dirige a operação da UAL. O que a UAL faz é adicionar, subtrair, deslocar etc., quando ela é suprida com a seqüência correta de sinais de entrada. Ela é até o elemento de controle para fornecer estes sinais, pois constitui função das unidades de memória fornecer ao elemento aritmético a informação que deve ser usada. Será portanto admitido que as seções de controle e de memória da máquina são capazes de distribuir os sinais de controle corretos, e que os dados sobre os quais devem ser efetuadas as operações estão disponíveis. A função da UAL é portanto adicionar, subtrair ou efetuar qualquer operação que o elemento de controle dirija.

6.1 CONSTRUÇÃO DA UAL

A informação manipulada num computador é geralmente dividida em "palavras", cada uma delas consistindo de um número fixo de bits.† Por exemplo, as palavras manipuladas por uma certa máquina binária podem ter 32 bits de comprimento. Neste caso, a UAL teria de ser capaz de somar, subtrair etc. palavras de 32 bits de comprimento. Os operandos usados são então fornecidos pelo dispositivo de armazenamento do computador, e o elemento de controle dirige as operações que são efetuadas. Se tiver de ser feita uma adição, o primeiro e o segundo somandos serão fornecidos à UAL que deve adicionar os números e, em seguida, pelo menos temporariamente, armazenar o resultado (soma).

A fim de introduzir vários conceitos, consideremos a construção de uma UAL típica de computador. Os dispositivos de armazenamento consistirão de um conjunto de *registros* de biestáveis, cada um dos quais consiste de um ou mais biestáveis. O *comprimento* de cada registro é definido como a quantidade máxima de informação que o registro pode armaze-

†Alguns computadores também proporcionam a capacidade de manipular operandos de comprimento variável.

nar. Num registro binário, o comprimento do registro é igual ao número máximo de dígitos binários que podem ser armazenados; e num registro de decimal-codificado-em-binário (BCD), o comprimento do registro será o número de dígitos decimais que o registro pode armazenar.

Por conveniência, aos vários registros da UAL são geralmente atribuídos nomes tais como registro *X*, registro *B*, registro *MQ* etc., e aos biestáveis são depois atribuídos os mesmos nomes, de modo que o registro *X* deve conter os biestáveis X_1, X_2, X_3 etc.

Alguns computadores possuem um registro chamado *acumulador* que é o principal registro das operações aritméticas e lógicas. Este registro armazena o resultado de cada operação aritmética ou lógica, e o conjunto de circuitos lógicos é anexado a este registro de modo que as operações necessárias possam ser efetuadas sobre seu conteúdo e quaisquer outros registros envolvidos.

Um acumulador é, portanto, um registro de armazenamento básico do elemento aritmético. Se a máquina for instruída para *carregar* o acumulador, o elemento de controle inicialmente limpará o acumulador de tudo que possa ter sido armazenado nele e depois coloca o operando selecionado no armazenamento dentro do registro acumulador. Se o computador for instruído, para *somar*, o número armazenado no acumulador representará o segundo somando. O primeiro somando será então localizado na memória, e o conjunto de circuitos do computador somará este número (o primeiro somando) ao número previamente armazenado no acumulador (o segundo somando) e armazenará a soma no acumulador. Observemos que o segundo somando original não mais será armazenado no acumulador após a adição. Além disso, a soma pode então ou permanecer no acumulador ou ser transferida para a memória, dependendo do tipo de computador. Este capítulo tratará somente dos processos da adição, subtração etc., e não do processo da localização na memória do número a ser adicionado ou da transferência de números para a memória. Estas operações serão tratadas nos capítulos seguintes.

Alguns computadores, em vez de terem um único acumulador, terão dois ou mais acumuladores, e estes são chamados, por exemplo, acumulador *A* e acumulador *B* (como nos computadores Hewlett-Packard) ou ACC1, ACC2 etc. (como nos computadores da Data General Corporation). Quando o número de registros fornecidos para reter os operandos se tornar maior do que quatro, no entanto, os registros são muitas vezes chamados *registros gerais*, e aos registros individuais são atribuídos nomes tais como registro 4 geral, registro 8 geral etc.

6.2 REPRESENTAÇÃO DE INTEIROS

Os números usados nas máquinas digitais devem ser representados usando-se dispositivos de armazenamento tais como os biestáveis. O sistema de representação de números mais direto para os dispositivos de armazenamento de valor binário é um *sistema de representação de inteiros*. A Fig. 6.1*(a)* mostra um registro de quatro biestáveis, X_1, X_2, X_3 e X_4, usado para armazenar números. Escrevendo-se simplesmente os valores ou estados dos biestáveis, obtém-se o número na forma de inteiro, de modo que $X_1 = 1$, $X_2 = 1$, $X_3 = 0$, $X_4 = 0$ produz 1 100, ou 12 em decimal, enquanto que $X_1 = 0$, $X_2 = 1$, $X_3 = 0$, $X_4 = 1$ dá 0101, ou 5 em decimal.

Geralmente é necessário representar tanto os números positivos quanto os negativos; assim, é necessário um bit adicional chamado *bit de sinal*. Este bit é geralmente colocado à esquerda dos bits de magnitude. Na Fig. 6.1*(b)* escolhemos X_0 como bit de sinal, e X_1, X_2, X_3 e X_4 darão a magnitude. Um 0 em X_0 significa que o número é positivo, e um 1 em X_0 significa que o número é negativo (esta é a convenção usual); portanto, $X_0 = 0$, $X_1 = 1$, $X_2 = 1$, $X_3 = 0$ e $X_4 = 1$ produz 1101 positivo, ou + 13 em decimal, e $X_0 = 1$, $X_1 = 1$, $X_2 = 1$, $X_3 = 0$ e $X_4 = 1$ dá 1101 negativo, ou −13 em decimal.

Este sistema é chamado *sistema binário de inteiros com sinal* ou *sistema de inteiros binários de magnitude com sinal*.

Se um registro contiver oito biestáveis, um número binário com sinal no sistema deve ter magnitude 7, ou inteiro, bits e um único bit de sinal; assim, 00001111 seria +15 e 10001111 seria −15, uma vez que o 0 e o 1 na frente dariam o + e o − apenas.

As magnitudes dos números que podem ser armazenados nos dois sistemas representativos na Fig. 6.1 são como segue:

1. Na representação de inteiros em binário, um registro com n biestáveis pode armazenar de 0 (decimal) até $2^n - 1$. Assim, num registro de 6 bits, podemos armazenar de 000000 até 111111, onde 111111 é 63, que equivale a $2^6 - 1$, ou $64 - 1$.

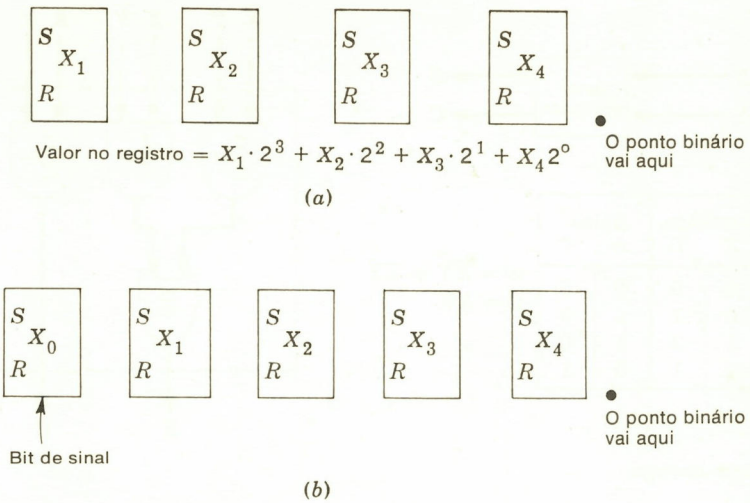


Fig. 6.1 Sistemas de representação. *(a)* Representação de inteiros; *(b)* sistema de magnitude-mais-sinal.

2. O sistema de representação de inteiros binários com sinal tem uma faixa de desde $-(2^{n-1} - 1)$ até $+2^{n-1} - 1$ para um dispositivo de armazenamento binário. Por exemplo, um registro com 7 biestáveis pode armazenar desde −111111 até +111111, que equivale a −63 até +63 $[-(2^6 - 1)$ até $+(2^6 - 1)]$.

Nas seções seguintes aprenderemos como efetuar várias operações aritméticas e lógicas nos registros.

6.3 O CIRCUITO MEIA-SOMA BINÁRIO

Um módulo básico usado nos elementos aritméticos binários é o *circuito meia-soma* ou *semi-somador*. A função do circuito semi-somador é adicionar dois dísticos binários, produzindo uma soma e um transporte de acordo com as regras da adição binária mostradas no Quadro 6.1. A Fig. 6.2 mostra um projeto de um circuito semi-somador. Há duas entradas para o semi-somador, designadas *X* e *Y* na Fig. 6.2, e duas saídas, designadas *S* e *C*. O semi-somador efetua a operação de adição binária das duas entradas binárias mostradas no Quadro 6.1. Esta é a *adição aritmética*, não a adição lógica ou de álgebra booleana.

QUADRO 6.1

Entradas	Bit de soma	
0 + 0	0	
0 + 1	1	
1 + 0	1	
1 + 1	0	com um transporte de 1

Conforme indicado na Fig. 6.2, há duas entradas e duas saídas no semi-somador. Se qualquer uma das entradas for um 1, mas não ambas, a saída na linha *S* será um 1. Se ambas as entradas forem 1, a saída na linha *C* (de transporte = *carry*) será um 1; para todos os outros estados haverá uma saída 0 na linha de TRANSPORTE (linha CARRY). Estas relações podem ser escritas em forma booleana como segue:

$$S = X\bar{Y} + \bar{X}Y$$
$$C = XY$$



ENTRADA X	ENTRADA Y	SAÍDA S	SAÍDA C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$S = \bar{X}Y + X\bar{Y}$
$C = XY$

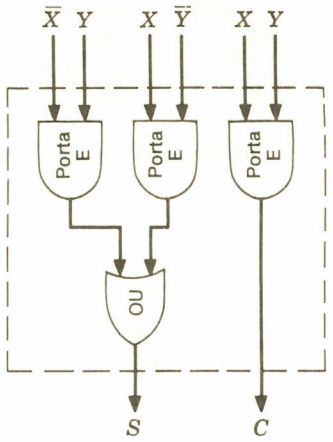


Fig. 6.2 Semi-somador.

Um circuito de *um quarto de soma* consiste das duas entradas para o semi-somador e da saída S somente. A expressão lógica para este circuito é portanto $S = X\bar{Y} + \bar{X}Y$. Esta é também a relação *OU exclusivo* da álgebra booleana (ver Cap. 3).

6.4 O CIRCUITO SOMADOR COMPLETO

Quando devem ser adicionados mais de dois dígitos binários, não serão adequados vários circuitos semi-somadores, pois o semi-somador não tem entrada para manipular os transportes provenientes de outros dígitos.

Consideremos a adição dos seguintes dois números binários:

$$\begin{array}{r} 1011 \\ +\ 1110 \\ \hline 11001 \end{array} = \text{soma} \qquad \begin{array}{r} 1011 \\ +\ 1110 \\ \hline 0101 \\ 1\ \ 1\ \ \ \\ \hline 11001 \end{array} \begin{array}{l} \\ \\ = \text{soma parcial} \\ = \text{bits de transporte} \\ = \text{soma completa} \end{array}$$

Conforme indicado, os transportes gerados em cada coluna devem ser considerados durante o processo da adição; portanto, o conjunto de circuitos somadores capaz de adicionar os conteúdos de dois registros deve incluir provisões para manipular transportes bem como os bits do primeiro e do segundo somandos. Deve haver portanto três entradas para cada estágio de um somador de múltiplos dígitos — excetuando-se o estágio dos bits menos significativos — um para cada entrada dos números que estão sendo adicionados e um para qualquer transporte que possa ter sido gerado ou propagado pelo estágio anterior.

O símbolo de diagrama-bloco de um *somador binário completo*, que manipulará estes transportes, é ilustrado na Fig. 6.3, como é a tabela completa das relações de entrada-saída do somador completo. Há três entradas para o somador completo: as entradas X e Y dos respectivos dígitos dos registros a serem adicionados, e a entrada C_i, que é para qualquer transporte gerado pelo estágio anterior. As duas saídas são S, que é o valor de saída para aquele estágio da adição, e C_o, que produz o transporte a ser adicionado no estágio seguinte.† As expressões booleanas das relações de entrada-saída para cada uma das duas saídas são também apresentadas na Fig. 6.3, como é a expressão da saída C_o em forma simplificada.

†C_i é para *vem um* e C_o é para *vai um*.

Um circuito somador completo pode ser construído de dois semi-somadores, conforme ilustrado na Fig. 6.4. A construção de um circuito somador completo a partir de dois circuitos semi-somadores pode não ser necessariamente a técnica mais econômica, no entanto, e geralmente os somadores completos são projetados diretamente das relações de entrada-saída ilustradas na Fig. 6.3.



ENTRADA			SAÍDA	
X	Y	C_i	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Fig. 6.3 Somador completo.

$$S = \bar{X}\bar{Y}C_i + \bar{X}Y\bar{C}_i + X\bar{Y}\bar{C}_i + XYC_i$$

$$C_o = \bar{X}YC_i + X\bar{Y}C_i + XY\bar{C}_i + XYC_i$$

ou

$$C_o = XC_i + XY + YC_i$$

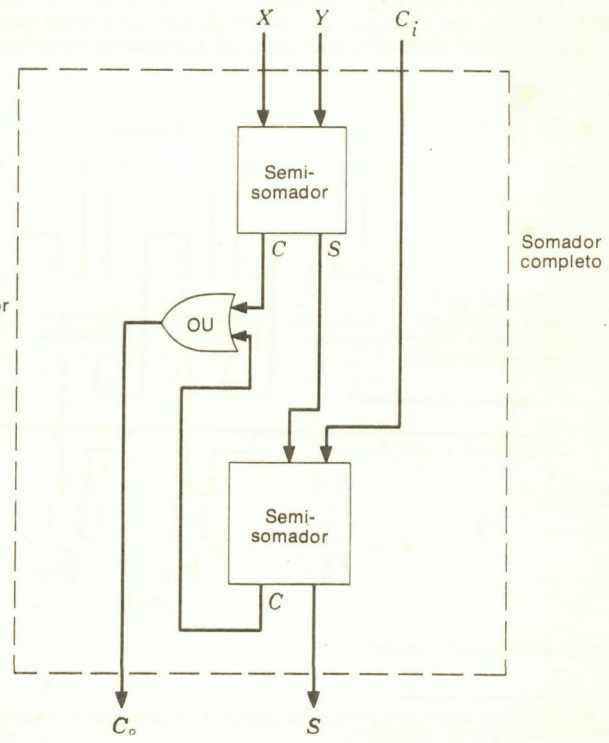


Fig. 6.4 Relação entre semi-somador e somador completo.

6.5 UM SOMADOR BINÁRIO PARALELO

Um somador binário paralelo de 4 bits é ilustrado na Fig. 6.5. O propósito deste somador é adicionar dois inteiros binários de 4 bits. As entradas do primeiro somando são denominadas X_1 até X_4, e os bits do segundo somando são representados por Y_1 até Y_4.* O somador mostrado não possui a capacidade de manipular bits de sinal das palavras binárias a serem adicionados, mas apenas adiciona as magnitudes dos números armazenados. O conjunto de circuitos adicional necessário a manipular bits de sinal depende do fato de os números negativos serem representados em verdadeira grandeza ou nos sistemas de complemento de 1 ou de 2, e este problema será descrito posteriormente.

Consideremos a adição dos dois seguintes números binários de 4 bits:

$$\begin{array}{rll} & 0\,1\,1\,1 & \text{onde } X_1 = 0,\ X_2 = 1,\ X_3 = 1 \text{ e } X_4 = 1 \\ & 0\,0\,1\,1 & \text{onde } Y_1 = 0,\ Y_2 = 0,\ Y_3 = 1 \text{ e } Y_4 = 1 \\ \text{Soma} = & 1\,0\,1\,0 & \end{array}$$

A soma deve ser, portanto, $S_1 = 1$, $S_2 = 0$, $S_3 = 1$ e $S_4 = 0$.

A operação do somador pode ser verificada como segue: Uma vez que X_4 e Y_4 são os dígitos menos significativos, eles não podem receber um transporte proveniente de um estágio anterior. No problema acima, X_4 e Y_4 são ambos 1, sua soma é portanto 0, e um transporte é gerado e adicionado no somador completo dos bits X_3 e Y_3. Os bits X_3 e Y_3 são também ambos 1, como é a entrada de transporte para este estágio; portanto, a linha S_3 de saída de soma transporta um 1, e a linha CARRY para o estágio seguinte também transporta um 1. Já que X_2 é um 1, Y_2, um zero e a entrada CARRY é 1, a linha S_2 de saída de soma transportará um 0, e o transporte para o estágio seguinte será um 1. Ambas as entradas X_1 e Y_1 são iguais a 0, e a linha de entrada CARRY para este estágio somador é igual a 1; portanto, a linha S_1 de saída de soma representará um 1 e a linha de saída CARRY, designada como "estouro" ou "transbordamento" na Fig. 6.5, terá uma saída zero.

A mesma configuração básica ilustrada na Fig. 6.5 pode ser estendida a qualquer número de bits. Um somador de 7 bits pode ser construído usando-se 7 somadores completos, e um somador de 20 bits usando-se 20 somadores completos.

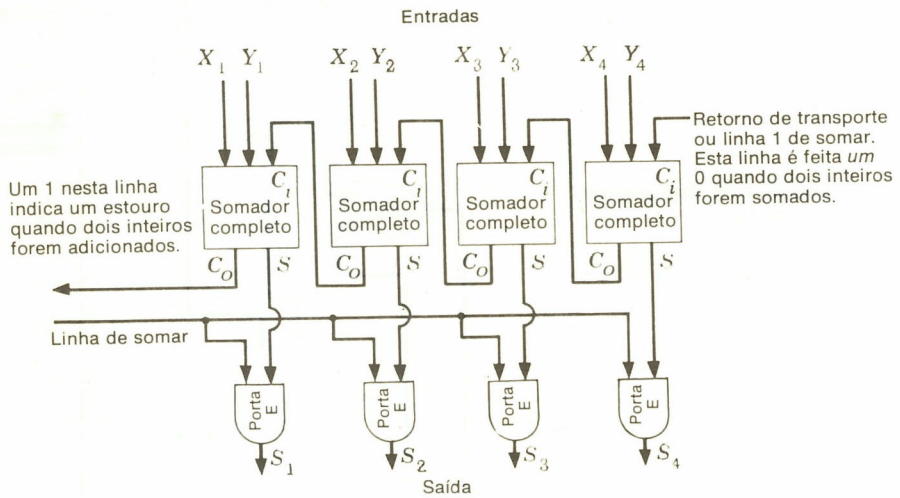


Fig. 6.5 Somador paralelo.

*Estas entradas seriam normalmente dos registros X e Y de biestáveis, e o somador adicionaria o número em X ao número em Y dando a soma, ou S_1 até S_4.

Deve ser observado que a linha OVERFLOW (linha de TRANSBORDAMENTO) pode ser usada para ativar o somador de 4 bits na Fig. 6.5 para ter uma saída de 5 bits. Em geral, isto não é feito, no entanto, porque o primeiro e o segundo somandos vêm ambos do dispositivo de armazenamento, e, portanto, seus comprimentos constituem o comprimento da palavra básica do computador, e uma palavra mais comprida não pode ser prontamente armazenada pela máquina. Foi explicado inicialmente que uma máquina com um comprimento da palavra de n bits [consistindo em um bit de sinal e de $(n - 1)$ bits para designar a magnitude] pode expressar números binários de $(-2^{n-1} + 1)$ a $(2^{n-1} - 1)$. Um número dentro destes limites é chamado *representável*. Uma vez que o somador simples de 4 bits na Fig. 6.5 não possui bit de sinal, ele pode apenas representar inteiros binários de 0 a 15. Se 1100 e 1100 forem adicionados no somador ilustrado na Fig. 6.5, haverá uma saída 1 na linha OVERFLOW porque a soma destes dois números é 11000. Este número é o decimal 24 e não pode ser representado neste sistema. Tal número é denominado não-representável para este registro muito pequeno, e, quando dois inteiros são adicionados de tal modo que sua soma seja não-representável (isto é, contenha bits em grande número), então dizemos que a soma *transborda* ou *estoura*, ou que ocorre um *estouro* ou *transbordamento*. Um 1 na linha CARRY do somador completo conectado aos dígitos mais significativos indica este fato. O estouro gerado neste caso é freqüentemente conectado ao conjunto de circuitos que alerta o computador, gerando uma indicação de que ocorreu uma operação de transbordamento ou estouro. Constitui uma das funções do computador detectar tais estouros e tratar deles.

As portas E conectadas às linhas de saída S provenientes dos quatro somadores são usadas para comandar a soma para dentro do registro correto.

6.6 NÚMEROS POSITIVOS E NEGATIVOS

Quando se escreve números no sistema decimal, a prática comum é escrever o número como uma grandeza precedido de um sinal mais ou menos, que indica se o número é positivo ou negativo. Assim, +125 é positivo e −125 é 125 negativo. A mesma prática é geralmente usada com números binários: +111 é 7 positivo e −110 é 6 negativo. Para manipular tanto os números positivos quanto os negativos, o computador deve possuir algum meio de distinguir um positivo de um negativo, e, como foi explicado anteriormente, a palavra do computador usualmente contém um bit de sinal, geralmente adjacente ao bit mais significativo na palavra do computador. Nos sistemas a serem descritos, um 1 no bit de sinal indicará um número negativo, e um 0 no bit de sinal indicará um número positivo.

Na Seção 6.1 examinamos a representação de números usando um sistema de representação da magnitude do inteiro com sinal. Há dois outros sistemas de representação, no entanto, que são mais freqüentemente usados — os sistemas de complemento de 2 e de 1. (O sistema de complemento de 2 é o mais freqüentemente usado no momento.) A vantagem destes sistemas é que tanto os números positivos quanto os negativos podem ser adicionados ou subtraídos usando apenas um somador do tipo já explicado.

Eis aqui os três sistemas básicos.

1. Os números negativos podem ser armazenados em sua *forma de verdadeira grandeza*. O número binário −0011 será portanto armazenado como 1,0011, onde o 1 indica que o número armazenado é negativo e o 0011 indica a magnitude do número.†
2. O *complemento de 1* da magnitude pode ser usado para representar um número negativo. O número binário −0111 será portanto representado como 1, 1000, onde o 1 indica que o número é negativo e 1000 é o complemento de 1 na magnitude. (O complemento de 1 é formado simplesmente complementando-se cada bit da magnitude positiva.)
3. O *complemento de 2* pode ser usado para representar um número binário negativo. Por exemplo, −0111 deve ser armazenado como 1,1001, onde o 1 no bit de sinal indica que o número é negativo e o 1001 é o complemento de 2 da magnitude do número. (O complemento de 2 é formado achando-se o complemento de 1 da parte da magnitude de 0111, obtendo-se 1000, e depois adicionando-se 1 ao dígito menos significativo, obtendo-se 1001.)

†Novamente notamos que um ponto binário é usado para separar o bit de sinal dos bits de magnitude. Portanto, 0,011 é +3. Outros símbolos foram usados, mas este é convencional e o usaremos.

6.7 ADIÇÃO NO SISTEMA DE COMPLEMENTO DE 1

O sistema de complemento de 1 para representação de números negativos é freqüentemente usado em máquinas binárias paralelas. A principal razão para isto é a facilidade com que o complemento de 1 de um número binário pode ser formado, uma vez que somente é requerida a complementação de cada bit de um número binário armazenado num registro com biestáveis. Antes de discutir as implementações de um somador no sistema de complemento de 1, as quatro situações básicas possíveis que podem surgir na adição das combinações de números positivos e negativos no sistema de complemento de 1 serão observadas:

1. Quando um número positivo for adicionado a um outro número positivo, a adição de todos os bits, inclusive do bit de sinal, é direta. Uma vez que ambos os bits de sinal serão 0, não serão gerados nenhuma soma ou transporte no somador de bit de sinal e a saída permanecerá 0. Eis aqui um exemplo da adição de dois números positivos de 4 bits:†

NOTAÇÃO NORMAL	PALAVRA DE COMPUTADOR
+0011	0,0011
+0100	0,0100
+0111	0,0111

2. Quando são adicionados um número positivo e um número negativo, a soma pode ser positiva ou negativa. Se o número positivo tiver magnitude maior, a soma será positiva; e, se o número negativo for maior em magnitude ou módulo, a soma será negativa. No sistema de complemento de 1, a resposta estará correta como está se a soma dos dois números for de valor negativo. Neste caso, não será gerado nenhum estouro (ou transbordamento) quando os números forem adicionados. Por exemplo:

+0011	0,0011
−1100	1,0011
−1001	1,0110

Neste caso, a saída do somador será 10110, da qual os últimos 4 bits são o complemento de 1 de 1001, a magnitude correta da soma. O 1 no bit de sinal também está correto, indicando um número negativo.

3. Se o número positivo for maior do que o número negativo, a soma, antes que o retorno de transporte seja adicionado, estará incorreta. A adição do retorno (ou realimentação) de transporte corrigirá esta soma. Haverá um 0 no bit de sinal indicando que a soma é positiva.

+1001 =	0,1001	+0011 =	0,0011
−0100 =	1,1011	−0010 =	1,1101
+0101	0,0100	+0001	0,0000
	→ 1		→ 1
	0,0101		0,0001

Observemos o que acontece quando dois números de igual magnitude, mas de sinais contrários, são adicionados:

+1011 =	0,1011	+0000 =	0,0000
−1011 =	1,0100	−0000 =	1,1111
0000	1,1111	0000	1,1111

†Nesta e em todas as discussões que se seguem, admitimos que o resultado (soma) não excede a capacidade do número de dígitos que estão sendo usados. Isto será discutido posteriormente.

O resultado nestes casos será um zero negativo (1,1111), que está correto.

4. Quando são adicionados dois números negativos, sempre será gerado um retorno de transporte, como será um transporte proveniente do somador dos primeiros bits das magnitudes dos números. Isto colocará um 1 no bit de sinal.

$$\begin{array}{rr} -0011 = & 1,1100 \\ -1011 = & 1,0100 \\ \hline -1110 & \ulcorner 1,0000 \\ & \quad \longrightarrow 1 \\ & \hline & 1,0001 \end{array} \qquad \begin{array}{rr} -0100 = & 1,1011 \\ -0111 = & 1,1000 \\ \hline -1011 & \ulcorner 1,0011 \\ & \quad \longrightarrow 1 \\ & \hline & 1,0100 \end{array}$$

A saída do somador estará em forma de complemento de 1 em cada caso, com um 1 na posição do bit de sinal.

Do exposto acima, vemos que a fim de implementar um somador que irá manipular números de complemento de 1 com sinal e magnitude de 4 bits, podemos simplesmente acrescentar um outro somador completo à configuração na Fig. 6.5. As entradas de sinal serão designadas por X_0 e Y_0, e a saída C_0 do somador conectado a X_1 e Y_1 será conectada à entrada C_i do novo somador completo para X_n e Y_n. A saída C_n do somador de X_0 e Y_0 será conectada à entrada C_i do somador de X_1 e Y_1. A saída S_0 do novo somador dará o dígito de sinal para a soma. (O estouro não será detectado neste somador, são necessárias portas adicionais.)

6.8 ADIÇÃO NO SISTEMA DE COMPLEMENTO DE 2

Quando forem representados números negativos no sistema de complemento de 2, a operação de adição será muito semelhante à da adição no sistema de complemento de 1. Nas máquinas paralelas, o complemento de 2 de um número armazenado num registro pode ser formado complementando-se de início o registro e depois adicionando-se 1 ao bit menos significativo do registro. Este processo requer duas etapas e, portanto, consome† um tempo maior do que o sistema de complemento de 1; no entanto, o sistema de complemento de 2 tem a vantagem de não requerer um retorno de transporte durante a adição.

As quatro situações que podem ocorrer na adição de dois números quando for usado o sistema de complemento de 2 são como segue:

1. Quando ambos os números forem positivos, a situação é completamente idêntica àquela do caso 1 no sistema de complemento de 1 que foi discutida.
2. Quando um número for positivo e o outro negativo, e o maior número for o número positivo, será gerado um transporte através do bit de sinal. Este transporte pode ser rejeitado, já que as saídas do somador estão corretas, conforme mostrado abaixo.

$$\begin{array}{rr} +0111 = & 0,0111 \\ -0011 = & +1,1101 \\ \hline +0100 & \ulcorner 0,0100 \\ & \llcorner \text{o transporte é rejeitado} \end{array} \qquad \begin{array}{rr} +1000 = & 0,1000 \\ -0111 = & +1,1001 \\ \hline +0001 & \ulcorner 0,0001 \\ & \llcorner \text{o transporte é rejeitado} \end{array}$$

3. Quando forem adicionados um número positivo e um número negativo, e o número negativo for o maior, não resultará nenhum transporte no bit de sinal e a resposta estará novamente correta em sua posição.

$$\begin{array}{rr} +0011 = & 0,0011 \\ -0100 = & 1,1100 \\ \hline -0001 & 1,1111 \end{array} \qquad \begin{array}{rr} +0100 = & 0,0100 \\ -1000 = & 1,1000 \\ \hline -0100 & 1,1100 \end{array}$$

†Geralmente, este 1 é "introduzido sem ser percebido" durante o cálculo, como será mostrado.

Nota: Um 1 deve ser acrescentado ao bit menos significativo de um número negativo de complemento de 2 quando se o converte numa magnitude. Por exemplo:

$$\begin{array}{rl} 1,0011 = & 1100 \text{ forma de complemento de 1} \\ & 0001 \text{ acrescentar 1} \\ \hline & -1101 \end{array}$$

Quando ambos os números forem da mesma magnitude, o resultado é como segue:

$$\begin{array}{rr} +0011 = & 0,0011 \\ -0011 = & 1,1101 \\ \hline 0000 & 0,0000 \end{array}$$

Quando forem adicionados um número positivo e um número negativo da mesma magnitude, o resultado será um zero positivo.

4. Quando dois números negativos forem adicionados, será gerado um transporte no bit de sinal e também no bit à direita do bit de sinal. Isto fará com que 1 seja colocado no bit de sinal, que está correto, e o transporte do bit de sinal pode ser rejeitado.

$$\begin{array}{rrrr} -0011 = & 1,1101 & -0011 = & 1,1101 \\ -0100 = & 1,1100 & -1011 = & 1,0101 \\ \hline -0111 & \ulcorner 1,1001 & 1110 & 1,0010 \end{array}$$

$\llcorner\!\!\rightarrow$ o transporte é rejeitado

Nas máquinas paralelas, a adição de números positivos e negativos é absolutamente simples, já que qualquer estouro do bit de sinal é simplesmente desprezado. Assim, para o somador paralelo na Fig. 6.5, simplesmente acrescentamos um outro somador completo, com X_0 e Y_0 como entradas e com a linha C_o de transporte do somador completo, que adiciona X_1 e Y_1, conectada à entrada C_i de transporte para o somador completo de X_0 e Y_0. Um 0 é colocado na entrada C_i para o somador conectado a X_4 e Y_4.

Esta simplicidade na adição e na subtração tem feito o sistema de complemento de 2 o mais popular para as máquinas paralelas. De fato, quando são usados os sistemas de magnitude-com-sinal, os números geralmente são convertidos em complemento de 2 antes de a adição ou subtração de números negativos ser efetuada; então, os números são novamente convertidos à magnitude-com-sinal.

6.9 ADIÇÃO E SUBTRAÇÃO NUM ELEMENTO ARITMÉTICO PARALELO

Examinaremos agora o projeto de uma rede lógica que ou adiciona ou subtrai dois números. A rede deve ter uma linha de entrada de SOMAR (ADD) e uma linha de entrada de SUBTRAIR (SUBTRACT), bem como as linhas que transportam a representação dos números a serem adicionados ou subtraídos. Quando a linha ADD for um 1, a soma dos números deve estar sobre as linhas de saída e, quando a linha SUBTRACT for um 1, a diferença deve estar sobre as linhas de saída. Se ambas as ADD e SUBTRACT forem 0, a saída deverá ser 0.

Observemos inicialmente que, se a máquina for capaz de adicionar tanto números positivos quanto negativos, a subtração pode ser efetuada complementando-se o subtraendo e adicionando-se em seguida. Por exemplo, $8 - 4$ produz o mesmo resultado que $8 + (-4)$, e $6 - (-2)$ produz o mesmo resultado que $6 + 2$. Portanto, a subtração pode ser efetuada pelo elemento aritmético capaz de adicionar, formando o complemento do subtraendo e em seguida adicionando. Por exemplo, no sistema de complemento de 1, quatro casos podem surgir:

DOIS NÚMEROS POSITIVOS

$$\begin{array}{r} 0,0011 \\ -0,0001 \end{array}$$

Complementando o subtraendo e adicionando

$$\begin{array}{r} 0,0011 \\ 1,1110 \\ \hline \ulcorner 0,0001 \\ \llcorner\!\!\rightarrow \text{transporte 1} \\ \hline 0,0010 \end{array}$$

DOIS NÚMEROS NEGATIVOS

$$\begin{array}{r} 1,1101 \\ -1,1011 \\ \hline \end{array} \quad \text{Complementando} \quad \begin{array}{r} 1,1101 \\ 0,0100 \\ \hline 0,0001 \\ \rightarrow \text{transporte 1} \\ \hline 0,0010 \end{array}$$

MINUENDO POSITIVO
SUBTRAENDO NEGATIVO

$$\begin{array}{r} 0,0010 \\ -1,1101 \\ \hline \end{array} = \begin{array}{r} 0,0010 \\ 0,0010 \\ \hline 0,0100 \end{array}$$

MINUENDO NEGATIVO
SUBTRAENDO POSITIVO

$$\begin{array}{r} 1,0101 \\ -0,0010 \\ \hline \end{array} = \begin{array}{r} 1,0101 \\ 1,1101 \\ \hline 1,0010 \\ \rightarrow \text{transporte 1} \\ \hline 1,0011 \end{array}$$

As mesmas regras básicas se aplicam à subtração no sistema de complemento de 2, exceto que qualquer transporte gerado nos somadores de bit de sinal é simplesmente abandonado. Neste caso, o complemento de 2 do subtraendo é formado, e o número complementado é depois adicionado ao minuendo sem nenhum retorno de transporte.

Examinaremos agora a implementação de uma rede somadora e subtratora combinada. O problema primordial consiste em formar o complemento do número a ser subtraído. Esta complementação do subtraendo pode ser efetuada de diversas maneiras. No sistema de complemento de 1, se o registro de armazenamento for composto de biestáveis, o complemento de 1 pode ser formado ligando-se simplesmente o complemento de cada entrada ao somador. O 1 que deve ser adicionado à posição menos significativa para formar um complemento de 2 pode ser adicionado quando os dois números forem adicionados, conectando-se um 1 na entrada de TRANSPORTE (CARRY) do somador dos bits menos significativos.

Um circuito lógico completo capaz de adicionar ou subtrair dois números de complemento de 2 com sinal é mostrado na Fig. 6.6. Um número é representado por X_0, X_1, X_2, X_3 e X_4, e o outro número por Y_0, Y_1, Y_2, Y_3, Y_4. Há dois sinais de controle, ADD (SOMAR) e SUBTRACT (SUBTRAIR). Se nenhum dos sinais de controle for um 1 (isto é, ambos forem 0), então as saídas dos cinco somadores completos, que são S_0, S_1, S_2, S_3, S_4, serão todas 0. Se a linha de controle ADD for feita um 1, a soma do número X e do número Y aparecerá como S_0, S_1, S_2, S_3 e S_4. Se a linha SUBTRACT for feita um 1, a diferença entre X e Y (isto é, $X - Y$) aparecerá em S_0, S_1, S_2, S_3 e S_4.

Observemos que a rede de porta E/OU conectada a cada entrada Y seleciona ou Y ou $\bar{Y}$, de modo que, por exemplo, um ADD faz com que Y_1 entre no somador completo apropriado, enquanto que um SUBTRACT faz com que $\bar{Y}_1$ entre no somador completo.

A fim de ou SOMAR (ADD) ou SUBTRAIR (SUBTRACT), cada entrada X é conectada ao somador completo apropriado. Quando for solicitada uma subtração, o complemento de cada biestável Y é controlado ou comandado para dentro do somador completo, e um 1 é adicionado conectando-se o sinal SUBTRACT à entrada C_i do somador completo dos bits X_4 e Y_4 de ordem mais baixa. Uma vez que a linha SUBTRACT será um 0 quando SOMARMOS (ADD), um transporte 0 estará sobre esta linha quando a adição for efetuada.

A simplicidade da operação da Fig. 6.6 torna a adição e a subtração muito convenientes para o uso em computador, e este sistema† é, por isso, o mais freqüentemente usado.

A configuração na Fig. 6.6 é a de maior emprego para adição e subtração porque ela proporciona um meio simples e direto tanto para somar quanto para subtrair números positivos ou negativos. Muito freqüentemente, as linhas S_0, S_1, . . ., S_4 são novamente controladas para dentro dos biestáveis X, de modo que a soma ou a diferença dos números X e Y substitui o valor original de X.

Uma consideração importante é o transbordamento ou estouro. Nos computadores digitais, um *estouro* ou *transbordamento* ocorre quando a execução de uma operação resul-

†Um somador-subtrator paralelo de complemento de 1 pode ser feito conectando-se a linha de "vai um" do somador de X_0, Y_0 à linha de "vem um" do somador de X_4, Y_4 (desconectando a linha de subtrair deste somador completo, certamente).

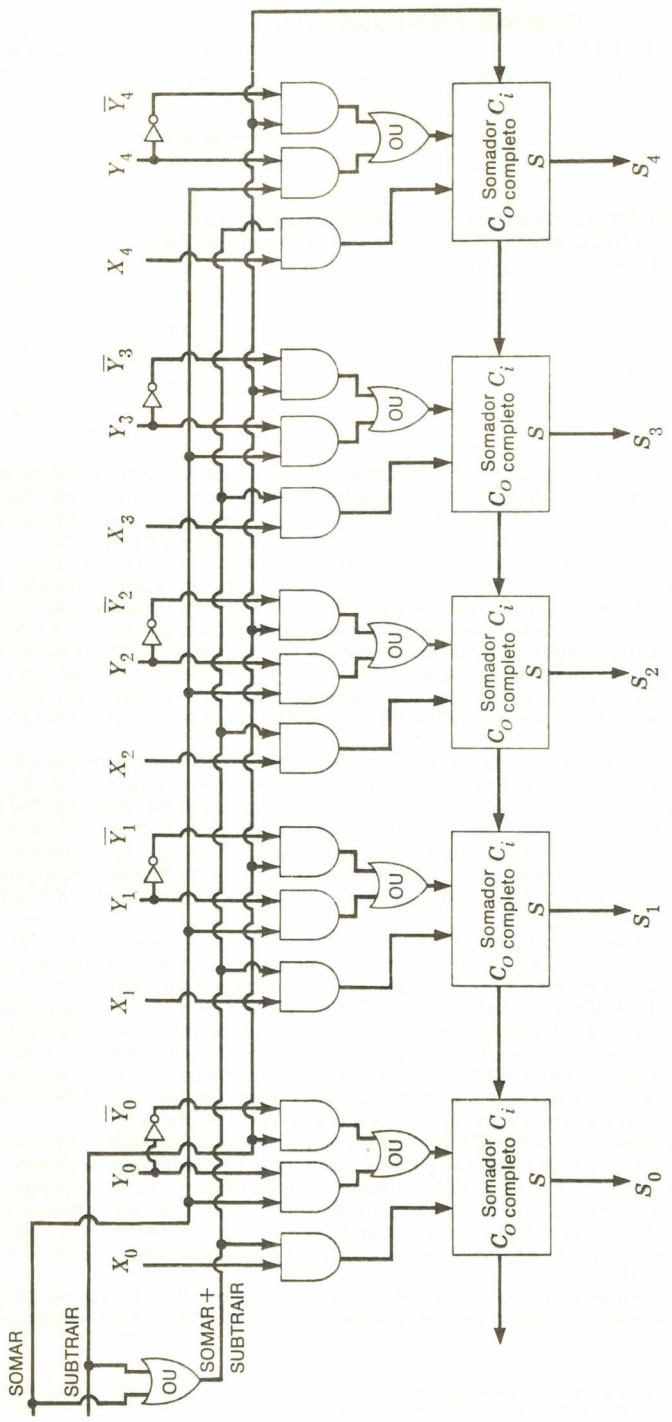


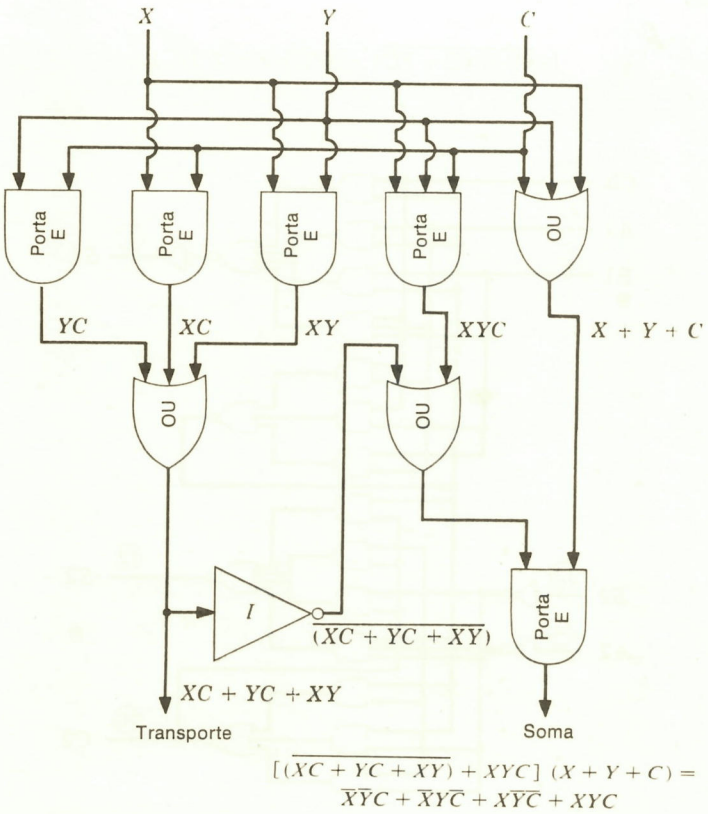
Fig. 6.6 Adição e subtração paralelas.

tar numa quantidade além da capacidade do registro (ou registro de armazenamento) que deve receber o resultado. Uma vez que os registros na Fig. 6.6 têm um bit de sinal e mais quatro bits de magnitude, eles podem armazenar de +15 até −16 em forma de complemento de 2. Portanto, se o resultado de uma adição ou subtração fosse maior do que +15 ou menor do que −16, diríamos que ocorreu um estouro ou transbordamento. Admita que adicionemos +8 a +12; o resultado deve ser +20, e isto não pode ser representado (adequadamente) em complemento de 2 nas linhas S_0, S_1, S_2, S_3 e S_4. O mesmo acontece caso adicionemos −13 e −7 ou subtraiamos −8 de +12. Em cada caso, o conjunto de circuitos lógicos é usado para detectar a condição de estouro e o sinal do elemento de controle do computador. Existem então várias opções, e o que é feito pode depender do tipo de instrução que está sendo executada. (Estouros deliberados são, às vezes, usados em rotinas de dupla-precisão. A multiplicação e a divisão usam os resultados como estão.) Adiaremos isto para o Cap. 9, exceto que uma das questões ou exercícios no fim do capítulo pede um circuito para teste de estouro.

A configuração somador-subtrator paralela na Fig. 6.6 é muito importante, e é instrutivo tentar somar e subtrair vários números em forma de complemento de 2 usando lápis e papel e este circuito lógico.

6.10 O SOMADOR COMPLETO

O somador completo é um componente básico de um elemento aritmético. A Fig. 6.3 ilustra o símbolo de diagrama-bloco do somador completo, junto com uma tabela de combinações dos valores de entrada-saída e as expressões que descrevem as linhas de soma e transporte. As figuras seguintes e o texto descreveram a operação do somador completo. Obser-



$$[\overline{(\overline{XC + YC + XY}) + XYC}]\ (X + Y + C) =$$
$$\overline{X}\overline{Y}C + \overline{X}Y\overline{C} + X\overline{Y}\overline{C} + XYC$$

Fig. 6.7 Somador completo usado nas máquinas IBM.

vemos que um sistema de adição paralela requer um somador completo para cada bit na palavra básica.

Há naturalmente muitas configurações de porta para os somadores binários completos. Exemplos de um somador IBM e de uma embalagem MSI, que contém dois somadores completos, são apresentados a seguir.

1. *Somador binário completo* A Fig. 6.7 ilustra a configuração de somador binário completo usada em vários computadores digitais de propósitos gerais da IBM. Há três entradas no circuito: a entrada X provém de um dos dispositivos de armazenamento no acumulador, a entrada Y provém do correspondente dispositivo de armazenamento no registro a ser acrescentado ao registro do acumulador e a terceira entrada é a entrada CARRY (TRANSPORTE) proveniente do somador do próximo bit menos significativo. As duas saídas são a saída SUM (SOMA) e a saída CARRY (TRANSPORTE). A saída SUM conterá o valor da soma para este dígito particular da saída. A saída CARRY será conectada à entrada CARRY do somador do próximo bit mais significativo (ver Fig. 6.5).

   As saídas das três portas E conectadas diretamente às entradas X, Y e C estão logicamente adicionadas pelo circuito porta OU diretamente abaixo. Se as linhas de entrada ou X e Y, X e C ou Y e C contiverem um 1, haveria uma saída CARRY. A saída deste circuito, escrita sob a forma de equação lógica, é mostrada na figura. Esta pode ser comparada com a expressão deduzida na Fig. 6.3.

   A dedução da saída SUM não é tão direta. A expressão da saída CARRY $XY + XC + YC$ é primeiro invertida (complementada), resultando $\overline{(XY + XC + YC)}$. O produto lógico de X, Y e C é formado por uma porta E e é logicamente adicionado a esta, formando $\overline{(XY + XC + YC)} + XYC$. A soma lógica de X, Y e C é depois multiplicada, formando a expressão

$$[\overline{(XY + XC + YC)} + XYC](X + Y + C)$$

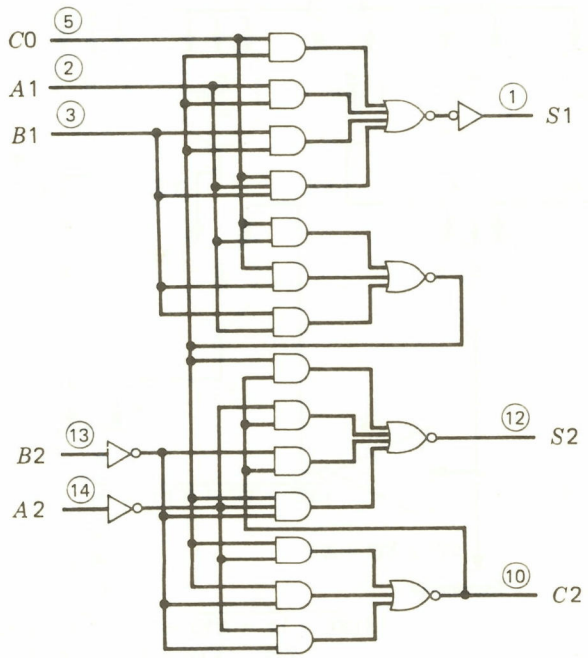


Fig. 6.8 Dois somadores completos num receptáculo de C.I. (Cortesia da Texas Instruments).

Efetuando-se as multiplicações e simplificando-se, esta expressão será $\bar{X}\bar{Y}C + \bar{X}Y\bar{C} + X\bar{Y}\bar{C} + XYC$, que é a expressão deduzida na Fig. 6.3. Reconstituindo a operação lógica do circuito para diferentes valores, verificaremos que a saída SUM será 1 quando apenas um dos valores da entrada for igual a 1, ou quando todos os três valores da entrada forem iguais a 1. Para todas as outras combinações das entradas, o valor da saída será um 0.

2. *Dois somadores completos em um receptáculo de circuito integrado (CI).* A Fig. 6.8 mostra dois somadores completos. Esta embalagem foi desenvolvida para circuitos integrados que usam a lógica transistor-transistor (TTL). O circuito inteiro é embalado num receptáculo de CI. O retardo máximo de uma variação de entrada para uma variação de saída numa saída *S* é da ordem de 8 nanossegundos (ns).† O retardo máximo da entrada para saída da *C2* é de cerca de 6 ns.

O valor do retardo associado a cada transporte constitui uma grandeza importante no cálculo de um somador completo para um sistema paralelo, porque o tempo necessário para somar dois números é determinado pelo tempo máximo decorrido para um transporte se propagar através dos somadores. Por exemplo, se adicionarmos 01111 a 10001 no sistema de complemento de 2, o transporte gerado pelo 1 no dígito menos significativo de cada número deverá propagar-se pelos quatro estágios de transporte e um estágio de soma antes que possamos comandar com segurança a soma para dentro do acumulador. Um estudo da adição destes dois números utilizando a configuração na Fig. 6.5 tornará mais claro este assunto. Este problema é chamado *problema da propagação do "vai um"*.

Há inúmeras técnicas que são usadas nas máquinas de alta velocidade para atenuar este problema. A mais usada é a de um circuito de "vai um antecipado" ou circuito de transposição que calcula o "vai um" de vários estágios simultaneamente e depois fornece este transporte aos estágios seguintes. (Isto é explorado nos exercícios.)

6.11 O SOMADOR DE DECIMAL-CODIFICADO-EM-BINÁRIO (BCD)

Unidades aritméticas que efetuam operações com números armazenados em forma de BCD devem ter a capacidade de somar representações de 4 bits de dígitos decimais. Com esta finalidade é usado um somador de BCD. Um símbolo de diagrama-bloco de um somador é mostrado na Fig. 6.9. O somador tem uma entrada de dígito de segundo somando que consiste de quatro linhas, uma entrada de dígito de primeiro somando de quatro linhas, um "vem um" e um "vai um", e um dígito de soma com quatro linhas de saída. O dígito do segundo somando, o dígito do primeiro somando e o dígito de soma são representados cada um em código BCD 8, 4, 2, 1.

O propósito do somador de BCD na Fig. 6.9 é adicionar os dígitos do primeiro somando e do segundo somando e o "vem um" e produzir um dígito de soma e um "vai um". Este somador pode ser projetado usando-se as técnicas do Cap. 3 e as regras da adição decimal.

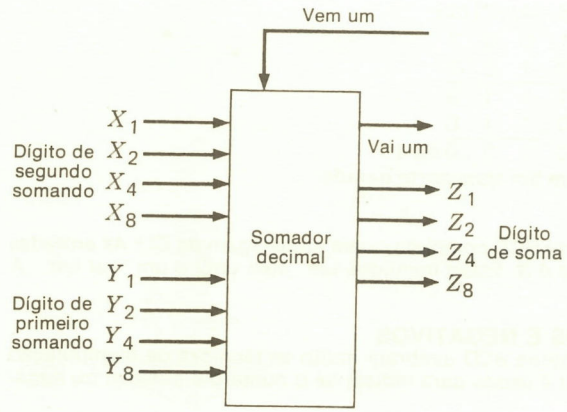


Fig. 6.9 Adição série-paralelo.

†1 ns = 10^{-9} s.

Também é possível fazer um somador de BCD usando-se somadores completos e portas E ou OU. Um somador feito deste modo é mostrado na Fig. 6.10.

Há oito entradas no somador de BCD, quatro X_i, ou dígitos de segundo somando, e quatro Y_i, ou dígitos de primeiro somando. Cada uma destas entradas representará um 0 ou um 1 durante uma determinada adição. Se 3(0011) deve ser adicionado a 2(0010), então $X_8 = 0$, $X_4 = 0$, $X_2 = 1$, e $X_1 = 1$; $Y_8 = 0$, $Y_4 = 0$, $Y_2 = 1$ e $Y_1 = 0$.

O somador básico na Fig. 6.10 consiste dos quatro somadores binários completos no alto da figura e efetua a adição na base 16 quando o objetivo é efetuar a adição na base 10. Portanto, providências devem ser tomadas: (1) gerar transportes e (2) corrigir somas superiores a 9. Por exemplo, se $3_{10}(0011)$ fosse adicionado a $8_{10}(1000)$, o resultado seria $1_{10}(0001)$ com um transporte gerado.

O conjunto de circuitos real, que determina quando um transporte deve ser transmitido aos próximos dígitos mais significativos a serem adicionados, consiste do somador binário completo ao qual estão conectadas as saídas da soma *(S)* provenientes dos somadores das entradas 8, 4, 2 e da porta OU a que o transporte *(C)* dos bits de oito posições está conectado. Um exame do processo da adição indica que um transporte deve ser gerado quando as saídas da soma 8 E 4, ou 8 E 2, ou 8 E 4 E 2 do somador de base 16 representarem 1, ou quando a saída de TRANSPORTE (CARRY) do somador de oito posições contiver um 1 (isto ocorre quando todos os 8 ou todos os 9 forem adicionados).

Sempre que a soma de dois dígitos exceder a 9, a linha CARRY TO NEXT HIGHER ORDER ADDER (TRANSPORTAR PARA O PRÓXIMO SOMADOR DE ORDEM MAIS ALTA) contém um 1 para o somador na Fig. 6.10.

Surge uma outra dificuldade quando é gerado um transporte. Se $7_{10}(0111)$ for adicionado a $6_{10}(0110)$, será gerado um transporte, mas a saída do somador de base 16 será 1101. Este (1101) não representa nenhum dígito decimal no sistema 8, 4, 2, 1 e deve ser corrigido. O método usado para fazer a correção consiste em acrescentar $6_{10}(0110)$ à soma dos somadores de base 16 sempre que um transporte for gerado. Esta adição é efetuada acrescentando-se 1 às linhas de saída da posição de peso 2 e de peso 4 provenientes do somador de base 16 quando for gerado um transporte. Os dois semi-somadores e o somador completo na parte inferior do diagrama executam esta função. Essencialmente, então, o somador efetua a adição na base 16 e corrige a soma, se ela for maior do que 9, acrescentando 6. Vários exemplos deste caso são mostrados abaixo.

$8 + 7 = 15 \qquad 1000 + 0111 =$

$$\begin{array}{cccccc} & & (8) & (4) & (2) & (1) \\ & & 1 & 1 & 1 & 1 \\ & + & 0 & 1 & 1 & 0 \\ \hline 1 & & 0 & 1 & 0 & 1 = 5 \end{array}$$

↑ com um transporte gerado

$9 + 5 = 14$

$$\begin{array}{cccccc} & & (8) & (4) & (2) & (1) \\ & & 1 & 0 & 0 & 1 \\ & & 0 & 1 & 0 & 1 \\ \hline & & 1 & 1 & 1 & 0 \\ & + & 0 & 1 & 1 & 0 \\ \hline 1 & & 0 & 1 & 0 & 0 \text{ ou } 4 \end{array}$$

↑ com um transporte gerado

A Fig. 6.11 mostra um somador de BCD completo numa embalagem de CI.† As entradas são o dígito *A* e o dígito *B*, e a saída é *S*. Estão incluídos um "vem um" e um "vai um". A linha de circuito usada é a CMOS.

6.12 NÚMEROS BCD POSITIVOS E NEGATIVOS

As técnicas de manipulação de números BCD lembram muito as técnicas de manipulação de números binários. Um bit de sinal é usado para indicar se o número é positivo ou nega-

†As embalagens de CI nas Figs. 6.11, 6.13, 6.14 e 6.15 são embalagens típicas MSI de BCD. A notação *A*1, *A*2, *A*3, *A*4 (em vez de X_1, X_2 X_4, X_8) é muitas vezes usada nos 4 bits de um dígito de BCD, e os pesos 1, 2, 4, 8 são compreendidos. Assim, um dígito de BCD em *B*1, *B*2, *B*3, *B*4 teria peso 1 ou *B*1, peso 2 ou *B*2, peso 4 ou *B*3 e peso 8 ou *B*4.

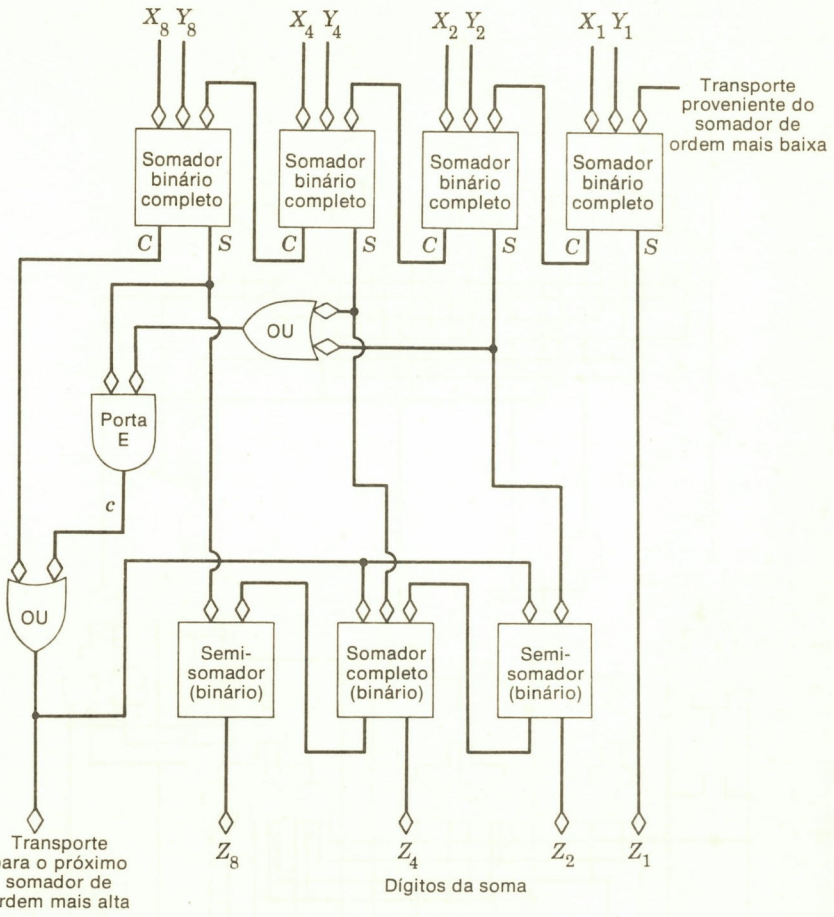


Fig. 6.10 Somador de BCD.

tivo, e há três métodos de representação de números negativos que devem ser considerados. O primeiro método e mais óbvio consiste, certamente, em representar um número negativo em forma de verdadeira grandeza com um bit de sinal, de modo que −645 seja representado por 1,645. As outras duas possibilidades consistem em representar números negativos em forma de complemento de 9 ou de 10, cujas formas lembram as formas binárias do complemento de 1 e de 2.

6.13 ADIÇÃO E SUBTRAÇÃO NO SISTEMA DE COMPLEMENTO DE 9

Quando os números decimais são representados num código binário em que o complemento de 9 é formado quando o número é complementado, a situação é quase a mesma que quando é usado o complemento de 1 para representar um número binário. Podem surgir quatro casos: dois números positivos podem ser adicionados; um número positivo e um negativo podem ser adicionados, produzindo um resultado positivo; um número positivo e um negativo podem ser adicionados, produzindo um resultado negativo; e dois números negativos podem ser adicionados. Já que não há problema quando dois números positivos são adicionados, serão ilustradas as três últimas situações.

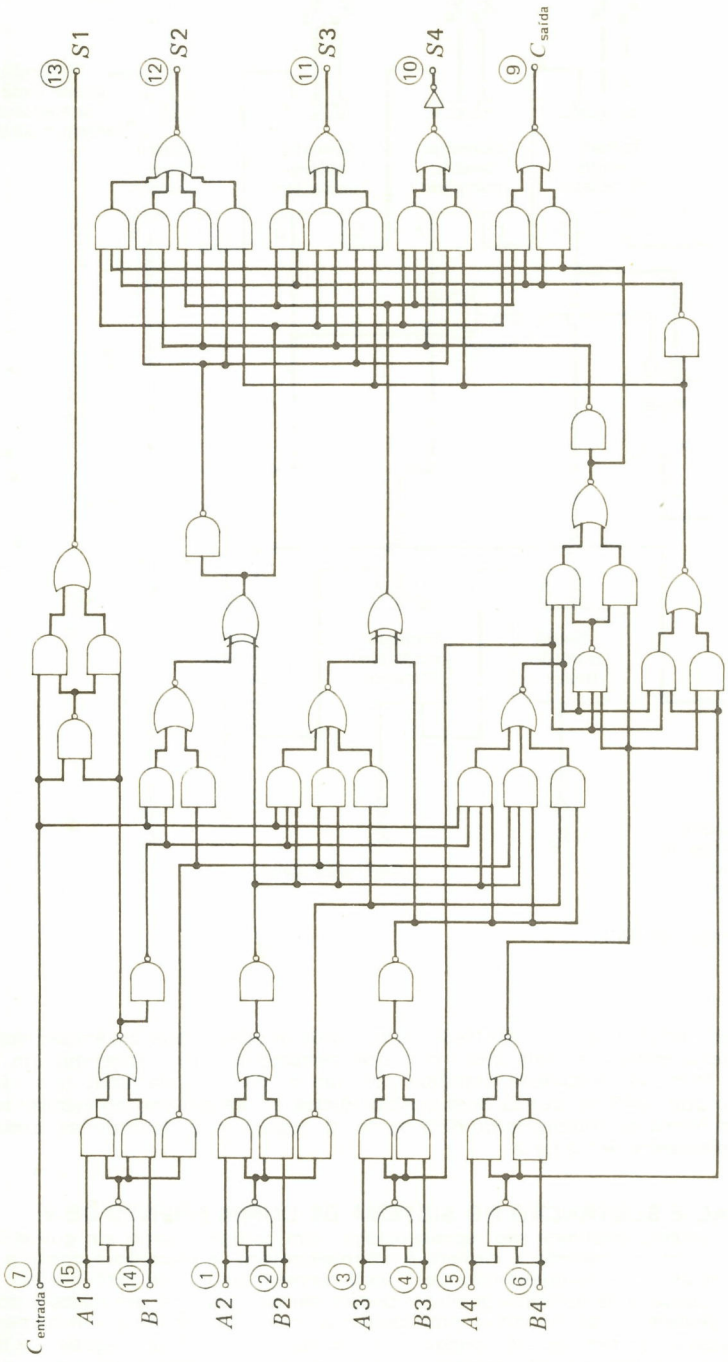


Fig. 6.11 Somador de BCD completo numa embalagem de CI.

Número negativo e número positivo — soma positiva:

$$
\begin{array}{r}
+692 = 0,692 \\
-342 = 1,657 \\
\hline
+350 \quad 0,349 \\
\qquad\qquad \rightarrow 1 \\
\hline
0,350
\end{array}
$$

Número positivo e número negativo — soma negativa:

$$
\begin{array}{rr}
-631 = 1,368 & \\
+342 = 0,342 & \\
\hline
-289 \quad 1,710 & = -289
\end{array}
$$

Dois números negativos:

$$
\begin{array}{r}
-248 = 1,751 \\
-329 = 1,670 \\
\hline
-577 \quad 1,421 \\
\qquad\qquad \rightarrow 1 \\
\hline
1,422 = -577
\end{array}
$$

As regras de manipulação de números negativos no sistema de complemento de 10 são as mesmas das do sistema binário de complemento de 2 quanto a que nenhum transporte deve ser realimentado ou retornado. Um somador BCD paralelo pode portanto ser construído usando apenas o somador de BCD completo como componente básico, e todas as combinações de números positivos e negativos podem portanto ser manipuladas.

Há uma complexidade adicional na adição em BCD, no entanto, porque o complemento de 9 de um dígito BCD não pode ser formado complementando simplesmente cada bit na representação. Em conseqüência, um bloco de circuito lógico chamado *complementador* deve ser usado.

Para ilustrar o tipo de circuito que pode ser usado para formar complementos dos

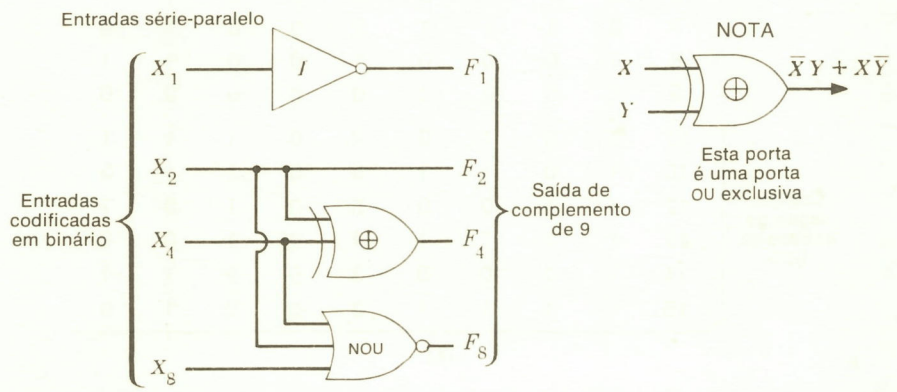
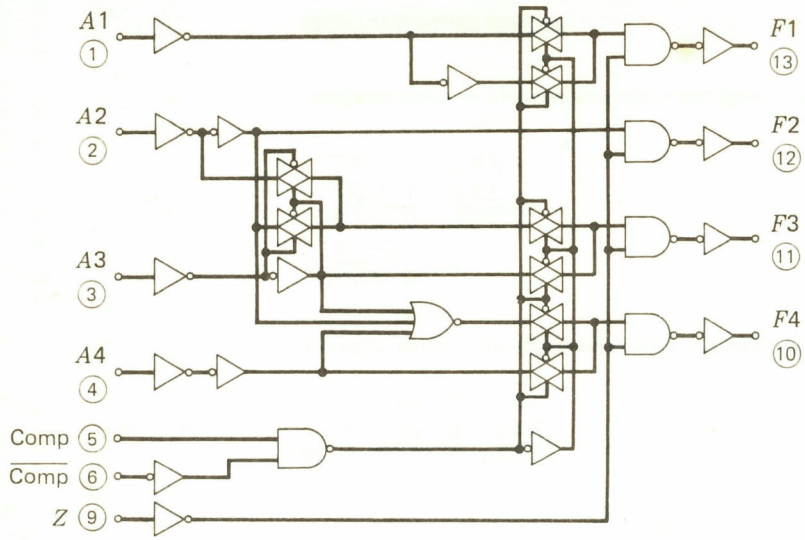


Fig. 6.12 Circuito lógico para formação de complemento de 9 de dígitos de BCD 8421.



(*a*)

ENTRADA EQUIVALENTE DECIMAL	ENTRADAS				SAÍDA EQUIVALENTE DECIMAL	SAÍDAS			
	*A*4	*A*3	*A*2	*A*1		*F*1	*F*2	*F*3	*F*4
0	0	0	0	0	9	1	0	0	1
1	0	0	0	1	8	1	0	0	0
2	0	0	1	0	7	0	1	1	1
3	0	0	1	1	6	0	1	1	0
4	0	1	0	0	5	0	1	0	1
5	0	1	0	1	4	0	1	0	0
6	0	1	1	0	3	0	0	1	1
7	0	1	1	1	2	0	0	1	0
8	1	0	0	0	1	0	0	0	1
9	1	0	0	1	0	0	0	0	0
Códigos ilegais de entrada de BCD									
11	1	0	1	0	7	0	1	1	1
10	1	0	1	1	6	0	1	1	0
12	1	1	0	0	5	0	1	0	1
13	1	1	1	1	4	0	1	0	0
14	1	1	0	0	3	0	0	1	1
15	1	1	1	1	2	0	0	1	0

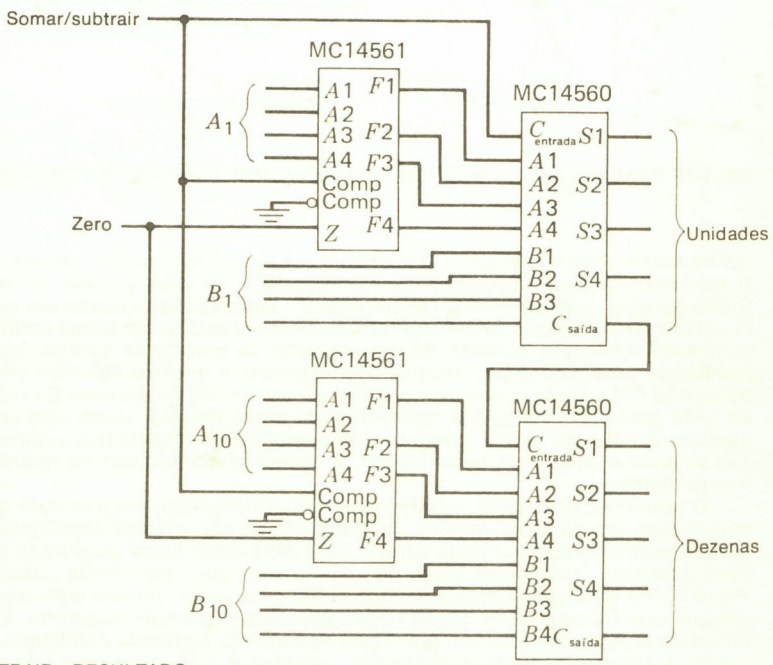
(*b*)

Fig. 6.13 Complementos de 9 em embalagem de CI. *(a)* Diagrama lógico; *(b)* tabela da verdade.

grupos de código dos números BCD, um diagrama-bloco de um circuito lógico é mostrado na Fig. 6.12 e formará o complemento de 9 de um grupo de código que representa um número decimal na forma BCD 8421. Há quatro entradas no circuito, X_1, X_2, X_4 e X_8. Cada uma dessas entradas leva um peso diferente: X_1 tem peso 1, X_2 tem peso 2, X_4 tem peso 4 e X_8 tem peso 8. Se as entradas representarem um dígito decimal do número a ser complementado, as saídas representarão o complemento de 9 do dígito de entrada. Por exemplo, se a entrada for 0010 (2 em decimal), a saída será 0111 (7 em decimal), o complemento de 9 da entrada.

A Fig. 6.13 mostra um complementador de 9 completo numa embalagem de CI. Os circuitos usados são CMOS, e, portanto, portas† de transmissão aparecem no diagrama-bloco bem como portas convencionais. (Isto não incomoda o usuário do circuito porque ele está principalmente interessado na função do circuito.) Quando a entrada COMP for um 1, as saídas *F*1-*F*4 representam o complemento do dígito em *A*1-*A*4. Enquanto que, se COMP for um 0, as entradas *A*1-*A*4 são simplesmente colocadas em *F*1-*F*4 sem alteração.

Interligando-se as embalagens de CI nas Figs. 6.11 e 6.13, pode ser formado um somador-subtrator de BCD conforme indica a Fig. 6.14. Isto mostra que uma embalagem de CI de somador-subtrator de dois dígitos deve ser necessária para mais dígitos. A fim de adicionar os dígitos nas entradas, a entrada ADD-SUBTRACT deve ser feita um 1; para



ZERO	SOMAR/SUBTRAIR	RESULTADO
0	0	*B* mais *A*
0	1	*B* menos *A*
1	*X*	*B*

X = *don't care* = Não-especificada

Fig. 6.14 Circuito de adição-subtração paralelo (complemento de 10).

†As portas de transmissão podem ser explicadas em termos de portas convencionais. A Fig. 5.30 explica as portas de transmissão.

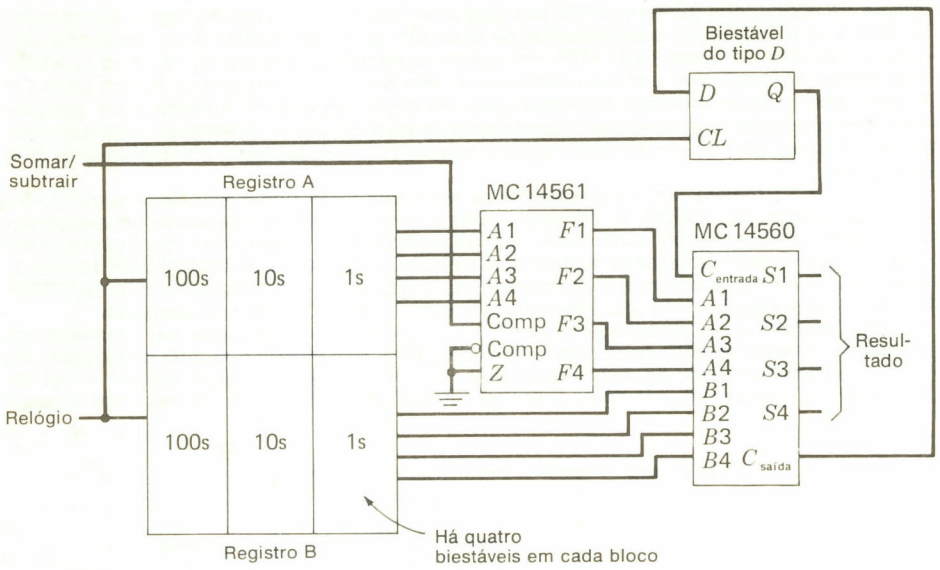


Fig. 6.15 Somador-subtrator de BCD série-paralelo usando registro de deslocamento.

subtrair, este sinal é feito um 0. (Fazendo a entrada ZERO um 1, faremos com que o valor de *B* se transmita sem alteração.) Os números BCD também podem ser representados em forma paralela, conforme mostramos, mas é freqüentemente usado um modo de operação chamado *série-paralelo*. Se um número decimal for escrito em forma codificada em binário, o número resultante consiste de um conjunto de grupos de código, cada um dos quais representa um único dígito decimal. Por exemplo, o decimal 463 num código BCD 8421 é 0100 0110 0011. Cada grupo de 4 bits representa um dígito decimal. É conveniente manipular cada grupo de código que representa um dígito decimal como uma unidade, isto é, em paralelo. Ao mesmo tempo, como os comprimentos da palavra nos computadores decimais são sujeitos a serem um tanto longos, é desejável economizar na quantidade de equipamento utilizado.

O sistema *série-paralelo* proporciona um compromisso em que cada grupo de código é manipulado em paralelo, mas os dígitos decimais são manipulados seqüencialmente. Isto requer quatro linhas para cada caráter BCD 8421, cada linha de entrada das quais leva um peso diferente. O diagrama-bloco de um somador que opera neste sistema é mostrado na Fig. 6.15. Há dois conjuntos de entradas para o somador; um consiste das quatro linhas de entrada que transportam o dígito codificado para o primeiro somando, e as outras quatro linhas de entrada transportam um dígito de segundo somando codificado. Os conjuntos de entradas chegam seqüencialmente dos registros *A* e *B*, cada uma dos quais consiste de quatro registros de deslocamento; chegam primeiro os dígitos menos significativos do primeiro somando e decimal-codificado-em-binário do segundo somando, seguidos pelos dígitos mais significativos decimais.

Se for usado o código 8421, representemos 324 como o segundo somando e 238 como primeiro somando. O sinal ADD será um 0. O somador receberá de início 0100 nas linhas do segundo somando e, ao mesmo tempo, receberá 1000 nas linhas do primeiro somando. Após o primeiro pulso de relógio, essas entradas serão substituídas por 0010 nas linhas do segundo somando e por 0011 nas linhas do primeiro somando. Antes do primeiro sinal de relógio, as linhas de soma devem conter 0010, e, antes do segundo, 0110. Será gerado um transporte durante a adição dos dois primeiros dígitos; este transporte será retardado e adicionado na utilização do biestável do tipo *D*. O processo continuará até que cada um dos três dígitos tenha sido adicionado. Para subtrair *B* de *A*, temos apenas que fazer a entrada ADD-SUBTRACT um 1 e depois aplicar os relógios.

6.14 A OPERAÇÃO DE DESLOCAMENTO

Uma operação de DESLOCAMENTO é uma operação que movimenta os dígitos armazenados num registro para novas posições no registro. Há duas operações de DESLOCAMENTO distintas, uma operação de DESLOCAMENTO-À ESQUERDA e uma operação de DESLOCAMENTO-À DIREITA. Uma operação de DESLOCAMENTO-À ESQUERDA movimenta cada bit de informação armazenado num registro para a esquerda por algum número especificado de dígitos. Consideremos os 6 dígitos binários seguintes, 000110, que admitiremos estar armazenados num registro binário paralelo. Se os conteúdos do registro forem deslocados 1 para a esquerda, após o deslocamento, o registro conterá 001100. Se for efetuado um deslocamento à direita de 1 na palavra 000110, após o deslocamento, o registro conterá 000011. O processo do deslocamento num registro decimal é idêntico: se o registro contiver 0,01234, após um deslocamento de 1 à direita o registro conterá 0,00123 ou, após um deslocamento de 1 à esquerda, o registro conterá 0,12340. A operação de DESLOCAMENTO é usada nas instruções de MULTIPLICAR e de DIVIDIR da maioria das máquinas e também é fornecida como uma instrução que pode ser usada pelos programadores. Por exemplo, uma máquina pode ter instruções SHR e SHL, onde as letras representam em forma mnemônica a ordem das instruções DESLOCAMENTO-À DIREITA e DESLOCAMENTO-À ESQUERDA.

Um diagrama-bloco do conjunto de circuitos lógicos de um único estágio (biestável) num registro que pode ser deslocado ou à esquerda ou à direita é mostrado na Fig. 6.16. Como pode ser visto, o bit à esquerda é deslocado para dentro de X como se DESLOCAMENTO-À DIREITA fosse um 1, e o bit à direita é deslocado para dentro de X quando DESLOCAMENTO-À ESQUERDA for um 1.

A Fig. 6.17 mostra uma embalagem MIS que contém quatro biestáveis e um conjunto de circuitos lógicos de modo que o registro possa ser deslocado à direita ou à esquerda, e também de modo que os quatro biestáveis possam ser carregados em paralelo a partir das quatro linhas de entrada W, X, Y e Z. Os circuitos são circuitos TTL e estão sincronizados em paralelo. Combinando módulos tais como este, pode ser formado um registro de um comprimento escolhido e que pode ser deslocado à esquerda ou à direita ou carregado em paralelo.

6.15 OPERAÇÕES BÁSICAS

A unidade aritmética e lógica de um computador digital consiste de um certo número de registros em que a informação pode ser armazenada e um conjunto de circuitos lógicos que torna possível efetuar certas operações com a informação armazenada nos registros e efetuar certas operações entre os registros.

Conforme vimos, os dados armazenados num certo registro de biestáveis podem ser operados das seguintes maneiras:

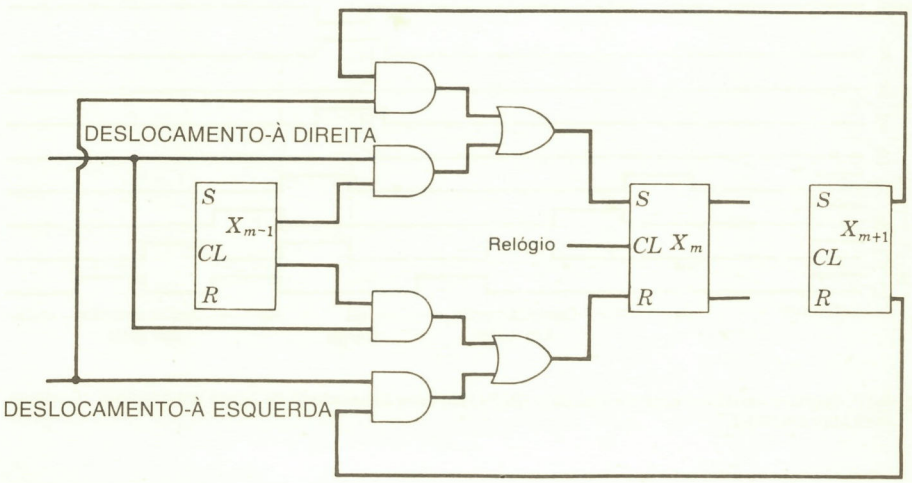


Fig. 6.16 Estágio de registro com deslocamento-à esquerda e deslocamento-à direita.

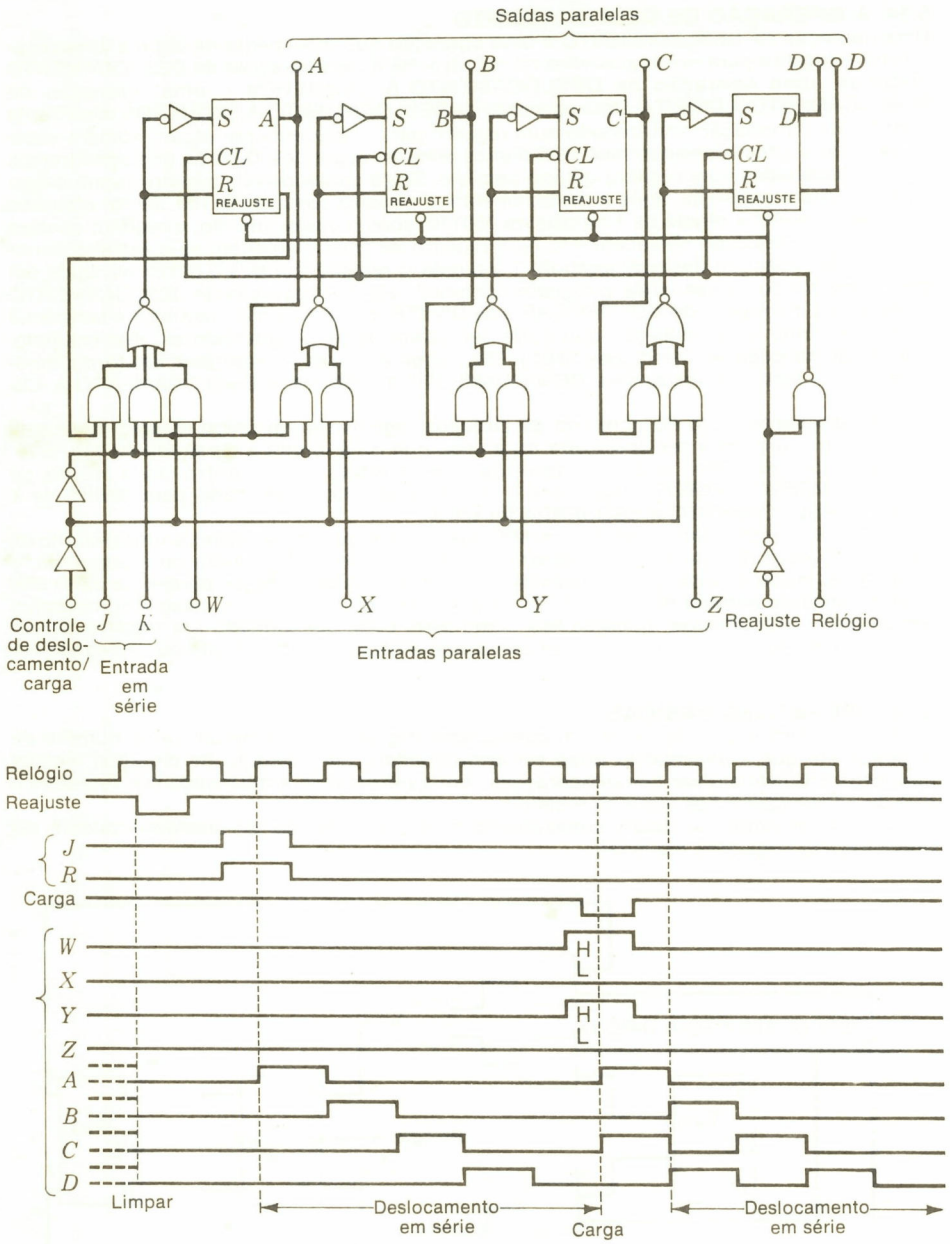


Fig. 6.17 Registro de deslocamento (modelo SN 74195) com capacidade de carga em paralelo. (Cortesia da Texas Instruments.)

1. O registro pode ser reajustado em todos os 0.
2. Os conteúdos de um registro podem ser complementados ou em forma de complementos de 1 ou de 2 em binário, ou em forma de complementos de 9 ou de 10 em decimal.
3. Os conteúdos de um registro podem ser deslocados à direita ou à esquerda.
4. Os conteúdos de um registro podem ser incrementados ou decrementados.

Também foram descritas várias operações entre registros. Estas incluem:

1. Transferência dos conteúdos de um registro para um outro registro.
2. Adicionar a ou subtrair de os conteúdos de um registro dos conteúdos de um outro registro.

A maioria das operações aritméticas que uma unidade aritmética e lógica efetua consiste destes dois tipos de operações ou de conjuntos em seqüência destes dois tipos de operações. Instruções complicadas, tais como as de multiplicação e divisão, podem requerer um grande número destas operações, mas estas instruções podem ser efetuadas usando apenas seqüências das operações simples já descritas.

Um outro ponto importante necessita ser destacado. Certas operações que ocorrem dentro das instruções são *condicionais*. Isto significa que uma certa operação pode ou não ocorrer, dependendo do valor de determinados bits dos números armazenados. Por exemplo, pode ser desejável multiplicar usando apenas números positivos. Neste caso, os bits de sinal dos dois números a serem multiplicados serão examinados pelo conjunto de circuitos de controle, e, se qualquer deles for um 1, o número correspondente será complementado antes que a multiplicação comece. Esta operação, complementação do registro, é uma operação condicional.

Muitas seqüências diferentes de operações podem produzir o mesmo resultado. Por exemplo, dois números podem ser multiplicados adicionando-se simplesmente o multiplicando a si mesmo pelo número de vezes indicado pelo multiplicador. Se isto fosse feito com lápis e papel, 369×12 seria efetuado adicionando-se 369 a si próprio 12 vezes. Isto seria um processo laborioso comparado com o algoritmo que desenvolvemos para a multiplicação, mas obteríamos o mesmo resultado. O mesmo princípio se aplica à multiplicação com máquina. Dois números podem ser multiplicados transferindo-se um dos números para dentro de um contador que contasse para baixo cada vez que uma adição fosse efetuada, e depois adicionando o outro número a si próprio até que o contador alcançasse o zero. Esta técnica tem sido usada, mas técnicas muito mais rápidas são também usadas e serão explicadas.

Muitos algoritmos têm sido usados para multiplicar e dividir números em máquinas digitais. A divisão, especialmente, é um processo complicado e, em máquinas decimais em particular, muitas técnicas diferentes são usadas. A técnica particular usada por uma máquina é geralmente baseada no custo dessa máquina e no incentivo pela velocidade da máquina. Como em quase todas as operações, a velocidade é dispendiosa, e um processo de divisão mais rápido geralmente significa uma máquina mais cara.

Para explicar as operações da multiplicação e da divisão binárias, usaremos um diagrama-bloco de uma máquina binária generalizada. A Fig. 6.18 ilustra, em forma de diagrama-bloco, os registros de uma UAL. A máquina tem três registros básicos, um acumulador, registro *Y* e registro *B*. As operações que podem ser efetuadas foram descritas:

1. O acumulador pode ser limpo (ou reajustado).
2. Os conteúdos do acumulador podem ser deslocados à direita ou à esquerda. Além disso, o acumulador e o registro *B* podem ser formados num longo registro de deslocamento. Se em seguida deslocarmos este registro dois dígitos para a direita, os dois dígitos menos significativos do acumulador serão deslocados para dentro dos dois primeiros lugares do registro *B*. Vários deslocamentos à esquerda deslocarão os dígitos mais significativos do registro *B* para dentro do acumulador. Já que há 5 bits na palavra básica da máquina, há cinco dispositivos binários de armazenamento em cada registro. Um deslocamento de cinco lugares para a direita transferirá os conteúdos do acumulador para dentro do registro *B*, e um deslocamento de cinco lugares para a esquerda deslocará os conteúdos do registro *B* para dentro do acumulador.
3. Os conteúdos do registro *Y* podem ser ou adicionados a ou subtraídos do acumulador. A soma ou a diferença será então armazenada no registro do acumulador.
4. As palavras provenientes da memória podem ser lidas no registro *Y*. Para ler uma palavra no acumulador, é necessário primeiro limpar o acumulador, em seguida ler a palavra da memória no registro *Y* e depois adicionar o registro *Y* ao acumulador.

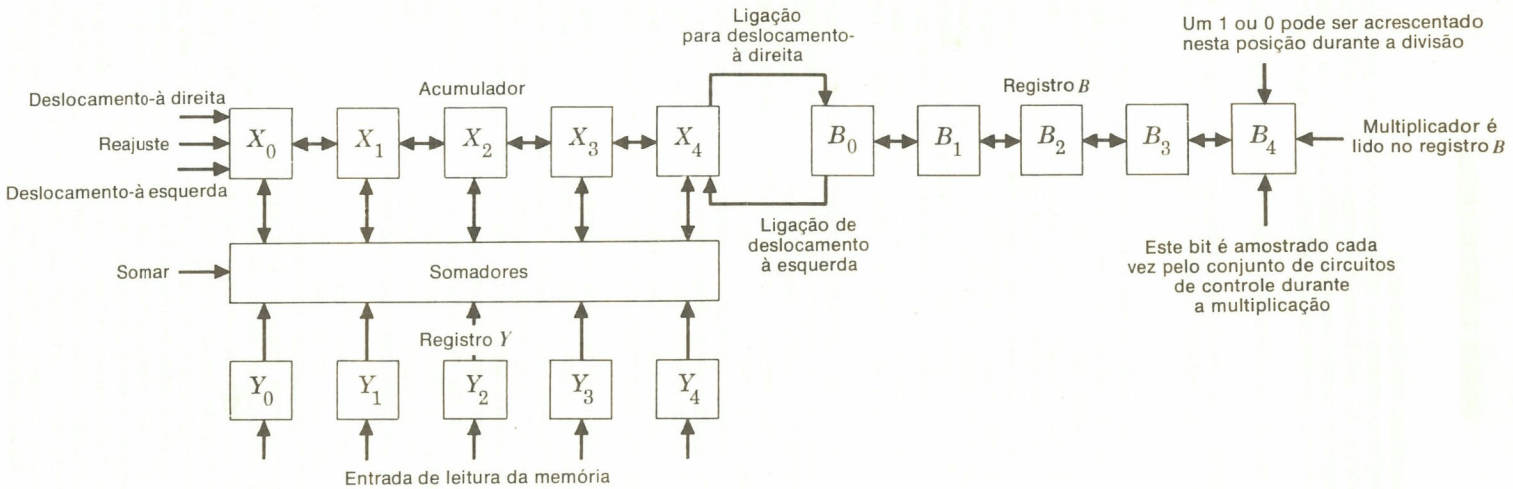


Fig. 6.18 Elemento aritmético paralelo generalizado.

Um elemento aritmético que possa efetuar estas operações com seus registros pode ser posto em seqüência para efetuar todas as operações aritméticas. É, de fato, possível construir uma máquina usando menor número de operações do que estas, mas a maioria dos computadores de propósitos gerais terá usualmente um elemento aritmético no mínimo com estas capacidades.

6.16 MULTIPLICAÇÃO BINÁRIA

O processo da multiplicação de números binários pode ser melhor examinado quando se escreve a multiplicação de dois números binários:

```
  1 0 0 1 = multiplicando
  1 1 0 1 = multiplicador
  -------
    1 0 0 1  \
  0 0 0 0     } produtos parciais
1 0 0 1      /
1 0 0 1     /
-----------
1 1 1 0 1 0 1 = produto
```

O fato importante a observar neste processo é que há realmente apenas duas regras para a multiplicação de um único *número* binário por um *dígito* binário; (1) se o dígito do multiplicador for um 1, o multiplicando é simplesmente copiado e (2), se o dígito do multiplicador for um 0, o produto é 0. O exemplo acima ilustra estas regras como segue: O primeiro dígito à direita do multiplicador é um 1; portanto, o multiplicando é copiado como o primeiro produto parcial. O próximo dígito do multiplicador à esquerda é um 0; portanto, o produto parcial é um 0. Observemos que cada vez que é formado um produto parcial, ele é deslocado de um lugar para a esquerda do produto parcial anterior. Mesmo se o produto parcial for um 0, o próximo produto parcial é deslocado de um lugar à esquerda do produto parcial anterior. Este processo prossegue até que todos os dígitos do multiplicador tenham sido usados, e então os produtos parciais são somados.

As três operações que o computador deve ser capaz de efetuar para multiplicar desta maneira são, portanto, (1) a capacidade de perceber se um bit do multiplicador é ou um 1 ou um 0, (2) a capacidade de deslocar produtos parciais e (3) a capacidade de adicionar os produtos parciais.

Não é necessário esperar até que todos os produtos parciais tenham sido formados antes de somá-los. Eles podem ser somados dois de cada vez. Por exemplo, começando com os dois primeiros produtos parciais no exemplo acima,

```
  1 0 0 1
0 0 0 0
-------
0 1 0 0 1
```

O próximo produto parcial pode então ser adicionado a esta soma, deslocando-a de uma posição para a esquerda:

```
  0 1 0 0 1
1 0 0 1
---------
1 0 1 1 0 1
```

e, finalmente,

```
  1 0 1 1 0 1
1 0 0 1
-----------
1 1 1 0 1 0 1
```

Um multiplicador pode ser construído exatamente desta maneira. Amostrando-se cada bit do multiplicador alternativamente, adicionando-se o multiplicando em algum registro e, em seguida, deslocando-se o multiplicando para a esquerda cada vez que um novo bit do multiplicador for amostrado, deve ser formado um produto da soma dos produtos parciais. De fato, o processo da multiplicação na maioria das máquinas binárias é efetuado de maneira muito semelhante a esta.

Para examinar a técnica típica da multiplicação, os elementos aritméticos generalizados na Fig. 6.18 serão usados. Com referência à Fig. 6.18, armazenemos o multiplicador no registro *B* e o multiplicando no registro *Y;* o acumulador contém todos os dígitos 0. Admitamos também que tanto o multiplicador quanto o multiplicando sejam positivos; se qualquer um deles for negativo, deve ser convertido à forma positiva antes que a multiplicação comece. Admitamos *n* bits em cada operando.

Uma multiplicação requer *n* etapas básicas. Cada etapa é iniciada pelo conjunto de circuitos de controle examinando o bit mais à direita no registro *B.* A etapa básica é depois efetuada como segue. Se o bit mais à direita no registro *B* for um 0, o combinado acumulador e registro *B* é deslocado de um lugar à direita; se o bit mais à direita no registro *B* for um 1, o número no registro *Y* é adicionado aos conteúdos do acumulador, e o combinado acumulador e registro *B* é em seguida deslocado de um lugar à direita. O novo bit mais à direita do registro *B* é então novamente examinado, e a próxima das *n* etapas é iniciada.

Consideremos a mesma multiplicação que foi usada no exemplo anterior, isto é, 1101 × 1001, onde 1101 é o multiplicador. Neste caso, o acumulador contém 0,0000, o registro *B,* 0,1101 e o registro *Y,* 0,1001.

Uma vez que o bit mais à direita do registro *B* contém um 1 (o bit menos significativo do multiplicador), durante a primeira etapa, os conteúdos do registro *Y* são adicionados ao acumulador, e o combinado acumulador e registro *B* são em seguida deslocados para a direita. O segundo bit menos significativo do multiplicador ocupará agora o bit mais à direita do registro *B* e controlará a próxima operação.

O registro *Y* ainda conterá o multiplicando 0,1001, os conteúdos do acumulador serão 0,0100 e os conteúdos do registro *B* serão 1,0110. O bit mais à direita do registro *B* é um 0, e já que ele controla a próxima operação, um sinal de DESLOCAMENTO-À DIREITA será iniciado e o acumulador e o registro *B* serão deslocados para a direita, dando 0,0010 no acumulador e 0,1011 no registro *B.* Um 1 estará agora no bit mais à direita do registro *B.*

O registro *Y* será, portanto, novamente adicionado ao acumulador, e o combinado acumulador-registro *B* será novamente deslocado para a direita. O bit menos significativo do registro *B* será um outro 1; portanto, o registro *Y* será novamente adicionado ao acumulador, e o acumulador deslocado para a direita.

Após o deslocamento para a direita referido acima, o combinado acumulador e registro *B* conterá 0,011101010. Um deslocamento final para a direita dá 0,001110101, o produto correto de nosso sistema de números inteiros. Os dígitos mais significativos serão armazenados no acumulador, e os dígitos menos significativos, no registro *B.*

Agora pode ser vista a razão para o combinado acumulador e registro *B.* O produto de dois números de 5 bits com sinal pode conter até nove dígitos significativos (inclusive o bit de sinal), e, portanto, dois registros de 5 bits, e não um, são necessários para reter o produto. O resultado final é tratado como um número de 10 bits estendendo-se através dos dois registros com o bit mais à esquerda no registro esquerdo, o bit de sinal e o bit mais à direita no registro direito e o dígito binário menos significativo no produto. Assim, nosso resultado nos dois registros combinados foi 0,001110101, que é +117 em decimal.

O conjunto de circuitos de controle é projetado para efetuar o exame dos bits do multiplicador, depois ou desloca ou adiciona e desloca no número correto de vezes, e depois pára. Neste caso, o comprimento do multiplicador, ou registro *Y,* era de 4 bits mais um bit de sinal; portanto, são realizadas quatro de tais etapas. A prática geral consiste em examinar cada bit da palavra da máquina, exceto o bit de sinal, alternadamente. Por exemplo, se a palavra básica da máquina for 25 bits, isto é, 24 bits em que a magnitude de um número é armazenada mais um bit de sinal, cada vez que uma multiplicação é efetuada, a máquina examinará 24 bits, cada um na sua vez, efetuando a operação SOMAR-E-DESLOCAR ou apenas a operação DESLOCAR durante 24 vezes. Outras tantas sejam vistas, isto torna a operação de multiplicação mais longa do que operações tais como soma ou subtração. Algumas máquinas paralelas dobram sua velocidade normal de operação durante a multiplicação: se a máquina efetuar operações tais como adição, complementação, transferências etc., numa velocidade de 4 MHz por segundo nas instruções ordinárias, a velocidade será aumentada para 8 MHz nas combinações somar-e-deslocar efetuadas durante a multiplicação. Algumas máquinas são capazes de deslocar para a direita enquanto estão somando; assim, a soma do acumulador e do registro *Y* aparece deslocada de um lugar à

direita de cada vez, e a operação DESLOCAMENTO-À DIREITA após cada adição pode ser omitida.

Os bits de sinal do multiplicador e do multiplicando podem ser manipulados de diversas maneiras. Por exemplo, o sinal do produto pode ser determinado por meio do conjunto de circuitos de controle antes de ser iniciado o procedimento da multiplicação. Este bit de sinal é armazenado durante o processo da multiplicação, após o qual ele é colocado no bit de sinal do acumulador, e o acumulador é então complementado, se necessário. Portanto, os bits de sinal do multiplicador e do multiplicando são examinados primeiro; se eles forem ambos 0, o sinal do produto deve ser 0; se ambos forem 1, o sinal do produto deve ser 0; e se qualquer um, mas não ambos, for 1, o sinal do produto deve ser 1. Esta informação, retida num biestável enquanto a multiplicação estiver ocorrendo, pode ser transferida posteriormente para dentro do bit de sinal. Se a máquina manipula números no sistema de complemento de 1 ou de 2, tanto o multiplicador quanto o multiplicando podem ser manipulados como grandezas positivas durante a multiplicação, e, se o sinal de qualquer número for negativo, o número é complementado para uma grandeza positiva antes que comece a multiplicação. Às vezes, a multiplicação é efetuada com números complementados usando-se algoritmos mais complicados. Estes são descritos nas referências.

6.17 MULTIPLICAÇÃO DECIMAL

A multiplicação decimal é um processo mais complicado do que a multiplicação binária. Enquanto o produto de um dígito binário por um número binário é ou o número ou 0, o produto de um dígito decimal por um número decimal envolve o uso de uma tabela de multiplicação transportando e adicionando mais. Por exemplo,

$$7 \times 24 = (7 \times 4) + (7 \times 20) = 28 + 140 = 168$$

Até a multiplicação de dois dígitos decimais podem envolver dois dígitos de saída, por exemplo, 7×8 é igual a 56. Na discussão seguinte, os dois dígitos que podem resultar quando um dígito decimal for multiplicado por um dígito decimal são denominados *dígitos de mão direita e de mão esquerda*. Assim, para 3×6, temos 1 para o dígito da mão esquerda e 8 para o dígito da mão direita. Para 2×3, temos 0 para o dígito da mão esquerda e 6 para o dígito da mão direita.

Exceto no caso de adicionar unicamente o multiplicando a si próprio, o número de vezes indicado pelo multiplicador, um processo fácil mas que consome tempo, o método mais simples para a multiplicação decimal consiste em carregar o dígito mais à direita do multiplicador para dentro de um contador que conta para baixo e depois adicionar o multiplicando a si próprio e simultaneamente indexar o contador até que ele alcance 0. O produto parcial assim formado pode ser deslocado de um dígito decimal para a direita, o próximo dígito do multiplicador carregado para dentro do contador e o processo repetido até que todos os dígitos do multiplicador tenham sido usados. Esta é uma técnica relativamente lenta, porém direta.

O processo pode ser acelerado formando produtos que usem o multiplicando e o dígito mais à direita do multiplicador como no esquema anterior, mas formando realmente os produtos parciais de mão direita e de mão esquerda obtidos quando da multiplicação de um dígito por um número e somando-os em seguida. Por exemplo, 6×7164 forneceria 2664 para os dígitos do produto da mão direita e 4032 para os dígitos do produto da mão esquerda. A soma então seria

$$\begin{array}{r} 2664 \\ +4032 \\ \hline 42984 \end{array}$$

A multiplicação em máquina decimal é, em geral, um processo complicado se se deseja velocidade, e há quase tantas técnicas para multiplicação de números BCD quanto há tipos de máquinas.† As embalagens de CI são produzidas contendo uma rede de circuitos lógicos que tem dois caracteres BCD como entradas que produzem a saída de dois dígitos requerida. Os exercícios e as referências exploram este assunto com mais detalhes.

†Várias máquinas usam as técnicas de pesquisa em tabela para formação de produtos, onde o produto de cada par de dígitos é armazenado na memória.

*6.18 DIVISÃO

A operação de divisão é a mais difícil e a mais demorada que a UAL da maioria das máquinas de propósitos gerais pode efetuar. Embora a divisão possa parecer mais fácil do que a multiplicação, há vários problemas em conexão com o processo da divisão que introduzem etapas extras que consomem tempo.

A divisão, com emprego de lápis e papel, é um processo de tentativas. Por exemplo, se desejamos dividir 77 por 4610, observamos primeiro que 77 não "caberá" em 46; portanto, tentamos dividir 77 por 461. Podemos estimar que 77 "caberá" 6 vezes, no entanto:

```
        6
77 ) 4610
     462
    ----
      -1
```

Portanto, fizemos uma estimativa muito alta e devemos reduzir o primeiro dígito do quociente, que desenvolveremos, a 5.

O mesmo problema enfrenta o computador quando ele tenta dividir desta maneira. Ele deve "tentar" uma subtração a cada etapa do processo e depois verificar se o resto é negativo. Consideremos a divisão de 1111 por 11.

```
      101
11 ) 1111
     11
    ----
     0011
       11
    ----
       00
```

É fácil determinar visualmente em qualquer etapa do processo se o quociente deve ser um 1 ou um 0, mas o computador não pode determinar isto sem fazer, de cada vez, uma subtração por tentativa. Depois de ter sido tentado um quociente e o divisor subtraído, se o resultado for negativo, ou o dividendo normal deverá ser "rearmazenado" ou utilizada uma outra técnica de divisão.

Há diversos pontos a serem observados concernentes à divisão binária de valor inteiro de ponto fixo. A divisão é geralmente efetuada com dois inteiros binários com sinal e do mesmo comprimento fixo. O resultado, ou o quociente, é armazenado com um número, com tantos dígitos quanto o divisor ou dividendo, e o resto é também armazenado como um número do mesmo comprimento.†

Usando os registros indicados na Fig. 6.18, mostraremos como dividir um número armazenado no acumulador por um número no registro *Y*. O quociente é então armazenado no registro *B*, e o resto no acumulador. Este é o formato de divisão mais comum.

Admitamos que os registros *B* e *Y* na Fig. 6.18 sejam de 5 bits de comprimento (4 bits + bit de sinal) e o acumulador também seja de 5 bits de comprimento. Antes de iniciar o procedimento, o dividendo é lido no acumulador, e o divisor, no registro *Y*. Depois da divisão, o quociente será armazenado no registro *B* e o resto será no acumulador. Tanto o divisor quanto o dividendo devem ser positivos.

Há duas técnicas gerais de divisão em máquinas binárias: as técnicas de *rearmazenamento* e *sem armazenamento*. Nosso exemplo usará a técnica de restabelecimento.

A *etapa básica* na técnica de restauração consiste de uma divisão tentativa que é feita procurando subtrair o registro *Y* do acumulador.

1. Se o resultado for negativo, o divisor não "caberá"; portanto, um 0 é colocado no bit mais à direita do registro *B*, e o dividendo (acumulador) é rearmazenado adicionando-se o divisor ao resultado da subtração. O combinado registro *B* e acumulador será então deslocado para a esquerda.

*Esta seção pode ser omitida numa primeira leitura.

†Se dividirmos um inteiro por outro, o quociente e o resto serão ambos inteiros. A regra é como segue: se a for o dividendo, y o divisor, b o quociente e r o resto, então $a = y \times b + r$.

2. Se o resultado de uma divisão tentativa for positivo ou 0, não haverá necessidade de rearmazenar o dividendo parcial no acumulador, pois a divisão tentativa foi bem-sucedida. Um 1 é colocado no bit mais à direita do registro *B*, e o acumulador e o registro *B* são ambos deslocados para a esquerda.

O computador determina se ou não o resultado de uma divisão tentativa é positivo ou negativo, examinando o bit de sinal do acumulador após cada subtração.

A fim de demonstrar todo o procedimento, é necessário primeiro explicar como iniciar a divisão e como começar e parar a execução das etapas básicas.

1. Conforme descrito acima, se o divisor for maior do que o dividendo, então o quociente deve ser 0, e o resto é o valor do dividendo. (Por exemplo, se tentamos dividir 7 por 17, o quociente é 0 e o resto é 7.) Para testar isto, o dividendo em *Y* pode ser subtraído do acumulador e, se o resultado for negativo, tudo o que resta é rearmazenar o acumulador adicionando o registro *Y* ao acumulador. O registro *B* agora tem valor 0 que está certo para o quociente, e o acumulador tem o valor original que é o resto.
2. Após ser feito o teste acima, é necessário alinhar o bit 1 mais à esquerda no divisor com o bit 1 mais à esquerda no dividendo, deslocando o divisor para a esquerda, e registrar o número de deslocamentos necessário para fazer este alinhamento. Se o número de deslocamentos for *M*, então a etapa básica deve ser realizada *M* + 1 vezes.†
3. Agora, a etapa básica é realizada as *M* + 1 vezes necessárias.
4. Finalmente, para ajustar o resto, o acumulador deve ser deslocado para a direita *M* + 1 vezes após ser realizada a última etapa básica. Nos Quadros 6.2 e 6.3 são mostrados exemplos. A etapa 1 acima, que testa um quociente zero, não é mostrada nos dois exemplos.

QUADRO 6.2

Registro *B*	Acumulador	Registro *Y*	Observações
0,0000	0,0110	0,0011	Dividimos 6 por 3.
0,0000	0,0110	0,0110	O registro *Y* é deslocado à esquerda uma vez, alinhando 1 nos registros *Y* do acumulador; a etapa básica deve ser executada duas vezes.
0,0000	0,0000	0,0110	O registro *Y* foi subtraído do acumulador. O resultado é 0, e, portanto, o registro *B* e o acumulador são deslocados à esquerda e um 1 colocado no bit mais à direita do registro *B*.
0,0001	0,0000	0,0110	
0,0001	1,1010	0,0110	O registro *Y* é subtraído do acumulador.
0,0010	0,0000	0,0110	O registro *Y* é adicionado ao acumulador, e o registro *B* e o acumulador são deslocados de 1 à esquerda. Um 0 é colocado no último bit do registro *B*.
0,0010	0,0000	0,0110	O acumulador deve agora ser deslocado à direita duas vezes, mas ele é 0 e, portanto, não resulta nenhuma mudança.

†Isto pode ser conseguido fazendo *Y* um registro de deslocamento e providenciando um contador para contar os deslocamentos até que o primeiro bit 1 de *Y* esteja alinhado com o bit 1 do acumulador. Tanto o acumulador quanto *Y* podem ser deslocados para a esquerda até que haja um bit 1 em sua primeira posição, mas o resto terá então que ser ajustado movendo-se mesmo para a direita no acumulador.

QUADRO 6.3

Registro *B*	Acumulador	Registro *Y*	Observações
0.0000	0.1101	0.0011	Dividimos 13 por 3.
0.0000	0.1101	0.0110	Deslocar registro *Y* à esquerda.
0.0000	0.1101	0.1100	Deslocar registro *Y* à esquerda. Os bits 1 mais à esquerda no acumulador e no registro *Y* são alinhados. A etapa básica será executada três vezes.
0.0000	0.0001	0.1100	Registro *Y* foi subtraído do acumulador. Resultado é positivo.
0.0001	0.0010	0.1100	Registro *B*-acumulador deslocado à esquerda e 1 colocado no registro *B*.
0.0001	1.0110	0.1100	Registro *Y* é subtraído do acumulador. Resultado é negativo.
0.0010	0.0100	0.1100	Registro *Y* é somado ao acumulador. Ambos são então deslocados à esquerda e um 0 colocado no registro *B*.
0.0010	1.1000	0.1100	Registro *Y* é subtraído do acumulador. Resultado é negativo.
0.0100	0.1000	0.1100	Registro *Y* é adicionado ao acumulador. O acumulador e o registro *B* são deslocados à esquerda. Um 0 é colocado no bit do registro *B*.
0.0100	0.0001	0.1100	O acumulador foi deslocado à direita de 3. O quociente é 4 e o resto é 1.

A Fig. 6.19 mostra um fluxograma do algoritmo. Os fluxogramas são freqüentemente usados para representar algoritmos. Um fluxograma com maiores detalhes deve separar algumas das etapas, tal como "deslocar o acumulador $M + 1$ vezes para a direita", num único deslocamento efetuado num laço que é controlado por um contador. Muitas vezes, quando os algoritmos são razoavelmente complicados, como é este algoritmo, é conveniente desenhar um fluxograma do algoritmo antes de tentar incrementar o conjunto de circuitos de controle.

Há diversas técnicas para divisão sem rearmazenamento, mas um algoritmo amplamente usado emprega um procedimento em que o divisor é alternadamente subtraído e adicionado.

O princípio envolvido aqui pode ser explicado como segue: Seja X o número armazenado no acumulador e Y o divisor. Se o divisor for subtraído, o resultado é $X - Y$. Se este resultado for armazenado no acumulador e depois deslocado de 1 bit para a esquerda, o resultado será $2(X - Y)$. Se $X - Y$ for negativo, o divisor é agora adicionado ao acumulador, resultando em $2(X - Y) + Y$, ou $2X - Y$, que estará correto para esta etapa. O quociente é formado do mesmo modo que na divisão com rearmazenamento. Se o resultado de uma divisão tentativa for positivo, um 1 é colocado no bit mais à direita do quociente, e, se o resultado de uma divisão tentativa for negativo, o quociente é deixado inalterado. Após cada tentativa, tanto o acumulador (o dividendo parcial) quanto o quociente são deslocados para a esquerda.

As etapas básicas requeridas em cada divisão tentativa são portanto como segue:

1. Na primeira etapa, o registro *Y* é subtraído do acumulador.
2. Se o resultado for positivo ou zero, um 1 é colocado no bit menos significativo do registro *B*, e o acumulador e o registro *B* são deslocados para a esquerda. A próxima etapa será uma outra subtração.

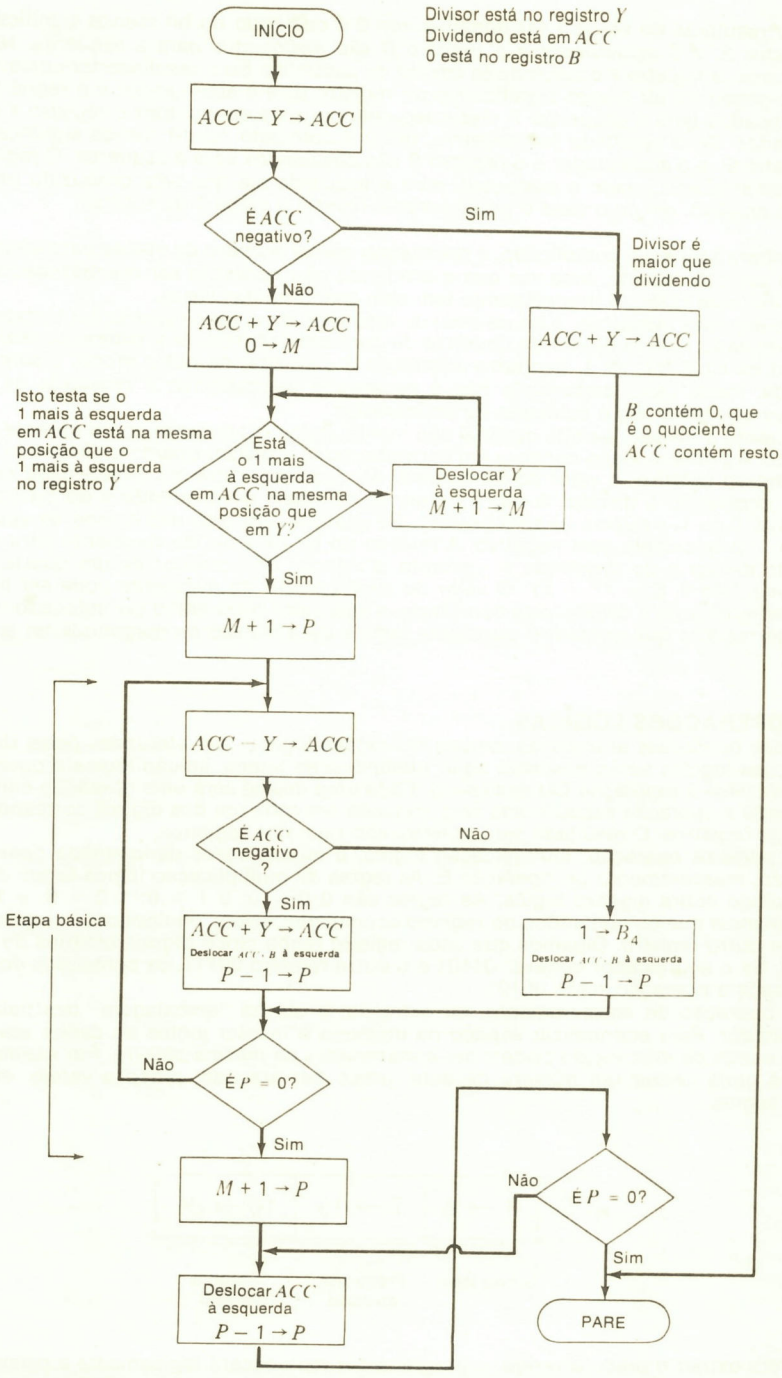


Fig. 6.19 Fluxograma de algoritmo da divisão.

3. Se o resultado da etapa 1 for negativo, um 0 é colocado no bit menos significativo do registro *B*, e o acumulador e o registro *B* são deslocados para a esquerda. Na etapa seguinte, o registro *Y* é adicionado em vez de subtraído. Se o resultado for positivo, um 1 é colocado no bit menos significativo do registro *B*, e o acumulador e o registro *B* são deslocados para a esquerda. A etapa seguinte consiste em subtrair o registro *Y* do acumulador. Se o resultado for negativo, um 0 é colocado no bit menos significativo do registro *B*, e o acumulador e o registro *B* são deslocados para a esquerda. O registro *Y* é novamente adicionado e deslocado para a esquerda até que seja produzido um resultado positivo, em cujo caso o procedimento reverte à etapa 1 de subtrair.

Embora seja mais complicado, é necessário menor número de etapas para efetuar uma divisão por esta técnica, uma vez que o dividendo não necessita ser rearmazenado a cada etapa. Por esta razão, o procedimento tem sido usado muitas vezes.

Em algumas máquinas, o bit de sinal do resultado da divisão (subtração) tentativa pode ser examinado pelo conjunto de circuitos de controle sem efetuar a subtração real. Então, se o bit de sinal for um 1, nenhuma subtração é efetuada; de outro modo, a subtração é efetuada. Neste caso, certamente não é necessário rearmazenar; o resultado da divisão tentativa que falha não é colocado no acumulador.

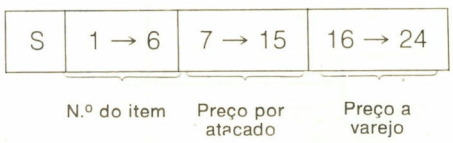
Durante a divisão, os bits de sinal são manipulados exatamente como durante a multiplicação. A primeira etapa consiste em converter tanto o divisor quanto o dividendo à forma de grandeza positiva; o valor do bit de sinal do quociente deve ser armazenado enquanto estiver ocorrendo a divisão. A regra é que, se os sinais do dividendo e do divisor forem ambos ou 0 ou 1, o quociente será positivo. Se qualquer um, mas não ambos, de seus sinais for um 1, o quociente será negativo. A relação do bit de sinal do quociente para o bit de sinal do divisor e do dividendo é, portanto, a relação do somador de um quarto, ou OU exclusivo, isto é, $S = X\bar{Y} + \bar{X}Y$. O valor do sinal correto do quociente pode ser lido num biestável enquanto a divisão está ocorrendo, e este valor pode então ser colocado no bit de sinal do registro que contém o quociente depois de a divisão de magnitude ter sido concluída.

6.19 OPERAÇÕES LÓGICAS

Além das operações aritméticas, muitas operações lógicas são efetuadas pelas UAL. Três operações lógicas serão descritas aqui: multiplicação lógica, adição lógica e operação de *soma na base 2* (operação OU exclusiva). Cada uma destas será uma operação entre registros, onde a operação especificada será efetuada em cada um dos dígitos correspondentes nos dois registros. O resultado será armazenado num dos registros.

A primeira operação, multiplicação lógica, é muitas vezes denominada operação de *extração, mascaramento* ou operação E. As regras da multiplicação lógica foram definidas no capítulo sobre álgebra lógica. As regras são $0 \cdot 0 = 0$; $0 \cdot 1 = 0$; $1 \cdot 0 = 0$; e $1 \cdot 1 = 1$. Suponhamos que os conteúdos do registro acumulador sejam "multiplicados logicamente" por um outro registro. Digamos que cada registro tenha cinco dígitos binários de comprimento. Se o acumulador contiver 01101 e o outro registro 00111, os conteúdos do acumulador após a operação serão 00101.

A operação de mascaramento, ou extração, é útil na "embalagem" das palavras de computador. Para economizar espaço na memória e manter juntos os dados associados, várias partes de informação podem ser armazenadas na mesma palavra. Por exemplo, uma palavra pode conter um número de item, preço por atacado, preço a varejo, embalado como segue:



Para extrair o preço a varejo, o programador multiplicará logicamente a palavra acima por uma palavra que contenha 0 no dígito de sinal até o dígito 15, e com 1 nas posições 16 até 24. Após a operação, apenas o preço a varejo permanecerá na palavra.

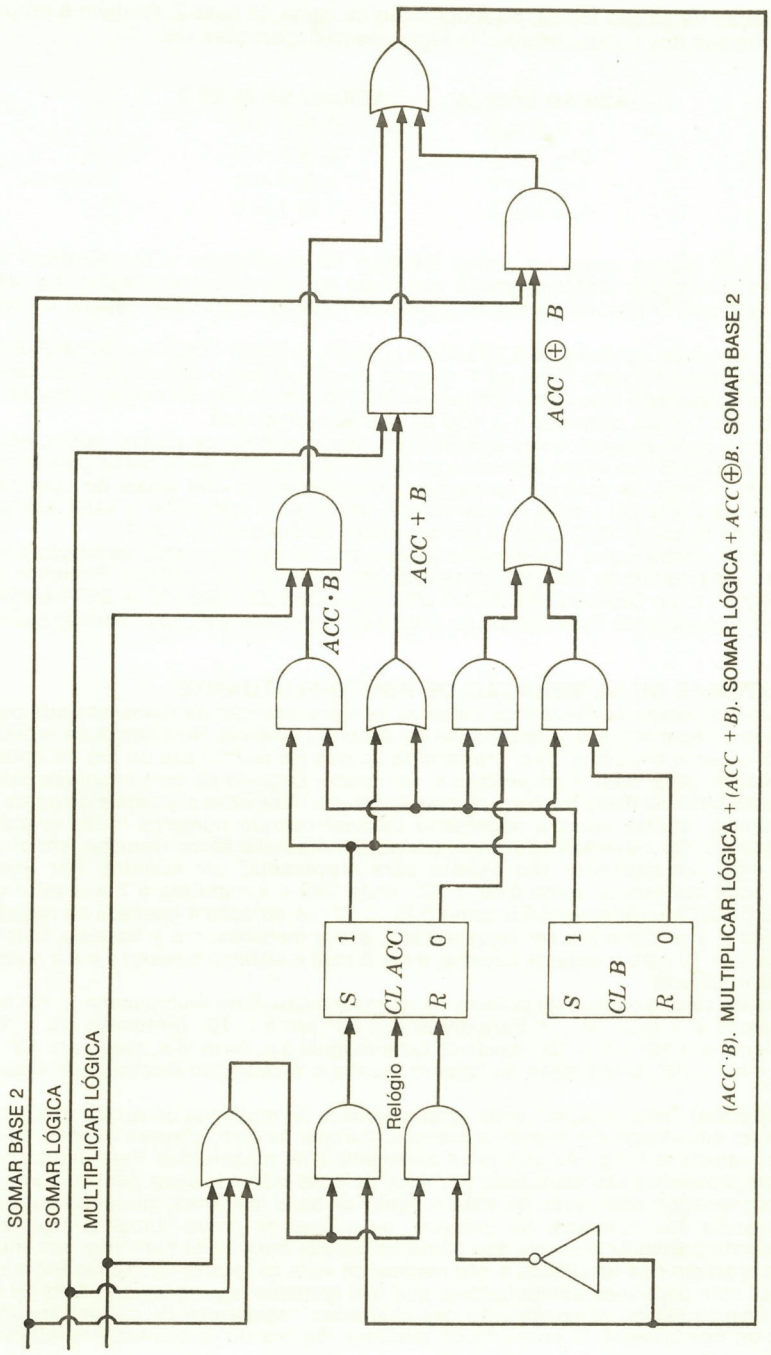


Fig. 6.20 Circuito para comandar operações lógicas no biestável do acumulador.

A operação de adição lógica, ou a operação de soma na base 2, também é proporcionada pela maioria dos computadores. As regras destas operações são:

ADIÇÃO LÓGICA	ADIÇÃO NA BASE 2
$0 + 0 = 0$	$0 \oplus 0 = 0$
$0 + 1 = 1$	$0 \oplus 1 = 1$
$1 + 0 = 1$	$1 \oplus 0 = 1$
$1 + 1 = 1$	$1 \oplus 1 = 0$

A Fig. 6.20 mostra como um único biestável de acumulador e um biestável tipo *B* podem ser comandados juntos de modo que todas as três dessas operações lógicas possam ser efetuadas. O circuito na Fig. 6.20 deve ser repetido para cada estágio do registro acumulador.

Há três sinais de controle, MULTIPLICAR LÓGICA, SOMAR LÓGICA e SOMAR BASE 2. Se um destes sinais for para cima, ou 1, quando chega um pulso de relógio esta operação é efetuada e o resultado colocado no biestável ACC (acumulador). Se nenhum dos sinais de controle for um 1, nada acontece e o ACC permanece como está.

Os valores reais desejados são achados por três conjuntos de portas; assim, ACC·B e ACC + B e ACC $\oplus$ B são todos formados primeiro. Cada um destes é então introduzido na porta E com o sinal de controle apropriado. Finalmente, os três sinais de controle são introduzidos na porta OU juntos, e este sinal é usado para comandar o valor apropriado para dentro do biestável ACC quando um dos sinais de controle for um 1.

A Fig. 6.20 mostra como uma escolha de diversos valores diferentes da função pode ser comandada para dentro de um único biestável usando sinais de controle. Podemos incluir um sinal SOMA e um DESLOCAMENTO-À DIREITA e DESLOCAMENTO-À ESQUERDA simplesmente acrescentando mais portas. No Cap. 9 encontraremos outros informes sobre este assunto.

*6.20 SISTEMAS DE NUMERAÇÃO DE PONTO-FLUTUANTE

As seções precedentes descrevem os sistemas de representação de números onde os inteiros positivos e negativos são armazenados em palavras binárias. No sistema de representação usado, o ponto binário é "fixo" no sentido de que ele permanece no fim de cada palavra, e, portanto, cada valor representado é um inteiro. Quando os computadores calculam com números binários neste formato, as operações são chamadas *aritmética de ponto fixo.*

Em ciência, muitas vezes é necessário calcular usando números muito grandes ou muito pequenos. Os cientistas adotaram, portanto, uma notação conveniente em que uma *mantissa* mais um *expoente* são usados para representar um número. Por exemplo, 4.900.000 pode ser escrito como $0{,}49 \times 10^7$, onde 0,49 é a mantissa e 7 é o valor do expoente, ou 0,00023 pode ser escrito como $0{,}23 \times 10^{-3}$. A notação é baseada na relação $y = a \times r^p$, onde y é o número a ser representado, a é a mantissa, r é a base do sistema de numeração ($r = 10$ para o sistema decimal, e $r = 2$ para o sistema binário) e p é a potência a que a base é elevada.

É possível calcular com este sistema de representação. Para multiplicar $a \times 10^n$ por $b \times 10^m$, formamos $(a \times b) \times 10^{m+n}$. Para dividir $a \times 10^m$ por $b \times 10^n$, formamos $a/b \times 10^{m-n}$. Para adicionar $a \times 10^m$ a $b \times 10^n$, devemos fazer m igual a n. Se $m = n$, então $a \times 10^m + b \times 10^n = (a + b) \times 10^m$. O processo de fazer m igual a n é chamado *mudança de escala* dos números.

Considerável "escrituração" pode estar envolvida na mudança de escala dos números, e pode haver dificuldade em manter a precisão durante as computações quando os números podem variam ao longo de uma faixa muito ampla de magnitudes. Para uso do computador, estes problemas são atenuados por meio de duas técnicas pelas quais o computador (não o programador) não perde de vista o ponto de base (decimal), mudando automaticamente a escala dos números. Na primeira, as *rotinas de ponto-flutuante* programadas mudam automaticamente a escala dos números usados durante as computações enquanto mantêm a precisão dos resultados e não perdem de vista os fatores de escala. Estas rotinas são usadas com pequenos computadores que têm somente operações de ponto-fixo. Uma segunda técnica reside na construção das chamadas "operações de ponto-flutuante" no hardware do computador. O conjunto de circuitos lógicos do computador é então usado

*Esta seção pode ser omitida durante uma primeira leitura.

para executar automaticamente a mudança de escala e acompanhar o desenvolvimento dos expoentes quando os cálculos são efetuados. Para conseguir isto, é usado um sistema de representação de números chamado sistema de ponto-flutuante.

Um número de ponto-flutuante num computador usa o sistema de notação exponencial descrito acima, e, durante os cálculos, o computador acompanha o desenvolvimento do expoente bem como da mantissa. Uma palavra de número do computador num sistema de ponto-flutuante pode ser dividido em três partes: a primeira é o bit de sinal, indicando se o número é negativo ou positivo; a segunda parte contém o expoente do número a ser representado; e a terceira parte, a mantissa.

Como exemplo, consideremos um computador cujo comprimento da palavra é de 12 bits e com uma palavra de ponto-flutuante. A Fig. 6.21 ilustra este caso. É prática comum chamar a parte exponencial da palavra *característica* e a seção da mantissa de *parte inteira;* adotaremos esta notação.

A parte inteira da palavra de ponto-flutuante mostrada representa seu valor em forma de grandeza com sinal (em vez de complemento de 2, embora este tenha sido usado). A característica também está em forma de grandeza com sinal. O valor do número expresso é $I \times 2^C$, onde I é o valor da parte inteira e C é o valor da característica.

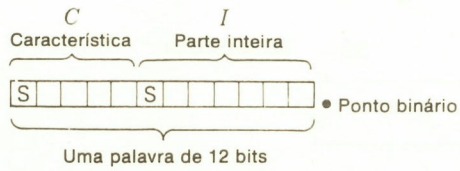


Fig. 6.21 Palavra de 12 bits de ponto-flutuante.

A Fig. 6.22 mostra diversos valores de números de ponto-flutuante tanto em forma binária quanto depois de serem convertidos à forma decimal. Uma vez que a característica tem 5 bits e está em forma de grandeza com sinal, o C em $I \times 2^C$ pode ter valores desde -15 até $+15$. O valor de I é um inteiro binário de magnitude com sinal mais e de 7 bits, e, portanto, I pode ter valores desde -63 até $+63$. O maior número representado por este sistema teria um I máximo e seria 63×2^{15}. O menor número seria -63×2^{15}.

Este exemplo mostra o uso de um sistema de representação de números de ponto-flutuante para armazenar números "reais" de extensão considerável numa palavra binária.

Uma outra prática amplamente seguida consiste em expressar a mantissa da palavra como uma fração e não como um inteiro. Isso está de acordo com o uso científico em geral, uma vez que comumente dizemos que $0{,}93 \times 10^4$ está em forma "normal" de notação exponencial (e não 93×10^2). Neste emprego, uma mantissa em decimal normalmente tem um valor de 0,1 a 0,999.... Similarmente, uma mantissa binária em forma normal teria um valor

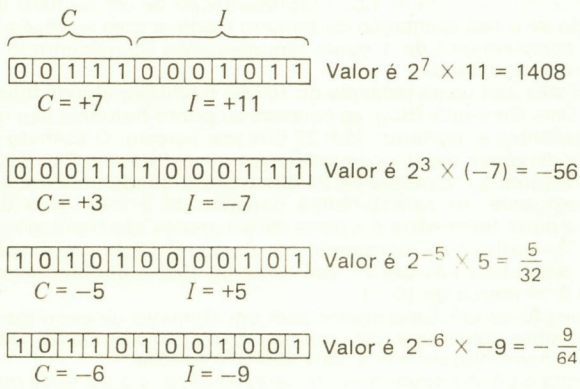


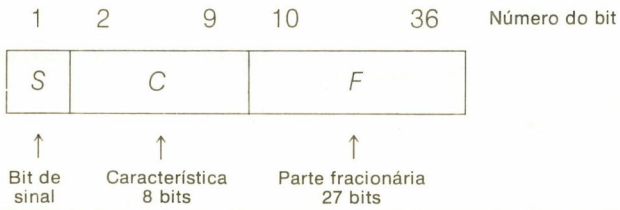
Fig. 6.22 Valores de números de ponto-flutuante nos sistemas de todos os inteiros de 12 bits.

desde 0,5 (decimal) a menos do que 1. A maioria dos computadores mantém suas seções mantissa em forma normal, ajustando continuamente as palavras, e assim um bit (1) significativo está sempre na posição mais à esquerda da mantissa (próximo ao bit de sinal).

Quando a mantissa está em forma de fração, esta seção é chamada *fração*. Em nosso exemplo de 12 bits podemos expressar números de ponto-flutuante com característica e fração supondo simplesmente que o ponto binário esteja à esquerda da magnitude (e não à direita como na representação do inteiro). Neste sistema, um número a ser representado tem valor $F \times 2^C$, onde F é a fração binária e C é a característica.

Na palavra de 12 bits considerada antes, as frações teriam valores a partir de $1 - 2^{-6}$ que é 0,111111, até $-(1 - 2^{-6})$, que é 1,111111; portanto, os números desde $(1 - 2^{-6}) \times 2^{15}$ até $-(1 - 2^{-6}) \times 2^{15}$ podem ser representados, ou cerca de +32.000 até −32.000. O menor valor que a parte fracionária deve ter é agora a fração 0,1000000, que é 2^{-1} vezes a menor característica, a qual é 2^{-15}, portanto, o menor número positivo representável é $2^{-1} \times 2^{-15}$ ou 2^{-16}. A maioria dos computadores usa este sistema faccional para a mantissa, embora os computadores da Burroughs Corporation e da National Cash Register Company usem o sistema de inteiros descrito anteriormente.

A Univac 1108 representa os números de ponto-flutuante de precisão simples neste formato:



Para os números positivos, a característica C é tratada como um inteiro binário, o bit de sinal é um 0 e a parte fracionária é uma fração binária com valor $0,5 \leq F < 1$. O valor do número representado é $2^{C - 128} \times F$. Este é o chamado *sistema deslocado* porque o valor da característica é simplesmente o valor inteiro naquela porção da palavra menos um afastamento que neste caso é 128. O expoente pode portanto variar de −128 a +127, uma vez que o inteiro na seção característica tem comprimento de 8 bits.

Por exemplo, a palavra binária

0,10000001 1100.....0
Característica Fração

tem valor $2^{129 - 128} \times 3/4 = 2 \times 3/4 = 1,5$. A representação de um número negativo pode ser deduzida formando-se a representação do número positivo com a mesma magnitude e formando depois o complemento de 1 desta representação (considerando todos os 36 bits como um único número binário).

Em computadores que usam palavras de 16 bits (inclusive alguns fabricados pela DEC, Hewlett-Packard, Data General e IBM), as palavras de ponto-flutuante são representadas por *duas* palavras adjacentes e, portanto, têm 32 bits por palavra. O formato real das palavras de ponto-flutuante de vários desses computadores é mostrado na Fig. 6.23. Nestes computadores, a parte fracionária F consiste de 24 bits representando uma fração de 23 bits e um bit de sinal. O expoente ou característica consiste de 8 bits. (Nos computadores de Hewlett-Packard, a parte fracionária e a parte característica são representadas em forma de complemento de 2, conforme programado em Fortran.) Cada um desses computadores pode representar magnitudes de até 2^{127} (ou cerca de 10^{38}) e frações aproximadamente tão pequenas quanto 2^{-138} (cerca de 10^{-38}).

Um outro exemplo de um computador com um conjunto de circuitos internos que efetua operações de ponto-flutuante e usa uma representação de uma só palavra de computador de números de ponto-flutuante é a série 360/370 da IBM.

A IBM denomina a parte exponencial de *característica*, e a parte da mantissa de *fração*. Na série 360/370, as palavras de dados de ponto-flutuante podem ser ou de 32 ou de 64 bits de comprimento. Os formatos básicos são como segue:

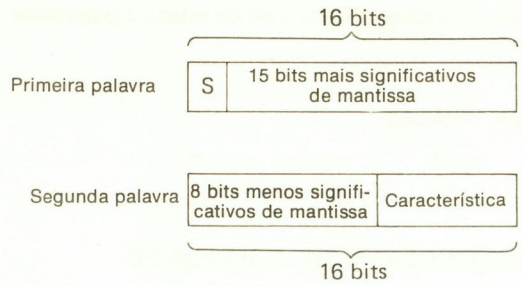
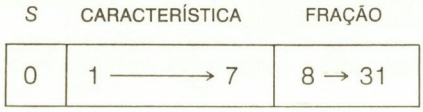
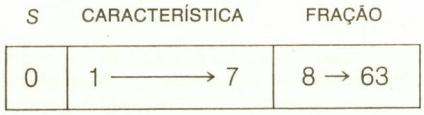


Fig. 6.23 Representação de ponto-flutuante usando duas palavras.

Número de ponto-flutuante de palavra-simples ou curta:



Número de ponto-flutuante de palavra-dupla ou longa:



Em ambos os casos, o bit *S* de sinal está na posição mais à esquerda e dá o sinal do número. A parte característica da palavra compreende então os bits 1 – 7 e é simplesmente um inteiro binário que designaremos por *C*, variando de 0 a 127. O valor real do fator de escala é formado subtraindo-se 64 deste inteiro *C* e elevando 16 a esta potência. Assim, o valor 64 em bits de 1 a 7 dá um fator de escala de $16^{64-64} = 16^{64-64} = 16^{0}$; um 93 (decimal) em bits de 1 a 7 dá um fator de escala de $16^{93-64} = 16^{93-64}$ que é 16^{29}; e um 24 em bits de 1 a 7 dá um 16^{-40}.

A magnitude do número real representado numa certa palavra de ponto-flutuante é igual a este fator de escala multiplicado pela fração contida nos bits 8 a 31 do número curto, ou 8 a 63 de um número longo. O ponto de base é suposto estar à esquerda do bit 8 em qualquer caso; portanto, se os bits 8 a 31 contiverem 1000 . . .00, a fração tem valor $^1/_2$ (decimal); isto é, a fração é 0,1000 . . .000 em binário. Similarmente, se os bits 8 a 31 contiverem 11000 . . .000, o valor da fração é o decimal $^3/_4$, ou o binário 0,11000 . . .000.

O número real representado tem então magnitude igual ao valor da fração multiplicado pelo valor determinado pela característica. Consideremos um número curto:

	SINAL	CARACTERÍSTICA	FRAÇÃO
N.º de ponto-flutuante :	0	1 0 0 0 0 0 1	1 1 1 0 0 . . . 0
Posição do bit :	0	1 2 3 4 5 6 7	8 9 10 11 12 . . . 31

O bit de sinal é um 0, e, portanto, o número representado é positivo. A característica tem valor 1000001, que é o decimal 65, e, portanto, o fator de escala é 16^1. A parte fracionária tem o valor binário 0,111, ou o decimal $^7/_8$, e assim o número representado é $^7/_8 \times 16$, ou 14 em decimal.

Novamente, consideremos o seguinte número:

	SINAL	CARACTERÍSTICA	FRAÇÃO
N.º de ponto-flutuante :	1	1 0 0 0 0 0 1	1 1 1 0 0 . . . 0
Posição do bit :	0	1 2 3 4 5 6 7	8 9 10 11 12 . . . 31

Este tem valor −14 já que cada bit é o mesmo que antes, exceto o bit de sinal. (O sistema de representação do número é de magnitude com sinal.)

Como outros exemplos, temos:

SINAL	CARACTERÍSTICA	FRAÇÃO	
0	1000011	110...0	$16^{3} \times \frac{3}{4} = 3072$
0	0111111	110...0	$16^{-1} \times \frac{3}{4} = \frac{3}{64}$

6.21 EXECUÇÃO DE OPERAÇÕES ARITMÉTICAS COM NÚMEROS DE PONTO-FLUTUANTE

Um computador requer, obviamente, um conjunto adicional de circuitos para manipular automaticamente os números de ponto-flutuante. Certas máquinas vêm equipadas com instruções de ponto-flutuante. (Nos computadores tais como DEC PDP-11/45 e outros, o conjunto de circuitos de ponto-flutuante pode ser comprado e acrescentado para permitir que eles efetuem operações de ponto-flutuante.)

Para manipular números de ponto-flutuante, a máquina deve ser capaz de realizar extensas operações de deslocamento e comparação. As regras da multiplicação e da divisão seguem abaixo:

$$(a \times r^{p}) \times (b \times r^{q}) = ab \times r^{p+q}$$

$$(a \times r^{p}) \div (b \times r^{q}) = \frac{a}{b} \times r^{p-q}$$

O computador deve ser capaz de somar ou de subtrair as seções de expoente dos números de ponto-flutuante e também de efetuar as operações de multiplicação ou divisão com as seções de mantissa dos números. Além disso, a precisão é geralmente mantida pelo deslocamento dos números armazenados até que os dígitos significativos estejam nas seções mais à esquerda da palavra. A cada deslocamento, o expoente deve ser mudado. Se a máquina estiver deslocando a seção de mantissa para a esquerda, em cada deslocamento para a esquerda o expoente deve ser diminuído.

Por exemplo, num computador de BCD, consideremos a palavra

0	10	0064
Sinal	Expoente	Mantissa

Para alcançar a precisão, o computador desloca a seção de mantissa para a esquerda até que o 6 esteja na posição mais significativa. Já que são necessários dois deslocamentos, o expoente deve ser diminuído de 2, e a palavra resultante é 0,08 6400. Se todos os números a serem usados são mudados de escala desta maneira, a precisão máxima pode ser mantida durante todos os cálculos.

Na adição e na subtração, os valores dos expoentes devem coincidir. Por exemplo, para somar $0{,}24 \times 10^{5}$ a $0{,}25 \times 10^{6}$, devemos mudar a escala dos números de modo que os expoentes coincidam. Assim

$$(0{,}024 \times 10^{6}) + (0{,}25 \times 10^{6}) = 0{,}274 \times 10^{6}$$

A máquina deve também seguir este procedimento. Os números devem ser mudados de escala conforme foi descrito, de modo que o dígito mais significativo da seção de mantissa do computador de cada palavra contenha o dígito mais significativo do número armazenado. Em seguida, o maior dos dois expoentes dos operandos é selecionado, e a outra mantissa do número é deslocada e seu expoente ajustado até que os expoentes de ambos os números coincidam. Os números podem então ser adicionados ou subtraídos de acordo com estas regras:

$$(a \times r^{p}) + (b \times r^{p}) = (a + b) \times r^{p}$$

$$(a \times r^{p}) - (b \times r^{p}) = (a - b) \times r^{p}$$

EXERCÍCIOS

1. Desenhar um diagrama-bloco de um conjunto de circuitos com dois registros X e Y de três biestáveis cada um, de modo que Y possa ser transferido para dentro de X ou o complemento de 1 de Y possa ser transferido para dentro de X.
2. Dois registros binários paralelos, designados como registro X e registro Y, consistem ambos de três biestáveis. Desenhar um diagrama-bloco dos registros e do conjunto de circuitos lógicos necessário de modo que *(a)* o registro X possa ser limpo, ou complementado de 1, e *(b)* os conteúdos do registro Y possam ser transferidos para dentro do registro X.
3. Se um computador binário manipula números no sistema de inteiros de magnitude de sinal mais, e os números são de 5 bits (sinal mais 4 bits) cada, como seriam representados os seguintes números decimais? Por exemplo, +5 = 0,0101.

   (a) +6 *(b)* +10 *(c)* −12 *(d)* −16

4. Se um computador binário representa números numa magnitude de sinal mais com 5 bits por número, os seguintes números decimais seriam representados:

   (a) +7 *(b)* +10 *(c)* −6
   (d) −8 *(e)* −17 *(f)* −14

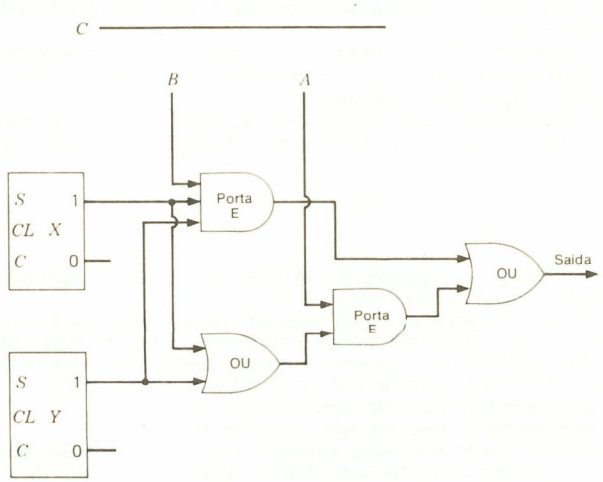
5. Se uma máquina binária manipula números negativos na forma de verdadeira grandeza, como seria armazenado −4 num registro com um bit de sinal e 4 bits que representam magnitude? Se a mesma máquina armazenasse números no sistema de complemento de 1, como seria armazenado −4?
6. Se usamos o símbolo $\oplus$ que significa OU exclusivo e o definimos como $X \oplus Y = \bar{X}Y + X\bar{Y}$, então a saída S de um somador completo pode ser escrita como $S = X \oplus Y \oplus C_i$. Mostrar por que este é o caso.
7. As entradas no somador completo na Fig. 6.3 são como segue: $X = 1$, $Y = 1$ e $C = 1$. O que representará a saída nas linhas S e C?
8. Se um registro contiver cinco biestáveis como na Fig. 6.1*(b)* e o registro contiver $X_0 = 1$, $X_2 = 1$, $X_3 = 0$, $X_4 = 1$, $X_5 = 1$, dará o valor decimal do número no registro se for usado o sistema de numeração de complemento de 1? Qual será o valor se for usado complemento de 2?
9. Se carregarmos o número binário 1,0011 nos biestáveis da Fig. 6.1*(b)*, isto é, se $X_0 = 1$, $X_1 = 0$, $X_2 = 0$, $X_3 = 1$ e $X_4 = 1$, qual será o valor do registro no sistema de numeração de complemento de 1? Dar a resposta em decimal. Qual será o valor deste número se for usado o sistema de numeração de complemento de 2?
10. Se o registro X contiver 0,0110 e o registro Y contiver 1,1011, o que representam os dois números em decimal se for usado o complemento de 1? Se for usado complemento de 2? Somar os dois números em complemento de 1, depois em complemento de 2, e dar os resultados em decimal.
11. O registro X contém 0,1100, e o registro Y contém 0,1101 (onde os 0 que precedem o ponto binário designam que o número armazenado é positivo). Se os dois registros forem adicionados, qual será o resultado?
12. Se for usado complemento de 1, para qual das seguintes expressões será gerado um estouro ou um retorno de transporte? Por quê? Admitir registros de 5 bits, inclusive um bit de sinal.

    (a) +5 + (−6) *(b)* +5 + (−3)
    (c) +12 + (−12) *(d)* +12 + 2

13. Acrescentar um estágio ao registro de CARGA e DESLOCAMENTO na Fig. 6.17. Copiar apenas D em seu desenho, omitindo A, B e C.
14. Um registro binário consiste de cinco dispositivos de armazenamento binários; um armazena o bit de sinal e os outros quatro armazenam os bits de magnitude. Se o número armazenado for 0,0100 e este número for depois deslocado à direita de um lugar binário, qual será o resultado? Admitir que um 0 cabe no bit de sinal.
15. Projetar um semi-somador que use apenas portas NOU.
16. Projetar um semi-somador que use apenas portas NE.
17. Um binário semi-subtrator tem duas entradas x e y e duas saídas que são o valor da "diferença" $x - y$ e uma saída de "empréstimo" que é 1 se o valor de $x - y$ for negativo

($x - y$ então recebe o valor 1). Desenhar um diagrama-bloco de um semi-subtrator usando portas NE e admitindo que x, $\bar{x}$, y e $\bar{y}$ sejam todos disponíveis como entradas.

18. Projetar um semi-subtrator usando apenas portas NOU.
19. Se uma entrada de empréstimo for acrescentada ao semi-subtrator nos Exercícios 17 e 18, um subtrator completo fica formado. Projetar um subtrator completo usando apenas portas NE.
20. Projetar um subtrator completo usando apenas portas NOU.
21. Projetar um somador completo usando apenas portas NE.
22. É possível ocorrer um estouro durante a multiplicação em máquina binária com números armazenados em magnitude de sinal mais de ponto fixo? Admitir um produto de comprimento duplo.
23. Explicar como é formado o complemento de 2 do subtraendo quando subtraímos usando a configuração na Fig. 6.6. Como é acrescentado o 1 para formar este complemento?
24. Mostrar como a configuração da Fig. 6.6 soma e subtrai adicionando e subtraindo +6 e +3 em binário, mostrando qual será cada saída.
25. Somar e subtrair +6 e −4 usando a configuração na Fig. 6.6, mostrando o que será cada saída e verificando a correção da soma e da diferença.
26. Acrescentar uma linha NE de controle e o conjunto de circuitos à Fig. 6.20 de modo que o NE de ACC e B possam ser transferidos para dentro de ACC.
27. Se um registro que contém 0,110011 é logicamente adicionado a um registro que contém 0,101010, qual será o resultado? Qual será o resultado se os registros forem multiplicados logicamente? Se os registros forem introduzidos juntos em circuitos OU exclusive?
28. Achar uma seqüência de operações lógicas que farão com que o ACC tenha valor 0 independente de como comecem ACC e B na Fig. 6.20.
29. Com referência à Fig. 6.20, mostrar que uma multiplicação lógica seguida de uma adição lógica transferirá os conteúdos de B para dentro de ACC.
30. Acrescentar uma linha NOU de controle e o conjunto de circuitos à Fig. 6.20 de modo que o NOU de ACC e B possa ser transferido para dentro de ACC.
31. Demonstrar por meio de uma tabela de combinações que dois semi-somadores mais uma porta OU fazem um somador completo conforme indicado na Fig. 6.4.
32. Acrescentar portas e um biestável X_{m+2} de modo que possamos deslocar para a esquerda e para a direita no X_{m+1} na Fig. 6.16.
33. Acrescentar portas ao seguinte desenho de modo que, se o sinal C de entrada de controle for feito um 1, o valor $\bar{X}\bar{Y}$ aparecerá na linha de saída. Se A for um 1, então $X + Y$ aparece na saída; se B for um 1, então $X \times Y$ aparece na saída; se A, B e C forem 0, então a saída deve ser um 0; e somente um dentre A, B ou C pode ser um 1 num certo instante.



34. Em que caso a subtração de um número negativo de 0 causará um estouro no sistema de complemento de 2?
35. Se forem representados números num sistema de inteiros de complemento de 2, magnitude de 6 bits mais 1 bit de sinal e ignorarmos o estouro (isto é, qualquer resultado será armazenado mesmo se requerer mais bits de magnitude), então o maior inteiro positivo que pode resultar da adição de dois números é _____, e o maior inteiro positivo que pode resultar da subtração de um número de um outro é _____? Os números menos negativos que podem resultar de uma adição e de uma subtração são _____ e _____. Em conseqüência, qualquer soma ou diferença pode ser armazenada em _____ bits.
36. Anotar todos os números de complemento de 2 com 4 bits (isto é, números de 3 bits mais sinal) e seus valores decimais. Mostrar que há um número mais negativo do que número positivo. Considerar que o 0 não é nem negativo nem positivo.
37. Quando é efetuada adição numa máquina binária que usa o sistema de numeração de complemento de 2 para representar números negativos, pode ocorrer um estouro num registro somente quando dois números positivos ou dois negativos forem somados. Mostrar que a adição de um número positivo e de um número negativo não pode resultar numa condição de estouro.
38. Mostrar que há tantos números negativos quanto positivos num sistema de complemento de 1.
39. As expressões de álgebra booleana nas linhas de saída das portas na Fig. 6.8 não estão completas. Desenvolver as expressões de álgebra booleana para as saídas S_1, S_2 e C_2 da rede ou estrutura.
40. Qual dos seguintes sistemas de numeração tem dois 0:
    (a) Magnitude com sinal mais *(b)* Complemento de 1 *(c)* Complemento de 2
41. Quando o sistema de numeração de complemento de 2 é usado e a adição é efetuada, designemos o "vai um" do somador completo conectado no somador completo para os dígitos de sinal como C_1 (ver Fig. 6.6). A regra do estouro é que dois números causam um estouro quando são adicionados se ambos forem positivos e C_1 for igual a 1, ou se ambos forem negativos e C_1 for diferente de 1. Portanto, examinando-se os dígitos de sinal dos dois números que estão sendo somados e o transporte do somador completo que adiciona os dois dígitos mais significativos da magnitude dos números que estão sendo somados, podemos formar uma rede lógica cuja saída será um 1 quando surgir uma condição de estouro e um 0 se a adição for legitimada. Façamos X_0 armazenar o dígito de sinal do primeiro somando, Y_0 armazenar o dígito de sinal do segundo somando e C_1 ser novamente o transporte dos somadores completos conectados aos biestáveis X_1 e Y_1 como na Fig. 6.6. Mostrar que a equação lógica de uma condição de estouro é $X_0Y_0\bar{C}_1 + \bar{X}_0\bar{Y}_0C_1$ = estouro.
42. Modificar a Fig. 6.16 de modo que o complemento de X_{m+1} possa ser deslocado para dentro de X_m, isto é, de modo que DESLOCAMENTO-À ESQUERDA faça com que X_{m+1} entre em X_m.
43. Mostrar que, quando adicionamos 7 a 9 no sistema BCD usando o somador de BCD série-paralelo na Fig. 6.10, a resposta estará correta. Faça isto indicando as saídas do circuito, completando o valor binário de cada X e Y mostrados na figura e mostrando também os valores de Z_8, Z_4, Z_2 e Z_1.
44. Mostrar como o somador de BCD na Fig. 6.11 soma +6 a +5, calculando cada saída das portas e depois as saídas finais.
45. Verificar como as portas na Fig. 6.12 formam um complemento de 9 experimentando 6 e 2 em BCD nas entradas.
46. Explicar a operação da Fig. 6.15 esclarecendo como 324 pode ser somado ou subtraído a/de 523 nesta configuração.
47. Qual é a função do biestável tipo *D* na Fig. 6.15?
48. Explicar como carregar os valores de entrada sobre *W*, *X*, *Y* e *Z* nos biestáveis da Fig. 6.17.
49. Explicar como fazer com que os biestáveis na Fig. 6.17 se desloquem para a direita três vezes. Admitir que (1) façamos a entrada de reajuste um 0, depois um 1, (2) mantenhamos *J* e *K* em 1 e desloquemos em 1, e depois (3) apliquemos três pulsos de relógio à linha de *relógio*. Desenhar as formas-de-onda de saída para *A*, *B*, *C* e *D* nesta seqüência de entradas.
50. Qual será o resultado se multiplicarmos 0,1101 por 0,0011 em nossa máquina generalizada na Fig. 6.18? Dar os valores em cada *X* e biestável *B*.
51. Qual será o número binário que representa −1 no sistema de numeração fracionário de complemento de 2 se representarmos o número que usa um dígito de sinal mais quatro

dígitos de magnitude?

52. Usando o sistema BCD 8421 com um único dígito para o dígito de sinal, escrever os seguintes números usando um sistema de numeração de magnitude de sinal mais.

    (a) +0015 *(b)* +0191
    (c) −2246 *(d)* −0264

53. Usando o sistema BCD 8421, escrever em forma binária os seguintes números em decimal. Usar um único dígito para o dígito de sinal e expressar os números como magnitude de sinal mais.

    (a) +0042 *(b)* −0122 *(c)* +1034 *(d)* −1097

54. Usando o código 8421 como no Exercício 52, dar os mesmos números, mas usar complemento de 9 para os números negativos.
55. Expressar cada um dos números no Exercício 53 usando o complemento de 9 e o sistema BCD 8421. Por exemplo, −1024 = 1,1000 1001 0111 0101.
56. Escrever a forma binária dos números no Exercício 52 usando 8421, mas usar complementos de 10 para os números negativos.
57. Escrever os números decimais no Exercício 53 usando o sistema de numeração de complemento de 10 e novamente o sistema BCD 8421 para representar estes números. Por exemplo, −1420 = 1,1000 0101 1000 0000.
58. Se adicionarmos dois números binários de 20 dígitos usando os somadores completos mostrados na Fig. 6.8 e se são necessários 3 ns para um sinal passar através de uma porta, qual será o tempo máximo que irá requerer um sinal de TRANSPORTE para se propagar dos bits de ordem mais baixa aos bits de ordem mais alta, admitindo um somador completo que está em paralelo como na Fig. 6.6? O circuito de transporte na Fig. 6.8 pode ser substituído por uma "ponte de transporte" com expressão $C2 = C0 \cdot A1 \cdot B2 + C0 \cdot A1 \cdot B2 + C0 \cdot A2 \cdot B1 + C0 \cdot B1 \cdot B2 + A1 \cdot B1 \cdot A2 + A1 \cdot B1 \cdot B2 + A2 \cdot B2$. Mostrar como isto funcionaria e como pode funcionar uma ponte de transporte de 3 bits.
59. Explicar como se acrescentaria portas e entradas à Fig. 6.17 de modo que o registro de biestável pudesse ser deslocado para a esquerda bem como para a direita.
60. Se multiplicarmos 6 × 11 nos registros na Fig. 6.18, mostrar a colocação de dígitos binários no início e no fim da multiplicação.
61. Mostrar como 7 × 9 e 5 × 6 seriam multiplicados nos registros na Fig. 6.18.
62. Desenhar um fluxograma (como na Fig. 6.19) para o processo descrito para a multiplicação binária.
63. Se dividirmos 23 por 5 nos registros na Fig. 6.18, mostrar o posicionamento inicial dos números (em binário) e o resultado no fim. (Mostrar onde o quociente e o resto são colocados.)
64. Mostrar como dividir 14 por 3 usando os registros na Fig. 6.18 e mostrando como o quociente e o resto são colocados após a divisão.
65. Examinar a divisão de 14 por 5 usando a técnica mostrada no texto.
66. Usando o algoritmo indicado na Fig. 6.19, mostrar como dividir 11 por 3.
67. Mostrar como representar +7 na palavra de 12 bits em ponto-flutuante na Fig. 6.21.
68. Mostrar como representar −12 na palavra de 12 bits de ponto-flutuante na Fig. 6.21.
69. *(a)* Dar a representação de ponto-flutuante da série 360/370 da IBM para 52,5 e −52,5.
    (b) Dar a representação de ponto-flutuante da Univac 1108 para +24,0 e −24,0.
    (c) Comparar as variações, precisão e outras considerações de projeto do sistema para os dois sistemas de numeração de ponto-flutuante de computador acima.
70. Mostrar dois números decimais que, quando convertidos ao sistema de numeração de ponto-flutuante IBM 360/370, terão 0,0001 em bits 9, 10, 11 e 12.
71. Podem-se apresentar quaisquer razões que justifiquem o fato dos projetistas de sistemas em decisão da IBM usar a base hexadecimal no sistema de numeração de ponto-flutuante da série 360/370 em vez da base 2 convencional? Assim, no sistema 360/370, uma palavra completa de ponto-flutuante de 32 bits tem como característica um inteiro de 7 bits com valor C e como fração, um número binário de 25 bits de fração de magnitude com sinal e com valor F. O valor do número representado é então $(16^{C-64})F$. Por que 16^{C-64} em vez de 2^{C-64}? Dar as considerações de projeto do sistema.

72. Dar o valor de um *inteiro* positivo não-zero menor do que 16^{62} que não pode ser representado no sistema de numeração de ponto-flutuante IBM 360/370 (usando uma única palavra de 32 bits). Não tenha receio de usar uma expressão tal como $2^{16} + 3$ em sua resposta, mas procure justificar por que se acredita que a resposta esteja correta.
73. Um computador binário com um código básico de 16 bits usa um sistema de numeração de inteiros de complemento de 2. O elemento aritmético contém um acumulador e registro *MQ*, cada um contendo 16 biestáveis (ou bits). Quando é efetuada uma multiplicação, a palavra de 16 bits tirada da memória é multiplicada pela palavra de 16 bits no acumulador e o produto é armazenado no combinado ACC-*MQ* com o bit menos significativo no bit mais à direita do *MQ*. O bit de sinal do MQ é ajustado exatamente como o sinal do ACC e não é usado para armazenar um bit de magnitude. Em que bit do ACC aparece o bit mais significativo de um produto de números de magnitude máxima?

7

O ELEMENTO DE MEMÓRIA

A memória de um computador não está realmente concentrada num local; os dispositivos de armazenamento estão espalhados por toda a máquina. Por exemplo, os *registros de operação* são registros de biestáveis usados nas unidades aritmética e de controle do computador. Os cálculos reais com os dados ocorrem nestes registros. Por exemplo, adições, multiplicações, deslocamentos etc. são todos efetuados nestes registros da máquina. O processamento real da informação é executado nestes registros e sob a direção deles.

Numa consideração superficial, a próxima categoria de dispositivo de armazenamento normalmente encontrada é chamada *memória de alta velocidade, memória interna* ou *memória principal.* Esta seção da memória do computador consiste de um conjunto de registros de armazenamento, cada um dos quais é identificado com um endereço que permite que a unidade de controle ou grave num registro particular ou leia de um registro particular.

É desejável que a velocidade de operação desta seção da memória do computador seja tão rápida quanto possível, pois a maioria das transferências de dados para/da seção de processamento de informação da máquina será através da memória principal. Por esta razão, os dispositivos de armazenamento com tempos de acesso muito rápidos são geralmente escolhidos para a memória principal; infelizmente, os dispositivos encontrados agora, com velocidade suficiente para executar esta função a contento, não possuem a capacidade de armazenamento que, às vezes, é requerida. Em conseqüência, memória adicional, que é chamada *memória auxiliar* ou *memória secundária,* é acrescentada à maioria dos grandes computadores. Esta seção da memória do computador é caracterizada pelo baixo custo por dígito armazenado, mas geralmente possui uma velocidade de operação bem mais baixa do que a de qualquer dos registros de operação ou da memória principal. Esta seção da memória é, às vezes, designada *armazenamento de reserva* (ou armazenamento secundário), pois sua função é manipular quantidades de dados além das que podem ser armazenadas na memória interna.

Os dispositivos de armazenamento mais exteriores e finais são os que se utilizam para introduzir no computador informação proveniente do "mundo exterior" e armazenar os resultados fornecidos pelo computador para o usuário deste computador. Os meios de armazenamento neste caso consistem geralmente de meios de entrada tais como cartões perfurados ou fita de papel perfurada, e as saídas da máquina consistem, em geral, de caracteres impressos. Novamente, o custo por bit é baixo, mas as velocidades de operação das leitoras de fitas e de cartões, das impressoras etc. são provavelmente da ordem de 1.000 vezes menor do que as velocidades dos registros de operação. Estes dispositivos serão descritos no Cap. 8 sob o título de dispositivos de entrada-saída. Este capítulo será limitado ao *armazenamento interno* da máquina, que se define por aqueles dispositivos de armazenamento que constituem uma parte integral da máquina e são diretamente controlados pela máquina.

Cada uma das divisões da memória tem certas características. Por exemplo, o estímulo da velocidade é muito grande nos registros de operação. Estes registros devem geralmente efetuar operações numa velocidade que é várias vezes maior que a velocidade da memória principal. A memória principal também requer altas velocidades de operação, mas devido a que é desejável armazenar maiores quantidades de dados (talvez 10^4 a 10^8 bits) nesta seção

da memória, deve ser feito, em geral, um compromisso entre custo e velocidade. A mesma espécie de compromisso muitas vezes deve ser feita no caso da memória auxiliar. Em máquinas de grande porte, a memória auxiliar pode ter de armazenar de 10^8 a 10^{12} dígitos binários, e, nestas situações, ela pode mostrar-se excessivamente dispendiosa no emprego de dispositivos tais como os utilizados na memória principal.

Um ponto importante a observar quando se considera a velocidade de operação é que, antes que uma palavra possa ser lida, é necessário localizá-la. O tempo necessário para localizar e ler uma palavra da memória é chamado *tempo de acesso.* Os procedimentos para a localização da informação podem ser divididos em duas classes: acesso aleatório e acesso seqüencial. Um dispositivo de armazenamento de *acesso aleatório* é aquele em que qualquer local no dispositivo pode ser selecionado ao acaso, o acesso à informação armazenada é direto, e, para cada local, é requerido tempo de acesso aproximadamente igual. Um registro de biestáveis constitui um exemplo de um dispositivo de armazenamento de acesso aleatório, como também é o sistema de memória de núcleos magnéticos, que será descrito abaixo. Um dispositivo de acesso seqüencial é aquele em que a chegada ao local desejado pode ser precedida pela colocação em seqüência através de outros locais, de modo que o tempo de acesso varia de acordo com o local.† Por exemplo, se tentarmos ler uma palavra armazenada num carretel de fita magnética e a parte da fita em que a palavra está armazenada ficar próxima ao centro do carretel, será necessário pesquisar em seqüência através de toda a fita interveniente antes que a palavra possa ser lida.

Uma outra maneira de subdividir os dispositivos de armazenamento é a seguinte: dispositivos de armazenamento estáticos ou dinâmicos. Um dispositivo de armazenamento *estático* é aquele em que a informação não muda de posições; os registros de biestáveis, os registros de núcleos magnéticos e até os cartões ou fitas perfurados são exemplos de dispositivos de armazenamento estático. Os dispositivos de armazenamento *dinâmico,* por outro lado, são dispositivos em que a informação armazenada está continuamente mudando de posição. Os registradores circulantes que utilizam linhas de retardo são exemplos de dispositivos de armazenamento dinâmico.

Este capítulo se concentrará nos quatro dispositivos mais freqüentemente usados para armazenamento da informação digital nas seções de memória interna dos computadores. Estes dispositivos são (1) memórias de núcleos magnéticos, que são dispositivos de acesso aleatório usados principalmente na memória interna devido a suas altas velocidades de operação e ao moderado custo por bit; (2) memórias de CI, que são de alta velocidade e de custo moderado; (3) memórias de disco e de tambor magnético, que são dispositivos de armazenamento de acesso direto geralmente usados para armazenamento auxiliar; e (4) memórias de fita magnética, que são usadas exclusivamente como um armazenamento auxiliar ou de reserva, mas que são capazes de armazenar grandes quantidades de informação a custos baixos. Em seqüência às seções sobre dispositivos de tambor, disco e fita magnéticos, serão descritas as técnicas usadas para registrar informação digital numa superfície magnética.

7.1 MEMÓRIAS DE ACESSO ALEATÓRIO

A memória principal de um computador é organizada de maneira particularmente desejável. A Fig. 7.1 mostra que uma memória principal de alta velocidade num computador é organizada em palavras de comprimentos fixos. Como o número indica, uma dada memória é dividida em N palavras, onde N geralmente é alguma potência de 2, e a cada palavra é atribuído um *endereço* ou *local* na memória. Cada palavra tem o mesmo número de bits, chamado *comprimento da palavra,* e se lermos, por exemplo, a palavra no local 72, receberemos da memória uma palavra com este número de bits.

Os endereços ou números de endereço na memória passam consecutivamente, começando pelo endereço 0 e passando até o maior endereço. Assim, no endereço 0, achamos uma palavra, no endereço 1, uma segunda palavra, no endereço 2, uma terceira palavra e etc. até a palavra final no maior endereço.

O computador pode ler uma palavra de ou gravar uma palavra em cada local na memória. Para uma memória com uma palavra de 8 bits, se gravarmos a palavra 01001011 no endereço 17 da memória e posteriormente lermos deste mesmo endereço, leremos a pala-

†Os dispositivos de acesso seqüencial são ainda separados em *dispositivos de armazenamento de acesso direto* (DASD) e *dispositivos de acesso em série.* Os dispositivos de armazenamento de acesso direto têm endereços, mas o tempo de acesso para alcançar os dados num certo endereço pode variar. Por exemplo, o tempo para localizar dados num disco de cabeçote móvel (a ser explicado) depende da posição do cabeçote e da posição do disco quando o endereço for fornecido. Os dispositivos de acesso em série são verdadeiramente em série em suas propriedades de acesso; a fita magnética constitui o exemplo clássico.

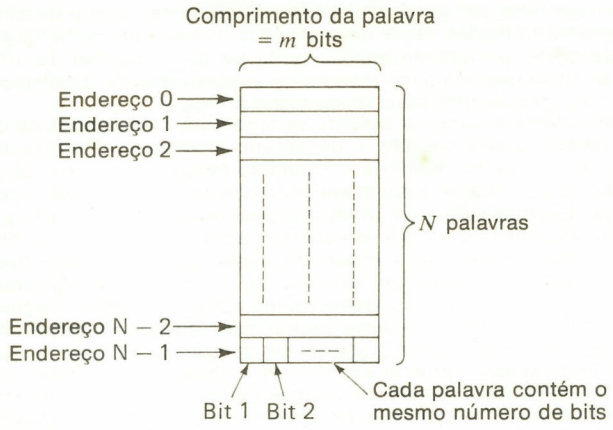


Fig. 7.1 Palavras na memória de alta-velocidade.

vra 01001011. Se novamente lermos deste endereço num instante mais tarde (e não tivermos gravado uma outra palavra), a palavra 01001011 será novamente lida. Isto significa que a memória é de *leitura não-destrutiva* no sentido de que a leitura não destrói ou altera uma palavra armazenada.

É importante compreender a diferença entre os *conteúdos* de um endereço da memória e o próprio endereço. Uma memória é como um grande armário que contém tantas gavetas quantos são os endereços na memória. Em cada gaveta está uma palavra, e o endereço de cada palavra está escrito no lado externo da gaveta. Se gravarmos ou armazenarmos uma palavra no endereço 17, é como colocar a palavra na gaveta etiquetada com o número 17. Mais tarde, a leitura do endereço 17 é como olhar naquela gaveta para verificar seu conteúdo. Não removemos a palavra num endereço quando lemos, porém mudamos os conteúdos num endereço somente quando armazenamos ou gravamos uma nova palavra.

De um ponto de vista exterior, uma memória de alta velocidade se assemelha muitíssimo a uma "caixa preta" com vários locais ou endereços nos quais os dados podem ser armazenados ou dos quais podem ser lidos os dados. Cada endereço ou local contém um número fixo de bits unitários, sendo este número denominado *comprimento da palavra* da memória. Uma memória com 4.096 locais, cada um com um endereço diferente e cada local armazenando 16 bits, é chamada uma "memória de 4.096 palavras de 16 bits", ou, na linguagem comercial de computadores, uma "memória de 4K de 16 bits". (Uma vez que as memórias geralmente vêm com um número de palavras igual a 2^n para um certo n, se a memória tiver $2^{14} = 16.384$ palavras, na literatura e jargão de computadores, esta memória é denominada "memória de 16K" porque sempre se compreende que o total de 2^n palavras realmente ocorre na memória. Assim, uma memória de 2^{15} palavras de 16 bits é chamada uma "memória de 32K de 16 bits".)

As memórias podem ser "lidas de" (isto é, os dados podem ser extraídos) ou "gravados em" (isto é, os dados podem ser introduzidos na memória). As memórias que podem ser tanto "lidas de" quanto "gravadas em" são chamadas *memórias de leitura/gravação*. Algumas memórias têm programas ou dados permanentemente armazenados e são chamadas *memórias de leitura exclusiva* (ou memórias fixas).

Um diagrama-bloco de uma memória de leitura exclusiva é mostrado na Fig. 7.2. O computador coloca o endereço do local em que os dados devem ser lidos no *registro de endereço da memória*. Este registro consiste de n dispositivos binários (geralmente, *biestáveis*), onde 2^n é o número de palavras que podem ser armazenadas na memória. Os dados a serem gravados na memória são colocados no *registro da memória intermediária*, que tem tantos dispositivos binários de armazenamento quantos forem os bits em cada palavra da memória. Diz-se que a memória grava por meio de um sinal 1 na linha GRAVAR (WRITE). A memória armazenará então os conteúdos do registro da memória intermediária no local especificado pelo registro de endereço da memória.

As palavras são lidas colocando-se o endereço do local a ser lido no registro de endereço da memória. Um sinal 1 é então colocado na linha LER (READ), e os conteúdos da-

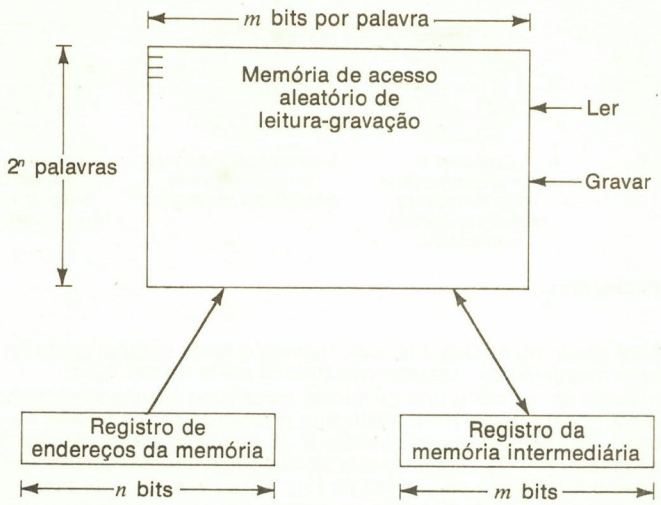


Fig. 7.2 Memória de acesso aleatório de leitura-gravação.

quele local são colocados pela memória no registro de memória intermediária.

Como pode ser visto, o computador se comunica com a memória por meio do registro de endereço da memória, do registro de memória intermediária e das entradas de LER e GRAVAR. As memórias são geralmente encapsuladas em módulos separados ou encapsulamentos. É possível comprar um módulo de memória com um tamanho especificado em vários fabricantes diferentes, e, por exemplo, um módulo de memória de 8K com 16 bits pode ser comprado num painel de circuito pronto para o uso. Similarmente, se um computador for comprado com uma certa quantidade de memória principal, posteriormente poderá ser adicionada mais memória comprando-se outros módulos e "ligando-se os mesmos".

Se for possível extrair ou introduzir dados de qualquer local "imediatamente", isto é, se não houver nenhum outro retardo para alcançar um local que está oposto a um outro local, a memória é chamada *memória de acesso aleatório*. Os modernos computadores quase invariavelmente usam memórias de leitura/gravação de acesso aleatório em sua memória principal de alta velocidade e depois usam memórias de reserva ou de velocidade mais baixa para guardar ou reter os dados auxiliares.

Os dois tipos principais de memórias de acesso aleatório de alta velocidade são a memória de núcleos magnéticos e a memória de CI. Estas são descritas nas seções imediatamente seguintes.

7.2 ARMAZENAMENTO EM NÚCLEOS MAGNÉTICOS

O dispositivo básico de armazenamento numa memória de núcleos magnéticos consiste de uma pequena peça toroidal (em forma de anel) de material magnético chamada núcleo magnético. O núcleo magnético é geralmente uma peça sólida de material cerâmico ferromagnético.

A Fig. 7.3 ilustra um núcleo magnético com dimensões ampliadas. Cada núcleo magnético na memória é capaz de armazenar 1 bit de informação digital. Um enrolamento (fio) de entrada é mostrado em espiras através deste núcleo; se a corrente atravessar este enrolamento, será produzido fluxo magnético com um sentido que depende do sentido da corrente através do enrolamento. O núcleo é em forma de anel e constituído de material com uma alta permeabilidade, de modo que apresentará um caminho de baixa relutância para o fluxo magnético. Dependendo do sentido da corrente através do enrolamento de entrada, o núcleo se tornará magnetizado ou no sentido horário ou no sentido anti-horário.† A retenti-

†Há uma regra ótima para a determinação do sentido do fluxo ao redor do condutor. Segure o fio com a mão direita, mantendo o polegar no sentido do fluxo da corrente. Os dedos então apontam no sentido do fluxo ao redor do condutor. Esta é a chamada "regra da mão direita".

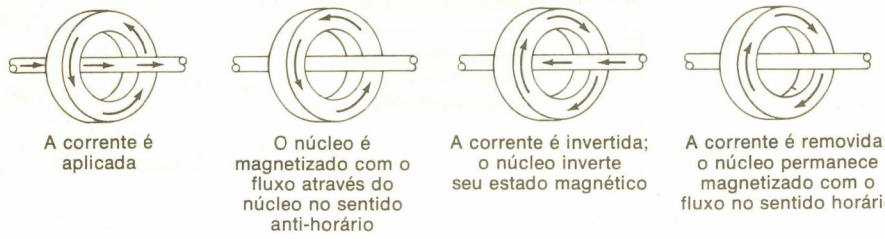


Fig. 7.3 O núcleo magnético.

vidade do material usado no núcleo é tal que, quando a força magnetizante for removida, o núcleo permanece magnetizado, retendo uma grande parte de seu fluxo.

As características de um certo tipo de núcleo magnético são geralmente estudadas por meio de um gráfico, com a força H magnetizante produzida pela corrente do enrolamento "plotada" ao longo da abscissa, e a densidade B de fluxo resultante através do núcleo ao longo da ordenada (Fig. 7.4). Se uma corrente cíclica que é alternadamente positiva e negativa for aplicada ao enrolamento de entrada na Fig. 7.3, a cada valor da corrente de entrada haverá uma força H magnetizante aplicada ao núcleo. Se a densidade B de fluxo no núcleo for então "plotada" contra esta força magnetizante, a curva resultante será chamada *laço de histerese*. Na Fig. 7.4, a força H aplicada é suficiente para saturar o material do núcleo em ambas as extremidades positiva e negativa da corrente de entrada. A máxima densidade de fluxo através do núcleo quando o núcleo estiver saturado com um valor positivo de H é designada $+B_m$, e o valor negativo correspondente de H é designado $-B_m$.

Observemos que, para cada valor da força magnetizante, há dois valores de densidade de fluxo. Um ocorre quando a força magnetizante está aumentando, e o outro, quando ela está diminuindo. Se a força magnetizante for variada de $-H_m$ a $+H_m$, a densidade de fluxo se deslocará ao longo da parte inferior da curva, e, se a força magnetizante for deslocada de $+H_m$ a $-H_m$, a densidade de fluxo se deslocará ao longo da parte superior da curva, novamente acompanhando as setas, até o ponto $-B_m$.

Se uma corrente suficiente para fazer com que o núcleo fique saturado for aplicada ao enrolamento de entrada e depois removida, a densidade de fluxo através do núcleo reverterá ou para o ponto $+B_r$ ou para o ponto $-B_r$, dependendo da polaridade da corrente. Estes dois pontos de operação se denominam *pontos remanentes*, e, quando o fluxo através do núcleo estiver em qualquer um destes pontos, o núcleo lembra um pequeno ímã permanente. Se um núcleo estiver no ponto $+B_r$ do gráfico e for aplicada uma corrente de amplitude suficiente para produzir uma força de $-H_m$, o fluxo produzido estará em oposição ao fluxo remanente no núcleo, $+B_r$, e o fluxo através do núcleo será invertido, deslocando o ponto de operação para $-B_m$. Depois de ter sido retirada a corrente, a maior parte do fluxo permanecerá, e a densidade de fluxo no núcleo estará no ponto $-B_r$ da curva. O fato de que o núcleo possa estar em qualquer um de dois únicos estados de magnetização torna possível considerar que um estado representa um 0 ($-B_r$ na figura) e que o outro estado representa um 1 ($+B_r$ na figura).

Um outro enrolamento, chamado *enrolamento de leitura* [Fig. 7.5*(b)*], é usado para determinar se um núcleo contém um 0 ou um 1. Para ler (perceber) o estado do núcleo, é aplicada ao enrolamento de entrada uma corrente suficiente para produzir uma força $-H_m$ magnetizante. Se o núcleo estiver em $+B_r$, o ponto de operação será deslocado para baixo ao longo das setas na Fig. 7.4 até o ponto $-B_m$. Se o núcleo estiver no ponto $-B_r$, o ponto de operação será deslocado horizontalmente até $-B_m$. O valor da variação no fluxo no núcleo será bem diferente para cada um desses casos. Se o núcleo esteve inicialmente no estado 1 (no ponto $+B_r$ da curva de histerese), o sentido da magnetização do núcleo será alterado, e o fluxo através do núcleo variará de $+B_r$ a $-B_m$. A magnitude desta variação na magnetização do núcleo é indicada como $+B_r$ a $-B_m$ na figura. Se, no entanto, o núcleo esteve originalmente no estado 0 em $-B_r$ sobre a curva, a variação na magnetização será pequena, somente de $-B_r$ a $-B_m$, e a magnitude desta variação no fluxo é indicada como $-B_r$ a $-B_m$ na figura. Uma variação na densidade de fluxo através do núcleo induzirá uma tensão, no enrolamento de leitura, proporcional à taxa de variação de fluxo. Se o núcleo esteve originalmente no estado 0, a variação e a taxa de variação no fluxo serão pequenas, e, portanto, a tensão induzida no enrolamento de leitura será pequena. Se o núcleo esteve originalmente no estado 1, o valor da variação e a taxa de variação no fluxo serão grandes,

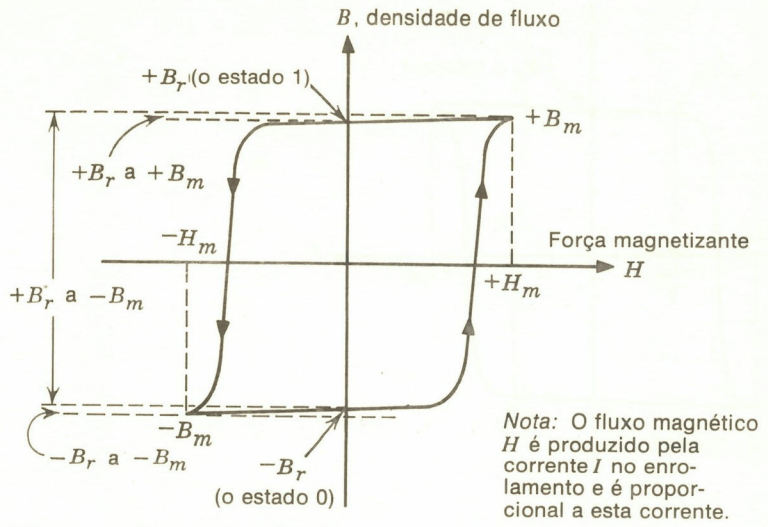


Fig. 7.4 Laço de histerese de um material magnético.

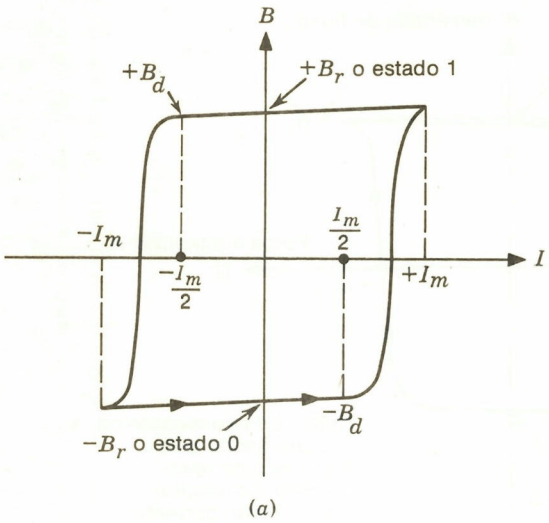
e, conseqüentemente, a tensão induzida no enrolamento de leitura será grande. Uma pequena saída do enrolamento de leitura indicará portanto que o núcleo continha um 0 originalmente, e uma saída grande indicará que o núcleo continha originalmente um 1.

A dificuldade no emprego desta técnica de leitura (percepção) do estado de um núcleo reside em que o núcleo é sempre reajustado para o estado 0. Esta técnica de leitura é portanto "destrutiva" no sentido de que o núcleo não contém mais a informação que esteve previamente armazenada nele. Geralmente é desejável reter informação em armazenamento depois de ela ter sido lida. Uma vez que a leitura de um núcleo é destrutiva, é necessário gravar o estado original do núcleo novamente nele. Detectando e amplificando a saída do enrolamento de leitura quando o estado do núcleo for lido, é possível usar o sinal para ajustar um dispositivo de armazenamento com biestáveis ou de outro tipo temporário e apagável. Os conteúdos do biestável podem em seguida ser gravados novamente no núcleo através do enrolamento de entrada, retornando o núcleo a seu estado original.

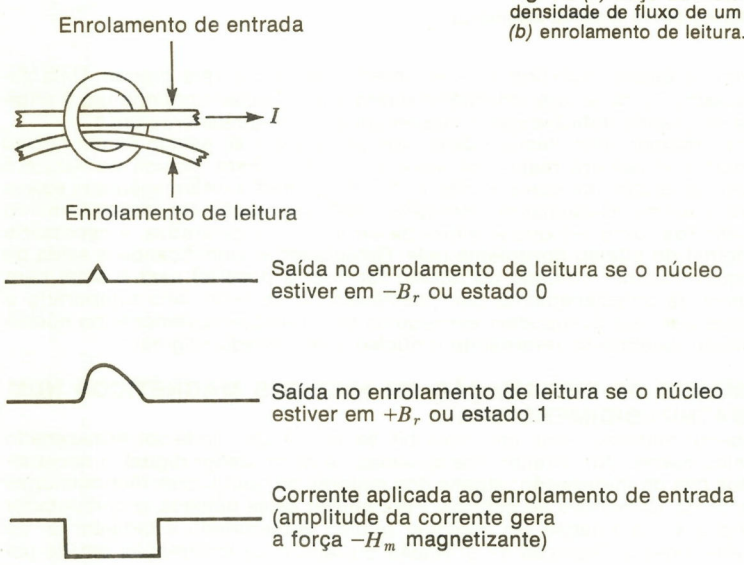
7.3 ARMAZENAMENTO DE INFORMAÇÃO EM NÚCLEOS MAGNÉTICOS NUM ARRANJO (MATRIZ) BIDIMENSIONAL

A discussão precedente mostrou como um único bit de informação pode ser armazenado em e lido de um único núcleo. No entanto, nos sistemas de computador digital, é necessário armazenar muitos bits de informação; alguns dos maiores computadores têm memórias de núcleos com cerca de 10 milhões de núcleos. Para utilizar estes núcleos, o computador deve ser capaz de gravar informação em qualquer núcleo determinado e também ler de qualquer núcleo determinado. Técnicas de gravação ou leitura de informação em/de um núcleo selecionado ou grupo de núcleos numa memória são conhecidas como *técnicas de seleção*. A técnica de seleção que será descrita é chamada *técnica de seleção de corrente de coincidência* e constitui a técnica mais comumente usada em sistemas de computadores de grande porte.

A Fig. 7.5*(a)* "plota" os valores da densidade de fluxo de um núcleo magnético *versus* corrente I através do enrolamento de entrada de um núcleo. Se a corrente alcançar um valor de $+I_m$, que é suficiente para saturar o núcleo, e for depois removida, o núcleo será colocado no $+B_r$, ou estado 1. Se o ponto de operação sobre a curva estiver em $+B_r$ sem nenhuma corrente de entrada através do enrolamento, e um pulso negativo de corrente for aplicado com amplitude $-I_m$, o núcleo será comutado para o estado 0. O fato importante a observar na Fig. 7.5 é que, se o núcleo estiver no estado 1 e uma corrente de apenas $-I_m/2$ for aplicada e depois removida, o estado do núcleo não será alterado. Em vez disso, o ponto de operação se deslocará de $+B_r$ a $+B_d$ e, quando a corrente for removida, se deslocará de volta para aproximadamente $+B_r$. O mesmo princípio se aplica se o núcleo estiver



(a)

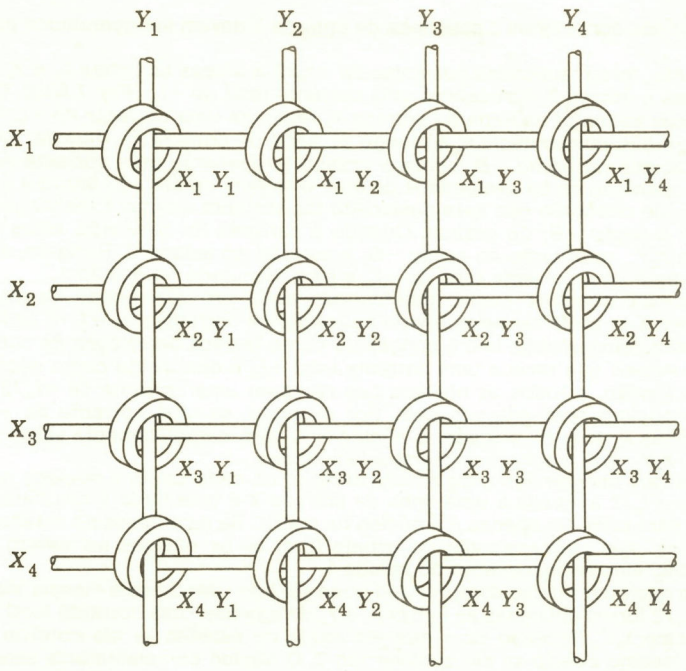


(b)

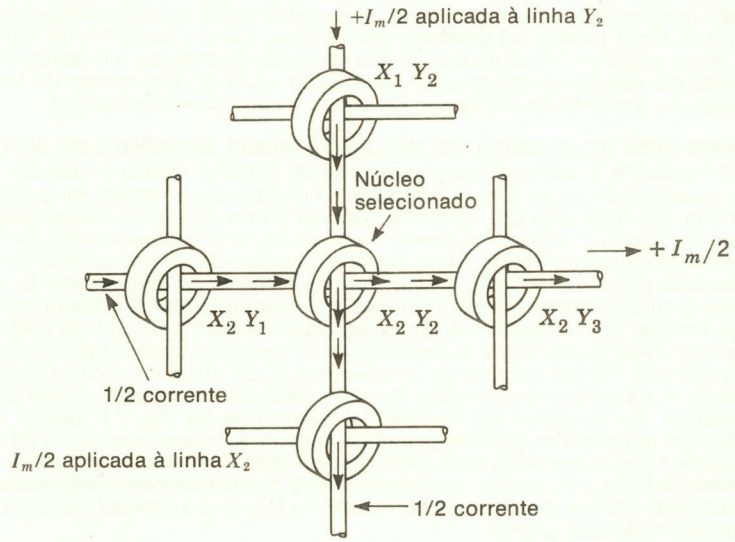
Fig. 7.5 *(a)* Laço de histerese corrente *versus* densidade de fluxo de um material magnético; *(b)* enrolamento de leitura.

no estado 0 no ponto $-B_r$ sobre a curva. Uma corrente de $+I_m$ comutará o núcleo para o estado 1, mas uma corrente de $+I_m/2$ apenas deslocará o ponto de operação para $-B_d$ e, quando a corrente for removida, o núcleo retornará ao ponto $-B_r$.

A Fig. 7.6*(a)* mostra 16 núcleos arranjados numa matriz quadrada. Cada núcleo tem dois enrolamentos de entrada, um de um conjunto de X linhas de entrada e um de um conjunto de Y linhas de entrada. Admitamos que todos os núcleos estejam no estado 0 e que um 1 deva ser gravado no núcleo X_2Y_2. Se uma corrente com uma amplitude de $+I_m$ for aplicada à linha X_2, o núcleo X_2Y_2 será comutado para o estado 1; no entanto, todos os núcleos conectados à linha de entrada X_2 também serão comutados ao estado 1. O mesmo se mantém verdadeiro se uma corrente de $+I_m$ for aplicada à linha Y_2 de entrada, caso em



(*a*)



(*b*)

Fig. 7.6 Plano de núcleos bidimensional. *(a)* Fiação; *(b)* seleção de X_2Y_2.

que todos os núcleos conectados a esta linha de entrada Y devem ser comutados ao estado 1.

Se, no entanto, forem aplicadas correntes de $+I_m/2$ a ambas as linhas X_2 e Y_2 simultaneamente, apenas o núcleo X_2Y_2 receberá uma corrente total de $+I_m$ [Fig. 7.6*(b)*]. Cada um dos outros núcleos ao longo da linha de entrada Y_2 receberá uma corrente de $+I_m/2$, como receberá cada um dos outros núcleos ao longo da linha de entrada X_2. Uma vez que estes núcleos terão iniciado no ponto $-B_r$ sobre o gráfico e receberão uma corrente de $+I_m/2$, seus pontos de operação serão deslocados de uma distância de até $-B_d$ sobre a curva; no entanto, o ponto de operação não será deslocado tão distante quanto a inclinação excessiva que conduz à seção $+B_r$ do gráfico. Quando a corrente for removida, estes núcleos, com exceção de X_2Y_2, retornarão ao ponto $-B_r$ e estarão no estado 0. Portanto, apenas o núcleo X_2Y_2 receberá uma corrente total de $+I_m$ e será comutado no estado 1.

Por esta técnica, qualquer núcleo determinado pode ser selecionado da matriz. Por exemplo, o núcleo X_1Y_3 pode ser selecionado aplicando-se correntes de $+I_m/2$ às linhas de entrada X_1 e Y_3 simultaneamente. Isto é conhecido como técnica de seleção de corrente de coincidência. O núcleo que recebe uma corrente total $+I_m$ é conhecido como núcleo *completamente selecionado,* e todos os núcleos que recebem uma corrente de $+I_m/2$ são conhecidos como núcleos *semi-selecionados.* Por exemplo, se uma corrente de $+I_m/2$ for conectada às linhas X_1 e Y_3 de entrada, o núcleo X_1Y_3 será completamente selecionado, e os núcleos X_1Y_1, X_1Y_2, X_2Y_3 e X_3Y_3 serão semi-selecionados.

Observemos que independente do estado anterior de cada um dos núcleos na matriz, uma corrente de $+I_m/2$ aplicada a uma linha de entrada X e uma linha Y resultará em corrente suficiente para comutar apenas um núcleo da matriz. Se os núcleos no estado 0 forem semi-selecionados, eles permanecerão no estado 0; e, se os núcleos no estado 1 forem semi-selecionados, eles permanecerão no estado 1.

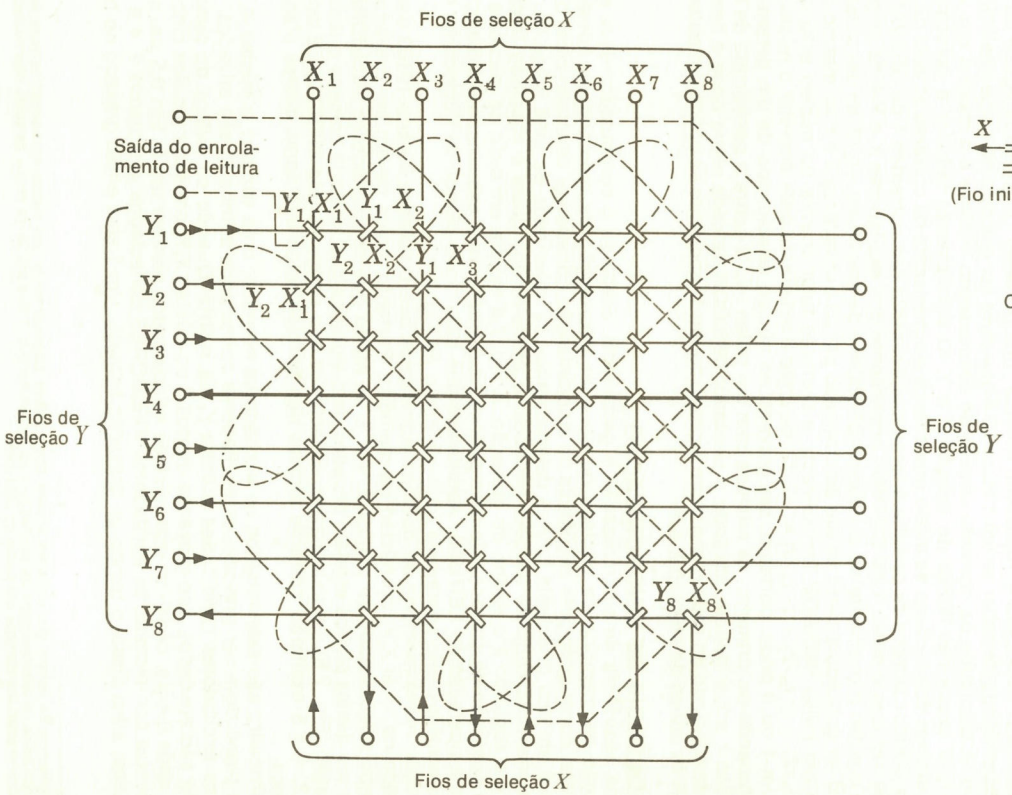
O estado de qualquer núcleo determinado pode ser lido usando-se a mesma técnica. Se correntes de $-I_m/2$ forem aplicadas às linhas X_2 e Y_2 de entrada, uma corrente total $-I_m$ será aplicada ao núcleo X_2Y_2, fazendo com que ele mude de estados se ele contiver um 1 e permanecer no mesmo estado se ele contiver um 0. O núcleo completamente selecionado será o único núcleo que pode mudar de estados e, portanto, o único núcleo que é capaz de produzir uma saída apreciável num enrolamento de leitura. Um enrolamento de leitura deve estar em espiras através de todos os núcleos; se um certo núcleo for selecionado, a saída no enrolamento de leitura representará o estado daquele núcleo somente.

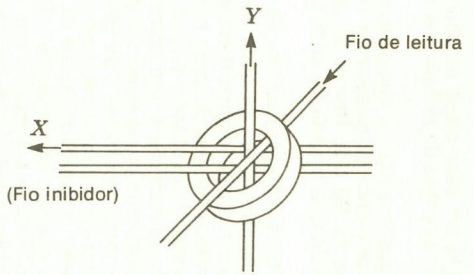
Pode ser visto que há duas operações separadas envolvidas: primeira, a gravação de informação num núcleo, e segunda, a leitura de informação que foi armazenada num núcleo. Um 0 ou um 1 podem ser gravados em qualquer núcleo na matriz aplicando-se uma corrente de ou $-I_m/2$ ou $+I_m/2$ ao par correto de linhas de entrada, e o estado de qualquer núcleo pode ser lido aplicando-se uma corrente de $-I_m/2$ ao par correto de linhas de entrada e lendo-se a magnitude da tensão de saída no enrolamento de leitura.

7.4 MONTAGEM DE PLANOS DE NÚCLEOS NUMA MEMÓRIA DE NÚCLEOS

A Fig. 7.7 ilustra um pequeno *plano de memória de núcleos.* Qualquer núcleo neste plano pode ser selecionado, e um 0 ou 1 gravados no núcleo, ou o estado de qualquer núcleo pode ser lido, por aplicação das correntes corretas a uma das linhas de entrada X e a uma das linhas de entrada Y simultaneamente. Uma memória de núcleos completa consiste de vários de tais planos, empilhados numa matriz retangular. Os enrolamentos X de cada plano são conectados em série, de modo que a corrente aplicada ao enrolamento X_1 do primeiro plano deve progredir através do enrolamento X_1 do segundo plano, e assim por diante, até que passe através do enrolamento X_1 do último plano na matriz. Se o plano ilustrado na Fig. 7.7 fosse conectado numa matriz, o enrolamento X_1 na parte superior esquerda do desenho seria conectado a uma extremidade do enrolamento X_1 do plano de núcleos precedente, e o enrolamento X_1 na parte inferior esquerda seria conectado à conexão X_1 do plano de núcleos seguinte. Se 10 planos de núcleos das dimensões na Fig. 7.7 fossem empilhados dessa maneira, um pulso na linha de entrada X_1 teria de progredir através de 80 núcleos. Os enrolamentos Y são conectados do mesmo modo. Cada plano, no entanto, tem seu próprio enrolamento de leitura, e os enrolamentos de leitura dos planos não são de maneira alguma interconectados; em vez disso, um *amplificador de leitura* é conectado à saída do enrolamento de leitura de cada plano.

Designemos um pulso de corrente de $+I_m/2$ por pulso 1 de GRAVAR, pois tal corrente aplicada a um enrolamento X ou Y tende a gravar um 1 nos núcleos ao longo do enrolamento. Pode ser visto que aplicando-se simultaneamente um pulso 1 de GRAVAR numa linha Y selecionada e numa linha X selecionada, gravar-se-á um 1 num único núcleo na mesma posição relativa em cada um dos planos. Por exemplo, se as linhas de entrada X_3 e





O detalhe mostra um núcleo típico

Fig. 7.7 Plano de memória de núcleos.

Y_4 forem pulsadas com uma corrente de $+I_m/2$, o núcleo X_3Y_4 em cada plano receberá uma corrente $+I_m$. Se um pulso $-I_m/2$ for então aplicado às mesmas linhas de seleção X e Y, uma saída será lida no enrolamento de leitura† de cada plano.

Em geral, haverá tantos planos de núcleos na matriz quantos bits no comprimento da palavra da memória em que a matriz for usada. Se o comprimento da palavra for 15 bits, haverá 15 planos na matriz; se o comprimento da palavra for 35 bits, haverá 35 planos. Haverá um enrolamento de leitura para cada plano, de modo que a saída de cada núcleo que é selecionado será lida.

Permanece ainda um problema: um pulso $(+I_m/2)$ 1 de GRAVAR sobre um par de linhas de entrada X e Y gravará um 1 no núcleo na mesma posição relativa de cada plano e, portanto, em cada posição da palavra, e pode ser necessário gravar dígitos 0 em alguns núcleos e dígitos 1 em outros núcleos. Um quarto enrolamento através de cada núcleo é portanto acrescentado. Este enrolamento é chamado *enrolamento inibidor* e é usado para inibir a gravação de um 1 num determinado núcleo selecionado num plano. Um único enrolamento inibidor é colocado em espiras através de cada núcleo num plano numa direção de modo que uma corrente de $-I_m/2$ sobre o enrolamento se oporá à corrente 1 de GRAVAR. Há, portanto, tantos enrolamentos inibidores quantos planos de núcleos. Há também um único excitador para cada enrolamento inibidor, que pode ser comandado ligado ou desligado, dependendo de saber se um 1 ou um 0 é que deve ser gravado no núcleo selecionado do plano. A amplitude da corrente através do enrolamento inibidor é a mesma que a corrente $-I_m/2$ usada nas linhas de seleção X e Y. O enrolamento inibidor é colocado em espiras através dos núcleos de modo que a força magnetizante da corrente através do enrolamento inibidor sempre se oponha à força magnetizante das correntes 1 de GRAVAR.

Uma vez que o pulso 1 de GRAVAR proveniente dos excitadores X e Y é de amplitude $+I_m/2$ e o pulso proveniente do excitador INIBIDOR é de amplitude $-I_m/2$, se um pulso 1 de GRAVAR for aplicado a partir de um determinado par de excitadores X e Y ao mesmo tempo quando um pulso INIBIDOR for aplicado, a corrente total através do núcleo selecionado será $(+I_m/2) + (+I_m/2) + (-I_m/2) = +I_m/2$, que não é suficiente para comutar o núcleo para um 1. Um 1 ou 0 pode, portanto, ser gravado no núcleo selecionado de um determinado plano limpando-se, primeiro, os núcleos com um pulso $-I_m/2$ sobre o par correto de linhas de seleção X e Y e, depois, ativando o excitador INIBIDOR de cada plano onde é desejado um 0 e desativando de cada plano onde é desejado um 1, enquanto se aplica um pulso 1 de GRAVAR às linhas de seleção X e Y.

7.5 SEQÜÊNCIA DE SINCRONIZAÇÃO

A mesma seqüência de sincronização é geralmente usada se o computador deve gravar informação na memória de núcleos ou deve ler informação da memória de núcleos. O tempo total consumido pela seqüência de sincronização completa é chamado *ciclo de memória* e é um dos principais fatores determinantes da velocidade de uma máquina. Cada ciclo de memória consiste de dois períodos, o primeiro dos quais é chamado tempo de leitura, e o segundo é o tempo de gravação. A Fig. 7.8 mostra um ciclo de memória completo. A operação de gravação de uma palavra na memória será examinada de início, e, em seguida, a operação de LEITURA. Admitamos que haja cinco planos de núcleos na matriz total e que cada plano contém 64 núcleos, como na Fig. 7.7.

1. *Gravação numa memória de núcleos* Admitamos que o número binário 10100 deva ser gravado no local X_3Y_4 do núcleo da memória. Portanto, zeros devem ser gravados nos núcleos selecionados de planos 2, 4 e 5, de modo que os excitadores INIBIDORES de cada um destes planos estejam ativados; isto significa que, quando começa o tempo de gravação (Fig. 7.8), os excitadores conectados aos enrolamentos INIBIDORES dos planos 2, 4 e 5 excitarão corrente através dos enrolamentos inibidores destes planos de núcleos.

   Primeiro são selecionados os excitadores corretos; uma vez que o núcleo X_3Y_4 em cada plano deve ser gravado, os excitadores conectados à linha X_3 são ativados, como são os excitadores conectados à linha Y_4 da matriz. Quando os pulsos do *tempo de leitura* são aplicados, são gravados dígitos 0 no núcleo X_3Y_4 em cada um dos cinco planos; após 0,4 μs, cada um dos núcleos selecionados conterá um 0. Em 0,5 μs, após a seqüência ser iniciada, os excitadores INIBIDORES conectados aos planos 2, 4 e 5 são ativados; além disso, o tempo de gravação começa. Um 1 será então gravado no núcleo

†Os enrolamentos de leitura são enfiados através dos núcleos diagonalmente aos outros enrolamentos para reduzir o acoplamento entre os enrolamentos de excitação e os de leitura (Fig. 7.7) e anular os sinais de saída dos núcleos semi-selecionados produzidos durante um pulso de LEITURA.

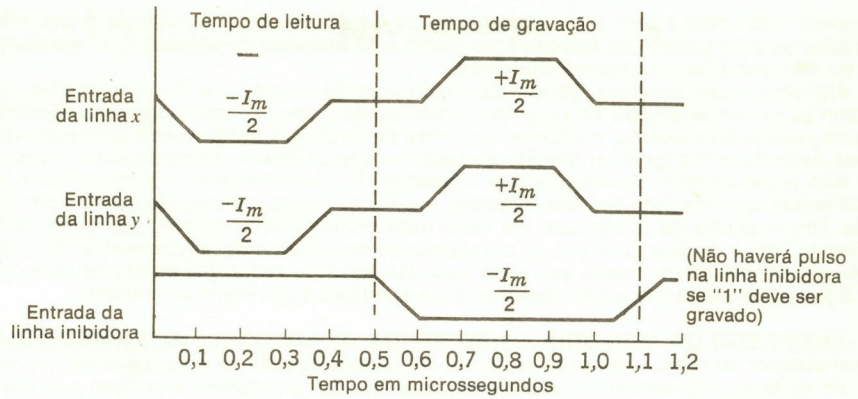


Fig. 7.8 Seqüência de sincronização da memória de núcleos.

selecionado nos planos 1 e 3, e a subtração da corrente INIBIDORA da corrente coincidente através dos núcleos selecionados nos planos 2, 4 e 5 resultará nos núcleos selecionados nestes planos que permanecem no estado 0. Após a seqüência de pulsos, os núcleos selecionados nos planos 1 e 3 conterão dígitos 1, os núcleos selecionados nos planos 2, 4 e 5 conterão dígitos 0 e a palavra 10100 de computador terá sido armazenada na memória no local X_3Y_4.

2. *Leitura de uma memória de núcleos* Admitamos que, num instante de tempo mais tarde, o local X_3Y_4, que foi gravado na operação de GRAVAR acima, deva ser lido. A seqüência de sincronização ilustrada na Fig. 7.8 ainda se aplica, embora haja várias diferenças na operação. Primeiro, as correntes de LER são aplicadas aos núcleos selecionados (desde o instante 0 até 0,5 μs na Fig. 7.8). Se for recebido um grande sinal no amplificador de leitura conectado a um determinado plano durante este período, o núcleo selecionado naquele plano continha um 1; se um pequeno sinal for recebido, o núcleo selecionado no plano continha um 0. Os enrolamentos de leitura conectados aos planos 1 e 3 produzirão, portanto, sinais indicando 1, e os enrolamentos de leitura conectados aos planos 2, 4 e 5 produzirão sinais pequenos, indicando 0. A palavra armazenada foi, portanto, 10100. Esta palavra deve agora ser gravada novamente na matriz da memória. A saída de cada enrolamento de leitura é amplificada e usada para ajustar um dispositivo de armazenamento no estado 0 ou 1 durante o tempo de leitura, e os conteúdos de cada um destes dispositivos de armazenamento são depois usados para controlar os excitadores do enrolamento INIBIDOR durante o tempo de gravação. Somente os excitadores INIBIDORES conectados aos planos 2, 4 e 5 serão portanto ativados pelos sinais dos respectivos dispositivos de armazenamento e conduzirão durante o tempo de gravação, e os núcleos selecionados nestes planos permanecerão no estado 0. Após o tempo de gravação, os núcleos selecionados conterão novamente 10100, tal como antes da operação de LER.

7.6 CARACTERÍSTICAS DA MATRIZ DA MEMÓRIA DE NÚCLEOS

Cada núcleo de uma memória de núcleos do tipo descrito é atravessado por quatro enrolamentos, como na Fig. 7.7.† Uma vez que os núcleos são geralmente bem pequenos, talvez 12/1.000 a 50/1.000 pol. ou milésimo de pol.* de diâmetro externo, o tamanho total das matrizes pode ser feito muito pequeno. Cada plano de uma das memórias usadas no computador TX-2 no Laboratório Lincoln é um plano de núcleos de 64 × 64 e, portanto, contém 4096 núcleos; há 37 planos na matriz, e portanto, um total de 151.552 núcleos na matriz

†Os enrolamentos mostrados na Fig. 7.7 estão numa forma comumente usada. Ela tem as vantagens de acoplamento mínimo entre os enrolamentos de excitação e os enrolamentos de leitura, cancelamento ótimo dos ruídos da semiseleção, e distribuição uniforme dos circuitos e dos terminais de excitação perto do plano de memória.

*As dimensões do núcleo são geralmente dadas em milésimos de polegada assim: um diâmetro externo de 50 milésimos de pol., núcleo de diâmetro interno de 30 milésimos de pol., ou, em linguagem de computadores OD de 50 milésimos de pol., núcleo ID de 30 milésimos de pol.

completa. Esta matriz tem aproximadamente as dimensões de um cubo de 5 pol. Há 64 entradas de enrolamento de seleção *X* na matriz e 64 entradas de seleção *Y*, 37 enrolamentos de leitura e 37 enrolamentos inibidores.

Algumas características gerais das memórias de núcleo de corrente coincidente tornam-se evidentes a partir do seguinte: o número de planos usados é igual ao número de bits na palavra da memória; o número de linhas de entrada *X* multiplicado pelo número de linhas de entrada *Y* é igual ao número de palavras que pode ser armazenado na matriz.

Nem todas as memórias de núcleos consistem de núcleos individuais montados e individualmente atravessados por fios. Algumas memórias utilizam placas de ferrite com aberturas, isto é, placas de ferrite com um certo número de orifícios através dos quais podem passar as linhas de seleção *X* e *Y*, os enrolamentos de leitura e os enrolamentos inibidores. A placa é construída de modo que cada uma das áreas ao redor das aberturas atue como um simples núcleo; a placa tem características similares a um plano de núcleos.

7.7 ENDEREÇO DA MEMÓRIA E REGISTROS DE MEMÓRIA INTERMEDIÁRIA

O computador se comunica com a memória de núcleos através de dois registros: (1) *registro de endereço da memória*, que contém o endereço da palavra que deve ser lida ou gravada, e (2) *registro da memória intermediária*, que contém a palavra que foi lida da memória (após um ciclo de LER) ou a palavra que deve ser gravada na memória. Para

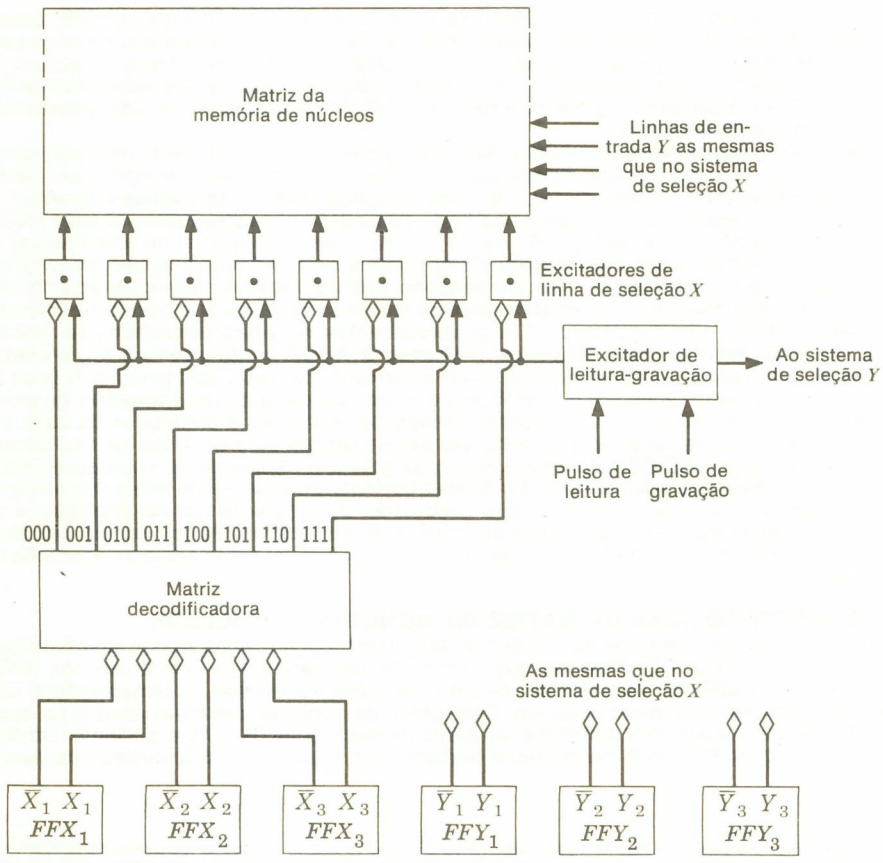


Fig. 7.9 Sistema de seleção *X* e *Y*.

localizar uma palavra na memória, o endereço da palavra é primeiro carregado no registro de endereço da memória. Se a memória deve ser lida, os estados dos núcleos no endereço especificado serão lidos e gravados no registro da memória intermediária. O computador pode, em seguida, ler a palavra que foi procurada ou consultada no registro de memória intermediária. Se uma palavra deve ser gravada na memória, o endereço que deve ser gravado é carregado no registro de endereço da memória, e a palavra que deve ser gravada na memória é carregada no registro de memória intermediária. Após o ciclo de memória, a palavra será armazenada na memória no endereço especificado.

7.8 EXCITAÇÃO DAS LINHAS DE SELEÇÃO *X* E *Y*

Examinemos a colocação em seqüência das correntes através de uma memória de núcleos que usa um simples modelo idealizado. Os biestáveis são geralmente usados como dispositivos de armazenamento tanto no registro de endereço da memória quanto no registro de memória intermediária. A Fig. 7.9 ilustra em forma de diagrama-bloco a operação de um sistema de seleção de uma memória de núcleos, tendo cada plano as dimensões ilustradas na Fig. 7.7. Uma vez que há oito linhas de seleção *X* e oito linhas de seleção *Y*, serão necessários três biestáveis para selecionar uma linha *X* e três outros para selecionar uma linha *Y*. Há, portanto, seis biestáveis no registro de endereço da memória, de modo que qualquer um dos 64 (2^6) locais em cada plano pode ser selecionado. Os biestáveis no registro ilustrado foram designados X_1, X_2, X_3 e Y_1, Y_2, Y_3 para indicar que os três primeiros são usados para selecionar o enrolamento *X* e os três seguintes para selecionar o enrolamento *Y*.

1. *Matriz decodificadora* As saídas dos biestáveis X_1, X_2 e X_3 na Fig. 7.9 são conectadas a um *decodificador*. Este estágio é uma configuração freqüentemente usada, possuindo várias linhas de entrada e de saída e a característica de que, para cada combinação de valores de entrada, apenas uma linha de saída representará um 1. A Fig. 7.10 ilustra um decodificador constituído de portas E. Há quatro linhas de entrada no decodificador: duas de um biestável designado *A* e duas de um biestável designado *B*. Os dois biestáveis podem estar em qualquer um de quatro estados possíveis. Por exemplo, o biestável *A* pode conter um 0, e o *B*, um 1, em cujo caso a entrada $\bar{A}$ será "alta" e a entrada *B* será "alta". Neste caso, a linha de saída $\bar{A}B$ será alta (representando um 1) e as outras três linhas de saída serão "baixas", ou 0.

   Este tipo de decodificador é muitas vezes denominado *decodificador de uma ou várias funções* (várias entradas e uma única saída) porque as entradas selecionam uma das linhas de saída. Se tivermos entradas de três biestáveis, como na Fig. 7.9, e designarmos os biestáveis por FFX_1, FFX_2, FFX_3, os estados dos biestáveis poderão ser gravados como um número binário tal como 011, significando que FFX_1 contém um 0, FFX_2 contém um 1 e FFX_3 contém um 1. Podemos então designar cada linha de saída por um número binário (Fig. 7.10), e a linha de saída que será alta corresponderá ao número de entrada representado pelo estado dos biestáveis de entrada.

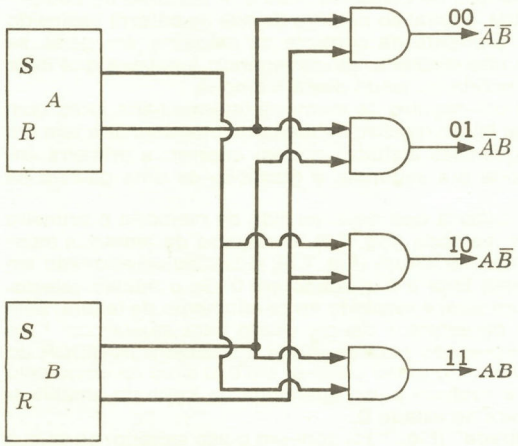


Fig. 7.10 Decodificador de quatro saídas.

Uma vez que há três biestáveis *X* na Fig. 7.9, os biestáveis podem assumir qualquer um dos oito estados, representando de 000 a 111. Para cada estado da entrada, somente uma das saídas do decodificador será alta, e, portanto, apenas um dos excitadores da linha de seleção *X* (Fig. 7.9) será ativado.

Se o decodificador for grande, pode ser mais econômico usar outras configurações diferentes da indicada, mas a função destes decodificadores é a mesma: uma linha de saída é selecionada para cada estado da entrada.

2. *Excitadores da linha de seleção X* Na Fig. 7.9 há oito excitadores da linha de seleção *X*, cada um dos quais tem duas entradas, uma do decodificador e outra do excitador de LER-GRAVAR. As correntes de saída do excitador de LER-GRAVAR podem ser negativas ou positivas; os excitadores de linha de seleção *X* são capazes portanto de transmitir corrente em qualquer direção. Os excitadores de linha de seleção *X* também funcionam como portas E, pois eles transmitirão corrente apenas quando o sinal de entrada proveniente da matriz decodificadora representa um 1. Apenas um dos excitadores de linha de seleção *X* será ativado pelo decodificador em qualquer instante de tempo determinado. Por exemplo, se os biestáveis *X* no registro de endereço da memória contiverem 000, somente o excitador mais à esquerda na figura será ativado. Todos os excitadores de linha de seleção *X* receberão um pulso de corrente LER e depois um de GRAVAR (de polaridade oposta) originários do excitador de LER-GRAVAR. Já que somente um excitador de linha de seleção *X* é ativado por um sinal 1 do decodificador, somente um dos enrolamentos *X* na matriz receberá as correntes de excitação. O mesmo se mantém verdadeiro no sistema de linha de seleção *Y*, que é idêntico ao sistema *X*. Uma vez que apenas uma linha *X* e apenas uma linha *Y* são pulsadas durante cada ciclo da memória, apenas um núcleo em cada plano receberá um pulso completamente selecionado de LER e GRAVAR.
3. *Excitador de LER-GRAVAR* Este excitador recebe dois pulsos de relógio em cada ciclo da memória. O primeiro é o pulso de LER, que faz com que o excitador de LER-GRAVAR forneça um pulso positivo SEMI-SELECIONADO de corrente para ambos os excitadores de linha de seleção *X* e *Y*. O segundo pulso de entrada para o excitador LER-GRAVAR é o pulso de GRAVAR, que inicia um pulso negativo de corrente para os excitadores de linha de seleção *X* e *Y*. A corrente fornecida pelos excitadores de LER-GRAVAR é geralmente da ordem de 0,1 a 1A.

A operação do registro de endereço da memória e de seu conjunto de circuitos associado faz com que um único núcleo em cada plano da memória receba um pulso de corrente de leitura completa e depois um pulso de corrente de gravação completa. O registro de endereço da memória é geralmente carregado diretamente da instrução que está sendo interpretada pela máquina. Isto significa que a seção de endereço da palavra é lida diretamente no registro de endereço da memória. (Isto será descrito no Cap. 9.)

7.9 REGISTRO DE MEMÓRIA INTERMEDIÁRIA E CONJUNTO DE CIRCUITOS ASSOCIADO

O registro de memória intermediária consiste de um registro de biestáveis que contém um biestável para cada plano na matriz. O número de planos na matriz, e portanto no comprimento do registro de memória intermediária, é igual ao número de bits na palavra básica do computador. O elemento aritmético e o elemento de controle da máquina, em geral, se comunicam diretamente com a memória intermediária, ou carregando a palavra que deve ser gravada em armazenamento nesta memória ou lendo desta memória.

A Fig. 7.11 ilustra dois biestáveis de um registro de memória intermediária junto com seu conjunto de circuitos associado. Cada bit no registro de memória intermediária tem um conjunto de circuitos idêntico. Duas seqüências distintas podem ocorrer: a primeira envolve a leitura de uma palavra da memória e a segunda, a gravação de uma palavra na memória.

1. *Leitura da memória* A localização da palavra que deve ser lida da memória é primeiro carregada no registro de endereço da memória (Fig. 7.9). O registro da memória intermediária é então limpo. Durante o tempo de leitura (Fig. 7.8), o núcleo selecionado em cada plano recebe um pulso de corrente total e é ajustado em 0. Se o núcleo selecionado num certo plano contiver um 1, um sinal é recebido no enrolamento de leitura. Este sinal é amplificado pelo amplificador de leitura e depois usado para ajustar um 1 no biestável da memória intermediária empregado para controlar o excitador INIBIDOR do plano de núcleos. Se o núcleo selecionado no plano contiver um 0, o sinal recebido pelo amplificador de leitura será pequeno e nenhum pulso aparecerá na saída do amplificador de leitura, e o biestável permanecerá no estado 0.

O amplificador de leitura é "amostrado" (Fig. 7.11) com um pulso estreito, porque, a

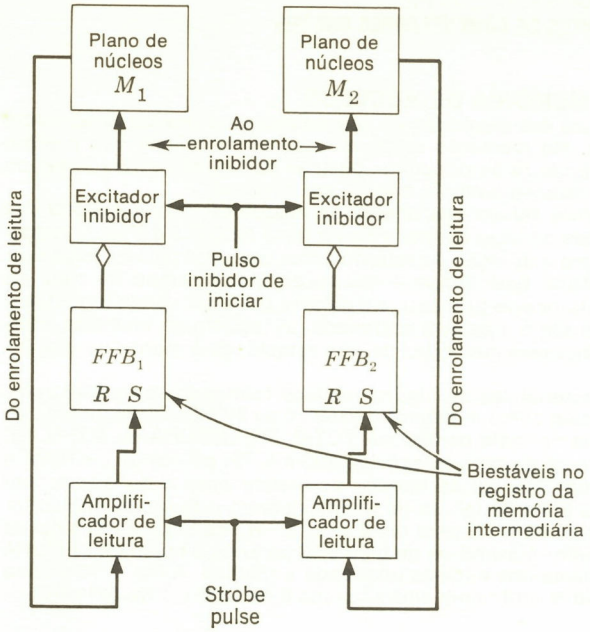


Fig. 7.11 Operação de registro da memória intermediária.

despeito do fato de que um núcleo que contém um 0 produzirá um pequeno sinal quando "lido", muitos dos núcleos em cada plano receberão pulsos semi-selecionados e cada um gerará uma pequena quantidade de ruído no enrolamento de leitura. Se este ruído for aditivo, ele pode aproximar-se do nível do sinal de um núcleo cuja polaridade esteja invertida. Verificou-se que o ruído gerado pelos núcleos semi-selecionados desaparece rapidamente após serem iniciados os pulsos semi-selecionados; portanto, o amplificador de leitura é amostrado durante a última parte do tempo de leitura, quando a saída de um núcleo selecionado invertendo os estados é mais alta em proporção ao ruído dos núcleos semi-selecionados. Somente o sinal de saída durante o pulso de amostragem é comandado na linha de AJUSTE do biestável.

Durante o tempo de gravação (Fig. 7.8), cada um dos núcleos selecionados recebe uma corrente total selecionada do sistema de seleção *X* e *Y*. Em conseqüência da operação de leitura, todos esses núcleos tinham sido previamente ajustados no estado 0; no entanto, os biestáveis da memória intermediária contêm agora dígitos 1 para cada plano em que um 1 foi armazenado no núcleo completamente selecionado. As saídas desses biestáveis são usadas para ativar os excitadores INIBIDORES de cada plano em que um 0 deve permanecer. Todos os outros núcleos completamente selecionados serão ajustados para o estado 1. A saída 0 de cada biestável é, portanto, usada para ativar o excitador INIBIDOR do plano associado ao biestável. Depois do tempo de gravação, o núcleo selecionado em cada plano estará em seu estado original.

2. *Gravação na memória de núcleos* O sistema de seleção *X* e *Y* atravessa a mesma seqüência de sincronização quando uma palavra deve ser gravada na memória e também quando uma palavra é lida da memória. A operação do registro da memória intermediária, no entanto, é diferente. A palavra a ser gravada no local selecionado é carregada no registro da memória intermediária, e o endereço na memória a ser gravado é carregado no registro de endereço da memória. Durante o tempo de leitura, o amplificador de leitura não é amostrado; portanto, o registro da memória intermediária não é perturbado.

Após o tempo de leitura, todos os núcleos selecionados terão sido ajustados em 0. Durante o tempo de gravação, os mesmos núcleos receberam todos pulsos de GRAVAR de corrente total oriundos do sistema de seleção *X* e *Y;* no entanto, os planos em que os 0 devem permanecer têm seus excitadores inibidores ativados pelos biestáveis da memória intermediária (Figs. 7.9 e 7.11), e os pulsos de meia-corrente provenientes destes excitadores anulam a corrente de seleção, deixando os núcleos completamente selecionados no estado 0. Todos os outros núcleos completamente selecionados são ajustados para 1.

Após o tempo de gravação, a palavra no registro da memória intermediária terá sido gravada nos núcleos.

7.10 CARACTERÍSTICAS DA MEMÓRIA DE NÚCLEOS

A pequena memória descrita é típica das memórias de acesso aleatório usadas atualmente, exceto quanto a suas dimensões. No momento estão em uso as memórias que contêm cerca de 10 milhões de núcleos. Embora os princípios básicos de tais memórias sejam os mesmos, o conjunto de circuitos necessariamente diferirá em detalhes.

Os tempos de ciclo de memória variam nas diferentes máquinas, mas a maioria das máquinas comerciais possui tempos de ciclo de memória na faixa de 0,275 a 2 μs. O *tempo de acesso de leitura* é definido como a duração do retardo antes que uma palavra possa ser usada durante o processo de leitura. Este tempo é menor do que o tempo de ciclo de memória, porque o tempo de leitura ocorre primeiro, e a palavra pode ser usada logo que o amplificador de leitura seja amostrado e a palavra carregada no registro da memória intermediária. O tempo de acesso de algumas memórias de alta velocidade é menor do que 0,1 μs.

O Quadro 7.1 mostra as características de alguns núcleos fabricados pela Electronic Memories, Inc. Por exemplo, o núcleo com um diâmetro externo de 30 milésimos de pol., do tipo 31-103 do fabricante, tem uma corrente de 580 mA TOTAL DE LEITURA ou TOTAL DE GRAVAÇÃO e, portanto, uma corrente de semi-seleção de 290 mA. Os pulsos de LEITURA e GRAVAÇÃO devem ser 0,5 μs, com 0,05 μs de tempos de crescimento e de queda. Isto conduz a uma saída do núcleo na linha de leitura de 61 mV quando o fluxo do núcleo for comutado de 1 a 0, *versus* uma saída de 8 mV para um 0 ($-B_r$ a $-B_m$ na Fig. 7.4). O sinal de saída da linha de leitura tem seu valor máximo ou de pico 0,23 μs após o pulso de LEITURA ser aplicado. Estas figuras são típicas dos núcleos pequenos e rápidos. A Fig. 7.12 mostra como estes parâmetros são definidos junto com sinais típicos 0 de LER e 1 de GRAVAR.

7.11 DECODIFICADORES

Uma parte importante do sistema que selecionou os núcleos a serem lidos de e gravados numa memória de corrente de coincidência é a matriz decodificadora mostrada na Fig. 7.10. Este circuito particular chamado *decodificador de uma ou várias funções, matriz decodificadora* ou simplesmente *decodificador* tem a característica de que, para cada um dos 2^n possíveis números binários de entrada que podem ser tomados pelas n células de entrada, a matriz terá uma única de suas 2^n linhas de saída selecionada.

A Fig. 7.13 mostra um decodificador que é completamente paralelo por construção e é

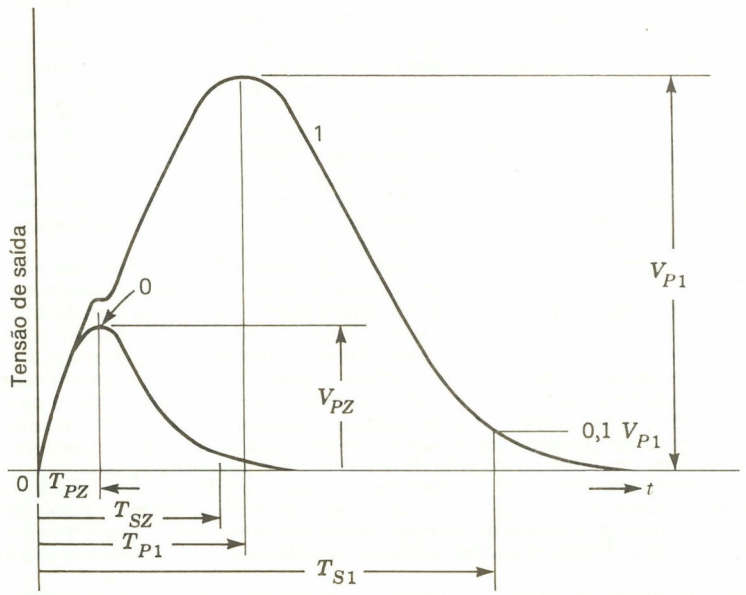


Fig. 7.12 Sinais da linha de leitura do núcleo.

QUADRO 7.1 RESUMO DAS CARACTERÍSTICAS* DE NÚCLEOS TÍPICOS

				Condições de operação típicas a 25°C						
				Características dos pulsos de excitação			Sinais de saída			
Descrição	Milésimos de polegadas	Tipo	Dimensões OD/ID/altura, milésimos de pol.	Amplitude, ma	Duração, μs	Tempos de queda e de crescimento, μs	Amplitude 1, mv	Amplitude 0, mv	Tempo de pico, μs	Tempo de comutação, μs
Corrente coincidente	20	22-100	22/14/05	700/350	0,4	0,05	55	5	0,13	0,22
	30	31-103	31/20/07	580/290	0,5	0,1	61	8	0,23	0,37
		31-108	31/20/07	720/360	0,5	0,1	62	5	0,21	0,39
	50	52-106	50/30/15	500/250	2,0	0,2	90	9	0,43	0,75
		51-110	50/30/15	500/250	2,0	0,2	68	8	0,52	1,02
		51-119	50/30/15	400/200	2,0	0,5	90	5	0,75	1,25
	80	81-104	80/50/25	410/203	6,0	0,5	60	6	1,4	2,6
Lítio de corrente coincidente para aplicações de faixa ampla de temperaturas	20	24-100	22/14/05	780/390	0,4	0,05	40	4	0,145	0,270
	30	34-102	31/20/10	760/380	0,5	0,1	63	4	0,26	0,49
Corrente coincidente para aplicações de $2^1/_2$D	20	22-101	22/14/04	850/425	0,4	0,05	60	5	0,12	0,20
	30	31-115	30/21/07	750/375	0,5	0,1	52	4	0,21	0,37

*Cortesia da Electronic Memories, Inc.

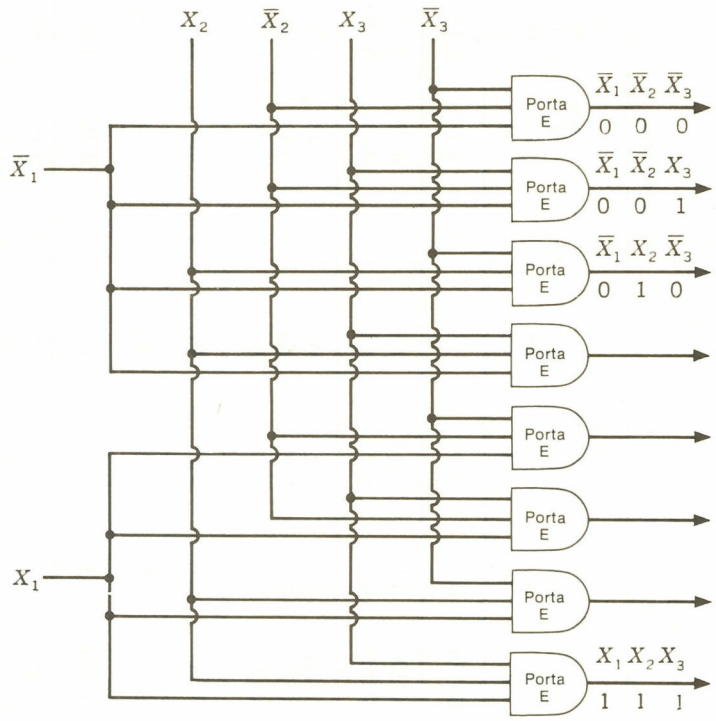


Fig. 7.13 Decodificador paralelo.

projetado para decodificar três células binárias ou biestáveis. Neste caso, há seis linhas de entrada para a rede ou estrutura (X_1, $\bar{X}_1$, X_2, $\bar{X}_2$, X_3 e $\bar{X}_3$). Há então $2^3 = 8$ linhas de saída, e, para cada um dos oito estados que os três biestáveis podem assumir, será selecionada uma única linha de saída. Este tipo de decodificador é muitas vezes construído usando diodos nas portas E. A regra é a seguinte: O número de diodos usado em cada porta E é igual ao número de entradas para cada porta E. Na Fig. 7.13, isto é igual ao número de biestáveis que estão sendo decodificados. Além disso, o número de portas E é igual ao número de linhas de saída, que é igual a 2^n (n é o número de biestáveis que estão sendo decodificados). O número total de diodos é, portanto, igual a $n \times 2^n$, e, para a matriz de decodificação binária na Fig. 7.13, são necessários 24 diodos para construir a rede. Como pode ser visto, o número de diodos requerido aumenta nitidamente com o número de entradas da rede. Por exemplo, para decodificar um registro de oito biestáveis, necessitaríamos de $8 \times 2^8 = 2048$ diodos se o decodificador fosse construído desta maneira.

Em conseqüência, há diversos outros tipos de estruturas freqüentemente usados na construção de redes decodificadoras. Uma tal estrutura, chamada *rede de decodificação tipo árvore*, é mostrada na Fig. 7.14. A rede tipo árvore particular mostrada na Fig. 7.14 decodifica quatro biestáveis e, portanto, tem $2^4 = 16$ linhas de saída, uma única das quais é selecionada para cada estado dos biestáveis. Um exame mostrará que são necessários 56 diodos para construir esta rede particular, enquanto que $2^4 \times 4 = 64$ diodos seriam requeridos para construir o tipo de decodificador paralelo mostrado na Fig. 7.13.

Na Fig. 7.15 é mostrado ainda um outro tipo de rede decodificadora, e esta rede é chamada *rede decodificadora multiplicativa balanceada*. Observemos que esta rede particular requer apenas 48 diodos. Pode ser provado que o tipo de rede decodificadora mostrada na Fig. 7.15 requer o número mínimo de diodos para uma rede decodificadora completa, e a diferença no número de diodos, ou elementos de decodificação, para construir uma rede tal como indicado na Fig. 7.15, comparada com a das Figs. 7.14 e 7.13, se torna mais significativa quando o número de biestáveis a serem decodificados aumenta. O tipo de rede mos-

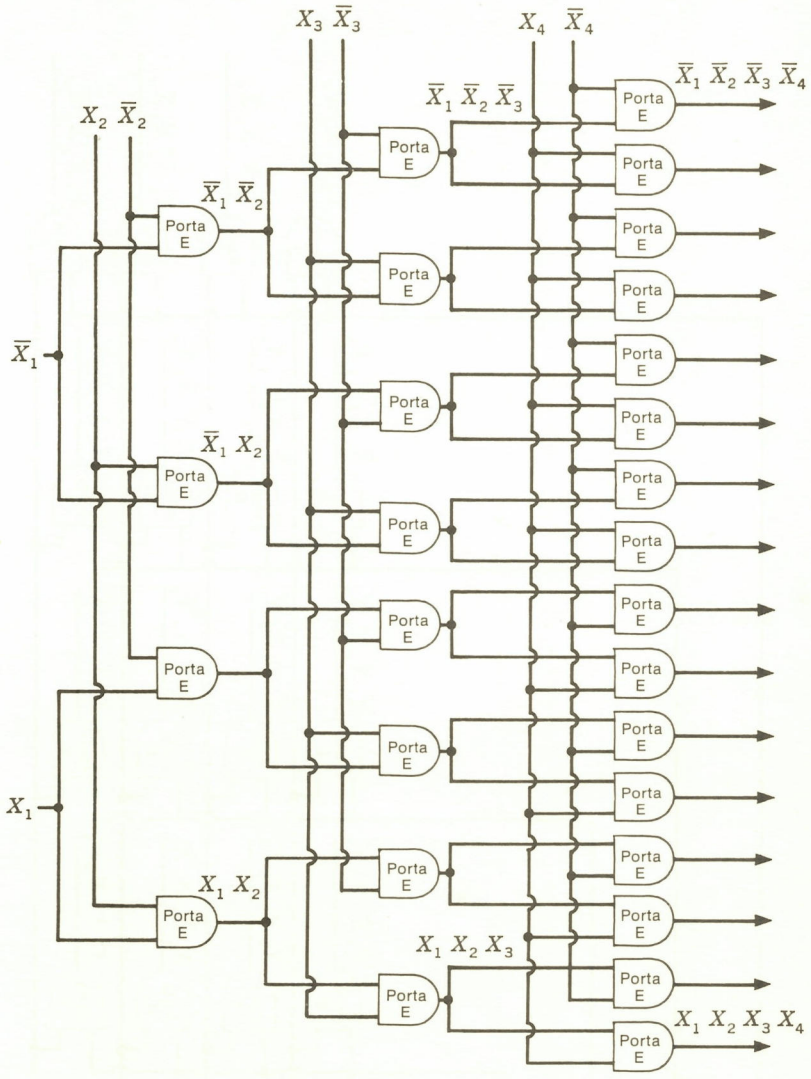


Fig. 7.14 Decodificador em árvore.

trado na Fig. 7.11, no entanto, tem a vantagem de ser o mais rápido dos três tipos de redes. Isto é contrabalançado pelo fato de que ele é o mais dispendioso.

Havendo estudado os três tipos de matrizes decodificadoras que são atualmente usadas em máquinas digitais, simplesmente desenharemos as redes decodificadoras como uma caixa com n entradas e 2^n saídas, com a compreensão de que um dos três tipos de circuitos mostrados nas Figs. 7.13 a 7.15 serão usados na caixa.

7.12 ORGANIZAÇÃO DA MEMÓRIA DE NÚCLEOS E ESQUEMAS DE FIAÇÃO

A organização da memória de núcleos descrita anteriormente é a clássica *memória de núcleos de quatro fios de acesso aleatório e de corrente coincidente.* Diversas variações deste esquema básico serão delineadas nesta seção.

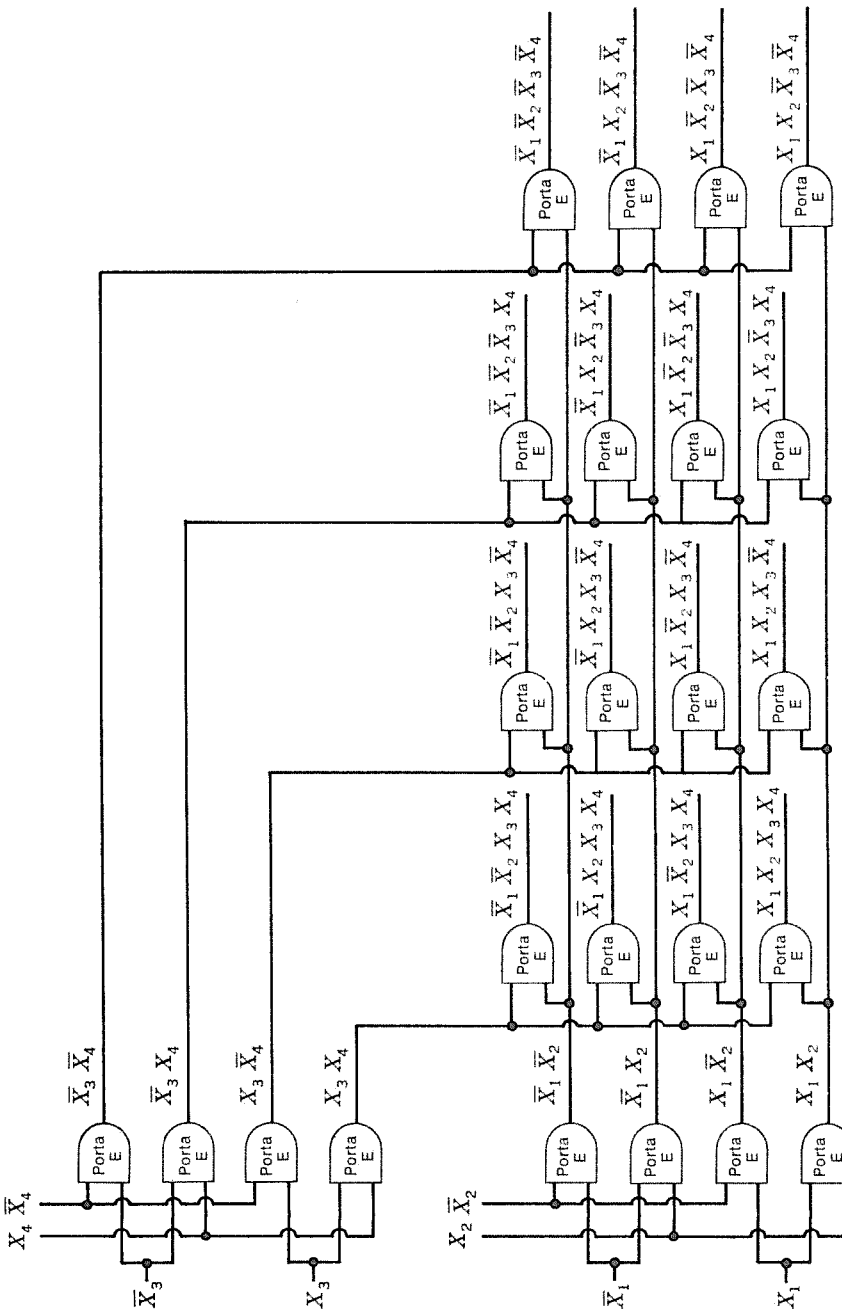


Fig. 7.15 Um decodificador balanceado.

Primeiro, observemos que os enrolamentos da linha de leitura e da linha do inibidor podem ser combinados num único enrolamento. O amplificador de leitura é usado somente durante a seção de leitura do ciclo de sincronização da memória, enquanto que o excitador inibidor está ligado apenas durante a porção de gravação do ciclo de sincronização da memória. Conectando-se simplesmente o amplificador de leitura ao enrolamento inibidor e removendo o enrolamento de leitura, uma matriz pode ser formada tendo apenas três fios por núcleo. Para visualizar este fato, imagine a Fig. 7.7 com o enrolamento de leitura removido. O amplificador de leitura é depois conectado ao enrolamento inibidor, e este enrolamento é chamado *enrolamento de dígito* ou *enrolamento de bit.* O amplificador de leitura deve, certamente, ser capaz de suportar a grande entrada proveniente do excitador inibidor. Uma memória deste tipo de três fios e de corrente coincidente é mostrada na Fig. 7.16, que é uma memória de 16 palavras de 4 bits.

Além da simplificação da construção da memória, a redução do número de enrolamentos através dos núcleos tem o efeito desejável de tornar possível reduzir as dimensões dos núcleos. Núcleos menores podem ser comutados mais rápido que os núcleos maiores, e o fator de limitação no tamanho do núcleo é muitas vezes o número de fios através dos núcleos; portanto, as memórias mais rápidas são fabricadas com menos fios. O preço que é pago está na complexidade adicional do circuito.

Há um outro esquema de seleção freqüentemente usado nas memórias de núcleo. Trata-se da chamada *seleção linear.* A seleção básica de um núcleo de cada plano de memórias coincidentes é mostrada na Fig. 7.17*(a)*, que é uma memória de três fios de corrente de coincidência. A Fig. 7.15*(b)* mostra a técnica básica de seleção linear, em que cada palavra tem uma linha de *palavra,* ou *de excitação,* separada, e há, portanto, tantas *linhas de palavras* quantas forem as palavras na memória. As linhas de leitura e de inibidor são combinadas na Fig. 7.17*(b)*, no que é geralmente chamada *linha de dígitos,* e há tantas linhas de dígitos quanto bits (dígitos binários) na palavra. Observemos que há somente dois fios por núcleo na Fig. 7.17*(b)*, e, portanto, os núcleos podem ser bem pequenos. As correntes de LER e GRAVAR são ambas correntes totais; assim, a linha de palavra selecionada envia primeiro uma corrente $-I_m$ de LER, depois uma corrente $+I_m$ de GRAVAR através da única linha selecionada. Os amplificadores de leitura determinam então se o bit previamente armazenado ao longo da linha de dígitos era um 0 ou um 1 durante o tempo de leitura. Durante o tempo de gravação, o excitador inibidor de cada linha de dígitos é ativado (para gravar um 0), e a corrente deste excitador é $-I_m/2$, ou desativado (para gravar um 1). Uma vez que a corrente é uma meia-corrente, ela não afeta outros núcleos ao longo da linha de dígitos, mas, quando aplicada, ela anula metade da corrente $+I_m$ normal através da linha da palavra, de modo que o núcleo na interseção da linha de dígitos com a linha de palavra selecionada recebe apenas a corrente $+I_m/2$ e permanece no estado 0.

Uma vez que há tantas linhas de palavra quanto palavras numa memória de seleção linear, a chave decodificadora deve ser muito maior. Por outro lado, as correntes de LEITURA e GRAVAÇÃO podem ser feitas grandes e ativadas e desativadas "fortemente", uma vez que somente os núcleos selecionados recebem alguma corrente. As memórias de seleção linear são geralmente rápidas, mas elas custam mais para se construir por causa do decodificador e por várias "pequenas" razões. (Por exemplo, os menores núcleos proporcionam menores saídas, de modo que os amplificadores de leitura devem ser cuidadosamente montados.)

Uma memória de seleção linear de 16 palavras de 4 bits é mostrada na Fig. 7.18.

Há um outro arranjo de memória de núcleos chamado memória de $2^1/_2$D. Uma variação da memória $2^1/_2$D é usada em grandes memórias de custos baixos, e uma outra em pequenas memórias rápidas. Não estudaremos este tipo de memória com detalhes, exceto para observar que o princípio básico consiste em fazer uma pilha, que é basicamente uma pilha de seleção de corrente de coincidência, mas onde os enrolamentos *X* são contínuos de plano a plano; no entanto, cada plano tem um enrolamento *Y* separado. Durante o ciclo de LEITURA, a corrente é então comandada através do enrolamento *X* selecionado, o qual atravessa todos os planos, e também através de todos os enrolamentos *Y;* portanto, cada excitador *Y* selecionado é ativado. Durante o ciclo de GRAVAÇÃO, somente são ativados aqueles excitadores *Y* dos planos em que deve ser gravado um 1, enquanto que os excitadores *X* são usados da maneira de costume. O CDC tem uma memória de núcleos com núcleos de 12 milésimos de pol. que tem um ciclo de memória de 275 ns e uma organização de $2^1/_2$D.

A memória de $2^1/_2$D pode ser de três fios com um enrolamento de leitura separado, ou há algumas memórias de dois fios onde a saída do núcleo selecionado é lida na linha de excitação *Y* por uma técnica de circuito. (Isto é usado numa memória IBM muito grande.) A memória de $2^1/_2$D é uma estratégia que usa a técnica de seleção de corrente de coincidên-

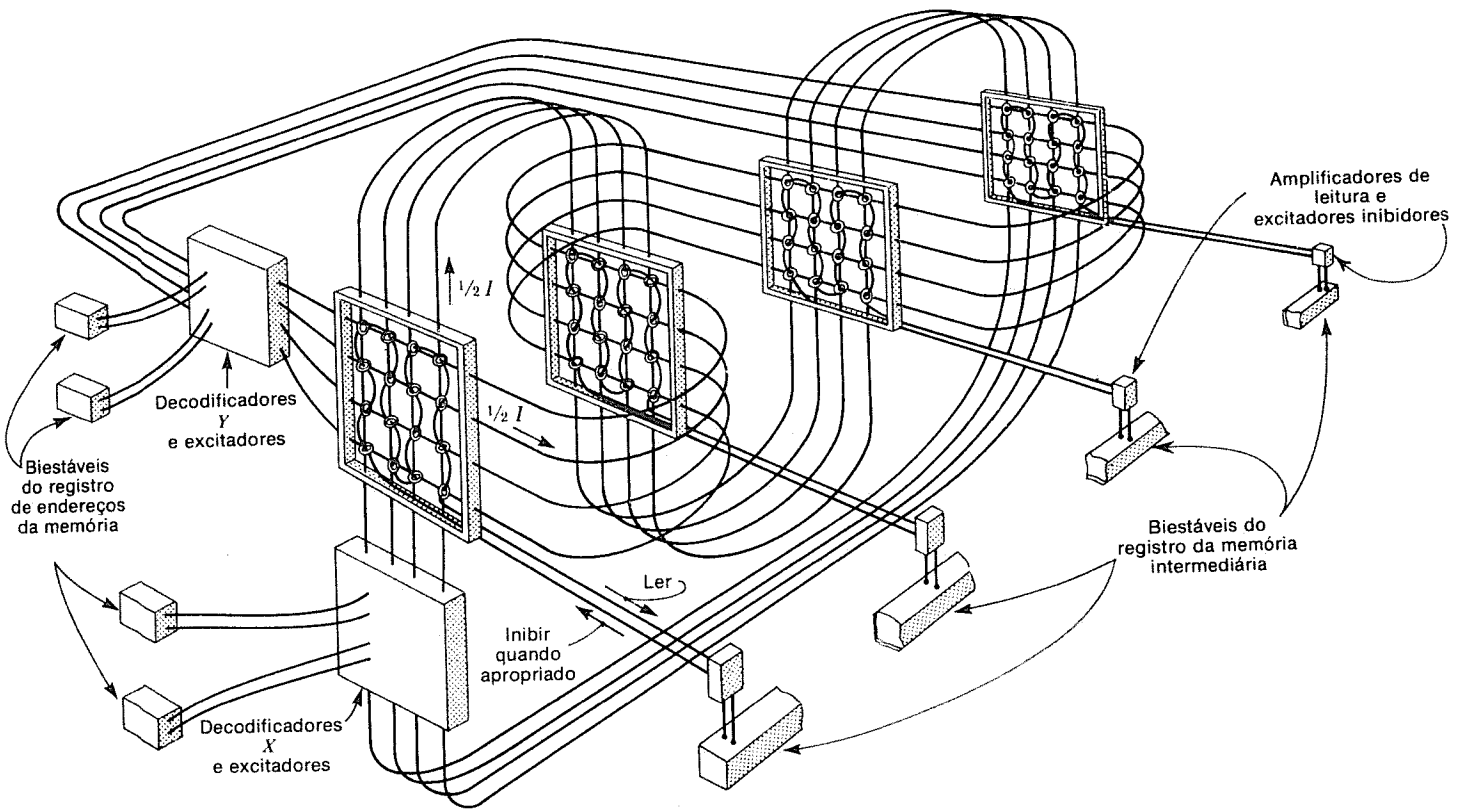
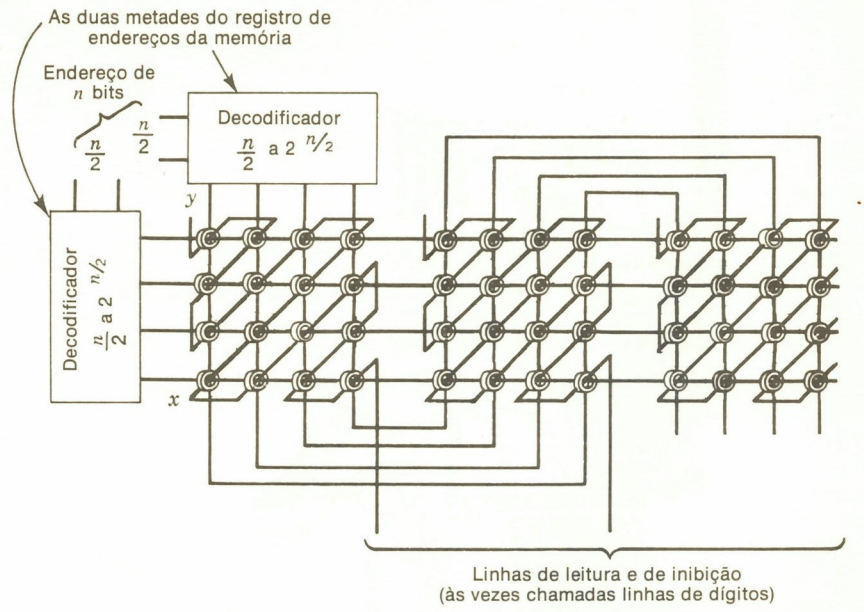
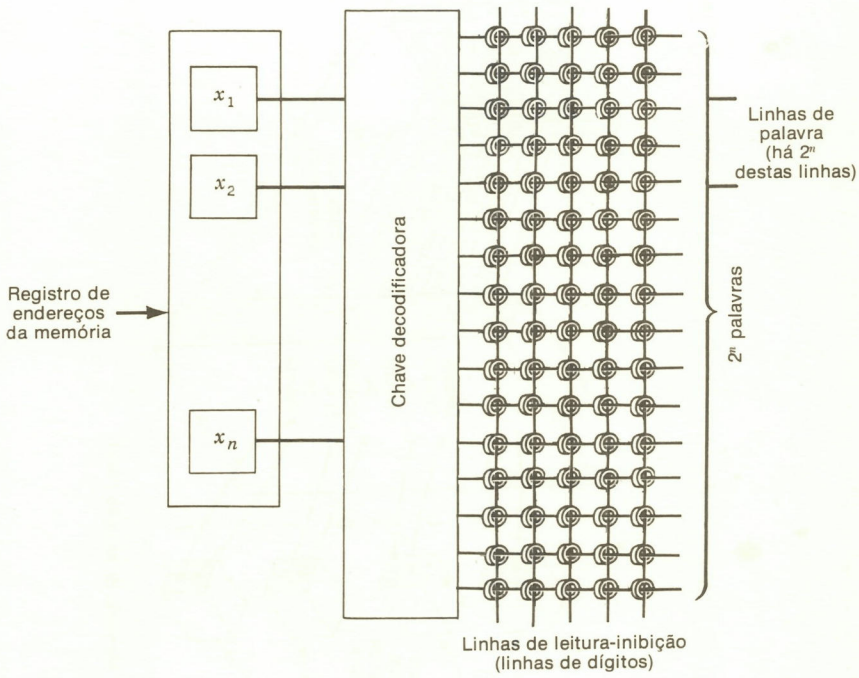


Fig. 7.16 Memória de três fios de corrente coincidente.



(a)



(b)

Fig. 7.17 Técnicas de sistemas de seleção de memória. *(a)* Sistema de seleção de corrente coincidente ou tridimensional; *(b)* memória de seleção linear, ou organizada em palavras, que é bidimensional.

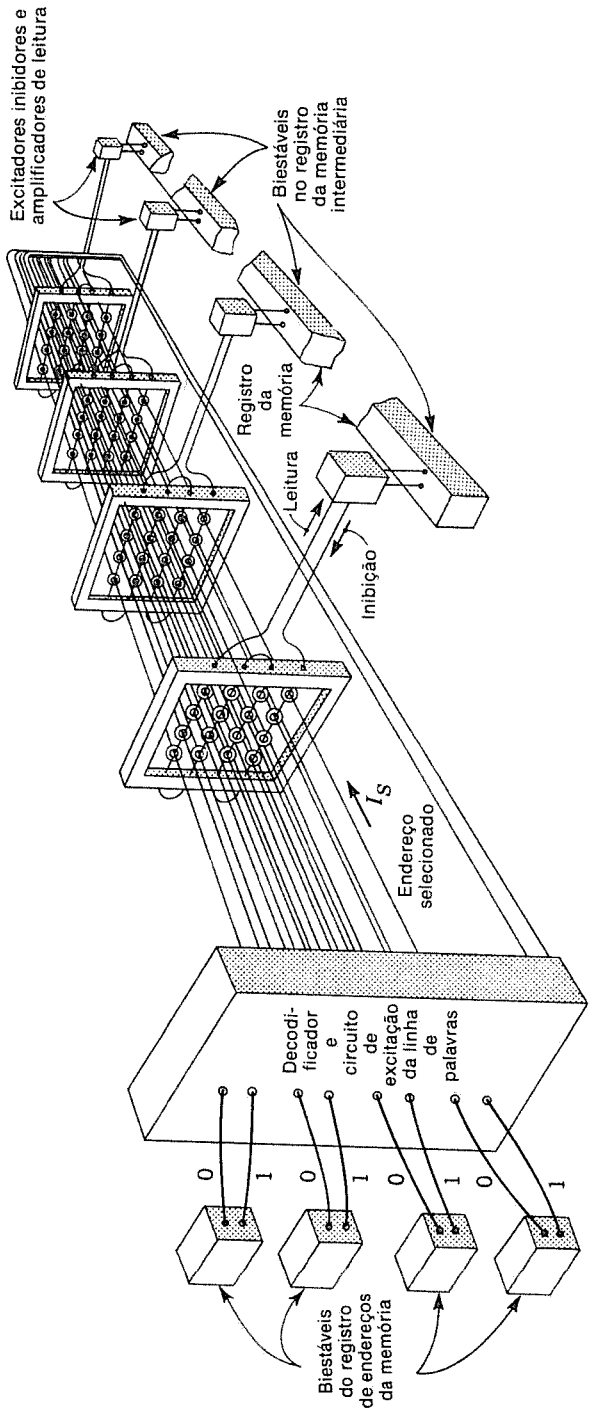


Fig. 7.18 Memória de seleção linear.

cia das meias-correntes, mas usa um enrolamento *Y* dividendo para facilitar a gravação, empregando circuitos lógicos mais complicados nos enrolamentos *Y* em vez de enrolamentos inibidores.

7.13 MEMÓRIAS DE ACESSO ALEATÓRIO A SEMICONDUTOR

A capacidade de fabricar grandes matrizes de componentes eletrônicos, empregando técnicas diretas de processamento, e de fabricar estas matrizes em pequenos receptáculos a preços razoáveis tornou possível a fabricação de memórias de circuitos eletrônicos, ou de *semicondutores*, que são competitivas com as memórias de núcleos. As primeiras destas memórias foram as pequenas memórias de biestáveis chamadas "memórias borrão", que eram de altas velocidades (mais rápidas do que o núcleo) e foram usadas em pequenas memórias operacionais ou auxiliares.

As pequenas memórias borrão com biestáveis têm comprimentos que variam de 16 a diversas centenas de palavras em comprimento e podem ser lidas ou gravadas num tempo da ordem de nanossegundos a dezenas de nanossegundos. Um único biestável é usado para cada bit de armazenamento, e são usadas portas para selecionar palavras e para ler nos biestáveis e ler dos biestáveis.

Quando os custos de fabricação se tornaram menores e as matrizes maiores se tornaram exeqüíveis, as memórias de semicondutor se tornaram muito maiores. Embora haja diversos esquemas e dispositivos disponíveis diferentes, há três principais categorias de memórias de CI atualmente.

1. *Memórias bipolares* Estas são essencialmente memórias de biestáveis, sendo estes biestáveis fabricados com emprego de transistores de junção *p-n* padronizados. Estas memórias são rápidas, mas tendem a ser dispendiosas.
2. *Memórias estáticas de MOS* Estas são fabricadas de dispositivos de MOS de efeito de campo para fazer os circuitos de biestáveis. Estas memórias são inferiores às memórias bipolares quanto à velocidade, porém, custam menos, consomem potência menor e possuem altas densidades de integração.
3. *Memórias dinâmicas de MOS* Estas são fabricadas empregando dispositivos de MOS, mas, em vez de usar um biestável como célula básica de memória, uma carga é depositada num capacitor (ou capacitores) fabricado na pastilha de CI, e a presença ou ausência desta carga determina o estado da célula. Os dispositivos de MOS são usados para ler e depositar a carga nos capacitores empregados. Uma vez que a carga usada se dissipará lentamente com o tempo, é necessário reavivar periodicamente esta carga, e as memórias são, portanto, chamadas *memórias dinâmicas* (as memórias com biestáveis bipolares ou de MOS são chamadas *memórias estáticas*). Estas memórias tendem a ser mais lentas do que as de outros tipos, mas são também menos dispendiosas, consomem potência menor e possuem uma alta densidade de integração.

Todos os três tipos de memórias acima têm uma vantagem sobre as memórias de núcleos — a leitura de dados é não-destrutiva. O Quadro 7.2 apresenta algumas características-chave da memória.

As técnicas de seleção, técnicas de leitura e outras idéias básicas usadas nas memórias de núcleos também se aplicam às memórias de semicondutores. Uma vez que os dispositivos de armazenamento são fabricados numa pastilha, os amplificadores de leitura, os decodificadores etc. podem ser colocados na mesma pastilha, e muito freqüentemente pastilhas individuais contêm em essência memórias pequenas e completas. As seções seguintes estudam algumas das características e detalhes importantes das memórias de semicondutor.

7.14 MEMÓRIAS BIPOLARES DE CI

As memórias bipolares são fabricadas usando-se versões de alta densidade do biestável com transistor bipolar. Um problema que surge no aumento da densidade é o do isolamento dos transistores individuais, um do outro, mas os fabricantes desenvolveram processos para reduzir o espaço necessário à isolação do transistor. Uma vantagem dos dispositivos bipolares é a simplicidade do conjunto de circuitos da interface requerida por causa da compatibilidade com a lógica bipolar. Uma outra vantagem é a velocidade atingível.

As memórias bipolares são fabricadas usando processos similares aos descritos no Cap. 5, e os tipos principais de circuitos são os de lógica TTL e ECL. (A dopagem do ouro para reduzir o tempo de recuperação do transistor bem como o grampeamento de diodo Schottky para evitar a saturação do transistor são usados nas configurações TTL para aumentar a velocidade. As memórias de ECL são adequadas apenas onde o custo é secundário para velocidade muito alta.)

Uma célula básica de memória na tecnologia bipolar é mostrada na Fig. 7.19*(a)*. Este

QUADRO 7.2 CARACTERÍSTICAS DE MEMÓRIAS COM SEMICONDUTORES

Característica	Canal *p* de MOS	Canal *n* de MOS Alta velocidade	Canal *n* de MOS Alta densidade	Bipolar
Número de bits por pastilha	4096	4096	16.384	2048
Tempo de acesso, ns	300-400	50	150	20
Dissipação de potência, mW/bit	0,2	0,1	0,02	0,25
Custo médio de grandes quantidades, *cents* por bit	0,15	0,5	0,1	0,6

biestável é fabricado com dois transistores de emissores múltiplos. Os transistores Q_3 e Q_4 formam o biestável, e, ou Q_3 estará conduzindo e Q_4 cortado, ou Q_3 estará cortado e Q_4 conduzindo, e assim o circuito é biestável.

Estes biestáveis são montados numa matriz para formar uma memória. A Fig. 7.19*(a)* também mostra uma matriz bidimensional destes biestáveis e a maneira pela qual eles são colocados num receptáculo de CI. Cada quadrado no diagrama-bloco da embalagem de CI, mostrando as conexões e o traçado geral, representa um único biestável na matriz de 16 biestáveis.

Um biestável na matriz é selecionado por meio das linhas de seleção *X* e *Y*. As duas linhas de LEITURA S_0 e S_1 são compartilhadas por todos os biestáveis e possuem retorno para a terra através de uma baixa resistência, e as saídas estão conectadas aos amplificadores de leitura.

A operação da memória é como segue. As quatro linhas X_1, X_2, X_3 e X_4 de seleção *X* e as quatro linhas Y_1, Y_2, Y_3 e Y_4 de seleção *Y* estão normalmente baixas, ou um 0. Fazendo-se uma única linha *X* e uma única linha *Y* um 1, seleciona-se a célula na interseção das duas linhas. Para gravar um 1 numa célula selecionada, a célula é selecionada aumentando-se uma entrada *X* e uma *Y* ao mesmo tempo. W_0 é então feita um 0, e W_1, um 1. Para gravar um 0, a célula é selecionada, W_0 é feita um 1, e W_1 é feita um 0.

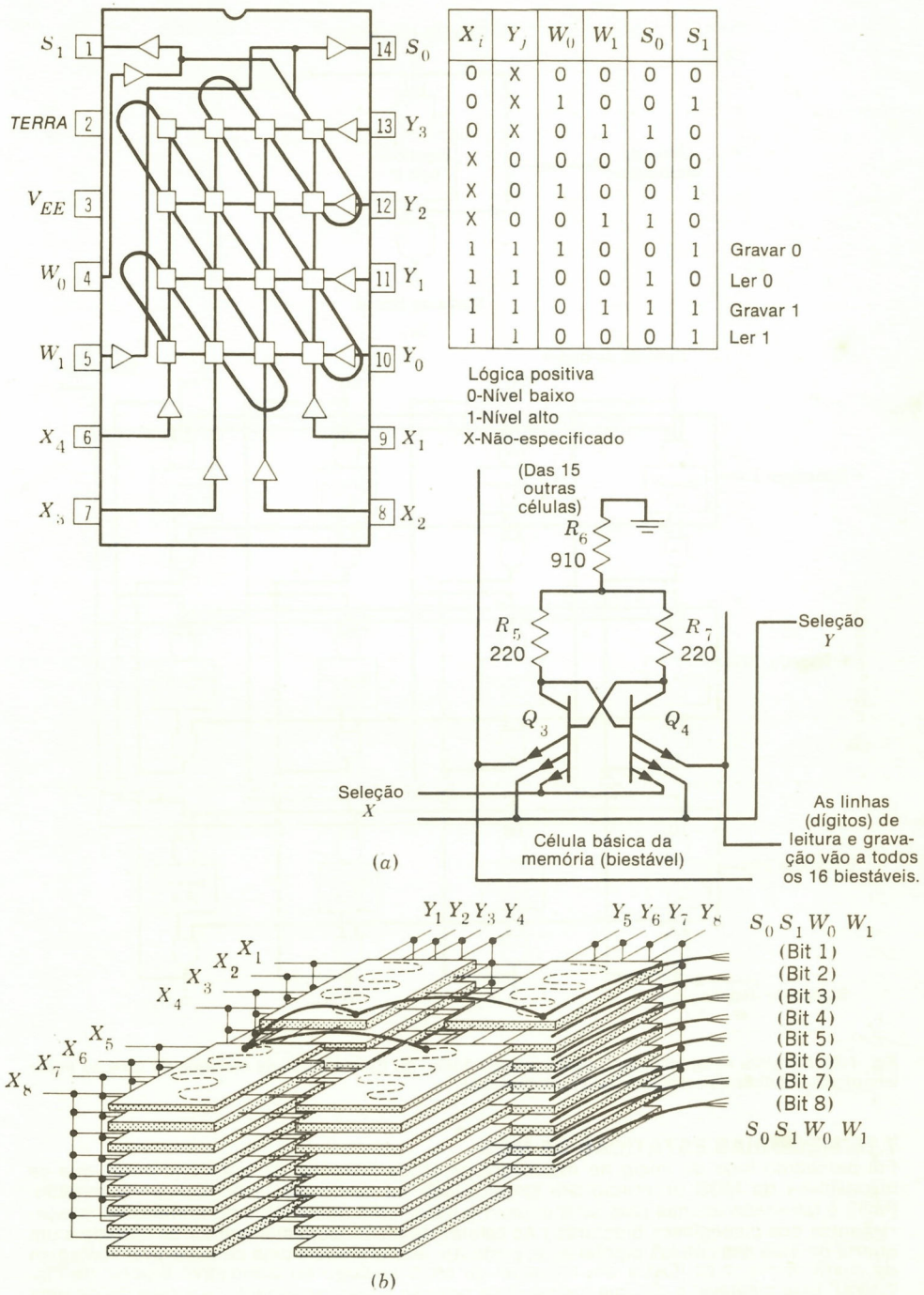
Para ler da memória, tanto W_0 quanto W_1 são mantidas em 0, e uma célula é selecionada aumentando uma linha *X* e uma *Y*. Então a linha S_1 ou linha S_0 terá uma saída de corrente; por exemplo, se a célula selecionada contiver um 0, então S_0 fornecerá corrente. (Portanto, S_0 e S_1 são linhas de LEITURA.) A leitura é não-destrutiva porque o valor lido não necessita ser gravado novamente na célula.

Como fabricados pela RCA, TI e Fairchild, cada receptáculo de CI constitui um receptáculo padrão de 14 pinos, conforme indicado, contendo 16 bits, incluindo circuitos de excitação e amplificadores de leitura. A Fig. 7.19*(b)* mostra 32 destas embalagens interligadas para formar uma memória de 64 palavras de 8 bits.

A memória precedente é muito rápida, com tempos de leitura e de gravação da ordem de poucos nanossegundos. Muito freqüentemente, mais bits são embalados num único receptáculo de CI e, muitas vezes, os decodificadores que excitam as linhas *X* e *Y* são incluídos na embalagem.

Um exemplo de uma memória de CI com seleção linear é mostrado na Fig. 7.20. Cada célula ou bit é como na Fig. 7.20*(a)*, que mostra que, para ler de uma célula, a linha de ENDEREÇO é feita um 1. Para gravar numa célula, a linha de ENDEREÇO e as linhas de GRAVAR e de ATIVAR são feitas um 1, e a linha de DADOS em seguida fornece o valor a ser gravado. A Fig. 7.20*(b)* mostra um diagrama-bloco de uma memória de 16 palavras de 4 bits que é de seleção linear, quanto a que, para ler ou gravar, uma única linha de ENDEREÇO do 16 é feita alta e depois, quando da leitura, os valores em S_1, S_2, S_3 e S_4 são lidos. Para gravar, uma única linha de ENDEREÇO é feita um 1, os valores a serem gravados são colocados em I_1, I_2, I_3 e I_4, e depois a linha GRAVAR ATIVAR é feita um 1.

A Fig. 7.21 mostra um diagrama-bloco de uma única embalagem contendo todos os decodificadores e memórias intermediárias de uma memória de 256 bits com acesso aleatório. As dimensões também estão indicadas. Na prática real, as memórias contêm da ordem de 4.000 bits numa única pastilha de semicondutor, incluindo todos os decodificadores, memórias intermediárias e excitadores.



X_i	Y_j	W_0	W_1	S_0	S_1	
0	X	0	0	0	0	
0	X	1	0	0	1	
0	X	0	1	1	0	
X	0	0	0	0	0	
X	0	1	0	0	1	
X	0	0	1	1	0	
1	1	1	0	0	1	Gravar 0
1	1	0	0	1	0	Ler 0
1	1	0	1	1	1	Gravar 1
1	1	0	0	0	1	Ler 1

Fig. 7.19 Memória de semicondutores de corrente coincidente e alta velocidade. *(a)* Modelo básico de memória (cortesia da RCA); *(b)* módulos montados numa memória de 64 palavras de 8 bits.

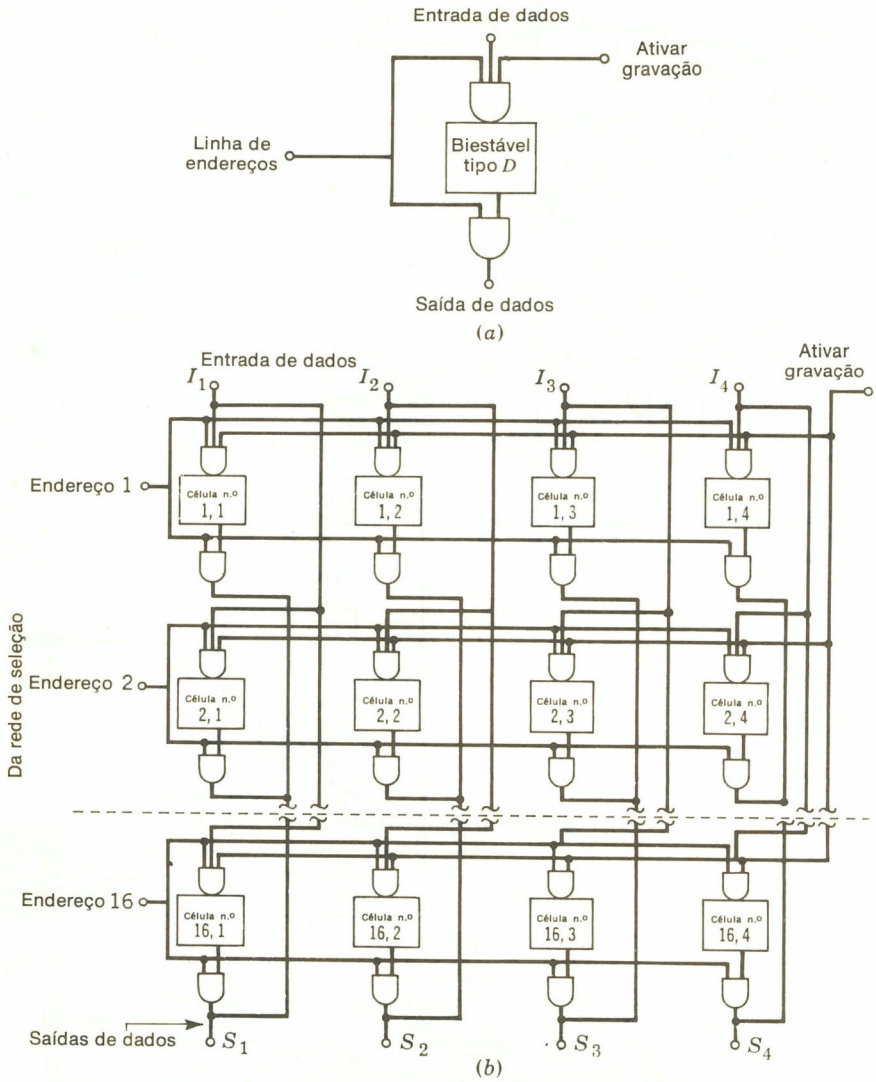


Fig. 7.20 Memória de semicondutores de seleção linear. *(a)* Célula típica da memória de seleção linear em *(b)*; *(b)* detalhes da seleção e da leitura-gravação.

7.15 MEMÓRIAS ESTÁTICAS DE MOS

Foi percebido logo de início no desenvolvimento da memória de semicondutores que os dispositivos de MOS proporcionam simplicidade na fabricação e economia de traçado. (MOS é fabricado apenas com uma difusão e talvez com dois terços do número de mascaramentos dos dispositivos bipolares.) As células de MOS ocupam somente de metade a um quarto da área das células bipolares, e, portanto, elas oferecem uma considerável vantagem de custo. A Fig. 7.22 ilustra um biestável de MOS análogo ao dispositivo bipolar da Fig. 7.19*(a)*. Este biestável é de uma memória de seleção linear, no entanto, e a seleção de uma célula é feita levantando-se a linha da palavra. Os transistores *Q*3 e *Q*4 atuam como resistores, substituindo os resistores de área relativamente grande usados nos circuitos de coletor

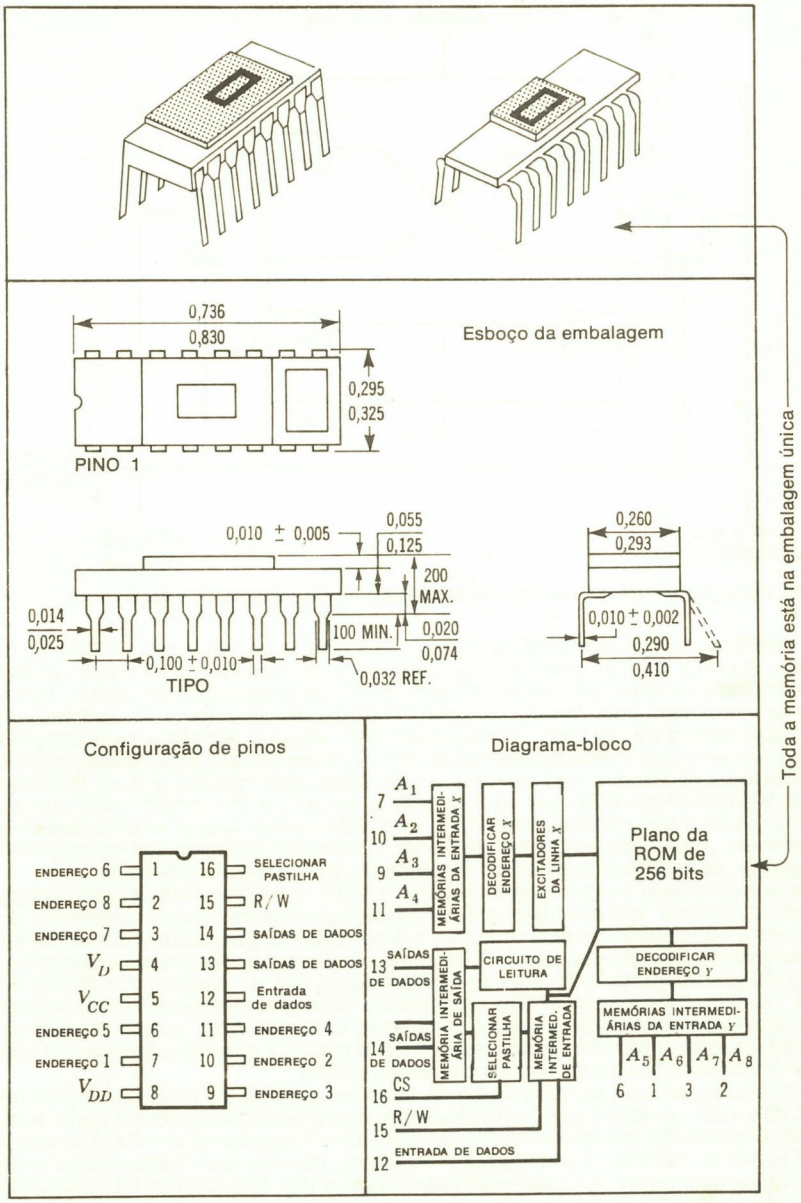


Fig. 7.21 Memória de 256 bits numa única pastilha. (Cortesia da INTEL Corp.)

das unidades bipolares. *Q*5 e *Q*6 servem como portas, conectando-se às linhas de dígitos. O estado da célula de armazenamento pode ser lido levantando-se a linha da palavra que ativará ou *Q*5 ou *Q*6, dependendo do estado do biestável. A operação GRAVAR é executada levantando-se a linha da palavra e depois ajustando-se o valor desejado pela colocação de uma tensão alta e baixa nas linhas de dígitos corretas.

O tipo predominante de dispositivo de MOS nas primeiras memórias foi a unidade de

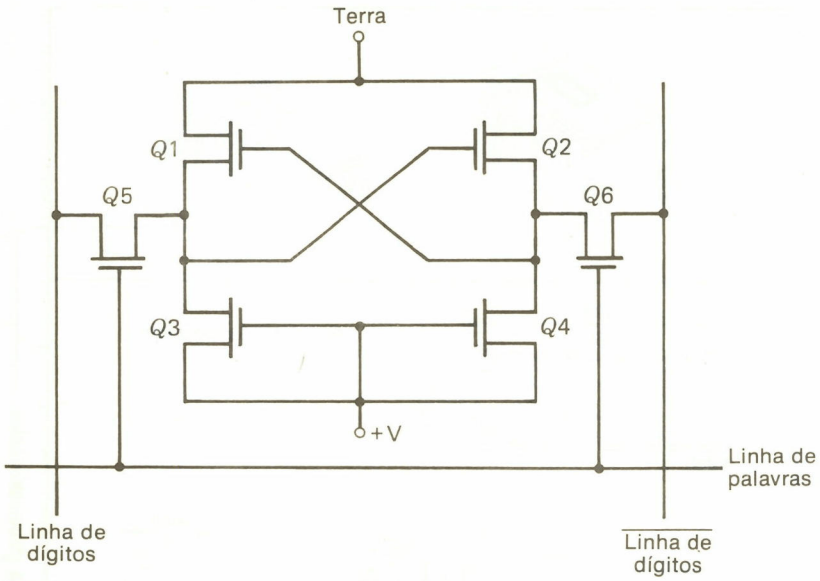


Fig. 7.22 Célula de armazenamento de bits MOS estático para memória de seleção linear.

modo reforçado de canal *p* (PMOS), em que os buracos constituem o veículo do fluxo da corrente. Embora fácil de produzir, de custo baixo e confiável, o PMOS é relativamente lento e limitado na densidade de integração de LSI. Os dispositivos de canal *n* tornaram-se agora os mais populares, proporcionando importantes vantagens de desempenho, tais como baixa tensão de operação e velocidade mais alta, em relação a seus congêneres de canal *p*. Devido a que a mobilidade dos elétrons é maior do que a mobilidade dos buracos, os transistores de canal *n* são de duas a três vezes mais rápidos que os transistores de canal *p*, o que equivale dizer que os dispositivos de canal *n* têm ganho maior que os dispositivos de canal *p* das mesmas dimensões. Com velocidades iguais, as unidades de canal *n* são menores, permitindo maior densidade de integração; e podem ser empregados níveis de dopagem de substrato mais altos, aumentando ainda mais a densidade.

Se compararmos as tecnologias bipolar e MOS, a bipolar oferece uma vantagem de velocidade, embora, até recentemente, as limitações impostas pela necessidade do isolamento entre transistores tenham reduzido a densidade de integração e, portanto, a capacidade de armazenamento por pastilha. Os componentes bipolares podem proporcionar tempos de acesso inferiores a 10 ns, em contraste com 300 ou mais ns dos PMOS e 30 dos NMOS. Os dispositivos de MOS têm capacitância interna e impedância relativamente elevadas, acarretando tempos de acesso e constantes de tempo mais longos. Nossos símbolos esquemáticos não mostrarão as ligações do canal nos FET como é convencional nos dispositivos de memória. Todos os dispositivos serão NMOS.

A operação de uma pastilha típica de memória estática de 4096 bits é detalhada nas Figs. 7.23 a 7.26. Os 4096 bits de memória estão organizados numa matriz de 64 filas por 64 colunas. O acesso às células de memória é feito decodificando-se simultaneamente o endereço X A_0 até A_5 nas filas, e o endereço Y A_6 até A_{11} nas colunas. (Cada coluna contém um amplificador de pré-leitura, cujas saídas são introduzidas num circuito OU e conectadas ao estágio de saída.) Cada bit ou célula de memória é um biestável padronizado conforme indica a Fig. 7.23, consistindo de R_1, R_2 (que são na realidade dispositivos MOS), $Q2$, $Q4$, $Q1$ e $Q3$. Os resistores de carga $R1$ e $R2$ são tipicamente de 60 MΩ e se ligam à fonte de alimentação. $Q1$ e $Q3$ são usados para conectar a célula às linhas de dígitos sempre que a linha de seleção X estiver alta. Para ler desta célula, a linha X é levantada; então, a célula puxará uma das linhas de dígitos para o baixo, a partir de seu estado normalmente alto. O circuito de pré-leitura selecionado pelo decodificador da coluna Y detectará a tensão diferencial sobre a linha de dígitos selecionada e a amplificará. Para gravar na célula, uma linha X é selecionada, e depois uma linha de dígitos é selecionada e forçada para baixo pelo circuito, e a célula selecionada assume o estado forçado pela linha de dígitos selecionada.

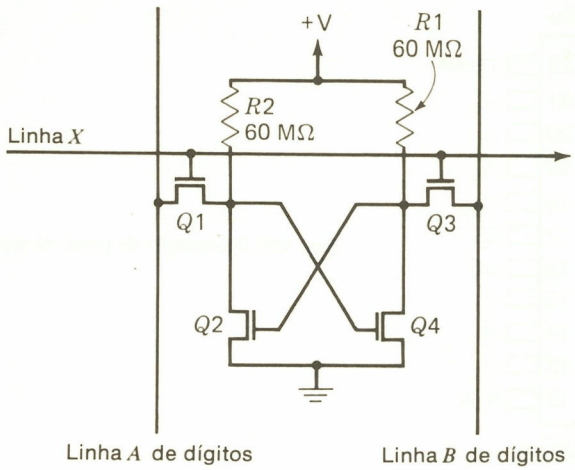


Fig. 7.23 Célula de memória MOS estática. (Figs. 7.23 a 7.26 cortesia da Electronic Memories and Magnetics Corp.)

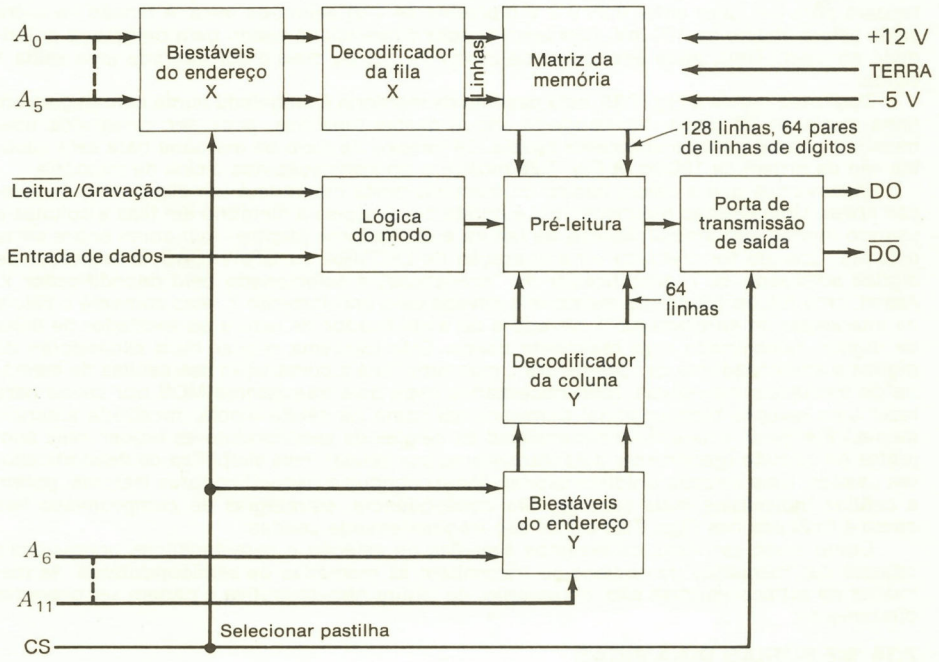


Fig. 7.24 Diagrama-bloco da memória MOS estática.

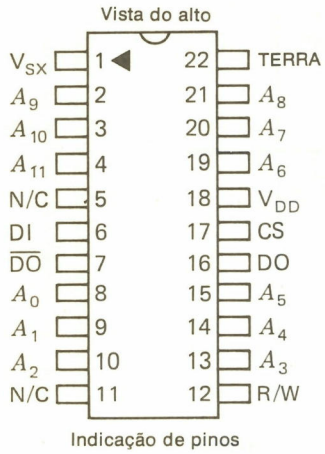


Fig. 7.25 Designação de pinos da memória MOS estática.

A entrada de SELEÇÃO DE PASTILHA (CS) controla a operação da memória. Quando a SELEÇÃO DE PASTILHA for baixa, as memórias intermediárias do endereço da entrada, os decodificadores, os circuitos de leitura e os estágios de saída são mantidos no estado desligado, e a energia é fornecida apenas aos elementos da memória. Quando a SELEÇÃO DE PASTILHA for alta, a memória está ativada. O pulso de SELEÇÃO DE PASTILHA sincroniza os endereços do nível lógico da TTL, LEITURA-GRAVAÇÃO, e entrada de DADOS nos biestáveis tipo *D* e ativa o estágio de saída. Quando uma célula for lida, de uma das duas saídas produzirá corrente (DO para os dados originalmente introduzidos como um 1, $\overline{DO}$ para os dados originalmente introduzidos como um 0). As saídas DO e $\overline{DO}$ têm um retorno para a terra através de uma impedância; em conseqüência, a tensão na saída que produz corrente tenderá para um valor entre 0,35 e 2 V (tipicamente 1 V) acima da terra. A tensão na outra saída estará abaixo de 100 mV. (Um amplificador diferencial é usado para detectar a polaridade do sinal.) Uma saída diferencial de 25 mV ($V_{saída}$) ou mais é considerada uma saída 1 válida.

Conforme indica a Fig. 7.25, esta pastilha de memória é embalada numa embalagem em linha dupla de 22 pinos. Montando-se várias destas pastilhas, pode ser construída uma memória grande e moderadamente rápida. Os tempos de ciclo de memória para esta pastilha são da ordem de 100 ns. A Fig. 7.26 mostra a sincronização dos ciclos de memória.

Observemos que a célula básica de memória nesta memória é um dispositivo de seleção linear. Uma seleção bidimensional é obtida dividindo-se a memória em filas e colunas e usando muitos (64) amplificadores de leitura e excitadores (dígitos) inibidores sobre estas colunas. Quando for efetuada uma operação de LEITURA ou GRAVAÇÃO, o excitador de dígitos adequado ou o amplificador de "pré-leitura" é selecionado pelo decodificador *Y*. Assim, uma fila de células de memória é ativada pelo enrolamento *X*, mas somente a célula na interseção do enrolamento *X* ativado e do amplificador de leitura ou excitador de linha de dígitos selecionado será realmente usada. Este esquema requer mais excitadores de dígitos e amplificadores de "pré-leitura", mas simplifica a construção das células de memória de biestáveis individuais. (São necessários mais dois transistores MOS por célula para fazer uma seleção bidimensional convencional como na célula bipolar mostrada anteriormente.) Em geral, a seleção bidimensional de células de semicondutores requer mais conjuntos de circuito (geralmente dois transistores por célula), mas simplifica os decodificadores usados. Uma seleção unidimensional (linear) conduz a decodificadores maiores, porém a células individuais mais simples. Em conseqüência, estratégias de compromisso tais como a indicada nas Figs. 7.23 e 7.24 são freqüentemente usadas.

Como pode ser visto, os mesmos sistemas de seleção e características operacionais básicas das memórias de núcleos se transmitem às memórias de semicondutores. As memórias de semicondutores são, no entanto, de leitura não-destrutiva e podem ser gravadas diretamente.

7.16 MEMÓRIAS DINÂMICAS

As células de MOS são freqüentemente usadas como base de um sistema de memória dinâmica. Numa dada célula de memória, um 1 é gravado colocando-se uma carga num ca-

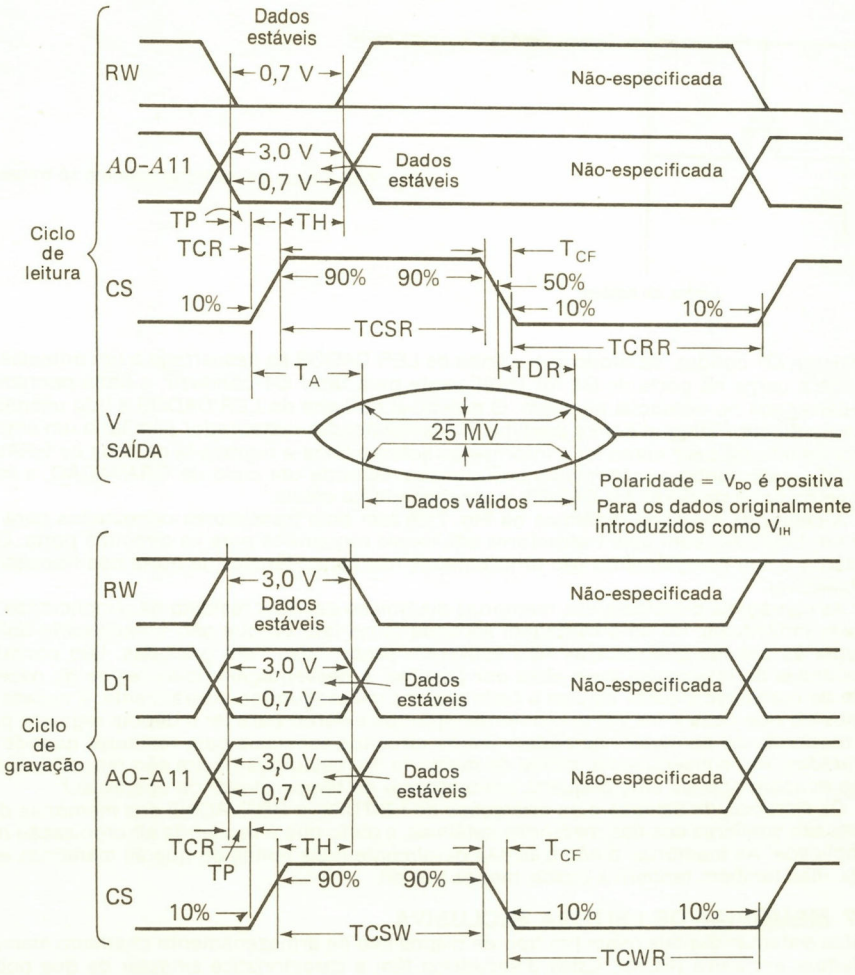


Fig. 7.26 Diagramas de sincronização.

pacitor; descarregando-se o capacitor, armazena-se um 0. A leitura então impõe a percepção da presença ou ausência de uma carga no capacitor. Contudo, devido a que sempre existe uma certa fuga da carga para a terra, é necessário uma recarga periódica.

A Fig. 7.27 descreve o tipo mais simples de célula de armazenamento de dados, usando um único transistor de comutação MOS e um capacitor de armazenamento (também um dispositivo MOS). Durante o ciclo de GRAVAR, uma carga pode ou não ser armazenada no capacitor. Durante o ciclo de LER, a carga do capacitor determina a tensão na linha de DADOS. Embora a célula de memória de transistor único requeira circuitos de leitura e gravação sofisticados, suas pequenas dimensões se ajustam ao armazenamento de alta densidade tal como nas memórias com 16.000 células por pastilha de CI.

Uma célula de três transistores usada em várias memórias dinâmicas de MOS é mostrada na Fig. 7.28. A capacitância de porta de *Q*1 é usada para armazenar a carga, eliminando a necessidade de um capacitor adicional, e, assim, esta célula pode ser feita extremamente pequena, permitindo alta densidade de integração de bits. Para ler desta célula, a linha de GRAVAR DADOS é elevada ao nível 1. A linha LER SELECIONAR é então elevada de modo que *Q*2 possa conduzir. Se a tensão na porta de *Q*1 exceder o valor limite do

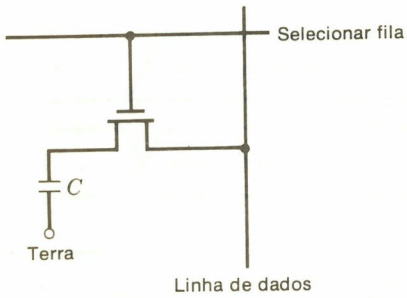


Fig. 7.27 Célula de memória com um só transistor.

transistor, $Q1$ conduz, de modo que a linha de LER DADOS se descarrega a um potencial de 0 V. Se a carga na porta de $Q1$ for insuficiente para fazer $Q1$ conduzir, a linha permanece pré-carregada no potencial negativo. O potencial da linha de LER DADOS é lido usando-se um amplificador integral com a pastilha. Um amplificador restaurador (similar a um amplificador de leitura) deve então ler a informação sobre a linha e regravá-la na linha de GRAVAR DADOS, restaurando a informação armazenada. Durante um ciclo de GRAVAÇÃO, a informação na linha de GRAVAR DADOS é armazenada na célula.

A célula da memória dinâmica na Fig. 7.29 tem dois transistores conectados para formar um biestável, com dois transistores adicionais requeridos para os circuitos porta. O bit de dados e seu complemento são armazenados nas capacitâncias de porta dos transistores do biestável.

As vantagens principais das memórias dinâmicas de MOS residem na simplicidade das células individuais. Há uma vantagem secundária no fato de que não é necessário aplicar energia às células quando elas não estiverem sendo lidas nem gravadas. Isto conduz a densidades de integração mais altas por pastilha. A desvantagem óbvia reside na necessidade de reabastecer estas células a cada poucos milissegundos. (Geralmente, é necessário menos do que cada 2 ms.) O conjunto de circuitos externo para ler e depois regravar pode ser requerido ou as memórias podem incluir circuitos especiais para restaurar quando comandados. Em conseqüência, ciclos de memória de restauração extra são requeridos, mas estes ocupam apenas uma pequena porcentagem do tempo global de operação.†

Os sistemas de seleção e as operações de LEITURA e GRAVAÇÃO das memórias dinâmicas são similares aos das memórias estáticas, exceto que é requerida sincronização mais complicada. As memórias dinâmicas são geralmente mais lentas do que as memórias estáticas, mas também tendem a custar menos por bit.

7.17 MEMÓRIAS DE LEITURA EXCLUSIVA

Muitos sistemas digitais usam um tipo de dispositivo de armazenamento chamado *memória de leitura exclusiva* (ROM). Estes dispositivos têm a característica singular de que podem ser lidos de, mas que o computador não pode gravar informação neles. A informação armazenada nestas memórias é, portanto, introduzida na memória de algum modo tal que a informação seja semipermanente ou permanente. Às vezes, a informação armazenada numa ROM é colocada na memória no tempo de construção, e, às vezes, os dispositivos são usados onde a informação pode ser alterada manualmente. Nesta seção estudaremos diversos tipos de memórias de leitura exclusiva (ROM), tendo em mente que vários tipos diferentes estão atualmente sendo construídos. Os tipos de memória que estudaremos, no entanto, são característicos desta classe particular de dispositivos de memória, e a maioria dos dispositivos representa variações dos princípios que serão apresentados.

Basicamente, uma ROM é um dispositivo com várias linhas de entrada e de saída tais que, para cada combinação de entrada, há uma única combinação de saída. Assim, uma ROM concretiza fisicamente uma tabela da verdade, ou tabela de combinações. Uma tabela (pequena) típica pode ser o Quadro 7.3.

Esta lista de valores de entrada-saída é realmente uma lista de valores do código binário/Gray (o código Gray será discutido no capítulo seguinte). É importante verificar que

†Há mesmo alguns mecanismos interessantes de restauração interna chamados *bombas de carga* que requerem aplicação de uma onda senoidal numa das entradas e tornam a restauração transparente (invisível) ao usuário. Outras estratégias de restauração regravam colunas inteiras usando um único pulso de restauração.

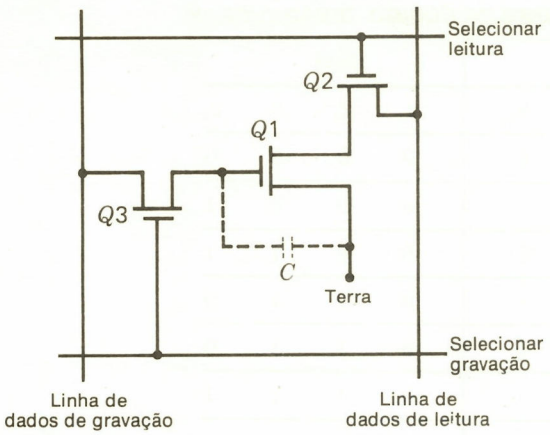


Fig. 7.28 Célula de memória com três transistores para memória dinâmica.

a lista pode ser utilizada de duas maneiras, no entanto: primeiro, como uma tabela de uma rede de circuitos lógicos com quatro entradas e quatro saídas e, segundo, como uma lista de endereços de 0 a 15, conforme dada pelos valores X, e os conteúdos de cada endereço, como dados pelos valores dos Z.

Portanto, podemos construir uma rede ou estrutura de circuitos lógicos como na Fig. 7.30 que deve fornecer a saída Z correta para cada entrada X. (As caixas com $\oplus$ no interior são somadores em base 2.)

A tabela também pode ser construída por uma memória de núcleos de 16 palavras com quatro bits por palavra na qual tenhamos lido 0000 no endereço 0; 0001 no endereço 1; 0011 no endereço seguinte; e assim por diante até 1.000 no último endereço. Se não gravássemos sempre depois disso nesta memória, ela seria uma memória ROM e serviria com mesmo propósito que a rede lógica da Fig. 7.30.

A Fig. 7.31*(a)* mostra um esquema para implementação da tabela usando uma estrutura decodificadora com quatro entradas X_1, X_2, X_3 e X_4 e vários diodos. Com uma certa combi-

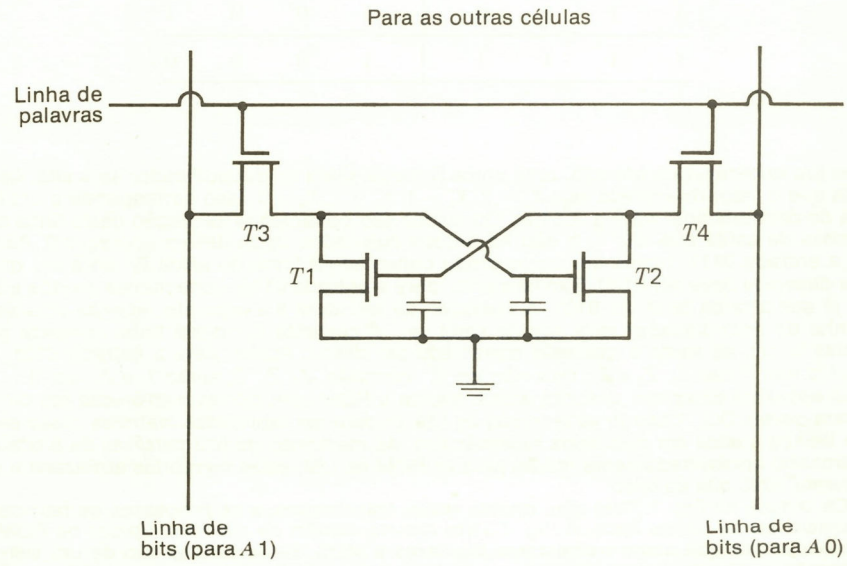


Fig. 7.29 Célula de memória MOS dinâmica com quatro transistores MOS.

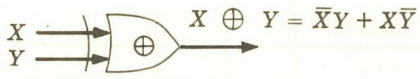
QUADRO 7.3 VALORES DE CÓDIGO BINÁRIO/GRAY

Entrada				Saída			
X_1	X_2	X_3	X_4	Z_1	Z_2	Z_3	Z_4
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

nação (ou endereço) da entrada, uma única linha de saída do decodificador será alta. Admitamos que o valor da entrada seja $X_1 = 0$, $X_2 = 1$, $X_3 = 1$, $X_4 = 1$. Isso corresponde a 0111 na saída do decodificador na Fig. 7.31*(a)*. Os diodos são conectados na junção desta linha com as linhas de saída que são 1, e não há diodos colocados onde devam aparecer 0. Assim, para a entrada 0111, temos um único diodo conectado à linha de saída Z_2, uma vez que a saída desejada deve ser 0100. Similarmente, para a entrada 0110, conectamos diodos a Z_2 e a Z_4, já que a saída deve ser 0101. A operação da estrutura é a seguinte: Apenas uma saída da linha do decodificador será alta de cada vez. A corrente flui desta linha somente para aquelas linhas de saída a que está conectado um diodo. Assim, para a entrada 0101, flui corrente nas linhas Z_2, Z_3 e Z_4, mas não em Z_1; portanto, Z_2, Z_3, Z_4 serão 1, e Z_1, um 0.

O esquema completo descrito acima realiza a ROM com portas eletrônicas (os diodos formam portas OU). Usando as técnicas de LSI, podem ser fabricadas matrizes dessa espécie a baixos custos em pequenos receptáculos. As memórias de 512 palavras de 8 bits são de tamanho aproximadamente médio para as ROM de LSI; estas memórias armazenam efetivamente 4096 bits ao todo.

Os diodos na Fig. 7.31*(a)* são, muitas vezes, transistores, e os processos de fabricação costumam ser de vários tipos. A Fig. 7.31*(b)* mostra células de memória típicas de ROM de semicondutores que usam transistores bipolares e MOS. Estas células são de um sistema de seleção unidimensional; observemos as linhas de seleção de palavra e a linha de bits. Em ROM, são utilizadas seleção unidimensional e bidimensional. As ROM de MOS geral-



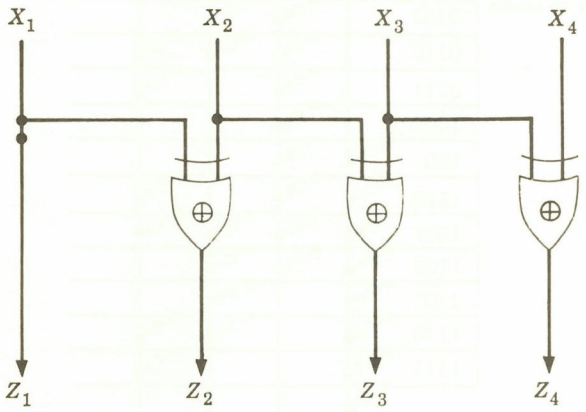


Fig. 7.30 Rede combinacional para código binário/Gray.

mente possuem tempo de acesso de 50 a 200 ns; o tempo de acesso das memórias bipolares variam de 5 a 100 ns.

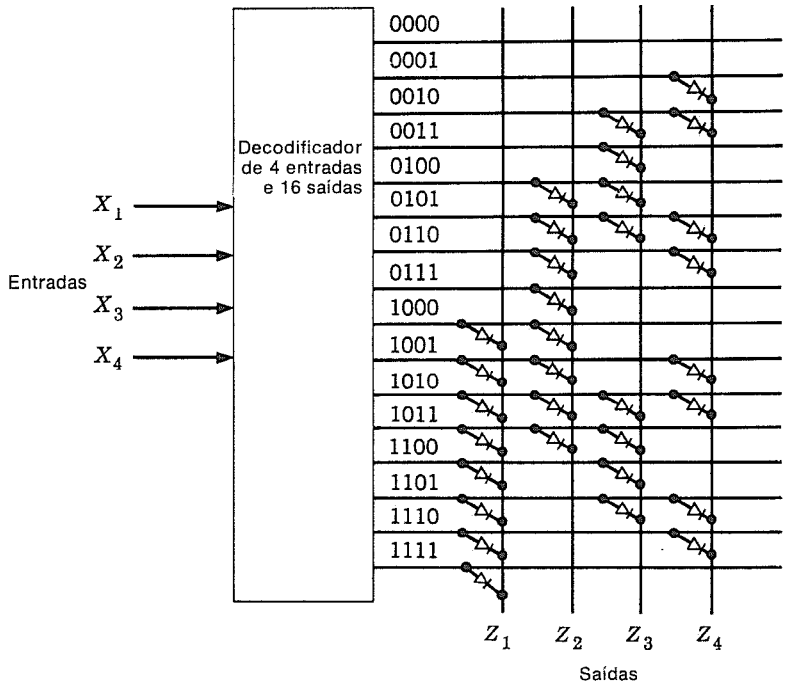
Quando uma ROM é construída de modo que o usuário possa gravar eletricamente (ou usando outras técnicas) os conteúdos da memória, a memória é chamada *ROM programável* ou PROM.† Freqüentemente é usado um esquema onde uma pastilha de memória é fornecida com 1 em cada posição, mas os 0 podem ser introduzidos em certas posições colocando-se um endereço nas linhas de entrada e elevando-se depois cada linha de saída que deve ser um 0 para uma tensão especificada, destruindo assim uma conexão para a célula selecionada. (Às vezes, as memórias contêm todas 0, e os 1 são gravados pelo usuário.) Também são fabricados dispositivos que programam PROM pela leitura de fitas de papel, fitas magnéticas, cartões perfurados etc., e colocando seus conteúdos na PROM.

Os fabricantes de ROM por encomenda fornecem formulários por meio dos quais um usuário pode preencher 1 e 0 num formulário fornecido, e o fabricante produzirá uma máscara feita de encomenda e depois produzirá pastilhas de LSI que realizarão os conteúdos de memória especificados pelo usuário. Uma única pastilha pode custar mais do que uma tal memória, mas uma grande produção geralmente torna o custo menor por pastilha.

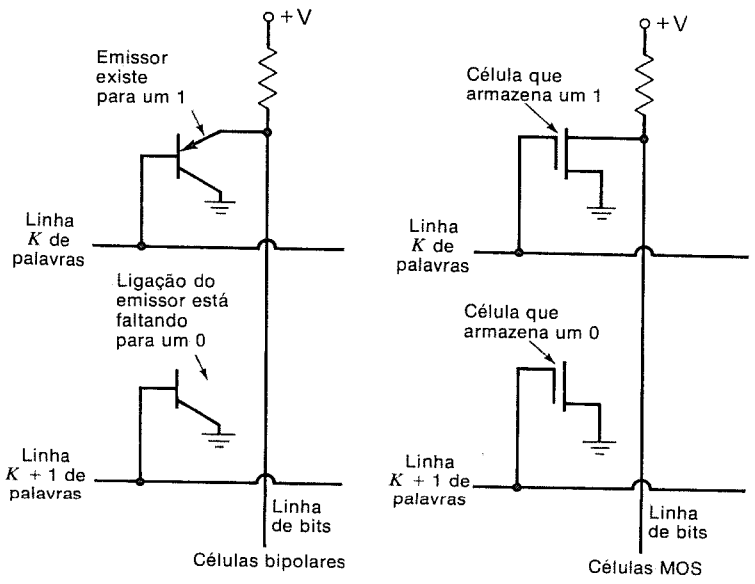
A Fig. 7.32 mostra um diagrama-bloco de uma memória de MOS de 1024 bits que é organizada como uma memória de 256 palavras de 4 bits por palavra. O usuário coloca o endereço nas oito linhas de entrada de A_0 a A_7 e depois eleva a linha de ATIVAR. O abaixamento da linha de relógio fará com que os biestáveis de saída sejam carregados, e a palavra desejada aparecerá nas linhas B_0, B_1, B_2 e B_3. Esta é uma memória fabricada de encomenda onde os conteúdos desejados da memória são fornecidos ao fabricante pelo usuário num formulário. O fabricante então faz uma máscara para criar os padrões de bits desejados numa pastilha de CI e fabrica ROM com este padrão por encomenda. O tempo de retardo da memória é da ordem de 100 ns.

Quando uma ROM é fabricada de modo que os conteúdos das memórias possam ser ajustados conforme desejado pelo usuário e a memória possa mais tarde ter os conteúdos apagados e valores novos colocados na memória, a ROM é dita ser *apagável e reprogramável*. Por exemplo, algumas pastilhas de memória são feitas com uma tampa transparente. A exposição da pastilha (através da tampa) à luz ultravioleta* apagará o padrão na pastilha, e

†Estas memórias são também chamadas *ROM de campo programável*.
*Lâmpadas ultravioleta padronizadas podem ser usadas para apagar a memória. São necessários cerca de 20 a 30 min de exposição.



(*a*)



(*b*)

Fig. 7.31 *(a)* ROM com diodos; *(b)* células de ROM usando transistores MOS e bipolares.

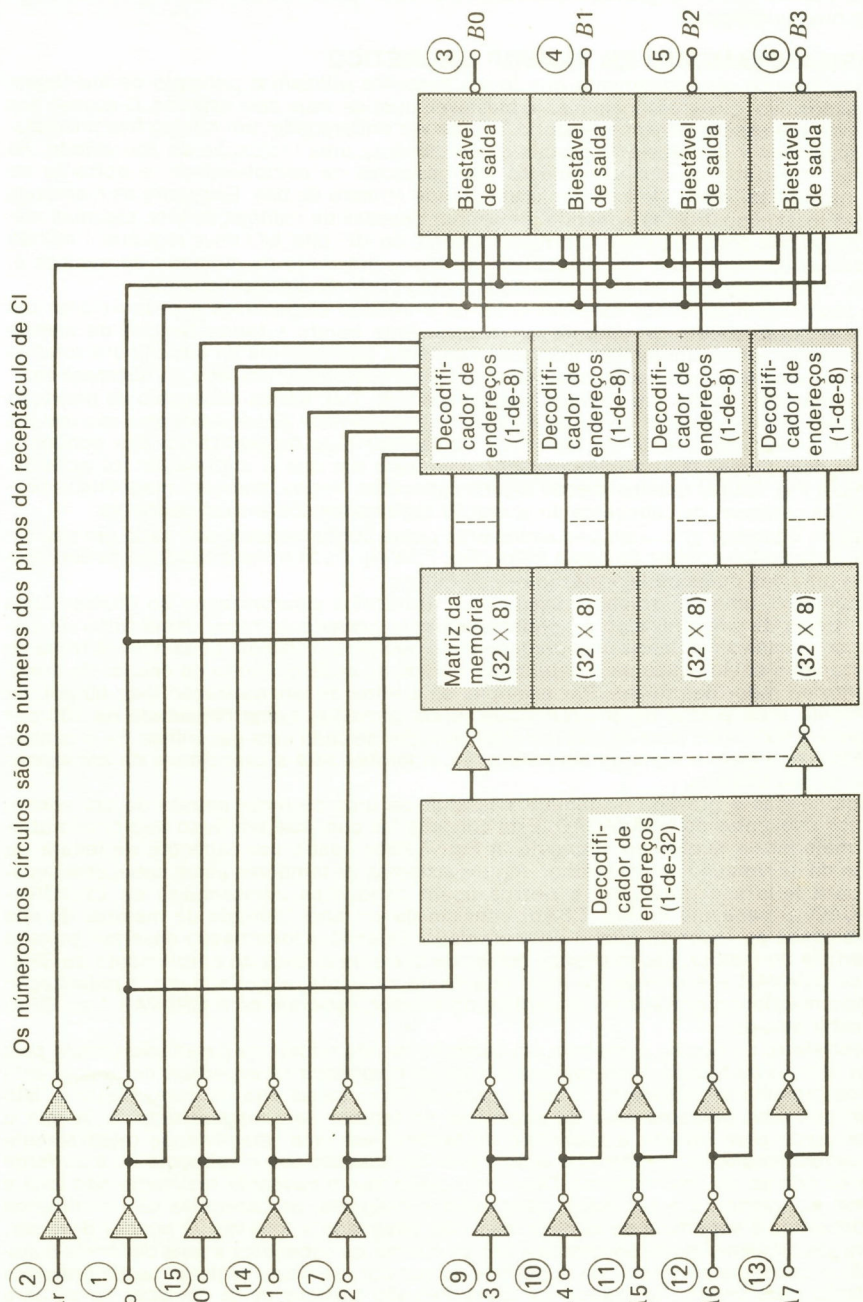


Fig. 7.32 Diagrama-bloco de ROM de CMOS de 1024 bits. O dispositivo de ROM é organizado como 256 × 4 bits. (Cortesia da Motorola Corp.)

um novo padrão pode ser gravado eletricamente nela. Isto pode ser repetido tão freqüentemente quanto desejado.

7.18 ARMAZENAMENTO EM TAMBOR MAGNÉTICO

Os dispositivos de armazenamento que foram descritos utilizam o princípio de ajustagem de um dispositivo, que é essencialmente biestável, num de seus dois estados. O acesso aos dispositivos era essencialmente aleatório — uma vez endereçado, um núcleo fica imediatamente disponível e um biestável produz continuamente uma indicação de seu estado. As limitações deste tipo de armazenamento são baseadas na complexidade, e portanto no custo e na confiabilidade, de armazenar um grande número de bits. Enquanto as memórias de núcleo foram construídas podendo armazenar dezenas de milhões de bits, algumas máquinas de grande porte requerem o armazenamento de 10^{13} bits. Isto deve requerer 1 milhão de memórias de núcleos. A alta velocidade do armazenamento da memória de núcleos é, portanto, a compensação pela complexidade e pelo custo do armazenamento.

Os tambores magnéticos estavam entre os primeiros dispositivos a proporcionar um meio de armazenamento de informação relativamente barato e tendo tempos de acesso razoavelmente curtos. Um tambor magnético consiste basicamente de um cilindro rotativo coberto por uma fina camada de material magnético que possui um laço de histerese similar ao do material usado em núcleos magnéticos (Fig. 7.4). Vários cabeçotes de gravação [Fig. 7.33*(a)*] estão montados ao longo da superfície do tambor. Estes cabeçotes são usados para ler e gravar informação da superfície do tambor pela magnetização das pequenas áreas, ou pela percepção da magnetização das áreas em que a informação foi gravada. Enquanto a Fig. 7.33*(a)* mostra apenas alguns cabeçotes, certos tambores magnéticos possuem várias centenas de cabeçotes de gravação espalhados sobre suas periferias.

Quando o tambor gira, uma pequenina área passa continuamente sob cada um dos cabeçotes. Esta área é conhecida como *trilha* [Fig. 7.33*(a)*]. Cada trilha é subdividida em células, cada uma das quais pode armazenar 1 bit binário.

Geralmente, uma das trilhas é usada para fornecer a sincronização do tambor. Uma série de sinais de sincronização é permanentemente gravada ao redor desta trilha de sincronização, e cada sinal define uma *unidade de tempo* para o sistema de tambor. A trilha de sincronização é então usada para determinar o local de cada conjunto de células de armazenamento ao redor das trilhas. Por exemplo, se a trilha de sincronização tiver 60 pol. de comprimento e os pulsos de sincronização forem gravados numa densidade de 100 por pol., haverá 6.000 locais para os bits (células) ao redor de cada uma das trilhas. Se o tambor tiver 30 trilhas e mais a trilha de sincronização, o tambor terá a capacidade de armazenar um total de 180.000 bits.

A informação é gravada no tambor passando-se uma corrente através de um enrolamento nos cabeçotes de GRAVAÇÃO. Esta corrente faz com que um fluxo seja criado através do material do núcleo do cabeçote. A Fig. 7.33*(b)* ilustra um cabeçote de leitura ou gravação de informação de um tambor. Alguns sistemas de tambores usam cabeçotes separados para leitura e gravação, e outros usam cabeçotes combinados de LEITURA-GRAVAÇÃO. O cabeçote na Fig. 7.33*(b)* consiste de um anel formado de material de alta permeabilidade ao redor do qual o fio é enrolado. Quando a informação deve ser gravada na superfície do tambor, são excitados pulsos de corrente através do enrolamento de GRAVAR 0 ou GRAVAR 1. A direção do fluxo através do cabeçote, e, por sua vez, a polarização do campo magnético gravado na superfície do tambor, depende de o GRAVAR 1 ou GRAVAR 0 estar ativado.

O entreferro no núcleo apresenta um caminho de alta relutância para fluxo gerado pela corrente através da bobina. Uma vez que o material magnético na superfície do tambor está passando próximo ao entreferro, a maior parte do fluxo passa através desse material. Isto faz com que uma pequena área da superfície do tambor seja magnetizada, e, já que o material usado para revestir a superfície do tambor tem uma retentividade relativamente alta, o campo magnético permanece após a área ter passado sob o cabeçote ou a corrente através da bobina ser interrompida. Deve ser notado que o cabeçote realmente não toca a superfície do tambor. Em vez disso, para evitar o desgaste, os cabeçotes são localizados muito próximos à superfície do tambor, mas não chegando a tocá-los. O tambor deve ser, portanto, de diâmetro muito constante a fim de manter os cabeçotes a uma distância constante. Se o cabeçote se afastar da superfície do tambor, os sinais registrados ou gravados se tornarão mais fracos. Na prática, os cabeçotes são capazes de se deslocar na ou para fora da superfície e são empurrados em direção à superfície por um mecanismo de mola de carga. Uma almofada de ar entre o cabeçote e a superfície rotativa mantém então a distância relativamente constante requerida a partir da superfície magnética.

Os sinais gravados na superfície do tambor são lidos de maneira idêntica. Quando as

áreas que foram magnetizadas passam sob o cabeçote, parte do fluxo magnético é acoplada ao cabeçote, e as variações neste fluxo induzem sinais no enrolamento. Estes sinais são depois amplificados e interpretados. Uma descrição das técnicas de gravação usadas será achada na seção final deste capítulo.

As dimensões e as capacidades de armazenamento dos tambores magnéticos variam enormemente. Tambores pequenos com capacidades inferiores a 25.000 bits têm sido construídos. Os tambores destas dimensões possuem geralmente de 15 a 25 trilhas e de 15 a 50 cabeçotes. Para diminuir o tempo de acesso, os cabeçotes são, às vezes, localizados em conjuntos ao redor da periferia do tambor, de modo que um tambor com 15 trilhas pode ter 30 cabeçotes divididos em dois conjuntos de 15 cabeçotes, cada conjunto localizado a 180° em relação ao outro. Para um tempo de acesso muito rápido, pode haver até mais de dois conjuntos de cabeçotes.†

Tambores muito maiores podem armazenar até 10^9 bits e podem ter de 500 a 1.000 trilhas. Os maiores tambores são geralmente rodados muito mais lentamente do que os tambores pequenos, e as velocidades variam de 120 rpm (em modelos de laboratório) até 75.000 rpm. Os tempos de acesso obviamente diminuem quando as velocidades do tambor aumentam; contudo, há um outro fator importante: a densidade de informação ao longo de uma trilha. A maioria dos tambores da atualidade possui uma densidade de informação desde 600 a 1.000 bits/pol.; embora mantendo os cabeçotes muito próximos à superfície do tambor e rodando o tambor lentamente, densidades de informação superiores a 1000 bits/pol. podem ser alcançadas. Um sistema de tambor, o IBM 2303, tem 800 trilhas e 4892 bytes por trilha (e portanto uma capacidade de cerca de 3,9 milhões de bytes), e o tambor efetua uma rotação completa em 17,5 ms. O Quadro 7.4 fornece algumas características dos sistemas de tambores da atualidade.

7.19 OPERAÇÃO EM SÉRIE E EM PARALELO DE UM TAMBOR MAGNÉTICO

É possível operar um tambor ou no modo em série ou no modo em paralelo. Na operação em paralelo, todos os bits de uma palavra podem ser gravados simultaneamente e lidos de maneira idêntica. Se a palavra básica do computador contiver 40 bits, o tambor pode ler 41 trilhas (uma para sincronização) simultaneamente, lendo assim uma palavra inteira de computador no tempo de um bit. Quando o tambor for lido e gravado em paralelo, é necessário um amplificador separado de LEITURA e GRAVAÇÃO para cada trilha que é usada simultaneamente, de modo que, para ler uma palavra de 40 bits num tempo de bit, são necessários 40 amplificadores de LEITURA. O conjunto correto de cabeçotes é então selecionado, e o sistema de tambor localiza o conjunto de células selecionado.

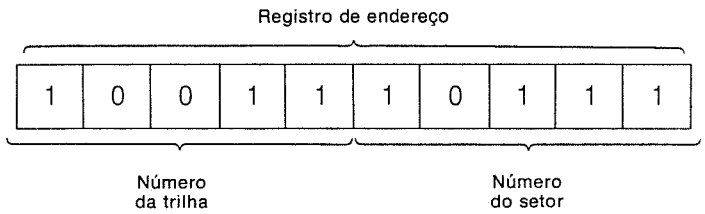
Observemos que as palavras num sistema paralelo podem ser localizadas por meio de uma trilha de sincronização. Se cada trilha contiver 8192 bits, um contador de 13 bits pode ser ajustado em 0 na mesma posição cada vez que o tambor gira e escalonado por um cada vez que aparece um pulso de sincronização. Deste modo, o local 1096 estará na milésima nonagésima sexta fenda (ranhura) ao redor da trilha a partir do local 0. Se o endereço da palavra a ser lida é carregado num registro, os sinais do tambor podem ser comandados para dentro do computador quando o computador estiver de acordo com os conteúdos do registro. Deste modo, as palavras podem ser localizadas no tambor.

Um tambor magnético também pode ser operado em série. Neste caso, apenas uma trilha será lida ou gravada num certo tempo. Uma vez que há várias trilhas para cada tambor, a trilha correta bem como o local dos bits desejados ao redor da trilha devem ser selecionados.

A cada trilha é portanto atribuído um número. Além disso, cada trilha é dividida em setores, cada setor contendo uma ou mais palavras completas de computador.

Para especificar o endereço de uma palavra num tambor magnético operado em série, tanto o número da trilha quanto o número do setor devem ser dados em primeiro lugar. Se apenas uma palavra estiver num setor, isto bastará; se mais de uma palavra estiver em cada setor, esta palavra também deve ser especificada. Consideremos um tambor com 32 trilhas mais uma trilha de sincronização e 32 palavras (setores) ao redor de cada trilha. O endereço de uma palavra no tambor numa máquina binária consistirá de 10 bits, 5 bits para especificar a trilha e 5 bits para especificar o setor. Quando gravado, o endereço conterá 10 bits. Os primeiros 5 bits podem ser usados para designar o número da trilha, e os 5 bits seguintes, o número do setor.

†O tempo de acesso real pode ser reduzido por uma codificação habilidosa do programa de computador. Se o computador gravar ou ler palavras próximo aos cabeçotes de leitura ou de gravação quando uma instrução for iniciada, o tempo de acesso poderá ser minimizado.



Os cinco biestáveis que contêm o número da trilha podem ser conectados a um *decodificador,* que selecionará depois o cabeçote correto de LEITURA-GRAVAÇÃO.

Diversas técnicas que envolvem as trilhas de sincronização podem ser usadas para localizar o setor selecionado. Uma técnica envolve o uso de várias trilhas de sincronização em vez de uma. A Fig. 7.34 mostra uma técnica que utiliza três trilhas de sincronização. Uma das trilhas contém um conjunto de sinais que indica o local de cada bit ao redor das trilhas. A segunda trilha contém um conjunto de pulsos com um pulso no começo de cada tempo de palavra. (Os sinais de tempo de palavra são afastados de 12 bits; portanto, a palavra básica deve ter 11 ou 12 bits de comprimento.) Além disso, o *número do setor* da

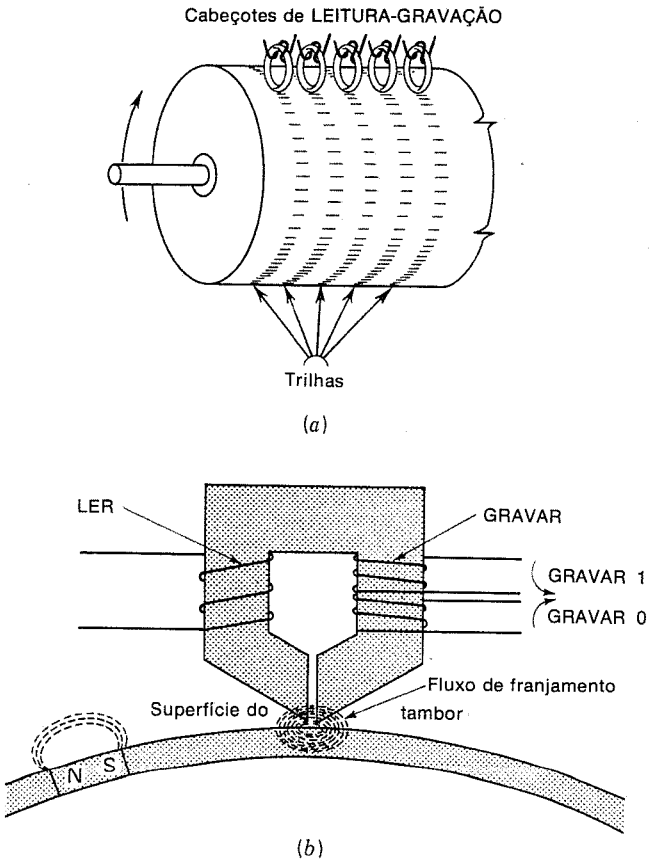


Fig. 7.33 Memória de tambor magnético. *(a)* Trilhas de tambor magnético; *(b)* cabeçote de LEITURA-GRAVAÇÃO.

QUADRO 7.4 CARACTERÍSTICAS DE TAMBORES MAGNÉTICOS

Características	FH 432 UNIVAC 1108	2301 IBM 360	2303 IMB 360	1964 ICL Série 1900	FASTRAND II UNIVAC 1108
Trilhas por tambor	144	220	800	512	6.114
Características por trilha	12.288	20.483	4.892	4.048	10.752
Caracteres por unidade	1.572.864	4.096.600	3.910.000	2.072.576	132.120.756
Velocidade de transferência, quilobits	1.440	1.200	312	100	153
Disposição do cabeçote (fixo ou móvel)	Fixo	Fixo	Fixo	Fixo	Móvel
Cabeçotes por tambor	128	200	800	512	32 × 192 posições
Diâmetro do tambor, pol.	10,5	10,7		18,5	32,8
Velocidade do tambor, rpm	7.100	3.490	3.428	1.500	870
Densidade de informação ou registro, bits/pol.	627	1.250	1.105	1.000	1.000
Tempo médio de acesso, ms	4,25	8,6	8,75	20,5	93
Custo aproximado por caráter de 6 bits armazenados, *cents*	8	5	4	3	0,2

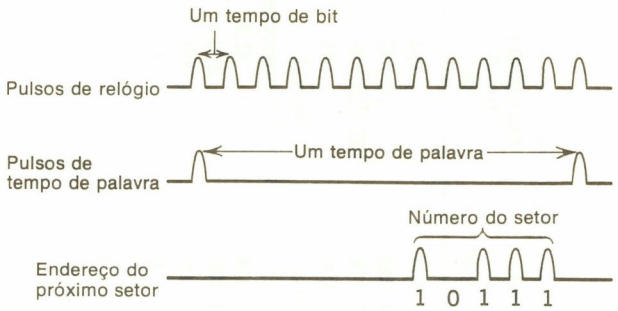


Fig. 7.34 Sinais de sincronização para um tambor magnético.

palavra *seguinte* ao redor do tambor é gravada ao redor de uma terceira trilha de sincronização. O computador lê nesta trilha os números do setor, e, quando o número lido está de acordo com o número do setor no endereço, o computador pode então ler a palavra selecionada do setor seguinte, começando com o próximo pulso de tempo de palavra.

7.20 MEMÓRIAS DE DISCOS MAGNÉTICOS

Um outro tipo de memória, chamado *memória de disco magnético,* se assemelha bastante à memória de tambor magnético quanto à operação. A memória de disco magnético proporciona capacidades de armazenamento muito grandes com velocidades de operação moderadas. Há atualmente no mercado um número consideravelmente grande de diferentes tipos de memórias de disco magnético. Embora diferindo quanto a detalhes específicos, todos eles são baseados nos mesmos princípios de operação.

No aspecto, uma memória de discos magnéticos se assemelha a um toca-discos automático, acionado por moedas, ou "vitrola automática". Vários discos rotativos revestidos de um material magnético são empilhados com espaço entre cada disco (ver Fig. 7.35). A informação é gravada na superfície dos discos rotativos por cabeçotes magnéticos que são posicionados contra os discos. (A informação é gravada em faixas em vez de numa espiral.) Cada faixa de informação ao redor de um certo disco é chamada "trilha". De um lado de um disco típico pode haver de uma centena a vários milhares destas trilhas de dados. Os bits são gravados ao longo de uma trilha numa densidade de 500 a 7000 bits/pol. aproximadamente. Em alguns sistemas, as trilhas exteriores contêm mais bits que as trilhas interiores, porque a circunferência de uma trilha externa é maior do que a de uma trilha interna, porém muitos discos possuem o mesmo número de bits ao redor de cada trilha. A velocidade em que os discos giram varia, certamente, com o fabricante, mas as velocidades típicas são da ordem de 3.000 rpm.

Uma vez que cada disco contém várias trilhas de informação e pode haver diversos discos numa certa memória, inúmeras técnicas têm evoluído para a colocação do cabeçote magnético de LEITURA e GRAVAÇÃO na posição correta numa trilha selecionada. Uma vez que o mesmo cabeçote é geralmente usado na leitura e na gravação, o problema se resume então na colocação deste cabeçote com precisão e rapidez sobre a trilha que foi selecionada.

Há dois tipos básicos de sistemas de disco. O primeiro tipo de sistema tem seus cabeçotes fixos em posição sobre cada trilha. Estes são chamados *sistemas de cabeçote fixo.* A segunda espécie de sistema tem um ou mais pares de cabeçotes de LEITURA-GRAVAÇÃO para cada par de superfícies de discos adjacentes (porque a informação é geralmente gravada em ambas as partes superior e inferior de cada disco). Estes cabeçotes de LEITURA-GRAVAÇÃO são montados sobre braços que podem ser movidos para dentro e para fora. Estes são os chamados *sistemas de cabeçote movível.*†

O posicionamento dos cabeçotes por meio do movimento mecânico dos braços constitui uma tarefa difícil e enganadora, particularmente se as trilhas forem muitas vezes grava-

†Alguns sistemas foram feitos com apenas um par de cabeçotes de LEITURA-GRAVAÇÃO na memória inteira. Nestes sistemas, os dois cabeçotes de gravação são posicionados num braço que é inicialmente movido entre o par correto de discos, e depois seleciona a superfície correta das superfícies adjacentes (novamente porque a informação é gravada tanto na parte superior quanto na inferior de cada disco). O cabeçote de LEITURA-GRAVAÇÃO é finalmente colocado sobre a trilha correta selecionada do disco.

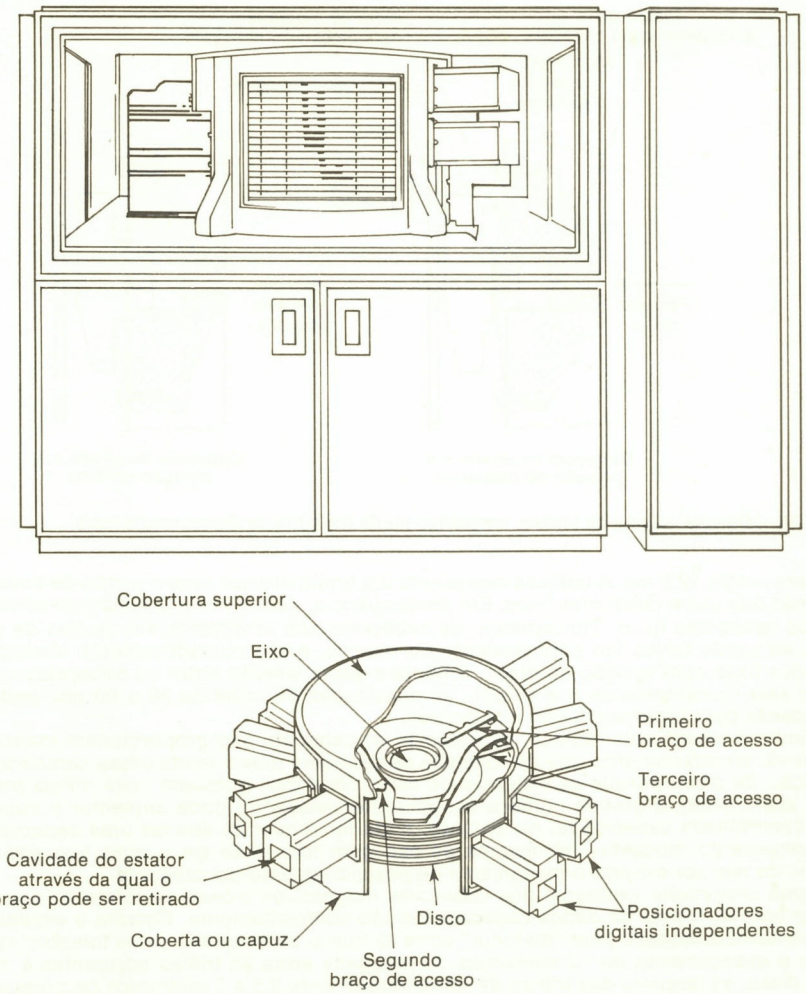


Fig. 7.35 Sistema de memória de discos magnéticos.

das a uma distância inferior a centésimos de polegada no disco. É evidente que as memórias de arquivo em discos de muitos cabeçotes podem localizar e gravar ou ler numa trilha selecionada mais rápido do que as memórias que têm apenas alguns cabeçotes, uma vez que o movimento mecânico antes de a trilha ser alcançada será de valor menor no sistema de múltiplos cabeçotes.

O tempo total consumido para começar a leitura dos dados selecionados ou para começar a gravação de uma trilha selecionada num local particular é chamado *tempo de acesso*.

O tempo necessário para posicionar um cabeçote sobre a trilha selecionada é chamado *tempo de busca* e está geralmente situado entre uma média de vários milissegundos e frações de segundo. O outro retardo na localização de dados selecionados é o *retardo* rotacional ou de latência, que é o tempo necessário para que os dados desejados alcancem o cabeçote magnético uma vez que o cabeçote esteja posicionado. Assim, o tempo de acesso total de um disco é a soma do tempo de busca e da latência.

Para uma velocidade rotativa de 2.400 rpm, por exemplo, a latência é um máximo de 25

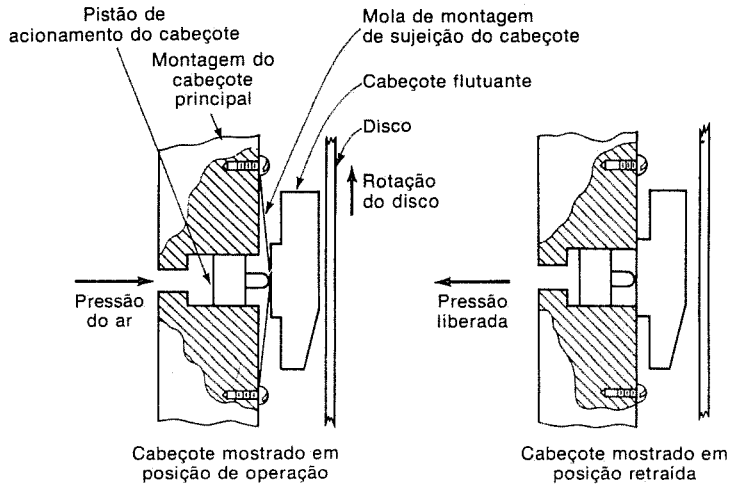


Fig. 7.36 Cabeçote volante de tambor magnético ou de memória de discos magnéticos.

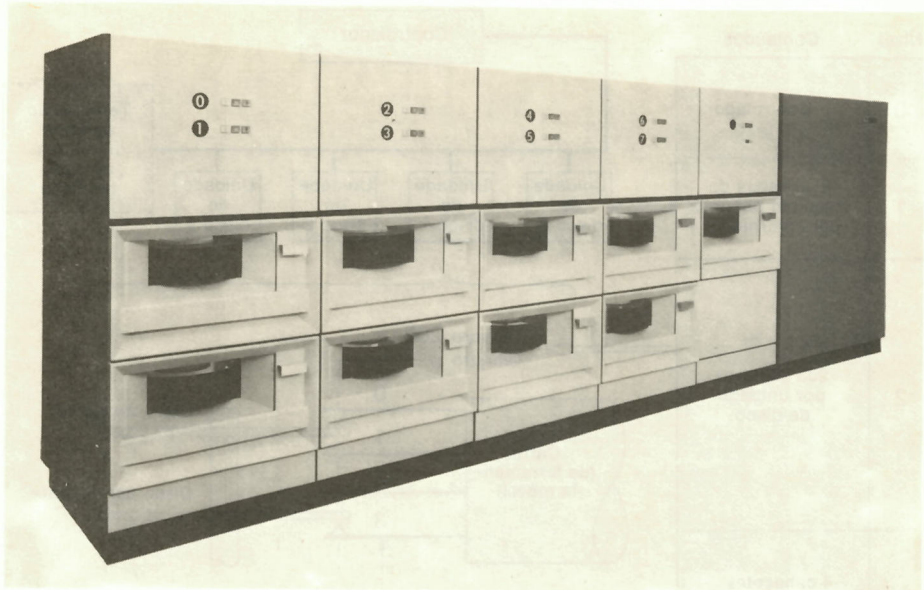
ms e em média 12,5 ms. A latência representa um limite inferior para o tempo de acesso nos sistemas que usam cabeçotes fixos. Em conseqüência, para o mínimo tempo de acesso, são usados cabeçotes fixos. Tipicamente, os cabeçotes são arranjados em grupos de oito ou nove, incluindo talvez um sobressalente ou reserva, e são cuidadosamente alinhados em posições fixas com relação ao disco. Embora o espaçamento entre os cabeçotes em cada grupo seja tipicamente de 8 a 16/pol., as densidades de trilha de 30 a 60/pol. podem ser alcançadas pelo entrelaçamento dos grupos.

Embora sejam mais rápidos, os sistemas de cabeçote fixo proporcionam menor capacidade de armazenamento que os sistemas de cabeçote móvel tendo áreas comparáveis de gravação de disco porque os sistemas de cabeçote móvel possuem mais trilhas por polegada. Além disso, o grande número de cabeçotes requerido pode aumentar o custo para uma determinada capacidade; no entanto, onde for requerida apenas uma capacidade de armazenamento modesta, um sistema de cabeçote fixo pode ser o mais econômico, em virtude de não ser exigido o mecanismo de posicionamento do cabeçote.

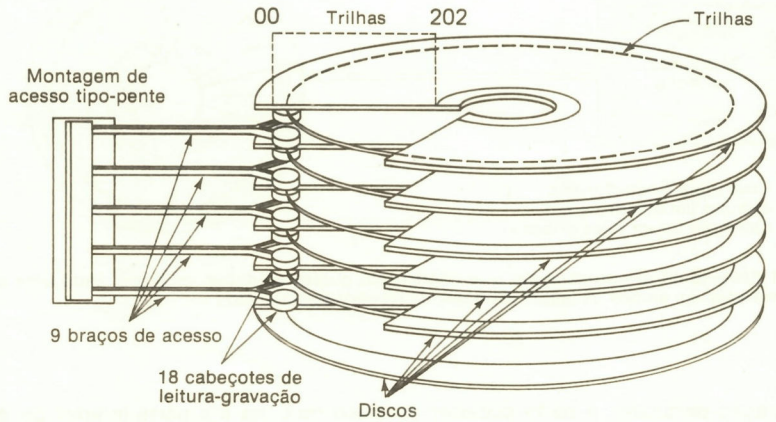
Uma importante vantagem dos cabeçotes magnéticos móveis se refere ao seu alinhamento com as trilhas de dados espaçadas muito compactamente. Embora o espaçamento das trilhas seja limitado pela "diafonia" entre as trilhas adjacentes e pelas tolerâncias mecânicas, o espaçamento de 10 milésimos de polegada entre as trilhas adjacentes é comum. Além disso, as larguras das trilhas de aproximadamente 3,5 a 5 milésimos de polegada são consistentes com as precisões de 0,5 a 1,5 milésimos de polegada do posicionamento do cabeçote.

Os cabeçotes de LEITURA e GRAVAÇÃO usados nas memórias de discos magnéticos são quase invariavelmente de um tipo denominado *cabeçotes volantes*. Um diagrama simplificado de um cabeçote volante é mostrado na Fig. 7.36. Quando um disco gira numa alta velocidade, uma camada limite fina mas resiliente de ar gira com o disco. O cabeçote é de tal forma que ele funciona sobre esta camada de ar rotativa, o que faz com que o disco mantenha a separação do cabeçote, evitando assim o desgaste na superfície do disco. Com efeito, a camada de ar que gira com o disco age como uma mola cuja rigidez ultrapassa a vários milhares de libras por polegada, forçando assim o cabeçote para longe da superfície do disco. Para forçar o cabeçote na proximidade correta com o disco, têm sido usados vários mecanismos. Muitas vezes, sopra-se simplesmente ar comprimido dentro de um mecanismo que força o cabeçote em direção à superfície do disco, usando um dispositivo idêntico a um pistão, conforme indicado na Fig. 7.36.

Há muitos tamanhos e velocidades para as memórias de disco. Alguns discos são muito grandes, tendo até 4 pés de diâmetro (1 pé = 30,48 cm). Outros são menores, giram mais rápidos, são mutáveis etc. Em virtude do grande mercado para estas memórias e da variedade aparentemente infinita de configurações em que elas podem ser fabricadas, ao usuário do sistema é concedida ampla liberdade na escolha.



(*a*)



(*b*)

Fig. 7.37 *(a)* Memória de discos magnéticos do tipo IBM 2314; *(b)* unidade de discos para a IBM 2314 (apenas cinco discos são mostrados).

As memórias de discos que possuem conjuntos de discos mutáveis (ou *módulos*†) são muito populares. Cada conjunto de discos contém discos que giram juntos. A memória IBM 2314, mostrada na Fig. 7.37*(a)*, usa um conjunto de discos mutáveis, por exemplo, que contém 20 superfícies de discos gravadas, consistindo de 10 discos. [Ver Fig. 7.37*(b)*.] Nesta

†Um *módulo* é uma unidade de disco que tem os cabeçotes de LEITURA-GRAVAÇÃO e os braços de posicionamento todos embalados junto com os discos. Estes módulos são dispendiosos, mas possibilitam um sistema com desempenho mais alto.

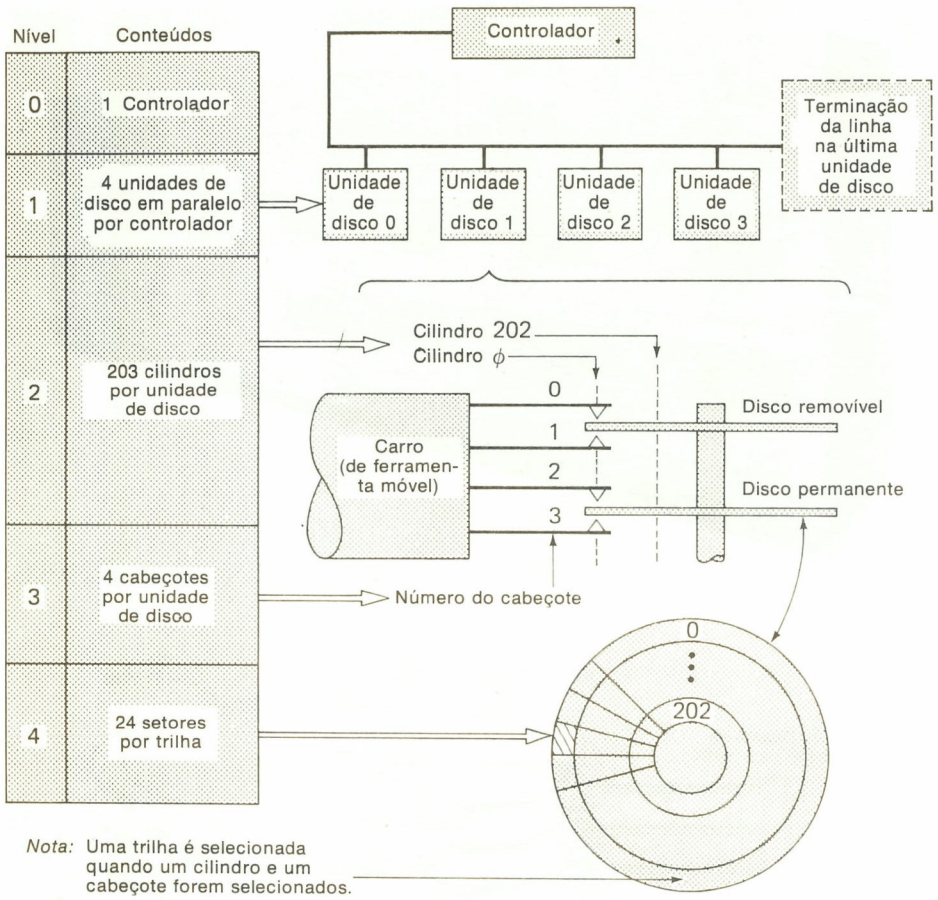


Fig. 7.38 Sistema de armazenamento que contém até quatro unidades de disco, cada uma com dois discos gravados em ambos os lados. (Cortesia de Hewlett-Packard Co.)

configuração particular, a parte superior do disco de cima e a parte inferior do disco de baixo não são usadas. São empregados dezoito cabeçotes de LEITURA-GRAVAÇÃO, dois para cada superfície sobre a qual a informação é gravada. Os discos giram a uma velocidade de 2.400 rpm. Os cabeçotes são montados num mecanismo de posicionamento conforme indicado. Cada superfície do disco é dividida em 200 trilhas magnéticas concêntricas por polegada e cada trilha contém cerca de 4.400 bits/pol.

A Fig. 7.38 mostra as características de um sistema de discos a preço inferior fabricado pela Hewlett-Packard.

O Quadro 7.5 lista as características de diversos dos maiores sistemas de disco, e o Quadro 7.6, algumas das características de um dos menores e menos dispendiosos sistemas.

Devido ao custo por bit de informação armazenada nas memórias de disco ser relativamente baixo e por causa dos tempos de acesso relativamente baixos e as velocidades de transferência atingíveis serem altas quando da leitura ou da gravação de dados num arquivo de discos, as memórias de disco magnético tornaram-se um dos mais importantes dispositivos de armazenamento nos modernos computadores digitais.

QUADRO 7.5 TECNOLOGIA DAS UNIDADES DE DISCO/COMANDOS DE DISCO

Tecnologia das unidades de disco	IBM 2314	CDC 215	UNIVAC 8440	IBM 3330	IBM 3330-11	IBM 3340
Projeto da unidade	IBM 2316	IBM 2316	2316*	IBM 3336	IBM 3336-11	IBM 3348
Número de superfícies	20	20	20	19	19	24
Capacidade armazenamento em megabytes	29	54	108	100	200	35 a 70
Trilhas por polegada	200	200	200	192	384	300
Transições por polegada	4.400	4.400	4.400	4.040	4.040	5.072
Velocidade dos dados (milhão de bits/s)	2,5	2,5	5,0	6,5	6,5	7,1
Tempo médio de acesso (ms)	60	30	30	30	30	25
Rpm	2.400	2.400	2.400	3.600	3.600	3.000
Detentor	Mecânico†	Ótico	Ótico	Servo	Servo	Servo
Acionador	Hidráulico†	Bobina de voz	Bobina de voz	Bobina de voz	Bobina de voz	Bobina de voz
Revestimento do disco, mínimo (μpol.)	90	90	50	50	40	40
Altura de vôo do cabeçote, mínimo (μpol.)	80	80	70	50	30	30

*Unidade especial feita com *hardware* da unidade 2316 e com discos 3336.
†Os detectores não-híbridos da IBM existem com bobina de voz e máscara ótica.

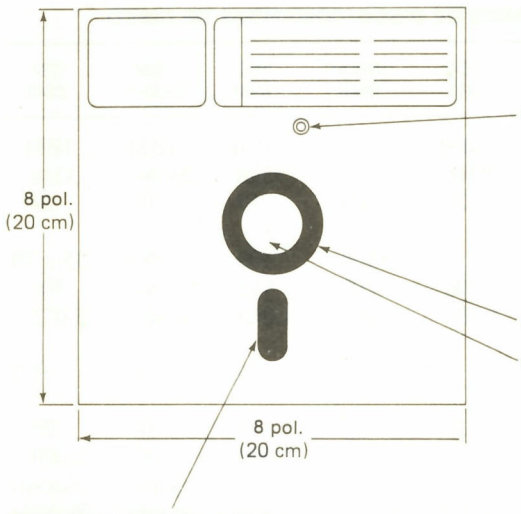
7.21 SISTEMAS DE ARMAZENAMENTO EM DISCO FLEXÍVEL — O DISCO *FLOPPY*

Uma inovação no armazenamento em discos, originalmente desenvolvido na IBM, usa um disco pequeno, magnético, substituível (disco *floppy*), com uma base de plástico, em lugar do disco mais rígido, convencional com base metálica. Este meio de armazenamento tem aproximadamente a forma e as dimensões de um disco de 45 rpm e pode ser "encaixado" quase tão facilmente quanto um cartucho ou estojo de fitas. Cada disco *floppy* custa apenas alguns dólares.

Os discos *floppy* são mutáveis e cada um deles vem num envoltório conforme indicado na Fig. 7.39. Os discos são montados em unidades de discos com o envoltório no lugar, e a informação é gravada e lida através de uma abertura no envoltório. (Alguns sistemas requerem a remoção do envoltório.) Vários fabricantes atualmente fornecem sistemas completos a mil dólares para uso com estes discos. A conveniência do uso e dos preços baixos ampliaram o emprego de memórias de discos *floppy* em muitas aplicações.

Na maioria das unidades de discos *floppy*, a montagem do cabeçote de LEITURA-GRAVAÇÃO está em real contato físico com o material de gravação. (Para aumentar a vida, o contato do cabeçote é geralmente mantido apenas no instante da leitura ou da gravação.) O disco flexível é encerrado numa jaqueta ou envoltório quadrado de 8 pol., e o disco tem um diâmetro de 7,88 pol. A superfície de gravação é uma camada espessa de 100 micropolegadas de óxido magnético num substrato de poliéster de 0,003 pol. de espessura. A jaqueta proporciona proteção contra o manuseio, e, além disso, tem um forro especial que fornece uma ação de limpeza para remover os produtos de desgaste e outras sujeiras que devem ser abrasivos para o meio e para o cabeçote se deixado na superfície.

Na versão da IBM há um único orifício de 1,5 pol. de índice de diâmetro 0,100 pol. a partir do centro do disco para indicar o início de uma trilha. Uma trilha gravada tem largura de 0,012 pol., e o espaçamento padronizado das trilhas é de 48/pol. O número de trilhas é 77. A capacidade de uma superfície que usa um código padrão e uma densidade de bits de



Orifício indicador

O círculo exterior mostra um orifício na jaqueta; o círculo interior mostra o orifício indicador no disco. Quando estes dois orifícios forem alinhados enquanto o disco gira durante as operações de processamentos de dados, um feixe de luz que brilha num lado do pequeno disco flexível (diskette) é percebido do outro lado e usado para funções de sincronização.

Abertura de acesso de acionamento na jaqueta

Orifício do eixo de acionamento no disco

Após o diskette ter sido colocado na máquina e o eixo de acionamento do disco ter sido introduzido no orifício do eixo de acionamento no disco, o mecanismo de acionamento aperta uma porção do disco exposta pela abertura de acesso de acionamento na jaqueta.

Fenda da almofada de pressão
(Uma abertura idêntica no outro lado do diskette é chamada fenda do cabeçote.)

A fenda do cabeçote expõe a superfície de gravação do disco quando o disco gira em sua jaqueta na máquina. A unidade de leitura e gravação de dados do disco, que é chamada *cabeçote de leitura-gravação* e é idêntica ao cabeçote de gravar/reproduzir num gravador de fita, se desloca para posições especificadas ao longo do comprimento da fenda. O movimento para uma posição específica é chamado *acesso a uma trilha*. (Os dados são gravados apenas no lado do diskette que contém a fenda do cabeçote.)

Fig. 7.39 Disco flexível *(floppy)* com invólucro.

3.268 bits/pol. na trilha mais interna é de cerca de 400.000 bytes de 8 bites cada um. O Quadro 7.7 mostra algumas características dessa espécie de sistema de disco flexível.†

Várias empresas oferecem sistemas de discos flexíveis *(floppy)* que usam adaptações do conceito de cabeçote volante. Um exemplo de sistema é o fabricado pela Pertec, e suas especificações estão indicadas no Quadro 7.8. A maioria dos discos flexíveis usa um sistema de endereços (como o de discos) onde a unidade de disco, a trilha e o setor são fornecidos; no entanto, alguns sistemas gravam "cabeçalhos" em cada bloco de dados gravados numa trilha, e a informação do cabeçalho é especificada em cada acesso.

7.22 FITA MAGNÉTICA

Atualmente, o meio mais popular para armazenamento de quantidades muito grandes de informação é a fita magnética. Embora, por causa de seu longo tempo de acesso, a fita magnética não seja um meio desejável para o principal armazenamento em alta velocidade de um computador, as modernas técnicas de produção em massa tornaram o custo da fita muito baixo, de modo que enormes quantidades de informação podem ser armazenadas sem despesas. Além disso, já que é possível apagar e regravar informações na fita, a mesma fita pode ser usada repetidas vezes. Uma outra vantagem da fita magnética é que a informação armazenada não "se desvanece", e, portanto, os dados ou programas armazenados

†A IBM usa somente cerca de 250.000 bytes para dados; os bits restantes são usados para formatação, correção de erro e sinais de sincronização.

QUADRO 7.6 ESPECIFICAÇÕES DA UNIDADE DE DISCOS HP 2888A

Característica	Especificação
Tempo de acesso de dados	
Mínimo	7 ms
Médio	29 ms
Máximo	55 ms
Retardo rotacional (latência)	
Médio (1/2 revolução)	12,5 ms
Máximo (1 revolução)	25 ms
Tempo médio da resposta global	
Posição e rotação	41,5 ms
Resposta máxima	80 ms
Características operacionais	
Partida	20 s
Tempo de parada	Máximo de 25 s
Velocidade de rotação do disco	2.400 rpm
Número de cabeçotes	20
Número de cilindros	40 (20 trilhas por cilindro)
Capacidade de armazenamento	47.104.000 bytes
Velocidade de transferência de dados	320.000 bytes/s; velocidade (média) de rompimento de 236.000 bytes/s
Acionador	Acionador eletromagnético com posicionamento de servo ótico de malha fechada
Unidade de disco — HP 30333A	
Superfícies	20
Trilhas/superfície	400, mais 6 sobressalentes

QUADRO 7.7 CARACTERÍSTICAS DE UMA UNIDADE DE DISCOS FLEXÍVEIS COMPATÍVEL DA IBM

Característica	Especificação
Capacidade*	400.000 bytes
Velocidade rotacional	360 rpm
Velocidade de transferência	250.000 bits/s
Tempo de acesso trilha a trilha	16-20 ms
Tempo médio de acesso	176 ms
Densidade de bits	
Trilha interna	3268 bits/pol.
Trilha externa	1836 bits/pol.
Densidade de trilhas	48 trilhas/pol.
Número de trilhas	77

*Esta é um máximo; usando-se o sistema de gravação formatado da IBM, somente 250.000 bytes de dados são gravados por disco.

QUADRO 7.8 ESPECIFICAÇÕES DA UNIDADE DE DISCOS FLEXÍVEIS (Cortesia da Pertec Corporation)

Característica	Especificação
Médias	*diskette* da IBM ou equivalente
Trilhas/pol.	48
Número de trilhas	77
Largura da trilha	0,012 pol.
Parâmetros de gravação	
Velocidade de transferência	250 quilobits/s
Densidade de bits	6631 bits/rad
Máximo de bits por disco (não-formatado)	3,2 milhões
Máximo de bits por trilha (não-formatada)	41.665
Formato*	
Setores por trilha	26
Trilhas indicadoras	1
Trilhas de dados	73
Trilhas sobressalentes	2
Trilhas reservadas	1
Bytes por setor	128
Desempenho	
Velocidade do disco	360 rpm ± 1,5%
Tempo de início/parada	3 s máximos
Tempo de busca	10 ms trilha a trilha
Tempo de fixação do cabeçote	10 ms (última trilha endereçada)
Tempo de carregamento do cabeçote	20 ms máximos
Taxa de erros (máxima)	1 por 10^9 recuperado, 1 por 10^{12} não-recuperado
Vida no cabeçote	10.000 h em uso normal
Vida média	3500 h
Dimensões	3,45 pol. de altura, 8,58 pol. de largura, 13,28 pol. de profundidade
Peso	5,5 kg máximos

*Compatível com os discos gravados nos sistemas IBM 3540 ou 3740. Facilmente interligados com os sistemas não-compatíveis da IBM.

durante um mês podem ser usados novamente no mês seguinte.

Uma outra vantagem da utilização de fita magnética para armazenamento de grandes quantidades de dados surge do fato de que os carretéis de fita num mecanismo de fita podem ser trocados. Deste modo, o mesmo mecanismo de manipulação de fita magnética e seu conjunto associado de circuitos pode ser usado com muitos carretéis diferentes de fita, cada carretel contendo dados diferentes.

Há quatro partes básicas de um sistema digital de fita magnética, a saber:

1. *Fita magnética* Este é geralmente uma fita plástica flexível com uma delgada camada de algum material ferromagnético ao longo da superfície.
2. *Mecanismo de transmissão da fita* Este consiste de um mecanismo projetado para movimentar uma fita diante dos cabeçotes de gravação sob o comando do computador. Estão incluídos os próprios cabeçotes e as instalações de armazenamento da fita que está sendo usada, bem como os carretéis sobre os quais a fita é enrolada.
3. *Sistema de leitura e gravação* Esta parte do sistema inclui os amplificadores de leitura e gravação e os "tradutores" que convertem os sinais da fita em sinais digitais que podem ser usados no sistema de computação central.

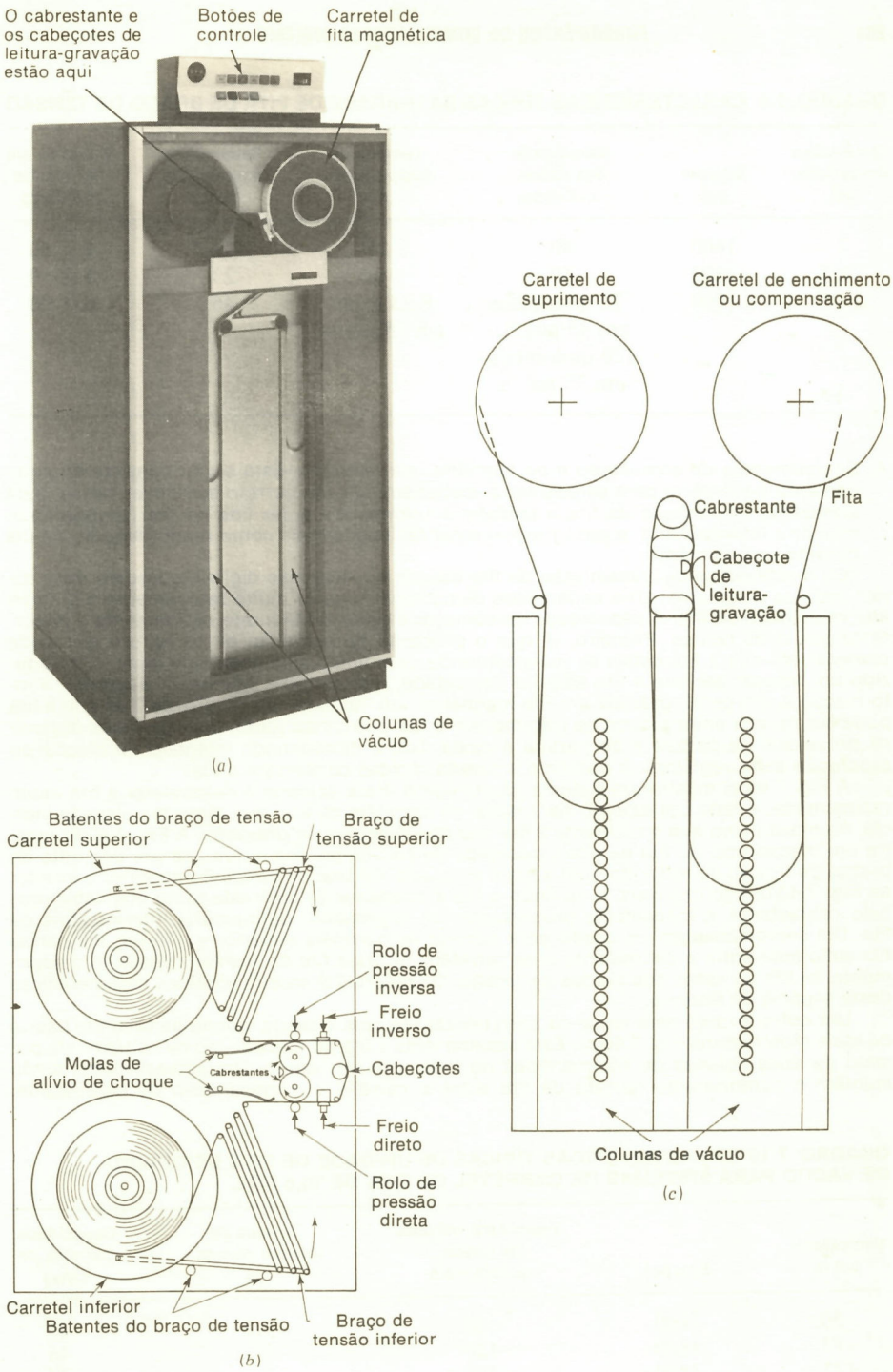


Fig. 7.40 *(a)* Sistema de fita IBM 3420; *(b)* Mecanismo de fita magnética que usa braços de tensão; *(c)* Mecanismo de fita magnética que usa colunas de vácuo.

QUADRO 7.9 CARACTERÍSTICAS TÍPICAS DA UNIDADE DE FITA DE BRAÇO DE TENSÃO

Dimensões do carretel, pol.	Bits por pol.	Velocidade dos dados, quilobits	Tempos de início-parada, ms	Capacidade, milhões de bytes	Custo de um carretel de fita típico
7	1600	40	15	11,5	$ 7.50
8,5	1600	60	10	23	$ 8.50
10,5	1600	72 quilobits/s em 45 pol./s; 120 quilobits/s em 75 pol./s	8,33 (5 ms em 75 pol./s)	46	$11.00

4. *Equipamentos de comutação e de memória intermediária* Esta seção consiste do equipamento necessário para selecionar o mecanismo de fita correto se houver vários, para armazenar informação da fita e também a informação a ser lida na fita (proporcionar memória intermediária) e para proporcionar facilidades tais como reenrolamento da fita dirigido manualmente.

Os mecanismos de transmissão da fita usados em sistemas digitais possuem duas características singulares: (1) a capacidade de começar e parar muito rapidamente e (2) uma alta velocidade de fita. A capacidade de começar e parar a fita muito rapidamente é importante por duas razões. Primeiro, já que o processo da gravação ou da leitura não pode começar até que a fita esteja se movimentando com uma velocidade suficiente, é introduzido um retardo até que a fita adquira velocidade, retardando a operação. Segundo, a informação é, em geral, gravada em fita magnética em "blocos" ou "registros". Já que a fita pode ser parada entre blocos de informação, a fita que passa junto aos cabeçotes durante os processos de parada e de partida é gasta. Isto é denominado *intervalo de blocos* ou *separação entre registros.* A partida e a parada rápidas conservam a fita.

A Fig. 7.40*(a)* mostra um sistema de fita típico. Para acelerar e desacelerar a fita muito rapidamente, é feito um esforço para isolar os carretéis de fita, que têm uma elevada inércia, do mecanismo que movimenta a fita sob os cabeçotes de gravação. A Fig. 7.40*(b)* mostra um mecanismo de fita de alta velocidade de partida-parada e que usa um conjunto de braços de tensão ao redor do qual a fita é enlaçada. Os braços de tensão superior e inferior na Fig. 7.40*(b)* são movíveis, e, quando a fita é subitamente acionada junto aos cabeçotes pelo cabrestante, o mecanismo proporciona um suprimento de memória intermediária de fita. Um servomecanismo é usado para acionar os carretéis superior e inferior, mantendo fita suficiente entre o cabrestante e os carretéis de fita a fim de manter constante o suprimento de fita ao redor dos braços de tensão. O Quadro 7.9 mostra algumas características desta espécie de sistema.

Um outro arranjo para isolar os carretéis de fita com elevada inércia da unidade básica de fita é mostrado na Fig. 7.40*(c)*. Este sistema isola a fita do mecanismo de cabrestante por meio de duas colunas de fita mantidas no lugar por um vácuo. Um servossistema então mantém o comprimento correto de fita entre o carretel e o mecanismo de cabrestante.

QUADRO 7.10 CARACTERÍSTICAS TÍPICAS DE UNIDADE DE FITA DE COLUNA DE VÁCUO PARA SISTEMAS DE CARRETEL DE FITA DE 10,5 POL.

Velocidade, pol./s	Bits/pol.	Velocidade máxima de dados, quilobytes/s	Tempos de início e parada, ms	Capacidade, milhões de bytes
50	1600	80	7,5	46
75	1600	120	5	46
125	1600	200	3	46
125	6250	780	1,2	180

Tanto este quanto o sistema anterior usam cabrestantes que giram continuamente para acionar realmente a fita, e "rolos de pressão" para comprimir a fita contra o cabrestante quando o mecanismo de transmissão for acionado. Também são fornecidos freios para uma parada rápida. O Quadro 7.10 mostra alguns dados típicos desta espécie de sistema.

Usando sistemas desta categoria, os tempos de início e de parada podem ser inferiores a 5 ms. Estes são os tempos requeridos para acelerar uma fita a uma velocidade adequada à leitura ou gravação e o tempo requerido para parar completamente uma fita que se move. As velocidades em que as fitas são movidas junto aos cabeçotes variam enormemente, e a maioria dos mecanismos de transmissão da fita tendo velocidades na faixa de 12,5 a 250 pol./s. Um mecanismo muito rápido (900 pol./s) foi construído no Laboratório Lincoln em que a fita é acionada diretamente pelos carretéis do mecanismo de transmissão. Um sistema de fita desta natureza é projetado para transferir somente grandes quantidades de dados para/do armazenamento principal de acesso rápido da máquina numa única operação. As mais lentas velocidades da fita, combinadas com a rápida partida e parada, são provavelmente mais adaptáveis aos sistemas nos quais menores quantidades de dados devem ser transferidas. Alguns sistemas possuem estojos ou cartuchos mutáveis com um carretel de fita em cada cartucho. Os fabricantes destes sistemas sentem que isto protege a fita e facilita a troca dos carretéis. Isto será discutido numa seção seguinte.

A maioria dos sistemas de fita possui cabeçotes de LEITURA-GRAVAÇÃO de dois intervalos (ou entreferros). Os dois entreferros (ver Fig. 7.41) são úteis porque, durante a gravação, o entreferro de LEITURA é posicionado depois do entreferro de GRAVAÇÃO e é usado para verificar o que foi gravado por leitura e comparação.

As fitas variam de 1/4 a 3 pol. de largura; no entanto, a maioria das fitas é do tipo *mylar* de 1/2 pol. de largura e 1,5 milésimo de polegada de espessura. Um carretel de 10,5 pol. tem tipicamente 2.400 pés de fita. Geralmente, cerca de nove canais ou trilhas são usados para 1/2 pol. de largura. A superfície da fita está usualmente em contato com o cabeçote de LEITURA-GRAVAÇÃO. Os sinais de saída dos cabeçotes de LEITURA são geralmente da faixa de 0,1 a 0,5 V. A densidade de gravação varia; no entanto, 200, 556, 800, 1600 e 6250 bits/pol. por canal são nitidamente padrões.

Os dados são gravados em fita magnética usando algum sistema de codificação. Geralmente, é armazenado um caráter por fila (ver Fig. 7.42) ao longo da fita. A fita na Fig. 7.42 tem sete trilhas, ou canais, uma das quais é um bit de paridade, que é acrescentado para tornar ímpar o número de 1 em cada fila (isto será estudado no próximo capítulo, assim como serão os códigos usados em fita magnética). Os dados são gravados em blocos nas fitas magnéticas, com intervalos entre os blocos e usualmente com caracteres únicos de partida e parada para assinalar o início e o fim de um bloco.

Uma pequena peça de material metálico refletor é presa à fita no início e no fim do carretel, e são usadas células fotoelétricas para perceber ou ler estes marcadores e evitar a ultrapassagem da fita [ver Fig. 7.43*(a)*].

Os códigos usados para gravar em fita variam, mas dois deles comumente usados são os códigos da IBM mostrados na Fig. 7.43*(b)* e *(c)*. A fita padrão da IBM tem 1/2 pol. de largura e 1,5 milésimo de polegada de espessura, com sete ou com nove trilhas. O código de sete trilhas é mostrado na Fig. 7.43*(b)*, onde os 0 são simplesmente brancos e os 1 são indicados por uma linha vertical. A Fig. 7.43*(c)* mostra o código de nove trilhas. São as seguintes as densidades de gravação: 200, 556, 800, 1600, ou 6250 bits (ou filas) por polegada (o que significa 200, 556, 800, 1600 ou 6250 caracteres (ou bytes) por polegada, já que um caráter é gravado em cada fila).

Algumas outras características dos sistemas de fita são mostradas no Quadro 7.11, e o Quadro 7.12 apresenta as características de um sistema de Hewlett-Packard a custos baixos.

7.23 ESTOJOS E CASSETES DE FITA

O cassete de fita mutável usado nos gravadores caseiros é um meio atrativo de gravação de dados digitais. Os cassetes são pequenos, mutáveis e de custos baixos. Infelizmente, a qualidade da fita e o mecanismo de movimentação da fita no cassete de fita "caseiro" convencional não satisfazem aos requisitos para uso em computadores. Em conseqüência, vários cassetes digitais de alta qualidade com preços na faixa do dólar (em geral de 2 a 15 dólares) foram desenvolvidos. Estes cassetes são de pequenas dimensões — da ordem da dos cassetes familiares — e têm um aspecto semelhante.

Há também *estojos de fita* maiores que contêm longas tiras de fita magnética e que se assemelham aos grandes cassetes. Estes estojos fornecem um modo mais conveniente para embalar a fita e simplificam enormemente a montagem dos carretéis de fita (que podem constituir um problema com os carretéis de fita convencionais onde a fita deve ser posicio-

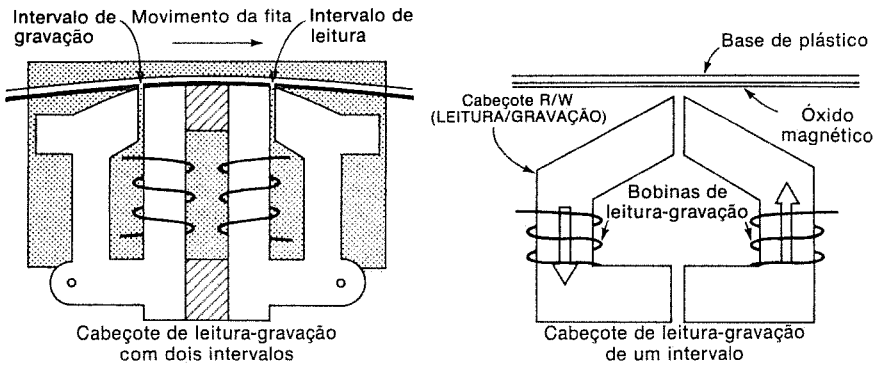


Fig. 7.41 Cabeçotes de fita com um e dois entreferros (intervalos).

nada manualmente no mecanismo). Os estojos de fita também proporcionam proteção contra sujeira e contaminação, já que a fita é lacrada no estojo.

A Fig. 7.44 mostra um cassete típico. Vários cassetes digitais diferentes estão agora em produção e cada um tem características diferentes.† Por exemplo, a Data General Corporation oferece um cassete com uma velocidade média de fita de 31 pol./s, uma fita magnética de 200 pés e 0,15 pol. por cassete, armazenamento por fita de 800.000 bits e velocidade de transferência (leitura) de 12.800 bits/s. O cassete pode reenrolar-se em 85 s. Um líder refletor de 22 pol. e um reboque são usados para marcar o início e o fim da fita (um fotodiodo lê esta tira).

Os estojos ou cartuchos constituem um meio de armazenamento em fita magnética de alto desempenho. Há vários projetos de estojos disponíveis. Estes projetos variam não somente nas capacidades de desempenho, mas também na divisão de componentes físicos *(hardware)* entre o estojo e o mecanismo de transmissão.* O estojo e mecanismo de acionamento da 3M mostrado na Fig. 7.45 é representativo. O estojo contém 300 pés de fita de

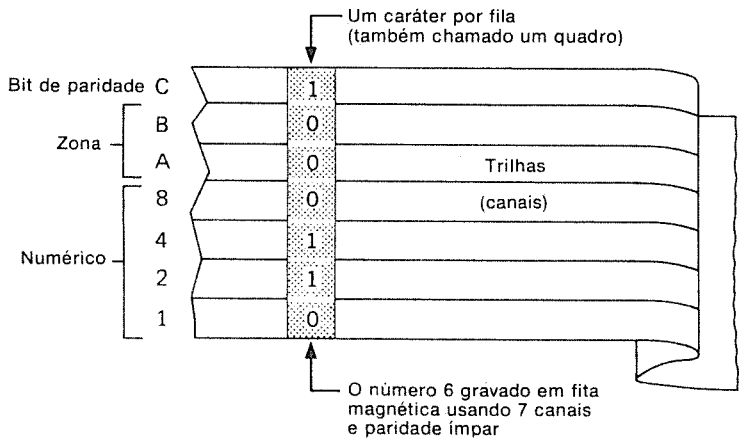
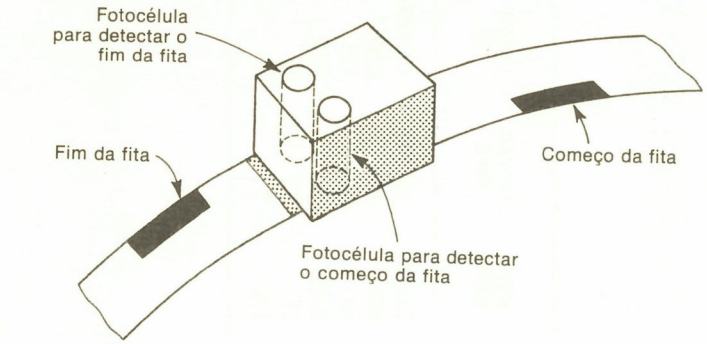
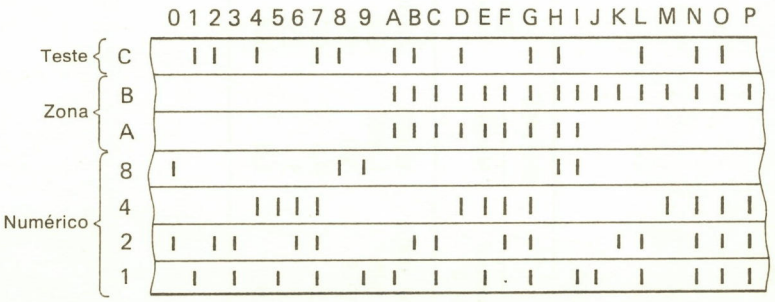


Fig. 7.42 Traçado básico de fita magnética.

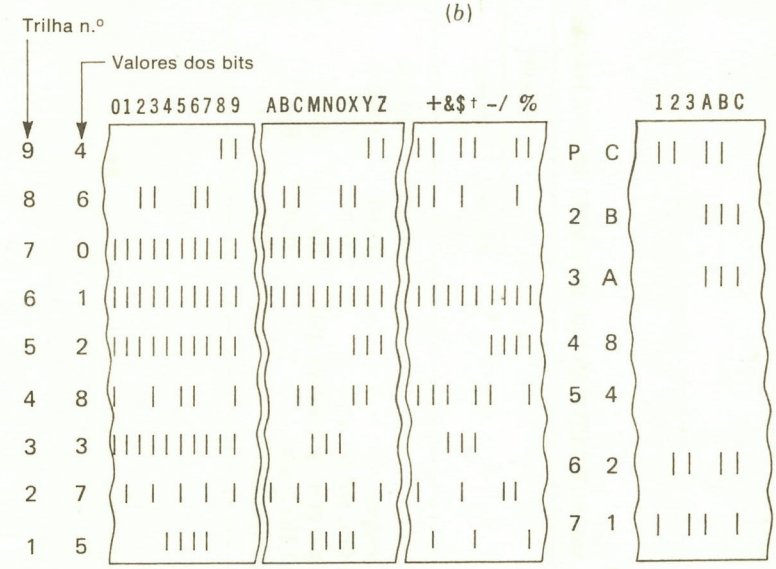
†Organizações padronizadas têm tentado desenvolver padrões de cassetes. O cassete "Phillips" é um destes padrões.
*Por exemplo, os cabeçotes podem ou não ser incluídos em cada estojo.



(*a*)



(*b*)



†A posição de bit P produz paridade ímpar.

(*c*)

Fig. 7.43 Codificação da fita magnética. *(a)* Começo e fim da marcação da fita; *(b)* gravação magnética do código BCD de sete trilhas na fita; *(c)* comparação do formato de dados na fita de sete trilhas e na fita de nove trilhas (EBCDIC).

QUADRO 7.11 CARACTERÍSTICAS DAS UNIDADES DE FITA MAGNÉTICA

Característica	IBM 2401			IBM 2420	BGH	BGH	UNIVAC	UNIVAC
	Modelo 4	Modelo 5	Modelo 6	Modelo 7	424	9392	VIII C	16
Bytes/s	60.000	120.000	180.000	15.000	66.400	72.000	96.000	192.000
Densidade, bytes/pol.	1.600	1.600	1.000	1.600	800	800	800	1.600
Velocidade da fita, pol./s	37,5	75,0	112,5	200	83	90	125	125
Intervalo nominal entre gravações, pol.	0,6	0,6	0,6	0,6	0,75	0,75	0,75	0,75
Tempo nominal IRG, ms	16,0	8,0	5,3	5	25	25	12	12
Tempo de reenrolar, min	3,0	1,4	1,0	1,0	1,5	1,6	3	2

QUADRO 7.12 SISTEMAS DE FITA 7090E DA HEWLETT-PACKARD

Característica	Especificação
Número de trilha	9
Velocidade de leitura-gravação nos sistemas	
Com base no 2100	25, 37,5, 45 pol./s
Sistema 3000	45 pol./s
Densidade	1600 caracteres/pol. (8 bits/caráter)
Velocidade de transferência de dados	72.000 caracteres/s máximos
Diâmetro do carretel	$10^1/_2$ pol. máximos
Fita (grau de computador)	
Largura	0,5 pol.
Espessura	1,5 milésimos de polegada
Velocidade de reenrolamento	160 pol./s
Tempos de início-parada	8,33 ms (LER-APÓS-GRAVAR) em 45 pol./s
Fim da fita e começo da detecção da tira refletora da fita	IBM compatível

$^1/_4$ pol. capaz de gravar até quatro trilhas em 1600 bits/pol. para uma capacidade máxima de armazenamento de mais do que 2×10^7 bits. O mecanismo de transmissão da 3M opera em 30 pol./s quando lendo ou gravando, e a 90 pol./s em modo de busca. Um novo dispositivo de fita elástica movimenta a fita e também supre tensão de fita. Unidade de fita, cilindro e componentes de guia são citados na base do estojo e não requerem orientação externa.

O Quadro 7.13 fornece algumas especificações do estojo 3M e da unidade de estojo 3M. O Quadro 7.14 compara alguns dos dispositivos de memória que foram descritos dando alguma idéia do potencial destes dispositivos. Observemos que estas são estimativas da potencialidade, e a maior parte dos dispositivos dos dias de hoje fica abaixo destes limites. O Quadro 7.15 fornece alguns dados sobre dispositivos de baixos custos adequados aos minicomputadores e microcomputadores em oposição aos dispositivos mais caros. Estes são novamente dados representativos.

*7.24 TÉCNICAS DE GRAVAÇÃO DIGITAL

Embora as características e construção de dispositivos de armazenamento tais como tambores magnéticos, gravadores de fita e dispositivos de armazenamento em disco magnético possam variar enormemente, o processo fundamental de armazenamento em cada um deles consiste em armazenar um binário 0 ou 1 numa pequena área de material magnético. O armazenamento em cada caso é dinâmico, pois o meio no qual a informação é gravada é deslocado junto ao dispositivo de leitura ou de gravação.

Embora o processo de gravação de um 0 ou de um 1 numa superfície possa parecer direto, uma pesquisa considerável foi realizada tanto no desenvolvimento dos padrões gravados usados para representar os 0 e os 1 quanto nos meios de determinação do valor gravado. Aqui há duas necessidades: (1) a densidade de integração deve ser feita tão grande quanto possível; assim, cada célula ou bit deve ocupar o mínimo de espaço possível, economizando portanto, por exemplo, a quantidade de fita usada para armazenar uma certa informação; e (2) o processo da leitura e da gravação devem ser feitos tão confiáveis quanto possível. Estes dois interesses são conflitantes porque, quando os bits gravados são cada vez mais condensados rigorosamente juntos, a distorção do sinal reproduzido aumenta enormemente.

Na gravação de informação numa superfície magnética, a informação digital é fornecida ao conjunto de circuitos de gravação, que então codifica esta informação num padrão que é gravado pelo cabeçote de gravação. As técnicas usadas para gravar informação num

*As seções sobre técnicas de gravação podem ser omitidas sem perda de continuidade.

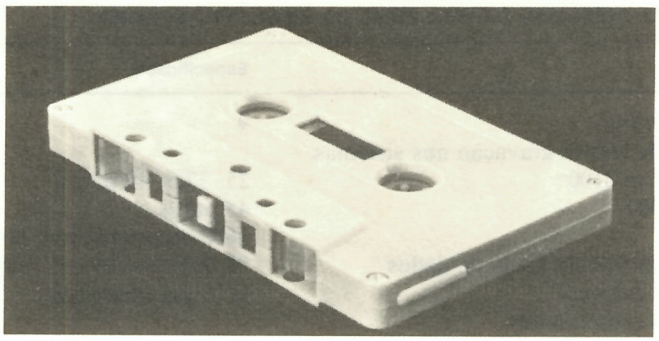


Fig. 7.44 Cassete de fita magnética.

meio magnético podem ser divididas em várias categorias, a saber: a técnica *com retorno a zero* (RZ), a técnica *com retorno à magnetização fundamental* (RB) e a técnica de *não-retorno a zero* (NRZ). Os métodos de leitura da informação gravada com emprego destas técnicas também variam. As técnicas básicas serão descritas abaixo, junto com as formas-de-onda gravadas e com as formas-de-onda lidas mais tarde pelos cabeçotes de LEITURA e traduzidas pelo sistema de leitura.

*7.25 TÉCNICAS DE GRAVAÇÃO DE RETORNO A ZERO E DE RETORNO À MAGNETIZAÇÃO FUNDAMENTAL

A Fig. 7.46 ilustra a técnica de gravação de retorno ao zero. Na Fig. 7.46*(a)* não há corrente no enrolamento do cabeçote de GRAVAÇÃO, exceto quando um 1 ou um 0 deve ser gravado. Se um 1 deve ser gravado, um pulso de polaridade positiva é aplicado ao enrolamento no cabeçote de GRAVAÇÃO e, se um 0 deve ser gravado, um pulso negativo é aplicado ao enrolamento. Em qualquer caso, a corrente através do enrolamento do cabeçote de GRAVAÇÃO é retornado a zero após o pulso, e permanece ali até que o próximo bit seja gravado. O segundo conjunto de formas-de-onda neste desenho ilustra o padrão de fluxo re-

QUADRO 7.13 ESPECIFICAÇÕES DO ESTOJO 3M E DA UNIDADE DE ESTOJO DCD-3

Característica	Especificação
Velocidade de operação	
Ler-gravar	30 pol./s direto e inverso
Direto rápido, reenrolar, busca de intervalo	90 pol./s direto e inverso
Densidade de informação	1600 bits/pol.
Velocidade de transferência	48 quilobits/s máxima
Intervalo entre gravações	1,33 pol. típico; 1,2 pol. mínimo pelo padrão proposto da ANSI
Velocidade início-parada máxima recomendada	Três operações por segundo sem resfriamento forçado de ar
Variação total da velocidade	±4% máx.
Cabeçote de fita	Cabeçotes disponíveis de LER-ENQUANTO-GRAVA de 1, 2 ou 4 canais
Lógica da interface	TTL compatível
Alimentação	5 V cc ±5%, ±18 V cc ±5%

QUADRO 7.14 *COMPARAÇÃO DOS MEIOS DE ARMAZENAMENTO

Característica do sistema	Cassete tipo Phillips	Estojos tipo 3M	Discos flexíveis
Custo por bit, *cents*	10^{-4}	$0{,}5 \times 10^{-4}$	$2{,}5 \times 10^{-4}$
Capacidade	3×10^{6} bits máxima	2×10^{7} bits sobre quatro trilhas	$0{,}3 \times 10^{6}$ bits típicos
Tempos de acesso	Dezenas de segundos em média	20 s em média	Centenas de milissegundos, mas depende da velocidade rotacional
Velocidades de transferência de dados	25×10^{3} bits/s	50×10^{3} bits/s	Centenas de quilobits por segundo

*São dados os valores típicos.

manente sobre a superfície magnética após o cabeçote de gravação ter passado. Há alguma distorção neste padrão devido ao franjamento (oscilações) do fluxo ao redor do cabeçote.

Se este padrão de magnetização for passado sob um cabeçote de LEITURA, parte do fluxo magnético será acoplado ao núcleo do cabeçote. O fluxo segue o caminho de relutância mais baixa através do material do núcleo do cabeçote em vez de transpor o entreferro no cabeçote (Fig. 7.33), e, quando a quantidade de fluxo através do material do núcleo varia, será induzida uma tensão na bobina enrolada ao redor do núcleo. Assim, uma variação na amplitude do campo magnético gravado resultará numa tensão induzida na bobina no cabeçote de LEITURA. As formas-de-onda na Fig. 7.46*(a)* e *(b)* ilustram sinais de saída típicos nos enrolamentos do cabeçote de LEITURA para cada uma das técnicas. Observemos que a forma-de-onda no cabeçote de LEITURA não é uma reprodução da corrente de entrada durante o processo de GRAVAÇÃO, nem do padrão realmente magnetizado sobre o material magnético.

O problema consiste, portanto, em distinguir uma saída 1 ou uma saída 0 no enrolamento de leitura. Diversas técnicas têm sido usadas para isto. Uma consiste em amplificar inicialmente a forma-de-onda de saída do enrolamento de LEITURA num amplificador linear. A saída deste amplificador é então amostrada de maneira idêntica à da saída do enrolamento de leitura de um plano de núcleos. Em sistemas de tambor, a sincronização correta do pulso de amostragem, que deve ser de muita precisão, pode ser determinada pelos sinais de sincronização gravados na trilha de sincronização. Se a saída do amplificador de LEITURA for conectada a uma porta E e o pulso de amostragem também for conectado como uma entrada na mesma porta E, a saída será um pulso positivo quando o sinal gravado representar um 1.

É importante que o pulso de sincronização seja muito estreito ou aguçado e ocorra no

QUADRO 7.15 ALGUMAS CARACTERÍSTICAS DO SISTEMA DE ARMAZENAMENTO DE CUSTOS BAIXOS

Característica do sistema	Capacidade	Custo, *cents*
Disco flexível (*floppy*)	250 quilobits	4×10^{-3}
Cassete de alto desempenho	2 milhões de bits	5×10^{-4}
Cassete Phillips	3 milhões de bits (máximos)	1×10^{-4}
Cassete de baixo desempenho	100 quilobits	20×10^{-4}
Estojo 3M	20 milhões de bits	$0{,}5 \times 10^{-4}$
Carretéis de fita de 7 pol.	30 milhões de bits	$0{,}5 \times 10^{-4}$

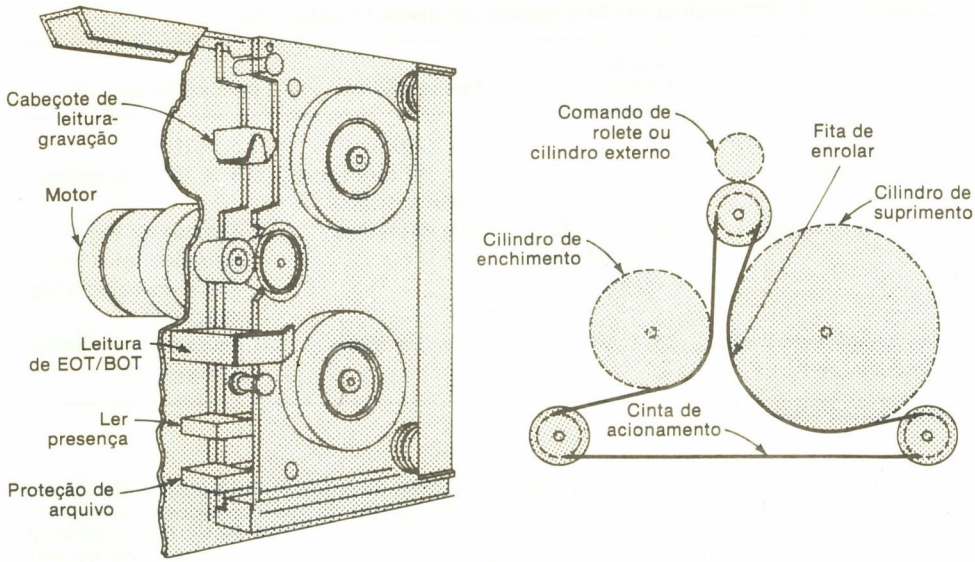


Fig. 7.45 Estojo digital e interface. (Cortesia da 3M Co.)

tempo certo em relação à leitura e gravação dos bits.

Uma característica fundamental da gravação de retorno a zero [Fig. 7.46*(a)*] é que, para um 1, o sinal de saída durante a primeira metade de cada tempo de bit será positivo com relação à segunda metade; e que, para um 0, a primeira metade do sinal de saída durante cada tempo de bit será positiva com relação à segunda metade do sinal. Isto é, às vezes, explorado na tradução do sinal lido novamente.

No sistema de retorno a zero na Fig. 7.46*(a)*, o campo magnético retorna ao fluxo zero quando um pulso 1 ou 0 não está presente. Isso torna impossível gravar sobre informação que tenha sido previamente gravada, a menos que a posição de cada célula esteja localizada com muita precisão. Se um pulso 0 for gravado diretamente sobre um 1 gravado anteriormente, o fluxo gerado inverterá a polaridade do campo gravado somente se o cabeçote de gravação estiver exatamente na posição certa quando o 0 for gravado. A sincronização da gravação de informação é, portanto, muito crítica neste sistema e ele raramente é usado exceto com tambores magnéticos, onde a sincronização pode ser estabelecida com precisão pelas trilhas de sincronização. Uma técnica alternativa envolve o apagamento de todo o fluxo antes da gravação de nova informação, mas isto requer um cabeçote adicional de apagamento e é raramente usado.

O segundo método de gravação de informação é o sistema de retorno à magnetização fundamental, ilustrado na Fig. 7.46*(b)*. Neste caso, a corrente através do enrolamento mantém o cabeçote saturado no sentido negativo, a não ser que um 1 deva ser gravado. Quando um 1 é gravado, um pulso de corrente no sentido oposto é aplicado ao enrolamento no centro do tempo de bit. As saídas no enrolamento de leitura também estão ilustradas na figura. Neste caso, haverá uma saída no enrolamento de leitura somente quando um 1 for gravado. Esta saída pode ser amplificada e amostrada tal como no caso anterior. A sincronização aqui não é tão crítica quando a informação está sendo "gravada sobre", porque o fluxo negativo do cabeçote magnetizará a superfície no sentido correto independente do que foi gravado anteriormente. A corrente através do enrolamento neste caso, e em todos aqueles que se seguem, é suposta ser suficiente para saturar o material sobre o qual os sinais devem ser gravados. Um problema primordial aqui é o que concerne às seqüências de 0. Em fita magnética, ou uma trilha de relógio deve ser usada ou o código usado deve ser tal que ocorra pelo menos um 1 em cada linha da fita. Observemos que isto é porque somente os 1 geram variações de fluxo magnético e, portanto, sinais de saída no cabeçote de leitura.

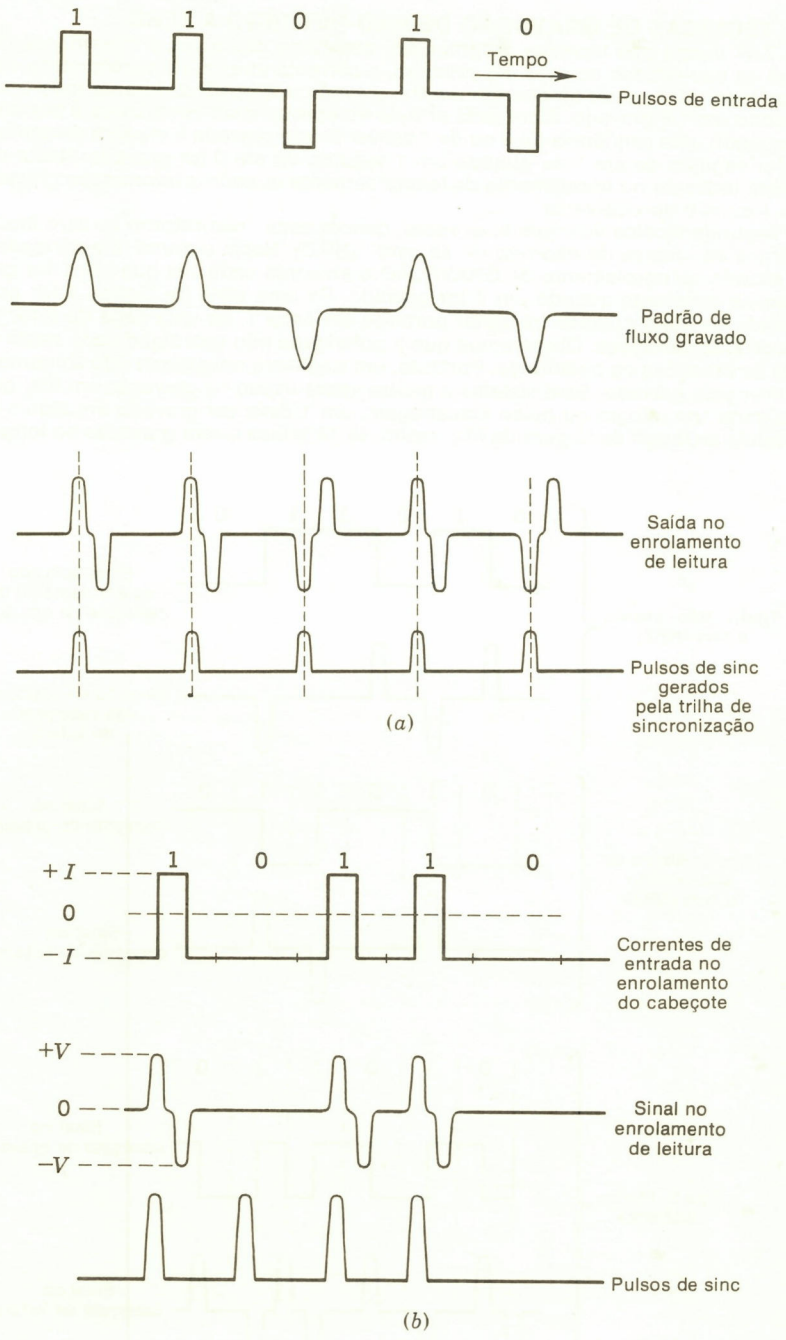


Fig. 7.46 Técnicas de gravação. *(a)* Gravação pelo retorno a zero (RZ); *(b)* técnica de retorno à magnetização fundamental (RB).

*7.26 TÉCNICAS DE GRAVAÇÃO DE NÃO-RETORNO A ZERO

A Fig. 7.47 ilustra três técnicas de gravação, cada uma das quais é classificada como um sistema de não-retorno ao zero. Na primeira, a corrente através do enrolamento é negativa durante todo o tempo de bit quando um 0 é gravado, e é positiva durante todo o tempo de bit quando um 1 é gravado. A corrente através do enrolamento permanecerá portanto constante quando uma seqüência de 0 ou de 1 estiver sendo gravada e mudará somente quando um 0 for seguido de um 1 ou quando um 1 seguido de um 0 for gravado. Neste caso, um sinal será induzido no enrolamento de leitura somente quando a informação gravada variar de um 1 a um 0 ou vice-versa.

A segunda técnica ilustrada é, às vezes, denominada "não-retorno ao zero modificada" ou técnica de "marca de não-retorno ao zero" (NRZI). Neste sistema, a polaridade da corrente através do enrolamento de GRAVAÇÃO é invertida cada vez que um 1 for gravado e permanece constante quando um 0 for gravado. Se uma série de dígitos 1 for gravada, a polaridade do fluxo gravado irá variar portanto em cada 1. Se uma série de 0 for gravada, não ocorrerão variações. Observemos que a polaridade não tem significado neste sistema; apenas as variações na polaridade. Portanto, um sinal será novamente lido somente quando um 1 tiver sido gravado. Esse sistema é muitas vezes usado na gravação em fita, quando, a fim de gerar um relógio ou pulso amostragem, um 1 deve ser gravado em algum lugar de cada célula ao longo da largura da fita. Assim, se 10 trilhas forem gravadas ao longo da fita,

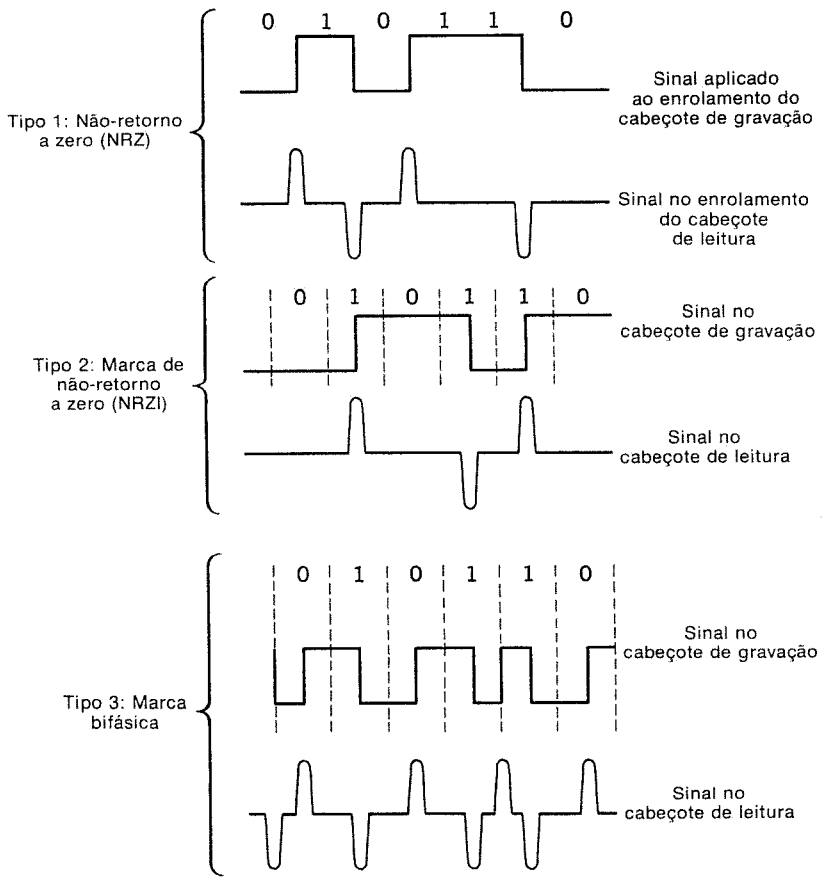


Fig. 7.47 Três tipos de gravação pelo não-retorno a zero.

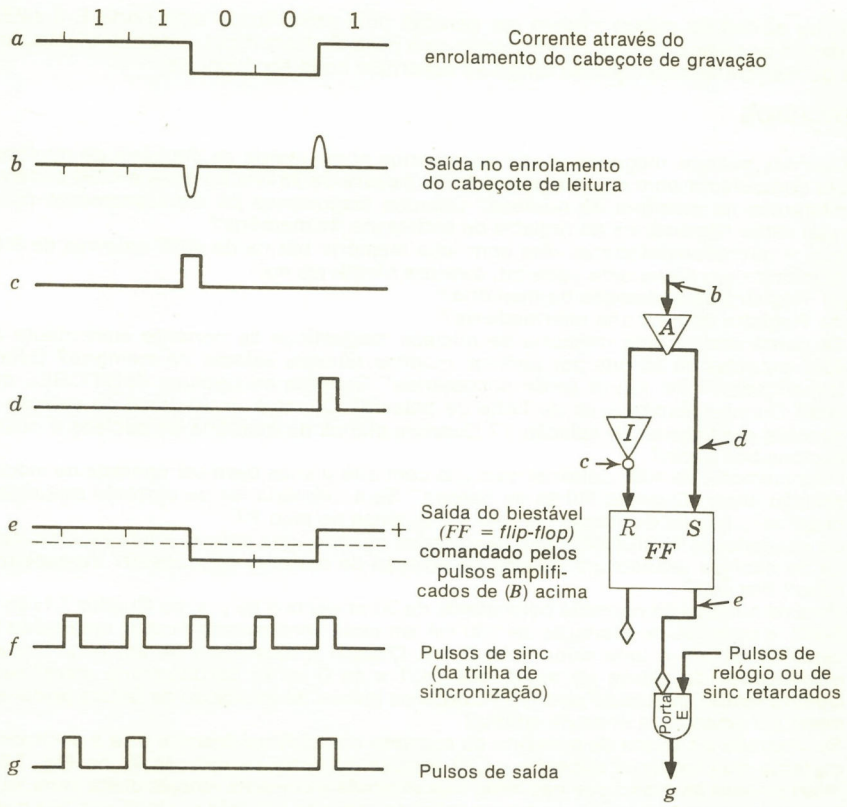


Fig. 7.48 Gravação pelo não-retorno a zero.

uma destas deverá ser uma trilha de sincronização que grava uma seqüência de dígitos 1, cada um dos quais define um conjunto diferente de células a serem lidas, ou a informação deverá ser codificada de modo que ocorra um 1 em cada conjunto de 10 células que são lidas. A informação† codificada em alfanumérico é freqüentemente gravada em fita, e o código pode ser arranjado de modo que ocorra um 1 em cada grupo de código.

A terceira técnica de não-retorno a zero na Fig. 7.47 é, às vezes, chamada sistema de freqüência dividida, ou sistema Manchester, ou sistema Harvard, ou sistema de marca bifásica, ou sistema de *fase codificada*. Neste caso, um 0 é gravado como um pulso negativo em $^1/_2$ tempo de bit seguido por um pulso positivo em $^1/_2$ tempo de bit, e um 1 é gravado como um pulso positivo em $^1/_2$ tempo de bit seguido de um pulso negativo em $^1/_2$ tempo de bit. Essa técnica é freqüentemente usada nos sistemas de alta velocidade.

A leitura de informação que foi gravada consiste de duas etapas. Primeiro, a saída do cabeçote de LEITURA é amplificada e, depois, os sinais amplificados são traduzidos pelo conjunto de circuitos lógicos. A Fig. 7.48 mostra uma técnica de tradução para o primeiro sistema de não-retorno a zero ilustrado na Fig. 7.47. Os sinais de saída podem ser ou do biestável de saída ou pulsos em série. Os pulsos de sinc ocorrem cada vez que uma célula passa sob os cabeçotes de LEITURA no sistema.

O biestável (Fig. 7.48) responde apenas a pulsos positivos. Sinais de pulsos positivos no cabeçote de gravação "ajustarão" portanto o biestável em 1. O inversor na entrada *C* fará com que os pulsos negativos sejam tornados positivos. Estes pulsos positivos depois limparão (reajustarão) o biestável. A saída do biestável pode ser usada diretamente pelo compu-

†Códigos alfanuméricos serão descritos no Cap. 8.

tador, ou saídas de pulsos podem ser geradas pela conexão de uma porta E à saída 1, retardando os pulsos de sinc e conectando-os à porta E. Além disso, uma representação em série do número armazenado ao longo da superfície pode ser formada.

EXERCÍCIOS

1. Quantos núcleos magnéticos são necessários numa matriz de memória de núcleos de um computador com um comprimento de palavra de 35 bits binários e 4096 endereços diferentes na memória de núcleos? Quantos dispositivos de armazenamento (biestáveis) serão necessários no registro de endereços da memória?
2. Vários microcomputadores vêm com uma memória básica de 4096 palavras de 8 bits. Admitindo um plano para cada bit, quantos biestáveis no:
   (a) Registro de endereços da memória?
   (b) Registro de memória intermediária?
3. Se construirmos uma memória de núcleos magnéticos de corrente coincidente com 4096 palavras de 36 bits por palavra, quantos núcleos estarão na memória? Quantos amplificadores de leitura serão necessários? Quantos excitadores INIBIDORES de linha? Quantos excitadores de linha de palavra? Quantos excitadores de seleção X e quantos excitadores de seleção Y? Quantos planos na memória de núcleos e quantos núcleos por plano?
4. Uma memória de 4096 palavras de 8 bits com oito planos deve ser operada no modo de seleção linear. Quantas linhas de palavra? Se a memória for de corrente coincidente, quantas linhas de excitação no eixo X e quantas no eixo Y?
5. Os excitadores INIBIDORES não são usados quando uma palavra é lida de uma memória de núcleos usando um sistema de seleção de corrente coincidente. Verdadeiro ou falso? Por quê?
6. Para os núcleos de corrente coincidente de 30 milésimos de pol. no Quadro 7.1, do tipo 3-103, a corrente de excitação de 290 mA em cada enrolamento produz uma saída 1 de cerca de 61 mV e uma saída 0 de 8 mV. Discutir a ajustagem de um nível limite no amplificador de leitura de modo que os 1 e os 0 serão corretamente identificados. Quanto tempo depois de serem aplicados os pulsos de excitação de leitura poder-se-ia medir ou amostrar a linha de leitura?
7. Se usarmos o sistema de memória de corrente coincidente descrito e se o excitador de corrente que excita a corrente $-I_m/2$ através da linha de seleção X_2 na Fig. 7.6 for interrompido de modo que não mais seja excitada a corrente através desta linha, nunca seremos capazes de ler ou de gravar satisfatoriamente nos três núcleos, um dos quais é o núcleo X_2Y_1. Quais são as designações dos outros dois núcleos nos quais nunca seremos capazes de ler ou gravar corretamente?
8. Numa memória de seleção linear, o excitador pode ser ajustado para "superexcitar" os núcleos por aplicação de maior corrente do que é necessário para saturar o núcleo. Isto fará com que seja necessário também aumentar a quantidade de corrente inibidora que é aplicada. Por quê?
9. Se o excitador inibidor que é conectado ao plano de núcleos que armazena o primeiro dígito binário de cada palavra de computador ficar defeituoso, qual será o valor do dígito binário gravado no núcleo selecionado neste plano durante cada ciclo da memória?
10. Desenhar o sinal de entrada, o padrão de fluxo gravado e a saída no cabeçote de LEITURA para gravação de retorno a zero e de retorno à magnetização fundamental se os bits 1001 forem gravados.
11. Desenhar as formas-de-onda para gravação da seqüência binária 101, mostrando o sinal aplicado ao enrolamento do cabeçote de GRAVAÇÃO e o sinal no enrolamento do cabeçote de LEITURA, para as técnicas de gravação dos tipos 1, 2 e 3 de não-retorno a zero.
12. Um diagrama-bloco de uma pastilha de memória INTEL MOS LS1 é mostrado na Fig. 7.21. As linhas A_1 até A_8 são para os endereços 256. R/W diz se é para ler ou gravar (um 0 nessa linha é LER, um 1 é GRAVAR). A "seleção de pastilha" desativa a memória para uma entrada 1 e ativa a memória com uma entrada 0. Discutir a construção de uma memória de 256 palavras de 8 bits usando estas embalagens. Quantas embalagens são necessárias? Cada biestável no registro (externo) de endereços teria sua saída 1 conectada a quantas pastilhas?
13. Desenhar a esquemática de uma matriz decodificadora, do tipo de múltiplas funções para uma, com entradas de quatro biestáveis e 16 linhas de saída. Usar a mesma configuração básica como a ilustrada na Fig. 7.10.

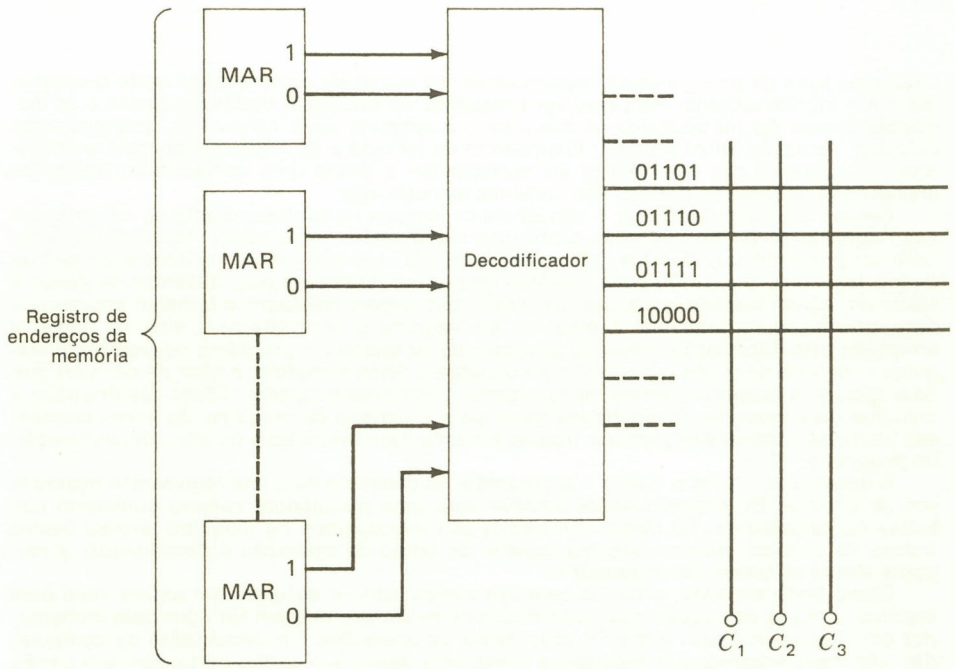
14. A embalagem INTEL no Exercício 12 é representativa de vários produtos de fabricantes de circuitos integrados. Mostrar como a *seleção de pastilha* pode ser usada para acrescentar palavras a uma memória.
15. Como é possível combinar as linhas de leitura e do INIBIDOR numa memória de núcleos para formar uma única linha de bits ou de dígitos? Por que isto é desejável?
16. Responder as seguintes perguntas sobre expansão de memória de núcleos magnéticos de corrente coincidente.
    (a) Aumentando o número de bits por palavra, numa memória de 4096 palavras de 18 bits, de 15 a 20 bits, o número de planos de núcleos aumenta de _____ a _____.
    (b) Numa memória de 4096 palavras de 18 bits, aumentando-se o número de palavras de 4096 para 8192, o número de enrolamentos de excitação *X* e *Y* aumenta de _____ até _____.
    (c) Numa memória de 4096 palavras de 15 bits, aumentando-se o número de palavras para 8192 e o número de bits para 24, o número de enrolamentos INIBIDORES aumenta de _____ para _____.
17. *(a)* Explicar por que as memórias de núcleos mais lentas e menos dispendiosas tendem a usar uma seleção de corrente coincidente, e as memórias de núcleo mais rápidas e mais dispendiosas usam a seleção linear.
    (b) Como o número de dimensões num esquema de seleção da memória de núcleos afeta as dimensões mínimas dos núcleos que possam ser usados?
    (c) As memórias de CI tendem a ser memórias de leitura destrutiva ou não-destrutiva? Por que?
18. *(a)* Discutir os efeitos sobre o tempo de ciclo da memória produzidos pelos dispositivos de leitura destrutiva *versus* dispositivos de leitura não-destrutiva.
    (b) Por que motivo os amplificadores de leitura têm de ser protegidos contra a corrente de excitação inibidora numa memória de núcleos de três fios de corrente coincidente, mas não numa memória de núcleos de quatro fios de corrente coincidente?
19. Discutir o conceito da dimensionalidade nas técnicas de seleção de memória e seus efeitos sobre a velocidade, o custo e a complexidade da memória. (Como exemplos, podem-se contrastar os esquemas de seleção bidimensionais e tridimensionais nas memórias de CI ou comparar a seleção linear e a seleção de coincidência de corrente nas memórias de núcleos.)
20. Qual é o propósito da linha INIBIDORA numa memória de núcleos?
21. Dar uma vantagem da memória de núcleos sobre a memória de semicondutores.
22. Quais são as principais vantagens da memória bipolar sobre a memória de MOS?
23. Explicar a operação da linha de dígitos numa memória de núcleos de seleção linear.
24. Explicar a diferença entre memória MOS estática e memória MOS dinâmica.
25. Num decodificador de 32 saídas, comparar a árvore paralela com as redes decodificadoras balanceadas. Assinalar o número de diodos usado em cada uma e o retardo existente devido ao número de portas que um sinal deve atravessar em cada um, admitindo que sejam usadas portas E com diodos.
26. Se houver uma falha na energia, a maior parte das memórias de núcleos tem seus circuitos cortados, e assim o computador fica parado após a gravação de seus registros na memória de núcleos, e esta fica então parada antes que a energia caia a um valor tão baixo que ocorram erros. Os conteúdos da memória de núcleos não são depois perdidos e a máquina pode ser reiniciada. Isso é possível com memórias de CI?
27. As baterias são, muitas vezes, usadas como fonte de energia de reserva para as memórias de CI a fim de "atenuar" as falhas de energia. Discutir isto apresentando razões que justifiquem.
28. Nas máquinas de grande porte, quando a energia de ca (corrente alternada) cai abaixo de um certo nível, os conteúdos da unidade de controle e da unidade aritmética lógica são despejados em fita ou em discos de modo que o computador possa ser reiniciado sem perda de dados. Se as memórias de CI forem usadas, seus conteúdos também devem ser despejados. Explicar por quê.
29. Explicar a operação da célula de memória na Fig. 7.19.
30. Explicar a operação da célula de memória na Fig. 7.22.
31. Quando for usada seleção linear em memórias de CI, as células individuais tendem a ser mais simples do que se fossem sistemas de seleção de células bidimensionais, mas os decodificadores tendem a ser mais complicados. Explicar por quê.
32. Explicar por que NMOS está substituindo PMOS no uso em memórias de CI.
33. A memória nas Figs. 7.23 e 7.24 tem células de memória de seleção linear; divide-se então a memória numa outra dimensão, usando várias linhas de bits ou de dígitos e seus conjuntos de circuitos associados. Isto simplifica a complexidade individual da

célula de memória enquanto que aumenta o conjunto de circuitos de leitura e GRAVAÇÃO. Explicar por que isto pode constituir custo efetivo para uma memória deste tipo.

34. As memórias dinâmicas que requerem restauração ou revigoramento externo introduzem complexidade extra na operação do computador. Podem-se apresentar as razões pelas quais isto acontece?
35. Explicar as vantagens e as desvantagens das memórias de CI dinâmicas.
36. Um tambor magnético tem uma circunferência de 50 pol. e uma densidade de integração de 1.000 bits/pol. Se o tambor tiver 40 trilhas, quantos bits podem ser armazenados na superfície do tambor?
37. No projeto de uma memória de 256 palavras de 8 bits, o pino 1 é conectado ao pino 1 de cada receptáculo da pastilha Intel no Exercício 12. Isto se aplica a todas as entradas *A* e aos bits de SELEÇÃO DE PASTILHAS e à LEITURA-GRAVAÇÃO, mas não à SAÍDA DE DADOS ou à ENTRADA DE DADOS. Por quê?
38. Um tambor magnético tem 128 trilhas (excluindo-se as trilhas de sincronização) e 1920 células ou bits binários gravados ao redor de cada trilha. Se o tambor for usado num computador em série com um comprimento de palavra de 29 bits binários, e 1 "bit espaçador" for deixado entre as palavras quando gravando, quantos bits serão necessários para endereçar uma palavra no tambor? Quantos bits binários serão necessários para endereçar uma trilha? Quantos bits binários serão necessários para endereçar um setor?
39. Mostrar como expandir a memória de 256 palavras de 1 bit no Exercício 12 numa memória de 1024 palavras de 1 bit, usando a SELEÇÃO DE PASTILHA e um decodificador de duas entradas/quatro saídas.
40. Na memória de discos IBM 3330 descrita, quantos dígitos binários podem ser gravados numa superfície de um determinado disco?
41. Se um tambor magnético girar a 25.000 rpm, qual é o máximo tempo de acesso necessário para ler uma palavra num sistema paralelo, admitindo que não haja retardo na decodificação do endereço? Qual é o tempo médio de acesso aleatório?
42. Qual das unidades de fita IBM transferirá dados mais rapidamente e com que rapidez podem ser lidos ou gravados os caracteres e bits?
43. Admitindo que um disco no sistema IBM 2314 gire a uma velocidade de 2.400 rpm como estabelecido e que lemos simultaneamente oito trilhas, quantos bits por segundo podem ser lidos de um destes discos?
44. Instituir uma fórmula para calcular quantos bits por segundo podem ser lidos numa memória de disco que gira numa velocidade de r revoluções por segundo e tem b bits por trilha. Qual será o tempo médio de acesso?
45. Um sistema de fita magnética tem sete trilhas para cada $^1/_2$ pol. largura de fita. A densidade de integração por trilha é 250 bits/pol. e a fita é movida a uma velocidade de 75 pol./s. Se a largura da fita for de 1 pol., quantos bits podem ser lidos por segundo?
46. Explicar tempo de busca e latência para memórias de disco.
47. Memórias de disco de cabeçote fixo reduzem o tempo total de acesso evitando ou o tempo de busca ou a latência. Qual deles é evitado e por quê?
48. Qual é o perigo em unidades de disco quando são usados cabeçotes volantes?
49. Comparar três das unidades de disco descritas no Quadro 7.5, dando suas idéias sobre suas conveniências. É possível o preço constituir um fator importante na avaliação destes sistemas?
50. Explicar os números apresentados como retardos no sistema de discos HP 2888A descrito no Quadro 7.6.
51. Comparar os números do disco flexível *(floppy)* do sistema Pertec com o sistema de unidade de discos HP 2888A.
52. No Exercício 51, o preço constitui um fator na avaliação dos sistemas?
53. Comparar o sistema de fita HP 7090E com um dos sistemas de fita da IBM.
54. Os cassetes de fita e os estojos de fita têm vantagens e desvantagens em comparação aos sistemas de fitas convencionais. É possível enumerar várias de cada?
55. A tabela que compara os meios de armazenamento destaca alguns dos contrastes em preço e desempenho dos sistemas de memória. Dizer quais dispositivos devem ser escolhidos para as seguintes espécies de computadores e justificar.
    (a) Microcomputadores
    (b) Minicomputadores
    (c) Sistemas de computadores de grande porte
56. Desenhar uma ROM com diodos, como na Fig. 7.31, mas que converta do código BCD 8421 para o código 2421.
57. Se lermos oito cabeçotes ou gravarmos oito cabeçotes simultaneamente na memória

de discos magnéticos descrita e que é fabricada pela Data Products Corporation, quantos bits por segundo poderemos transferir naquele sistema?

58. Desenhar uma ROM com diodos como na Fig. 7.31 que traduz de BCD ao código excesso de 3.
59. Para uma memória de corrente coincidente, os excitadores de linha de palavra são ligados duas vezes e, para a memória de filme delgado, os excitadores de linha de palavra são ligados apenas uma vez. Discutir a razão disto. Os excitadores de linha de palavra com filme delgado necessitam de corrente de excitação nos dois sentidos? Os excitadores de linha de bit necessitam?
60. Designar locais numa ROM com diodos num computador microprogramado, acrescentando diodos ao desenho abaixo. O sinal C_1 de controle deve ser um 1 nos locais 01110 e 01111 e é do contrário um 0; o sinal C_2 de controle deve ser um 1 nos locais 01101 e 10000 e é 0 de outro modo; o sinal C_3 de controle deve ser um 1 em 01101, 01110 e 01111 e é 0 de outro modo. Uma linha de saída de decodificador determinada é alta quando selecionada.



61. Eis aqui alguns dados sobre a unidade de discos PDP-11.

Número de cilindros, 203
Trilhas por cilindro, 2
Bytes por trilha, 6144
Tempo de rotação do disco, 40 ms
Tempo de busca S para mover N cilindros
Para $N = 0$ a 8, $S = 6 + 2N$
Para $N = 9$ a $N = 24$, $S = 16 + 3N/4$
Para N maior do que 24, $S = 26 + N/3$

Discutir o tempo total de pesquisa para achar uma amostra específica de dados, considerando que os cabeçotes estejam posicionados no cilindro central quando for dada a ordem de pesquisa. Um cilindro é o conjunto de todas as trilhas que podem ser lidas ou gravadas numa certa posição do mecanismo de posicionamento do cabeçote.

8

DISPOSITIVOS DE ENTRADA-SAÍDA

Os dispositivos de entrada-saída proporcionam os meios de comunicação entre o computador e o mundo externo. Para resolver problemas ou processar dados, os dados e as instruções devem ser introduzidos na máquina, e a máquina deve fornecer os resultados dos cálculos. Tem sido difícil produzir dispositivos de entrada e de saída que possam acompanhar o progresso das velocidades do computador, e existe uma demanda constante de impressoras, leitoras de cartões etc. cada vez mais rápidas.

Em sua mais simples forma, a seqüência de eventos no processamento de informações nos negócios ou na resolução de problemas científicos é como segue: Inicialmente é escrito um programa que descreve a seqüência de cálculos que a máquina deve executar; os dados, tais como o custo de uma tonelada de aço em Pittsburgh ou a distância de Vênus a Marte em algum instante particular de tempo, são depois coletados e também preparados para introdução na máquina. A máquina em seguida lê o programa e mais os dados e armazena esta informação. Havendo armazenado os dados e o programa necessário, a máquina é iniciada na primeira instrução no programa. Após completar a série de cálculos que deve efetuar, a máquina imprime os resultados. O programa contém ordens que orientam a máquina para imprimir os resultados finais para o usuário da máquina. Às vezes também são impressos resultados parciais, freqüentemente para finalidades de teste ou verificação do programa.

É necessário primeiro inserir o programa e os dados na máquina através dos dispositivos de entrada. Os dispositivos de entrada para uma calculadora comum consistem das teclas da calculadora. Os dados numéricos são introduzidos na máquina através destes botões de pressão; quando alguma espécie de botão de operação é pressionado, a máquina efetua as operações necessárias.

Como pode ser visto, a fim de usar um computador, é desejável ter algum meio para registrar tanto as instruções quanto os dados de modo que possam ser lidos pelo computador em altas velocidades e depois executadas as operações em velocidades de computador. Um meio proposto por Babbage e ainda hoje usado é o cartão perfurado. Um cartão retangular é perfurado por vários orifícios, e a posição de cada orifício determina seu significado. Os programas e dados são perfurados nos cartões. Os cartões são depois colocados numa leitora de cartões conectada à máquina, e é emitido o comando para a leitura. A leitora de cartões "sente" os orifícios nos cartões e transmite esta informação ao computador, que então a armazena. Desta maneira, tanto o programa quanto os dados podem ser lidos na máquina.

Quando o programa tiver sido lido e as operações necessárias tiverem sido concluídas, é necessário que o computador imprima os dados de modo que eles possam ser usados. Atualmente é encontrada uma grande variedade de dispositivos de saída. As aplicações comerciais geralmente requerem que os resultados sejam impressos em forma tabular ou talvez numa série de cheques como numa operação de contabilidade de folha de pagamento. Os resultados científicos consistem mais provavelmente de dados numéricos que devem ser impressos claramente e com pouca probabilidade de erros (e tal precisão é também de primordial importância nos computadores usados para calcular cheques de folhas de pagamento) ou de gráficos que mostram os resultados dos cálculos. Para cada uma destas aplicações, um tipo de dispositivo de saída pode ser mais desejável do que um

outro. No entanto, a grande maioria das aplicações requer que as saídas da máquina sejam impressas num pedaço de papel. Os principais dispositivos de saída são portanto as impressoras, que variam desde as máquinas de escrever eletromecânicas que imprimem uma letra ou dígito de cada vez até as impressoras de alta velocidade capazes de imprimir centenas de caracteres de cada vez. Serão descritas neste capítulo várias dessas, bem como dispositivos de saída mais especializados.

8.1 DIFERENÇAS ENTRE AS APLICAÇÕES COMERCIAIS E CIENTÍFICAS

Há nítidas diferenças na razão de dados para o comprimento do programa nos problemas comerciais e científicos. Os problemas comerciais geralmente têm programas menores que processam grandes quantidades de dados. Por exemplo, os programas que calculam folhas de pagamento ou os programas que mantêm um controle de inventário estão sujeitos a serem de comprimento relativamente pequeno em comparação com o volume de dados que eles devem processar. Por outro lado, os problemas científicos são mais sujeitos a terem longos e complicados programas que efetuam muitos cálculos mas que utilizam relativamente poucos dados de entrada.

Os computadores usados em problemas comerciais devem ser capazes não somente de manipular grandes quantidades de dados de entrada, mas também de fornecer muita informação, geralmente sob a forma de impressos, ao usuário. Uma máquina que calcula a folha de pagamento de uma grande empresa ou companhia deve imprimir milhares de cheques. O mesmo fato se aplica às máquinas que fazem cálculos para os bancos ou outros grandes estabelecimentos comerciais em que grandes números de informes comerciais devem ser impressos. Por esta razão, há um incentivo particularmente grande sobre velocidade e economia nos dispositivos de entrada-saída para as máquinas comerciais.

8.2 FITA PERFURADA

A fita perfurada foi um dos primeiros meios populares para armazenamento de programas e dados que devem ser lidos numa máquina digital. Quando foram projetados os primeiros

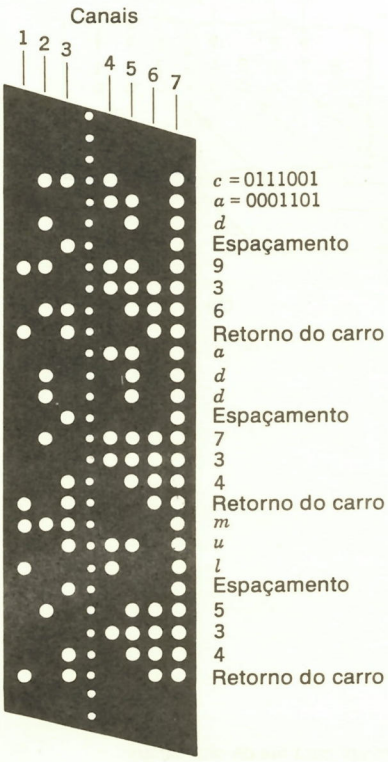


Fig. 8.1 Fita de papel perfurada.

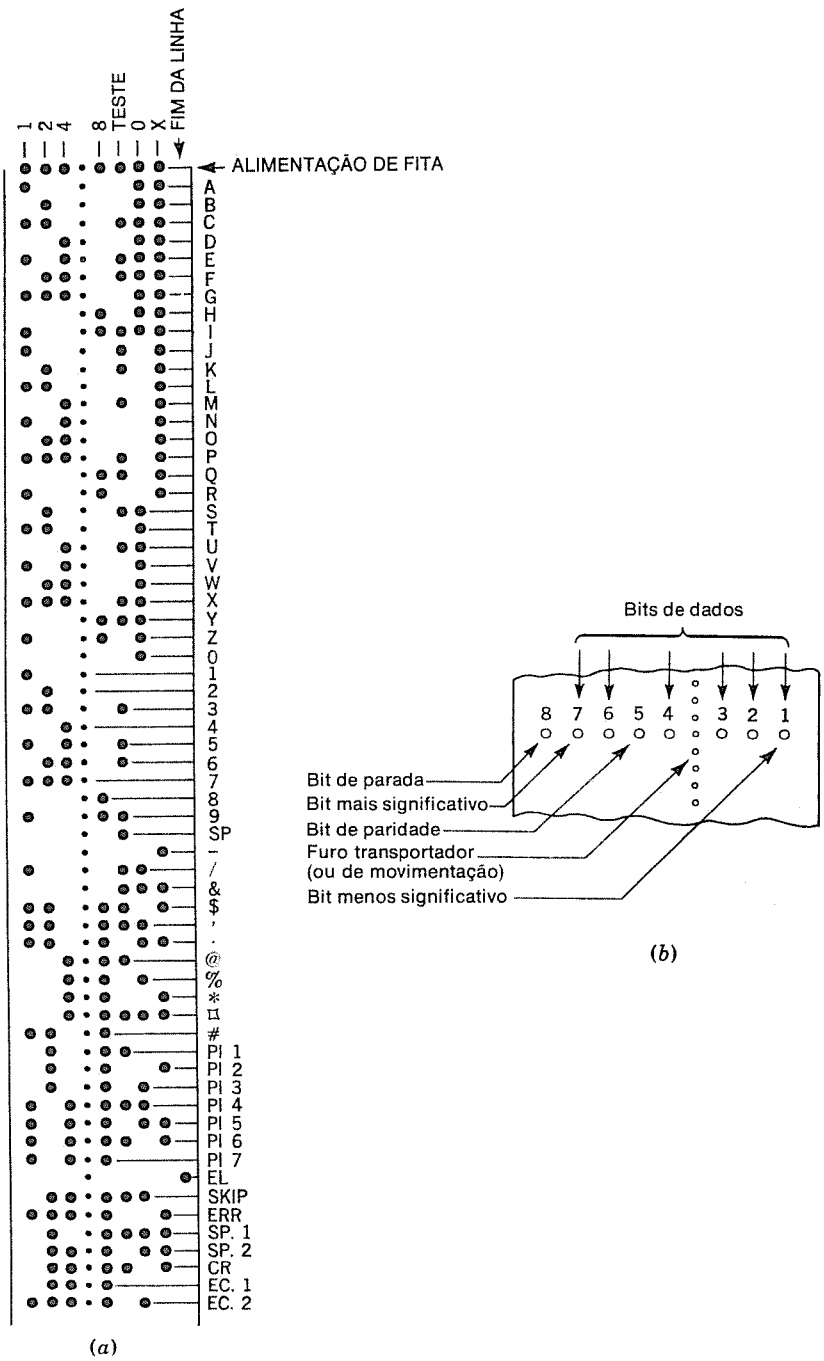


Fig. 8.2 (*a*) Fita perfurada com código de oito canais; (*b*) formato para fita de oito canais.

computadores de grande porte, os sistemas telegráficos tinham usado fitas de papel perfuradas durante algum tempo, e, em conseqüência, os dispositivos de perfuração e leitura de fitas de papel já haviam sido perfeitamente bem desenvolvidos. A fita usada é de muitos tipos e dimensões. Uma fita de papel de espessura média tem sido muito usada, e fitas oleadas e fitas plásticas são também usadas. As larguras das fitas usadas tinham variado de 1/2 a 3 pol. A informação é perfurada na fita à razão de uma linha de cada vez. A Fig. 8.1 ilustra uma seção de uma fita perfurada. São usados múltiplos canais (tal como numa fita magnética, um canal corre longitudinalmente ao longo da fita), e um único caráter ou código é perfurado como um padrão de bits em cada linha lateral.

A preparação destas fitas de papel de entrada é, às vezes, denominada *registra em teclado,* e uma típica máquina de perfuração de fita de papel é mostrada na Fig. 1.7. Nesta etapa, ao operador de uma máquina de perfuração de fita é apresentada uma cópia de um programa ou dados de entrada. O operador então perfura, por meio da máquina de perfuração de fita (perfuradora de fita), vários orifícios na fita. Os orifícios representam, em forma codificada, a informação de entrada para a máquina. O dispositivo de perfuração de fita usado pode ser de diversos tipos; contudo, os dispositivos mais populares se assemelham a uma máquina de escrever, e os teclados destas perfuradoras de fita contêm símbolos convencionais, similares aos de uma máquina de escrever comum. Os teclados de muitas máquinas de perfuração de fita são idênticos aos teclados das máquinas de escrever manuais usadas nas atividades comerciais, e algumas máquinas de escrever elétricas são convertidas em máquinas de perfuração de fitas adaptando-se um dispositivo de perfuração que é acionado pelo mecanismo da máquina de escrever.

Quando uma tecla no teclado da perfuradora de fitas é comprimida para baixo, o símbolo codificado em binário do caráter selecionado é perfurado na fita, e a fita então avança para a linha seguinte. Na maioria dos casos, o dispositivo perfurador de fita imprime também num pedaço de papel separado, tal como uma máquina de escrever, o caráter que foi perfurado, imprimindo igualmente os caracteres ao longo das filas da fita. Há, em seguida, uma cópia datilografada do programa que pode ser verificada quanto a erros em acréscimo ao tipo de papel perfurado com os símbolos codificados. Esta cópia impressa do programa é denominada "cópia de leitura direta". Muitas das máquinas de perfuração de fita são capazes de ler uma fita perfurada e datilografar cópia impressa desta fita. Uma seção perfurada de fita pode ser colocada na leitora de fita incorporada à máquina perfuradora de fita, e pode ser feita uma cópia datilografada da informação que foi perfurada na fita.

As Figs. 8.2 e 8.3 mostram dois códigos que foram freqüentemente usados em sistemas de fita de papel. A Fig. 8.2 mostra um código de oito canais no qual há oito canais passando

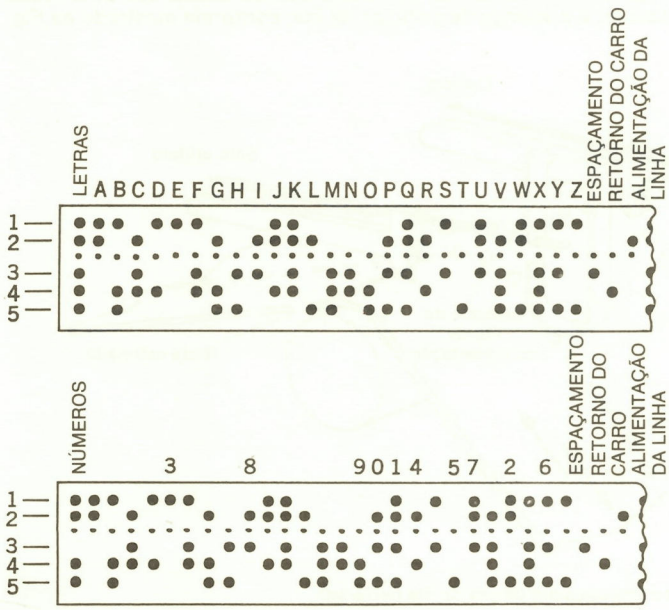


Fig. 8.3 Fita de papel com código de cinco canais.

longitudinalmente ao longo da fita. Um orifício num destes canais representa um 1, e a ausência de um orifício representa um 0. Os canais 1, 2, 4 e 8 são usados para representar os dígitos de 0 a 9; portanto, os 0 ou ausências de orifícios nas posições *EL*, *X* e 0 indicam que o caráter codificado é um dígito com o valor determinado pela soma das posições 8, 4, 2, 1 em que há orifícios. A posição de teste é usada num *teste de paridade ímpar*. Seu valor é determinado de modo que o número de 1 ou de orifícios em cada caráter seja ímpar. As posições 0 e *X* são usadas em conjunção com as posições 8, 4, 2, 1 para codificar os caracteres alfabéticos e especiais.

O código tem duas características notáveis. Uma perfuração no extremo-da-linha indica o fim de um registro na fita. Quando houver perfurações em todas as sete posições, isso indica um caráter branco chamado ALIMENTAÇÃO DE FITA (TAPE FEED), e a leitora de fita salta sobre tais posições. Isto é útil na correção de erros na preparação ou na edição de fitas, porque um caráter errado pode ser eliminado pela simples perfuração nas posições não-perfuradas.

O código de cinco canais mostrado na Fig. 8.3, o chamado código Baudot, foi muito popular nos sistemas de comunicações. Já que cinco posições são insuficientes para codificar o número essencial de caracteres, um certo artifício é usado com emprego de dois caracteres especiais para mudar o significado daqueles caracteres que se seguem. Assim, um caráter LETTERS seguido de uma linha com perfurações na posição 1 indica um *E*, enquanto que um caráter FIGURES seguido de uma perfuração na posição 1 indica um 3. Sempre que ocorrer um caráter LETTERS, todos os caracteres seguidos estão no código LETTERS até que ocorra um caráter FIGURES, e, similarmente, um caráter FIGURES significa que todos caracteres seguintes estão naquele modo até que ocorra um caráter LETTERS.

8.3 LEITORAS DE FITA

A função da leitora de fita de papel é ler (perceber) a informação codificada perfurada na fita e fornecer esta informação ao computador. As leitoras de fita usadas em equipamentos de teletipo e de escritório são, em sua maioria, dispositivos eletromecânicos. Em muitos destes dispositivos mecânicos, "pinos sensores" são usados para determinar o símbolo perfurado em cada linha da fita. Num sistema deste tipo, haverá um pino sensor para cada canal de informação, além de um meio de movimentar a fita e posicioná-la para leitura. A fita não é movida continuamente, mas apenas uma única linha de cada vez, parada enquanto a codificação estiver sendo lida, e depois movida para a linha seguinte. O movimento dos pinos sensores faz operar uma chave, cujos contatos são abertos ou fechados, dependendo de haver ou não um orifício na fita. Um outro tipo de leitora usa uma "roda estrelada" para ler a ausência ou a presença de orifícios na fita, conforme mostrado na Fig.

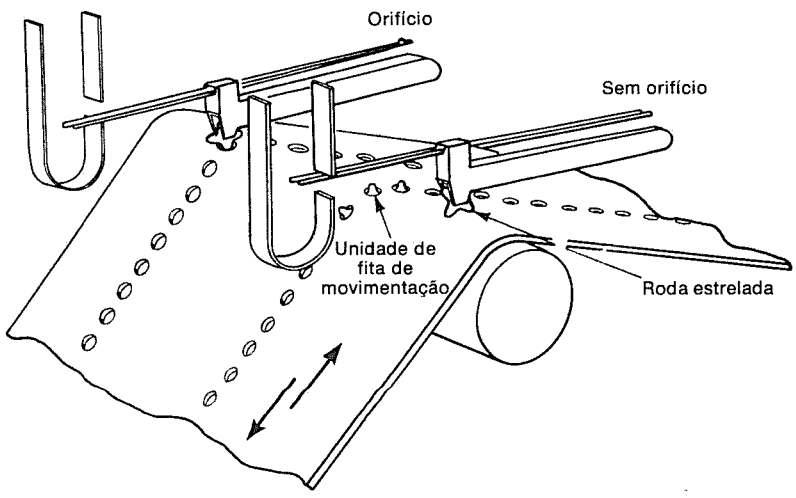


Fig. 8.4 Mecanismo de rodas estrelada para leitura de fita perfurada.

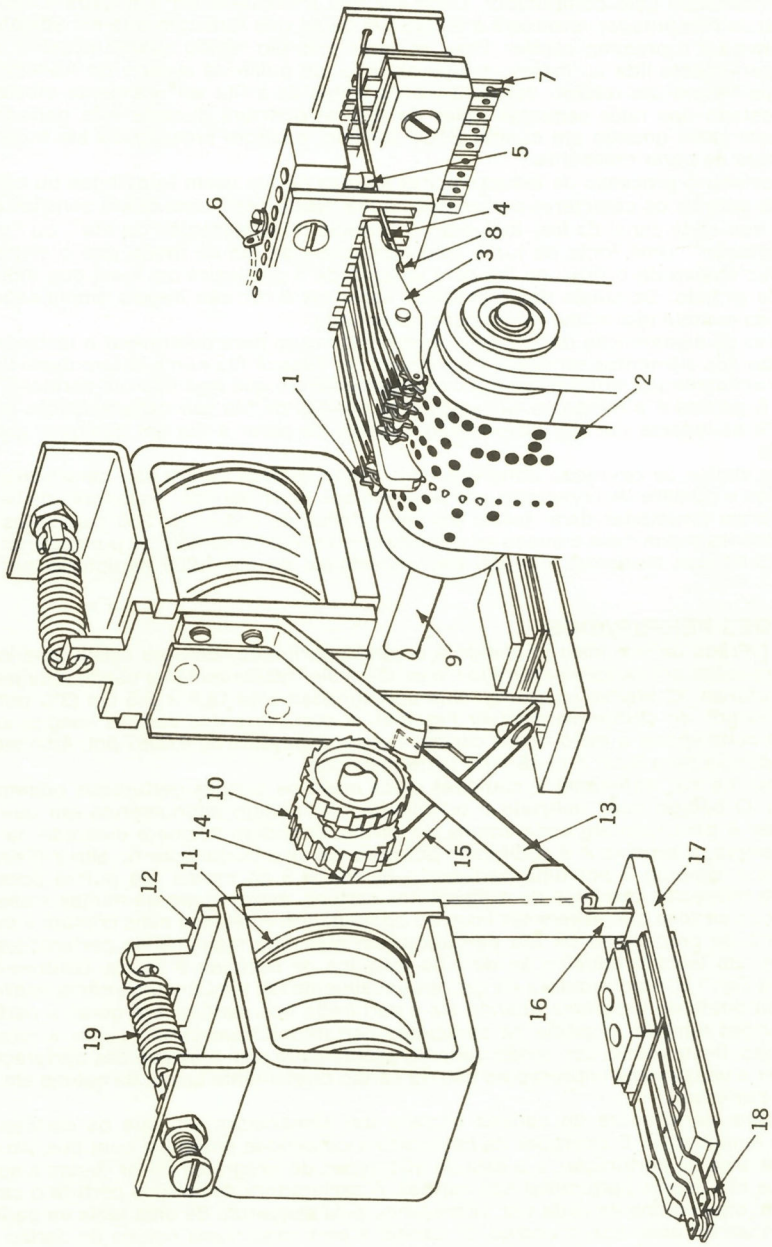


Fig. 8.5 Mecanismo de leitora de fita eletromecânica.

8.4. Um mecanismo completo de leitora é mostrado na Fig. 8.5, indicando a complexidade relativa destes dispositivos.

Quando a entrada for em máquina digital, o movimento da fita através da leitora será, em geral, controlado pelo computador. Cada vez que a fita deve ser avançada e lido um novo caráter, o computador fornecerá à leitora um pulso que fará com que ela adiante ou avance a fita para o próximo caráter. Para ler caracteres tão rápido quanto possível, uma linha será geralmente lida ao mesmo tempo em que um pulso de avanço for transmitido. Uma vez que haverá um retardo devido à inércia antes de a fita ser realmente movida, a leitura do estado dos relés sensores (relés de leitura) ocorrerá durante este período de retardo. Neste caso, quando um caráter STOP for lido, a leitora prosseguirá até o caráter seguinte antes de parar realmente.

Para acelerar o processo da leitura, muitas leitoras de fita usam fotodiodos ou células fotoelétricas para ler os caracteres perfurados na fita. Neste caso, uma célula sensível à luz é colocada sob cada canal da fita, incluindo o "orifício de alimentação da fita", ou "canal de movimentação". Uma fonte de luz é colocada acima da fita de modo que o elemento sensível à luz abaixo do orifício na fita será energizado e produzirá um sinal que indica a presença do orifício. Os sinais dos elementos sensíveis à luz são depois amplificados e fornecidos ao computador como informação de entrada.

O orifício de alimentação da fita será usado neste caso para determinar o instante em que as saídas dos elementos sensíveis à luz devem ser lidas. A fita numa leitora deste tipo é geralmente acionada por atrito e movida continuamente até que seja lido um caráter STOP (PARADA). A partida e a frenagem extremamente rápidas da fita são características muito desejáveis, e as leitoras em sua maioria são capazes de parar a fita em qualquer caráter determinado.

As velocidades de operação atingíveis com várias leitoras de fita são geralmente expressas como o número de caracteres por segundo que pode ser lido. As leitoras de leitura mecânica foram projetadas para operar em velocidades da ordem de 250 caracteres por segundo, embora sejam mais comuns as velocidades de 10 a 60 caracteres por segundo. As leitoras fotoelétricas modernas operam em velocidades de até 1.000 caracteres por segundo.

8.4 CARTÕES PERFURADOS

O meio de entrada de uso mais difundido é o cartão perfurado. Embora exista uma variedade de dimensões em cartões perfurados, o cartão mais freqüentemente usado atualmente é o de 80 colunas, 12 fileiras ou alturas com aproximadamente 18,5 × 8,5 cm ($3^1/_4$ pol. de largura e $7^3/_4$ pol. de comprimento) (ver Fig. 8.6). A espessura dos cartões usados varia, embora, em certa época, a maioria dos cartões tivesse espessura de 0,0067 pol. Atualmente há uma tendência para fazer o cartão um tanto mais fino.

Tal como na fita, há inúmeras maneiras pelas quais os cartões perfurados podem ser codificados. O código mais utilizado é o Hollerith, um código alfanumérico em que um único caráter é perfurado em cada coluna do cartão. O código básico é ilustrado na Fig. 8.6. Por exemplo, o símbolo *A* é codificado por meio de uma perfuração no alto e fileiras 1 do cartão, e o símbolo 8 por uma perfuração na fileira 8 do cartão. Há outros tipos de cartões com diferentes posições de orifícios nos cartões, assim como há muitas maneiras de preparar os cartões que devem ser lidos no computador. A técnica mais comum é muito semelhante à da preparação da fita perfurada, em que uma máquina de perfuradora de cartões com um teclado idêntico ao de uma máquina de escrever é usada, conforme foi mostrado na Fig. 1.8. A perfuradora de cartões usualmente faz também uma cópia impressa ou de leitura direta do programa quando ele é perfurado nos cartões. Em geral, a perfuradora de cartões também imprime os caracteres perfurados num cartão sobre a face do próprio cartão. Deste modo, um cartão pode ser identificado sem o exame das perfurações. Cada caráter é usualmente impresso no alto do cartão diretamente acima da coluna em que o caráter é perfurado.

A máquina perfuradora de cartões contém um alimentador em que os cartões em branco são empilhados. O operador da perfuradora de cartões então faz com que um cartão entre na área de perfuração, e a lista de instruções do programa ou os dados a serem processados são depois perfurados nos cartões. A perfuradora de cartões perfura o cartão lateralmente, uma coluna de cada vez, começando pela esquerda. Se uma tecla da perfuradora de cartões for abaixada, o código do caráter é perfurado numa coluna do cartão, e o cartão é então movimentado de modo que a coluna seguinte à direita fique sob a perfuradora.

Geralmente, quando um programa é perfurado em cartões, apenas uma instrução e seu endereço (ou endereços) associado é perfurado em cada cartão. Então, se for cometido um

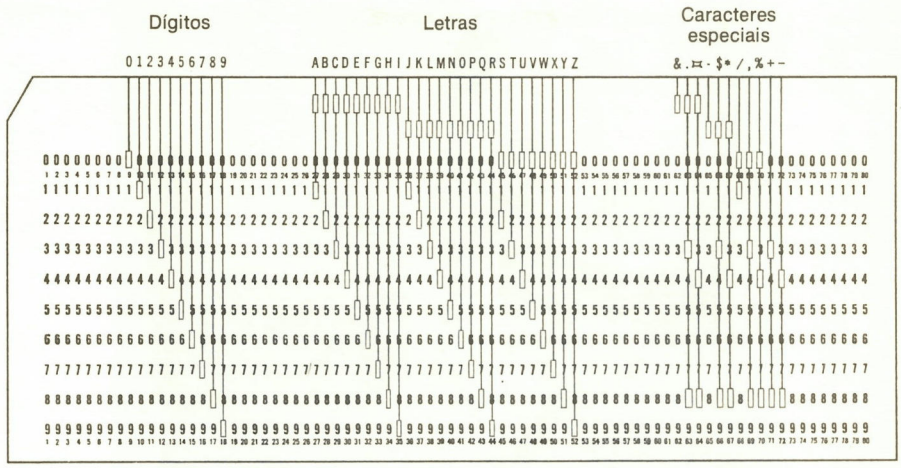


Fig. 8.6 Cartão perfurado com código Hollerith.

erro na programação, a instrução errada pode ser alterada afastando-se o cartão incorreto e substituindo-o por um cartão correto.

A Fig. 8.7 mostra uma perfuradora de cartões com uma pequena memória de circuito integrado (CI) capaz de reter todos os dados que podem ser perfurados nos dois registros de 80 colunas e seis cartões de programas. O operador da perfuradora acionada por teclado registra em teclado os dados na pequena memória e pode dar retrocesso e mudar caracteres até que os dados registrados em teclado estejam corretos. Após todos os dados estarem na pequena memória, é abaixada uma tecla de "introduzir dados", e o cartão é em seguida perfurado a partir dos caracteres na memória. Dispõe-se de instalações ou facilidades para controlar o formato dos cartões por meio de lógica de CI.

A preparação de cartões contendo dados para sistemas comerciais é uma área altamente desenvolvida. As grandes empresas comerciais, tais como as companhias de seguro, as agências de crédito e os bancos, devem reunir e processar quantidades quase incríveis de dados. A maior parte deste processamento de dados é atualmente feita pelos computadores eletrônicos, mas, até antes do advento do computador digital, os cartões perfurados foram extensivamente usados, e foi desenvolvida uma considerável tecnologia para a manipulação de cartões perfurados.

Em qualquer caso, os dados — informações, verificação de contabilidade, taxas de imposto de renda, ou quaisquer que sejam — devem primeiro ser introduzidos num cartão perfurado por um operador de perfuradora acionada por teclas. Para dar alguma idéia da grandeza desta operação, há cerca de 500.000 painéis ou controles de perfuração por teclas (conforme indicam as Figs. 1.8 e 8.7) no momento.

A preparação de cartões perfurados e algumas das operações que podem ser executadas nos cartões são facilitadas pelos vários dispositivos especiais, conforme indicado na Fig. 8.8. Por exemplo, uma *verificadora* é usada para testar a perfuração por tecla dos cartões. Depois de um cartão ter sido perfurado, já que os caracteres alfanuméricos são impressos acima das perfurações por tecla, podem ser feitos alguns testes comparando-se os cartões com os dados de entrada. Um procedimento comum, no entanto, consiste em colocar o condensado de cartões perfurados por um operador no alimentador de uma verificadora e ter um segundo operador para registrar em teclado os mesmos dados. A verificadora não perfura novos cartões, mas compara o que o segundo operador perfura com o condensado preparado pelo primeiro operador e, se ocorrer uma discrepância, ela notifica o segundo operador de modo que o cartão possa ser comparado com os dados.

Outras máquinas mostradas na Fig. 8.8 são a *classificadora*, que é usada para classificar os cartões em diferentes seqüências de acordo com certas chaves nos dados, e a *intercaladora,* que compara ou intercala vários condensados num só condensado de acordo com algumas intruções introduzidas na máquina.

Há diversas outras máquinas de processamento de cartões, tais como perfuradoras-



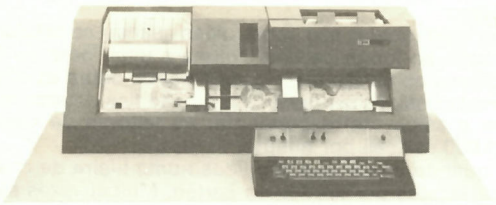
Fig. 8.7 Perfuradoras de cartões modelo IBM 129 em operação.

resumo, que combinam os dados de vários cartões nuns poucos cartões, e as reprodutoras, que duplicam os condensados. As operações realizadas por estas máquinas, no entanto, podem ser executadas por máquinas eletrônicas, que, na maioria dos casos, custam menos para operar e executam as operações necessárias com mais rapidez e eficiência.

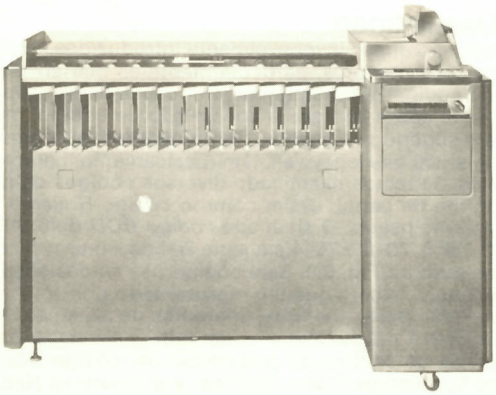
A perfuradora de cartões ainda é o dispositivo mais freqüentemente usado para introduzir dados num computador, e o cartão relativamente barato ($^1/_{10}$ de *cent* por cartão, talvez) ainda é de uso difundido.

8.5 LEITORAS DE CARTÕES

As leitoras de cartões em sua maioria são dispositivos eletromecânicos que lêem a informação perfurada num cartão, convertendo a presença ou ausência de um orifício num sinal elétrico que representa um binário 0 ou 1. Os cartões perfurados são colocados num alimentador, e, quando é emitido o comando de ler, uma alavanca empurra o cartão da base da pilha. Em geral, o cartão é depois movido longitudinalmente sobre uma fileira de 80 "escovas de leitura". Estas escovas lêem a informação perfurada ao longo da fileira da parte inferior do cartão. Se um orifício for perfurado numa determinada fileira, uma escova faz o contato elétrico através do orifício no cartão, fornecendo um sinal que pode ser usado pelo computador. A fileira seguinte acima é lida a seguir, e este processo continua até que todas as fileiras tenham sido lidas após o que o próximo cartão é movido para dentro da



(*a*)



(*b*)



(*c*)

Fig. 8.8 Máquinas de processamento de cartão perfurado. (*a*) Verificadora; (*b*) classificadora; (*c*) intercaladora.

posição sobre as escovas.

Leitoras de cartões mais rápidas são construídas usando-se células fotoelétricas sob as 12 posições de perfuração ao longo de uma coluna e uma fonte de iluminação acima do cartão. Quando cada coluna do cartão é obrigada a passar sobre as 12 células fotoelétricas, o fato de uma determinada posição ser perfurada ou não fica determinado pela presença ou ausência de luz sobre a célula correspondente. As leitoras de cartões operam em velocidades desde 12 a 1.000 cartões por minuto.

Observemos que, quando os cartões são lidos numa fileira de cada vez, os programas devem "decifrar" os caracteres perfurados no cartão. Muitas vezes são usados condensados "binários", particularmente nos programas em que as palavras de computador são distribuídas ao longo das fileiras em vez de nas colunas, e os dados podem ser carregados diretamente quando as fileiras são lidas.

8.6 CÓDIGOS ALFANUMÉRICOS

Os dados e os programas são quase invariavelmente introduzidos em forma alfanumérica, e a operação interna dos computadores, particularmente as que envolvem registros comerciais, faz uso extenso de códigos alfanuméricos. Por causa da diversidade das aplicações e dos muitos pontos de vista sobre códigos e construção de código, muitos códigos alfanuméricos diferentes foram sugeridos e usados. Computadores tais como 1401, 1410 e outros, da IBM, bem como a série 200 da Honeywell, têm (basicamente) caracteres de 6 bits com os códigos correspondentes. Já temos examinado diversos códigos de fita magnética e códigos de 5 e de 8 bits de fita de papel, assim como o código Hollerith de 12 bits. Cada um desses tem sido amplamente usado. O chamado *código BCD* de 6 bits, usado nos computadores IBM 1401, 1410, 7010, 7040 e 7044 e a série 200 da Honeywell, bem como em muitas outras máquinas, é mostrado na Fig. 8.9. Este código usa as posições 8, 4, 2, 1 convencionais para os dígitos decimais, exceto que 0 é representado por um 1 nas posições 8 e 2. O "relatório" e "programa" são para dois tipos diferentes de sistemas. Assim, em alguns sistemas, um 1 na posição 8, 2, 1 produz um $\neq$ e, em outros sistemas, produz um $=$.

Foram feitas tentativas no sentido de padronizar um código alfanumérico que satisfizesse os interesses dos fabricantes e dos usuários, e o American National Standards Institute publicou um American Standard Code for Information Interchange (abreviado código ASCII). Este código é agora amplamente usado nas máquinas mais modernas, e diversos dos principais fabricantes estão atualmente utilizando o código a fim de que seus equipamentos possam ser compatíveis com os de outros fabricantes. Este código é mostrado na Fig. 8.10. Observe que os dígitos decimais são representados pelo código 8, 4, 2, 1 normal precedido pelos três dígitos binários 011, de modo que o decimal 1 se torna 0110001, o decimal 2 é 0110010, o decimal 7 é 0110111 etc. Para expandir o código, a letra A é 1000001, B é 1000010 etc. Há vários códigos tais como "fim de mensagem", "quem é você", "salto", "retorno do carro" etc. que são muito úteis nos sistemas de comunicações e na edição de processos de dados em computadores. Em conseqüência, este é provavelmente o código de uso mais freqüente em sistemas de comunicações entre computadores.

Finalmente, a maioria dos computadores da IBM e diversos outros sistemas de computadores usam o código de intercâmbio BCD estendido (o chamado EBCDIC) mostrado na Fig. 8.11.

8.7 PREPARAÇÃO DE DADOS

Uma forma de memória intermediária consiste em gravar primeiro o programa a ser lido numa fita magnética e em seguida na leitura da fita na máquina. Existem vários dispositivos que devem ler a fita perfurada ou os cartões perfurados e transferir a informação perfurada neles para a fita magnética. Este processo ocorre fora do computador central. Já que a fita magnética pode ser lida muito mais rapidamente que os cartões ou fitas de papel perfurados, o tempo necessário para ler informação será reduzido.

O processo da conversão dos cartões perfurados para fita magnética foi derivado por muitos dispositivos de entrada. Diversas companhias oferecem atualmente uma linha de dispositivos de teclado/fita magnética. A Fig. 8.12 mostra um console deste tipo. Em sua mais simples forma, quando o operador abaixa uma tecla, o caráter selecionado aparece num indicador visual e é também introduzido na fita. Quando a fita magnética foi preenchida ou todos os registros foram transcritos, a fita pode ser lida pelo computador a ser utilizado.

Os mais aprimorados sistemas permitem que o operador de teclado datilografe vários caracteres; estes caracteres são exibidos num console onde podem ser lidos e verificados e também são armazenados numa pequena memória. Os caracteres podem ser editados (modificados), e, quando o operador está seguro de que os dados estão corretos, um botão de

CARÁTER Relatório	CARÁTER Programa	CÓDIGO BCD (Armazenamento em núcleos)						
b		C						
.			B	A	8		2	1
□	)	C	B	A	8	4		
[			B	A	8	4		1
<			B	A	8	4	2	
‡		C	B	A	8	4	2	1
&	+	C	B	A				
$		C	B		8		2	1
*			B		8	4		
]		C	B		8	4		1
;		C	B		8	4	2	
Δ			B		8	4	2	1
–			B					
/		C		A				1
,		C		A	8		2	1
%	(			A	8	4		
≂		C		A	8	4		1
\		C		A	8	4	2	
≠				A	8	4	2	1
b				A				
#	=				8		2	1
@		C			8	4		
:					8	4		1
>					8	4	2	
√		C			8	4	2	1
?		C	B	A	8		2	
A			B	A				1
B			B	A			2	
C		C	B	A			2	1
D			B	A		4		
E		C	B	A		4		1
F		C	B	A		4	2	
G			B	A		4	2	1
H			B	A	8			
I		C	B	A	8			1
!			B		8		2	
J		C	B					1
K		C	B				2	
L			B				2	1
M		C	B			4		
N			B			4		1
O			B			4	2	
P		C	B			4	2	1
Q		C	B		8			
R			B		8			1
‡				A	8		2	
S		C		A			2	
T				A			2	1
U		C		A		4		
V				A		4		1
W				A		4	2	
X		C		A		4	2	1
Y		C		A	8			
Z				A	8			1
ø		C			8		2	
1								1
2							2	
3		C					2	1
4						4		
5		C				4		1
6		C				4	2	
7						4	2	1
8					8			
9		C			8			1

Baixa → SEQÜÊNCIA DE CLASSIFICAÇÃO → Alta

Símbolo	Nome
‡	Marca de grupo
‡	Marca de registro
≠	Marca de segmento
≂	Separador de palavra
@	Arroba
#	Sinal de número
&	"E" comercial
+	Mais
*	Asterisco
%	Porcentagem
/	Barra
\	Barra invertida
□	Losango
b	Branco
ƀ	Substituir brancos
(	Parênteses da esquerda
)	Parênteses da direita
[	Colchete da esquerda
]	Colchete da direita
√	Marca de fita
<	Menor que
>	Maior que
=	Igual a
;	Ponto e vírgula
:	Dois pontos
.	Ponto
'	Apóstrofo
–	Menos ou hífen
Δ	Delta

Fig. 8.9 Código BCD.

	000	001	010	011	100	101	110	111
0000	*NULL*	① DC_0	b	0	@	*P*		
0001	*SOM*	DC_1	!	1	*A*	*Q*		
0010	*EOA*	DC_2	"	2	*B*	*R*		
0011	*EOM*	DC_3	#	3	*C*	*S*		
0100	*EOT*	DC_4 (Parada)	$	4	*D*	*T*		
0101	*WRU*	*ERR*	%	5	*E*	*U*		
0110	*RU*	*SYNC*	&	6	*F*	*V*		
0111	*BELL*	*LEM*	'	7	*G*	*W*	Não-designado	
1000	FE_0	S_0	(	8	*H*	*X*		
1001	*HT* / *SK*	S_1	)	9	*I*	*Y*		
1010	*LF*	S_2	*	:	*J*	*Z*		
1011	V_{TAB}	S_3	+	;	*K*	[		
1100	*FF*	S_4	(Vírgula) ,	<	*L*	\		*ACK*
1101	*CR*	S_5	−	=	*M*	]		②
1110	*SO*	S_6	.	>	*N*	↑		*ESC*
1111	*SI*	S_7	/	?	*O*	←		*DEL*

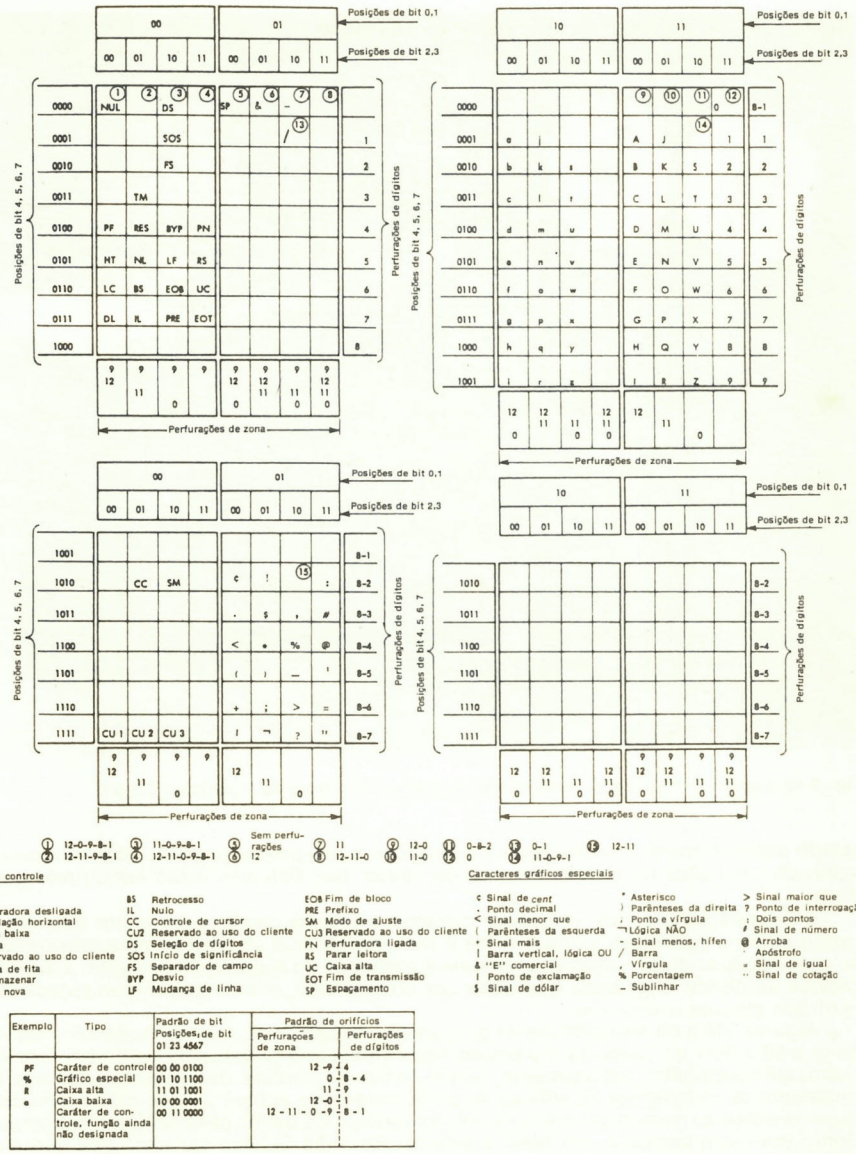
Exemplo: 100 0001 = *A*

b_7 ---------- b_1

As abreviações usadas na figura significam:

NULL	Nulo	*CR*	Retorno do carro
SOM	Início de mensagem	*SO*	Desliga caixa alta (ou letras maiúsculas)
EOA	Fim de endereço	*SI*	Liga caixa alta
EOM	Fim de mensagem	DC_0	Controle de dispositivo Reservado para dados Escape da linha de dados
EOT	Fim de transmissão	DC_1-DC_3	Controle de dispositivo
WRU	"Quem é você?"	*ERR*	Erro
RU	"É você ...?"	*SYNC*	Inutilização síncrona
BELL	Sinal audível de campainha	*LEM*	Fim lógico de suportes
FE	Efetuador de formato	SO_0-SO_7	Separador (informação)
HT	Tabulação horizontal		Separador de palavra (branco, normalmente não impresso)
SK	Salto (cartão perfurado)	*ACK*	Ciente
LF	Mudança de linha	②	Controle não-designado
V/TAB	Tabulação vertical	*ESC*	Escape
FF	Alimentação de formulário	*DEL*	Apaga

Fig. 8.10 American Standard Code for Information Interchange (ASCII). Código de padrões americanos para intercâmbio de informações.



Exemplo	Tipo	Padrão de bit Posições de bit 0 1 2 3 4 5 6 7	Padrão de orifícios Perfurações de zona	Perfurações de dígitos
PF	Caráter de controle	00 00 0100	12 -9	4
%	Gráfico especial	01 10 1100	0	8 - 4
R	Caixa alta	11 01 1001	11	9
a	Caixa baixa	10 00 0001	12 -0	1
	Caráter de controle, função ainda não designada	00 11 0000	12 - 11 - 0 -9	8 - 1

Fig. 8.11 Código de intercâmbio de BCD estendido (EBCDIC).

"registro" é abaixado, os dados são introduzidos em fita magnética e novos dados podem ser registrados em teclado.

Até os mais complexos sistemas usam vários teclados, todos conectados a uma pequena memória e a um processador equipado com um gravador de fita magnética. Isto permite ampla edição; verificação pelo pequeno computador-processador comparando entradas; detecção de erro, particularmente verificação de formatos de muitos tipos; e muitas operações de classificação e intercalação. A fita magnética preparada por um sistema desta espécie tem tido muitas operações de "pré-processamento" já executadas, e os dados têm

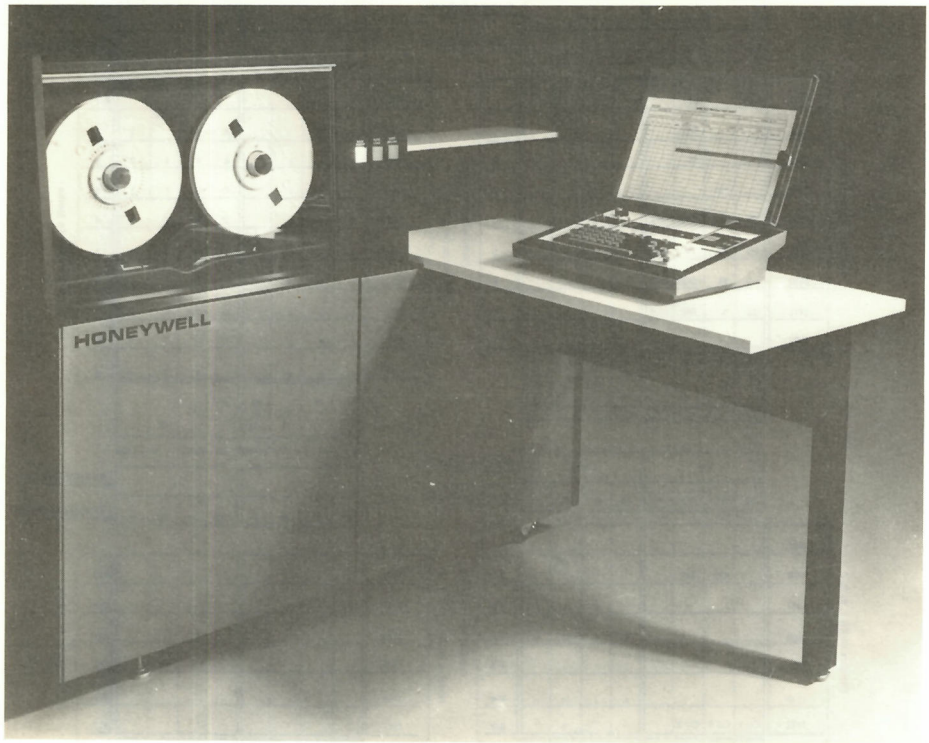


Fig. 8.12 Painel ou console de teclado/fita magnética. (Cortesia da Honeywell Corp.)

estado submetidos a consideráveis verificações e pré-processamento. Nas empresas com extensos requisitos de processamento de dados, tais sistemas estão adquirindo popularidade.

A preparação de fitas, incluindo cassetes e estojos, mostrou-se muito popular, oferecendo alguma economia, porque a fita é mais barata que os cartões para armazenamento de grande capacidade e mais conveniente à operação do sistema. No entanto, a ausência de cartões contendo impressão que pode ser armazenada, examinada e manipulada tem encontrado alguma resistência.

Alguma idéia da importância da preparação de dados de entrada pode ser vista do fato de que 30 a 50% do custo da instalação são atribuídos à preparação de dados. Isto é especialmente verdadeiro, naturalmente, nos sistemas comerciais de grande porte, mas até estritamente as instalações científicas de porte moderado acham que cerca de 20% do custo de seus sistemas podem ser atribuídos à preparação de dados para entrada no computador. Como um outro exemplo, em 1968, o Internal Revenue Service sozinho usou cerca de 400 milhões de cartões perfurados.

Numerosos dispositivos e meios especiais são usados para reduzir o custo para o comércio de entrada de dados. Por exemplo, agências tais como a American Express usam uma combinação de cartões perfurados com caracteres impressos para seus faturamentos. O cliente paga o cartão com seu cheque, e o cartão é depois entregue diretamente ao computador.

Algumas empresas usam cartões de identificação com uma codificação especial para seus empregados. Nestes sistemas, o empregado introduz seu cartão numa leitora especial que lê a identificação e também introduz o tempo de um relógio. Estes dados são registrados em fita ou num disco, e posteriormente o computador lê todos os dados de controle de entrada e de saída registrados durante o dia. Ainda outros sistemas usam um relógio de ponto para perfurar simplesmente os tempos de controle de entrada e de saída num cartão

que já tem perfurada nele a identificação do empregado. Estes cartões são depois coletados ao fim de cada semana. Cada um destes esquemas reduz a despesa de operadores que perfuram os dados. Reprodutoras que lêem os cartões de crédito de clientes e registram os valores das vendas, caixas registradoras que transportam os dados das vendas para um arquivo de fitas central num armazenamento de grande capacidade e muitas técnicas similares estão todas sendo usadas para atenuar os problemas e as despesas de preparação de dados para entrada no computador.

8.8 RECONHECIMENTO DE CARACTERES

As técnicas de entrada de dados se estendem em múltiplas direções. A leitura de caracteres escritos à mão ou datilografados em papel convencional parece oferecer um sistema de entrada ideal em muitas aplicações. Os sistemas normalmente em uso são como segue.

1. *Leitura de caráter em tinta magnética* (MICR) O registro de caracteres que usam uma tinta com propriedades magnéticas especiais e com caracteres que têm formas especiais foi originalmente usado em quantidade pelos bancos. O American Banking Association estabeleceu um sortimento completo de tipos de um tamanho e estilo, e vários de seus caracteres são mostrados na Fig. 8.13. Uma *leitora de caráter magnético* "lê" estes caracteres ao examinar suas formas com emprego de uma matriz de 7 × 10 e determina, a partir da resposta dos segmentos da matriz para a tinta magnética, qual dos caracteres passou sob o cabeçote da leitora. Esta informação é portanto, transmitida ao sistema. A determinação do caráter que é lido é grandemente facilitada pelo projeto cuidadoso dos caracteres e pelo uso da tinta magnética.
2. *Leitura do caráter ótico* Esta área assume uma entre duas formas. Na primeira, um sortimento de tipos (ou fontes) especial é usado para imprimir papel convencional usando tinta convencional. Os caracteres impressos são examinados passando-se os mesmos sob uma luz intensa e um sistema de lentes que diferencia luz (falta de tinta) das áreas com tinta e um sistema lógico que procura determinar qual dos possíveis caracteres no sistema está sendo examinado. Os sistemas em uso real dependem fortemente do fato de que apenas um número limitado de caracteres num sortimento particular de tipos é usado, mas tais sistemas são ainda muito úteis. O sortimento de tipos padronizados proposto pelo comitê de caracteres óticos da ANSI é mostrado na Fig. 8.14.

O sistema ideal seria, naturalmente, capaz de se adaptar a muitos diferentes sortimentos de tipos. Alguns sistemas, particularmente um desenvolvimento pelos Correios, lêem até caracteres escritos a mão. O sucesso limitado destes sistemas é devido às muitas formas que um certo caráter pode assumir; consideremos as maneiras pelas quais se pode escrever um *a* e a similaridade entre um *a* escrito a mão e um *o* ou um *b* e um *f*. Estes problemas são aumentados pela dificuldade da leitora ótica com a porosidade do papel, borrões de tinta nas extremidades das linhas etc. Muito trabalho prossegue nesta área, e ainda é ne-

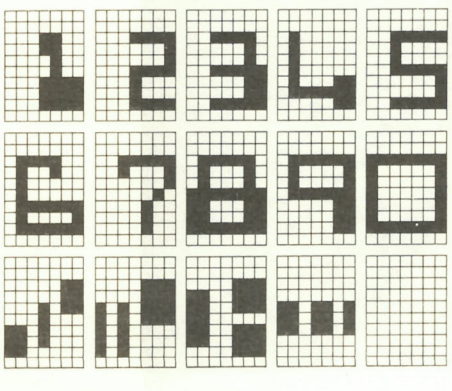


Fig. 8.13 Um conjunto de caracteres de leitora magnética.

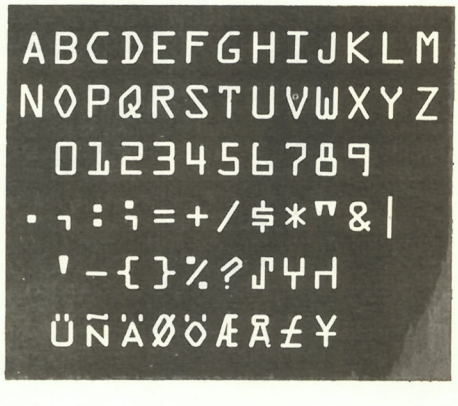


Fig. 8.14 Sortimento de tipos para reconhecimento de caracteres óticos.

cessário muito mais, mas as vantagens de tais sistemas continuam a fazer com que este trabalho seja apoiado e executado.

8.9 EQUIPAMENTO DE SAÍDA

Embora muitos tipos de equipamentos de saída estejam atualmente em uso na indústria de computadores, a forma de saída mais popular de um computador é indubitavelmente a palavra impressa. Outros tipos de dispositivos indicadores em uso comum incluem as lâmpadas a neon e os osciloscópios, e alguns computadores são até equipados com alto-falantes. (São comuns programas que tocarão música através dos alto-falantes, e tentativas são feitas muitas vezes para compor música, usando um computador.)

As lâmpadas são, em geral, usadas para indicar os estados dos dispositivos de armazenamento dos principais registros de máquina (o acumulador, os registros selecionados de entrada-saída etc.). Estas lâmpadas são, às vezes, usadas como dispositivos de saída em programas simples onde as respostas podem então ser lidas visualmente; no entanto, tais lâmpadas são, em geral, usadas como recursos de pesquisa de defeitos, na operação da máquina. A Fig. 8.15 mostra as lâmpadas no console do IBM 370 Modelo 155. Cada lâmpada está ligada a uma saída do biestável (via excitador de lâmpada). As saídas na maioria dos biestáveis importantes são monitoradas com emprego das lâmpadas. Vários conjuntos de biestáveis podem ser selecionados usando as chaves rotativas na parte inferior do painel, e as chaves e os botões de pressão podem ser usados para introduzir dados nos registros de biestáveis e para pôr em seqüência as operações do computador (bem como iniciar e parar



Fig. 8.15 Painel ou console de um sistema de computadores de grande porte, o IBM 370-155. (Cortesia da IBM Corp.)

o computador). As lâmpadas são, às vezes, usadas em programas de pesquisa de defeitos em máquinas pequenas. Se, por exemplo, um programa trabalha por si próprio num laço sem fim, a máquina pode ser parada, e a localização* da seqüência de instruções defeituosa no programa pode ser lida a partir das lâmpadas no console da máquina.

8.10 IMPRESSORAS

Em geral, o método mais conveniente e útil pelo qual o computador pode fornecer informação é por meio de caracteres impressos. Por questões de conveniência, a impressora deve ter capacidade de imprimir caracteres alfabéticos, dígitos decimais e também sinais comuns de pontuação. Deve ser possível imprimir toda a informação processada pelo computador em forma de binário puro ou alguma forma de binário codificado, mas não muitos bancos gostariam de receber um cheque de $1010.011, e a maioria dos cientistas e engenheiros, a despeito de suas possíveis capacidades neste sentido, prefeririam ter um impresso que leia 130 em vez de 10000010.

É óbvio que a impressora deve portanto operar com caracteres codificados em binário em vez de diretamente em binário. O processo aqui é o inverso do procedimento de codificação em que uma tecla correspondente a um caráter alfanumérico é abaixada, fazendo com que um caráter binário codificado seja perfurado numa fita ou cartão. Neste caso, grupos codificados de bits binários são fornecidos à impressora, que os decodifica e imprime então os caracteres corretos. Os grupos básicos de código binário podem conter de 5 a 9 bits, dependendo da codificação dos caracteres alfanuméricos que a impressora fornece.

A informação fornecida a uma impressora que opera em-linha será na forma de sinais eletrônicos diretamente da máquina. Se a impressora for operada fora-de-linha, a leitura e a decodificação de dados armazenados em fita perfurada, cartões perfurados ou em fita magnética podem ser uma parte da operação de impressão. Uma vez que o conjunto de circuitos eletrônicos de um computador é capaz de operar em velocidades muito maiores do que as dos dispositivos de impressão mecânica, é desejável que uma impressora operada em-linha seja capaz de imprimir numa velocidade muito alta. Mesmo se a impressora for operada fora-de-linha, a velocidade é altamente desejável, já que o volume de material a ser impresso pode ser muito grande.

As impressoras originais, em sua maioria, eram máquinas de escrever elétricas convertidas, e este tipo de impressora ainda é comum. Se o código usado contiver 8 bits por caráter e a impressora for operada em-linha, o computador fornecerá 8 bits à impressora, que os decodificará e energizará um solenóide que acionará a tecla correta da máquina de escrever. Os códigos usados incluem geralmente caracteres codificados tais como os discutidos de início e também os códigos para espaçamento entre palavras, retorno do carro e outras operações necessárias à datilografia. A velocidade de tais impressoras do tipo de máquina de escrever é relativamente baixa, talvez de 10 a 30 caracteres por segundo.

Outros tipos de impressoras de um caráter de cada vez utilizam princípios um tanto diferentes. Um tipo tem várias "palhetas", cada uma das quais tem um caráter em relevo sobre a superfície mais próxima ao rolete ou tambor de contato, montado numa cesta de tipos móvel. A cesta de tipos é posicionada eletromecanicamente por um decodificador, e o verso da palheta é então golpeado por um martelo que força a palheta de encontro à página. Uma fita de tinta é posicionada entre a palheta e a página (como uma fita de máquina de escrever); o caráter da palheta é, desse modo, impresso na página, o rolete é depois movido para a esquerda e um outro caráter é impresso. Às vezes, as palhetas estão contidas numa "caixa de tipos" retangular que é também posicionada pela ação do mecanismo de decodificação, e a palheta correta é então golpeada por um martelo. Tanto os mecanismos da máquina de escrever quanto os mecanismos da caixa de tipos e das alavancas de tipos são capazes de imprimir até 40 caracteres por segundo aproximadamente.

A fim de aumentar ainda mais a velocidade da impressora sem onerar significantemente os custos, vários fabricantes construíram *impressoras por pontos* em que um conjunto de pinos numa matriz de 9×7 acionada por solenóides é usado para imprimir caracteres. Um caráter é impresso selecionando os pinos adequados na matriz de 9×7, e os pinos selecionados são então dirigidos contra o papel por pequenos martelos. Velocidades de até 330 caracteres por segundo são atingíveis com estas impressoras.

Se várias alavancas de tipos estiverem localizadas numa fileira ao longo do tambor e forem fornecidos mecanismos de posicionamento e martelos para cada alavanca, pode ser

*O contador do programa contém o endereço da instrução que está sendo executada. Detalhes desta seção do elemento de controle serão apresentados no capítulo seguinte.

construída uma *impressora de uma linha de cada vez* (ou *impressora de linhas*). Neste caso, é geralmente desejável que a impressora seja dotada de alguma espécie de acumulador que possa armazenar todos os bits que determinarão os caracteres numa linha completa. Se cada caráter contiver 6 bits, e 50 caracteres forem impressos na página, será necessário armazenamento para 300 bits binários. Cada caráter codificado de 6 bits deve ser decodificado e a respectiva alavanca de tipos posicionada antes que os martelos possam causar a impressão de uma linha. Este tipo de impressora pode imprimir até 4 linhas por segundo, uma velocidade muito mais alta do que a da impressora de um caráter de cada vez.

Até mesmo impressoras mais rápidas são construídas de modo que os caracteres em relevo sejam distribuídos ao redor de uma "roda de impressão" que gira constantemente. Neste caso, a roda de impressão não contém partes móveis como as palhetas acima descritas, mas consiste de um tambor acionado a motor com várias faixas em número igual ao número de caracteres impressos por linha. Um conjunto de todos os caracteres que são usados é distribuído ao redor de cada faixa. A roda de impressão é girada continuamente. Quando o caráter selecionado estiver na posição, o martelo de impressão golpeia a fita contra o papel e, portanto, contra o caráter em relevo sobre a roda de impressão localizada atrás do papel. Uma impressora deste tipo requer um decodificador e uma memória para cada posição de caráter ao longo da linha e também um codificador de caráter de sincronização para cada posição, que determina o instante em que o caráter selecionado está em posição. Impressoras deste tipo podem imprimir até 1.250 linhas por minuto com 160 caracteres por linha.

Dois tipos destas impressoras são mostrados nas Figs. 8.16 e 8.17. A Fig. 8.16 mostra a impressora IBM 1403, em que o papel é movido verticalmente em frente a uma *cadeia* de caracteres elevados de tipos. Esta cadeia está continuamente movendo-se no sentido horizontal, de modo que cada um dos 48 caracteres diferentes passa continuamente em cada uma das 100 posições da impressora em cada linha. (Outros números de caracteres por linha são disponíveis.) Quando um caráter a ser impresso passa pela posição em que ele deve ser impresso, o ímã do martelo da armadura é energizado, batendo num martelo e forçando o papel contra o tipo naquela posição. Uma fita de tinta é colocada entre o papel e o tipo de modo que o caráter seja impresso a tinta no papel.

A Fig. 8.17 mostra os principais componentes mecânicos da impressora série N da Dian, que tem uma roda ou rolo de tipos com 17 trilhas circulares de caracteres impressos, e é, portanto, uma impressora de 17 caracteres por linha. Este rolo de tipos está continuamente girando, e, quando um caráter selecionado passa pela posição em que ele deve ser impresso, o acionador de impressão empurra o martelo balístico escolhido contra o papel, forçando o papel contra o caráter selecionado no rolo de tipos. O rolo de tipos é continuamente irrigado por um rolo de tinta, e nenhuma fita é utilizada. Neste sistema, um disco de código e um codificador de posição angular são usados para dizer que caráter está correntemente em posição de ser impresso. Os codificadores de posição angular serão estudados numa seção seguinte. A Fig. 8.17 mostra que o papel é movido horizontalmente após cada linha ser impressa.

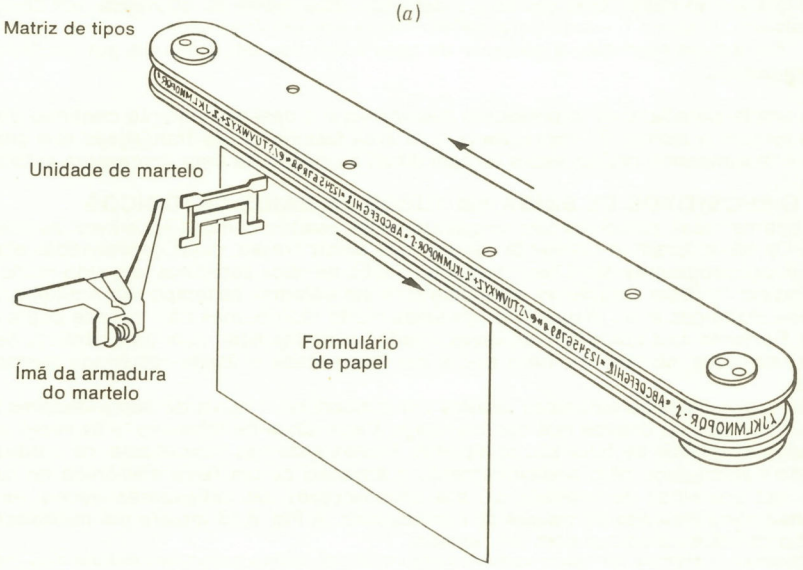
Vejamos a Fig. 1.12, que mostra uma impressora de fichas de razão usadas em sistemas de contabilidade. Esta impressora é típica das utilizadas em muitos sistemas orientados para as atividades comerciais. As fichas de razão são colocadas na fenda do alimentador no centro pouco acima do teclado. A impressora é muitas vezes usada junto com um arquivo em discos, afixando automaticamente os dados dos arquivos nas fichas de razão.

As limitações naturais da velocidade nos dispositivos eletromecânicos provocaram o desenvolvimento de impressoras chamadas *impressoras sem impacto*. Estas impressoras recaem principalmente nas seguintes categorias:

1. *Eletromagnéticas* Usando técnicas de gravação magnética, uma imagem magnética do que deve ser impresso pode ser gravada numa superfície de tambor. Esta superfície é depois passada através de pó magnético que adere às áreas carregadas. O pó é prensado em seguida no papel. Velocidades de até 250 caracteres por segundo são obtidas em tais sistemas.
2. *Eletrostáticas* Nas impressoras eletrostáticas, o papel é revestido com um material dielétrico não-condutor que requer as cargas quando são aplicadas tensões usando cabeçotes ou "pontas" de gravação. Estes cabeçotes gravam "pontos" no papel quando ele passa. Mais tarde, o papel passa através de um virador contendo material com partículas coloridas que transportam uma carga contrária à que foi gravada pelas pontas e, em conseqüência, as partículas aderem às áreas magnetizadas formando caracteres impressos.
3. *Impressoras térmicas* Um pulso elétrico pode ser convertido em calor sobre as seções



(*a*)



(*b*)

Fig. 8.16 (*a*) Impressora de linha de alta velocidade; (*b*) cadeia de tipos para a impressora.

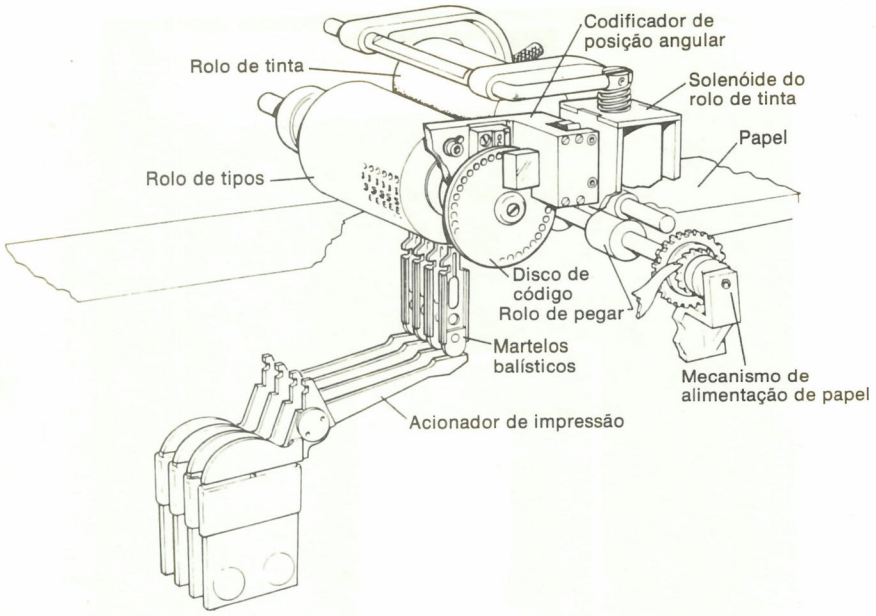


Fig. 8.17 Impressora de tambor de uma linha de cada vez.

selecionadas de um cabeçote de gravação ou sobre fios ou pontas de gravação. Quando este calor é aplicado a um papel sensível ao calor, é impresso um caráter.

4. *Jatos de tinta* Algumas impressoras orientam uma corrente de tinta de alta velocidade em direção ao papel. Esta corrente é defletida, geralmente ao atravessar um campo eletrostático tal como o usado para defletir feixes nos osciloscópios.

   Em alguns sistemas, a corrente de tinta é dividida em gotículas por um transdutor ultra-sônico.

A ampla variedade de impressoras disponíveis e o desenvolvimento contínuo de novas idéias tornam o campo de impressoras uma área fascinante. As transações que envolvem custo e desempenho muitas vezes tornam difícil a seleção de uma impressora satisfatória.

8.11 DISPOSITIVOS DE SAÍDA EM TUBOS DE RAIOS CATÓDICOS

Os tubos de raios catódicos são freqüentemente usados como dispositivos auxiliares de saída. Os tubos foram inicialmente usados para exibir curvas, cujas coordenadas eram calculadas por programas. Mais recentemente, o tubo de raios catódicos tem sido muito usado para indicar também caracteres, especialmente em sistemas de tempo real em-linha. O tubo de raios catódicos é um dispositivo de saída muito rápido, mas não fornece cópia permanente. Portanto, tais tubos são, às vezes, usados em conjunção com uma câmara, de modo que a indicação na face do tubo possa ser fotografada e assim registrada permanentemente.

Os tubos de raios catódicos usados em indicadores visuais de computadores são do mesmo tipo que os usados nos osciloscópios e aparelhos de televisão, e às vezes são empregados aparelhos de televisão completos. Nestes sistemas indicadores, os pontos exibidos são constituídos pelo posicionamento e ativação de um feixe eletrônico no tubo, tal como nos aparelhos de televisão ou nos osciloscópios. Os indicadores visuais, às vezes, são chamados indicadores visuais de *osciloscópio*. A Fig. 8.18 mostra um indicador visual de tubo de raios catódicos com um teclado.

Diversos sistemas têm sido usados para codificar dados para transmissão aos indicadores visuais de osciloscópio. O mais simples sistema consiste em transmitir simplesmente, em binário, as coordenadas x, y de cada ponto a ser indicado. A eletrônica do osciloscópio então, usando conversores D/A (conversores digitais/analógicos), converte os números bi-

nários em voltagens, que são usadas para posicionar o feixe.

O sistema acima é simples, mas ineficiente, no sentido de que muitos pontos devem ser transmitidos, o que consome um tempo considerável do computador. Além disso, se o console ou o indicador visual estiver num local distante e forem usadas linhas telefônicas, como é muitas vezes o caso, a transmissão dos pontos pode ser dispendiosa. (Na maioria destes indicadores visuais de osciloscópio, os pontos devem ser iluminados no mínimo 30, e geralmente mais, vezes por segundo. Há, no entanto, *osciloscópios de armazenamento*, que guardam os pontos exibidos e não requerem "revigoramento". Estes devem ser apagados, no entanto, antes da regravação e são relativamente caros atualmente.)

Em virtude do problema de transmitir repetidamente pontos de dados ao dispositivo

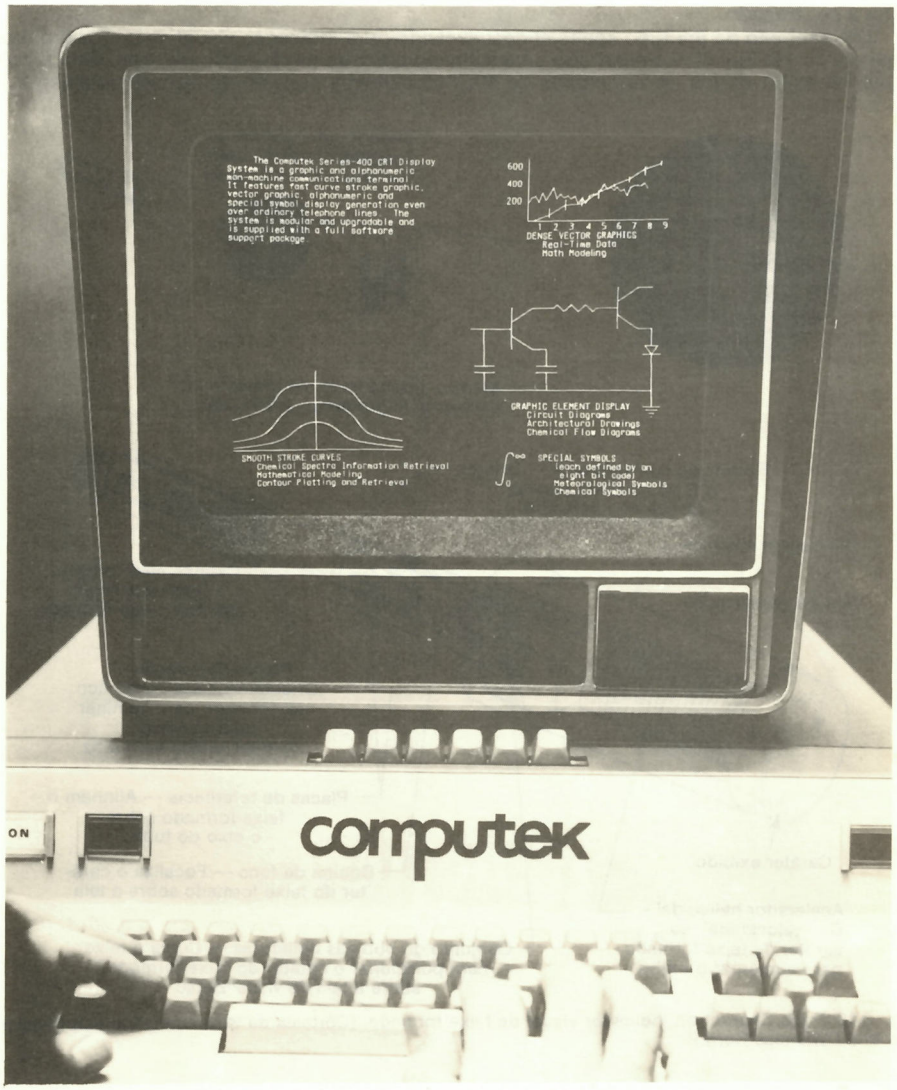


Fig. 8.18 Indicador visual de tubo de raios catódicos. (Cortesia da Computek, Inc.)

indicador visual, geralmente a eletrônica do dispositivo indicador inclui uma pequena memória que armazena os dados provenientes do computador e depois gera a representação visual dos últimos dados transmitidos alterando ou atualizando o indicador visual somente quando o computador assim determinar.

Algumas técnicas mais sofisticadas, mas também mais eficientes, para transmissão de dados aos indicadores visuais de osciloscópio são freqüentemente usadas. Se apenas dados alfanuméricos devem ser exibidos, o código ASCII é geralmente usado na transmissão de cada caráter a ser exibido. A eletrônica do osciloscópio em seguida converte cada caráter de 8 bits codificado em ASCII na seqüência de pontos requerida para exibir o caráter.* As linhas das representações visuais gráficas tais como a indicada na Fig. 8.18 são muitas vezes desenhadas com emprego de uma técnica pela qual o computador transmite a diferença em coordenadas entre a localização atual do feixe eletrônico e o segmento de linha a ser traçado.

Um exemplo de um tubo especial desenvolvido para indicação visual de dados é o tubo Charactron. Na Fig. 8.19 é mostrada uma vista em corte de um tubo Charactron. Este tubo é capaz de exibir dados na velocidade de 100.000 palavras por minuto aproximadamente. A

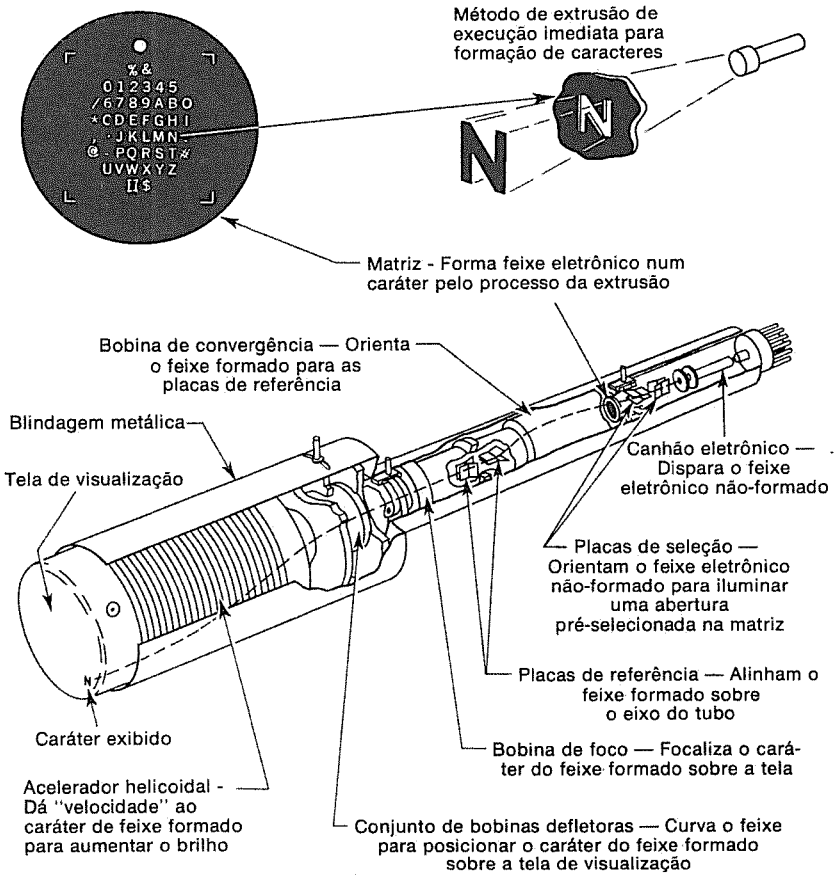


Fig. 8.19 Tubo Charactron indicador visual de feixe formado. (Cortesia da Stromberg-Carlson Division, General Dynamics Corp.)

*Os osciloscópios (*scopes*) são muito freqüentemente explorados à razão de uma linha de cada vez, como nos aparelhos de televisão, e, portanto, é necessária alguma eletrônica de conversão.

parte de formação de caráter do tubo Charactron ilustrada contém uma pequena matriz através da qual 64 caracteres foram cortados. O feixe eletrônico atravessa as placas de seleção que o orientam até o caráter selecionado na matriz. O feixe é, portanto, constituído na forma do caráter selecionado. A bobina de foco controla as dimensões do caráter formado, e o conjunto de bobinas de deflexão em seguida posiciona sobre a face do tubo o feixe formado.

8.12 OUTROS DISPOSITIVOS DE SAÍDA

Dois outros dispositivos de saída são as máquinas de perfuração de cartões ou de fitas em alta velocidade. A informação originária do computador é perfurada ou em cartões ou em fitas de papel, de modo que o meio de saída se assemelhe ao meio de entrada. Se a saída deve ser usada, os cartões ou fitas perfurados devem então ser interpretados e impressos; as fitas perfuradas e os cartões podem, portanto, ser imaginados como uma etapa intermediária.

A fita magnética também é usada como um meio de armazenamento intermediário para operação de equipamento de saída fora-de-linha. Uma vez que a fita magnética pode ser gravada muito mais rapidamente do que quaisquer cartões ou fitas possam ser perfurados, a operação de saída do computador é acelerada. Neste caso, uma leitora de fita magnética e um tradutor são necessários para ler e interpretar a informação codificada em binário gravada, e, finalmente, é necessária uma impressora para imprimir a informação.

Uma vantagem destes dispositivos reside na relação velocidade/custo atingível. Um gravador de fita magnética de alta velocidade será muito mais rápido que uma impressora de alta velocidade, e a impressão final pode então ocorrer numa impressora mais lenta, que interpreta a fita gravada e imprime o resultado final. Este é um outro caso de operação fora-de-linha, em que a operação final é efetuada na velocidade de um dispositivo de saída mais rápido e o computador não é tão retardado como seria se fosse usada uma impressora de baixa velocidade. Até os dispositivos gravadores de fita magnética, no entanto, são inerentemente muito mais lentos do que o computador.

8.13 CÓDIGOS DE CORREÇÃO DE ERRO E DE DETECÇÃO DE ERRO

O processo da transferência de informação para dentro da máquina e para fora da máquina é especialmente sujeito a erro. Embora as leitoras de fita e de cartão sejam construídas com a mais alta consideração possível à operação correta, e a ocorrência de erros seja relativamente pequena, os erros ainda ocorrem, e é desejável detectá-los sempre que possível. Para facilitar a detecção ou a correção de erros, foram inventadas duas classes de códigos, (1) códigos de detecção de erros e (2) códigos de correção de erros. O primeiro tipo de código permite que o equipamento detecte os erros que ocorrem nos grupos codificados de bits, e o segundo tipo de código corrige os erros automaticamente.

Ambos os códigos de correção de erros e de detecção de erros requerem que a informação redundante seja transmitida junto com a informação real que está sendo processada. O tipo mais comumente usado de código de detecção de erros é indubitavelmente o código de teste de paridade. Os códigos de teste de paridade são comumente usados em leitoras de fita e de cartão e no armazenamento de informação em fita magnética.

Teste de paridade

O teste de paridade se baseia no uso de um bit adicional, conhecido por "bit de paridade" ou bit de teste de paridade em cada grupo de código. O bit de paridade associado a cada grupo de código num sistema de teste de *bit de paridade par* tem um valor tal que o número total de 1 em cada grupo de código mais o bit de paridade é sempre par. (Um código de teste de *bit de paridade ímpar* tem um bit de paridade tal que a soma dos 1 no grupo de código mais o bit de paridade é sempre um número ímpar.) O exemplo mostrado no Quadro 8.1, que usa um código 8, 4, 2, 1, tem um bit de paridade par que torna a soma dos 1 em cada grupo de código um número par.

Se ocorrer um único erro na transmissão de um grupo de código — por exemplo, se 0010, 1 for erradamente alterado para 0011, 1 — o fato de que há um número ímpar de dígitos 1 no grupo de código mais o bit de paridade indicará que ocorreu um erro. Se os valores dos bits de paridade tivessem sido selecionados de modo que a soma total dos dígitos 1 em cada grupo de código mais o bit de paridade fosse ímpar em vez de par, cada bit de paridade seria o complemento do bit de paridade mostrado acima e o código seria um código de teste de bit de paridade ímpar.

A técnica do teste de paridade é sem dúvida o método mais popular de detecção de erros nos grupos de códigos armazenados, especialmente para os dispositivos de armazenamento tais como fita magnética, fita de papel e até para os sistemas de tambor e núcleos.

Se for usado o sistema de bit de paridade, um bit adicional deve ser transmitido com cada grupo de código. Por exemplo, se for usado o grupo de código Baudot de 5 bits, cada linha de uma fita perfurada terá uma posição de orifício adicional que conterá um 1 ou 0. Quando a fita for lida pela leitora de fitas, cada grupo de código será examinado, junto com o bit de paridade; e, num sistema de bit de paridade ímpar, será gerado um alarme se o número de dígitos 1 num grupo for par.

Este tipo de teste detectará todos os números ímpares de erros. Suponhamos que um teste de paridade par seja usado conforme no Quadro 8.1 e o grupo de código a ser transmitido seja 0010; o bit de paridade neste caso será um 1. Se o grupo de código for erroneamente lido como 0110, o número de dígitos 1 no grupo de código mais o bit de paridade será ímpar e o erro será detectado. Se, no entanto, for cometido um duplo erro e 0010 for alterado para 0111, o erro não será detectado já que o número de dígitos 1 será novamente par. Um teste de bit de paridade detectará somente números ímpares de erros.

(A regra acima também se aplicará quando o bit de paridade estiver errado. Por exemplo, consideremos um sistema de teste de bit de paridade par onde 0010 deve ser transmitido e o bit de paridade for 1. Se o bit de paridade for alterado para 0, o número de dígitos 1 no grupo de código mais o bit de paridade será ímpar e o erro será detectado.)

O teste de paridade também pode ser usado para testar palavras inteiras. Neste caso, o procedimento é o mesmo que nos grupos de código menores; o bit de paridade é arranjado de modo a fazer o número total de dígitos 1 em cada palavra mais o bit de paridade, par ou ímpar, dependendo do sistema usado.

Há muitos tipos de códigos de correção de erros, e alguns esquemas de codificação muito engenhosos e sofisticados são usados tanto nos sistemas de comunicações quanto nos sistemas de computadores. Por exemplo, a fita magnética é um dispositivo de memória que é especialmente propenso a erros. A maioria desses erros se deve a imperfeições na fita ou corpos estranhos que ficam entre os cabeçotes de leitura e a fita, fazendo com que a fita seja fisicamente empurrada para fora do cabeçote de leitura e o sinal registrado seja incorretamente interpretado. Diz-se que tais erros são causados por "desexcitação". Estes erros tendem a permanecer numa única trilha e há diversos códigos engenhosos que foram usados para detectar e corrigir tais erros.*

QUADRO 8.1

Decimal	BCD	BIT de Paridade par
0	0000	0
1	0001	1
2	0010	1
3	0011	0
4	0100	1
5	0101	0
6	0110	0
7	0111	1
8	1000	1
9	1001	0

8.14 TERMINAIS

Quando um teclado for combinado a um indicador visual de osciloscópio ou um mecanismo de impressão e a eletrônica adequada for fornecida de modo que os caracteres batidos no teclado possam ser introduzidos num computador e este computador possa ter também a capacidade de imprimir no osciloscópio ou no mecanismo de impressão, a impressora de teclado ou o osciloscópio de teclado constituem o que se chama um *terminal*.

*Um estudo dos códigos de correção de erro e de detecção de erro com uma certa profundidade pode ser achado em W. W. Peterson, *Error Correcting Codes*, e em G. Birkhoff e T. C. Bartee, *Modern Applied Algebra*.

Os terminais são amplamente usados para introduzir programas e dados nos computadores. O teletipo mostrado na Fig. 1.7 foi um terminal primitivo que continua a ser muito utilizado como um terminal de computador. Os teletipos proporcionam um meio de introduzir caracteres alfanuméricos num computador. Eles possibilitam ao computador imprimir respostas para um usuário, e uma leitora e uma perfuradora de fitas são geralmente incluídas de modo que os programas ou dados possam ser permanentemente registrados em fita de papel e de modo que tais programas ou dados possam ser lidos no computador quando desejado.

A Fig. 8.18 mostra um terminal que consiste de um osciloscópio e um teclado. Os caracteres batidos no teclado aparecerão na face do osciloscópio, e sinais de saída são também fornecidos adequados ao uso do computador. Um computador também pode imprimir no osciloscópio, fornecendo comunicações de dupla via. A desvantagem principal é que nenhum "impresso" ou registro do que é datilografado pelo operador do terminal ou impresso pelo computador permanece quando o terminal for desligado. Alguns meios de registro de dados selecionados estão agora sendo introduzidos nos mais dispendiosos sistemas, e estes incluem câmaras para fotografar a face do osciloscópio, pequenas impressoras e cassetes de fitas para registrar dados.

Os terminais variam desde mínimos sistemas que contêm quase nenhuma memória até os terminais "sofisticados" (smart) em que é incluído um pequeno microcomputador ou minicomputador. Os terminais sofisticados proporcionam muitas facilidades ao usuário, incluindo, às vezes, edição de textos, formatação da entrada e verificação de erros de datilografia cometidos pelos operadores.

As combinações de teclado-osciloscópio quase sempre contêm uma pequena memória na qual o conteúdo atual da tela é armazenado de modo que a tela possa ser revigorada pela eletrônica do terminal em vez de requerer que o computador regrave continuamente os dados na tela. Neste caso, uma pequena memória, bem como alguma eletrônica, é necessária, e os microprocessadores são muitas vezes usados para controlar a operação do terminal. Aumentando-se as capacidades da memória e o programa no microprocessador, um terminal sofisticado pode ser formado sem demasiado custo adicional.

Os terminais usualmente geram dados de saída em forma de série e aceitam os dados de entrada em forma de série. Os formatos, a codificação e as propriedades elétricas dos sinais gerados variam de terminal a terminal, mas há atualmente fortes movimentos ou tendências para padronizar os projetos das interfaces.

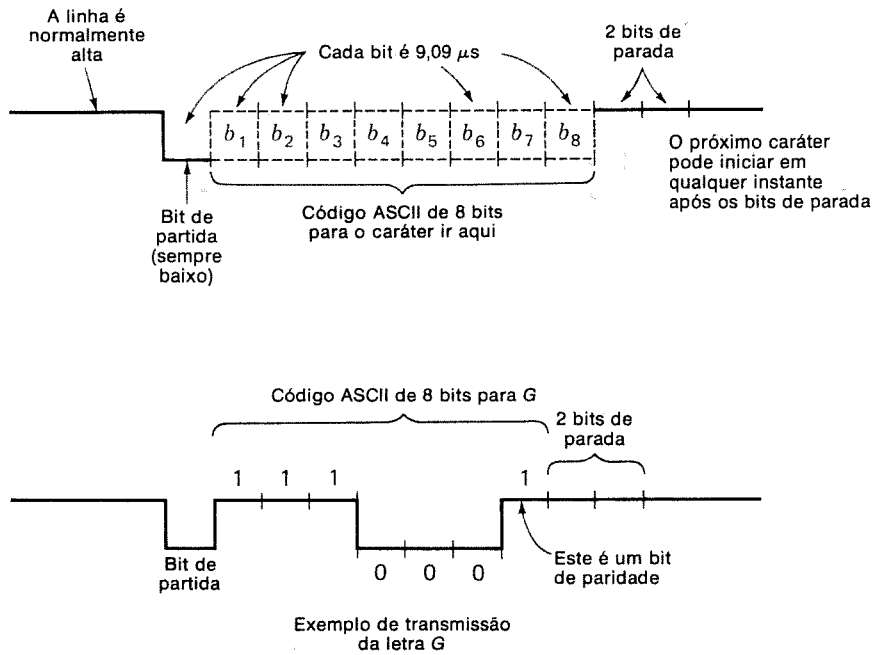
Os teletipos e a maioria dos terminais nos dias de hoje costumam gerar uma saída de 11 bits em forma de série sempre que uma tecla no teclado for abaixada. Esta mesma corrente é geralmente usada para ativar o mecanismo da impressora; no entanto, em teletipos e em muitos outros dispositivos é possível soltar o teclado do mecanismo de impressão. Neste caso, quando uma tecla é batida, o código do caráter é lido pelo computador que deve então gravar este caráter novamente na impressora do terminal. Isto é chamado "repercussão" do caráter.

A Fig. 8.20 mostra como opera a transmissão atual do caráter padrão. A linha de saída do teclado está normalmente no estado alto,* mas, quando uma tecla é abaixada, é gerado um *bit de partida* num nível baixo. Este bit de partida é seguido pelo caráter adequado de 8 bits codificado em ASCII, e depois dois *bits de parada* no nível alto são inseridos antes que um outro caráter possa ser iniciado (isto é, antes que um outro bit de partida possa ser gerado). O bit de partida, cada bit de parada e os bits no caráter codificado em ASCII são cada um da mesma duração de tempo. Na maioria dos sistemas comuns, isso representa $^1/_{110}$ s para cada um, e, já que são necessários 11 de tais períodos de tempo ou *tempos de bit*, um único caráter requer $^{11}/_{110}$ ou $^1/_{10}$ s.

Há diversas outras velocidades que são normalmente usadas, incluindo 300, 600 e 1200 bits/s. Em cada caso, a mesma construção do caráter com bits de partida e de parada é utilizada, e, portanto, são atingíveis as velocidades de caráter de $^{11}/_{300}$, $^{11}/_{600}$ e $^{11}/_{1200}$ s.

O sistema de transmissão de caráter aqui descrito é chamado *transmissão assíncrona*, pois o caráter e os bits de partida podem ocorrer em qualquer instante de tempo. Os sistemas de *transmissão síncrona* têm os bits sincronizados em períodos fixos de tempo, e os caracteres são também colocados em posições fixas na corrente de bit. Estes sistemas requerem tanto a sincronização de bits quanto a sincronização de caracteres a serem estabelecidas entre o dispositivo de transmissão e o receptor e são, portanto, mais complicados. A transmissão de caracteres, no entanto, podem ser em velocidades mais altas. (Já que os bits de partida e parada não são necessários, os começos e os finais dos caracteres são

*Os técnicos em Comunicações denominam esta grandeza valor de *marca*, e o outro nível é denominado *espaço*.



Nota: Para fazer um caráter de 8 bits de um código ASCII de 7 bits, um bit de paridade ímpar é colocado na 8.ª posição

Fig. 8.20 Transmissão de um caráter usando o código padrão.

estabelecidos pelo sistema.) Em conseqüência, a comunicação de dados em alta velocidade é geralmente em forma síncrona.

Os níveis de saída e os requisitos da interface para os caracteres codificados são examinados nos exercícios, e vários padrões são observados.

Quando os terminais forem operados a uma certa distância de um computador, o sistema telefônico é muitas vezes usado para fornecer o enlace de comunicações necessário. Se um terminal deve ser operado no sistema telefônico, dispositivos especiais são necessários para traduzir os níveis lógicos produzidos pelo terminal em sinais aceitáveis para transmissão em linha telefônica.

A Fig. 8.21 mostra um terminal com um *acoplador acústico.* O monofone ou aparelho de mão do telefone é colocado neste acoplador acústico. Quando uma tecla é abaixada, os bits que constituem o caráter são convertidos em tons de áudio usando um pequeno alto-falante no acoplador. Geralmente é usada uma freqüência para um 0 e uma outra para um 1. Estes sinais entram na parte do transmissor do aparelho de mão e são transmitidas à linha telefônica. Na extremidade do computador, estas freqüências são recebidas por um microfone conectado a um aparelho de mão e depois novamente convertidas a dígitos 1 e dígitos 0 em forma elétrica, de modo que a saída de níveis lógicos elétricos no receptor constitui réplicas dos sinais originalmente gerados no terminal.

O computador "fala" novamente para o terminal usando um acoplador acústico em sua extremidade voltada para a linha telefônica, e o acoplador acústico no terminal converte os sinais do computador novamente a níveis lógicos que são usados para comandar o indicador visual.

Quando os sinais de saída em forma de níveis lógicos são convertidos diretamente em sinais elétricos adequados à transmissão telefônica que usa conjunto de circuitos eletrônicos (em vez de acoplamento acústico), o dispositivo de conversão é chamado um *modem*.



Fig. 8.21 Indicador visual de osciloscópio e terminal.

Um modem pode geralmente converter níveis lógicos em sinais elétricos para o sistema telefônico e pode converter sinais recebidos da linha telefônica novamente em níveis lógicos. Isto significa, naturalmente, que as ligações para a linha telefônica são feitas eletricamente e diretamente na linha telefônica, e não é usado um fone de cabeça. Isso também significa que os sinais elétricos devem estar de acordo com as normas do sistema telefônico; e o projeto de modems que (1) irão transmitir bits através das linhas telefônicas em altas velocidades, (2) provocarão poucos erros na transmissão e (3) estarão de acordo com as normas da companhia telefônica; esta é uma área altamente desenvolvida e de interesse científico.

À semelhança dos acopladores acústicos, os modems geralmente usam uma freqüência (diferente) para um 1 e um 0. Por exemplo, o modem 103 da Bell transmite dados através de linhas telefônicas ou em 110 ou em 300 bits/s. O modem numa extremidade de uma linha telefônica usa 1.070 Hz (para os 1) e 1.270 Hz (para os 0) para transmitir, enquanto que o modem na outra extremidade usa 2.025 Hz (para os 1) e 2.225 Hz (para os 0). O motivo dos dois conjuntos de freqüências é que a transmissão pode ser em qualquer direção e, enquanto um modem estiver transmitindo um caráter, o outro estará ainda transmitindo sua freqüência alta (marca). (Uma única linha telefônica manipula comunicações em ambas as direções. Ambas as extremidades podem "conversar" ao mesmo tempo. Você pode, por exemplo, interromper alguém que esteja conversando.)

8.15 DISPOSITIVOS DE ENTRADA-SAÍDA DE SISTEMAS COM COMPONENTES ANALÓGICOS

Nem todas as entradas das máquinas digitais consistem de dados alfanuméricos que possam ser diretamente perfurados em cartões ou fita. Os computadores usados nos sistemas de redução de dados ou em sistemas de controle de tempo real muitas vezes têm entradas que são expressas como a posição física de algum dispositivo ou como sinais elétricos que são analógicos de natureza. Um exemplo de uma posição física que pode ser usada como

uma entrada para um computador digital é um sistema de controle em tempo real no qual um computador é usado automaticamente para apontar um telescópio. Se, por algum sistema de engrenagens, a posição do telescópio ao longo de um eixo geométrico for relacionada à posição de um eixo mecânico, a posição deste eixo mecânico deve ser lida no computador. Isto envolverá a tradução da posição do eixo mecânico num número codificado em binário que pode ser lido pela máquina.

A conversão de um deslocamento físico ou de um sinal elétrico analógico numa representação digital é chamada *conversão analógica/digital* ou *conversão A/D*. Dois tipos importantes de conversores A/D são: (1) os que convertem deslocamentos mecânicos numa representação digital e (2) os que convertem um sinal elétrico analógico em sinais codificados digitais.

Suponhamos que um dispositivo analógico tenha como saída uma tensão que deve ser usada por uma máquina digital. Admitamos que a tensão varie dentro do limite de 0 a 63 V cc. Podemos então representar os valores de tensão com um conjunto de números de 6 bits variando de 000000 a 111111 e, para cada valor que a tensão de entrada pode assumir, atribuímos um valor correspondente do número de 6 bits. Se a tensão de entrada for 20 V, o valor digital correspondente será 010100. Se, no entanto, o sinal de entrada estiver em 20,249 V cc, o número binário de 6 bits não descreverá completamente a tensão de entrada, mas apenas se aproximará do valor de entrada. O processo de aproximação do valor de entrada é chamado *quantização*. O número de bits no número binário que representa o sinal analógico é a *precisão* do codificador, e o valor do erro que existe entre os valores digitais da saída e os valores analógicos de entrada constitui uma medida da *exatidão* do codificador.

Não somente são as entradas do computador às vezes em forma analógica, mas é muitas vezes desejável que as saídas do computador sejam expressas em forma analógica. Um exemplo disto reside no uso de um tubo de raios catódicos como um dispositivo de saída. Se a saída do computador deve ser exibida com uma posição sobre um tubo de osciloscópio, então os sinais de saída codificados em binário do computador devem ser convertidos em tensões ou correntes, que podem ser usadas para posicionar o feixe eletrônico e que são proporcionais à magnitude do número binário de saída representado pelos sinais de saída do computador. Isto envolve a conversão digital/analógica, e um dispositivo que executa esta conversão é chamado *conversor* D/A. Quando os computadores digitais são usados em sistemas de controle, geralmente é necessário converter as saídas digitais da máquina em sinais do tipo analógico, que são depois usados para controlar o sistema físico.

8.16 CONVERSORES ANALÓGICO/DIGITAL — CODIFICADORES DE POSIÇÃO ANGULAR

O primeiro tipo de codificador que será sucintamente descrito abaixo converte uma posição de eixo mecânico (ou posição angular de um eixo) num número digital codificado em binário. Vários tipos diferentes de dispositivos executarão esta conversão; o tipo descrito é representativo dos dispositivos atualmente em uso, e deve ser estendido que codificadores mais sofisticados podem produzir precisão adicional. Além disso, é possível, em geral, converter uma posição física num sinal elétrico do tipo analógico e depois converter este sinal num sistema digital. Por exemplo, um eixo pode ser mecanicamente acoplado a um potenciômetro com uma tensão contínua (cc) aplicada às extremidades da resistência. Se o sinal de saída for depois tirado da derivação do potenciômetro, o sinal de saída de cc estará relacionado à posição do eixo. Em geral, no entanto, podem ser construídos codificadores precisos e mais diretos, eliminando-se a etapa intermediária da conversão de uma posição física num sinal elétrico analógico.

A Fig. 8.22(*a*) ilustra um disco de segmentos codificados que está acoplada ao eixo. Um conjunto de escovas é então ligado de modo que uma só escova seja posicionada no centro de cada faixa concêntrica do disco. Cada faixa é construída com vários segmentos feitos ou de material condutor (as áreas escurecidas) ou de algum material isolante (as áreas não-sombreadas). Um sinal é conectado ao condutor, e, se uma determinada escova estabelecer contato com um segmento do material condutor, resultará um sinal 1, mas, se a escova estiver sobre o material isolante, a saída da escova será um 0. As quatro linhas de saída do codificador mostrado representarão um número binário de 4 bits. Há 16 intervalos distintos ao redor do disco codificador, cada um correspondendo a um intervalo diferente da posição do eixo e cada um fazendo com que o codificador tenha uma saída de número binário diferente.

Um codificador fotoelétrico pode ser construído usando um disco codificador com faixas divididas em segmentos transparentes (as áreas sombreadas) e segmentos opacos (as áreas não-sombreadas). Uma fonte de luz é colocada sobre um lado do disco, e um con-

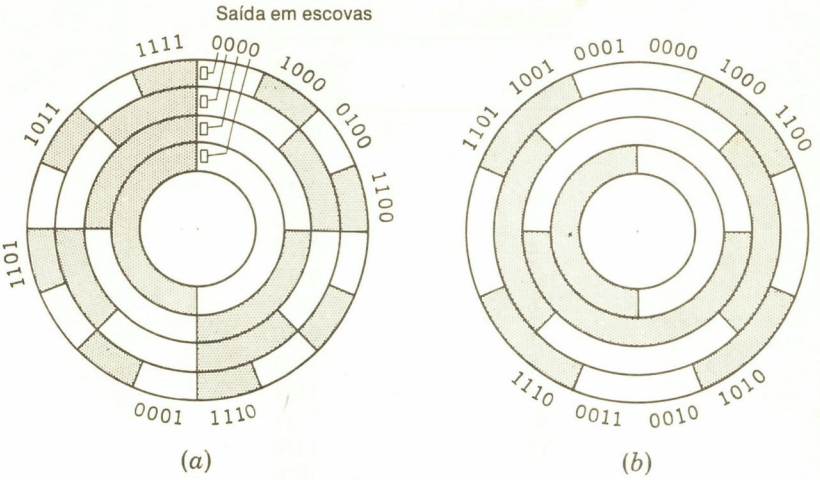


Fig. 8.22 Discos codificadores de posição de eixo mecânico. (*a*) Discos codificados em binário. (*b*) disco de código de distância unitária.

junto de quatro células fotoelétricas é colocado sobre o outro lado, arranjado de modo que uma célula esteja atrás de cada faixa do disco codificador. Se um segmento transparente estiver entre a fonte de luz e uma célula sensível à luz, resultará uma saída 1; e, se uma área opaca estiver em frente da célula fotoelétrica, haverá uma saída 0. Aumentando-se o número de faixas ao redor do disco, mais precisão será acrescentada ao codificador. O tipo fotoelétrico de codificador tem maior resolução do que o tipo de escova, e até maior resolução pode ser obtida usando-se engrenagens e vários discos. O estado atual da técnica é de cerca de 18 bits ou 2^{18} posições por revolução do eixo, mas a maioria dos codificadores comerciais tem 14 bits ou menos.

Há uma dificuldade básica com o codificador ilustrado: se o disco estiver numa posição em que o número de saída está variando de 011 a 100, ou em qualquer posição onde vários bits estão mudando de valor, o sinal de saída pode tornar-se ambíguo. Já que as escovas são de largura finita, elas cobrirão em parte a variação nos segmentos; e independente de quão cuidadosamente seja ele feito, o codificador terá saídas erradas em várias posições. Se isto ocorrer quando 011 estiver mudando para 100, vários erros são possíveis; o valor pode ser lido como 1,11 ou 000, qualquer um dos quais é um valor com erros consideráveis. Para contornar esta dificuldade, foram usados vários esquemas, geralmente envolvendo dois conjuntos de escovas, com um conjunto ligeiramente deslocado do outro. Escolhendo logicamente dentre as saídas disponíveis, a "ambigüidade" pode ser eliminada sacrificando-se ligeiramente a precisão.

Um outro esquema para evitar ambigüidade envolve o uso de um código "Gray", ou de "distância unitária," para formar o disco codificador [Fig. 8.22(*b*)]. Neste código, 2 bits nunca mudam de valor em números binários codificados sucessivos. Usando um disco codificado em Gray, um 6 pode ser lido como um 7 ou um 4 como um 5, mas erros maiores não serão cometidos. O Quadro 8.2 mostra uma listagem de um código Gray de 4 bits.

Se as entradas da máquina forem provenientes de um codificador que usa um código Gray, os grupos de código devem ser convertidos ao BCD ou ao binário convencionais antes do uso. Esta conversão pode ser simplesmente efetuada, ou seqüencialmente ou combinatorialmente. Os exercícios tratam do código Gray e da conversão com maiores detalhes.

8.17 CONVERSORES DIGITAL/ANALÓGICO

O problema da conversão de um sinal digital numa tensão analógica pode ser resolvido por inúmeras técnicas, a mais direta das quais envolve o uso de uma estrutura de resistores. Um tipo básico de conversor D/A com estrutura de resistores e ilustrado na Fig. 8.23, que mostra uma rede de resistores com quatro entradas. Cada entrada é convertida numa chave que conecta um resistor ou a 0 V ou a V+. Quando uma chave estiver na posição de con-

QUADRO 8.2

Decimal.	Código Gray $a_3a_2a_1a_0$
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

versão para V+, o bit binário representa um 1 e, quando a chave se conecta a 0 V, o bit de entrada é um binário 0. A saída em E_0 será então um potencial de cc na faixa de 0 a V+ e será proporcional ao valor do número binário representado pelas entradas.

Por exemplo, se o número de entrada for 0111, a tensão de saída em E_0 será $^7/_{15}$V+; se a entrada for 1111, a saída E_0 será V+; e, se a entrada for 0001, a saída E_0 estará em $^1/_{15}$V+ V cc. Para obter precisão, os resistores devem ser todos do tipo de precisão. Além disso, as tensões de entrada que representam os valores binários devem ser estabelecidas exatamente, e, já que a carga nas entradas irá variar com o valor dos números da entrada, a entrada deve ser capaz de fornecer a corrente necessária sem um deslocamento no nível da tensão. Mais precisão pode ser obtida aumentando-se o número de entradas e acrescentando-se um resistor para cada entrada.

Por exemplo, se o número de entrada for 0111, a tensão de saída em E_0 será $^7/_{15}$V+; se a entrada for 1111, a saída E_0 será V+; e, se a entrada for 0001, a saída E_0 estará em $^1/_{15}$V+ V cc. Para obter precisão, os resistores devem ser todos do tipo de precisão. Além disso, as tensões de entrada que representam os valores binários devem ser estabelecidas exatamente, e, já que a carga nas entradas irá variar com o valor dos números da entrada, a entrada deve ser capaz de fornecer a corrente necessária sem um deslocamento no nível da

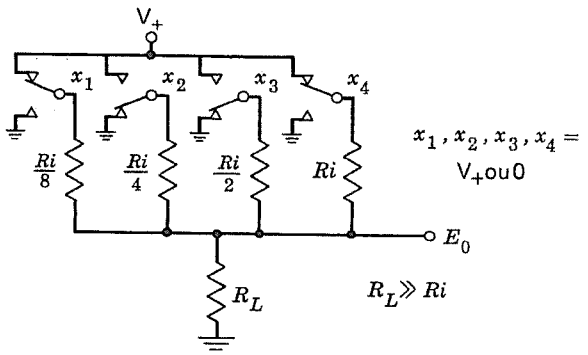


Fig. 8.23 Estrutura de resistores de D/A.

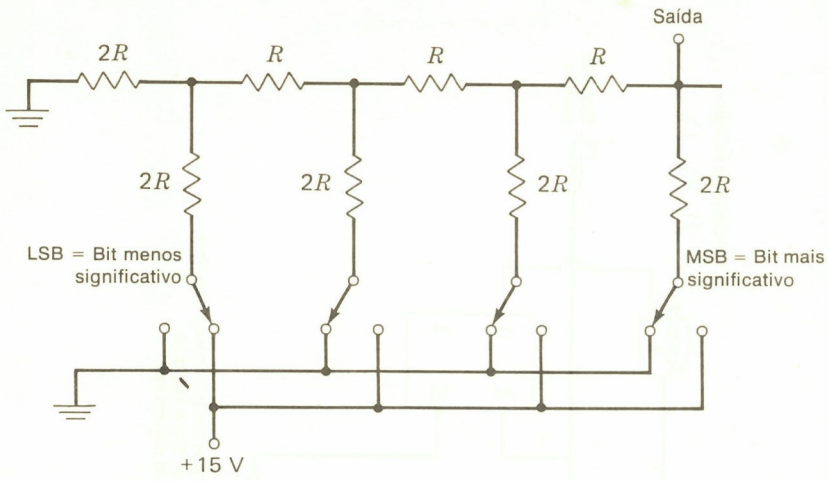


Fig. 8.24 Conversor D/A de resistores R/2R.

tensão. Mais precisão pode ser obtida aumentando-se o número de entradas e acrescentando-se um resistor para cada entrada.

A Fig. 8.24 mostra um outro tipo de estrutura de resistores que pode ser usada na conversão D/A. A vantagem desta estrutura é que apenas dois valores diferentes de resistores são usados. As entradas são mostradas como chaves, mas chaves de semicondutores ou *amplificadores de ajuste de nível* são muitas vezes empregados. A desvantagem deste conversor é que são necessários dois resistores por entrada.

Há muitos outros tipos de conversores D/A, e o projeto e construção destes dispositivos representa uma área em desenvolvimento no campo da computação. Muitos decodificadores são bastante especializados, principalmente aqueles utilizados em sistemas de controle. Por exemplo, as máquinas fresadoras automáticas são muitas vezes controladas por um computador digital que tem suas saídas digitais convertidas no movimento mecânico da maquinaria de fresagem. O uso de computadores digitais nas grandes indústrias, tais como a indústria do petróleo, a de fabricação de papel e a de fabricação de câmeras, também envolve a utilização de muitos tipos diferentes de conversores A/D e D/A.

8.18 CONVERSORES ELETRÔNICOS ANALÓGICOS/DIGITAIS DE ALTA VELOCIDADE

Há inúmeras técnicas usadas para converter um sinal analógico elétrico para a forma digital. Basicamente é necessário um dispositivo que converta um sinal de cc num número binário proporcional ao nível de cc do sinal. Suponhamos que temos uma tensão contínua (cc) com uma amplitude que está no intervalo 0 a +15 V. Se desejarmos converter esta tensão no número binário de quatro dígitos, queremos o número 0000 para 0 a 0,5 V, 0001 para 0,5 a 1,5 V, 0010 para 1,5 a 2,5 V, 1111 para 14,5 V ou maior etc. (Observemos que o número binário corresponde ao centro do intervalo.)

Um simples conversor A/D pode ser feito usando uma estrutura de resistores tal como a da Fig. 8.23, um biestável de PARTIDA-PARADA, um contador binário de quatro biestáveis, quatro conversores de nível e um amplificador de cc de alto ganho.

A configuração básica é mostrada na Fig. 8.25. Os biestáveis X_1, X_2, X_3 e X_4 são arranjados num contador binário que conta 0000, 0001, 0010, ..., 1111. Cada biestável comanda um conversor de nível tal que, quando a saída do biestável for um 1, a saída do conversor de nível é −16 V cc; e, quando a saída do biestável for um 0, a saída do conversor de nível é 0 V cc.

O amplificador de cc de alto ganho é projetado de modo que quando a entrada no amplificador for positiva, a saída do amplificador representa um 0; mas, quando a entrada for negativa, a saída vai a 1, ajustando, portanto, o biestável de PARTIDA-PARADA (e parando o contador).

Suponhamos agora que seja emitido um sinal de PARTIDA, limpando o contador de

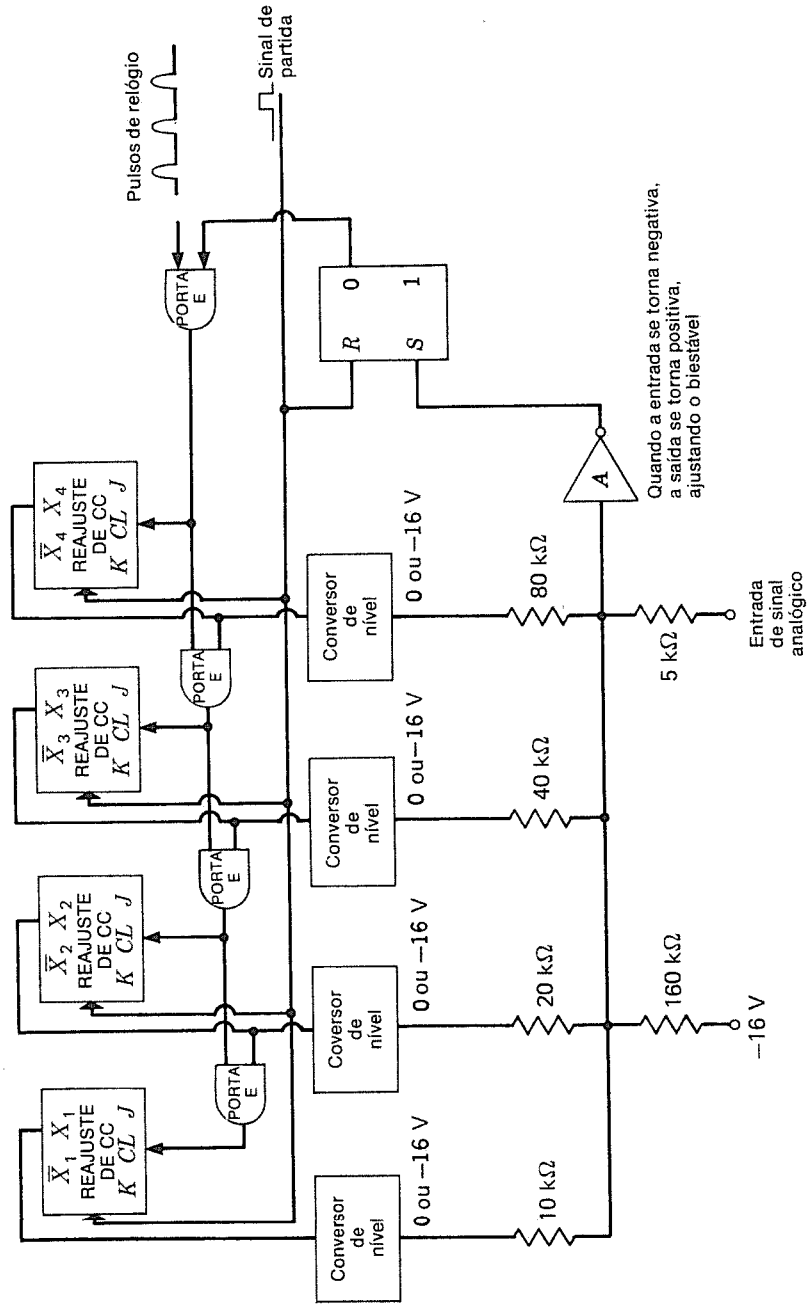


Fig. 8.25 Conversor A/D.

biestável para 0000. Todas as saídas dos conversores de nível estarão em 0 V (a menos que o sinal analógico de entrada seja inferior a +0,5 V cc, o resistor de 160 kΩ a −16 V fará com que a entrada no amplificador seja negativa*), a saída no amplificador será um 0, e o pulso de relógio atravessará a primeira porta E.

Se o sinal de entrada for de 0 a 0,5 V, o amplificador terá uma saída de nível de um 1 e ajustará o biestável de partida-parada em um 1 de modo que não passará nenhum dos pulsos de relógio.

Se o sinal de entrada for maior do que +0,5 V, o contador contará, fazendo assim com que a corrente negativa na junção dos resistores de 10, 20, 40 e 80 kΩ aumente até que a entrada do amplificador fique negativa, parando assim o contador.

Se, por exemplo, a entrada do sinal analógico estiver em +9 V cc, a entrada no amplificador será positiva até que o contador vá de 1000 a 1001; e, quando o contador for um 1001, os pulsos de relógio serão parados, atribuindo o valor correto ao sinal.

Há diversas outras técnicas usadas para a conversão A/D. A mais usada é o *método das aproximações sucessivas* por causa de sua combinação de alta resolução e alta velocidade. O conversor de aproximação sucessiva opera com um tempo de conversão fixo por bit, independente do valor da entrada analógica. O método pode ser explicado usando a Fig. 8.24, já que ela é uma variação desta técnica.

Consideremos que temos controle dos biestáveis individuais na Fig. 8.25 em vez de sua existência num contador. No início da conversão, o bit mais significativo do conversor D/A é ajustado no estado 1, e a saída do conversor, que é agora metade da faixa de entrada analógica em escala total, é comparada com a entrada. Se ela for menor do que a entrada, o bit mais significativo é deixado ligado e o bit seguinte é comprovado. Se a saída do bit mais significativo do conversor de nível para o bit mais significativo for maior do que a entrada, o bit mais significativo é desligado quando o próximo bit menos significativo for ligado.

Este processo de comparação continua em cada bit alternativamente abaixo de e incluindo o bit menos significativo, após o que o registro de saída contém o número digital de saída completo.

Velocidades tão altas quanto 100 ns/bit podem ser obtidas por este método. Os conversores de aproximação sucessiva também podem ser de muita exatidão, mas esta exatidão depende da estabilidade da referência, dos conversores de nível, da estrutura de resistores D/A e do comparador.

Os conversores mais rápidos são do tipo em paralelo. Estes conversores podem ser feitos com velocidades de até 25 milhões de conversões de 4 bits. Às vezes, esses conversores são chamados *conversores simultâneos* ou *instantâneos*.

O método em paralelo de fazer um conversor de n bits emprega um quantizador de entrada constituído de $2^n - 1$ comparadores com um valor de desvio de 1 bit menos significativo devido aos resistores conectados a uma tensão de referência; pela Fig. 8.26, tem-se um conversor A/D de 2 bits que, portanto, possui $2^2 - 1$, ou 3, comparadores. Cada comparador vê a entrada de sinal e a compara com sua tensão de referência particular. Se a entrada de sinal for menor, o comparador funciona; se a entrada de sinal for maior, o comparador fica desligado. Em conseqüência, para uma certa tensão de entrada de sinal nos comparadores, todos os comparadores com entradas provenientes da estrutura de resistores que estejam abaixo do nível de entrada de sinal, funcionam, enquanto que todos os comparadores acima deste nível são cortados. Portanto, o processo da quantização é realizado no tempo de comutação de um só comparador. As saídas do comparador, no entanto, não estão em código binário e devem, portanto, passar através de uma rede de circuitos lógicos para serem convertidas em binário, conforme indica a figura.

O método em paralelo tem a vantagem da velocidade ser mais rápida, mas é limitado a uns poucos bits relativamente, em geral 4, devido ao grande número de comparadores requeridos. A fim de converter um grande número de bits, é necessário empregar uma estratégia misturada pela qual um estágio de conversão em paralelo é seguido por um conversor D/A rápido, cuja saída é subtraída da tensão de entrada; a diferença é então amplificada e convertida usando um outro estágio em paralelo. Isto resulta num compromisso de velocidade, mas resolução mais alta.

Os conversores A/D são comumente feitos para dar ou de 7 a 20 dígitos binários ou de 3 a 7 dígitos decimais nas saídas. O conversor mais rápido atualmente no mercado realiza 250 milhões de conversões por segundo; conversores de preço mais modesto operam em

*A combinação de −16 V e 160 kΩ contribui com tanta corrente para a junção de resistores na entrada do amplificador quanto um sinal de +0,5 V na entrada do sinal analógico.

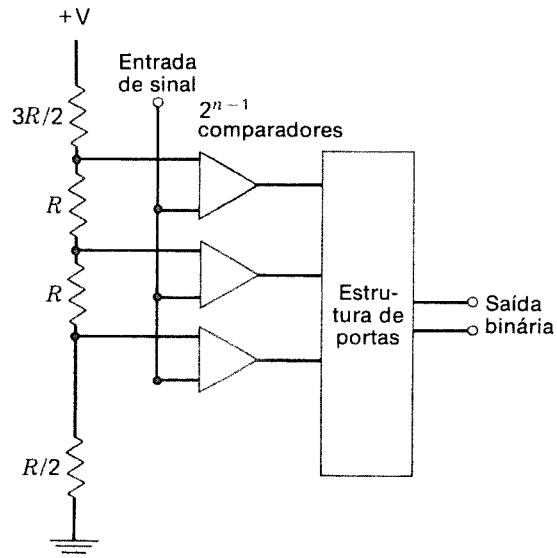


Fig. 8.26 Conversor A/D do tipo paralelo.

freqüências muito mais baixas. Os conversores A/D lentos principalmente destinados a uso manual ou a registro de baixa velocidade são chamados *voltímetros digitais.* Este quase invariavelmente operam usando BCD. Dispositivos indicadores visuais convertem depois o BCD em um número decimal que é exibido e pode ser lido por um operador ou impresso.

O Quadro 8.3 lista alguns dos termos que são usados na descrição de conversores A/D e D/A.

QUADRO 8.3 TERMINOLOGIA DA CONVERSÃO DE A/D E D/A

Resolução A menor variação analógica que pode ser distinguida por um conversor A/D ou produzida por um conversor D/A. A resolução é o valor da tensão analógica do bit menos significativo, que é a *tensão em escala total* $\div 2^n$ para um conversor binário de n bits. A resolução é muitas vezes especificada em porcentagem da escala total.

Linearidade O desvio máximo de uma linha reta traçada entre os níveis de entrada mínimo e máximo do conversor. A linearidade pode ser expressa como uma porcentagem da escala total ou como uma fração do valor da tensão do bit menos significativo. (A linearidade de um bom conversor é $\pm 1/2$ do bit menos significativo.)

Erro de quantização A incerteza básica associada à digitalização de um sinal analógico, devida à resolução finita de um conversor A/D. Um conversor ideal tem um erro máximo de quantização de $\pm 1/2$ do bit menos significativo.

EXERCÍCIOS

1. Usando o código na Fig. 8.2, um operador registra em teclado o seguinte:

ADD 641
CAD 932
SUB 841

Quantas linhas serão perfuradas na fita de papel? Quantos orifícios serão perfurados na fita?

2. O código na Fig. 8.2(*a*) tem pelo menos uma perfuração ou orifício para cada caráter. Isto permite que o leitor detecte quando lê. Explicar o caráter de ALIMENTAÇÃO DE FITA.
3. Escrever em forma binária a primeira linha do programa no Exercício 1 usando o código na Fig. 8.3.
4. O código na Fig. 8.2 é um código de teste de paridade. O teste de paridade é ímpar ou par?
5. Se o pequeno programa no Exercício 1 for perfurado em cartões de acordo com o código na Fig. 8.6 usando procedimentos normais, quantos orifícios serão perfurados nos cartões usados? Se for cometido um engano durante o registro em teclado, será mais fácil corrigir se forem usados cartões ou fita? Explicar por quê.
6. O caráter ALIMENTAÇÃO DE FITA pode ser usado para retirar qualquer caráter no código na Fig. 8.2 exceto um. Qual o caráter e por quê? *Sugestão*: Lembre-se dos testes de paridade.
7. Geralmente, os programas são perfurados em cartões com uma palavra de instrução por cartão. A primeira linha do programa no Exercício 1 deve ir sobre um cartão. Quantos orifícios serão perfurados no primeiro cartão para o código na Fig. 8.6?
8. Em EBCDIC, quando os primeiros quatro bits forem 1, os quatro bits remanescentes representam um dígito. O código destes bits é o BCD ou é o binário direto? Justificar sua resposta.
9. Listar os grupos de código binário para cada dígito decimal no código BCD excesso de 3 no Cap. 3 e atribuir um bit de paridade para um sistema de teste de bit de paridade par para cada grupo de código. Listar os valores dos bits de paridade para o mesmo código excesso de 3 de um sistema de teste de bit de paridade ímpar.
10. Discutir quaisquer problemas que possam ser previstos na tentativa de ler caracteres oticamente e que não devem ocorrer com caracteres magnéticos. Apresentar razões que expliquem por que os bancos adotaram as leitoras magnéticas antes das leitoras óticas.
11. Cada uma das seguintes fileiras de dígitos consiste de um grupo de código no código ASCII. Um único teste de paridade foi acrescentado como o bit mais à direita em cada fileira, e uma única fileira de testes de paridade foi acrescentada na extremidade, conforme explicado no Exercício 15. Além disso, foram acrescentados erros, de modo que os dados não estão corretos no momento. Corrigir estes grupos de dados e depois converter cada caráter de sete dígitos no caráter alfanumérico que ele representa. Os testes de paridade são testes de paridade ímpar:

(a) 0 0 1 0 0 0 1 1
0 1 1 0 1 0 0 0
0 1 1 0 0 0 0 1
0 0 1 0 0 0 1 1
0 0 1 1 0 0 1 0
0 0 0 1 0 0 0 0
0 1 0 0 1 1 0 1
0 1 1 1 1 1 0 1
0 1 1 0 0 0 1 1
0 1 1 0 1 0 0 0
0 0 1 0 0 0 1 1
1 1 0 0 1 0 0 0 fileira de teste de paridade

(b) 0 1 1 0 0 1 0 0
0 1 0 0 0 1 1 0
0 1 0 1 0 1 0 1
0 0 1 0 0 0 1 1
0 0 0 1 0 0 0 0
0 0 1 0 0 0 1 0
0 0 1 0 0 0 1 1
0 1 0 0 0 1 1 0
0 1 0 1 0 1 1 1
1 0 1 0 0 0 1 1 fileira de teste de paridade

12. Usando os esquemas de detecção de erros e de correção de erros no Exercício 11, foi transmitida uma mensagem. Ela chega conforme indicado a seguir. Determinar se ocor-

reram erros e corrigir qualquer um que for achado na mensagem seguinte:

1 0 1 0 0 1 0 0
1 0 0 0 0 0 1 0
1 0 0 0 1 1 0 1
1 0 0 1 0 0 0 1
1 0 1 0 1 0 0 0
0 1 1 1 1 1 1 1

13. Quantas faixas deve ter um disco codificador similar ao da Fig. 8.22 para um conversor A/D que tenha uma precisão de 10 dígitos binários? Listar os grupos de código sucessivos para um código de distância unitária de 5 bits que conta de 0 a 31_{10}.
14. Um código 2, 4, 2, 1 para codificação de 10 dígitos decimais em 4 dígitos binários pode ser feito de modo que não mais do que duas posições mudem a cada vez que um único dígito seja aumentado de um. Escrever este código.
15. Sistemas de teste de paridade mais poderosos podem ser formados acrescentando-se colunas com o número de dígitos 1 em cada coluna escrita na extremidade com um número binário. Por exemplo, se desejarmos codificar

1 0 1 1
1 1 0 1
1 1 0 0
1 1 1 0

acrescentamos

0
0
1
0 } testes de paridade

0 1 0 0 1
0 1 1 1 0 } número de dígitos 1 em cada coluna
1 0 0 0 0

↑ ↑—— indica dois dígitos 1
└ indica quatro dígitos 1

Acrescentando isto aos dados, forma-se este bloco de dados codificados:

1 0 1 1 0
1 1 0 1 0
1 1 0 0 1
1 1 1 0 0
0 1 0 0 1
0 1 1 1 0 } dígitos de teste
1 0 0 0 0

Agora, se ocorrerem um ou mais erros na mesma coluna, eles podem ser corrigidos observando-se simplesmente que a coluna não está de acordo com o número de dígitos 1 registrado na extremidade e que há testes de paridade nas fileiras dos blocos de dados que contêm erros. Mudando simplesmente estes erros, convertemos a mensagem novamente à sua forma original. Aqui estão dois outros blocos de dados que incluem erros. Corrigir os erros nestes blocos de dados e escrever o código alfanumérico para cada conjunto de sete dígitos numa fileira à direita das fileiras. O código é aquele

da Fig. 8.10; assim, os testes de paridade são como indicado na figura, e não na coluna mais à direita.

(a)

0 0 1 0 0 0 1 1
0 1 1 0 1 0 1 0
0 1 1 0 0 0 1 1
0 0 1 0 0 0 0 1
0 0 1 1 0 0 1 0
0 0 0 1 0 0 0 0
0 1 1 1 0 1 1 0
0 1 1 1 1 0 0 1
0 1 0 0 0 1 0 1
0 1 1 1 0 1 0 1
0 0 0 1 0 1 0 0
0 1 0 0 1 1 0 1
0 1 0 1 0 0 1 1
0 0 1 0 0 0 0 0

} dígitos de teste

(b)

0 1 1 0 0 0 1 0
0 1 0 0 0 0 0 1
0 1 0 0 0 1 0 0
0 0 0 1 0 0 0 0
0 0 1 0 0 0 1 1
0 1 1 0 0 0 0 1
0 1 0 1 0 1 1 1
0 1 1 1 0 1 0 1
0 0 1 1 0 0 1 0
0 0 0 1 0 0 0 0
0 0 1 1 0 1 0 1
0 1 1 0 0 1 0 0
0 1 1 1 0 0 1 1
0 0 0 0 0 0 0 0

} dígitos de teste

16. Explicar por que o código Gray não pode ser usado se tivermos que usar uma estrutura de resistores de D/A como na Fig. 8.23. Explicar por que o código 2, 4, 2, 1 no Exercício 14 pode ser usado.
17. Se os sinais de cada uma das quatro entradas na estrutura de resistores do conversor D/A na Fig. 8.23 são das saídas 1 de biestáveis que têm níveis lógicos +8 V e 0 V (+8 V representa um 1), qual será o potencial em E_0 se os biestáveis representarem o número binário 0101? 1010? 1111?
18. Colocar erros na mensagem do Exercício 12 que a codificação nem corrigirá nem detectará.
19. Mostrar um padrão de erros que o código no Exercício 15 não corrigirá, mas detecterá, e mostrar um outro que ele nem corrigirá nem detectará.
20. Se o código ASCII na Fig. 8.10 for transmitido em série em binário, desenhar a forma-de-onda do caráter 8, admitindo que um 1 é +4 V e um 0 é −4 V.
21. Caracteres são geralmente lidos numa fita de papel perfurada à razão de uma linha de cada vez. Quando o código na Fig. 8.2(*a*) é usado, o computador estará alimentado com os bits de informação cada vez que uma linha é lida. Se o computador usado for um computador em série, os bits chegarão em paralelo e devem ser mudados para a forma em série. Carregando-se em paralelo um registro de deslocamento de sete lugares e depois deslocando-se o registro na freqüência de repetição de pulsos da máquina, os bits que representam o caráter podem ser convertidos à forma em série. Desenhar um diagrama-bloco de um registro de deslocamento com sete biestáveis, junto com as linhas de entrada necessárias para carregar o registro. (Admitir que haja sete linhas de entrada — uma para cada biestável — da leitora de fita e que uma determinada linha de entrada conterá um pulso se um orifício estiver na respectiva posição da fita.)
22. Explicar por que, nos sistemas de computadores de grande porte, os dados de saída a serem impressos são quase sempre registrados em fita magnética para impressão fora-de-linha e não impressos imediatamente.
23. Quais dos conjuntos de erros do Exercício 11 teriam sido detectados se o sistema de captura de erros no Exercício 15 tivesse sido usado? Quais desses conjuntos de erros no Exercício 15 podiam ter sido corrigidos pelos esquemas de detecção de erro e de correção de erro no Exercício 11 e quais teriam sido apenas detectados?
24. É possível inventar um código de 7 bits binários que inclua um bit de teste de paridade ímpar e que contém 70 caracteres? Justificar sua resposta.
25. No conversor A/D na Fig. 8.25, se o resistor de 40 kΩ fosse erradamente feito de 50 kΩ, quais das seguintes tensões seriam convertidas incorretamente:

(a) +10 V *(b)* +3 V
(c) +2 V *(d)* +18 V
(e) +8 V *(f)* +13 V

26. Projetar um conversor A/D de 6 bits usando o mesmo esquema mostrado na Fig. 8.25.
27. O código no Exercício 15 não é, em geral, capaz de corrigir erros duplos numa fileira ou os erros nos números de teste na extremidade, mas quase sempre detectará cada um destes. Explicar esta afirmação.
28. Discutir as relações entre exatidão, precisão e velocidade num conversor A/D.
29. Mostrar como um teletipo leria o caráter *C* quando esta tecla fosse abaixada. Desenhar a forma-de-onda de saída.
30. Qual é o código ASCII para "quem é você" (*who are you*) que é também WRU na Fig. 8.10?
31. Qual é o código EBCDIC para *D* na Fig. 8.11?
32. Observar que os caracteres no conjunto de caracteres da leitora magnética na Fig. 8.13 estão "bloqueados", não curvados. Por que isto pode ser uma boa idéia?
33. Observar a diferença num 0, num Q e num O na Fig. 8.14. Achar outras letras que tenham formas impressas similares e que foram alteradas para torná-las mais legíveis pela máquina.
34. Qual é a diferença entre um acoplador acústico e um modem?
35. Podem-se imaginar algumas razões pelas quais constituiria uma boa idéia para um computador repercutir um caráter registrado num teclado que ele deve ler em vez de ter o caráter impresso imediatamente?
36. Por que são necessários bits de partida e pelo menos 1 bit de parada para a transmissão de código de teletipo explicada por terminais?
37. Explicar a diferença entre transmissão síncrona e assíncrona de dados digitais.
38. Explicar como o modem 103 da Bell descrito transmitiria o caráter *D* em ASCII numa linha e numa direção escolhida. Como ele transmitiria na outra direção? Usar o esquema do teletipo para codificar usando bits de partida e de parada.
39. Usar resistores de 1 e de 2 kΩ e desenhar o diagrama esquemático de um conversor D/A como na Fig. 8.24.
40. Com o esquema que foi desenhado no exercício precedente, use +7 V e 0 V como os dois níveis e represente a saída para os dois valores digitais de entrada diferentes de 0.
41. Repetir o exercício precedente, mas usar resistores de 5 e 10 kΩ.
42. Projetar um conversor D/A que converta usando a técnica das aproximações sucessivas. Usar uma versão modificada da Fig. 8.25.
43. Mostrar quanto tempo levaria um conversor D/A para converter 6 bits com emprego da técnica das aproximações sucessivas. Admitir que sejam necessários 10 ms para que a estrutura de resistores de conversor D/A estabilize sua saída. Explicar mostrando como são feitas as conversões para três tensões de entrada especificadas.
44. Admitindo novamente que são necessários 10 ms para que a estrutura de conversor D/A se estabilize, mostrar como um conversor de 6 bits usa a técnica das aproximações sucessivas para as três tensões +9 V, +1 V e +7 V, admitindo uma faixa de 0 a 10 V para as entradas (0 e +10 V como níveis de tensão para as saídas do conversor de nível).
45. Mostrar como o conversor na Fig. 8.25 converte as três tensões no exercício precedente e comparar os tempos de conversão.
46. Usando a informação nos três exercícios precedentes, pode-se comparar o tempo médio de conversão de um conversor A/D como na Fig. 8.25 com um conversor por aproximações sucessivas? Supor 6 bits.
47. Mostrar como um conversor instantâneo funciona para um sistema de 3 bits que usa resistores de 10 KΩ. Admitir tensões num intervalo de 0 a 6 V, de modo que 000_2 = 0 V, 001_2 = 2 V, 100_2 = 4 V, 110_2 = 6 V etc. Desenhar as portas das saídas do comparador para os números binários.
48. Explicar como o conversor no exercício precedente converte + 0,5, + 3,2 e + 5,4 V.
49. Discutir erros de resolução, de linearidade e de quantização. Pode o erro de quantização ser menor do que o de resolução? Por que ou por que não?

9

A UNIDADE DE CONTROLE

Este capítulo descreve a seção de controle dos computadores digitais. Os capítulos precedentes descreveram técnicas pelas quais as operações aritméticas e lógicas podem ser efetuadas e a informação lida em/de vários dispositivos de memória. Para utilizar as capacidades de manipulação das velocidades e da informação das técnicas e dispositivos que foram ilustrados, é necessário pôr automaticamente em seqüência as diversas operações que ocorrem em velocidades compatíveis com as do restante da máquina. O elemento de controle deve, portanto, ser construído de um conjunto de circuitos de alta velocidade. Os elementos básicos usados no elemento de controle de um computador digital são os descritos nos Caps. 4, 5 e 7, e a maioria dos conceitos que formam a base do funcionamento do elemento de controle foi apresentada nos Caps. 3 a 6.

Os modernos computadores digitais são geralmente divididos em duas classes: o computador de propósitos especiais e o computador de propósitos gerais. Um *computador de propósitos especiais* é aquele em que a seqüência de operações é predeterminada pela construção real do conjunto de circuitos de controle. Uma máquina de propósitos especiais, portanto, tem uma seqüência fixa de operações que não pode ser mudada. Uma *máquina de propósitos gerais*, por outro lado, é aquela em que a seqüência de instruções (programa) que a máquina executa é lida na máquina através dos dispositivos de entrada, e pode ser mudada pela simples leitura num novo programa. Este tipo de máquina é também denominado computador de *programa armazenado*.

Esta divisão de computadores digitais em duas categorias não é tão exata quanto possa parecer de início. Muitos computadores, tais como os usados em sistemas de controle de mísseis e de aviões ou como uma parte de sistemas de controle de processos industriais, são de propósitos especiais porque a seqüência das operações efetuadas é montada diretamente no conjunto de circuitos de controle. Outros computadores, tais como os computadores de grande porte usados nos centros de computação com atividades científicas e comerciais, são nitidamente de propósitos gerais, uma vez que os programas para estas máquinas são sempre lidos e alterados muito freqüentemente. Por outro lado, existe uma classe de computadores em que a seqüência do programa é determinada por uma memória de leitura exclusiva (ROM), cujos conteúdos podem ser alterados, alterando assim a seqüência das operações. Por exemplo, os microprocessadores são freqüentemente usados com um programa numa ROM de semicondutores. A ROM pode ser substituída, alterando assim o programa; e algumas ROM são reprogramáveis quanto a que seus conteúdos podem ser alterados sempre que for desejado. As técnicas empregadas aqui são relativamente ineficientes, mas, visualizados externamente, os computadores parecem operar na área definida como de propósitos gerais, e os fabricantes de microprocessadores assinalam que, muito freqüentemente, estes computadores são programados para operar como computadores de propósitos gerais. A classificação neste ponto é puramente arbitrária, e as diferenças entre os dois tipos de computadores são relativas uma vez que ambos estão executando funções similares.

Deve ser notado que a maioria dos computadores classificados como de propósitos especiais é capaz de alterar sua seqüência de operações, dependendo dos dados de entrada. Assim, eles têm *desvios condicionais* que dependem dos resultados numéricos, mas a seqüência das operações é alterada em que o computador salta para uma outra posição

no programa. A seqüência básica das operações em cada seção do programa não muda, apenas o percurso que o computador segue através do programa é alterado.

A *unidade de controle* pode ser definida como "aquelas partes de um computador digital que efetuam o transporte das instruções na seqüência adequada, a interpretação de cada instrução e a aplicação dos comandos adequados ao elemento aritmético e a outros circuitos de acordo com esta interpretação."*

A função do conjunto de circuitos de controle num computador de propósitos gerais é, portanto, interpretar as palavras da instrução e depois colocar em seqüência os sinais necessários àquelas seções da máquina que farão com que a máquina execute a instrução. Os capítulos anteriores mostraram como a aplicação da seqüência correta de sinais de controle ao conjunto de circuitos lógicos no elemento aritmético permite ao computador efetuar operações aritméticas, e como as palavras binárias podem ser armazenadas e posteriormente lidas em diversos tipos de dispositivos de memória. Para a máquina funcionar, a operação das seções da máquina deve ser orientada, e o conjunto de circuitos de controle executa aquela função.

9.1 CONSTRUÇÃO DA PALAVRA DA INSTRUÇÃO

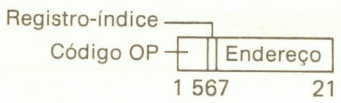
Definimos uma palavra de computador como um conjunto ordenado de caracteres manipulados como um grupo. Além disso, a palavra de computador é considerada como sendo uma unidade básica de informação numa máquina.† Basicamente, todas as palavras consistem de um conjunto de dígitos binários, e o significado dos dígitos depende de diversos fatores diferentes. Por exemplo, os bits 01000100 podem representar o número decimal 68 num computador de binário puro, e o número decimal 44 num computador de BCD que usa um código 8421. Assim, o significado de um conjunto de dígitos é, às vezes, determinado por seu tratamento ou manuseio. Além disso, são possíveis outras interpretações, pois as palavras de instrução são armazenadas exatamente como as palavras de dados, e os dígitos podem representar uma instrução para o computador. Já que os locais da memória podem armazenar ou palavras de instrução ou palavras de dados, os programadores e operadores de sistema devem ver se as palavras da instrução são usadas para determinar a seqüência de operações que o computador efetua e se significados razoáveis são atribuídos às palavras de dados.

Se admitirmos que cada local da memória pode conter uma única palavra de instrução, um computador iniciará com a palavra armazenada em algum endereço especificado, interpretará os conteúdos deste local como uma instrução e depois continuará tirando as palavras da instrução dos locais da memória em ordem, a menos que seja encontrada uma intrução de DESVIO ou de PARADA. Os dados a serem usados nos cálculos serão armazenados numa outra parte da memória. Uma vez que o computador pode armazenar ou instruções ou dados nos mesmos registros de armazenamento, resulta considerável flexibilidade de operação. Por exemplo, se o elemento de memória do computador pode armazenar apenas 8000 palavras, um programa pode conter 5000 instruções, deixando 3000 locais para o armazenamento de dados, enquanto que um outro programa pode usar apenas 1000 palavras de instrução, deixando 7000 locais para os dados. O usuário do computador está, portanto, livre para alocar a memória quando necessário, e os problemas com muitas instruções e poucos números, ou com muitos dados e poucas instruções, podem ser manipulados pelo mesmo computador.

Deve ser observado que os computadores que possuem tanto uma memória interna de alta velocidade quanto uma memória auxiliar de baixa velocidade usarão a memória interna de alta velocidade para as palavras de instrução e nos resultados intermediários, e os dispositivos de armazenamento de velocidade mais baixa para quaisquer dados que não possam ser acomodados na memória de alta velocidade. Por exemplo, se um computador tiver uma memória de núcleos magnéticos de alta velocidade mais alguns dispositivos de fita magnética, as intruções do programa podem ser armazenadas em armazenamento de núcleo magnético, e os dados a serem usados, em armazenamento de fita. Então, quando as porções dos dados são lidas na memória de núcleos e os cálculos são efetuados, mais dados são lidos da fita para o núcleo e mais cálculos são efetuados, até que todos os dados tenham sido processados. A máquina tira suas palavras de instrução apenas da memória

*Instituto de Engenheiros Eletrônicos e Eletricistas, *Normas do IEEE sobre Computadores Eletrônicos*, Definição de Termos.

†Alguns dos novos computadores manipularão palavras de diferentes comprimentos, em cujo caso a "palavra básica" é geralmente considerada como sendo a palavra normalmente manipulada como uma unidade. Alguns computadores têm comprimentos de palavra completamente variáveis.



Raytheon
7180

Fig. 9.1 Formato da palavra de instrução para o Raytheon 7180.

interna; se as instruções forem armazenadas em fita, elas devem ser transferidas para dentro da memória interna antes de serem executadas pelo computador.

Uma palavra de instrução numa máquina digital geralmente consiste de várias seções. O número de divisões na palavra depende do tipo de computador. Devido ao seu amplo uso e simplicidade, descreveremos as chamadas *palavras de instrução de um só endereço* nesta seção e nas seguintes, deixando os formatos mais complicados para as seções posteriores. A palavra de instrução de endereço único é amplamente usada em microcomputadores e minicomputadores bem como em muitos dos computadores de grande porte, e serve como uma boa base para introduzir as operações da unidade de controle. Basicamente, cada palavra de instrução de um só endereço contém duas seções: o *código de operação* (código OP), que define a intrução a ser executada, tal como adição, subtração etc.; e a *parte de endereço*, que contém o local do número a ser adicionado ou subtraído ou usado de outro modo (o operando).

Se forem usados 5 dígitos binários da palavra para especificar qual instrução deve ser executada, deve haver algum sistema de codificação para indicar cada uma das várias instruções. Já que 5 bits foram alocados para designar que instrução deve ser executada, 2^5, ou 32, instruções diferentes podem ser usadas, e um número binário diferente deve ser associado a cada uma. Cada tipo de instrução terá então um código de OP específico que consiste do conjunto de dígitos binários que a máquina deve interpretar.

A máquina Raytheon 7180 é um exemplo de máquina binária de endereço único tendo 32 instruções básicas e um código de OP de 5 dígitos binários compactado numa palavra de 21 dígitos binários. Quinze dígitos binários são usados para o endereço do operando na memória, e um único dígito é usado para indicar se um registro-índice deve ou não ser usado. (Este assunto será discutido mais tarde.) O formato básico da palavra de instrução para esse computador é tal que o código de operação ocupa os dígitos de 1 a 5; o dígito 6 é usado para indicar se uma instrução particular está indexada ou não; e os dígitos 7a 21 são usados para designar o endereço na memória da palavra a ser usada. (Ver Fig. 9.1.)

Seis instruções típicas nesta máquina são mostradas no Quadro 9.1, e o efeito de cada instrução na máquina é mostrado no mesmo quadro. Se, por exemplo, a máquina lê uma palavra de instrução binária da memória para a qual os cinco dígitos do código de OP são 00100, a máquina multiplicará o número que está no acumulador quando a palavra de instrução for lida pelo número localizado no endereço fornecido pelos dígitos binários de 6 a 20 na palavra de instrução.

O DDP-24 é fabricado pela Minneapolis-Honeywell e é um computador de porte médio e de endereço único. O conjunto completo de instruções para o DDP-24 é mostrado no Quadro 9.2. Neste quadro, *A* se refere ao acumulador, *B* ao registro usado para armazenar os dígitos menos significativos dos produtos etc., e a EA é a porção do endereço de uma palavra de instrução do computador. São fornecidos os códigos de OP octais, bem como os códigos mnemônicos. Para explicar uma instrução típica, a instrução de adição tem um código OP octal de 10; assim, o código binário é 001000. O código mnemônico é ADD, e a notação (A) + (EA) → (A) diz que a soma do acumulador com a palavra na memória no endereço fornecido por EA (que é o local fornecido pela parte de endereço da palavra de instrução) é colocada no acumulador.*

Como um outro exemplo, JMP com um código OP binário de 111100 faz com que a máquina tire a próxima palavra de instrução do endereço fornecido pela porção de endereço da palavra de instrução JMP. Observemos que a instrução de multiplicação multiplica uma palavra na memória pelo registro *B*, não o acumulador, que era uma característica de muitas das máquinas IBM mais antigas.

*Os parênteses são usados para indicar "o conteúdo de," portanto, a notação (A) significa "o conteúdo do acumulador" e a notação (A) → (B) significa "o conteúdo do registro *A* é transferido para dentro do registro *B*." [Isto é muitas vezes escrito (A) → B, mas a Honeywell usa (A) → (B).]

QUADRO 9.1 INSTRUÇÕES E CÓDIGO DE OPERAÇÃO PARA O COMPUTADOR RAYTHEON 7180

Código OP (código mnemônico)	Código OP em binário	Significado da instrução
ADD	00010	Soma o número no endereço na memória, dado pela parte de endereço da palavra de instrução, ao número no acumulador e armazena aquele número no acumulador.
MUL	00100	Multiplica o número no endereço na memória, dado pela parte de endereço da palavra de instrução, pelo número no acumulador e armazena os dígitos mais significativos e o sinal no acumulador e os dígitos menos significativos num registro *MQ*.
DVD	00101	O número combinado formado pelo acumulador e pelo registro *MQ* é dividido pelo número no endereço dado na parte de endereço da palavra de instrução. O quociente substitui o número originalmente no *MQ*, e o resto substitui o número originalmente no acumulador.
STO	11000	O conteúdo do acumulador é armazenado no local dado pela parte de endereço da palavra de instrução.
AND	01111	Cada bit no acumulador é introduzido numa porta E, com o bit correspondente da palavra localizado no endereço na memória dado pela parte de endereço da palavra da instrução. Os bits resultantes substituem o conteúdo original do acumulador.
CLA	00001	Carrega o número no endereço na memória, dado pela parte de endereço da palavra de instrução, para dentro do acumulador.

Índices tais como em $A_{2\sim24}$ constituem dispositivos de notação muito úteis. Eles indicam os bits do registro envolvido, de modo que, por exemplo, STD no Quadro 9.2 faz com que somente os bits na 10.ª até a 24.ª posições no acumulador sejam armazenados no local fornecido pelo endereço (EA) na palavra da instrução, e estes entram na 10.ª até a 24.ª posições naquela palavra da memória.

9.2 ORGANIZAÇÃO DO CICLO DE INSTRUÇÃO E DO CICLO DE EXECUÇÃO DOS REGISTROS DE CONTROLE

Um computador digital prossegue através da execução de um programa com um ritmo básico ou padrão em sua seqüência de operação que é produzida pela necessidade de extrair tanto as instruções quanto os operandos da mesma memória. Examinemos o tipo de computador de endereço único, tendo em mente que as máquinas de dois endereços, bem como as de três e de quatro endereços, operam essencialmente de maneira idêntica e requerem os mesmos registros de controle, embora o padrão da execução de uma instrução difira de acordo com o número de operandos na máquina particular e com o formato da palavra de instrução.

O padrão básico ou seqüência de operações para a maioria das instruções numa máquina digital do tipo de um só endereço consiste de uma alternância de um período de

QUADRO 9.2 CÓDIGOS DE OPERAÇÃO PARA COMPUTADOR DDP-24

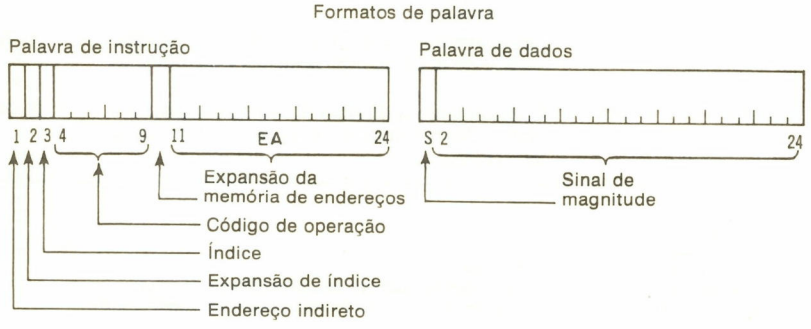
	Código OP					
	Mnemônico	Octal	Função	X	I	O'F
Carga & Armazenamento	CRA	60	$O \rightarrow (A)$			
	IAB	57	(A) (B)			
	LDA	24	$(EA) \rightarrow (A)$	X	X	
	LDB	23	$(EA) \rightarrow (B)$	X	X	
	STA	05	$(A) \rightarrow (EA)$	X	X	
	STB	03	$(B) \rightarrow (EA)$	X	X	
	STC	04	$(A)_{1-9} \rightarrow (EA)_{1-9}$	X	X	
	STD	06	$(A)_{10-24} \rightarrow (EA)_{10-24}$	X	X	
	TAB	55	$(A) \rightarrow (B)$			
Aritmética	ADD	10	$(A) + (EA) \rightarrow (A)$	X	X	X
	ADM	20	$(A) + \|(EA)\| \rightarrow (A)$	X	X	X
	BCD*	36	(EA) BCD $\rightarrow$ (A) Binário	X	X	
	BIN*	37	(EA) Binário $\rightarrow$ (A) BCD	X	X	(X)
	DIV	35	(A, B)/(EA) $\rightarrow$ (quociente para B, resto para A)	X	X	(X)
	MPY	34	$(B) \times (EA) \rightarrow (A, B)$			X
	RND	62	$(A) + 1 \rightarrow (A)$, se $(B)_2 = 1$			
	SBM	21	$(A) - \|(EA)\| \rightarrow (A)$	X	X	X
	SUB	11	$(A) - (EA) \rightarrow (A)$	X	X	X
Lógica	ANA	15	$(A) \cap (EA) \rightarrow (A)$	X	X	
	ERA	17	$(A) \oplus (EA) \rightarrow (A)$	X	X	
	ORA	16	$(A) \cup (EA) \rightarrow (A)$	X	X	
Deslocamento	ALS**	41	Deslocamento à esquerda $(A)_{2-24}$, posições especificadas por $(EA)_{19-24}$	X	X	
	ARS**	40	Deslocamento à direita $(A)_{2-24}$, posições especificadas por $(EA)_{19-24}$	X	X	
	LGL**	47	Deslocamento à esquerda $(A)_{1-24}$, posições especificadas por $(EA)_{19-24}$	X	X	
	LLR**	43	Rotação à esquerda $(A, B)_{1-24}$, posições especificadas por $(EA)_{19-24}$	X	X	
	LLS**	45	Deslocamento à esquerda $(A, B)_{2-24}$, posições especificadas por $(EA)_{19-24}$	X	X	
	LRR**	42	Rotação à direita $(A, B)_{1-24}$, posições especificadas por $(EA)_{19-24}$	X	X	
	LRS**	44	Deslocamento à direita $(A, B)_{2-24}$, posições especificadas por $(EA)_{19-24}$	X	X	
	NRM	46	Deslocamento à esquerda $(A, B)_{2-24}$ até $(A_2) = 1$, Ver Manual			
	SCL**	65	Deslocamento à esquerda $(A, B)_{2-24}$ e registro-índice de decremento, posições especificadas por $(EA)_{19-24}$	X	X	
	SCR**	64	Deslocamento à direita $(A, B)_{2-24}$ e registro-índice de incremento, posições especificadas por $(EA)_{19-24}$	X	X	

X = indexável
I = indiretamente endereçável
O'F = possível estouro
(X) = possível dividir impróprio
$\cap$ = símbolo para E
$\cup$ = símbolo para OU

*Opcional.
**Se endereço indireto não especificado (I = O), porção de endereço de instrução é operando efetivo.

QUADRO 9.2 (CONT.) CÓDIGOS DE OPERAÇÃO PARA COMPUTADOR DDP-24

	Código OP			
	Mnemônico	Octal	Função	X I O'F
Salto	JMP	74	Saltar para EA	X X
	JOF	73	Saltar para EA, se indicador de estouro ajustado	X X
	JPL	70	Saltar para EA, se sinal (A) = 0	X X
	JRT	25	Saltar para local especificado por $(EA)_{11-24}$, rearmazenar interrupção	X X
	JST	27	Saltar para EA + 1 e armazenar local em (EA)	X X
	JZE	71	Saltar para EA, se (A) = 0	X X
	SKG	12	Saltar instrução seguinte, se (A) > (EA)	X X
	SKQ	13	Saltar instrução seguinte, se (A) ≥ (EA)	X X
Índice	ADX**	54	$(X) + (EA)_{11-24} \rightarrow (X)$	X X
	IRX	67	$(EA)_{11-24} + 1 \rightarrow (EA)_{11-24}$ e (X)	X X
	JIX	72	Saltar para EA, se (X) ≠ 0	X X
	JXI	75	(X) + 1 → (X), saltar para EA, se resultante (X) = 0	X X
	LDX**	56	$(EA)_{11-24} \rightarrow (X)$	X X
	STX	66	$(X) \rightarrow (EA)_{11-24}$	X X
	TAX	63	$(A)_{11-24} \rightarrow (X)_{11-24}$	
Entrada/Saída	DMB	32	Listagem da memória começando em EA	X X
	FMB	31	Carga da memória começando em EA	X X
	INA**	52	Entrada → (A), de acordo com a máscara em (EA)	X X
	INM	07	Entrada → (EA)	X X
	ITC**	51	Inibir ou ativar interrupção, de acordo com a máscara em (EA)	X X
	OCP**	53	Selecionar I/O, de acordo com a máscara em (EA)	X X
	OTA**	50	(A) → Saída de acordo com a máscara em (EA)	X X
	OTM	22	(EA) → Saída	X X
	SKS**	61	Saltar instrução seguinte, se linha de leitura não ajustada linha de leitura especificada por $(EA)_{11-24}$	X X
Controle	HLT	00	Parar operação de computador até o início de botão pressionado	
	NOP	77	Não executar nenhuma operação	
	XEC	02	Executar instrução em EA	X X



tempo chamado *ciclo de instrução*, seguida de um período de tempo chamado *ciclo de execução*. Durante o ciclo de instrução, uma palavra de instrução é obtida da memória e interpretada, e à memória é fornecido o endereço do operando a ser usado. Durante o ciclo de execução, a memória obtém o operando a ser usado (por exemplo, o multiplicador se a instrução for uma multiplicação, ou o segundo somando se a instrução for uma adição), e a operação chamada pela palavra de instrução é então efetuada neste operando.

Quase todas as máquinas que estão sendo fabricadas agora usam ou uma memória de núcleos ou uma memória de CI para o armazenamento tanto das palavras de instrução quanto dos operandos ou dados. Não haverá nenhum prejuízo se admitirmos que temos uma memória de núcleos, já que a maioria dos tipos de memória opera essencialmente com as mesmas características quando visualizada externamente.

O tempo do ciclo de uma memória de núcleos é fixo, e, uma vez que dizemos à memória que desejamos ler dela ou gravar nela, decorrerá um certo período de tempo antes que possamos instruir a memória de que estamos novamente prontos para ler ou gravar. Se estivermos lendo da memória, a palavra selecionada será fornecida num tempo curto após ter sido fornecido à memória o endereço da palavra a ser lida e a instrução a ser lida. Durante a porção restante do ciclo da memória, a memória regravará esta palavra no local do qual ela foi lida.

Se a memória deve ser gravada, a palavra a ser gravada bem como o endereço no qual desejamos gravá-la devem ser fornecidos à memória. Um sinal de GRAVAR também deve ser emitido, dizendo à memória para gravar esta palavra no local ou endereço que fornecemos. Conforme discutido no Cap. 7, o endereço em que gravamos ou lemos na memória é fornecido por meio de um registro de endereços da memória, e a palavra a ser gravada na memória é fornecida a um registro chamado registro de memória intermediária; quando lemos da memória, a palavra também é fornecida a um registro de memória intermediária.

Durante o ciclo de instrução, a palavra de instrução é transferida pela memória para dentro do registro de memória intermediária. Para obter esta palavra, devemos dizer à memória para ler e fornecer o endereço para leitura. Durante o ciclo de instrução, a palavra de instrução que foi lida no registro de memória intermediária é interpretada, e o endereço do operando a ser usado é fornecido ao registro de endereços da memória. Para muitas instruções, isto será uma parte da palavra de instrução, cuja parte de endereço foi lida na memória durante o ciclo de instrução. Durante o tempo de execução ou ciclo de execução, é obtido um operando da memória ou gravado na memória, dependendo da palavra de instrução que foi interpretada durante o período de tempo de instrução anterior.

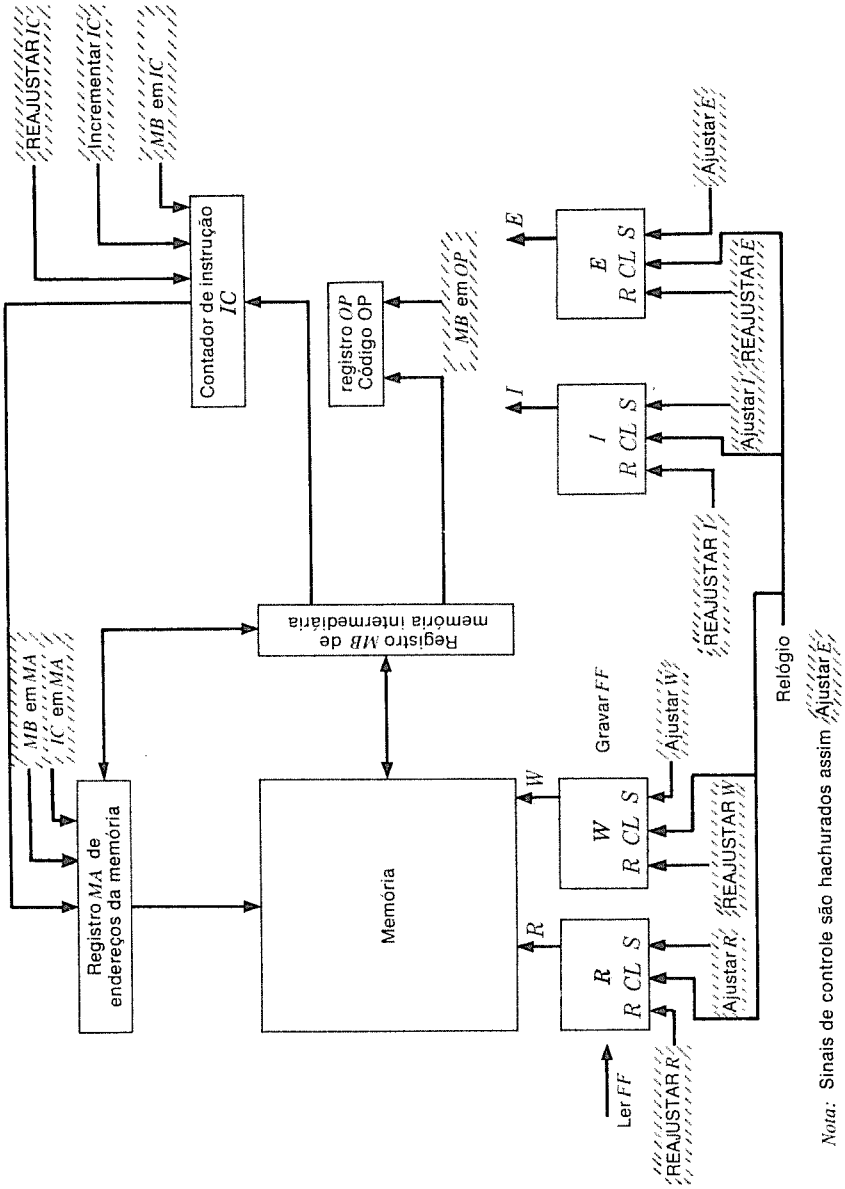
Se a instrução que está sendo interpretada for uma instrução de ADD (SOMAR), o local do segundo operando é fornecido na parte de endereço da palavra de instrução, e este endereço deve ser fornecido ao registro de endereços da memória. A memória então obtém a palavra desejada da memória e a coloca no registro de memória intermediária. O computador deve somar esta palavra à palavra que já está no acumulador. Posteriormente, o computador deve fornecer à memória o endereço da palavra de instrução seguinte a ser usada e comandar a memória para ler esta palavra.

Observemos que a máquina se reveza entre os ciclos de instrução e os ciclos de execução. Observemos também que durante um ciclo de execução devemos armazenar, em algum lugar de nosso conjunto de circuitos de controle, o código OP da palavra de instrução que foi lida na memória, o endereço do operando a ser usado (que era uma parte da palavra de instrução lida na memória) e também o endereço da palavra de instrução seguinte a ser lido na memória e usado.

Em conseqüência, há diversos registros que são básicos para quase todos os computadores digitais. Estes são mostrados na Fig. 9.2 e são descritos como segue:

1. *O contador de instrução* Este é um contador do mesmo comprimento que a seção de endereço da palavra de instrução. O contador pode ser ou reajustado ou incrementado. Um diagrama lógico típico do contador de instrução pode consistir do contador mostrado na Fig. 4.10(*a*), tendo uma linha de REAJUSTE e uma linha de ativar ou incrementar. Este contador guarda a trilha das instruções a serem usadas no programa, de modo que normalmente, durante cada tempo de instrução, o contador será incrementado de 1, que fornecerá o local da próxima palavra de instrução a ser usada no programa. Se, no entanto, a instrução for uma instrução de SALTO ou DESVIO, podemos desejar colocar o conteúdo do registro *B* neste contador, e a linha de CI EM MB faz isto. O contador pode ser reajustado a 0 quando um programa for iniciado.*

*A maioria das máquinas torna possível carregar um endereço selecionado no contador de operação e, desse modo, iniciar a máquina naquele endereço selecionado.



Nota: Sinais de controle são hachurados assim

Fig. 9.2 Registros de controle.

Também deve ser possível transferir o conteúdo deste contador para dentro do registro de endereços da memória, que é usado para localizar uma palavra na memória. Normalmente, o contador de instrução será aumentado de 1 durante a execução de cada instrução, e o conteúdo do contador será transferido para o registro de endereços da memória no começo de cada tempo de instrução.

2. *O registro do código OP* Quando uma palavra de instrução for lida na memória, a seção do código OP desta palavra deve ser armazenada a fim de determinar que instrução deve ser executada. Se o computador tiver um código OP com um comprimento de 5 dígitos binários, o registro de operação terá um comprimento de 5 dígitos binários e conterá a parte do código OP da palavra de instrução que é lida na memória. Devemos, portanto, ser capazes de transferir uma seção do registro de memória intermediária para dentro do registro do código OP durante o período de tempo de instrução.
3. *O registro de endereços da memória* Este registro contém o local da palavra na memória a ser lido ou o local a ser gravado.
4. *Biestável tipo R* Quando este biestável for ativado, ele diz à memória para ler uma palavra. (O biestável pode ser desativado logo depois disso, pois ele não necessita estar funcionando durante todo o ciclo da memória.)
5. *Biestável tipo W* A ativação deste biestável diz à memória para gravar a palavra localizada no registro de memória intermediária, no local fornecido pelo registro de endereços da memória.
6. *Biestável tipo I* Quando este biestável está ativado, o computador está num ciclo de instrução.
7. *Biestável tipo E* Quando este biestável está ativado, a máquina está num ciclo de execução.

9.3 SEQÜÊNCIA DE OPERAÇÃO DOS REGISTROS DE CONTROLE

Consideremos ainda a construção do conjunto de circuitos de controle de um computador digital, usando novamente o diagrama-bloco dos registros de controle, memória, registro de endereços da memória e registro de memória intermediária mostrados na Fig. 9.2.

Os sinais de controle necessários à operação desta pequena máquina de um só endereço são também mostrados no diagrama e descritos como segue: Há uma linha de CI de REAJUSTE que limpará o contador de instrução para 0. (Este é muitas vezes conectado a um botão de pressão que limpa o contador quando o programa deve ser iniciado.) Há um sinal de controle de CI EM MB que faz com que o conteúdo de registro de memória intermediária seja transferido para dentro do contador de instrução e há um sinal de controle de CI de INCREMENTO que faz com que o contador de instrução seja incrementado de 1. Um outro sinal de controle é o OP EM MB, que transfere os cinco primeiros dígitos do registro de memória intermediária que contém o código de operação de uma palavra de instrução para dentro dos cinco biestáveis no registro de operação. O registro de endereços da memória tem dois sinais de controle. O sinal de controle MA EM CI faz com que o conteúdo do contador de instrução seja transferido para dentro do registro de endereços da memória, e o sinal de controle MA EM MB faz com que os últimos 16 dígitos do registro de memória intermediária (que constituem a parte de endereço de uma palavra de instrução) sejam transferidos para dentro do registro de endereços da memória.

Durante cada ciclo de instrução da máquina devemos, primeiro, ativar o biestável de LEITURA e, ao mesmo tempo (ou antes), transferir o conteúdo do contador de instrução para dentro do registro de endereços da memória. A memória lerá agora uma palavra de instrução dentro do registro de memória intermediária, depois de cujo tempo podemos ativar a linha de OP EM MB, transferindo a seção de código OP da palavra de instrução para dentro do registro de código OP. As ações seguintes que a máquina desencadeará dependerão agora do conteúdo do registro do código OP.

9.4 CONTROLE DAS OPERAÇÕES ARITMÉTICAS

Consideremos o problema de orientação do elemento aritmético quando ele executa uma palavra de instrução. Acrescentemos um acumulador e um registro *B* aos registros na Fig. 9.2, formando assim o diagrama-bloco mostrado na Fig. 9.3. São necessários mais cinco sinais de controle para executar instruções tais como somar, subtrair, limpar e somar e armazenar:

1. REAJUSTAR ACC Este sinal ajusta todos os biestáveis no acumulador em 0.
2. SOMAR Este sinal faz com que o registro *B* seja adicionado ao acumulador, e a soma transferida para dentro do acumulador.

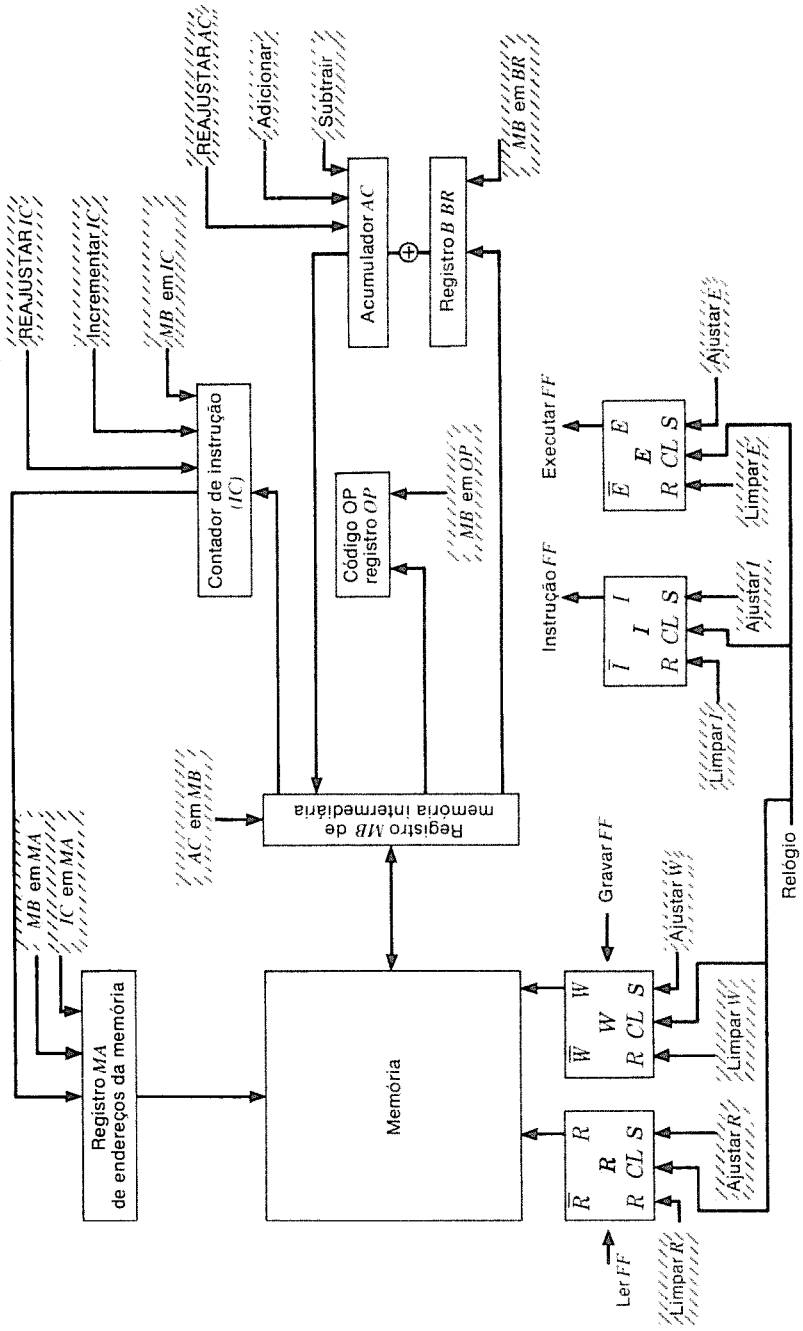


Fig. 9.3 Registros de controle e registros aritméticos.

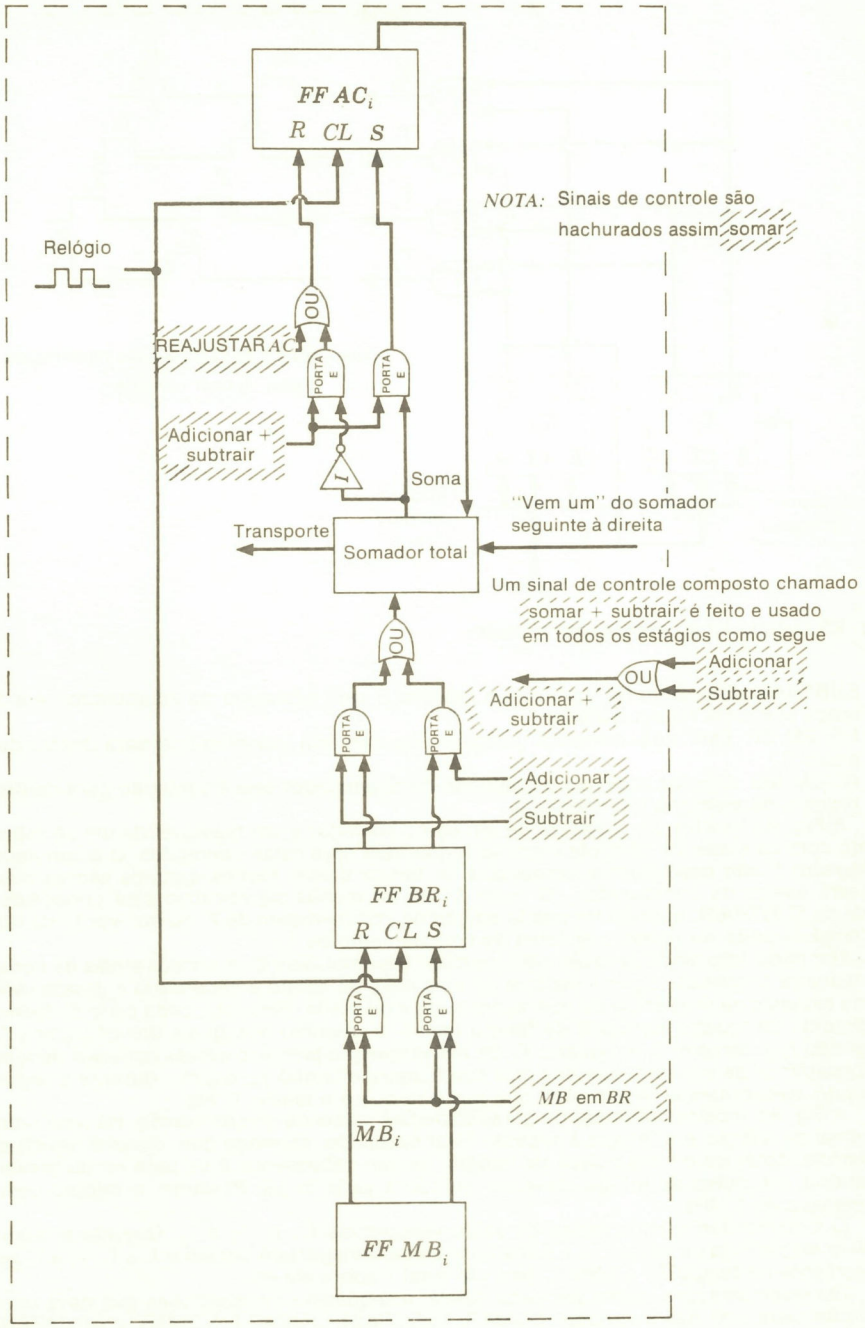


Fig. 9.4 Biestável do acumulador e biestável do registro *B* com sinais de controle.

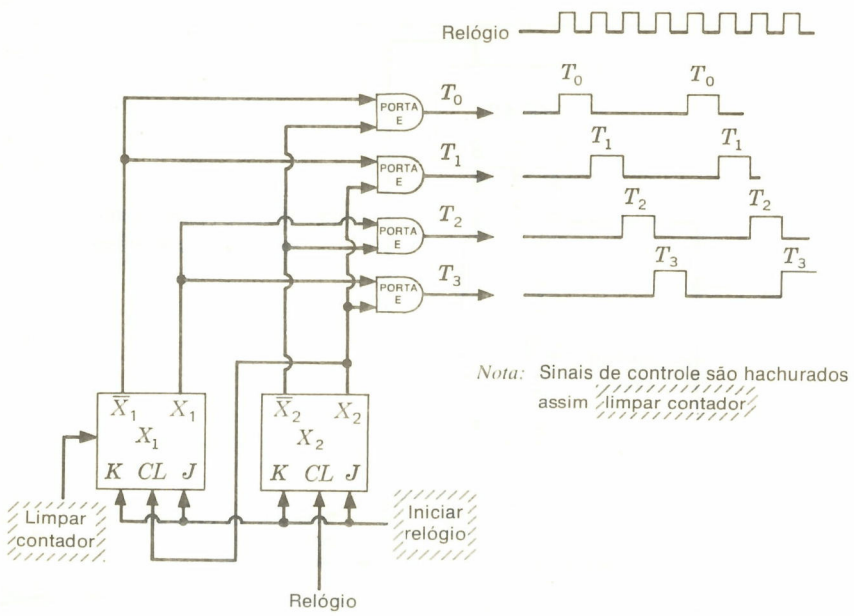


Fig. 9.5 Distribuidor do sinal de sincronização.

3. SUBTRAIR Este sinal faz com que o registro *B* seja subtraído do acumulador, e a diferença colocada no acumulador.
4. MB EM BR Este sinal transfere o registro da memória intermediária para dentro do registro *B*.
5. AC EM MB Este faz com que o conteúdo do acumulador seja transferido para dentro do registro da memória intermediária.

A Fig. 9.4 mostra um biestável de um só acumulador e um biestável de um só registro, junto com os sinais de controle e portas requeridos para estas operações. O acumulador e o registro *B* são basicamente compostos de tantos destes blocos quantos são os bits na palavra básica do computador. (O "vem 1" do bit menos significativo está conectado ao sinal de SUBTRAIR, quando for usada adição de complemento de 2, ou ao "vai 1" do dígito de sinal, quando for usado o sistema de complemento de 1.)

Um outro fato ainda deve ser considerado. Devemos distribuir nossos sinais de controle de maneira ordenada; alguma espécie de uma base de tempo é necessária e deverá indicar onde estamos na seqüência de operações a ser executada. Para isto, cada ciclo da memória é dividido em quatro períodos de tempo iguais, o primeiro dos quais designaremos T_0, o segundo T_1, o terceiro T_2 e o quarto T_3. Se estivermos no primeiro destes períodos de tempo, necessitamos de um sinal que nos dirá que é agora o tempo T_0; depois, durante o segundo período, necessitamos de um sinal que nos dirá que é o tempo T_1 etc.

A Fig. 9.5 mostra um modo de geração de tais sinais de sincronização. Há uma entrada de sinal de relógio, e o relógio é suposto estar passando, de modo que, durante um ciclo da memória, obtemos quatro pulsos de relógio. Se são necessários 2 μs para ler ou gravar na memória, um pulso de relógio deve ser gerado a cada $^1/_2$ μs. Portanto, o relógio passará numa taxa de 2 MHz.

O circuito tem quatro linhas de saída designadas T_0, T_1, T_2 e T_3. Quando a máquina estiver no período de tempo T_0, a linha T_0 de saída transportará um sinal 1, e T_1, T_2 e T_3 serão 0; somente no tempo T_1, a linha T_1 terá um sinal 1 sobre ela etc.

Escrevamos agora, numa pequena tabela, a seqüência de operações que deve ocorrer durante cada uma das instruções SOMAR, SUBTRAIR, LIMPAR E SOMAR, e ARMAZENAR. Observemos que, quando o biestável *I* do ciclo de instrução estiver ligado, as operações durante os tempos T_0 e T_1 são sempre as mesmas. No Quadro 9.3, o sinal de controle a ser ativado ou feito um 1 está listado à esquerda, e o que o sinal faz está listado à direita.

Desta tabela de operações, é possível projetar a seção de controle desta pequena má-

QUADRO 9.3 COLOCAÇÃO EM SEQÜÊNCIA DOS SINAIS DE CONTROLE

Somar	Lista de sinais de controle a serem ligados	Comentários
I e T_0	AJUSTAR R	Diz à memória para ler palavra de instrução.
I e T_1	MB EM OP, REAJUSTAR R	Transfere parte do código OP da palavra de instrução para dentro do registro do código OP; desliga o biestável de LEITURA.
I e T_2	INCREMENTAR IC	Adiciona um ao contador de instrução, preparando para a instrução seguinte.
I e T_3	MB EM MA, REAJUSTAR I, AJUSTAR E	Transfere a parte de endereço da palavra de instrução (que está no registro de memória intermediária) para dentro do registro de endereços da memória. Coloca o computador no ciclo de execução.
E e T_0	AJUSTAR R	Liga o biestável de LEITURA, dizendo à memória para ler uma palavra.
E e T_1	MB EM BR, REAJUSTAR R	Transfere o conteúdo do registro de memória intermediária para dentro do registro B. Uma vez que o registro de memória intermediária contém agora o que foi lido da memória, o subtraendo é transferido para dentro do registro B; também desliga o biestável de LEITURA.
E e T_2	SOMAR	O conteúdo do registro B é subtraído do acumulador e a diferença colocada no acumulador.
E e T_3	IC EM MA, AJUSTAR I, REAJUSTAR E	O conteúdo do contador de instrução é transferido para dentro do registro de endereços da memória, dando o local da próxima palavra de instrução para a memória. O biestável do ciclo de instrução é ligado e o biestável do ciclo de execução é desligado.

Limpar e somar	Lista de sinais de controle a serem ligados	Comentários
I e T_0	AJUSTAR R	Diz à memória para ler palavra de instrução.
I e T_1	MB EM OP, REAJUSTAR R	Transfere parte do código OP da palavra de instrução para dentro do registro do código OP; desliga o biestável de LEITURA.
I e T_2	INCREMENTAR IC	Adiciona um ao contador de instrução, preparando para a instrução seguinte.
I e T_3	MB EM MA, REAJUSTAR I, AJUSTAR E	Transfere a parte de endereço da palavra de instrução (que está no registro de memória intermediária) para dentro do registro de endereços da memória.

QUADRO 9.3 (CONT.) COLOCAÇÃO EM SEQÜÊNCIA DOS SINAIS DE CONTROLE

Limpar e somar	Lista de sinais de controle a serem ligados	Comentários
E e T_0	AJUSTAR R	Liga o biestável de LEITURA, dizendo à memória para ler uma palavra.
E e T_1	MB EM BR, REAJUSTAR AC REAJUSTAR R	Transfere o registro de memória intermediária para dentro do registro B e também limpa o acumulador, e assim, se o registro B é agora adicionado ao acumulador, o acumulador conterá a palavra lida na memória.
E e T_2	SOMAR	Os conteúdos do acumulador são adicionados ao registro B e a soma colocada no acumulador.
E e T_3	IC EM MA, AJUSTAR I, REAJUSTAR E	O conteúdo do contador de instrução é transferido para dentro do registro de endereços da memória, dando o local da próxima palavra de instrução para a memória. O biestável do ciclo de instrução é ligado e o biestável do ciclo de execução é desligado.

Subtrair	Lista de sinais de controle a serem ligados	Comentários
I e T_0	AJUSTAR R	Diz à memória para ler palavra de instrução.
I e T_1	MB EM OP, REAJUSTAR R	Transfere parte do código OP da palavra de instrução para dentro do registro do código OP; desliga o biestável de LEITURA.
I e T_2	INCREMENTAR IC	Adiciona um ao contador de instrução, preparando para a instrução seguinte.
I e T_3	MB EM MA, REAJUSTAR I, AJUSTAR E	Transfere a parte de endereço da palavra de instrução (que está no registro de memória intermediária) para dentro do registro de endereços da memória. Coloca o computador no ciclo de execução.
E e T_0	AJUSTAR R	Liga o biestável de LEITURA, dizendo à memória para ler uma palavra.
E e T_1	MB EM BR, REAJUSTAR R	Transfere o conteúdo do registro de memória intermediária para dentro do registro B. Uma vez que o registro de memória intermediária contém agora o que foi lido da memória, o subtraendo é transferido para dentro do registro B; também desliga o biestável de LEITURA.
E e T_2	SUB	O conteúdo do registro B é subtraído do acumulador e a diferença colocada no acumulador.

QUADRO 9.3 (CONT.) COLOCAÇÃO EM SEQÜÊNCIA DOS SINAIS DE CONTROLE

Subtrair	Lista de sinais de controle a serem ligados	Comentários
E e T_3	IC EM MA, AJUSTAR I, REAJUSTAR E	O conteúdo do contador de instrução é transferido para dentro do registro de endereços da memória, dando o local da próxima palavra de instrução para a memória. O biestável do ciclo de instrução é ligado e o biestável do ciclo de execução é desligado.

Armazenar	Lista de sinais de controle a serem ligados	Comentários
I e T_0	AJUSTAR R	Diz à memória para ler palavra de instrução.
I e T_1	MB EM OP, REAJUSTAR R	Transfere parte do código OP da palavra de instrução para dentro do registro do código OP; desliga o biestável de LEITURA.
I e T_2	INCREMENTAR IC	Adiciona um ao contador de instrução, preparando para a instrução seguinte.
I e T_3	MB EM MA, REAJUSTAR I, AJUSTAR E	Transfere a parte de endereço da palavra de instrução (que está no registro de memória intermediária) para dentro do registro de endereços da memória.
E e T_0	AJUSTAR W, AC EM MB	Transfere palavra a ser lida na memória do acumulador para dentro do registro de memória intermediária.
E e T_1	REAJUSTAR W	Desliga o biestável de GRAVAR.
E e T_2		Conteúdo do registro de memória intermediária gravado para dentro da memória.
E e T_3	IC EM MA, AJUSTAR I, REAJUSTAR E	O conteúdo do contador de instrução é transferido para dentro do registro de endereços da memória, dando o local da próxima palavra de instrução para a memória. O biestável do ciclo de instrução é ligado e o biestável do ciclo de execução é desligado.

quina. As entradas são o código de operação armazenado no registro do código OP, o distribuidor de sinal de sincronização e os biestáveis I e E.

Observemos, por exemplo, que, quando for o tempo T_0 e estivermos num ciclo de instrução, sempre ativamos o biestável de LEITURA, dizendo à memória para ler a palavra de instrução localizada no endereço no registro de endereços da memória. Então, admitimos que a memória coloca esta palavra no registro de memória intermediária antes do tempo T_1; portanto, no tempo T_1 transferimos a parte de código OP da palavra de instrução para dentro do registro de código OP. Estes dois fatos nos dizem que devemos operar logicamente num circuito E e a linha de saída T_0 do distribuidor de sinal de sincronização com a saída 1 do biestável I e conectar o sinal $T_0 \cdot I$ à entrada de AJUSTE do biestável de LEITURA; e depois devemos conectar um sinal $T_1 \cdot I$ à linha de controle que transfere os primeiros 5 bits do registro da memória intermediária para dentro do registro OP. Isto é mostrado na Fig. 9.6.

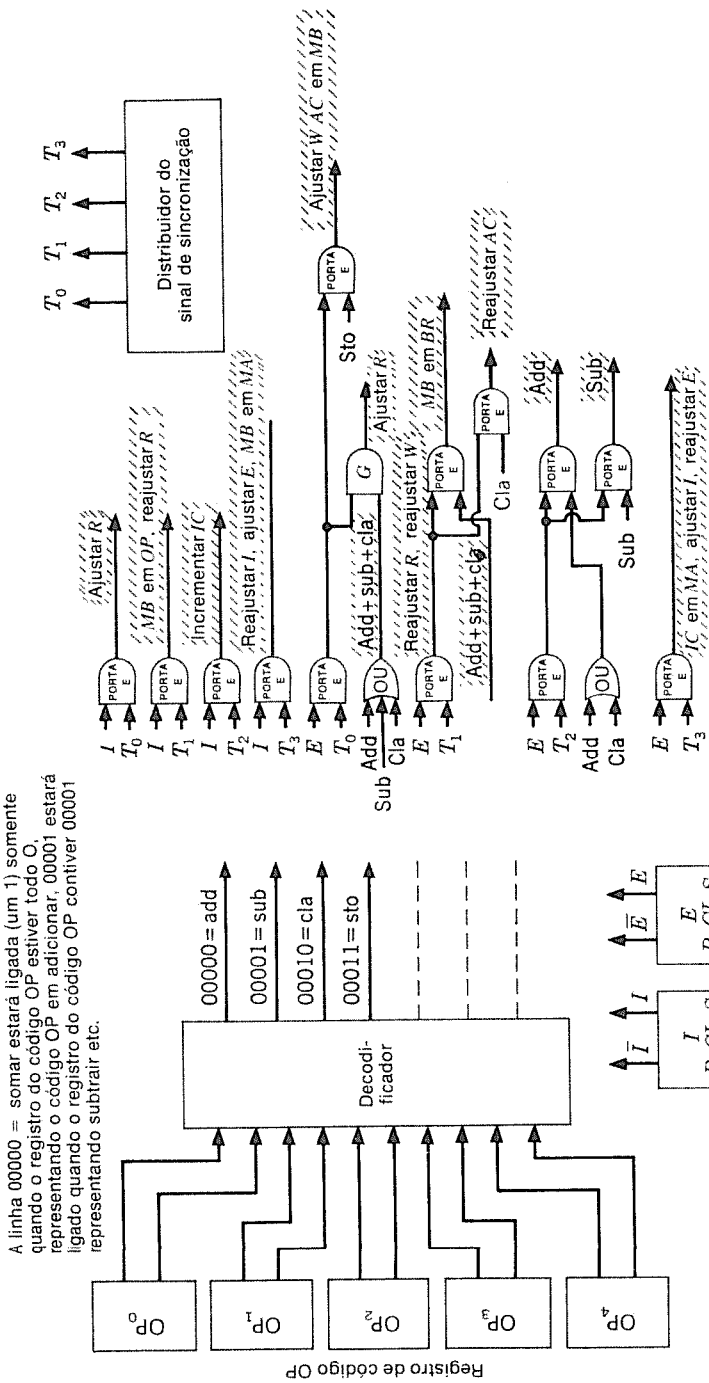


Fig. 9.6 Conjunto de circuitos de controle para computador de quatro instruções.

O que acontece a seguir depende sempre do registro do código OP; portanto, conectamos agora uma matriz seletora ou decodificadora com $2^5 = 32$ saídas para aquele registro (admitindo que usaremos todas as combinações acrescentando mais instruções). Temos, então, um conjunto de linhas de sinal, de modo que a linha 00000 = ADD (SOMAR) transportará um sinal 1 quando estivermos somando (já que o código de operação de ADD é 00000); 00001 = SUB transportará um 1 se, e somente se, estivermos subtraindo, já que o código OP para subtrair é 00001; 00010 = CLA será um 1 somente quando limparmos e somarmos etc. Combinamos estas linhas e as linhas do distribuidor do sinal de sincronização e as linhas dos biestáveis *I* e *E* para nos dar todos os sinais de controle necessários a passar pela máquina. A Fig. 9.6 mostra o conjunto de circuitos de controle requerido. Uma comparação desta figura como a carta do sinal de controle e de sincronização mostrará como são fabricados os sinais e as palavras do conjunto de circuitos de controle quando são necessários.

Podem ser acrescentadas mais instruções à carta do sinal de controle e de sincronização e também acrescentando-se as portas requeridas ao conjunto de circuitos de controle. Analisando a máquina desta maneira, podemos ver prontamente como o conjunto de circuitos de controle orienta as operações executadas na máquina, alternando a aquisição de instruções da memória e a execução das instruções.

9.5 SEQÜÊNCIA TÍPICA DAS OPERAÇÕES

Será instrutivo analisar o conjunto de circuitos de controle na Fig. 9.6 durante ambas as instruções de SOMAR (ADD) e de ARMAZENAR (STORE). Cada instrução será iniciada com o biestável *I* (ciclo de instrução) ligado e com o distribuidor do sinal de sincronização tendo uma saída sobre a linha T_0, de modo que a porta E na parte superior direita da figura estará ativada por *I* e T_0, ajustando assim o biestável de LEITURA no estado 1 e iniciando uma LEITURA da memória. Neste instante, o registro de endereços da memória é suposto ter o endereço da instrução que será lida no registro da memória intermediária.

No tempo T_1, a palavra lida na memória terá sido lida no registro de memória intermediária, de modo que, quando temos o estado de controle *I* e T_1, o conteúdo do registro da memória intermediária que constitui a seção de código OP da instrução será transferido para dentro do registro de código OP, e o computador estará numa posição para decodificar o código OP e determinará que instrução deve ser executada.

Nos tempos de *I* e T_2, o contador de instrução é incrementado de 1, de modo que o contador de instrução agora contém o endereço da instrução seguinte a ser lida na memória. A porta E conectada aos sinais de entrada *I* e T_2 é usada alternadamente no sinal de controle de INCREMENTAR CI, e sua saída é projetada com o nome do sinal de controle.

Similarmente, nos tempos *I* e T_3, o registro de memória intermediária é transferido para dentro do registro de endereços da memória pelo sinal MB EM MA, transferindo assim a parte de endereço da palavra de instrução para dentro do registro de endereços da memória. A próxima palavra lida ou gravada na memória estará então no endereço designado pela parte do endereço da palavra de instrução que acabou de ser lida na memória.

Ao mesmo tempo, o biestável do ciclo de instrução é limpo pelo sinal REAJUSTAR I, e o biestável de execução é ajustado pelo sinal AJUSTAR E, mudando assim o estado da máquina de um ciclo de instrução para um ciclo de execução.

Nos tempos *E* e T_0, então, durante uma instrução de SOMAR, ajustamos o biestável *R* conduzindo, dizendo, portanto, à memória para ler a palavra no endereço comumente no registro de endereços da memória. Neste caso, este endereço será a parte de endereço da palavra de instrução que está sendo executada. Então, nos tempos *E* e T_1, transferimos o conteúdo do registro da memória intermediária para dentro do registro *B*. O registro da memória intermediária naquele tempo contém a palavra que foi lida na memória, de modo que temos agora a palavra que foi endereçada pela palavra de instrução no registro *B* para a adição. Ao mesmo tempo, reajustamos o biestável de LEITURA.

Observemos que as linhas de REAJUSTAR R e REAJUSTAR W são usadas para reajustar simultaneamente ambos os biestáveis de LEITURA e de GRAVAÇÃO. Não há prejuízo no reajuste de ambos os biestáveis, já que somente um estará conduzindo em qualquer instante de tempo determinado.

Se a instrução for uma instrução de SOMAR nos tempos *E* e T_2, somaremos o conteúdo do registro *B* ao conteúdo do acumulador. O registro *B* contém a palavra que foi lida na memória, e o acumulador não foi alterado, e, portanto, sua soma será transferida para dentro do acumulador. Assim, a soma da palavra lida na memória e o conteúdo anterior do acumulador serão colocados no acumulador. Então, nos tempos *E* e T_3, transferimos o contador de instrução para dentro do registro de endereços da memória (fornecendo assim o endereço da próxima instrução a ser executada para a memória), ao mesmo tempo lim-

pando o biestável de EXECUTAR, ajustando o biestável do ciclo de instrução para funcionar e mudando a máquina de um ciclo de execução para um ciclo de instrução.

Uma vez que o biestável *I* está funcionando e o tempo é T_0, a linha de controle de AJUSTAR R estará agora alta, dizendo, assim, à memória para ler uma palavra. A próxima palavra de instrução será lida na memória e poderá então ser interpretada.

Examinemos também a operação da instrução ARMAZENAR que acaba de ser descrita. Quando o biestável de instrução estiver funcionando e estivermos num ciclo de instrução quando chega o tempo T_0, o biestável *R* estará ajustado funcionando, dizendo à memória para ler tal como numa adição, subtração ou limpar e somar; e o mesmo acontecerá durante os tempos *I* e T_1. Uma vez que os biestáveis do registro de memória intermediária contêm agora o código OP da instrução nos tempos *I* e T_1, ele será transferido para dentro do registro de código OP.

Nos tempos *I* e T_2, incrementaremos o contador de instrução de modo que o endereço da instrução seguinte na memória agora permaneça no contador de instrução; e, nos tempos *I* e T_3, reajustaremos o biestável de instrução e ativaremos o biestável do ciclo de execução, colocando, portanto, a máquina num ciclo de execução.

No entanto, nos tempos *E* e T_1, se a instrução for uma instrução de ARMAZENAR, ajustaremos o biestável de GRAVAÇÃO (WRITE) em vez do biestável de LEITURA, iniciando assim, uma GRAVAÇÃO na memória. Também transferiremos o conteúdo do acumulador para dentro do registro de memória intermediária, de modo que a palavra gravada na memória será o conteúdo normal do registro do acumulador e de modo que, após o ciclo de GRAVAÇÃO ter sido terminado, o acumulador terá sido gravado na memória no endereço que foi fornecido pela palavra de instrução.

Nos tempos *E* e T_1, reajustamos o biestável de GRAVAÇÃO, desde que já tenhamos dito à memória para gravar; nada necessita ser feito em *E* e T_2, pois agora estamos gravando a palavra na memória. Nos tempos *E* e T_3, o contador de instrução é transferido para dentro do registro de endereços da memória pelo sinal de controle CI EM MA, fornecendo assim o endereço da instrução seguinte para a memória. O biestável do ciclo de instrução é ativado, e o biestável do ciclo de execução é desativado, voltando assim a máquina ao estado do ciclo de instrução. A máquina agora executará um ciclo de instrução, lendo a próxima palavra de instrução na memória, interpretando-a e continuando à frente no programa.

O exemplo precedente demonstra como é possível projetar uma máquina que irá executar uma determinada seqüência de operações e, desse modo, fazer com que a máquina execute cada palavra de instrução que é lida na memória. Embora apenas quatro instruções fossem demonstradas neste exemplo particular, mais instruções podem ser acrescentadas exatamente de maneira idêntica, anotando simplesmente o que deve ser feito quando uma palavra de instrução é lida na memória, listando as operações que devem ser executadas e fornecendo portas que irão gerar os sinais de controle necessários à execução de cada instrução. As seções subseqüentes discutirão as instruções de deslocamento, de desvio, de ponto flutuante e de indexação. Todas essas podem ser incorporadas à máquina mostrada, acrescentando simplesmente portas ao conjunto de circuitos de controle e fornecendo as portas adicionais necessárias às transferências e operações entre os registros da máquina.

A forma geral do esquema de geração do sinal de controle é mostrada na Fig. 9.7. Ela mostra um distribuidor de pulso de sincronização com oito diferentes divisões de tempo, em cujo caso cada período de tempo seria um oitavo do tempo de ciclo da memória.

9.6 INSTRUÇÕES DE SALTO, DE DESVIO OU DE TRANSFERÊNCIA

As instruções de SALTO, de DESVIO ou de TRANSFERÊNCIA variam em relação à instrução normal de diversas maneiras.* Nas máquinas de endereço único, apenas uma palavra, a palavra de instrução, deve ser localizada na memória. Além disso, os conteúdos de contador de instrução podem ser modificados em vez de serem simplesmente aumentados de 1. Há dois tipos de instruções de DESVIO: condicional e incondicional. Na instrução de DESVIO incondicional, o conteúdo da porção de endereço do registro da memória intermediária é sempre transferido para dentro do contador de instrução. A próxima instrução executada será então a instrução no local indicado pela seção de endereço da palavra de instrução. Na maioria das máquinas de endereço único, no entanto, a instrução de DESVIO condicional fará com que a máquina desvie apenas se o número armazenado no registro acumulador do elemento aritmético for negativo. Se o número no acumulador for positivo, o con-

*Uma pesquisa indica que alguns fabricantes chamam estas instruções de DESVIO, outros chamam-nas instruções de TRANSFERÊNCIA, enquanto ainda outros chamam-nas instruções de SALTO. Todas têm o mesmo significado.

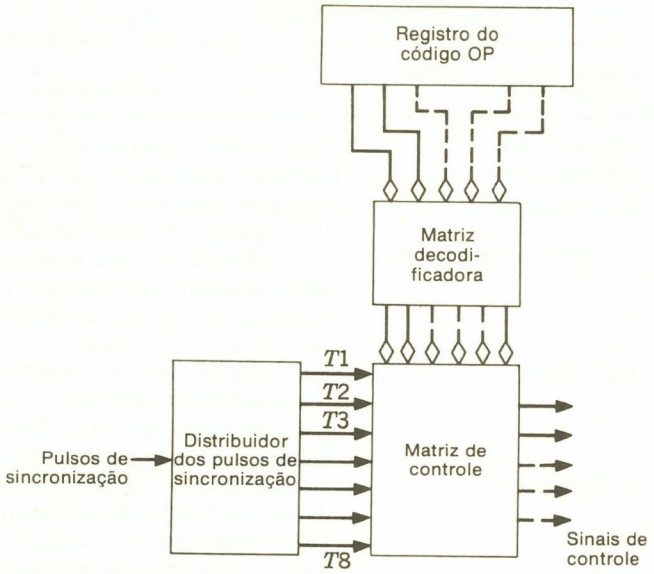


Fig. 9.7 Configuração geral do conjunto de circuitos de controle.

teúdo do contador de instrução será simplesmente aumentado de 1, e a instrução seguinte será tirada na ordem normal.

Durante uma instrução de DESVIO condicional, o bit de sinal do acumulador do elemento aritmético deve ser examinado pelo conjunto de circuitos de controle. Se o bit de sinal for um 1, o número armazenado é negativo, e assim o número na parte de endereço da palavra de instrução é transferido para dentro do contador de instrução. Se o bit de sinal for um 0, um 1 é acrescentado ao contador de instrução e o computador prossegue.

Para demonstrar como uma instrução típica de DESVIO-EM-MENOS opera (BRM) num computador de endereço único, modificaremos o conjunto de circuitos de controle mostrado na Fig. 9.6 de modo que a pequena máquina venha a incluir também uma instrução BRM. Atribuamos à BRM o código de operação 00100, de modo que a linha abaixo da linha 00011 = STO será alta a partir do decodificador ligado ao registro do código OP, na Fig. 9.6, quando uma instrução BRM estiver no registro.

Os dois primeiros períodos de tempo do ciclo de instrução são os mesmos para todas as instruções. Primeiro é dito à memória para ler, e depois a palavra de instrução é lida da memória no registro de memória intermediária. A parte do código OP é depois transferida para dentro do registro de código OP, de modo que, após o tempo T_1 e o começo do tempo T_2, a linha 00100 = BRM será alta e todas as outras linhas de saída do decodificador serão baixas. Agora, preparemos uma pequena tabela para uma instrução BRM mostrando o que deve ser feito a fim de transportar para fora esta instrução. O Quadro 9.4 mostra as etapas que devem ser consideradas.

Se, no tempo T_2, durante o ciclo de instrução, um código OP da instrução BRM estiver no registro de código OP, um dentre dois fatos deve acontecer. Ou desejamos incrementar o contador de instrução e fornecer este número como o endereço da instrução seguinte a ser tirada da memória, ou desejamos transferir o conteúdo da porção de endereço da palavra de instrução no contador de instrução. Qual das duas escolhas faremos depende do bit de sinal do acumulador, chamado AC_0. Se o acumulador contiver um número negativo, ele terá um 1 em AC_0 e, se ele contiver um número positivo, ele terá um 0 no biestável 0. Portanto, se I E T_2 E AC_0, queremos incrementar o contador de instrução; e se I E T_2 E AC_0 acontece ser o caso, desejamos transferir o registro de memória intermediária para dentro do contador de instrução. Isto é mostrado na tabela. Durante o tempo T_3 deste ciclo de instrução, queremos transferir o contador de instrução para dentro do registro de endereços da memória. Não necessitamos colocar a máquina num ciclo de execução, mas podemos simplesmente prosseguir para um outro ciclo de instrução, tirando a palavra no

QUADRO 9.4

Desvio em menos	Lista de sinais de controle a serem ligados	Comentários
I e T_0	AJUSTAR R	Diz à memória para ler palavra de instrução.
I e T_1	MB EM OP, REAJUSTAR R	Transfere parte do código OP da palavra de instrução para dentro do registro do código OP; desliga o biestável de LEITURA.
I e T_2 e AC_0	INCREMENTAR IC	Se o dígito de sinal do acumulador AC_0 for um 0, queremos incrementar o contador de instrução e usar seu conteúdo como endereço da instrução seguinte.
I e T_2 e AC_0	MB EM IC	Se o dígito de sinal do acumulador for um 1, o acumulador é negativo, e queremos usar o endereço na palavra de instrução como o endereço da palavra de instrução seguinte.
I e T_3	IC EM MA	Isso transfere o contador de instrução para dentro do registro de endereço da memória. Observemos que o biestável E não está ligado quando estamos prontos para ler uma outra palavra de instrução; um ciclo de execução não é necessário.

endereço que foi transferido para dentro do registro de endereços da memória como a instrução seguinte. Portanto, não limpamos o biestável do ciclo de instrução ou colocamos um 1 no biestável do ciclo de execução, mas simplesmente transferimos o contador de instrução para dentro do registro de endereços da memória.

O conjunto de circuitos de controle que implementará estas operações é mostrado na Fig. 9.8. O significado disto é que as duas portas E particulares na Fig. 9.6 que são conectadas às entradas I, T_2 e I, T_3 serão substituídas pelos dois circuitos mostrados na Fig. 9.8. Observemos que este conjunto de circuitos lógicos, mais o conjunto de circuitos na Fig. 9.6, é tudo o que é necessário para gerar os sinais de controle requeridos para a instrução BRM.

Observemos também que um sinal $\overline{\text{BRM}}$ (o complemento de BRM) é usado em vez de introduzir juntas, numa porta OU, as instruções que não são instruções BRM. Isto economiza o conjunto de circuitos de controle em que $\overline{\text{BRM}}$ pode ser feito como mostrado na figura, conectando-se simplesmente o sinal BRM a um inversor e designando este por $\overline{\text{BRM}}$.

Observemos que a instrução BRM requer apenas um acesso à memória e, portanto, somente um ciclo de instrução para sua execução.

9.7 INSTRUÇÕES DE DESLOCAMENTO

As instruções que examinamos até agora foram instruções sempre executadas dentro de um número básico fixo de ciclos de memória; assim, as instruções de SOMAR, SUBTRAIR, LIMPAR E SOMAR, e ARMAZENAR foram executadas dentro de exatamente dois ciclos de memória, e as instruções de TRANSFERÊNCIA ou DESVIO requereram apenas um ciclo de memória. Há diversas classes ou tipos de instruções que podem requerer vários ciclos de memória, porque a instrução não pode ser executada dentro, diga-se, de dois ciclos de memória. A multiplicação e a divisão são típicas destas instruções, que geralmente requerem mais tempo do que os dois ciclos de memória normalmente requerem. Similarmente, uma instrução tal como DESVIO À DIREITA ou DESVIO À ESQUERDA pode concebivelmente ser efetuada num ciclo de memória único, desde que o operando esteja no acumulador quando a palavra de instrução for obtida. No entanto, se a instrução requer um grande número de deslocamentos, ela pode requerer mais do que um ciclo de memória para completar o número de operações exigido. Neste caso, não podemos iniciar um outro

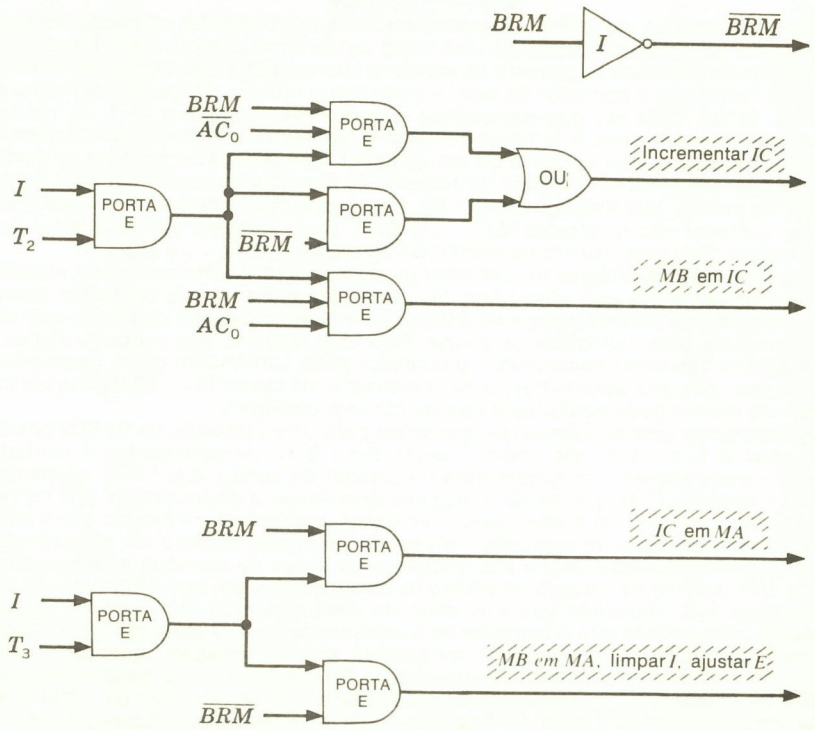


Fig. 9.8 Modificação do conjunto de circuitos de controle para instrução de DESVIO.

ciclo de memória até que tenhamos acabado o deslocamento no número requerido de vezes. Similarmente, na multiplicação e na divisão, não podemos iniciar um outro ciclo de memória até que tenhamos acabado nosso processo de multiplicação e de divisão.

Para implementar estes tipos de instruções, voltaremos portanto nosso controle da máquina para um simples elemento de controle que é dominado por um contador. Este contador porá em seqüência e contará o número de etapas que devem ser executadas até que a instrução tenha sido concluída, e depois colocará a máquina num ciclo de instrução e dirá à memória para ler a palavra de instrução seguinte.

A instrução de DESLOCAMENTO-À DIREITA consiste de duas partes: um código OP e uma parte de endereço. O código OP de 00101 diz à máquina para deslocar a palavra para a direita no acumulador, sendo o número de vezes dado na parte de endereço, de modo que gravemos 00101 para o código OP numa palavra de instrução e depois gravemos 8 em forma binária na parte de endereço, a máquina tem sido instruída para deslocar o número binário no acumulador 8 dígitos binários para a direita.

Admitindo que temos um acumulador com as portas de modo que possamos deslocar à direita os dígitos do acumulador conforme explicado no Cap. 6, tudo o que necessitamos é aplicar oito sinais de controle consecutivos de DESLOCAMENTO-À DIREITA ao acumulador e teremos deslocado o número de 8 lugares à direita. Uma vez que há somente quatro pulsos por ciclo de memória, não queremos usar a memória até que tenhamos completado nosso deslocamento. Se, por exemplo, a instrução dissesse deslocar 1 à direita, poderíamos acabar num tempo de pulso e começar o próximo ciclo de instrução imediatamente depois; mas, se a palavra de instrução dissesse deslocar 4, 5, ou 15 ou mais vezes à direita, teríamos que esperar até que tivéssemos concluído o deslocamento antes que pudéssemos iniciar um outro ciclo de instrução e buscar na memória a palavra de instrução seguinte.

Para isto, preparamos de início a máquina para a operação de deslocamento, incrementando o contador de instrução de modo que a palavra seguinte obtida do contador de instrução conterá o endereço da palavra de instrução seguinte; e, a fim de contar o número

de deslocamentos que efetuamos, acrescentamos um outro registro chamado contador de passos, que é um contador que conta para baixo de um certo número até 0. Depois transferimos a parte de endereço do registro da memória intermediária para dentro do contador de passos, de modo que o contador de passos contenha o número de deslocamentos a serem efetuados. Então, cada vez que deslocamos, diminuímos o contador de 1, de modo que, quando o contador alcança 0, teremos efetuado o número requerido de deslocamentos.

A Fig. 9.9 mostra dois estágios de um contador com redução e também as portas necessárias para transferir o conteúdo do registro da memória intermediária para dentro do contador de passos, que é designado *SC*. Os dois dígitos mais à direita, ou menos significativos, do contador são mostrados (SC_{n-1} e SC_n), como também são mostrados os dois dígitos mais à direita do registro da memória intermediária (MB_{n-1} e MB_n).

O número real de estágios no contador de passos será determinado pelo número máximo de deslocamentos que a máquina deve fazer sempre e, já que usaremos também o mesmo contador na multiplicação e na divisão, pelo número máximo de etapas que sempre serão requeridas para multiplicar ou dividir. Para uma máquina que conta 21 dígitos binários na palavra básica do computador, o contador pode contar bem cinco biestáveis. Para uma máquina com uma palavra básica de computador de talvez 35 ou 36 dígitos binários, o contador de passos pode contar bem seis ou até sete biestáveis.

Consideremos uma seqüência de operações para uma instrução de DESLOCAMENTO. Os tempos I E T_0 e I E T_1 são como o usual. Em I E T_2, incrementamos o contador de instrução e transferimos a contagem para o contador de passos. Em I E T_3, ajustamos um biestável chamado *SR* no qual se diz à máquina para iniciar o deslocamento (*SR* representa *deslocamento à direita*). Ao mesmo tempo, limpamos o biestável *I*, de modo que a máquina não esteja nem no ciclo de instrução nem no de execução, embora ela esteja realmente executando uma instrução. Assim não iniciamos os ciclos de memória subseqüentes, e a máquina efetivamente se congela no estado de deslocamento até que o contador de passos tenha contato a 0, indicando que o número de deslocamentos requerido foi executado. (Realmente, observemos que o contador de passos conta apenas até o binário 1 e não ao 0 antes de a ordem de parar contagem ser emitida, pois a contagem, quando o contador estiver em 0, introduziria um deslocamento extra). Se desligarmos ou desativarmos o biestável *SR* do computador quando a saída do contador estiver no sinal 00...001 e se, ao mesmo tempo, ativarmos o ciclo de instrução ou ciclo *I*, a máquina prosseguirá até o ciclo de instrução seguinte, buscando a próxima palavra de instrução da memória e executando-a. Isto é o que mostra o Quadro 9.5.

Quando o biestável *SR* estiver ativado, será necessário parar o distribuidor de sinal de sincronização; portanto, arranjamos desativar este circuito usando a saída do biestável *SR* com esta finalidade.

A implementação do procedimento acima é direta. Uma porta E de três entradas *I*, T_3 e 00101 = SHR (a saída do decodificador na Fig. 9.6) pode ser usada para ativar um biestável

QUADRO 9.5

Deslocamento à direita	Sinal de controle ligado	Comentários
I e T_0	AJUSTAR R	Diz à memória para ler.
I e T_1	MB EM OP, REAJUSTAR R	Código OP da palavra de instrução é transferido ao registro de código OP. O biestável de LEITURA é desligado.
I e T_2	INCREMENTAR IC, MB EM SC	O contador de instrução é preparado para obter a próxima palavra de instrução. A parte de endereço da palavra de instrução é transferida para dentro do contador de passos.
I e T_3	REAJUSTAR I, AJUSTAR SR	O biestável do ciclo de instrução é desligado. O biestável de deslocamento à direita é ligado.

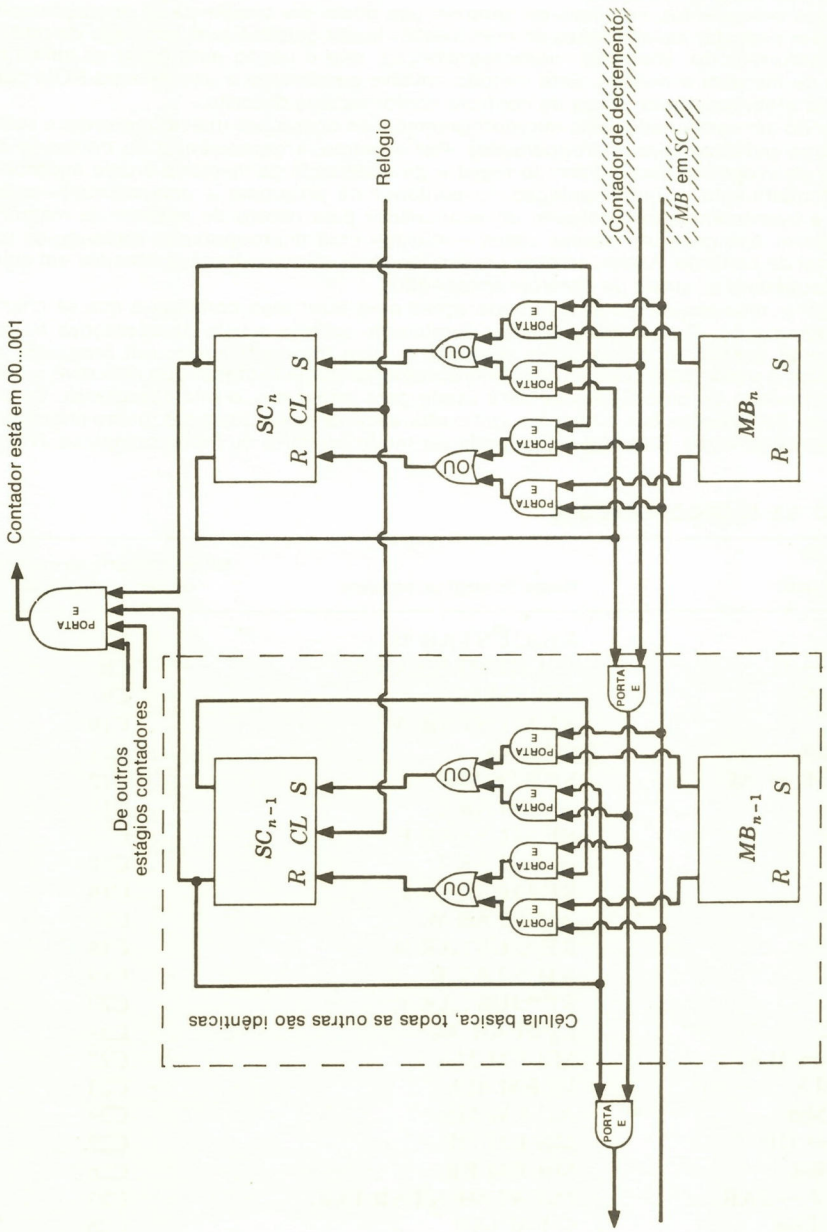


Fig. 9.9 Dois estágios de contador-com-redução e estrutura de transferência.

SR, e a saída parada do contador de passos para desativá-lo, também ativando *I*. A entrada para o relógio pode estar desligada quando *SR* estiver ativado.

9.8 MICROPROGRAMAÇÃO

Nas seções precedentes, os sinais de controle que põem em seqüência as operações efetuadas para executar as instruções do computador foram gerados com emprego de portas. Há um outro método, chamado *microprogramação*, que é usado para gerar os sinais de controle de maneira ordenada. Este método envolve geralmente o uso de uma ROM para armazenar efetivamente os sinais de controle conforme será descrito.

Quando um computador está microprogramado, as operações individuais entre e sobre os registros são chamadas *microoperações*. Por exemplo, a transferência do conteúdo do contador de programa para dentro do registro de endereços da memória é uma microoperação. Similarmente, a incrementação do contador de programa é uma microoperação, como é a transferência do conteúdo do acumulador para dentro do registro da memória intermediária. Em cada um desses casos é iniciada uma microoperação elevando-se um único sinal de controle. Assim, colocar em seqüência as microoperações consiste em colocar em seqüência os sinais de controle apropriados.

Imaginar uma seqüência de microoperações para fazer algo constitui o que se chama *microprogramação*. O microprogramador geralmente escreve a lista de operações ou *microprograma* usando uma linguagem especial, e, bem freqüentemente, um programa de computador é usado para traduzir este microprograma numa listagem que descreve o conteúdo apropriado de uma ROM que será usada para armazenar o microprograma. Os comandos ou declarações que o microprogramador escreve estão, portanto, numa linguagem de microprogramação. Esta linguagem pode ser muito primitiva ou muito complexa. (Varia-

QUADRO 9.6 MICROOPERAÇÕES

Microoperação	Nome do sinal de controle	Bit na memória de controle de leitura exclusiva
$0 \rightarrow IC$	REAJUSTAR IC	C7
$IC + 1 \rightarrow IC$	INCREMENTAR IC	C8
$MB \rightarrow IC$	MB EM IC	C9
$0 \rightarrow AC$	REAJUSTAR AC	C10
$AC + BR \rightarrow AC$	SOMAR	C11
$AC - BR \rightarrow AC$	SUBTRAIR	C12
$1 \rightarrow E$	AJUSTAR E	C13
$0 \rightarrow E$	REAJUSTAR E	C14
$1 \rightarrow I$	AJUSTAR I	C15
$0 \rightarrow I$	REAJUSTAR I	C16
$1 \rightarrow W$	AJUSTAR W	C17
$0 \rightarrow W$	REAJUSTAR W	C18
$1 \rightarrow R$	AJUSTAR R	C19
$0 \rightarrow R$	REAJUSTAR R	C20
$0 \rightarrow AC$	LIMPAR AC	C21
$MB_{5-21} \rightarrow MA$	MB EM MA	C22
$IC \rightarrow MA$	IC EM MA	C23
$AC \rightarrow MB$	AC EM MB	C24
$MB_{0-4} \rightarrow OP$	MB EM OP	C25
$MB \rightarrow BR$	MB EM BR	C26
$IAR + 1 \rightarrow IAR$	INCREMENTAR IAR	C27
$C_{0-6} \rightarrow IAR$	C EM IAR	C28
$OP + IAR + 1 \rightarrow IAR$	SOMAR OP A IAR	C29
$0 \rightarrow IAR$	REAJUSTAR IAR	C30

ções da linguagem de compilador PL/1 são usadas em alguns sistemas.) Além disso, os fabricantes de ROM muitas vezes fornecem um serviço onde uma fita de papel perfurada com o conteúdo desejado da ROM perfurado na fita pode ser automaticamente colocado na ROM.

A fim de explicar a microprogramação, usaremos o traçado do computador e as instruções nas seções precedentes e refaremos o projeto usando uma ROM para armazenar os sinais de controle. Portanto, os registros e os sinais de controle na Fig. 9.3 serão usados no projeto. Primeiro observamos que a lista básica de microoperações necessárias é mostrada no Quadro 9.6. Cada microoperação é descrita usando uma notação simbólica (com efeito, uma linguagem de microprogramação), e o sinal de controle correspondente que fará com que esta operação ocorra também é mostrado.

Por exemplo, a microoperação MB → BR, que significa "transferir o conteúdo do registro da memória intermediária (MB) para o registro *B* (BR)", ocorre aumentando-se o sinal de controle MB EM BR. Observemos a considerável semelhança entre a descrição na linguagem de microprogramação e o nome do sinal de controle. Esta é uma prática conveniente, embora os sinais de controle possam ser denominados X_1, A_1, ou o que se desejar, naturalmente.

A Fig. 9.10 mostra um diagrama-bloco do sistema de controle como ele será implementado. Há uma ROM com 64 locais e 30 bits por endereço e um registro de endereço para esta memória chamado IAR (de registro de endereços da microinstrução). Cada bit de saída da ROM é um sinal de controle que irá gerar uma microoperação, e estes sinais de controle são denominados C_0 a C_{30}. Sete destas saídas são especiais, pois elas são os *endereços seguintes* que podem ser carregados no IAR e que serão usados para colocar em seqüência o IAR em diversos casos. (Este ROM é muitas vezes chamada *memória de controle*.)

As operações que podem ser efetuadas nesta unidade de controle são que um 1 pode ser acrescentado ao IAR (em linguagem de microprogramação IAR + 1 → IAR, o sinal de controle é chamado INCREMENTAR IAR), e os bits de saída da memória de controle designados C_0 até C_6 podem ser transferidos para dentro do IAR. Também é possível adicionar o valor em OP (ver Fig. 9.3) mais 1 para o conteúdo normal do IAR.

Agora, o esquema básico é este; Os sinais de controle para gerar uma certa instrução de computador, diga-se ADD, são armazenados numa seção da memória de controle. O IAR fica em seqüência através dessa seção, e, em cada local, as saídas da memória de controle compreenderão os sinais de controle. Estas saídas da ROM substituem então os sinais de controle gerados pelas portas na Fig. 9.6.

O primeiro problema é que o IAR deve ser ajustado para o endereço correto, no co-

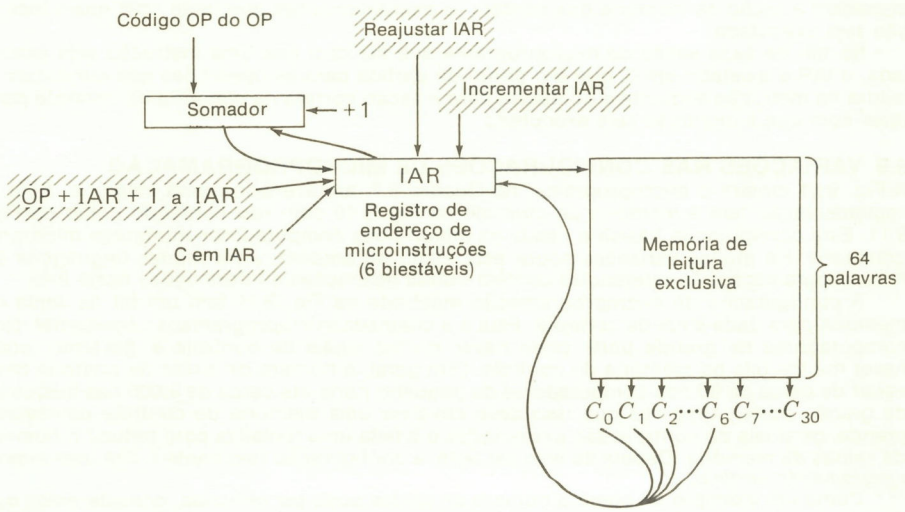


Fig. 9.10 Diagrama-bloco para sistema de controle.

meço da seção, na memória de controle que contém os bits que armazenam os sinais de controle da instrução a ser executada. Para isto, devemos examinar o conteúdo do registro de código OP após termos lido a palavra de instrução da memória e depois movido a seção de código OP no registro da memória intermediária para dentro do registro do código OP. O microprograma completo para a memória de controle é mostrado no Quadro 9.7. Observemos que as primeiras microoperações executadas são:

LOCALIZAÇÃO NA MEMÓRIA DE CONTROLE	MICROPROGRAMA	
0	$1 \rightarrow R$	$IAR + 1 \rightarrow IAR$
1	$MB_{0-4} \rightarrow OP$	$IAR + 1 \rightarrow IAR$
2	$OP + IAR + 1 \rightarrow IAR$	
3	$C_{0-6} \rightarrow IAR$	
4	$C_{0-6} \rightarrow IAR$	
5	$C_{0-6} \rightarrow IAR$	
6	$C_{0-6} \rightarrow IAR$	

A operação aqui é como segue: primeiro é dito à memória para ler (a instrução anterior carregou o registro de endereços da memória com o local da palavra de instrução). A palavra de instrução está depois no registro da memória intermediária quando a próxima microoperação é executada, e esta microinstrução carrega o registro do código OP com os primeiros 5 bits no registro de memória intermediária. A seguir, este valor é adicionado ao registro IAR mais 1. Agora, se a instrução for uma instrução de SOMAR (ADD) com o código 00000 de OP, então 1 será adicionado ao conteúdo normal do IAR (que dará 3 em decimal). Assim, a palavra seguinte na memória de controle a que se deve ter acesso estaria no local 3, e, no local 3, o valor de C nos primeiros sete bits é 20. Quando C é carregado no IAR, a próxima palavra de microinstrução endereçada será aquela no endereço 20 na ROM que contém a primeira microinstrução na seção ADD. Se a instrução em OP for uma de subtrair, o código OP será 00001 e, assim, o próximo local na memória de controle a ser usado estará no local 4 decimal, que causará uma transferência para o local 30 que contém as microinstruções para a instrução SUBTRAIR.

Portanto, uma instrução ADD causará um salto para o local 20 (decimal) na memória de controle, e uma SUBTRAIR causará um desvio para o local 25. Em cada caso, estes locais começam a seção de memória que contém as microinstruções que farão com que a instrução seja executada.

No fim de cada seção de microprograma que faz com que uma instrução seja executada, o IAR é ajustado em 0, que é o ponto de partida para as operações que conduzem à leitura na instrução seguinte e ao desvio para a seção correta na memória de controle para fazer com que a instrução seja executada.

9.9 VARIAÇÕES NAS CONFIGURAÇÕES DE MICROPROGRAMAÇÃO

A Fig. 9.11 mostra o microprograma, no Quadro 9.7, armazenado numa memória. Aqui, a implementação tem a memória de controle na Fig. 9.10 com seu conteúdo como na Fig. 9.11. Esta configuração básica é usada na maioria dos computadores modernos microprogramados. Há muitas variações sobre esta idéia, no entanto, e há muitas linguagens de microprogramação. As referências contêm outras descrições e informações nesta área.

A configuração de microprogramação mostrada na Fig. 9.11 tem um bit de saída da memória para cada sinal de controle. Esta é a chamada *microprogramação horizontal*. Nos computadores de grande porte pode haver muitos sinais de controle e, portanto, pode haver muitos bits na memória de controle. (Em geral, o número de sinais de controle deve variar de cerca de 60 nos computadores de pequeno porte até cerca de 3.000 nas máquinas de grande porte.) Uma vez que isso deve envolver uma memória de controle demasiado grande, os sinais de controle são examinados e é feita uma tentativa para reduzir o número de saídas da memória. Depois de isso ser feito, a configuração resultante é dita usar *microprogramação vertical*.

Como um exemplo de como o número de saídas pode ser reduzido, consideremos que em alguns casos em que um sinal de controle é elevado, um outro é sempre elevado, e, portanto, estes dois sinais podem ser combinados num único sinal.

QUADRO 9.7 MICROPROGRAMA PARA COMPUTADOR DE QUATRO INSTRUÇÕES

Local na memória de controle	Microprograma	Comentários
0	$1 \rightarrow R$, $IAR + 1 \rightarrow IAR$	Diz à memória para ler, incrementar IAR.
1	$MB_{0-4} \rightarrow OP$, $IAR + 1 \rightarrow IAR$	Colocar código OP na instrução em OP.
2	$OP + IAR + 1 \rightarrow IAR$	Somar código OP a 3, isto dá endereço seguinte em IAR a ser usado.
3	$C_{0-6} \rightarrow IAR$ C_{0-6} tem valor 20 decimal	Instrução era SOMAR vai para local 20 na ROM.
4	$C_{0-6} \rightarrow IAR$ C_{0-6} tem valor 25 decimal	Instrução era SUBTRAIR vai para local 25 na ROM.
5	$C_{0-6} \rightarrow IAR$ C_{0-6} tem valor 30 decimal	Instrução era CLA vai para local 30 na ROM.
6	$C_{0-6} \rightarrow IAR$ C_{0-6} tem valor 35 decimal	Instrução era STO vai para local 35 na ROM.
7		
8		
9	Deixado branco para acrescentar mais instruções	
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20	$0 \rightarrow I$, $1 \rightarrow E$, $MB_{5-21} \rightarrow MA$	Começam microoperações da instrução de SOMA; colocar endereço do segundo somando no registro de endereços da memória.
21	$1 \rightarrow R$, $IC + 1 \rightarrow IC$	Ler segundo somando na memória, incrementar contador de instrução.
22	$MB \rightarrow BR$, $0 \rightarrow R$	Colocar segundo somando no registro *B*.

QUADRO 9.7 (CONT.) MICROPROGRAMA PARA COMPUTADOR DE QUATRO INSTRUÇÕES

Local na memória de controle	Microprograma	Comentários
23	AC + BR → AC	Somar e colocar soma no acumulador.
24	IC → MA, 1 → I, 0 → E, 0 → IAR	Preparar a instrução seguinte colocando contador de instrução no registro de endereços da memória e indo para o local 0 na memória de controle.
25	MB_{5-21} → MA, 0 → I, 1 → E	Começar as microoperações da instrução de subtrair.
26	1 → R, IC + 1 → IC	
27	MB → BR, 0 → R	Colocar subtraendo no registro *B*.
28	AC − BR → AC	Subtrair e colocar diferença no acumulador.
29	IC → MA, 1 → I, 0 → E, 0 → IAR	Colocar endereço da palavra de instrução seguinte no registro de endereços da memória e ir para 0 na memória de controle.
30	MB_{5-21} → MA, 0 → I, → E	Começar operações da instrução CLA.
31	1 → R	
32	MB → BR, 0 → AC, 0 → R	Reajustar acumulador.
33	AC + BR → AC	Somar registro *B* ao acumulador.
34	IC → MA, 1 → I, 0 → E, 0 → IAR	Terminar instrução CLA; colocar endereço da instrução seguinte no registro de endereços da memória e ir para 0 na memória de controle.
35	0 → I, 1 → E, MB_{5-21} → MA	Começar instrução STO.
36	1 → W, AC → MB	Colocar conteúdo do acumulador no registro de memória intermediária de modo que ele possa ser armazenado, dizer à memória para gravar.
37	0 → W	Preparar instrução seguinte.
38	IC + 1 → IC	Fim de instrução, colocar endereço de instrução seguinte no registro de endereços da memória e ir para 0 na memória de controle.
39	IC → MA, 1 → I, 0 → E, 0 → IAR	

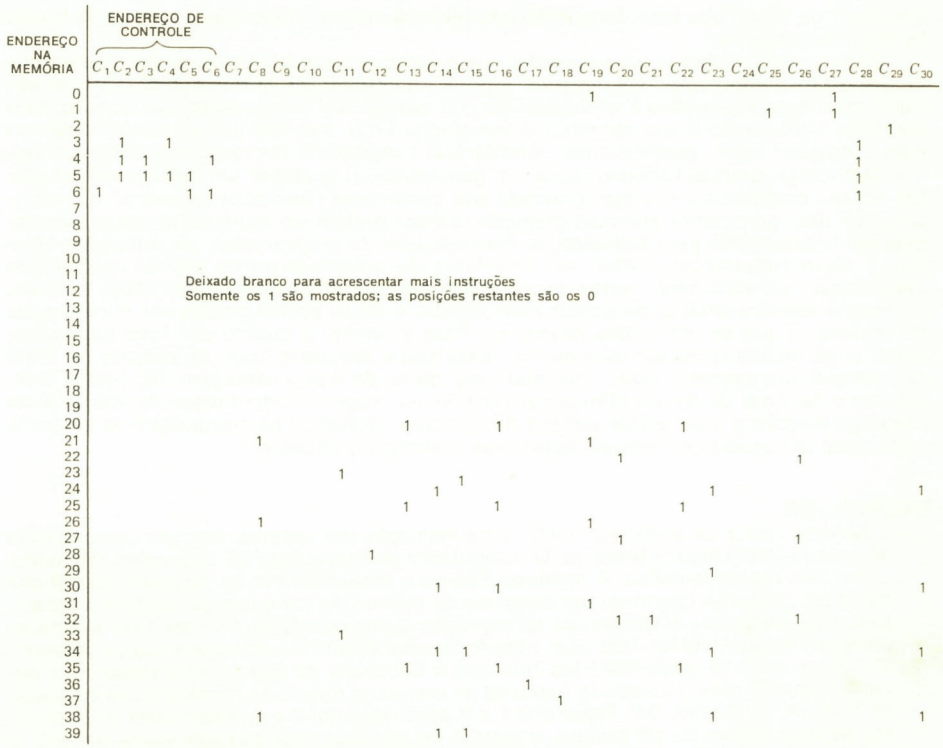


Fig. 9.11 Microprograma na memória. Endereço C_0 de controle tem 0 em todas as posições.

Em alguns casos, diferentes sinais de controle nunca são ligados ao mesmo tempo. Se puderem ser achados N de tais sinais, então apenas M linhas de controle, onde $2^M \geq N$, serão necessários, e um decodificador pode ser usado para fornecer os sinais de controle requeridos. Por exemplo, os sinais ADD, SUB, RESET AC e MB EM BR nunca são ligados ao mesmo tempo e, portanto, duas linhas de saída de controle com um decodificador de quatro saídas podem ser usadas para gerar estes sinais.

Quando é usada a microprogramação vertical, o sistema se torna menos flexível uma vez que, se um microprograma deve ser modificado ou ampliado, menos opções na geração de sinal de controle serão disponíveis. Em conseqüência, as máquinas comerciais em sua maioria são arranjadas de tal modo que estejam em algum lugar entre horizontal e vertical em sua construção. (Muitos esquemas foram usados e vários são descritos nos exercícios.)

A microprogramação é amplamente usada nas novas linhas de computador. Uma vez que o repertório de instruções para o computador é efetivamente armazenado na ROM, as instruções fornecidas podem ser alternadas ou adicionadas mudando ou acrescentando à ROM.

Além disso, a microprogramação é útil na *simulação* de um computador em outro. O significado disto é que temos um computador que possui um conjunto básico de registros e de operações entre registros e temos a capacidade de microprogramar este computador. Além disso, temos um segundo computador com um certo conjunto de instruções e um conjunto de programas escritos para passar neste segundo computador. Desejamos agora fazer o primeiro computador passar estes programas e fornecer os mesmos resultados que este segundo computador teria fornecido. Esta é a chamada *simulação*, e o primeiro computador é dito *simular* o segundo computador. Para isto, microprogramamos o computador

simulador de modo que uma dada instrução tenha o mesmo efeito que a mesma instrução na segunda máquina.†

Como pode ser visto, um computador que é microprogramado pode ser feito para simular um outro computador. Claramente, alguns computadores têm arquiteturas que são muito mais bem adequadas à simulação do que outras. Um computador que é muito bem adaptado à simulação é, por exemplo, o Burroughs 1700, que tem grande flexibilidade em suas operações entre registros e um considerável conjunto de microoperações disponível.

Os microprogramas fornecidos por um fabricante (ou qualquer um outro) a serem usados em seu computador microprogramado são geralmente chamados *firmware*.* As instruções que um computador microprogramado fornece podem ser muito complexas e cuidadosamente projetadas para satisfazer as necessidades do programador. As principais objeções à microprogramação podem ser: (1) velocidade, porque as portas lógicas usadas num computador "convencional" seriam mais rápidas que a ROM na maioria dos casos e, assim, a máquina convencional pode passar mais rápido; e (2) as portas podem ser minimizadas em número já que as instruções devem ser fixas e, assim, a quantidade total de equipamento pode muitas vezes ser feita menor. (Este não é sempre o caso; no entanto, as ROM são bastante compactas e pouco dispendiosas, de modo que a vantagem das portas diminui com o decorrer do tempo.) Em conseqüência, os "super" computadores de maior porte e rápidos tendem a usar portas lógicas de controle, enquanto os computadores de porte médio e de pequeno porte tendem agora a ser microprogramados.

EXERCÍCIOS

1. Um computador de endereço único, uma instrução por palavra, tem um comprimento de palavra de 22 dígitos binários. O computador pode executar 32 instruções diferentes e tem três registros-índice. A memória interna é uma memória de núcleos magnéticos de 16.000 palavras. Desenhar um diagrama da palavra do computador, alocando espaço para cada parte da palavra básica da instrução (parte de código OP, parte de endereço, parte de registro-índice). Não usar o dígito de sinal (dígito mais à esquerda) da palavra.
2. O código octal 15 (para ANA) faz com que o conteúdo do acumulador passe num circuito lógico E, com o conteúdo (palavra) no endereço fornecido na palavra de instrução no DDP-24 no Quadro 9.2. Fazer bits 1 a 3 para cada um 0 e também fazer bit 4 um 0; em seguida, escrever, em binário, a palavra de instrução que passará, numa porta E, o acumulador com o conteúdo do endereço 16 na memória.
3. Se a palavra básica de computador numa máquina tiver 24 dígitos binários e temos 55 instruções diferentes, construir um formato da palavra de instrução para uma máquina de endereço único. Quantos locais de memória diferentes podem ser diretamente endereçados pela porção de endereço desta palavra de instrução do computador? Construa a palavra de instrução do computador para um computador de endereço duplo com 24 dígitos binários na palavra de computador e 55 instruções. Quantos locais diferentes podem agora ser diretamente endereçados por uma única palavra de instrução?
4. Se o acumulador do computador DDP-24 contiver todos os 1 (tem um 1 em cada bit) e o endereço 30 na memória contiver todos os 0, uma instrução STC 30_{10}, na qual 30 se refere ao local 30 (decimal), resultará em que no local 30 na memória?
5. Escrever um programa simples, usando os códigos de operação para o computador Raytheon 7180 dados no Quadro 9.1, que avaliará o polinômio $ax^2 + bx + c$, sabendo que a, b, c e x estão nos endereços da memória 29, 28, 27 e 26, respectivamente. Armazenar o resultado em 30.
6. Era de esperar que o contador de instrução de um computador de endereço único tivesse tantos bits quanto a porção de endereço da palavra do computador? Por quê?
7. O código de instrução para o DDP-24 é listado no Quadro 9.2, junto com um diagrama da palavra básica da instrução. Escrever, em forma binária, a palavra de instrução que fará com que o conteúdo do registro *B* seja multiplicado pela palavra armazenada no local 24 da memória interna do computador.
8. Projetar um único estágio de um acumulador e registro *B* que somará e deslocará à esquerda numa operação (etapa) ou simplesmente deslocará à esquerda numa etapa. Usar como sinais de controle DESLOCAMENTO-À ESQUERDA e SOMAR E DESLOCAMENTO-À ESQUERDA, um somador total, portas E e OU e biestáveis tipo *RS*.

---

†Isto é freqüentemente chamado *emulação* quando é usada a microprogramação. É necessário dar novo nome aos registros e arranjar outras alterações para realmente efetuar isto, mas o princípio é essencialmente dado aqui.
**Firmware* = programas carregados em memórias ROM.

9. Organizar uma tabela de sincronização e modificar o conjunto de circuitos de controle na Fig. 9.6, incluindo a modificação na Fig. 9.8, de modo que a máquina tenha uma instrução BRA de DESVIO incondicional, bem como uma instrução BRM de DESVIO condicional, gerando os necessários sinais de controle.
10. Explicar a instrução RND no Quadro 9.2.
11. Escrever um programa para o DDP-24, usando os códigos OP no Quadro 9.2, que calcularão a expressão $(x + y^2)/z$, admitindo que x, y e z estão nos endereços 301, 302 e 303 na memória. Admitir que não seja necessário escalamento e armazenar o resultado no endereço 304 na memória.
12. Explicar a instrução STD no Quadro 9.2. Esta é chamada instrução de modificação do endereço. Por quê?
13. Explicar a instrução SBM no Quadro 9.2.
14. A linguagem de microprogramação esboçada neste capítulo é geralmente chamada *linguagem de transferência de registro*. Discutir como se expandiria ou talvez melhoraria esta linguagem. A microprogramação numa linguagem de nível mais alto, tal como PL/1 ou uma de suas variações, produziria um microprograma eficiente na memória de controle? Discutir este caso.
15. Mostrar como modificar a Fig. 9.8 de modo que a instrução BRM se torne uma instrução BRP, o que significa que o computador salta ou desvia quando o ACC é positivo em vez de negativo.
16. Mostrar como gerar sinais de sincronização (tais como T_0, T_1, T_2 e T_3 na Fig. 9.5) usando um registro de deslocamento com as saídas dos estágios mais à direita conectadas às entradas dos estágios mais à esquerda. Isto é o que se chama *contador em anel*. Em que estados se ajustariam os biestáveis DC para iniciar?
17. Por que o contador de instrução está sempre colocado no registro de endereços da memória no tempo T_3 durante a parte de execução de uma instrução no Quadro 9.3?
18. Explicar por que algumas instruções requerem ambos os ciclos de execução e instrução e algumas requerem apenas ciclos de instrução. Dar exemplos de ambas as espécies de instrução.
19. Por que os sinais de controle SET W e AC EM MB podem ser combinados na Fig. 9.6?
20. Por que os sinais de controle IC EM MA, SET I e CLEAR E podem ser combinados na Fig. 9.6?
21. Mostrar como acrescentar um botão de pressão conectado a um fio RESET e START que fará (usando *DC* SET e *DC* RESET nos biestáveis) com que o computador na Fig. 9.6 comece a execução de um programa principiando no local 0 na memória quando abaixado o botão.
22. Acrescentar uma instrução DESVIAR EM ZERO semelhante à do Quadro 9.4 e Fig. 9.8, exceto que o computador desvia quando o valor do acumulador for todos os 0.
23. Acrescentar uma instrução de DESLOCAMENTO-À ESQUERDA no controle do computador do exemplo usando uma técnica como no Quadro 9.5 e Figs. 9.8 e 9.9.
24. Quando a microprogramação é usada para gerar sinais de controle e uma ROM for usada, como pode ser alterado o repertório de instrução do computador?
25. Na Fig. 9.10, os sinais de controle podem ser carregados nos biestáveis tipo *D* e as saídas dos biestáveis usadas como sinal de controle reais. Dar as vantagens e desvantagens deste fato.
26. Explicar a diferença entre uma microoperação e o sinal de controle que a implementa.
27. A fim de implementar as instruções DESVIO ou SALTO NO ACUMULADOR NEGATIVO, um outro sinal C_{30} de controle pode ser acrescentado, o qual, quando for um 1, produz um teste do bit de sinal do acumulador e um salto na memória de controle para uma seção de microoperações que produz a mudança desejada na seqüência de operações do computador. Projetar esta instrução.
28. Ao escrever microprogramas, é conveniente ter uma microoperação IF. Por exemplo, para escrever um microprograma para implementar uma instrução de desvio, podemos ter uma microoperação como IF($AC_0 = 1$) THEN $C_{0-6} \rightarrow$ IAR ELSE IAR + 1 $\rightarrow$ IAR. Isso diz se AC_0 é um 1, então colocar o valor normal de C_0 a C_6 em IAR, o que significa que a microinstrução seguinte será do endereço dado em $C_0 - C_6$. Se AC_0 for um 0, a próxima microinstrução será do local seguinte no armazenamento de controle. Escrever um microprograma para a instrução desvio-registro-memória, usando esta microoperação.
29. Escrever um microprograma para uma instrução BRP (desvio ou positiva), usando a informação no exercício precedente.
30. Mostrar como implementar a instrução no exercício precedente.
31. Escrever um microprograma para uma instrução DESLOCAMENTO-À DIREITA usando o

tipo IF de declaração descrito há pouco (Há necessidade de um contador.)
32. Mostrar como implementar o microprograma no exercício precedente.
33. Escrever um microprograma para implementar uma instrução de multiplicação.
34. Mostrar como implementar a instrução de multiplicação no exercício precedente.
35. Escrever um microprograma para implementar uma instrução de DIVIDIR.
36. Mostrar como implementar sua instrução de DIVIDIR do exercício precedente.
37. Reduzir o número de sinais de controle usados na Fig. 9.11.
38. Explicar que características podem ser queridas num computador que deve ser microprogramado para simular diversos outros computadores.
39. Discutir algumas das vantagens e desvantagens da microprogramação como você as vê.

10

ORGANIZAÇÃO DE COMPUTADORES

Os computadores existem atualmente numa variedade enorme de tamanhos, custos e capacidades. Os menores computadores são chamados microcomputadores, os que se seguem em tamanho são os minicomputadores, os de pequeno porte, os de porte médio e, finalmente, os de grande porte ou super ou *maxi*computadores. A faixa de preços é desde alguns dólares (para um sistema de pastilhas num microcomputador) até vários milhões de dólares. As velocidades são de dezenas de microssegundos por instrução (ou operação) a dezenas de instruções por microssegundo.

O microcomputador é a mais nova aquisição na hierarquia de computadores, e um determinado microcomputador geralmente consiste de várias pastilhas de circuitos integrados (CI), incluindo uma pastilha (ou pastilhas) da unidade central de processamento (UCP) chamada pastilha (ou pastilhas) do *microprocessador,* inúmeras pastilhas de memória e uma ou mais pastilhas da interface de entrada-saída. Estes conjuntos de pastilhas podem ser muito baratos (alguns dólares em grandes quantidades) ou bem dispendiosos (várias centenas de dólares para conjuntos de pastilhas de alta velocidade). As conhecidas calculadoras de mão muitas vezes são montadas com pastilhas de CI, incluindo uma das pastilhas de microprocessador de preço mais baixo. Os microcomputadores completamente montados podem custar tão pouco quanto várias centenas de dólares.

As pastilhas de microprocessador são amplamente usadas no que se denomina sistemas ou dispositivos de fabricante de equipamentos originais (OEM); e as luzes do tráfego, as impressoras, os controladores de comunicações, os sistemas de instrumentação controlados automaticamente, registros (documentos) de caixa, instalações automáticas de ensaios gerais em supermercados e empórios etc., todos fazem ampla utilização de microprocessadores.

Similarmente, os minicomputadores, que, em geral, têm preços desde mil a dezenas de milhares de dólares (incluindo memória e entrada-saída), são amplamente usados em sistemas de controle e nos sistemas OEM bem como nas aplicações científicas e no processamento de dados comerciais nos pequenos negócios, nas escolas, nos laboratórios etc. O minicomputador precedeu em tempo o microcomputador, e o minicomputador continua a ser amplamente usado já que os computadores nesta faixa de preços proporcionam a muitos usuários as capacidades adicionais suficientes para garantir o custo extra.

O mercado de computadores de pequena e média escalas encontra aplicação nas atividades comerciais e nos laboratórios de todas as espécies, bem como nos hospitais, mercados, pequenos estabelecimentos bancários etc. Os maiores computadores devem ser encontrados nas grandes empresas tais como as companhias de seguro, os grandes bancos, os grandes laboratórios científicos, as grandes universidades etc. Estes supercomputadores compreendem desde os "trituradores de números," orientados para as aplicações científicas, até os grandes sistemas ou complexos de dispositivos de entrada-saída e de memórias usados no comércio onde a ênfase é na manutenção de grandes arquivos de dados, produzindo relatórios administrativos, faturamento, pedidos ou encomendas automáticas, controle de inventário etc.

As características destas categorias de computadores diferem consideravelmente de categoria a categoria e também de projeto a projeto. Os computadores para o processamento de dados comerciais têm características de sistemas diferentes das dos computado-

res de trabalhos científicos. Há também considerável variação de opiniões quanto a como devem ser configurados os computadores para a mesma área de aplicação, o que conduz a diferentes projetos de computadores.

O assunto de *arquitetura de computadores* se estende através de quase todos os aspectos de organização de computadores. Incluído está o comprimento das palavras de instrução, quer o comprimento seja variável quer haja diversos comprimentos diferentes, e quantos endereços na memória sejam consultados por uma palavra de instrução. Outras considerações arquitetônicas concernem ao número de bits em cada palavra da memória, se as palavras de instruções e de dados são do mesmo tamanho que as palavras da memória, se os números são manipulados em forma de complemento de 1 ou de 2 ou em BCD ou em alguma combinação destes. Quais são as instruções fornecidas, como são organizadas as memórias e como são as interfaces entre os dispositivos de entrada-saída? Como pode ser visto, a arquitetura do computador é um assunto rico e vasto que trata da maioria dos aspectos do projeto e da organização dos computadores e que interage com todos os aspectos do computador.

Este capítulo descreverá alguns dos formatos da palavra de instrução atualmente em uso. Em seguida serão descritas algumas arquiteturas típicas de computadores. Os detalhes verdadeiros dos computadores autênticos serão usados para ilustrar os princípios incluídos, e, embora nenhum computador seja descrito em detalhes, deve ser fornecida informação suficiente de modo que a arquitetura e a operação da maioria dos computadores possam ser prontamente compreendidas. A intenção é, portanto, a de se concentrar nos princípios envolvidos, usando os computadores autênticos para ilustrar estes princípios.

10.1 FORMATOS DA PALAVRA DE INSTRUÇÃO — NÚMERO DE ENDEREÇOS

Um determinado computador tem um ou mais formatos básicos para suas palavras de instrução. Temos enfatizado a palavra de instrução de endereço único que é muito popular em microcomputadores e constitui numa certa época o tipo predominante de palavra de instrução (incluindo o uso com a maioria dos grandes computadores). Existem atualmente diversos outros formatos em uso, no entanto, e estes são descritos abaixo.

Instruções de dois endereços

O número de divisões na palavra básica de instrução do computador é determinado principalmente pelo número de endereços que são consultados. Muitos computadores têm palavras de instrução de dois endereços. Um computador de dois endereços geralmente tem uma palavra básica de instrução com três seções (Fig. 10.1), consistindo a primeira da seção do código OP e as segunda e terceira seções contendo cada uma o endereço de um local na memória.

As diferentes máquinas usam estes endereços de modo diferente. Em geral, ambos os endereços numa máquina de dois endereços podem especificar os operandos, e o resultado é armazenado no primeiro dos endereços. A série 200 da Minneapolis-Honeywell e as séries 1400 e 360/370 da IBM têm instruções de dois endereços que fornecem exemplos nos quais cada endereço se refere a um operando na memória.

Em muitos computadores, em vez de um único acumulador há dois ou mais registros que são chamados ou *acumuladores múltiplos* ou *registros de propósitos gerais*. Uma palavra de instrução terá então que dizer à primeira seção do endereço [a seção do "endereço do operando *(A)*" na Fig. 10.1] que o registro geral contém um dos operandos. A segunda seção do endereço da palavra de instrução fornecerá então o endereço na memória do segundo operando. Se somente dois acumuladores ou registros gerais forem fornecidos, somente 1 bit é necessário para a primeira seção do endereço; se forem usados 16 registros gerais, então serão necessários 4 bits para o primeiro endereço. Os resultados são usualmente armazenados no registro geral (acumulador) especificado pelo primeiro endereço.

Em alguns computadores são fornecidas palavras de instrução nas quais cada um dos dois endereços se refere aos registros gerais; e assim, por exemplo, num computador com 16 registros gerais, uma palavra de instrução deve consistir do código OP mais duas seções com endereço de 4 bits.

As palavras de instrução de dois endereços em que um ou ambos os endereços se referem aos registros gerais são menores do que as palavras de instrução de dois endereços onde ambos os endereços se referem à memória, e este formato é muito comum em minicomputadores.

Instruções de três endereços

A instrução de três endereços geralmente usa dois endereços para indicar os operandos a serem usados e o terceiro endereço para indicar o local em que o resultado deve ser arma-

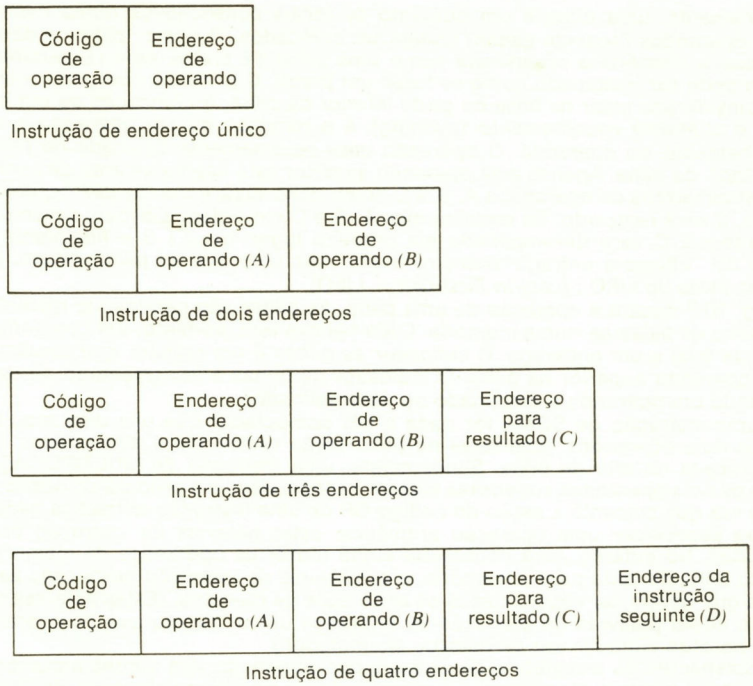


Fig. 10.1 Formatos das instruções.

zenado (Fig. 10.1). A seção do código de operação da palavra indica a operação a ser executada — por exemplo, ADIÇÃO, SUBTRAÇÃO, MULTIPLICAÇÃO etc. Os dois primeiros dos três endereços são usados para indicar os locais dos operandos a serem utilizados. Por exemplo, se a instrução for uma instrução de SOMAR, o conteúdo do local *A* será adicionado ao conteúdo do local *B*. O último endereço indica o local em que o resultado é armazenado, como na instrução "somar *A* a *B* e armazenar o resultado em *C*." A máquina tirará suas instruções da memória em ordem, a não ser que seja instruída para desviar ou saltar. A palavra de instrução, e portanto a palavra básica do computador para um sistema de três endereços, deve ser de comprimento moderado. As máquinas de três endereços não são rigorosamente populares nesta época, exceto na União Soviética, embora, numa certa época, elas tivessem sido mais populares do que as máquinas de dois endereços. As máquinas típicas de três endereços foram a Dy-Seac do National Bureau of Standards, as Datamatic 1000 e Datamatic 800 da Datamatic Corporation e a Bizmac da Radio Corporation of America.

Instruções de quatro endereços

Se a palavra básica de instrução contiver quatro endereços (Fig. 10.1), três são geralmente usados como na máquina de três endereços, e o quarto é usado para indicar o local da instrução seguinte. Se designarmos os quatro endereços por *A*, *B*, *C* e *D*, uma instrução de MULTIPLICAÇÃO fará com que o conteúdo do endereço *A* seja multiplicado pelo conteúdo de *B* e armazenado em *C*. A instrução seguinte será então tirada do endereço *D* da memória. Máquinas típicas de quatro endereços foram Swac do National Bureau of Standards e a Raydac da Raytheon Manufacturing Company. Atualmente, há poucas, se houver, máquinas de quatro endereços.

Instruções sem endereço — pilhas

Há um tipo de palavra de instrução que não especifica nenhum local na memória para um operando, mas que conta com o que é chamado uma *pilha* para fornecer operandos.

Basicamente, uma pilha é um conjunto de locais consecutivos numa memória nos quais os operandos (itens de dados) podem ser colocados. O nome "pilha" é derivado do fato de que a memória é organizada como uma pilha de pratos num restaurante. (Cada operando pode ser imaginado como se fosse um prato). O primeiro operando colocado na pilha é considerado estar na base ou parte inferior da pilha. A colocação de um operando na pilha é chamada *empilhamento* (pushing), e a remoção de um operando é chamada *desempilhamento* do operando. O operando mais recentemente colocado na pilha é dito estar no *topo* da pilha. Apenas este operando superior está imediatamente disponível.

Se empilharmos os operandos *A, B* e *C* numa pilha vazia e depois desempilharmos um operando, *C* será removido. Se empilharmos *A, B* e *C* em ordem e depois desempilharmos três operandos, *C* será desempilhado em primeiro lugar, depois *B* e finalmente *A*. (Este princípio do "último a entrar"/"primeiro a sair" faz com que as pilhas sejam, às vezes, chamadas listas de LIFO.) (*Last-In First-Out* = LIFO)

A Fig. 10.2 mostra a operação de uma pilha. As pilhas são geralmente mantidas como um conjunto de palavras numa memória. Cada palavra tem, portanto, um comprimento fixo (número de bits) e um endereço. O *indicador de pilhas* é um registro que contém o endereço do operando superior na pilha. O indicador de pilhas é incrementado ou decrementado quando um operando é empilhado ou desempilhado.

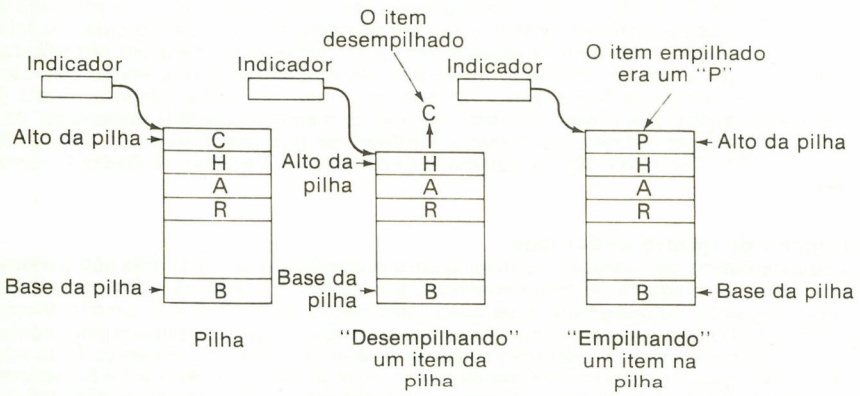
Se uma instrução de SOMA for dada a um computador que usa uma arquitetura de pilhas, os dois operandos superiores na pilha serão removidos e adicionados e a soma então colocada no alto da pilha. Similarmente, uma instrução de MULTIPLICAÇÃO faria com que os dois operandos superiores fossem multiplicados e o produto colocado na pilha.

Uma vez que somente a seção do código OP de uma instrução aritmética necessita ser dada para especificar uma operação aritmética, estas palavras de instrução podem ser muito curtas. No entanto, será necessário ainda mover os operandos da memória para a pilha e da pilha de volta para a memória, e as palavras de instrução neste caso serão mais longas já que devem ser especificados os endereços da memória. (Estas palavras de instrução serão como palavras de instrução de endereço único, exceto que os operandos são movidos para/da pilha em vez de para/do acumulador.)

Os defensores da arquitetura de computador empilhado têm alguns argumentos convincentes, mas existem problemas. Os computadores empilhados incluem os 5500 e 1700 da Burroughs e o 3000 da Hewlett-Packard. As pilhas são amplamente usadas em outras seções dos computadores, como será mostrado.

10.2 REPRESENTAÇÃO DE INSTRUÇÕES E DADOS

Os aspectos importantes de uma arquitetura de computador se referem ao número de bits nas palavras de instrução, ao tamanho das palavras de memória e à maneira pela qual os dados são representados no computador. Nos computadores mais antigos e em muitos dos



Nota: Neste caso, cada item na pilha é um caráter único. Os itens da pilha podem ser números, palavras, registros etc. O indicador contém o endereço do "alto da pilha".

Fig. 10.2 Operações das pilhas.

modernos computadores, a memória de alta velocidade contém o mesmo número de bits em cada endereço (em cada local) que as palavras de instrução. Similarmente, os números são representados novamente usando o mesmo número de bits. Isto torna direta a implementação. Um exemplo de um computador com esta estrutura pode ser achado no PDP-8, que tem uma memória de núcleos com 12 bits por palavra e palavras de instrução de 12 bits com um código OP de 3 bits, e os números são representados usando-se um sistema de numeração de 12 bits em complemento de 2 com sinal. Muitos computadores de grande porte foram e estão sendo feitos com esta estrutura; as séries de computadores 700, 704, 709, 7094, da IBM e 6000 e 60 da Minneapolis-Honeywell, todas têm palavras de memória de 36 bits e palavras de instrução de 36 bits, e os números binários são representados usando-se números de 36 bits de complemento de 2 com sinal (e, também, os números de ponto-flutuante são basicamente de 36 bits). A maioria das grandes máquinas trituradoras de números (orientadas para os problemas científicos) também usa esta estrutura, e a CDC produz vários computadores de grande porte com 64 bits por palavra e com esta estrutura básica, assim como alguns computadores menores com 24 bits por palavra.

É desejável que a eficiência esteja relacionada ao comprimento das palavras de instrução. Além disso, o processamento de dados comerciais envolve muita manipulação que usa cordões de caracteres (nomes, endereços, texto etc.). O desejo de criar arquiteturas de computador que conservem o comprimento da palavra de instrução e também permitam o armazenamento de cordões de caracteres de comprimento arbitrário eficientemente conduziu a inúmeras arquiteturas de computador com (1) somente 8 bits em cada endereço na memória, e portanto um único caráter alfanumérico pode ser armazenado em cada endereço; (2) palavras de instrução com comprimentos variáveis (cada comprimento de palavra é geralmente algum múltiplo de 8 bits).

As séries de computadores 1600 e 1400 da IBM, bem como a série 200 da Honeywell, todas têm 8 (ou 9) bits por palavra na memória de alta velocidade, e os códigos OP da palavra de instrução foram um caráter (8 bits) alfanumérico mais tantos bits quantos forem necessários para fornecer os dois endereços de memória requeridos para cada instrução. Esta espécie de computador é descrita na Seção 10.11.

A série 360/370 da IBM tem 8 bits por palavra na memória e vários comprimentos de palavra de instrução diferentes variando de 16 a 48 bits.

Em muitos dos modernos e maiores computadores, os números podem ser representados em BCD bem como em binário e em ponto-flutuante, e os cálculos podem ser efetuados em cada um destes sistemas de representação de números.

Os microcomputadores tendem a ter palavras de dados e palavras de memória de 4, 8 ou 16 bits. As palavras de instrução, em geral, são de comprimento variável, mas os comprimentos são quase sempre múltiplos do número de bits na memória.

10.3 TÉCNICAS DE ENDEREÇAMENTO

Quando um endereço na memória for designado numa palavra de instrução, a técnica mais óbvia consiste simplesmente em fornecer o endereço na memória em forma binária. Este é o chamado *endereçamento direto,* e as palavras de instrução nos exemplos no Cap. 9 usaram todas o endereçamento direto.

Embora o endereçamento direto proporcione a maneira mais direta (e mais rápida) de fornecer um endereço da memória, muitas outras técnicas também são usadas. Estas técnicas são geralmente motivadas por uma das seguintes considerações:

1. *Desejo de encurtar a seção de endereço.* Por exemplo, se tivermos um minicomputador com palavras de 16 bits e uma memória de 32K, serão necessários 15 bits para um endereço direto, deixando apenas 1 bit para o código OP — uma situação intolerável.
2. *Conveniência do programador.* Há diversas técnicas de endereçamento (tais como registros-índice, que serão examinados) que proporcionam uma conveniência para o programador na escrita de programas.
3. *Facilidades de operação do sistema.* Na maioria dos sistemas de computador de grande porte, o computador terá vários programas diferentes em sua memória num determinado instante e alternará a passagem destes programas. A fim de carregar eficientemente e remover estes programas da memória em locais diferentes, são fornecidas técnicas de endereçamento que tornam os programas *relocalizáveis,* o que significa que o mesmo programa pode ser passado em muitas seções diferentes da memória.

As seções seguintes descrevem várias das técnicas em uso comum que são variações do método de endereçamento direto.

10.4 REGISTROS-ÍNDICE

Há uma variação no endereçamento direto convencional da memória que facilita a programação, particularmente a programação de seqüências de instruções que devem ser repetidas muitas vezes em conjuntos de dados distribuídos por toda a máquina. Esta técnica é chamada *indexação*.

A indexação foi usada pela primeira vez num computador desenvolvido na Universidade de Manchester. Um registro denominado *caixa B* foi acrescentado à seção de controle. O conteúdo da caixa B podia ser acrescentado ao conteúdo do registro de endereços da memória quando desejado. Quando a caixa B era usada, o endereço do operando localizado na memória devia estar no endereço escrito pelo programador mais o conteúdo da caixa B. O computador Manchester tinha apenas uma caixa B, mas se comprovou que a idéia era tão valiosa que muitos computadores modernos possuem várias. O termo americano que corresponde à caixa B é *registro-índice*, e este termo será usado.

O uso dos registros-índice facilita a escrita de programas que processam dados em tabelas, tais como as descritas no Cap. 1, reduzindo grandemente o número de instruções requeridas num programa iterativo. Os registros-índice permitem a modificação automática dos endereços consultados sem que se altere as instruções armazenadas na memória.

Quando os registros-índice são incluídos numa máquina, geralmente é feita uma divisão adicional na palavra de instrução do computador. Esta seção da palavra de instrução diz ao computador se deve ser usado um registro-índice e, se for o caso, que registro-índice usar. A palavra básica de instrução é portanto dividida, para um computador de endereço único, em três partes em vez de duas. Uma divisão típica é mostrada na Fig. 10.3. Duas instruções adicionais também são acrescentadas; uma é usada para "carregar" o registro-índice, e a outra modifica o número armazenado no registro-índice especificado ou faz com que o computador desvie.

Se não for usado um registro-índice, o programador coloca 0 na seção de designação do registro-índice da palavra. Se houver três registros-índice, haverá dois dígitos binários na seção de registro-índice da palavra, e o registro-índice desejado pode ser selecionado colocando-se os dígitos corretos na seção de designação do registro-índice.

Para descrever a operação dos registros-índice, duas novas instruções serão introduzidas. Designaremos uma destas pelo código mnemônico SIR (ajustar registros-índice), e esta instrução fará com que a seção de endereço da palavra de instrução seja transferida para dentro do registro-índice designado pelos bits de designação de registro-índice na palavra. Por exemplo, 01 SIR 300 carregará o número 300 dentro do registro-índice 01. Uma vez que o registro de endereços contém normalmente a seção de endereço da palavra do computador, basta que o conteúdo do registro de endereço seja transferido para dentro do registro-índice designado.

Designaremos a segunda instrução pelo código mnemônico BRI (desvio de índice). Esta instrução fará com que o conteúdo do registro-índice designado seja diminuído de um se o número armazenado no registro-índice for positivo. Ao mesmo tempo, o computador desviará para o endereço na seção de endereço da palavra de instrução, tirando sua próxima instrução daquele endereço. Se o registro-índice designado contiver um 0, o computador não desviará mas, em vez disso, executará a instrução seguinte na ordem normal.

Os registros-índice podem ser usados durante qualquer instrução normal colocando-se simplesmente os dígitos que indicam o registro-índice a ser usado na seção de designação de registro-índice da palavra do computador. Por exemplo, se o registro-índice 01 contiver 300 e escrevermos uma instrução de limpar e somar (CAD) como segue:

```
01 CAD 200
```

o computador somará o conteúdo do registro-índice 01 ao conteúdo do registro de endereços, e o endereço usado será o total destes dois. Uma vez que o registro-índice 01 contém 300 e o registro de endereços conterá 200, o endereço do qual será tirado o operando será o endereço 500 na memória.

Um exemplo do uso de um registro-índice pode ser achado no pequeno programa mos-

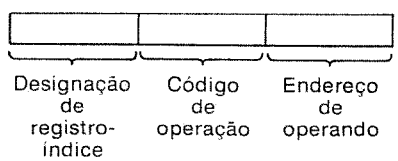


Fig. 10.3 Palavra de instrução do registro-índice.

QUADRO 10.1

Endereço na memória	Palavra da instrução			Comentários
	Designação do registro-índice	Código de operação	Seção de endereço	
0	01	SIR	99	O número 99 no registro-índice 01.
1	01	CAD	201	Apanha o número a ser adicionado.
2	00	ADD	301	Adiciona ao total até agora.
3	00	STO	301	Armazena a soma corrente.
4	01	BRI	1	Subtrai 1 do registro-índice 01 e depois desvia para a primeira instrução até que o registro-índice 01 contenha 0, em seguida prossegue para a instrução seguinte.
5	00	HLT	0	
.....				
201 a 300	Contêm números a serem adicionados.			
301	Local em que a soma é armazenada.			

trado no Quadro 10.1, que somará todos os números armazenados nos endereços de memória 201 a 300 e armazenará a soma no endereço 301.

O programa repete as intruções nos endereços 1 a 4 até que o registro-índice 01 esteja finalmente em 0; então o computador não desvia e é parado pela instrução seguinte.

10.5 PAGINAÇÃO E ENDEREÇAMENTO INDIRETO

Os preços baixos e a rápida acessibilidade dos circuitos integrados e memórias pouco dispendiosas deram origem ao mercado de computadores digitais pequenos de propósitos gerais chamados *microcomputadores* e *minicomputadores.*

Uma vez que o fabricante é pressionado a manter os custos baixos, várias estratégias engenhosas são usadas nestas máquinas. Primeiro, os comprimentos básicos da palavra de memória são geralmente curtos, com comprimentos de 8 a 16 bits. Se as palavras de instrução forem do mesmo comprimento que as palavras de memória, isto limita o comprimento da seção do código OP e, portanto, o número de diferentes instruções que podem ser usadas. Além disso, o comprimento curto da palavra significa seções curtas de endereço da memória nas palavras de instrução; portanto, técnicas especiais devem ser usadas para endereçar a memória inteira.

O projetista de microcomputador e de minicomputador atenua seu primeiro problema, um pequeno repertório de instruções, misturando engenhosas e complicadas instruções com as instruções convencionais. Por exemplo, o PDP-8 mínimo básico, um popular minicomputador fabricado pela Digital Equipment Corporation, tem uma palavra de instrução de 12 bits com um código OP de 3 bits. A máquina tem uma única instrução aritmética convencional, adição de complemento de 2. A subtração deve então ser programada e consiste da formação do complemento de 2 do subtraendo e depois adicionando. (Isto é facilitado por uma instrução que forma o complemento de 2 de um número no acumulador.) A multiplicação e a divisão também devem ser programadas. Uma única instrução lógica, E, é usada em conjunção com uma instrução de complemento (onde cada bit no acumulador é complementado) e a instrução E para formar OU exclusivo, OU inclusivo e todas as outras operações lógicas. Uma instrução de "incrementar e saltar se zero" que adicionará um a um local de memória selecionado e depois "saltar se um 0" é usada para formar registros-índice nos locais da memória. Várias instruções de deslocar em círculo ou girar o acumulador completam o repertório básico, junto com as requeridas instruções de entrada-saída. Este conjunto de instruções habilmente elaboradas não é somente adequado a todas as

finalidades, mas é altamente eficiente na execução da maioria das operações. A palavra de instrução desta máquina será discutida numa seção subseqüente.

O computador acima, embora bem projetado e bastante popular, não é nem único nem particularmente novo.† A maioria dos pequenos computadores combina os relativamente pequenos conjuntos de instruções com organizações engenhosas para formar computadores eficientes a despeito das dimensões pequenas. Os preços pagos pelos pequenos números de instruções numa máquina são, principalmente, o tempo de operação aumentado nos programas e maior talento (ou engenhosidade) requerido aos programadores de sistemas que escrevem os montadores e compiladores. (O usuário real não é geralmente bastante afetado pelos compiladores.)

Os microcomputadores e minicomputadores muitas vezes aliviam o problema de endereçamento de uma grande memória com uma palavra curta usando uma técnica que realmente surgiu num grande computador chamado Atlas. Esta técnica é chamada *paginação*. Quando é usada a paginação, a memória é dividida em *páginas*, cada uma das quais é de comprimento fixo.

Exemplo Para o PDP-8 mencionado acima, a memória de núcleos básica de 4096 palavras de 12 bits cada uma é dividida em 32 páginas de 128 palavras cada uma. Assim, a página 0 contém os locais de memória de 0 a 127 (decimal), a página 1 se refere aos locais de memória de 128 a 255, ... e, finalmente, a página 31 se refere aos locais de 3968 a 4095, conforme indicado na Fig. 10.4. Então são necessários 5 bits para consultar uma página e 7 bits para consultar um local dentro de uma página.

O endereçamento de dados num computador com paginação varia de computador a computador. Geralmente, o endereço dado na palavra de instrução pode referir-se ou à página em que a palavra de instrução permanece ou possivelmente a alguma outra página em particular, previamente especificada.

Exemplo No PDP-8, o quarto bit numa palavra de instrução de 12 bits é chamado *bit de página*. Este bit diz se o endereço de 7 bits na palavra de instrução se refere à página em que permanece a instrução, em cujo caso o bit é um 1, ou à primeira página na memória, em cujo caso o bit é um 0. Por exemplo, se o quarto bit for um 1 e o endereço dado for 23 e a palavra de instrução estiver na página 20, então o local real consultado é o 23.º na página 20 ou a 2583.ª palavra na memória.§ Se o quarto bit for um 0, o endereço 20 simplesmente se refere ao local 20 na memória.

O exemplo de paginação acima possibilita o uso de endereços que contêm menor número de dígitos do que normalmente seria necessário para endereçar a memória. Geralmente, isso é adequado uma vez que um programa é mais provavelmente para endereçar ou locais próximos ou alguns dados numa área particular. Quando for necessário endereçar fora dessas áreas, são usadas ou uma técnica chamada *endereçamento indireto* ou palavras de instrução especiais mais longas.

Muitas vezes, a paginação é relativa à palavra de instrução, em cujo caso a parte de endereço diz quantos endereços à frente (ou atrás) da palavra de instrução permanece a palavra do operando. Assim, se a palavra de instrução estiver no endereço 960, a parte de endereço tem valor 140, e a paginação relativa foi usada, o endereço consultado deve ser o endereço 1100. O Data General Computers, DEC PDP-11, o Lockheed Sue e muitos outros computadores usam paginação de endereço relativo. O endereço real é formado adicionando-se simplesmente o conteúdo do contador de instrução à seção de endereço da palavra de instrução.

Uma outra variação amplamente usada na técnica de endereçamento convencional é o chamado *endereçamento indireto*. Quando usado, o endereçamento indireto faz com que a palavra de instrução dê o endereço, não do operando a ser usado, mas do endereço de um local onde um outro endereço é armazenado. Por exemplo, se escrevermos ADD 302 e a instrução for uma instrução convencional de adição de endereçamento direto, a palavra no local 302 será adicionada à palavra normalmente no acumulador.

Se a instrução de adição for endereçada indiretamente e escrevermos IAD 302 (IAD para *soma indireta*), então o número armazenado no endereço 302 dará o endereço real do operando a ser usado. Por exemplo, quando a palavra de instrução no endereço 5 na memória no seguinte programa for executada, ela fará com que o número 164 seja adicionado ao conteúdo normal do acumulador.

---

†No entanto, ele era altamente original e bem desenvolvido, e estabeleceu um padrão para todos os minicomputadores que se seguiram.

§A página 20 começa com o local 20 × 128 = 2560, e o local referido será então 2560 + 23 = 2583.

ENDEREÇO DA MEMÓRIA	CONTEÚDO
5	IAD 302
...	...
302	495
...	...
495	164

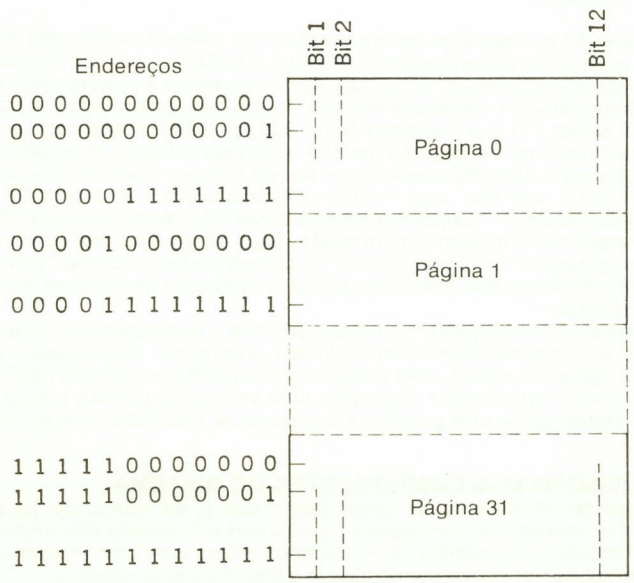
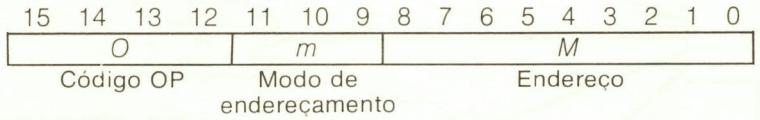


Fig. 10.4 Esboço da memória de 31 páginas com 4096 palavras de 12 bits.

Por exemplo, consideremos que sejam usados tanto registros-índice quanto endereçamento indireto em muitos dos mais modernos computadores. Assim, o Varian Data Model 6201 é um pequeno computador de propósitos gerais com comprimento básico de palavra de 16 bits que usa complementos de 2 para os números negativos. A palavra básica de instrução é como segue:



A seção de código OP diz que instrução deve ser executada; os *M* bits dão um endereço básico da memória de 9 bits, enquanto que os três *m* bits dizem o *modo* de endereçamento.

Quatro registros básicos podem ser usados nos vários modos de endereçamento. Estes são o contador de instruções, IC (o qual o Varian chama registro *P* ou contador do programa), o registro *B*, que é a "metade inferior" do acumulador, como na Fig. 9.3, e um registro *X*, que é um registro-índice.

Os *m* bits dizem então o modo de endereçamento como segue:

m = 000 O endereço do operando é dado diretamente pelo número binário *M* de 9 bits.
m = 100 O endereço do operando é achado adicionando-se o valor de *M* ao valor corrente no contador de instruções.
m = 101 O endereço do operando é achado adicionando-se o valor de *M* a *X*, o valor do registro-índice.
m = 110 O endereço do operando é formado adicionando-se o valor de *M* a *B*, o valor no registro *B*.
m = 111 O endereçamento é indireto. O endereço do operando é armazenado no local dado por *M*.

Há uma variação no esquema do endereçamento indireto usado pelo Varian, Data General e muitos outros. No endereçamento indireto, o endereço no local *M* acima é um endereço de 15 bits mais um bit de sinal que diz se o endereço é o endereço do operando ou o endereço de ainda um outro endereço indireto. Assim, se a palavra de instrução disser ADD e os bits 9 a 11 forem 111 e o endereço *M* for 320 (decimal), então, se o endereço 320 da memória contiver um 0 no bit de sinal, os bits restantes conterão o endereço do número a ser adicionado ao valor atual do acumulador. Se, no entanto, o bit de sinal no endereço 320 da memória for um 1, isto diz "usar os bits restantes como o endereço do local seguinte para procurar um endereço." Suponhamos que estes bits dêem o número 325; o conteúdo deste local da memória é tratado similarmente. Se o bit de sinal em 325 for um 0, os bits restantes dão o endereço do operando a ser adicionado ao acumulador; se o bit de sinal for um 1, o endereço formado dos bits restantes dá o endereço do próximo local a procurar o endereço do operando.

Este esquema de endereçamento indireto permite o *encadeamento* indefinido de endereços indiretos e é surpreendentemente útil em programas. Observamos que, quando o endereçamento indireto é usado, uma palavra inteira pode ser utilizada como um endereço; isto é muito útil num computador paginado, uma vez que a palavra inteira será maior do que a parte de endereço de uma palavra, e a maioria, se não toda, da memória será endereçável.

10.6 INTERLIGAÇÃO DOS COMPONENTES DO SISTEMA

Os componentes de um sistema de computador, isto é, as memórias, os dispositivos de entrada-saída, etc., devem ser interligados para formar um sistema de computadores. A maneira pela qual esses componentes são colocados juntos e se comunicam uns com os outros afeta profundamente as características de desempenho do sistema.

As unidades aritmética-lógica e de controle são, em geral, colocadas fisicamente juntas e denominadas unidade central de processamento (UCP). A UCP está então "encarre-

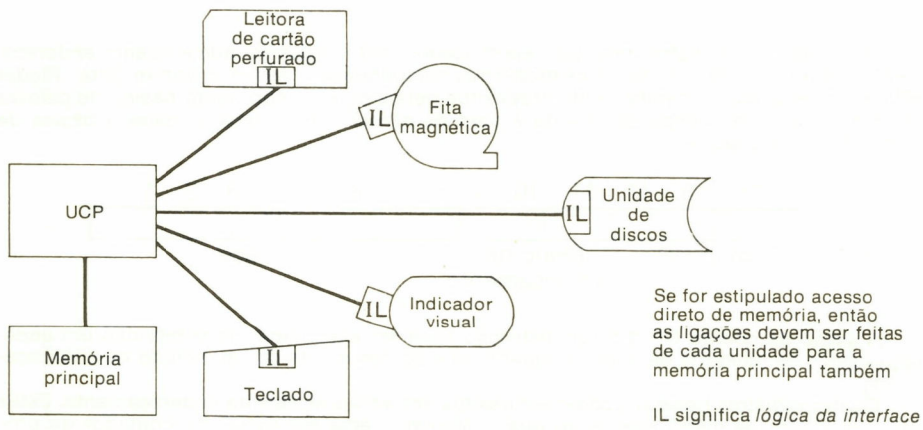


Fig. 10.5 Ligações individuais entre as unidades do sistema de computador.

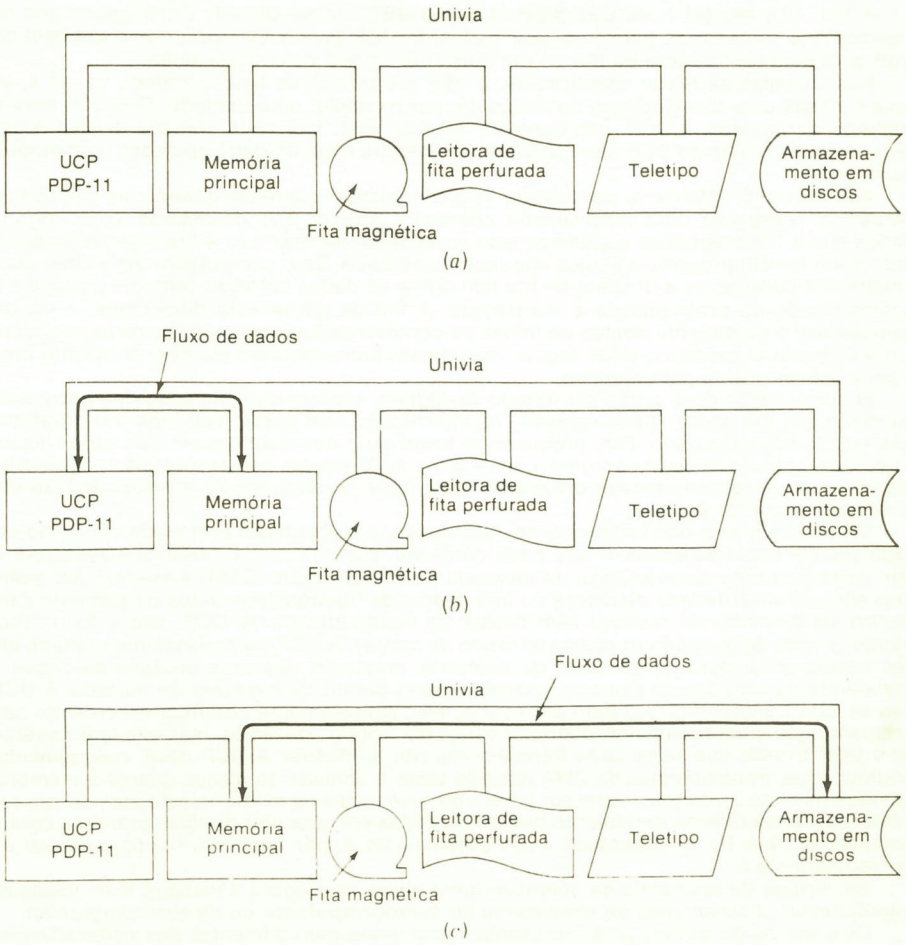


Fig. 10.6 Barramento para o PDP-11. *(a)* UCP, memória e outros dispositivos são conectados por um barramento único; *(b)* a UCP controla as transferências de dados em uso normal; *(c)* quando o sistema tem acesso direto de memória (DMA), os periféricos podem transmitir dados diretamente à memória no barramento. (Cortesia da Digital Equipment Corp.)

gada" da operação do sistema, dirigindo a operação das outras partes do sistema.

Nos primitivos computadores e em alguns modernos computadores menores, a UCP está conectada diretamente a cada dispositivo de entrada-saída e à unidade de memória usando um cabo separado para cada ligação. Isso é mostrado na Fig. 10.5. Então, se um cartão deve ser lido numa leitora de cartões, a UCP deve aceitar a informação e, se ela deve ser armazenada na memória, a UCP deve armazená-la. A UCP é, portanto, central e envolvida diretamente em cada transação.

O sistema acima apresenta as desvantagens de muitos cabos diferentes (ou vias) e considerável lógica de interface (em cada extremidade de cada cabo).

A fim de tornar a interligação dos componentes do sistema menos dispendiosa e padronizar a lógica de interface utilizada, uma técnica muito popular atualmente consiste na interligação de todos os componentes usando uma única *via*. Esta via consiste de vários fios ou ligações, e, na via, existem provisões para endereçamento dos componentes e transferência de dados de ou para cada componente.

A Fig. 10.6 mostra a via DEC PDP-11 que a DEC chama UNIVIA. Observemos que os mesmos fios são usados para transferir dados da UCP para a memória como também da UCP para uma perfuradora de fita ou um outro dispositivo de entrada-saída.

Nos sistemas de maior simplicidade, a UCP é o diretor de todo o tráfego na via e, se deve ser feita uma transferência de dados de, por exemplo, uma unidade de discos para a memória de núcleos, a UCP, sob controle do programa, lerá cada unidade de dados em seus registros gerais de UCP e em seguida armazenará cada unidade de dados na memória de núcleos.

Aqui há um problema na capacidade de o computador conhecer quando um dispositivo periférico† executou uma determinada operação. Admitir que desejamos achar alguns dados numa fita magnética e estamos sem vontade de esperar que a fita seja pesquisada, desejando executar outros cálculos enquanto esperando. Se o computador deve olhar continuamente para ver se a unidade de fita tem agora os dados corretos, tempo é perdido e a complexidade da programação é aumentada. A fim de aliviar esta dificuldade, a via do computador é geralmente dotada de linhas de controle denominadas *linhas de interrupção*, e um dispositivo periférico pode erguer uma destas linhas quando ela tiver concluído uma ação e estiver pronta para atenção.

O computador deve então ser dotado de alguma espécie de instalação de interrupção de modo que ele possa "fazer reparos" na interrupção sem perder seu lugar no programa que está sendo executado. Este problema se torna sério nos sistemas em que vários dispositivos de entrada-saída (tais como conversores A/D) devem ser pesquisados freqüentemente, e os computadores são projetados para fazer reparos nestas interrupções tão eficientemente quanto possível.

Mesmo com uma boa instalação de interrupção, o computador está ainda envolvido em cada transferência de dados, e isto pode consumir muito tempo. É possível acrescentar o que se chama uma característica de memória de *acesso* direto (DMA) à maioria dos sistemas em que uma unidade de discos ou uma leitora de fita transfere dados diretamente para dentro da memória de núcleos sem passar os dados através da UCP. Isto é feito "roubando" ciclos da memória (o chamado *roubo de ciclos*). A UCP é simplesmente mantida em seu estado atual durante um ciclo de memória, enquanto que uma unidade de dados é transferida da unidade de discos diretamente para dentro da memória de núcleos. A UCP não vê esta transferência quando ela ocorre, mas simplesmente continua executando seu programa que é um pouco retardado por causa do "roubo" de ciclos, mas não tanto quanto se a UCP tivesse que fazer cada transferência por si mesma. A UCP deve, naturalmente, originar estas transferências de DMA dizendo tanto à unidade de discos quanto à memória de núcleos onde os dados devem ser lidos e em que locais da memória os dados devem ser lidos. (As transferências geralmente podem ser feitas em qualquer direção quando a característica de DMA for acrescentada a um sistema, isto é, por exemplo, fita-para-núcleo ou núcleo-para-fita.)

Em virtude da economia da operação, uma única via é agora a maneira mais usada de interconectar componentes de um sistema de microcomputador ou de minicomputador.

Os sistemas de grande porte apresentam problemas bem diferentes dos sistemas muito pequenos, e assim diferentes configurações de interligação costumam ser usadas. Uma vez que os grandes sistemas contêm muitos componentes, eles são muito dispendiosos e é importante utilizar a UCP e outros componentes ao máximo, e assim o custo de configurações de interligação mais dispendiosas está justificado.

Em conseqüência, a fim de manter as grandes configurações de processador, tal como a indicada na Fig. 10.7, trabalhando em sua velocidade máxima, os sistemas são operados no que se chama um *modo de multiprocessamento*. Isto significa que vários programas são mantidos na memória ao mesmo tempo. Um determinado programa é então executado até que ele demande um dispositivo de entrada-saída ou talvez uma unidade de discos. Uma vez que esses dispositivos são lentos quando comparados à UCP, o dispositivo é iniciado em sua função, mas a UCP depois começa a executar um outro programa diferente até que este programa solicite para entrada-saída. Quando isto acontece, a UCP começa a executar um terceiro programa, e este processo continua.

Quando um programa conclui a execução, um outro programa é lido. Como pode ser visto, a UCP não perde de vista onde ele está em cada programa, e deve controlar todas as transferências de dados entre os componentes do sistema, mas deve ser irremediavelmente mantida erguida se tiver que participar em cada transferência.

---

†*Dispositivos periféricos* são os dispositivos de entrada-saída, unidades de disco, unidades de fita e outros dispositivos que não incluam a memória principal (de núcleos ou de CI).

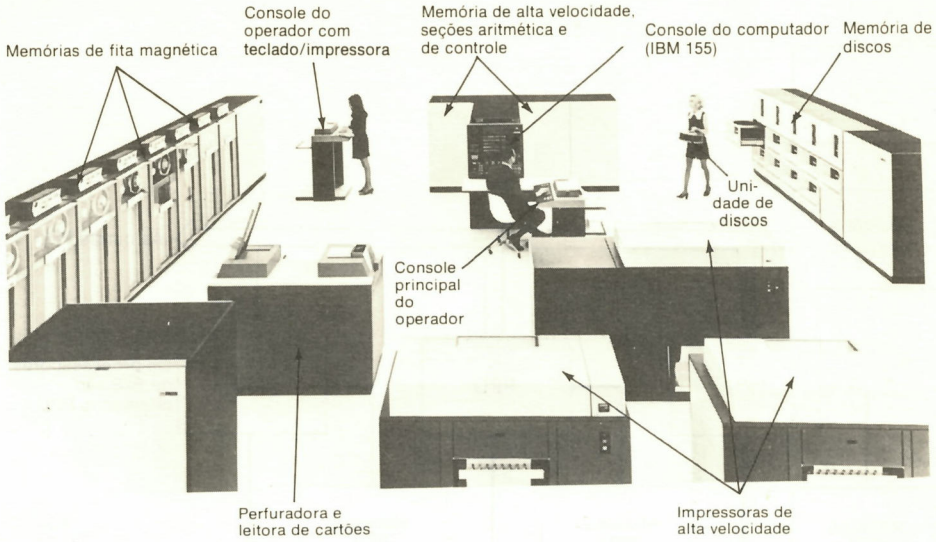


Fig. 10.7 Partes de um grande computador de propósitos gerais.

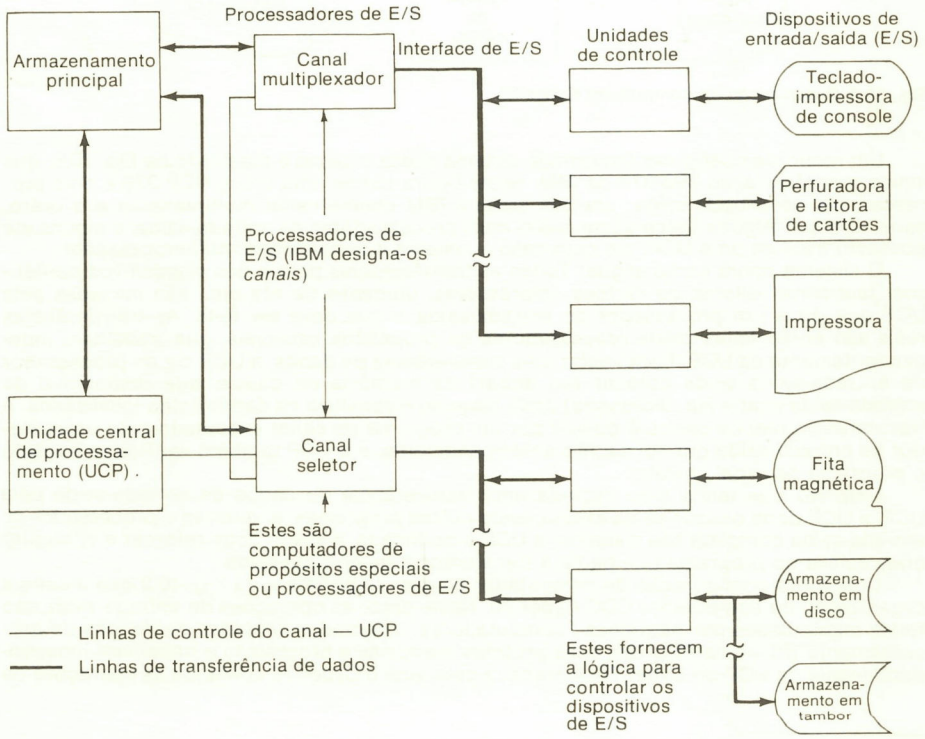


Fig. 10.8 Organização da série de computadores 360/370 da IBM.

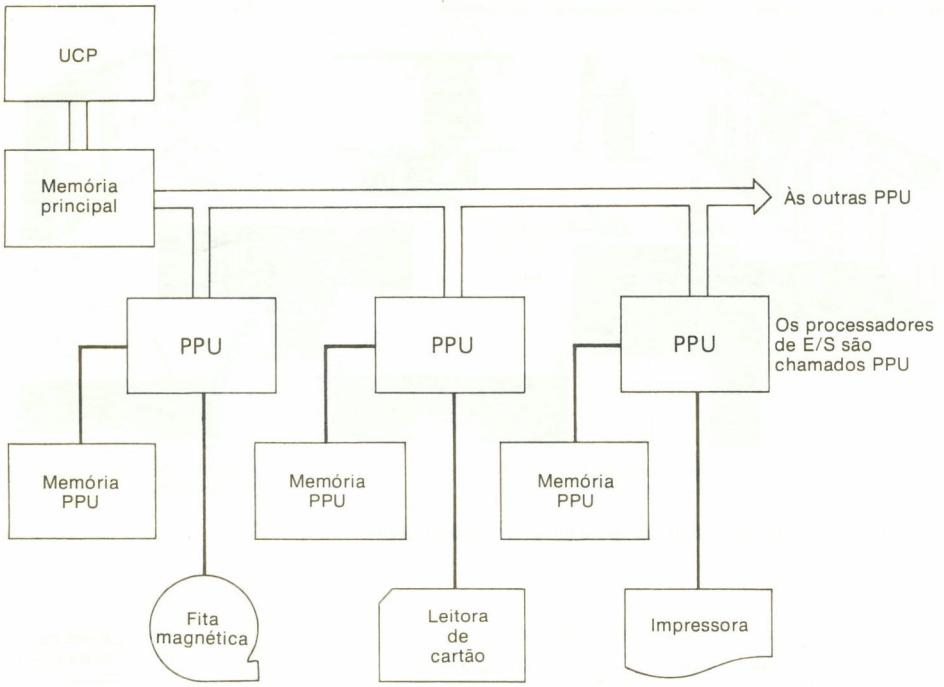


Fig. 10.9 Organização do computador Cyber 70.

Um modo de configurar um grande sistema dessa espécie é mostrado na Fig. 10.8, que indica a configuração 360/370 da IBM. Nesta figura temos uma única UCP 370 e dois processadores de entrada-saída; um dos quais a IBM chama canal multiplexador e o outro, canal seletor.† Alguns sistemas possuem mais processadores de entrada-saída, e alguns até possuem mais de uma UCP, em cujo caso o sistema é denominado *multiprocessador.*

O sistema opera como segue: Todas as transferências para e dos *dispositivos periféricos,* tais como leitoras de cartões, impressoras, unidades de fita etc., são iniciadas pela UCP que diz a um processador de entrada-saída o que deve ser feito. As transferências reais são então feitas pelos processadores de propósitos especiais, que trabalham independentemente da UCP. Para iniciar uma transferência de dados, a UCP diz ao processador de entrada-saída onde colocar (ou achar) na memória os dados, que dispositivo de entrada-saída usar e (se necessário) onde naquele dispositivo os dados estão localizados. A transferência real de dados é guiada por um *programa de canal* executado pelo processador de entrada-saída que foi escrito antecipadamente, e a UCP também verifica se é usado o programa de canal correto.

Uma vez que tenha sido iniciada uma transferência de dados de entrada-saída pela UCP, a UCP pode ocupar-se da execução de outros programas, e, quando o processador de entrada-saída completa seu trabalho, a UCP é notificada, e assim pode retornar e continuar onde estava no programa que pediu a transferência de entrada-saída.

Uma outra configuração de computador de grande porte é a da Fig. 10.9 que mostra a organização do computador CDC Cyber 70. Neste caso, as operações de entrada-saída são todas manipuladas por pequenos "computadores" chamados *unidades periféricas de processamento* (PPU). As PPU têm suas próprias memórias e programas e trabalham independentemente da UCP do Cyber 70. A maneira pela qual o Cyber 70 comanda as operações de

†Um canal multiplexador tem lógica particularmente adequada aos dispositivos relativamente lentos com características aleatórias. Os canais seletores são para rápidos rompimentos de dados gerados por unidades de disco e unidades de fita, por exemplo.

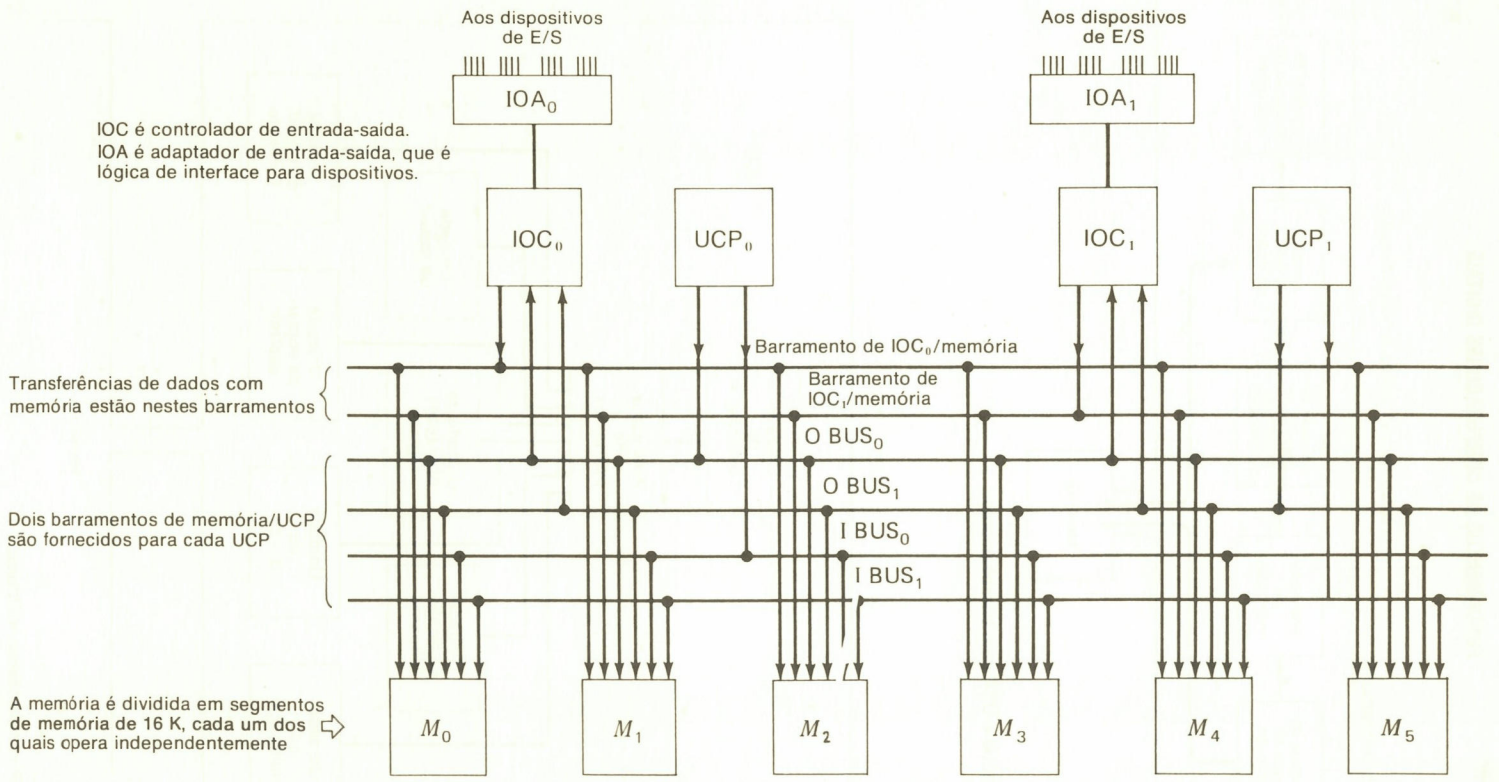


Fig. 10.10 Multiprocessador AN/UYK-7 da Univac.

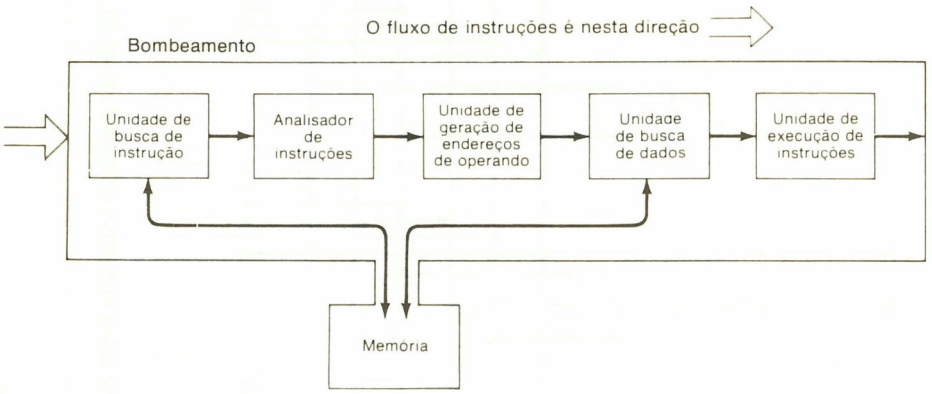


Fig. 10.11 Memória e UCP com bombeamento.

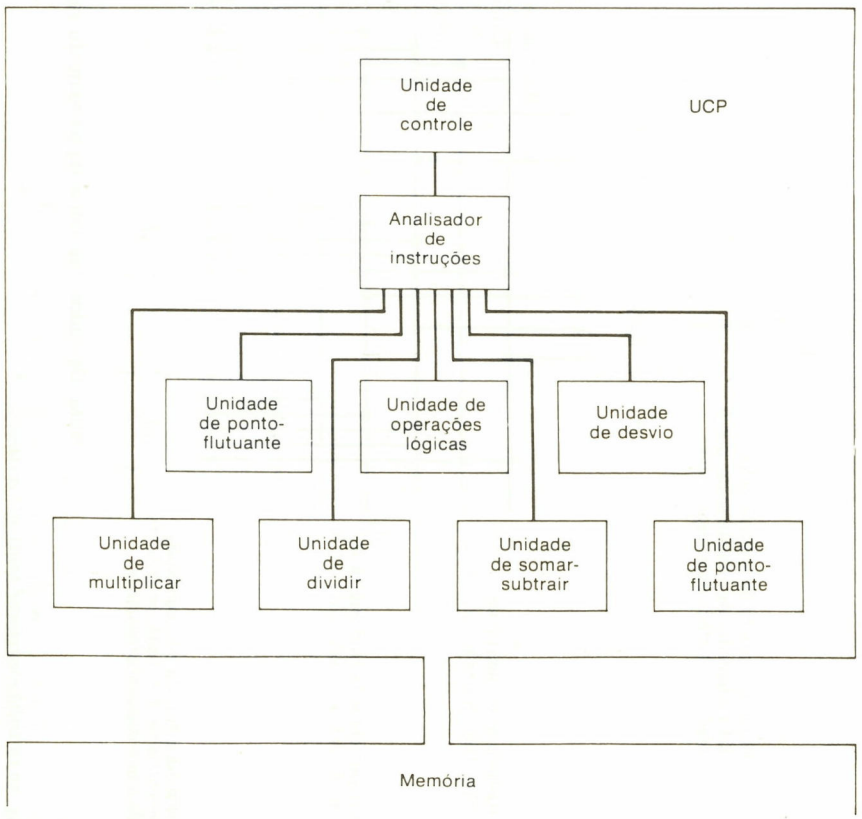


Fig. 10.12 UCP com unidades múltiplas.

entrada-saída consiste em "plantar" mensagens numa área específica de sua memória, dizendo o que ele gostaria. As PPU então pesquisam esta memória procurando ordens, e, quando elas acham alguma, elas executam as operações necessárias e plantam uma mensagem dizendo à UCP que suas ordens foram cumpridas e as operações necessárias realizadas.

A razão pela qual as estruturas complexas tais como o 360/370 da IBM e o Cyber 70 são bem compreensíveis é que vários dispositivos periféricos ou de entrada-saída podem estar operando simultaneamente nas velocidades relativamente baixas que eles podem manter, enquanto a UCP corre de serviço a serviço. Deste modo, a produtividade global do sistema de computadores pode ser aumentada por causa da operação paralela de todas as partes do sistema.

A idéia de fornecer várias UCP que executam programas em paralelo é uma idéia atrativa nos grandes sistemas. Tal como foi mencionado, tais sistemas são chamados *sistemas multiprocessadores,* e estes sistemas novamente têm alta produtividade e fazem boa utilização tanto das memórias grandes quanto de muitos dispositivos de entrada-saída freqüentemente achados em muitos sistemas grandes. A Fig. 10.10 mostra um tal sistema.

Há diversos problemas que têm a ver com os sistemas multiprocessadores e que conduzem a outras estratégias. A mais conhecida de tais estratégias consiste em construir uma única "super" UCP com o conjunto de circuitos o mais rápido possível e depois tentar forçar instruções através desta UCP a uma alta velocidade. Quando a UCP contiver um conjunto de circuitos lógicos que operam simultaneamente sobre várias instruções em ordem, efetuando operações em diferentes etapas da execução de cada instrução que usa uma unidade ao mesmo tempo, conforme indicado na Fig. 10.11, então o computador é chamado *computador de bombeamento.* Os computadores CDC Star, Amdahl e o MU5 da Universidade de Manchester constituem exemplos desta espécie de arquitetura.

Em alguns sistemas de computadores, a UCP simplesmente tem várias unidades diferentes, cada uma das quais pode executar uma categoria diferente de instrução (um multiplicador, um divisor, uma unidade de ponto-flutuante, por exemplo; ver Fig. 10.12). Um código de OP de instrução é então decodificado e a instrução transmitida à unidade apropriada para execução. Então, várias instruções podem ser executadas em paralelo por estas diferentes unidades. A CDC produz diversos computadores (incluindo o Cyber 70) que operam desta maneira. Estes computadores e os computadores de canalização têm todos circuitos lógicos suficientes para considerar a interação entre as instruções (porque uma instrução que precede uma outra instrução pode alterar o operando da segunda instrução, por exemplo) e verificar se os cálculos estão corretamente efetuados. Isto conduz a grandes e dispendiosas UCP, mas, em virtude de estes sistemas serem muito grandes e já dispendiosos, o preço pode ser pago.

10.7 DISPOSITIVOS DE INTERFACE — BARRAMENTOS OU VIAS

Quando os dispositivos de entrada-saída, os dispositivos de memória, a unidade aritmética-lógica e a unidade de controle forem todos combinados para formar um sistema de computador, todos estes devem ser conectados ou interligados. Quando um dispositivo ou unidade for conectado a um outro, é necessário uma interface que inclui a lógica necessária.

As principais desvantagens da utilização de um grande número de cabos individuais para interconectar partes de um sistema são o custo e a complexidade. A lógica da interface necessária deve ser repetida para cada conexão, junto com os circuitos de excitação dos cabos e com os circuitos de recepção.

Uma técnica amplamente usada para interligar os módulos eficientemente e com pequeno custo usa uma única via† (ou barramento) para interconectar todas as unidades. Isto é mostrado na Fig. 10.13, onde várias linhas ou condutores que formam o barramento passam através de e se conectam a inúmeras unidades ou módulos. Em geral, cada módulo pode ler do barramento ou escrever no barramento. A interface do barramento é geralmente padronizada, uma vez que o mesmo barramento conecta todas as unidades. Desde que cada unidade se conecta apenas uma vez ao barramento, a quantidade de lógica e de circuitos de interface requerida tende a ser mais baixa do que as ligações separadas entre as unidades. Em conseqüência, os barramentos são amplamente usados em microcomputadores e em minicomputadores e até em grandes sistemas de computadores com módulos onde o fluxo de dados não é excessivo.

†Um barramento (ou via) é um conjunto de linhas paralelas ou de conexões. A maioria dos barramentos usa fios para as ligações, embora alguma pesquisa tenha sido feita com feixes de luz e outros meios.

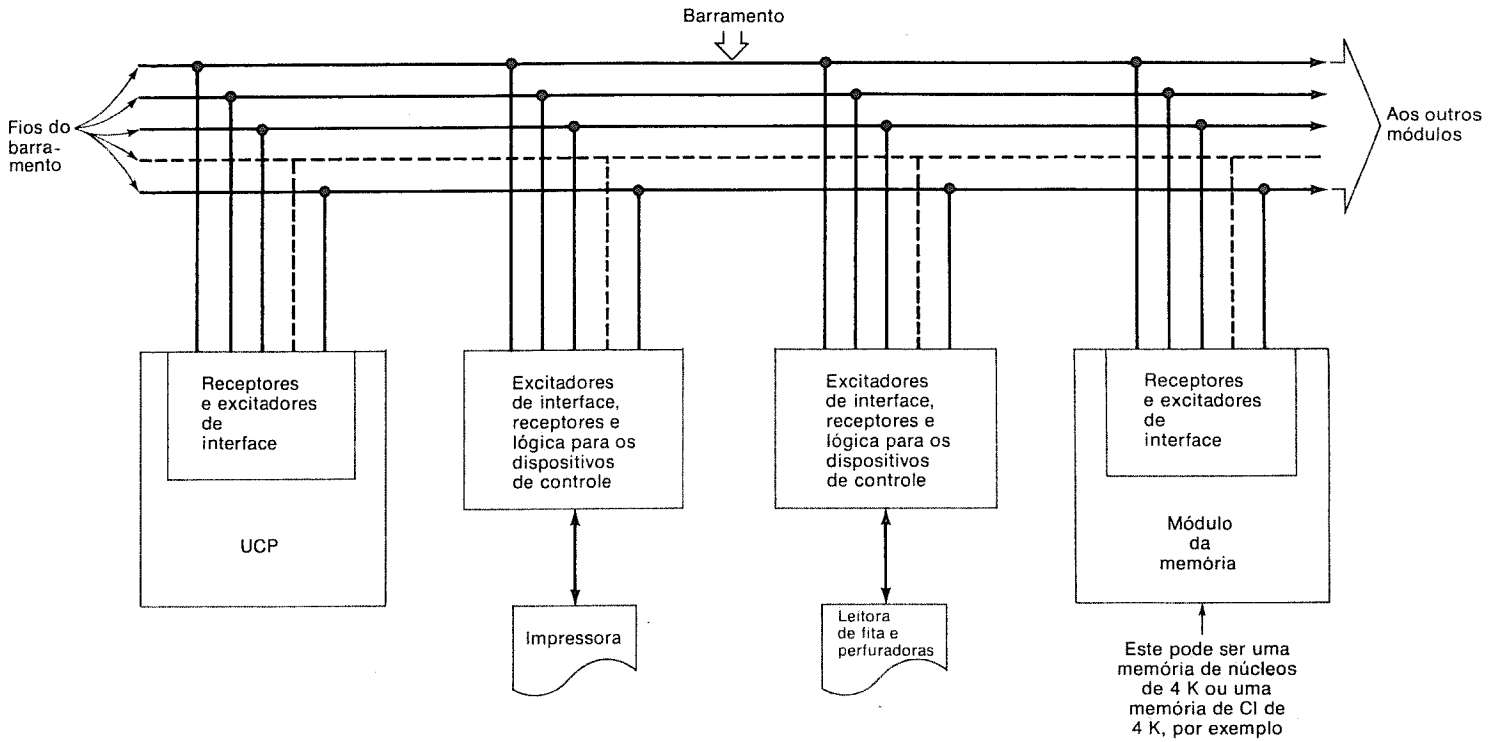


Fig. 10.13 Organização de computador do sistema de barramento único.

Uma vez que os módulos que são interligados devem partilhar as mesmas linhas de dados, é necessário que cada módulo seja capaz tanto de escrever quanto de ler num determinado fio. Há três técnicas gerais com esta finalidade, e elas estão indicadas na Fig. 10.14. (Algumas das considerações elétricas são tratadas nos exercícios.)

Na Fig. 10.14(*a*), a ligação de fio a ser partilhada está normalmente no estado alto. Um certo módulo pode escrever neste fio, forçando o fio para o nível de terra ou nível 0. Um circuito típico de escrita ou excitador é mostrado na figura. Cada módulo também tem um receptor, que é um simples inversor. (Muitas vezes, o estado alto no fio do barramento é feito uma lógica 0, e o estado baixo, um 1.)

Na Fig. 10.14(*b*), a ligação de barramento está normalmente no nível de terra. O circuito ou excitador de escrita do módulo escreve níveis altos (de lógica 1) nesta linha, deixando a linha baixa quando os 0 devem ser escritos. O receptor pode consistir ou de um amplificador de não-inversão ou de dois inversores em série.

Algumas linhas lógicas fornecem os chamados *excitadores de linha de três estados.* Estes circuitos têm uma entrada de *desativar* bem como uma entrada de nível lógico, conforme indicado na Fig. 10.14(*c*). Quando a entrada de desativar for baixa (um 0), o circuito excita o fio de barramento no nível da entrada de lógica. Portanto, se a entrada de desativar for um 0 e a entrada de lógica um 0, o fio de barramento será excitado em 0. Se a entrada de desativar for um 0 e a entrada de lógica um 1, o fio do barramento será excitado alto. Se a entrada de desativar for um 1, no entanto, o circuito apresenta uma impedância muito alta ao fio do barramento, que permite que um outro excitador de linha de três estados excite a linha do barramento para qualquer que seja o nível selecionado. Os receptores são novamente circuitos lógicos convencionais ou inversores. Os fabricantes de TTL, ECL, CMOS e outras linhas de circuito geralmente incluem os excitadores de linha de três estados em suas linhas de embalagens de CI.

Uma vez que várias unidades estão partilhando as mesmas linhas de barramento, os procedimentos de interface de módulos interligados devem ser cuidadosamente achados de modo que, por exemplo, dois módulos não tentem escrever dados no barramento ao mesmo tempo e de modo que, por exemplo, o módulo para o qual os dados são destinados saiba que ele é o módulo selecionado etc.

Várias organizações padronizadas (tais como a American National Standards Institute) incumbiram-se de desenvolver interfaces e procedimentos ou, como eles são freqüentemente chamados, *protocolos* para os sistemas digitais. Um protocolo amplamente usado e que foi desenvolvido para interligar instrumentos será descrito resumidamente, pois ele é representativo das interfaces achadas para os barramentos.

A Fig. 10.15 mostra a interface básica e as linhas de barramento que podem ser usadas para interligar vários módulos. Cada linha de barramento executa pelo menos uma função de interface, dependendo das capacidades da interface.

Num determinado instante, qualquer módulo particular conectado ao barramento pode estar inativo, monitorando a atividade no barramento ou funcionando como (1) um locutor, (2) um ouvinte ou (3) um controlador. Como um locutor, um módulo transmite dados sobre o barramento para um ouvinte (ou ouvintes). Como um ouvinte, um módulo recebe tais dados. Como um controlador, ele orienta o fluxo de dados sobre o barramento, principalmente designando que módulos devem transmitir dados e que módulos devem receber dados.

Observemos que o barramento consiste de 16 linhas de sinal, agrupadas funcionalmente em três barramentos componentes. O *barramento de dados* (oito linhas) é usado para transferir dados em paralelo, dos locutores aos ouvintes; ele também transfere certos comandos do controlador aos módulos subordinados. O *barramento de transferência* (três linhas) é usado no processo de confirmação de conexão, pelo qual um locutor ou controlador pode sincronizar sua prontidão ou receber dados. O *barramento geral de controle de interface* (cinco linhas), como seu próprio nome sugere, é principalmente usado pelo controlador.

Cada sistema deve ter um módulo para designar ouvintes e locutores, e este módulo é chamado *controlador.* O controlador usa um grupo de comandos, denominado *mensagens de interface,* para dirigir os outros módulos no barramento, levando a termo suas funções de conversa e escuta.

Normalmente, o controlador seria a UCP do computador, e esta unidade geraria os sinais de comando no barramento para os outros módulos que então responderiam. Em virtude de esta interface ser projetada para manipular um grande número de diferentes tipos de módulos, a especificação é razoavelmente complicada e geral. O procedimento básico para a geração de uma transferência de dados neste barramento é como segue, no entanto. Primeiro, o controlador designa um ouvinte, colocando o endereço do ouvinte (a

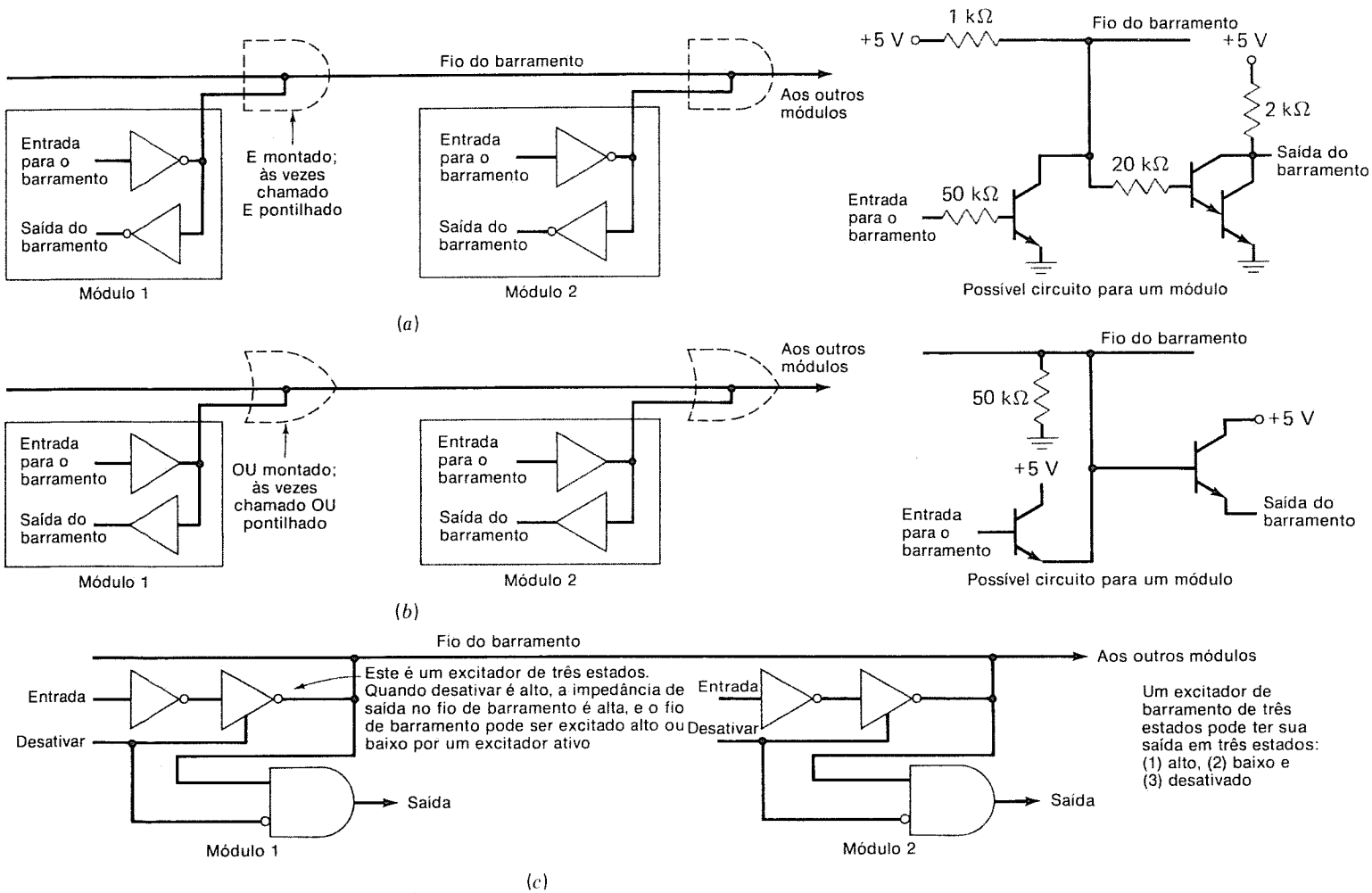


Fig. 10.14 Partilha de um fio de barramento. *(a)* Ligações de barramento usando E incorporado; *(b)* ligações do barramento usando OU incorporado; *(c)* excitador de barramento de três estados.

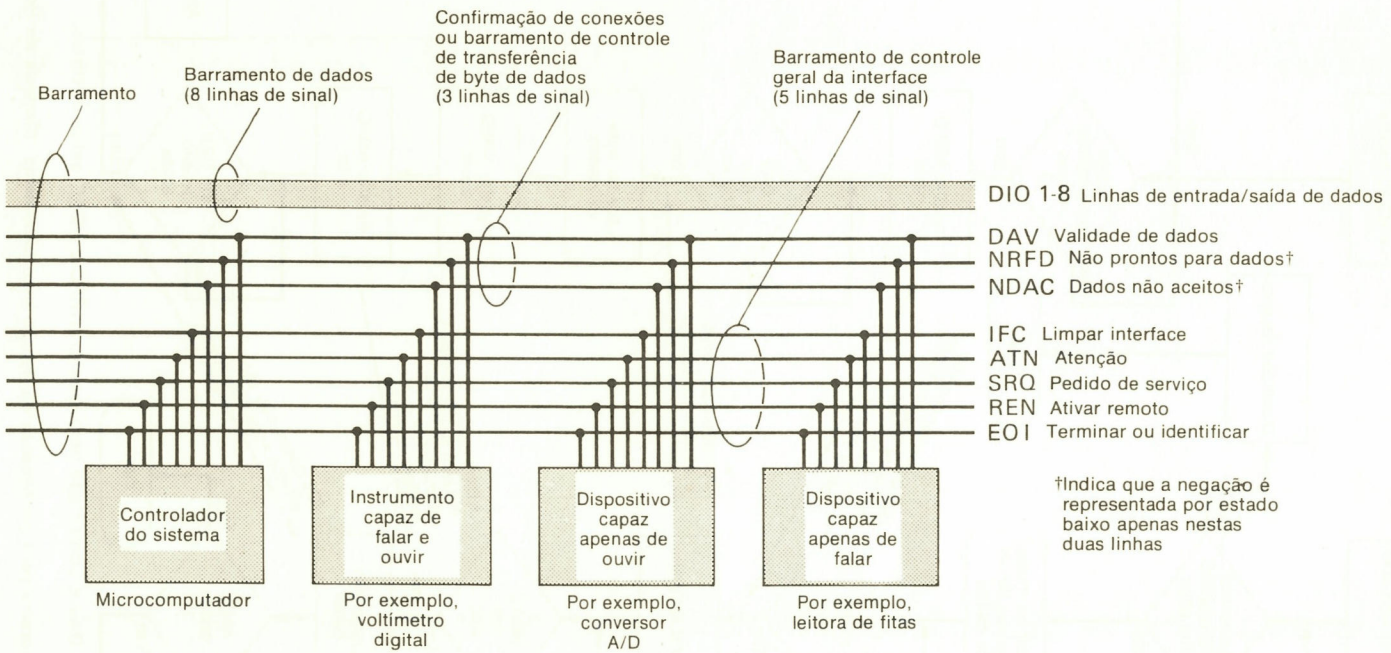
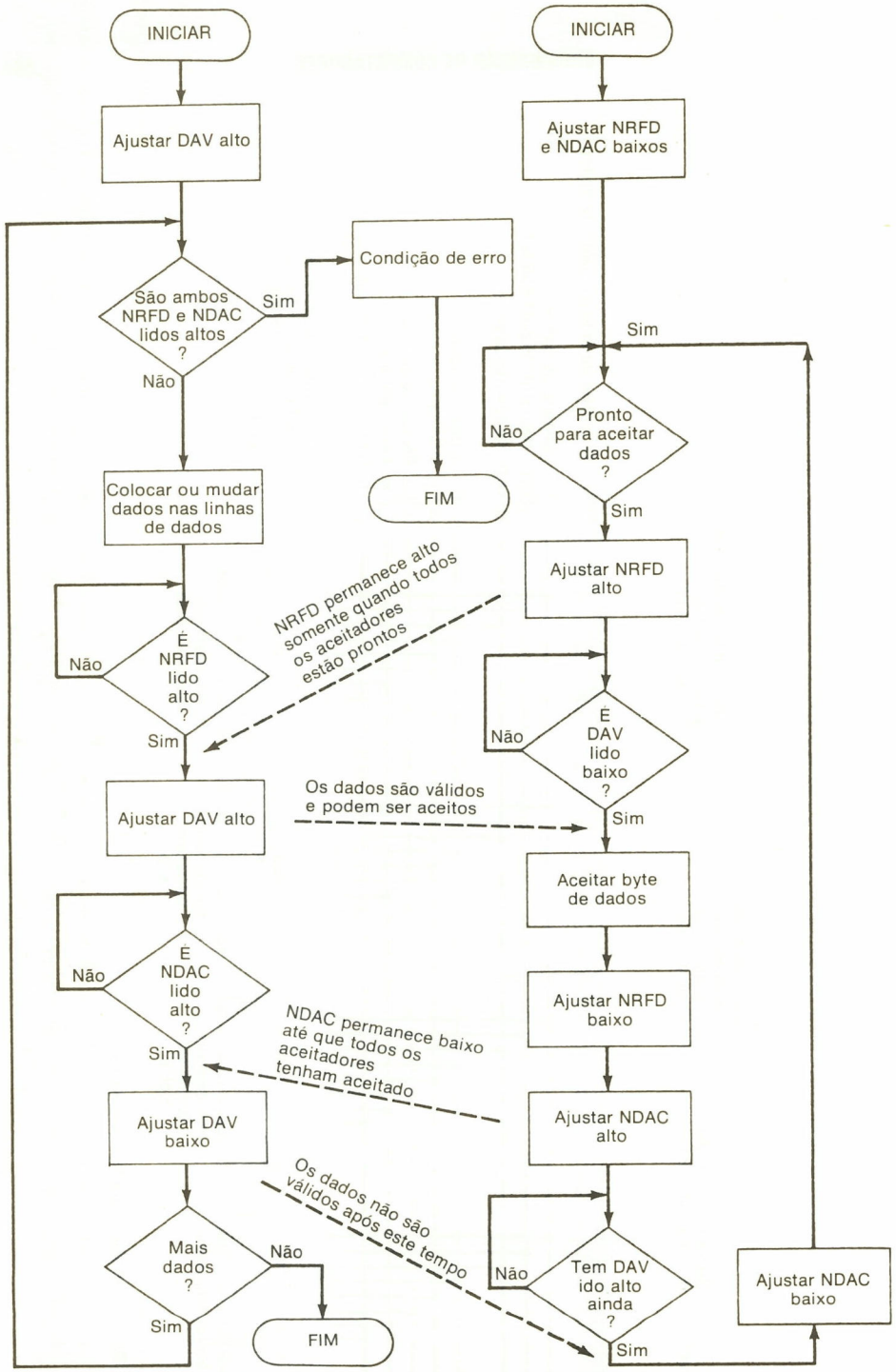


Fig. 10.15 Barramento padronizado internacional.



A linha DAV é controlada pelo locutor, NRFD e NDAC pelos ouvintes.

Fig. 10.16 Fluxograma para confirmação de ligação num barramento. (Cortesia da Hewlett-Packard Corp.)

cada ouvinte é atribuído um endereço de 5 bits) no barramento de dados e levantando as linhas de controle apropriadas. Um locutor é então designado, colocando o endereço do locutor nas linhas de dados e levantando as linhas de controle apropriadas. Finalmente, é dito ao locutor e ao ouvinte para prosseguirem e o locutor coloca os dados, 8 bits de cada vez, em paralelo nas linhas de dados.

Na transferência de dados do locutor ao ouvinte, surgem certos problemas básicos na operação de cada barramento.† Estes problemas são resolvidos por meio de um processo de confirmação de conexão pelo qual o locutor e o ouvinte interagem usando as linhas de controle. É conveniente descrever este procedimento usando um *fluxograma* conforme indicado na Fig. 10.16. Este diagrama mostra que são usadas três linhas de controle, denominadas DAV, NRFD e NDAC, para controlar cada transferência de byte de dados. Os locutores e os receptores, cada um, levantam e abaixam os sinais de controle, conforme indicado pelo fluxograma, e o locutor coloca os dados sobre o barramento de dados no instante de tempo apropriado.

O conjunto de circuitos de controle necessário para implementar esta confirmação de conexão e as outras funções requeridas deve ser fornecido em cada conjunto de circuitos de interface do módulo. É possível projetar uma linha de equipamento de saída incluindo instrumentos, perfuradoras de fita etc. e interligar cada uma dessas com o mesmo barramento usando a especificação da interface, e os fabricantes de CI muitas vezes fabricam pastilhas simples que fornecem a lógica necessária a uma interface.

Até tipos de conectores e regras de excitação e carregamento elétrico são convocados de modo que os projetistas de circuitos possam prosseguir com os detalhes necessários. Os fabricantes de minicomputadores e microcomputadores também fornecem especificações de barramento para os barramentos que eles usam de modo que os fabricantes de periféricos possam aceitar suas especificações de interface e de modo que os compradores destes computadores possam projetar seu próprio equipamento quando necessário. Os fabricantes de grandes computadores fornecem especificações similares para seus barramentos e outras interfaces.

O barramento mostrado é adequado à maioria dos propósitos dos computadores de instrumentos. Um barramento de alta velocidade pode ser projetado acrescentando-se mais linhas de controle e linhas de endereço em paralelo com as linhas de dados, e muitos computadores têm barramentos com 50 ou mais linhas. A seção seguinte mostrará um barramento de alta velocidade com linhas de dados e de endereços em paralelo e que é usado num microcomputador.

10.8 MICROPROCESSADORES E MICROCOMPUTADORES

Os sistemas de microprocessadores ou microcomputadores são baseados num conjunto de pastilhas de CI. Estas pastilhas são interligadas e se comunicam entre si usando um barramento. O conjunto de circuitos de interface do barramento geralmente está incluído nas pastilhas individuais.

A Fig. 10.17 mostra o esboço de um pequeno sistema de microcomputadores que usa quatro módulos. O barramento de endereços, o barramento de dados e o barramento de controle são as três seções do barramento. O microprocessador ou UCP controla o barramento e contém a unidade aritmético-lógica e a unidade de controle.§ Os módulos da memória podem ser de qualquer tamanho desejado e consistirão de tantas pastilhas quantas forem necessárias. As interfaces para dispositivos externos são fornecidas por pastilhas de interface chamadas *adaptadores de interface de periféricos*. Na figura, é mostrada a ligação a uma unidade de teletipo, mas as pastilhas da interface são disponíveis para serem conectadas a perfuradoras de fita, acopladores acústicos, modems, conversores A/D e D/A, impressoras etc.

O traçado básico mostrado aqui se aplica à maioria dos microprocessadores. (De fato, o mesmo traçado é usado na maioria dos minicomputadores, exceto que o barramento agora se torna um conjunto de fios numa armação física, e os módulos são geralmente

†Estes problemas se referem ao seguinte: (1) Como o ouvinte sabe quando os dados estão no barramento? (2) Como o locutor sabe quando o ouvinte recebeu os dados? O barramento descrito é um *barramento assíncrono*. Os microprocessadores e os minicomputadores muitas vezes usam barramentos síncronos onde um fio no barramento contém um relógio, e o sinal de relógio é usado para sincronizar as transferências de dados. Nestes sistemas, o locutor deve colocar dados nos fios de dados e o ouvinte deve estar pronto para receber os dados quando chegar o bordo do pulso de relógio. Um sistema síncrono é mais rápido e mais simples, porém menos flexível.

§O número da pastilha da UCP é geralmente usado como o número do microprocessador desenvolvido usando as pastilhas na série. Por exemplo, 6800 é o número da UCP, mas as RAM, ROM etc. têm outros números; no entanto, a série de pastilhas é chamada série 6800.

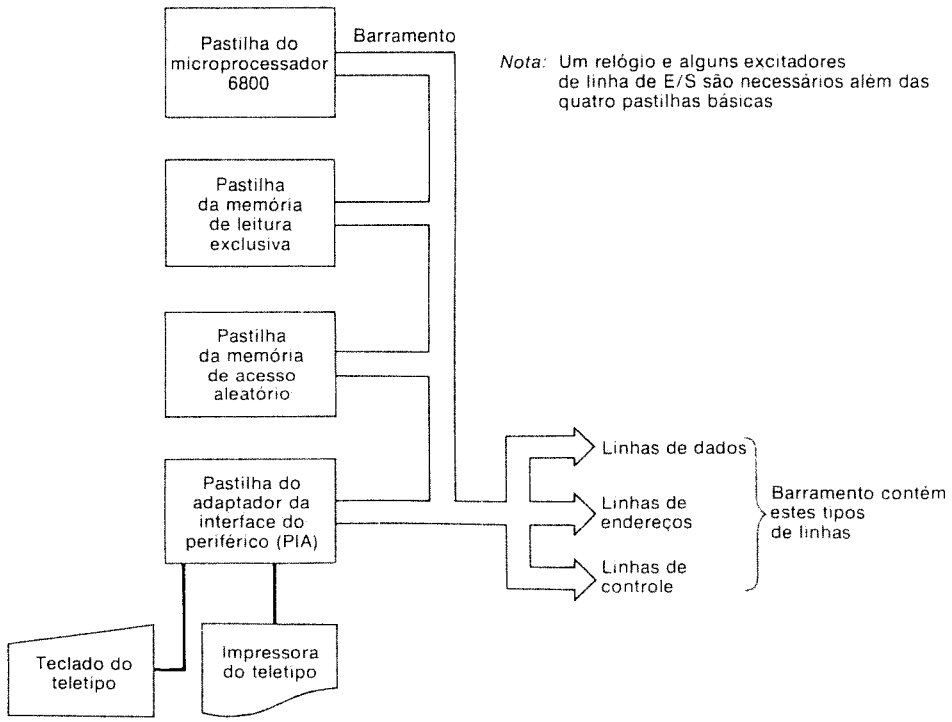


Fig. 10.17 Sistema de microcomputadores com quatro módulos.

painéis completos de circuitos que são interligados por este barramento e cada um dos quais contém várias pastilhas.†) A quantidade de memória por pastilha, pastilhas de interface disponíveis e repertório de instruções da UCP ou pastilha de microprocessador varia de fabricante a fabricante, naturalmente.

O número de linhas no barramento de dados é uma consideração importante. Nos primitivos microprocessadores (e nas calculadoras), este tende a ser de quatro linhas, de modo que um byte de 4 bits pode ser transferido ou operado. Os últimos microprocessadores tendem a ter 8 ou 16 linhas de dados de modo que palavras de tamanho moderado possam ser movidas. Observemos que as palavras de instrução também são transmitidas sobre estas linhas de dados, uma vez que elas são armazenadas na ROM e na RAM. Isto significa, em muitos casos, que são necessárias várias transferências para mover uma palavra de instrução da memória. Com efeito, a UCP monta a palavra de instrução em seus registros até que ela tenha bits suficientes para executar a instrução. Geralmente, a primeira transferência da memória conterá a seção de código OP da palavra de instrução, e isto dirá à UCP quantas transferências adicionais são necessárias para fornecer a informação do endereço.

Uma vez que a UCP controla todas as transferências no barramento, isto significa que a UCP deve dirigir as transferências de segmentos e operandos da palavra de instrução da memória, e o conjunto de circuitos lógicos para esta finalidade é fornecido na pastilha da UCP.

A seção de endereço do barramento é usada para ler endereços nas RAM, ROM e nas pastilhas da interface. A seção de endereços deve conter fios suficientes para fornecer um endereço completo numa transferência, e 16 fios são comumente usados; de outro modo,

†Aos barramentos são atribuídos vários nomes pelos fabricantes de minicomputadores, por exemplo, via única (univia), ônibus, callabus etc.

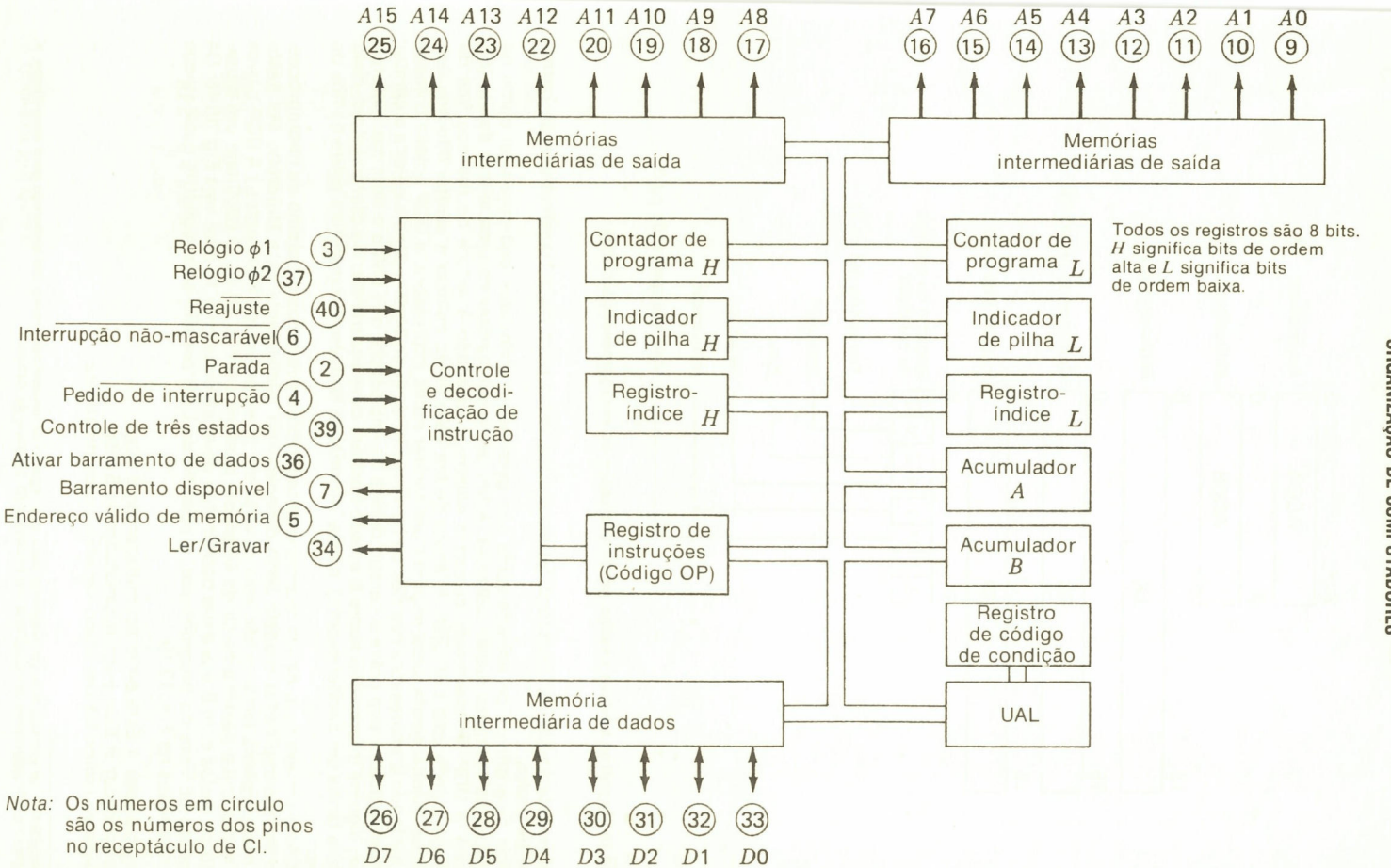


Fig. 10.18 Diagrama-bloco de pastilha de UCP para o microprocessador 6800. (Cortesia da Motorola Corp.)

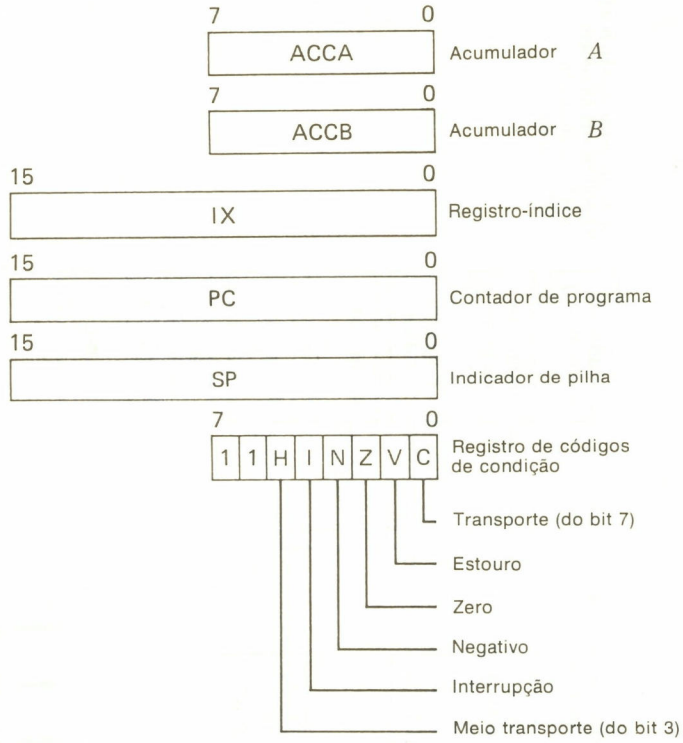


Fig. 10.19 Registros básicos usados na programação do microprocessador 6800.

várias transferências de endereço teriam de ser feitas para cada transferência de barramento de dados.

Aos dispositivos de entrada-saída são geralmente atribuídos endereços tal como os endereços da memória nestes sistemas. A fim de ler um caráter de uma leitora de fita (ou teclado), a UCP simplesmente coloca o endereço atribuído à leitora de fita nos fios de endereço e pede uma LEITURA. A leitura de fita deve então colocar o caráter lido nos fios de dados. Similarmente, para imprimir um caráter numa impressora, a UCP coloca o endereço atribuído à impressora nos fios de barramento de endereços e depois coloca o caráter nos fios de dados. Isto grava o caráter dentro da impressora que então o imprime. Naturalmente, alguma informação sobre o estado também deve ser transferida (por exemplo, caso a leitora tenha um caráter pronto), e esta informação também é transferida sobre os fios de dados.†

Como um exemplo de um microcomputador descreveremos o sistema de microprocessador-microcomputador modelo 6800 desenvolvido inicialmente pela Motorola. (As pastilhas deste sistema também são encontradas em diversos outros fabricantes.) A UCP básica está numa única pastilha de CI de 40 pinos, conforme a Fig. 10.18. O 6800 tem um barramento de dados com 8 bits e um barramento de endereços com 16 bits; veja Fig. 10.18. De um ponto de vista da programação, a pastilha da UCP contém seis registros básicos que são mostrados na Fig. 10.19.

1. *Acumulador A* Este é um acumulador de 8 bits.
2. *Acumulador B* Este é um acumulador de 8 bits.
3. *Registro-índice* Há um único registro-índice de 16 bits.

†Um dispositivo de entrada terá um endereço para seu registro de estado bem como um endereço para seus dados, e, similarmente, um dispositivo de saída terá um endereço de estado e de dados.

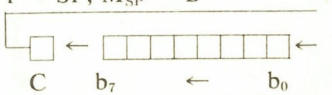
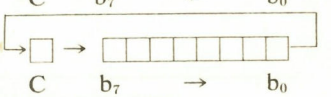
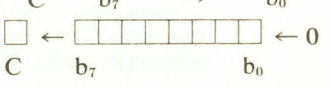
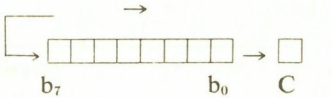
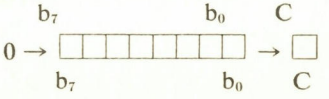
QUADRO 10.2 MODOS DE ENDEREÇAMENTO PARA MICROPROCESSADOR

Modo	Operação
Endereçamento do acumulador (ACCX)	No acumulador de endereçamento apenas, ou o acumulador *A* ou o acumulador *B* é especificado. Estas são instruções de 1 byte.
Endereçamento imediato	No endereçamento imediato, o operando está contido no segundo byte da instrução, exceto LDS e LDX que têm o operando nos segundo e terceiro bytes da instrução. O 6800 endereça este local quando ele busca a instrução imediata para execução. Estas são instruções de 2 ou de 3 bytes.
Endereçamento direto	No endereçamento direto, o endereço do operando está contido no segundo byte da instrução. O endereçamento direto permite que o usuário enderece diretamente os 256 bytes mais baixos na máquina, isto é, os locais 0 a 255. Tempos de execução incrementados são obtidos armazenando dados nestes locais. Na maioria das configurações, seria uma memória de acesso aleatório. Estas são instruções de 2 bytes.
Endereçamento estendido	No endereçamento estendido, o endereço contido no segundo byte da instrução é usado como os 8 bits mais altos do endereço do operando. O terceiro byte da instrução é usado como os 8 bits mais baixos do endereço do operando. Este é um endereço absoluto na memória. Estas são instruções de 3 bytes.
Endereçamento indexado	No endereçamento indexado, o endereço contido no segundo byte da instrução é acrescentado aos 8 bits mais baixos do registro-índice. Qualquer transporte gerado é depois adicionado aos 8 bits de ordem mais alta do registro-índice. Este resultado é depois usado para endereçar a memória. O endereço modificado é guardado num registro de endereços temporário e, assim, não há mudança para o registro-índice. Estas são instruções de 2 bytes.
Endereçamento implícito	No modo de endereçamento implícito, o código OP fornece o endereço (isto é, indicador de pilha, registro-índice etc.). Estas são instruções de 1 byte.
Endereçamento relativo	No endereçamento relativo, o endereço contido no segundo byte da instrução é acrescentado aos 8 bits mais baixos do contador de programas mais 2. O transporte ou empréstimo é então adicionado aos bits altos. Isso permite ao usuário endereçar dados dentro de uma faixa de −125 até +129 bytes da presente instrução. Estas são instruções de 2 bytes.

4. *Indicador de pilha* Este é um registro de 16 bits que indica uma pilha na memória.
5. *Contador de programa* Este é um contador de instruções ou contador de programa e contém 16 bits.

QUADRO 10.3 INSTRUÇÕES DA MEMÓRIA E DO ACUMULADOR (Cortesia da Motorola Corp.)

Operações	Mnemônico	Modos de endereçamento: Imediato OP	Direto OP	Indexado OP	Estendido OP	Implícito OP	Operação aritmética booleana (Todos os nomes de registro se referem a conteúdo)
Somar	ADDA	8B	9B	AB	BB		A + M → A
	ADDB	CB	DB	EB	FB		B + M → B
Somar acumuladores	ABA					1B	A + B → A
Somar com transporte	ADCA	89	99	A9	B9		A + M + C → A
	ADCB	C9	D9	E9	F9		B + M + C → B
E	ANDA	84	94	A4	B4		A · M → A
	ANDB	C4	D4	E4	F4		B · M → B
Bit de teste	BITA	85	95	A5	B5		A · M
	BITB	C5	D5	E5	F5		B · M
Limpar	CLR			6F	7F		00 → M
	CLRA					4F	00 → A
	CLRB					5F	00 → B
Comparar	CMPA	81	91	A1	B1		A − M
	CMPB	C1	D1	E1	F1		B − M
Comparar acumuladores	CBA					11	A − B
Complemento de 1	COM			63	73		M → M
	COMA					43	A → A
	COMB					53	B → B
Complemento de 2 (negar)	NEG			60	70		00 − M → M
	NEGA					40	00 − A → A
	NEGB					50	00 − B → B
Ajuste de decimal, A	DAA					19	Converte adição binária de caracteres BCD em formato BCD

Decremento	DEC			6A	7A		$M - 1 \rightarrow M$
	DECA					4A	$A - 1 \rightarrow A$
	DECB					5A	$B - 1 \rightarrow B$
OU exclusivo	EORA	88	98	A8	B8		$A \oplus M \rightarrow A$
	EORB	C8	D8	E8	F8		$B \oplus M \rightarrow B$
Incremento	INC			6C	7C		$M + 1 \rightarrow M$
	INCA					4C	$A + 1 \rightarrow A$
	INCB					5C	$B + 1 \rightarrow B$
Acumulador de carga	LDAA	86	96	A6	B6		$M \rightarrow A$
	LDAB	C6	D6	E6	F6		$M \rightarrow B$
OU inclusivo	ORAA	8A	9A	AA	BA		$A + M \rightarrow A$
	ORAB	CA	DA	EA	FA		$B + M \rightarrow B$
Empurrar dados	PSHA					36	$A \rightarrow M_{SP}$, $SP - 1 \rightarrow SP$
	PSHB					37	$B \rightarrow M_{SP}$, $SP - 1 \rightarrow SP$
Puxar dados	PULA					32	$SP + 1 \rightarrow SP$, $M_{SP} \rightarrow A$
	PULB					33	$SP + 1 \rightarrow SP$, $M_{SP} \rightarrow B$
Girar à esquerda	ROL			69	79	M	
	ROLA					49 A	
	ROLB					59 B	C b_7 ← b_0
Girar à direita	ROR			66	76	M	
	RORA					46 A	
	RORB					56 B	C b_7 → b_0
Desvio à esquerda, aritmético	ASL			68	78	M	
	ASLA					48 A	C b_7 b_0
	ASLB					58 B	
Desvio à direita, aritmético	ASR			67	77	M	
	ASRA					47 A	
	ASRB					57 B	b_7 b_0 C
Desvio à direita, lógico	LSR			64	74	M	
	LSRA					44 A	
	LSRB					54 B	b_7 b_0 C

QUADRO 10.3 (CONT.) INSTRUÇÕES DA MEMÓRIA E DO ACUMULADOR (Cortesia da Motorola Corp.)

		Modos de endereçamento					
		Imediato	Direto	Indexado	Estendido	Implícito	Operação aritmética booleana (Todos os nomes de registro se referem a conteúdo)
Operações	Mnemônico	OP	OP	OP	OP	OP	
Acumulador de armazenamento	STAA		97	A7	B7		A → M
	STAB		D7	E7	F7		B → M
Subtrair	SUBA	80	90	A0	B0		A − M → A
	SUBB	C0	D0	E0	F0		B − M → B
Subtrair acumulador	SBA					10	A − B → A
Subtrair com transporte	SBCA	82	92	A2	B2		A − M − C → A
	SBCB	C2	D2	E2	F2		B − M − C → B
Transferir acumuladores	TAB					16	A → B
	TBA					17	B → A
Testar, zero ou menos	TST			6D	7D		M − 00
	TSTA					4D	A − 00
	TSTB					5D	B − 00

OP Código de operação (hexadecimal)
+ Mais aritmético
− Menos aritmético
· E booleano
M_{SP} Conteúdo de local da memória indicado pelo indicador de pilha
+ OU inclusivo booleano
⊕ OU exclusivo booleano
$\bar{M}$ Complemento de M
→ Transferência para dentro de
0 Bit = zero
00 Byte = zero

Nota: As instruções do modo de endereçamento do acumulador estão incluídas na coluna de endereçamento *implícito*.

QUADRO 10.4 REGISTRO-ÍNDICE E INSTRUÇÕES DE MANIPULAÇÃO DA PILHA

Operações do indicador	Mnemônico	Imediato OP	Direto OP	Indexado OP	Estendido OP	Implícito OP	Operação aritmética booleana
Comparar registro-índice	CPX	8C	9C	AC	BC		$X_H - M, X_L - (M + 1)$
Decrementar registro-índice	DEX					09	$X - 1 \rightarrow X$
Decrementar indicador de pilha	DES					34	$SP - 1 \rightarrow SP$
Incrementar registro-índice	INX					08	$X + 1 \rightarrow X$
Incrementar indicador de pilha	INS					31	$SP + 1 \rightarrow SP$
Carregar registro-índice	LDX	CE	DE	EE	FE		$M \rightarrow X_H, (M + 1) \rightarrow X_L$
Carregar indicador de pilha	LDS	8E	9E	AE	BE		$M \rightarrow SP_H, (M + 1) \rightarrow SP_L$
Armazenar registro-índice	STX		DF	EF	FF		$X_H \rightarrow M, X_L \rightarrow (M + 1)$
Armazenar indicador de pilha	STS		9F	AF	BF		$SP_H \rightarrow M, SP_L \rightarrow (M + 1)$
Registro-índice → indicador de pilha	TXS					35	$X - 1 \rightarrow SP$
Indicador de pilha → registro-índice	TSX					30	$SP + 1 \rightarrow X$

QUADRO 10.5 INSTRUÇÕES DE SALTO E DESVIO

		Relativo	Indexado	Estendido	Implícito	
Operações	Mnemônico	OP	OP	OP	OP	Teste de desvio
Desviar sempre	BRA	20				Nenhum
Desviar se transporte limpo	BCC	24				$C = 0$
Desviar se transporte ajustado	BCS	25				$C = 1$
Desviar se = 0	BEQ	27				$Z = 1$
Desviar se ≥ 0	BGE	2C				$N \oplus V = 0$
Desviar se > 0	BGT	2E				$Z + (N \oplus V) = 0$
Desviar se mais alto	BHI	22				$C + Z = 0$
Desviar se ≤ 0	BLE	2F				$Z + (N \oplus V) = 1$
Desviar se mais baixo ou o mesmo	BLS	23				$C + Z = 1$
Desviar se < 0	BLT	2D				$N \oplus V = 1$
Desviar se menos	BMI	2B				$N = 1$
Desviar se não igual a 0	BNE	26				$Z = 0$
Desviar se estouro limpo	BVC	28				$V = 0$
Desviar se estouro ajustado	BVS	29				$V = 1$
Desviar se mais	BPL	2A				$N = 0$
Desviar para sub-rotina	BSR	8D				
Salto	JMP		6E	7E		
Saltar para sub-rotina	JSR		AD	BD		
Sem operação	NOP				02	Somente controla avanços de programas
Retorno de interrupção	RTI				3B	
Retorno de sub-rotina	RTS				39	
Interrupção de *software*	SWI				3F	
Espera de interrupção	WAI				3E	

QUADRO 10.6 BITS DE REGISTRO DO CÓDIGO DE CONDIÇÃO

Registro de código de condição O registro de código de condição indica os resultados de uma operação da unidade aritmética e lógica: negativo (N), zero (Z), estouro (V), transporte do bit 7 (C) e meio transporte do bit 3 (H). Estes bits do registro de código de condição são usados como condições testáveis para as instruções de desvio condicional. O bit 4 é o bit de máscara de interrupção (I). Os bits abandonados do registro de código de condição (b_6 e b_7) são 1.

6. *Registro de estado* Este é um registro de 6 bits que contém seis biestáveis *H, I, N, Z, V* e *C*. Os resultados das operações aritméticas e de outras operações estão armazenados nestes bits, conforme será descrito.

O repertório de instruções para esta pastilha da UCP inclui cerca de 100 instruções diferentes. O código de operação é de 8 bits, o tamanho de uma palavra na memória. Há sete diferentes modos de endereçamento, que são descritos no Quadro 10.2. Uma lista parcial das instruções para esta pastilha de microprocessador é mostrada nos Quadros 10.3 a 10.5.

A maneira pela qual operam as instruções de desvio condicional dispensa comentários. Quando uma operação aritmética ou booleana é efetuada, os bits de estado são ajustados de acordo com o resultado desta operação. Os Quadros 10.6 e 10.7 mostram os bits de estado e detalham suas funções. As instruções de desvio usam os valores desses bits para determinar se um desvio deve ser feito ou não.

Por exemplo, se for efetuada uma adição aritmética, seja o acumulador *A* adicionado ao acumulador *B*, então, se a soma for negativa, o bit *N* será ajustado em um 1. Também admitimos que não há estouro, e assim *V* será ajustado em um 0. Agora, se segue uma instrução "desviar se *LT* 0" (BLT), o computador desviará para o endereço fornecido na parte de endereço da instrução. Se o resultado da adição tiver sido 0 ou positivo, nenhum desvio deve ter ocorrido, e a instrução seguinte na seqüência seria tomada.

O uso dos bits de estado que registram o resultado de operações aritméticas e lógicas, e de instruções de desvio que usam estes valores nos bits para determinar se deve haver desvio ou não, é feito com muita freqüência nos computadores modernos.

O *bit de máscara de interrupção* (*I*) no registro de estado é ajustado ligado quando se permite que dispositivos externos de entrada-saída interrompam o computador. Quando ocorre uma interrupção, o computador salta para uma rotina (programa) de serviço de interrupções que está armazenada na memória. Para simplificar e encurtar o programa de serviço de interrupções, este microprocessador transfere automaticamente os valores em todos os registros da UCP para dentro de uma pilha na memória e coloca o endereço deste conteúdo do registro armazenado no indicador de pilha. O programa de serviço de interrupções pode então reparar a impressora, a leitora ou qualquer dispositivo que tenha gerado a interrupção e pode, mais tarde, substituir os conteúdos dos registros da UCP por seus estados quando interrompidos e reiniciar o programa onde ele foi interrompido.

A manutenção destes registros armazenados numa pilha também permite que o programa de serviço de interrupções seja interrompido já que os conteúdos dos registros são novamente colocados na pilha. Deste modo, várias interrupções podem seguir umas às outras, e o programa pode ainda reparar cada interrupção alternadamente e depois retornar ao programa original que estava operando quando ocorreu a primeira interrupção.

Os detalhes das características de operação desta UCP podem ser achados nos manuais de fabricantes listados na bibliografia.

Quando um conjunto de pastilhas dessa espécie é usado com um programa fixo, por exemplo, num controlador industrial, o programa é geralmente desenvolvido usando-se *software* de sistemas fornecido pelo fabricante das pastilhas e pelos vendedores de *software*. É prática comum escrever o programa para o microprocessador e montar este programa usando um outro computador maior. O programa também é geralmente testado usando este computador maior que passa o programa num simulador. O computador maior então produz uma fita perfurada que é usada para montar a ROM em que o programa será armazenado.

Considerável esforço é desenvolvido pelos fabricantes das pastilhas de microprocessador para facilitar a programação do microprocessador e facilitar o preparo das ROM e o

carregamento das RAM, quando requeridos. Às vezes, linguagens de nível mais alto fornecidas possibilitaram que os programas a serem escritos em Fortran, PL/1, ou em outras linguagens de compilador sejam depois traduzidos no programa para o microcomputador.

10.9 ORGANIZAÇÃO DE COMPUTADOR DE ENDEREÇO ÚNICO

Um exemplo direto de um computador de endereço único é o minicomputador PDP-8 da DEC. Este foi o primeiro lançamento importante na corrida dos minicomputadores e continua a ser um computador importante (em termos de vendas) para a DEC. Este computador será usado para assinalar alguns princípios básicos da arquitetura de computadores.

O PDP-8 tem uma palavra de instrução de 12 bits, conforme indicado na Fig. 10.20. A memória é também organizada em palavras de 12 bits, e um bloco básico da memória consiste de $2^{12} = 4096$ palavras. Uma vez que a palavra da instrução é muito curta, os projetistas têm permitido apenas 3 bits na parte do código OP, e assim, conforme indicado na Fig. 10.21, há somente oito classes básicas de palavras de instrução.

Diversas das instruções básicas são muito diretas. Por exemplo, a instrução TAD (para SOMAR em complemento de 2) simplesmente efetua uma adição, em complemento de 2, do operando endereçado na memória com o operando correntemente no único acumulador de 12 bits e coloca a soma neste acumulador. Há um biestável extra chamado *bit L* ou *bit de encadeamento* que recebe qualquer estouro desta adição. O código OP para TAD é 000_2 ou 0_8.

De modo similar, a instrução E simplesmente executa uma operação E bit-por-bit sobre o operando endereçado na memória e o conteúdo do acumulador, colocando o resultado no acumulador. O código OP para E é 001_2 ou 1_8.

As técnicas de endereçamento usadas pelos 9 bits para formar endereços foram descritas. O bit 3 numa palavra de instrução será um 0 se o endereço for direto e um 1 se indireto (o endereço do endereço). O bit 4 será um 0 se os 7 bits restantes (bits 5 a 11) fornecerem o endereço real na página 0 da memória, e um 1 se o endereço estiver na mesma página que a palavra de instrução.

Há uma instrução de ISZ (incrementar e saltar se zero) que incrementa a palavra endereçada e a retorna para a memória. Se esta palavra se torna um 0, a próxima instrução seguindo a instrução ISZ não é executada; em vez disso, o computador executa a seguir a palavra na memória que se segue a esta palavra. Esta instrução particular é muito útil na indexação através de tabelas e é fornecida como um substituto aos registros-índice.

A instrução de DCA (depositar e limpar o acumulador) armazena o acumulador na memória no endereço especificado e também limpa o acumulador em todos os 0. O código OP do DCA é 011 ou 3_8.

A JMS (saltar para sub-rotina) cria uma importante característica nos repertórios de instrução dos computadores. Constitui boa norma na escrita de um programa de computador dividir o programa em tantos subprogramas (partes separáveis do programa) quanto possível. Estes subprogramas ou sub-rotinas são então saltados sempre que a função que eles realizam é solicitada. Ver Fig. 10.22 *(a)*.

O problema com que se defronta o projetista de computadores é o de que um determinado subprograma pode ser saltado de diversos locais diferentes no programa. Isto é mostrado na Fig. 10.22(*b*). Por exemplo, na maioria dos minicomputadores e dos microcompu-

QUADRO 10.7 INSTRUÇÕES DE MANIPULAÇÃO DO REGISTRO DE CÓDIGO DE CONDIÇÃO

Operações	Mnemônico	OP	Operação booleana
Limpar transporte	CLC	0C	$0 \rightarrow C$
Limpar máscara de interrupção	CLI	0E	$0 \rightarrow 1$
Limpar estouro	CLV	0A	$0 \rightarrow V$
Ajustar transporte	SEC	0D	$1 \rightarrow C$
Ajustar máscara de interrupção	SEI	0F	$1 \rightarrow I$
Ajustar estouro	SEV	0B	$1 \rightarrow V$
Acumulador $A \rightarrow$ CCR	TAP	06	$A \rightarrow CCR$
$CCR \rightarrow$ acumulador A	TPA	07	$CCR \rightarrow A$

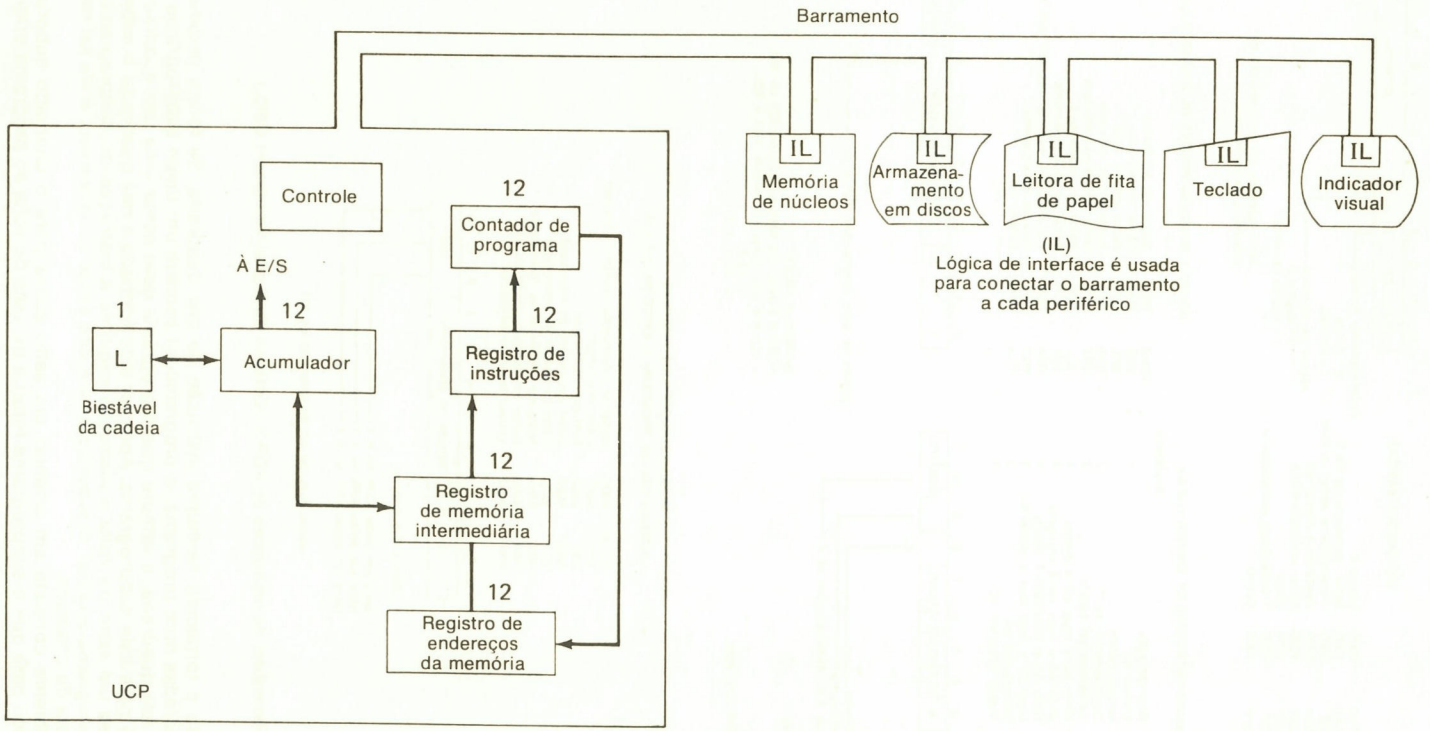


Fig. 10.20 Organização do PDP-8.

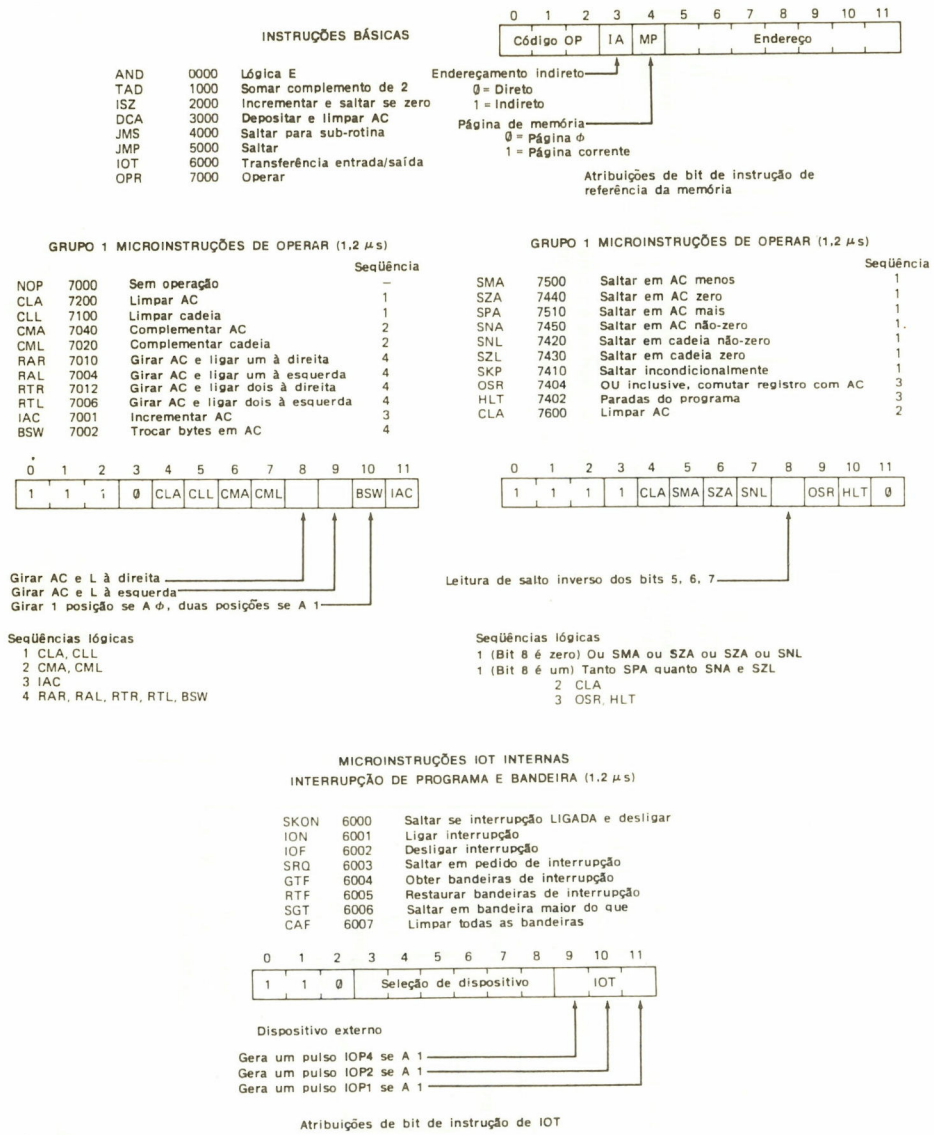


Fig. 10.21 Repertório de instruções do PDP-8. (Cortesia da Digital Equipment Corp.)

tadores não é fornecida nenhuma instrução de raiz quadrada. Se forem pedidas muitas raízes quadradas num programa, o programador escreve um único subprograma (ou sub-rotina) de raiz quadrada, e sempre que o programa deve achar uma raiz quadrada, é feito um SALTO para este subprograma. Após ter sido formada a raiz quadrada, o subprograma então causa um salto de volta à instrução seguinte à instrução do subprograma JMS na seção do programa original. O subprograma é dito então ser *chamado*, e ele sai retornando ao *programa de chamada*.

O problema consiste em arranjar um salto suave para o chamado subprograma e torná-lo fácil para que o subprograma retorne ou salte de volta ao programa original. Para

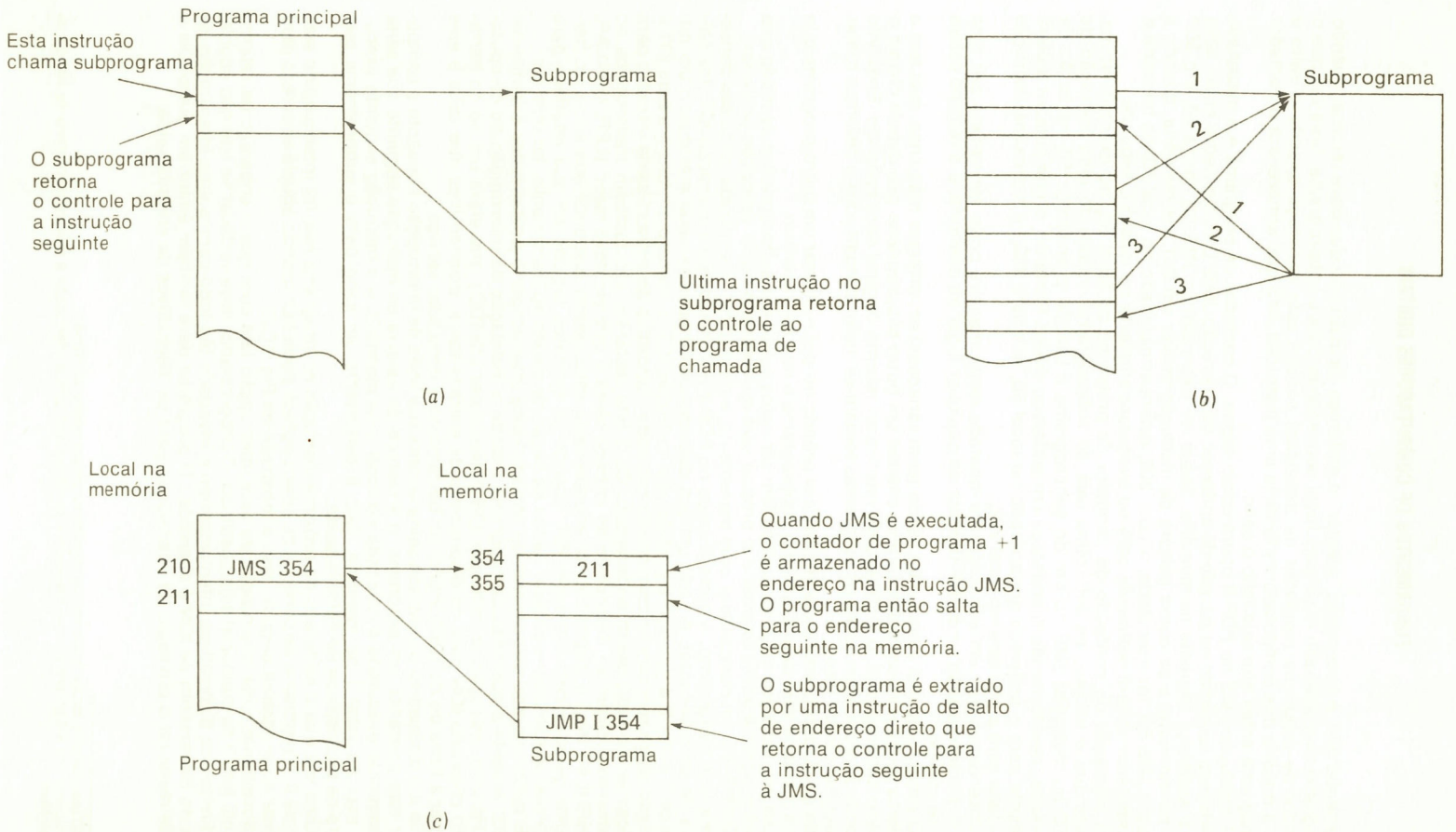


Fig. 10.22 *(a)* Chamada de um subprograma; *(b)* o mesmo subprograma pode ser chamado diversas vezes; (c) como o PDP-8 manipula chamadas e retornos do subprograma.

implementar isto, é necessário "plantar" o endereço da instrução que deve retornar quando o programa tiver acabado em algum local conveniente para o subprograma. Uma vez que o contador de programa (contador de instrução) contém este endereço quando o salto é feito, a maioria dos computadores fornece uma instrução JMS que armazenará o contador de programa antes que seja feito o salto.

A instrução JMS no PDP-8 opera como segue. O contador de programa é armazenado no endereço fornecido na porção de endereço da instrução JMS. O computador então salta para o endereço seguinte na memória. Assim, se escrevermos JMS 50_8 (onde 50_8 é o 50.º local na memória) e se nossa palavra de instrução estiver em 201_8 na memória, quando a instrução JMS 50_8 for executada, o valor 202_8 estará armazenado no local 50_8 na memória e o computador saltará realmente para ou executará a palavra no local 51_8 seguinte.†

A plantação do conteúdo do contador de programa no local 50_8 no exemplo acima, permite que o escritor da sub-rotina saia do subprograma colocando uma instrução de SALTO ou instrução JMP no fim do subprograma e usando o endereço 50_8 na seção de endereços, mas tornando o endereço um endereço direto. Portanto, o computador saltará realmente para o endereço armazenado no local 50_8 que será 202_8, e a próxima instrução executada será a que está no local 202_8.

Observemos que um subprograma montado desta maneira pode ser saltado por uma JMS e retirado usando uma JMP indireta de qualquer lugar na memória, e o retorno estará sempre correto.

O esquema acima é bom e foi usado (com variações) em diversas máquinas, mas tem o problema de que, se um subprograma chamar um outro subprograma que depois chama o primeiro subprograma, o endereço de retorno ao retorno original será apagado. Enquanto isto pode parecer improvável, os programas tornam-se muito complicados e devemos tomar precauções contra isso.

Se um subprograma pode chamar a si próprio ou pode chamar um outro programa que depois o chama sem danos, então o subprograma é dito ser *recursivo*.

Um esquema por meio do qual podem ser feitos saltos aos subprogramas de modo que um subprograma possa chamar a si próprio será apresentado numa descrição posterior.

O PDP-8 é um computador com barramento, e os dispositivos entrada-saída são conectados a este barramento. A entrada-saída no PDP-8 é fornecida pela instrução IOT (transferência de entrada-saída). A cada um dos dispositivos de entrada-saída é atribuído um número de 0 a $2^6 - 1$, e, quando é emitida uma instrução IOT, o número em bits 3 a 8 da IOT é colocado sobre seis fios no barramento que cada dispositivo de entrada-saída introduz para verificar se ele está sendo endereçado. Os 3 bits restantes são também transmitidos no barramento e dizem ao dispositivo de entrada-saída o que ele deve fazer. (LER, GRAVAR, REENROLAR etc.) o dispositivo de entrada-saída selecionado responde aos sinais do barramento gerados pela instrução IOT, usando os circuitos lógicos em sua interface para interpretar a instrução e colocar dados na seção de dados do barramento, ler dele etc.

É permitido aos dispositivos de entrada-saída interromper o processador e demandar serviço durante a operação do programa usando a instalação de interrupção do computador. Há uma instrução, a instrução ION (interromper LIGADO), que eleva um fio do barramento chamado ION para os dispositivos de entrada-saída, dizendo-lhes que cabe a eles levantar a linha de INTERRUPÇÃO no barramento e demandar serviço.

Se um dispositivo de entrada-saída levanta sua linha de interrupção enquanto o computador está operando um programa, o endereço da palavra de instrução seguinte que seria normalmente executada é colocada no local 0 na memória, e a instrução seguinte executada está no local 1 na memória. (Isto é geralmente um salto para o subprograma que repara os dispositivos de entrada-saída).

Uma vez que o endereço da próxima instrução no programa que foi interrompido está no local 0, o programa de serviços de interrupções§ pode sair usando aquele endereço para retornar ao programa original. Isso é mostrado na Fig. 10.23.

Novamente, uma interrupção de uma interrupção fará com que o endereço de retorno original (em 0 na memória) seja destruído. O programador deve verificar se isto não ocorre, e o direito dos dispositivos interromperem é revogado tão logo ocorra uma interrupção. (A linha do barramento de ION é abaixada.) O programa deve também emitir um outro ION a fim de restaurar o privilégio de interrupção para os dispositivos de entrada-saída.

---

†A DEC usa números octais quase exclusivamente na programação do PDP-8.

§O programa (ou subprograma) que manipula o periférico que gera a interrupção é chamado *programa de serviço de interrupções*.

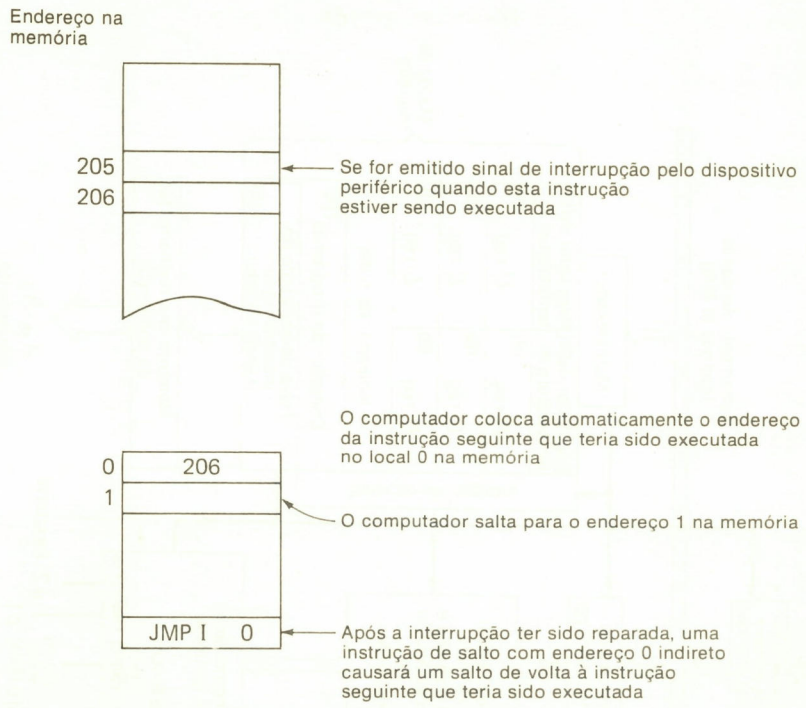


Fig. 10.23 Interrupções no PDP-8.

O PDP-8 tem várias características que ajudaram a torná-lo atraente, mas que estão fora do contexto desta descrição. Por exemplo, desde que haja apenas uma única instrução aritmética, a TAD, é necessário complementar um operando e depois adicionar a fim de subtrair. Para efetuar este complemento e também fornecer instruções de GIRAR ou DESLOCAR, algumas instruções de SALTAR e algumas outras operações lógicas, o código OP 111 é uma classe de instrução "sem endereço" onde os 9 bits restantes dizem qual dentre um número de diferentes possíveis operações podem ser feitas ocorrer. (Os detalhes podem ser achados nos manuais dos fabricantes listados nas referências.)

10.10 UM MICROPROCESSADOR DE ENDEREÇO ÚNICO

O Intel 8080 é um exemplo de uma série de pastilhas de microprocessadores com uma pastilha básica de CI de 40 pinos contendo a UCP 8080. Outras pastilhas disponíveis incluem RAM, ROM, pastilhas de interface etc. As pastilhas são interligadas usando um barramento que contém 8 linhas de dados, 16 linhas de endereço e 6 linhas de controle.

A UCP 8080 é basicamente uma organização de um só acumulador, mas são fornecidos vários outros registros, conforme indicado na Fig. 10.24 (ver adiante). Os formatos da palavra de instrução são mostrados na Fig. 10.25. Já que são usados 8 bits para o código OP, foi fornecido um grande repertório de instruções. Algumas dessas instruções são mostradas nos Quadros 10.8 e 10.9.

No começo de cada ciclo da instrução, o código OP de 8 bits é lido pela UCP 8080, e isso determina quantas buscas mais da memória a UCP deve fazer para executar a instrução. Algumas instruções requerem apenas o código OP de 8 bits, enquanto algumas requerem 8 bits e outras, operandos ou endereços de 16 bits, e, portanto, a UCP deve fazer os acessos necessários para executar a instrução.

O Intel 8080 possui um bom conjunto de modos de endereçamento (ver Quadro 10.10). Ele também tem instruções de SALTO ou DESVIO condicional que saltam ou não saltam, dependendo dos valores que são chamados *bandeiras de condição*. Estes consistem de cinco biestáveis (ver Quadro 10.11) que são ajustados no estado 0 ou 1 pelos resultados de

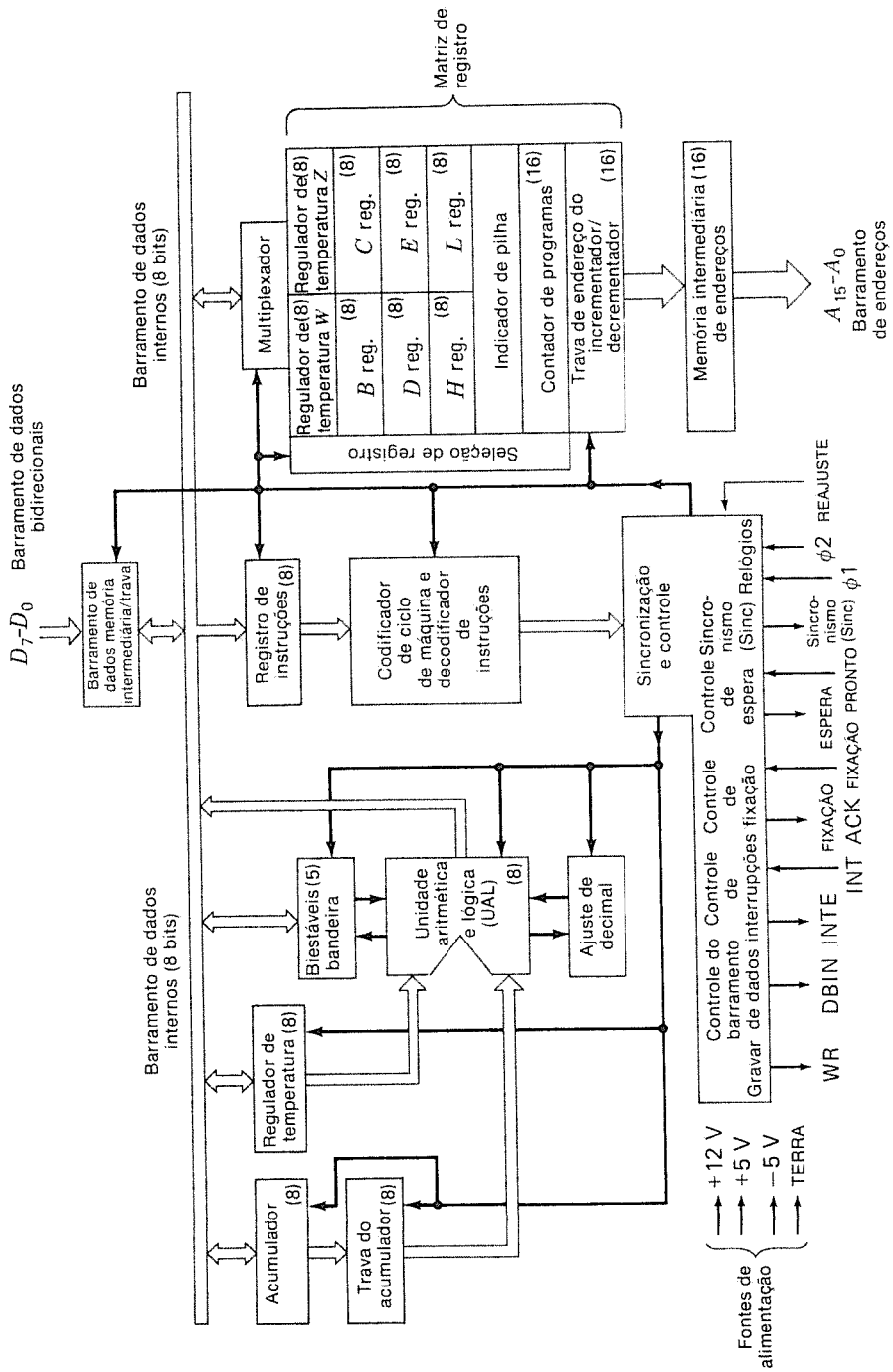


Fig. 10.24 Diagrama-bloco funcional da UCP do 8080. (Cortesia da Intel Corp.)

QUADRO 10.8 REPERTÓRIO DE INSTRUÇÕES DA UCP 8080 (NOTAÇÃO USADA NO QUADRO 10.9)

Símbolo	Significado
Acumulador	Registro A
Endereço	Quantidade de endereço de 16 bits
Dados	Quantidade de dados de 8 bits
Dados 16	Quantidade de dados de 16 bits
Byte 2	O segundo byte da instrução
Byte 3	O terceiro byte da instrução
Port	Endereço de 8 bits de um dispositivo de entrada-saída
r,r1,r2	Um dos registros A,B,C,D,E,H,L
DDD, SSS	O padrão de bit designando um dos registros A,B,C,D,E,H,L (DDD = destino, SSS = fonte):

DDD ou SSS	Nome de registro
111	A
000	B
001	C
010	D
011	E
100	H
101	L

Símbolo	Significado
rh	O primeiro (ordem alta) registro de um par de registros designado.
rl	O segundo (ordem baixa) registro de um par de registros designado.
rp	Um dos pares de registros: B representa o par B,C, com B como o registro de ordem elevada e C como o registro de ordem baixa; D representa o par D,E, com D como o registro de ordem alta e E como o registro de ordem baixa; H representa o par H,L, tendo H como registro de ordem alta e L como o registro de ordem baixa; SP representa o registro indicador de pilha de 16 bits.
RP	O padrão de bits que designa um dos pares de registros B,D,H,SP:

RP	Par de registros
00	B,C
01	D,E
10	H,L
11	SP

QUADRO 10.8 (CONT.) REPERTÓRIO DE INSTRUÇÕES DA UCP 8080 (NOTAÇÃO USADA NO QUADRO 10.9)

Símbolo	Significado	Símbolo	Significado
PC	Registro de 16 bits do contador de programa (PCH e PCL são usados para consulta aos 8 bits de ordem alta e de ordem baixa, respectivamente).	$\leftarrow$	"É transferido a"
		$\wedge$	Lógica E
		$\veebar$	OU exclusivo
		$\vee$	OU inclusivo
SP	Indicador de pilha	+	Adição
r_m	Bit m do registro r (os bits são número 7 a 0 da esquerda para a direita).	−	Subtração em complemento de 2
		*	Multiplicação
Z,S,P,CY,AC	As bandeiras de condição: zero, sinal, paridade, transporte e transporte auxiliar, respectivamente.	$\leftrightarrow$	"É trocado com"
		$\bar{}$	O complemento de 1 [por exemplo, $(\bar{A})$]
		n	O reinício número 0 a 7
( )	O conteúdo do local da memória ou dos registros encerrados nos parênteses.	NNN	A representação binária 000 a 111 para reiniciar número 0 a 7, respectivamente.

QUADRO 10.9 REPERTÓRIO DE INSTRUÇÕES DA UCP 8080

MOV r1, r2 (movimentar registro)
(r1) ← (r2)
O conteúdo do registro r2 é transferido para o registro r1.

0	1	D	D	D	S	S	S

Endereçamento: registro Bandeiras: nenhuma

MOV r, M (transferir da memória)
(r) ← ((H) (L))
O conteúdo do local da memória, cujo endereço está nos registros H e L, é transferido para o registro r.

0	1	D	D	D	1	1	0

Endereçamento: registro indireto Bandeiras: nenhuma

MOV M, r (transferir para a memória)
((H) (L)) ← (r)
Endereçamento: registro indireto Bandeiras: nenhuma

MOV M, r (transferir para a memória)
((H) (L)) ← (r)
O conteúdo do registro r é transferido para o local da memória, cujo endereço está nos registros H e L.

0	1	1	1	0	S	S	S

Endereçamento: registro indireto Bandeiras: nenhuma

ADD r (somar registro)
(A) ← (A) + (r)
O conteúdo do registro r é adicionado ao conteúdo do acumulador. O resultado é colocado no acumulador.

1	0	0	0	0	S	S	S

Endereçamento: registro Bandeiras: Z,S,P,CY,AC

ADD M (somar memória)
(A) ← (A) + ((H) (L))
O conteúdo do local da memória, cujo endereço está contido nos registros H e L, é adicionado ao conteúdo do acumulador. O resultado é colocado no acumulador.

1	1	0	0	0	1	1	0

Endereçamento: registro indireto Bandeiras: Z,S,P,CY,AC

ADI dados (somar imediato)
(A) ← (A) + (byte 2)
O conteúdo do segundo byte da instrução é adicionado ao conteúdo do acumulador. O resultado é colocado no acumulador.

1	1	0	0	0	1	1	0
Dados							

Endereçamento: imediato

QUADRO 10.9 (CONT.) REPERTÓRIO DE INSTRUÇÕES DA UCP 8080

INR M (incrementar memória)
((H) (L)) ← ((H) (L)) + 1
O conteúdo do local da memória, cujo endereço está contido nos registros H e L, é incrementado de um. *Nota:* Todas as bandeiras de condição, exceto CY, são afetadas.

0	0	1	1	0	1	0	0

Endereçamento: registro indireto Bandeiras: Z,S,P,AC

LDA addr (carregar acumulador direto)
(A) ← ((byte 3) (byte 2))
O conteúdo do local da memória, cujo endereço é especificado nos bytes 2 e 3 da instrução, é transferido para o registro A.

0	0	1	1	1	0	1	0
Endereço de ordem baixa							
Endereço de ordem alta							

Endereçamento: direto Bandeiras: nenhuma

STA addr (armazenar acumulador direto)
((byte 3) (byte 2)) ← (A)
O conteúdo do acumulador é transferido para o local da memória cujo endereço é especificado nos bytes 2 e 3 da instrução.

0	0	1	1	0	0	1	0
Endereço de ordem baixa							
Endereço de ordem alta							

SUB M (subtrair memória)
(A) ← (A) − ((H) (L))
O conteúdo do local da memória, cujo endereço está contido nos registros H e L, é subtraído do conteúdo do acumulador. O resultado é colocado no acumulador.

1	0	0	1	0	1	1	0

Endereçamento: registro indireto Bandeiras: Z,S,P,CY,AC

SUI dados (subtrair imediato)
(A) ← (A) − (byte 2)
O conteúdo do segundo byte da instrução é subtraído do conteúdo do acumulador. O resultado é colocado no acumulador.

1	1	0	1	0	1	1	0
Dados							

Endereçamento: imediato Bandeiras: Z,S,P,CY,AC

SBB r (subtrair registro com empréstimo)
(A) ← (A) − (r) − (CY)
O conteúdo do registro r e o conteúdo da bandeira CY são ambos subtraídos do acumulador. O resultado é colocado no acumulador.

1	0	0	1	1	S	S	S

DCR r (decrementar registro)
(r) ← (r) − 1
O conteúdo do registro r é decrementado de 1. *Nota:* Todas as bandeiras de condição, exceto CY, são afetadas.

0	0	D	D	D	1	0	1

Endereçamento: registro

LDAX rp (carregar acumulador indireto)
(A) ← ((rp))
O conteúdo do local da memória, cujo endereço está no par rp de registros, é transferido para o registro A. *Nota:* Somente os pares de registros rp = B (registros B e C) ou rp = D (registros D e E) podem ser especificados.

0	0	R	P	1	0	1	0

Endereçamento: registro indireto Bandeiras: nenhuma

STAX rp (armazenar acumulador indireto)
((rp)) ← (A)
O conteúdo do registro A é transferido para o local da memória cujo endereço está no par rp de registros. *Nota:* Somente os pares de registros rp = B (registros B e C) ou rp = D (registros D e E) podem ser especificados.

0	0	R	P	0	0	1	0

Endereçamento: registro indireto Bandeiras: nenhuma

XCHG (trocar H e L por D e E)
(H) ↔ (D)
(L) ↔ (E)
Os conteúdos dos registros H e L são trocados pelos conteúdos dos registros D e E.

1	1	1	0	1	0	1	1

Endereçamento: registro Bandeiras: nenhuma

INR r (incrementar registro)
(r) ← (r) + 1
O conteúdo do registro r é incrementado de 1. *Nota:* Todas as bandeiras de condição, exceto CY, são afetadas.

0	0	D	D	D	1	0	0

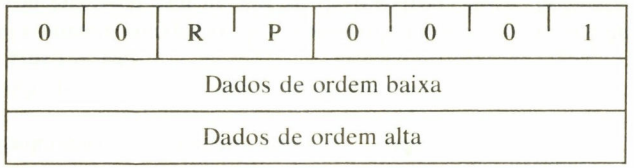
Endereçamento: registro Bandeiras: Z,S,P,AC

MVI M, dados (transferir para a memória imediata)
((H) (L)) ← (byte 2)
O conteúdo de byte 2 da instrução é transferido para o local da memória cujo endereço está nos registros H e L.

0	0	1	1	0	1	1	0
Dados							

Endereçamento: imediato registro indireto
Bandeiras: nenhuma

LXI rp, dados 16 (carregar par de registros imediato)
(rh) ← (byte 3) (rl) ← (byte 2)
Byte 3 da instrução é transferido para dentro do registro de ordem alta (rh) do par rp de registros. Byte 2 da instrução é transferido para dentro do registro de ordem baixa (rl) do par rp de registros.



Endereçamento: imediato Bandeiras: nenhuma

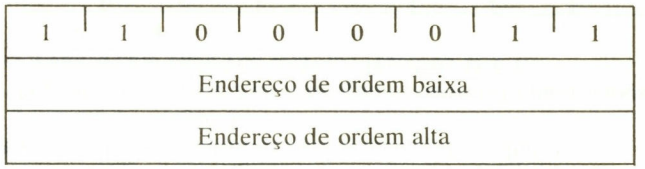
GRUPO DE DESVIO

Este grupo de instruções altera o fluxo normal do programa seqüencial. *As bandeiras de condição não são afetadas* por qualquer instrução neste grupo.

Os dois tipos de instruções de desvio são incondicionais e condicionais. As transferências incondicionais simplesmente executam a operação especificada no registro PC (o contador de programa). As transferências condicionais examinam o estado de uma das quatro bandeiras do processador para determinar se o desvio especificado deve ser executado. As condições que podem ser especificadas são como segue:

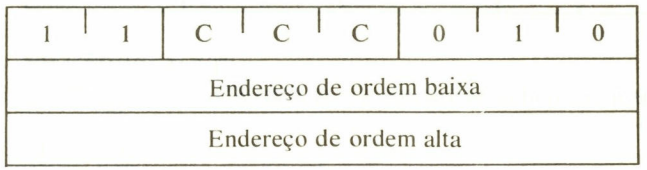
Condição	CCC
NZ — não zero (Z = 0)	000
Z — zero (Z = 1)	001
NC — sem transporte (CY = 0)	010
C — transporte (CY = 1)	011
PO — paridade ímpar (P = 0)	100
PE — paridade par (P = 1)	101
P — mais (S = 0)	110
M — menos (S = 1)	111

JMP addr (saltar)
(PC) ← (byte 3) (byte 2)
O controle é transferido para a instrução cujo endereço é especificado no byte 3 e no byte 2 da instrução corrente.



Endereçamento: imediato
Bandeiras: nenhuma

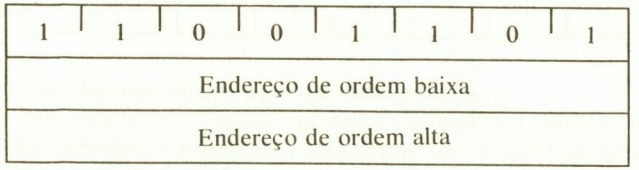
Jcondition addr (salto condicional)
Se (CCC), (PC) ← (byte 3) (byte 2)
Se a condição especificada for verdadeira, o controle é transferido para a instrução cujo endereço é especificado no byte 3 e no byte 2 da instrução corrente: de outro modo, o controle continua seqüencialmente.



Endereçamento: imediato
Bandeiras: nenhuma

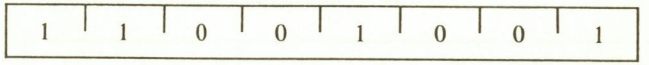
QUADRO 10.9 (CONT.) REPERTÓRIO DE INSTRUÇÕES DA UCP 8080

CHAMAR endereço (call)
((SP) − 1) ← (PCH) (SP) ← (SP) − 2
((SP) − 2) ← (PCL) (PC) ← (byte 3) (byte 2)
Os 8 bits de ordem alta do endereço da instrução seguinte são transferidos para o local da memória cujo endereço é menor do que o conteúdo do registro SP. Os 8 bits de ordem baixa do endereço da instrução seguinte são transferidos para o local da memória cujo endereço é 2 menos do que o conteúdo do registro SP. O conteúdo do registro SP é decrementado de 2. O controle é transferido para a instrução cujo endereço é especificado no byte 3 e no byte 2 da instrução corrente.



Endereçamento: imediato/registro indireto
Bandeiras: nenhuma

RET (retorno)
(PCL) ← ((SP)) (PCH) ← ((SP) + 1)
(SP) ← (SP) + 2
O conteúdo do local da memória cujo endereço é especificado no registro SP é transferido para os 8 bits de ordem baixa do registro PC. O conteúdo do local da memória cujo endereço é mais do que o conteúdo do registro SP é transferido para os 8 bits de ordem alta do registro PC. O conteúdo do registro SP é incrementado de 2.



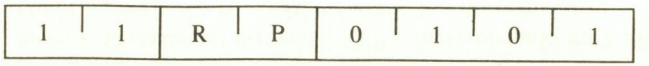
Endereçamento: registro indireto
Bandeiras: nenhuma

PILHA, ENTRADA-SAÍDA, E GRUPO DE CONTROLE DE MÁQUINA

Este grupo de instruções executa entrada-saída, manipula a pilha e altera as bandeiras de controle interno.

A menos que seja especificado o contrário, *as bandeiras de condição não são afetadas por quaisquer instruções neste grupo.*

EMPILHAR rp (empilhar)
((SP) − 1) ← (rh)
((SP) − 2) ← (rl)
(SP) ← (SP) − 2
O conteúdo do registro de ordem alta do par rp de registros é transferido para o local da memória cujo endereço é 1 menos do que o conteúdo do registro SP. O conteúdo do registro de ordem baixa do par rp de registros é transferido para o local da memória cujo endereço é 2 menos do que o conteúdo do registro SP. O conteúdo do registro SP é decrementado de 2. *Nota: O par de registros* rp = SP *pode não ser especificado.*



Endereçamento: registro indireto
Bandeiras: nenhuma

QUADRO 10.9 (CONT.) REPERTÓRIO DE INSTRUÇÕES DA UCP 8080

PUSH PSW (empilhar palavra de estado do processador)

((SP) − 1) ← (A)
((SP) − 2)$_0$ ← (CY), ((SP) − 2)$_1$ ← 1
((SP) − 2)$_2$ ← (P), ((SP) − 2)$_3$ ← 0
((SP) − 2)$_4$ ← (AC), ((SP) − 2)$_5$ ← 0
((SP) − 2)$_6$ ← (Z), ((SP) − 2)$_7$ ← (S)
(SP) ← (SP) − 2

O conteúdo do registro A é transferido para o local da memória cujo endereço é 1 menos do que o registro SP. O conteúdo das bandeiras de condição é montado numa palavra de estado do processador, e a palavra é transferida para o local da memória cujo endereço é 2 menos do que o conteúdo do registro SP. O conteúdo do registro SP é decrementado de 2.

1	1	1	1	0	1	0	1

Endereçamento: registro indireto
Bandeiras: nenhuma

POP rp (pop)

(rl) ← ((SP))
(rh) ← ((SP)) + 1)
(SP) ← (SP) + 2)

O conteúdo do local da memória, cujo endereço é especificado pelo conteúdo do registro SP, é transferido para o registro de ordem baixa do par rp de registros. O conteúdo do local da memória, cujo endereço é 1 mais do que o conteúdo do registro SP, é transferido para o registro de ordem alta do par rp de registros. O conteúdo do registro SP é incrementado de 2. *Nota: O par de registros* rp = SP *pode não ser especificado.*

1	1	R	P	0	0	0	1

Endereçamento: registro indireto
Bandeiras: nenhuma

POP PSW (desempilhar palavra de estado do processador)

(CY) ← ((SP))$_0$
(P) ← ((SP))$_2$
(AC) ← ((SP))$_4$
(Z) ← ((SP))$_6$
(S) ← ((SP))$_7$
(A) ← ((SP) + 1)
(SP) ← (SP) + 2

O conteúdo do local da memória cujo endereço é especificado pelo conteúdo do registro SP é usado para restaurar as bandeiras de condição. O conteúdo do local da memória cujo endereço é 1 mais do que o conteúdo do registro SP é transferido para o registro A. O conteúdo do registro SP é incrementado de 2.

1	1	1	1	0	0	0	1

Endereçamento: registro indireto
Bandeiras: Z,S,P,CY,AC

QUADRO 10.10 MODOS DE ENDEREÇAMENTO DA UCP 8080

MODOS DE ENDEREÇAMENTO

Quando forem usados dados numéricos de múltiplos bytes, os dados, como instruções, são armazenados nos sucessivos locais da memória, com o primeiro byte menos significativo, seguido por bytes crescentemente significativos. O 8080 tem quatro modos diferentes para endereçamento de dados armazenados na memória ou nos registros:

Direto Bytes 2 e 3 da instrução contêm o endereço exato da memória do item de dados (os bits de ordem baixa do endereço estão no byte 2, os bits de ordem alta, no byte 3).

Registro A instrução especifica o registro ou par de registros em que os dados estão localizados.

Registro indireto A instrução especifica um par de registros que contém o endereço da memória onde os dados estão localizados (os bits de ordem alta do endereço estão no primeiro registro do par, os bits de ordem baixa, no segundo).

Imediato A instrução contém os dados. Esta é uma grandeza ou de 8 bits ou de 16 bits (o primeiro byte menos significativo, segundo byte mais significativo).

A menos que orientada por uma instrução de INTERRUPÇÃO ou de DESVIO, a execução das instruções prossegue através dos aumentos consecutivos dos locais da memória. Uma instrução de DESVIO pode especificar o endereço da instrução seguinte a ser executada de um dos dois modos:

Direto A instrução de DESVIO contém o endereço da instrução seguinte a ser executada. (Exceto para a instrução RST, o byte 2 contém o endereço de ordem baixa e o byte 3, o endereço de ordem alta.)

Registro indireto A instrução de DESVIO indica um par de registro que contém o endereço da instrução seguinte a ser executada. (Os bits de ordem alta do endereço estão no primeiro registro do par, os bits de ordem baixa, no segundo.)

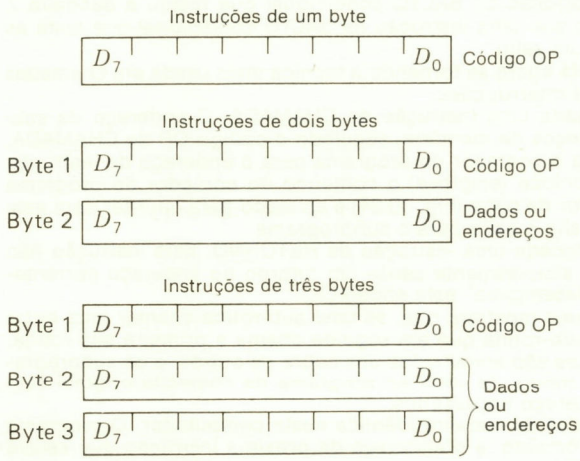


Fig. 10.25 Formatos da palavra de instrução para a UCP 8080.

QUADRO 10.11 BANDEIRAS USADAS NA UCP 8080

Palavra da bandeira

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	0	AC	0	P	1	CY

BANDEIRAS DE CONDIÇÃO

Há cinco bandeiras de condição associadas à execução de instruções no 8080. Elas são zero, sinal, paridade, transporte e transporte auxiliar, e cada uma é representada por um registro de 1 bit na UCP. Uma bandeira é "ajustada" forçando-se o bit para 1; o "reajuste" é feito forçando-se o bit para 0.

A menos que indicado o contrário, quando uma instrução afeta uma bandeira, ela a afeta da seguinte maneira:

Zero Se o resultado de uma instrução tiver o valor 0, esta bandeira está ajustada; do contrário, ela está reajustada.

Sinal Se o bit mais significativo do resultado da operação tiver o valor 1, esta bandeira está ajustada; do contrário, ela está reajustada.

Paridade Se a soma de módulo 2 dos bits do resultado da operação for 0 (isto é, se o resultado tiver paridade par), esta bandeira está ajustada; do contrário, ela está reajustada (isto é, se o resultado tiver paridade ímpar).

Transporte Se a instrução resultou num "vai um" (da adição) ou num empréstimo (da subtração ou de uma comparação) do bit de ordem alta, esta bandeira está ajustada; do contrário, ela está reajustada.

Transporte auxiliar Se a instrução causou um "vai um" do bit 3 e no bit 4 do valor resultante, o transporte auxiliar está ajustado; do contrário, ele está reajustado. Esta bandeira é afetada por simples operações de precisão de adições, subtrações, incrementos, decrementos, comparações e operações lógicas.

instruções aritméticas. Por exemplo, se uma adição for efetuada e o resultado for 0, então a bandeira Z será ajustada em 1 e as bandeiras S, P e C em 0 (contanto que nenhum transporte tenha sido gerado). Uma instrução de SALTO condicional que testou a bandeira Z causaria então um salto, enquanto que uma instrução de SALTO condicional que testa as bandeiras S, P ou C não causaria um salto.

Este computador usa o que está agora se tornando a técnica mais usada em chamadas de sub-rotina e nas reparações das interrupções.

Num salto de sub-rotina é usada uma instrução de CHAMADA. O endereço da sub-rotina está nos 16 bits (dois endereços da memória) seguindo o código OP de CHAMADA. Esta instrução primeiro incrementa o contador do programa para o endereço da instrução seguinte em seqüência e depois coloca (empurra) o conteúdo do contador do programa numa pilha na memória. O indicador de pilha (Fig. 10.24) é ajustado para apontar para este endereço na pilha. Em seguida é feito o salto para o subprograma.

No fim do subprograma, é colocada uma instrução de RETORNO. Esta instrução não especifica nenhum endereço, mas simplesmente causa um retorno no endereço correntemente no alto da pilha e depois "desempilha" este endereço.

Uma investigação deste esquema mostrará que, se uma sub-rotina chamar uma outra sub-rotina que chama uma outra sub-rotina que em seguida chama a primeira sub-rotina, os endereços sucessivos necessários são empilhados uns sobre os outros, e os subprogramas finalmente trabalharão a seu modo, de volta ao programa de chamada original sem perda de qualquer dos elos de endereço necessários.

As interrupções são manipuladas de maneira idêntica neste computador. O programa que está sendo executado é interrompido, e o endereço da próxima instrução que estava

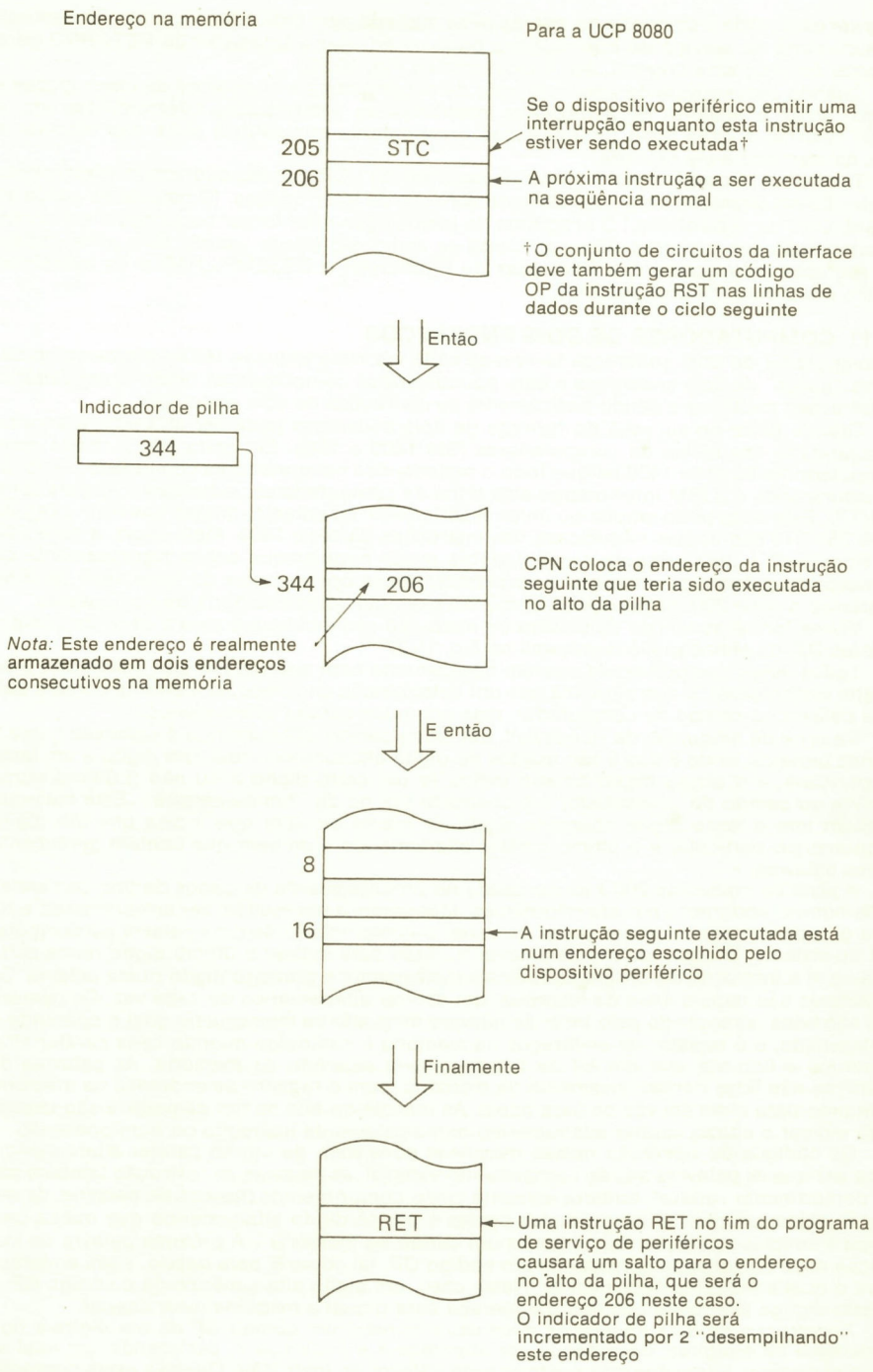


Fig. 10.26 Serviço de interrupções usando uma pilha.

para ser executada é colocado no alto da pilha mantida pelo indicador de pilha. Os retornos da sub-rotina de serviço de interrupções pode então usar a instrução de RETORNO para retornar ao programa original. Isto é mostrado na Fig. 10.26.

Quando esta espécie de empilhamento de subprograma e endereços de interrupções é usada, então é possível ter sub-rotinas interrompidas, interrupções interrompidas etc. e ainda retornar a cada faixa de instruções corretamente enquanto a pilha não estourar a área na memória a ela alocada.

Também é necessário economizar os registros na UCP quando ocorrerem interrupções se eles forem alterados pela sub-rotina de serviços de interrupções. (O mesmo se aplica às CHAMADAS de sub-rotinas.) O programa de interrupção deve tomar conta desta economia e restauração dos registros. Algumas idéias de como isto é feito usando uma pilha podem ser reunidas pelo exame das instruções de EMPILHAR e DESEMPILHAR e os exercícios tratam disso com maiores detalhes.

10.11 COMPUTADORES DE DOIS ENDEREÇOS

O computador de dois endereços tornou-se cada vez mais popular. Há inúmeros computadores "puros" de dois endereços e bem poucos outros computadores tendo acumuladores e instruções múltiplos e sendo basicamente as instruções de dois endereços.

Grande parte do sucesso do formato de dois endereços pode ser atribuído à enorme popularidade das séries de computadores IBM 1400 e 1600. Em certa época, havia mais computadores da série 1400 do que todo o restante dos computadores combinados. Isso foi há alguns anos e a IBM interrompeu esta linha de computadores, substituindo-a pela série 360/370. Esta série pode emular ou imitar a série 1400, no entanto, muitas das instruções da série 360/370 são cópias minuciosas das instruções da série 1400. Além disso, a série 200 da Honeywell é uma linha de computadores muito popular que segue rigorosamente os formatos da palavra de instrução e as características operacionais da série 1400 da IBM. Usaremos a série 200 da Honeywell como um exemplo desta categoria de computador.

Vários formatos típicos da palavra de instrução são mostrados para a série de computadores 200 da Minneapolis-Honeywell na Fig. 10.27.

Todos esses computadores operam basicamente num sistema em que a memória é de *caráter endereçável,* o que significa que um determinado endereço na memória contém não uma palavra ou campo de computador, mas um único caráter alfanumérico.

Na série de máquinas da Honeywell, um único caráter alfanumérico é expresso como 9 dígitos binários, onde 6 dos 9 são usados no dígito alfanumérico real. Um dígito é um teste de paridade, e o oitavo dígito binário indica se um certo dígito é ou não o último numa palavra ou campo do computador e é chamado um bit de "fim-de-palavra". Esta máquina também tem o nono dígito chamado *dígito de marca de item* que indica que um dígito alfanumérico particular é o último caráter alfanumérico num item que contém geralmente várias palavras.

A série de máquinas 200 é muito usada no processamento de dados de tipo comercial, onde nomes, endereços e outra informação alfanumérica necessitam ser armazenados, e há uma grande vantagem em ser capaz de variar o comprimento de uma palavra de computador no sistema. O dígito de fim-de-palavra é usado para indicar o último dígito numa certa palavra, e a instrução do programa indica simplesmente o primeiro dígito numa palavra. Os caracteres são depois lidos da memória, um caráter alfanumérico de cada vez. Os operandos são lidos começando pelo local de número mais alto na memória no qual o operando é armazenado, e o registro de endereços na memória é diminuído quando cada caráter alfanumérico é lido até que um bit de fim-de-palavra seja lido da memória. As palavras de instrução são lidas consecutivamente da memória, com o registro de endereço da memória contando para cima em vez de para baixo. As marcas ou bits de fim-de-palavra são usadas para indicar o último caráter alfanumérico numa palavra de instrução ou num operando.

Os códigos de operação nestas máquinas consistem de um só caráter alfanumérico. Uma vez que as palavras são de comprimento variável, as palavras de instrução também são de comprimento variável, embora somente cinco comprimentos básicos de palavras de instrução sejam usados. O mais simples destes é um só dígito alfanumérico que indica uma única e simples operação, tal como ler um cartão na memória". A próxima palavra de instrução mais complicada consiste de um código OP, tal como *B* para desvio, e um endereço para o qual a máquina deve desviar. Neste caso, um dígito alfanumérico dá o código OP, e quatro dígitos alfanuméricos dão o endereço para o qual a máquina deve desviar.

A instrução de dois endereços mais usual contém um código OP de um dígito e dois endereços de 4 dígitos, designados por endereço *X* e endereço *Y*, perfazendo um total de nove caracteres alfanuméricos neste tipo de palavra de instrução. Quando estes computadores recebem um código OP tal como *A*, indicando adição, eles adicionam o número loca-

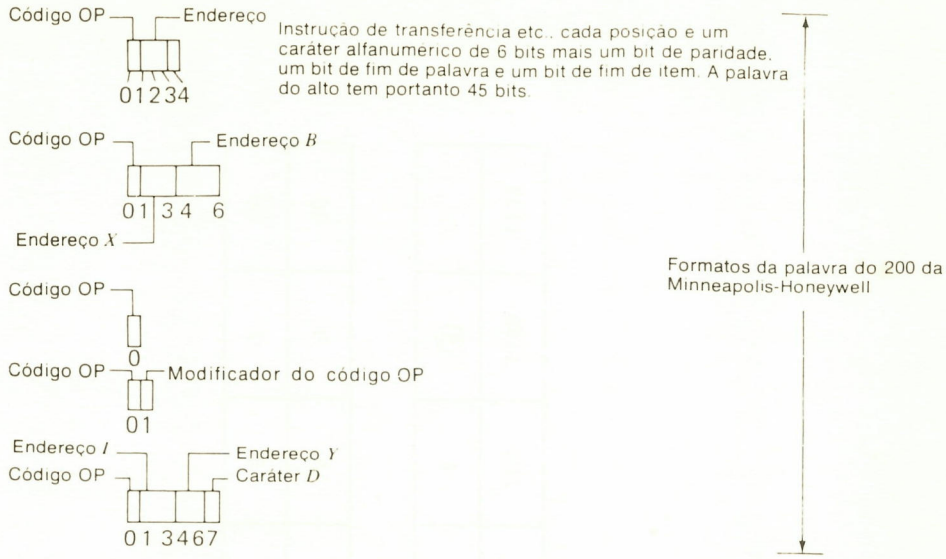


Fig. 10.27 Formatos típicos da palavra de instrução.

lizado no endereço fornecido na parte de endereço *X* ao número localizado no endereço fornecido na parte de endereço *Y*, e armazenam este número no local da memória fornecido na parte de endereço *Y*.

Há também palavras de instrução com somente um caráter de código OP e um modificador de código OP, e até palavras de instrução mais longas que contêm um código OP, um endereço *I*, um endereço *Y* e um caráter *D*. O caráter *D* para este último tipo de instrução é usado para aumentar o número de instruções; assim, o caráter *D* é usado para modificar o significado do código OP expresso no único caráter alfanumérico, empregado como caráter básico do código OP.

A operação interna não é muito diferente do tipo de máquina de endereço único descrito na seção sobre implementação de conjunto de circuitos. As principais diferenças surgem da natureza BCD da aritmética e do fato de que a memória é lida de maneira em série-paralelo. No entanto, os mesmos registros básicos são usados no conjunto de circuitos de controle; portanto, o registro de código OP, o contador de instruções, o registro de endereços da memória e os registros da memória intermediária são muitos como descritos anteriormente, bem como as instruções LER e GRAVAR para a memória e o fato de que a máquina alterna entre ciclos de instrução e ciclos de execução. Uma diferença principal é que os ciclos de execução estão sujeitos a requerer muito mais tempo do que os ciclos de instrução, já que são usados dois endereços na memória, e portanto dois operandos, e cada um pode requerer vários acessos de memória. Observamos também que devem ser mantidos dois contadores operando, a fim de manipular o endereçamento alternado dos dois operandos na memória.

Como foi mencionado, cada caráter no sistema Minneapolis-Honeywell consiste de 9 dígitos binários, dos quais 6 são usados para o caráter alfanumérico. Os três bits restantes consistem de um dígito de paridade, um dígito de marca de palavra e de um dígito de item de marca. Para simplificar a programação, a Honeywell desenha um círculo ao redor daqueles caracteres alfanuméricos em que há uma marca de palavra de 1 indicando o fim de uma palavra. Então, as palavras de instrução endereçam simples caracteres na memória, e a máquina entra em seqüência através de endereços crescentes consecutivos, puxando caráter após caráter para fora da memória e incrementando o registro de endereços de instruções ou a memória quando cada caráter é puxado para fora, até que seja alcançado um caráter com uma marca de palavra de 1.

A Fig. 10.28 mostra uma palavra típica da instrução com o código *A* de operação, um endereço *X* para 1197 e um endereço *Y* para 9. Esta palavra de instrução deve ocupar 9 posições de caráter ou endereços na memória e deve ser lida em série na memória. No

	Código OP	Endereço *X*	Endereço *Y*
Palavra da instrução	Ⓐ	1197	0009

Endereço na memória	1193	1194	1195	1196	1197	1198	1199
Dados em cada local	⑤	9	2	6	4	⑤	

Endereço na memória	4	5	6	7	8	9	10
Dados em cada local	6	②	4	2	5	7	⑨

Fig. 10.28 Execução de palavra no Honeywell 200.

entanto, o endereço *X* deve ser armazenado num registro, o endereço *Y* num outro registro e o código OP no registro de código OP.

O significado desta instrução é o seguinte: Somar o campo ou número que começa no endereço 1197 na memória ao campo ou número que permanece no endereço 9 na memória e colocar a soma destes dois números começando no endereço 9. Esta instrução particular opera em caracteres BCD, e possíveis conteúdos de cada um dos locais dos caracteres na memória também são mostrados na figura. A máquina continuará adicionando caracteres enquanto diminui ambos dos contadores de instruções até encontrar uma marca de palavra na série de caracteres do endereço *Y,* em cujo caso ela terminará a adição. Por exemplo, o caráter 2 no local 5 da memória está envolvido por um círculo, indicando que o bit de marca de palavra é um 1 para aquele caráter particular.

O computador adicionará primeiro o dígito no local 9 ao dígito no local 1197; assim, ele soma 4 mais 7, coloca um 1 no local 9 e armazena um dígito de transporte a ser acrescentado ao par de dígitos seguinte. Os dígitos seguintes estão nos locais 8 e 1196, que são 5 e 6, mais um transporte de 1 do último tempo: portanto, um 2 é colocado no local 8 e um 1 é transportado para a adição dos próximos dígitos. Estes permanecem em 1195 e 7, respectivamente. A adição destes resulta num 5 sendo colocado no local 7. Depois o caráter em 1194, que é um 9, é adicionado ao número no local 6, dando uma soma de 3 mais um transporte, e, finalmente, adicionamos 5 e 2 mais um transporte, dando 8 a ser colocado no local 5. Após a adição, os dados nos locais 1193 e 1197 são como quando a adição foi iniciada, enquanto que o número nos locais 5 a 9 na memória contém agora 83521.

Este computador também pode somar números binários, caso em que seis dos dígitos em cada local de caráter são considerados compreender seis dígitos de um número binário e são adicionados em forma de complemento de 2.

A Fig. 10.29 mostra uma outra palavra típica de instrução, que é a instrução de SUBTRAIR, o código OP para esta sendo um *S;* o endereço *X* fornecido é 297 e o endereço *Y* é 164. Nove locais na memória devem ser usados para a palavra de instrução mostrada, um local para cada um dos caracteres alfanuméricos. A palavra de instrução causa a subtração dos dados numéricos no campo *X* dos dados numéricos no campo *Y* e o armazenamento desta diferença no campo *Y.* A subtração termina quando uma marca de palavra é encontrada; portanto, ela terminará quando o caráter em 293 for subtraído do caráter em 160 na memória.

A execução desta palavra de instrução fará com que 65143 seja subtraído de 83182, e a diferença, 18039, seja colocada nos locais 160 a 164 na memória, substituindo o 83182 que esteve lá no começo da execução da instrução.

Uma outra palavra de instrução, uma instrução de "desviar se caráter igual", também é mostrada na figura. O código OP para esta é *B.* A figura indica um endereço *I* que é 93, um endereço *Y* que é 167 e um chamado caráter *D* que é *A.* Dez caracteres alfanuméricos seriam necessários para armazenar esta palavra de instrução particular; portanto, ela ocuparia dez locais na memória. Esta instrução faz com que o único caráter no endereço *Y* seja comparado ao caráter *D,* e, se ele tiver a mesma configuração de bit que o caráter *D,* o programa desvia para o endereço *I.* Do contrário, o programa continua seqüencialmente. O caráter *D* pode ser qualquer combinação de caracteres desejada dos seis dígitos usados para cada dígito BCD. (A presença ou ausência de uma marca de palavra não é considerada.) Para cada palavra de instrução particular fornecida, se o caráter localizado no endereço 167 for um *A,* o programa tomará a palavra de instrução seguinte começando no local 93 na memória. Se o caráter não for um *A,* a palavra de instrução seguinte será puxada da memória no endereço seguinte a esta palavra de instrução particular.

Um outro exemplo de linhas de computadores de dois endereços é o das máquinas Nova e Super Nova fabricadas pela Data General Corporation. Estes computadores são minicomputadores e os mesmos quanto à organização geral, e portanto descreveremos o Nova. A palavra básica nesta máquina tem 16 bits, conforme indicado na Fig. 10.30. Este computador é um exemplo de um computador com *múltiplos acumuladores.* O Nova tem quatro acumuladores, e uma instrução básica aritmética ou lógica se refere a dois destes acumuladores. Todas as instruções aritméticas ou lógicas efetuam apenas operações entre acumuladores. Por exemplo, uma instrução E faria uma operação E com os conteúdos dos dois acumuladores especificados pela palavra de instrução — não um acumulador e um local na memória principal (núcleos).

Uma vez que 2 bits são necessários para endereçar cada um dos quatro acumuladores, somente 4 bits são necessários para ambos dos endereços numa instrução de dois endereços. Conforme mostrado na figura, são usados bits 1 a 4 para endereçamento; bits 1 e 2 endereçam o *acumulador fonte,* chamado ACS, e os bits 3 e 4 endereçam o *acumulador de destino,* chamado ACD.

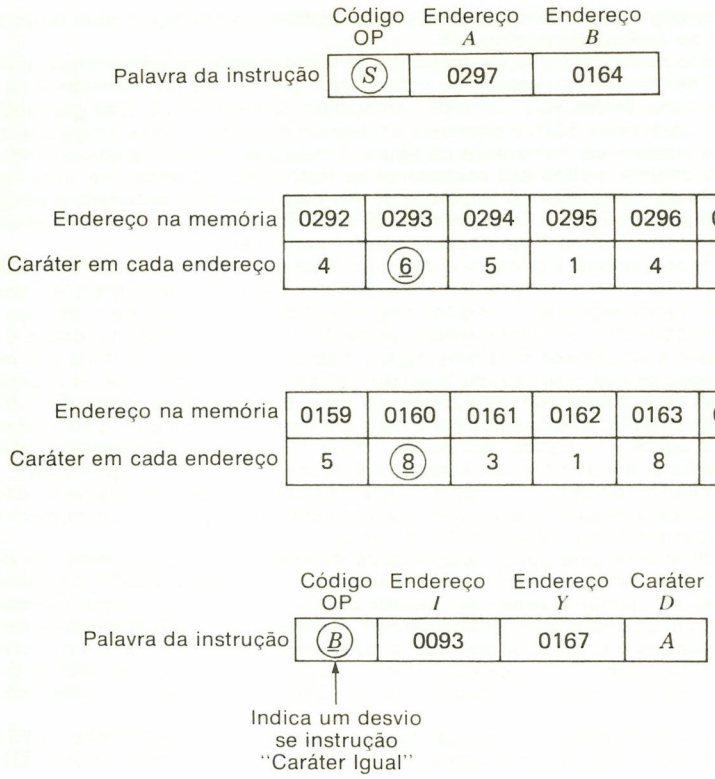


Fig. 10.29 Instrução de SOMAR no 200 da Minneapolis-Honeywell.

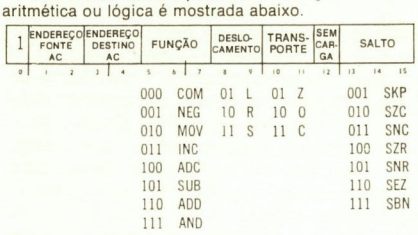
Uma instrução de adição no Nova faz com que o acumulador endereçado pelos bits 1 e 2 (isto é, ACS) seja adicionado ao acumulador endereçado pelos bits 3 e 4 (isto é, ACD), e o resultado é colocado no ACD. Uma vez que a aritmética é apenas entre os registros acumuladores, estes devem ser carregados com os dados da memória antes que as operações sejam efetuadas, e as *instruções de referência à memória* tais como CARREGAR, que transfere a palavra endereçada na memória para dentro do acumulador especificado, e ARMAZENAR, que transfere o conteúdo do acumulador selecionado para dentro do local da memória endereçada, são usadas para mover dados para dentro e para fora dos acumuladores.

Nas instruções de referência à memória, os locais da memória são endereçados por um *deslocamento,* que ocupa os bits 8 a 15, e um *índice,* que se refere a um dos acumuladores, e o endereço real usado é a soma do número no acumulador selecionado e do deslocamento (ver figura). O endereçamento indireto é indicado por um 1 no bit 5 de uma instrução de referência à memória, e um 1 no bit de sinal de uma palavra endereçada indiretamente indica que ele é um outro endereço indireto, conforme foi explicado anteriormente (a máquina permite encadeamento indireto de endereços).

As instruções aritméticas têm opções para operações adicionais após ser efetuada a operação aritmética básica. Portanto, um resultado pode ser deslocado (observar os bits de deslocamento na Fig. 10.30) ou não usado [ver o bit NO LD (para NO LOAD)], ou a instrução pode selecionar dentre várias opções ou saltar, se ou não um resultado aritmético for positivo, negativo, zero, estouros etc. Esta capacidade de combinar várias operações a serem efetuadas em ordem numa única instrução é usada por diversos fabricantes de minicomputadores.

INSTRUÇÕES ARITMÉTICAS E LÓGICAS

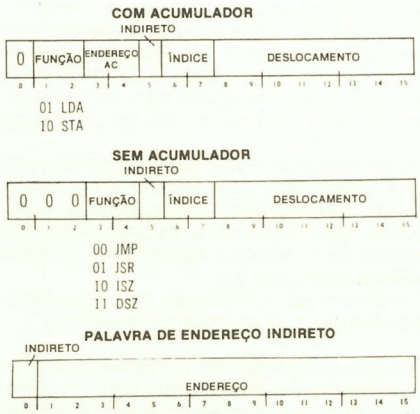
A estrutura de uma palavra de instrução aritmética ou lógica é mostrada abaixo.



Cada instrução nesta classe especifica um ou dois acumuladores para fornecer operandos para o gerador de funções, que executa a função especificada e produz um bit de transporte, cujo valor depende de um valor de base especificado pela instrução, da função executada e do resultado obtido. O valor de base pode ser deduzido do bit bandeira de transporte, ou a instrução pode especificar um valor independente.

INSTRUÇÕES DE REFERÊNCIA DA MEMÓRIA

Há dois formatos para as instruções de referência da memória, dependendo de um acumulador ser especificado.



ALGUMAS INSTRUÇÕES TÍPICAS

COM Complemento. Colocar o complemento (lógico) da palavra de ACS em ACD.

NEG Negar. Colocar o complemento de 2 do número de ACS em ACD. Complementar transporte se ACS contiver zero. (A formação do complemento de dois de zero gera um transporte, porque a complementação do zero produz uma palavra que contém todos os 1, e, somando 1 àquele resultado, produz todos os zeros novamente, mais um transporte.)

MOV Movimentar. Colocar o conteúdo de ACS em ACD.

INC Incrementar. Somar 1 ao número de ACS e colocá-lo em ACD. Se o resultado for 2^{16}, complementar transporte.

ADC Complementar soma. Somar o complemento (lógico) do número de ACS ao número de ACD e colocar o resultado em ACD. Se o original ACD > ACS, complementar transporte.

SUB Subtrair. Somar o complemento de dois do número de ACS ao número de ACD e colocar o resultado em ACD. Se o original ACD ≥ ACS, complementar transporte.

ADD Somar. Somar o número de ACS ao número de ACD e colocar o resultado em ACD. Se o resultado for ≥ 2^{16}, complementar transporte.

AND E. Colocar a função lógica E da palavra de ACS e da palavra de ACD em ACD.

OBSERVAÇÕES SOBRE ENDEREÇAMENTO DA MEMÓRIA

Cada palavra da instrução contém informação para a determinação do endereço efetivo, que é o endereço real usado para buscar ou armazenar o operando ou alterar o fluxo do programa. A informação de endereço compreende um deslocamento de 8 bits, uma seleção de índice de 2 bits e um único bit indireto. O deslocamento pode endereçar diretamente qualquer local em quatro grupos de 256 locais cada um. Ele pode ser um endereço absoluto. Assim, ele pode ser usado simplesmente para endereçar um local na página zero, os primeiros 256 locais na memória. Ele pode também ser tomado como um número com sinal usado para computar um endereço absoluto adicionando-se o deslocamento a um endereço base de 15 bits fornecido por um registro-índice. Os bits de índice podem selecionar AC2 ou AC3 como registro-índice; qualquer um desses acumuladores pode portanto ser usado como um registro-índice ordinário para variar o endereço computado de um deslocamento constante, ou como um registro base para um conjunto de diferentes deslocamentos. O programa pode também selecionar o contador de programas como o registro-índice, e assim qualquer instrução pode endereçar 256 palavras em sua própria vizinhança (endereçamento relativo).

O endereço (15 bits) absoluto computado pode ser o endereço efetivo. No entanto, a instrução pode usá-lo como um endereço indireto. Assim, ele pode especificar um local a ser usado para recuperar um outro endereço. A palavra lida de um local endereçado indiretamente contém um endereço absoluto e um bit indireto; este endereço pode ser o endereço efetivo ou pode ser um outro endereço indireto.

Fig. 10.30 Formatos da palavra de instrução para o computador Nova. (Cortesia da Data General Corp.)

10.12 SÉRIE DE COMPUTADORES IBM 360/370

Consideraremos agora a série de computadores IBM 360/370. Esta linha abrange desde os pequenos até os maiores computadores. Os projetos são arranjados de modo que os computadores sejam *compatíveis para cima*, e o significado disto é que um programa escrito para um computador pequeno passará em qualquer computador na série 360/370 que seja maior. (O artifício geral na determinação "maior" é que o 40 é maior do que o 30, o 50 é maior do que o 40 etc., o 370/158 é maior do que qualquer destes e, em geral, um número de modelo maior significa um computador maior. Para manter os computadores compatíveis para cima, o repertório de instruções de um pequeno computador é feito um subconjunto do repertório de instruções de um computador maior.

Os vários modelos são projetados em diferentes laboratórios da IBM e usam diferentes linhas de circuito e tecnologias de memória, e alguns são microprogramados enquanto que outros não. Apesar disso, mantendo as arquiteturas consistentes e padronizando os repertórios de instrução, é possível que esta compatibilidade para cima funcione. Outros fabricantes incidentalmente fazem a mesma coisa, e a série PDP-11 da DEC, a série 6000 da Honeywell etc. são todas linhas de computadores em que os repertórios de instrução e as capacidades gerais dos computadores maiores são versões expandidas dos repertórios de instrução e capacidades dos pequenos computadores.

A série 360/370 da IBM também é projetada de modo que ela possa competir eficientemente tanto no mercado de computação científica quanto no mercado de computadores de processamento de dados comerciais. A área de aplicação científica pede um repertório de instruções incluindo instruções numéricas que operam (somar, subtrair, multiplicar etc.) tanto com representação de inteiros quanto com números de ponto-flutuante. Estes cálculos devem ser efetuados com operandos que são relativamente longos (neste caso, longo representa 32 e 64 bits por operando). As aplicações comerciais, por outro lado, demandam instruções que operam com cordões de caracteres (como na série 1400) que consistem de caracteres alfanuméricos de 8 bits e são de comprimento arbitrário tal como pode ser achado nos nomes, endereços, descrições de inventário etc. É também uma boa idéia em computadores de processamento de dados comerciais ser capaz de calcular usando BCD (já que isto reduz o tempo de conversão), e tais números são também sujeitos a serem de comprimentos diferentes e geralmente estão armazenados em cordões.

A resposta da IBM a estas demandas é ter um repertório de instruções muito grande que inclua as instruções necessárias a satisfazer ambas as áreas de aplicação. Então, os compiladores Fortran — que geralmente manipulam os problemas do tipo numérico — geram instruções de aritmética binária orientadas para a palavra, enquanto que os compiladores Cobol geram instruções de manipulação de cordões de caracteres.

Uma outra consideração importante nos grandes sistemas é o uso de um *sistema operacional* programado para controlar a operação do computador. Um sistema operacional consiste de um conjunto de programas que são usados para controlar a execução do programa do *usuário* ou programas de *aplicações*. Para usar o sistema, os programas do sistema operacional são primeiramente lidos, e depois estes programas iniciam e "operam" outros programas lendo-os na memória, iniciando-os, parando programas que passam pela máquina num tempo muito longo ou que usam memória em excesso etc. Estes programas auxiliam efetivamente os operadores manuais do sistema a fim de obter a produtividade máxima do sistema. O sistema operacional às vezes datilografa comandos para os operadores nas impressoras do sistema, dizendo-lhes para montar unidades de fita ou de discos, para acrescentar papel às impressoras etc.

A fim de usar grandes sistemas em seu máximo de potencialidade, eles são geralmente utilizados no modo de *multiprogramação* onde o sistema operacional lê num programa e, se ele pedir alguma operação lenta (uma operação de entrada-saída, por exemplo), o sistema operacional então lê e inicia um outro programa. Isso continua, e os programas são reiniciados onde eles foram por último interrompidos pelo sistema operacional quando entrada-saída é disponível ou quando outros programas terminam etc.

A fim de ajudar o escritor do sistema operacional e tornar a operação do computador mais eficiente, muitos computadores têm agora arquiteturas que permitem o seguinte:

1. Um certo programa é *relocalizável*. Isto significa que um programa pode ser colocado em locais diferentes na memória em tempos diferentes e ele ainda passará pela máquina. Isto permite ao sistema operacional localizar vários programas diferentes nas diferentes partes da memória e operar primeiro sobre um programa e depois sobre outros, conforme a conveniência.
2. Deve haver algum modo do sistema operacional estabelecer limites superior e inferior na memória usada por um determinado programa. Alocando uma área específica na memória a um programa e não permitindo que o programa use outros locais na memória, pode-se evitar que o programa danifique a operação de outros programas que estão na memória ao mesmo tempo. Isso também ajuda no que é chamado *segurança*, que trata de evitar que os programas leiam deliberadamente outros programas ou interfiram de outro modo com eles. Por exemplo, um usuário pode tentar e ler os arquivos de um outro usuário, roubar aqueles programas do usuário, emitir um cheque para si próprio etc.

A fim de ajudar a obter os objetivos acima, a série 360/370 da IBM (e a maioria dos grandes fabricantes de computadores) tem duas classes de instruções: *instruções do modo supervisor* e *instruções do modo do usuário*. Os programas do sistema operacional usam as *instruções do modo supervisor*, e, quando elas estão sendo executadas, um biestável está ativado na UCP, e assim o computador sabe se ele está neste modo. Quando os programas do modo de aplicação ou do usuário estão sendo executados, certas instruções privilegiadas não são disponíveis para o programa. Incluídas nas instruções do modo supervisor, mas não nas instruções do modo de aplicação, estão aquelas que estabelecem os limites superior e inferior na memória onde um programa pode operar e controlar a entrada-saída — em particular, a leitura de arquivos — e várias instruções especiais para manipular emer-

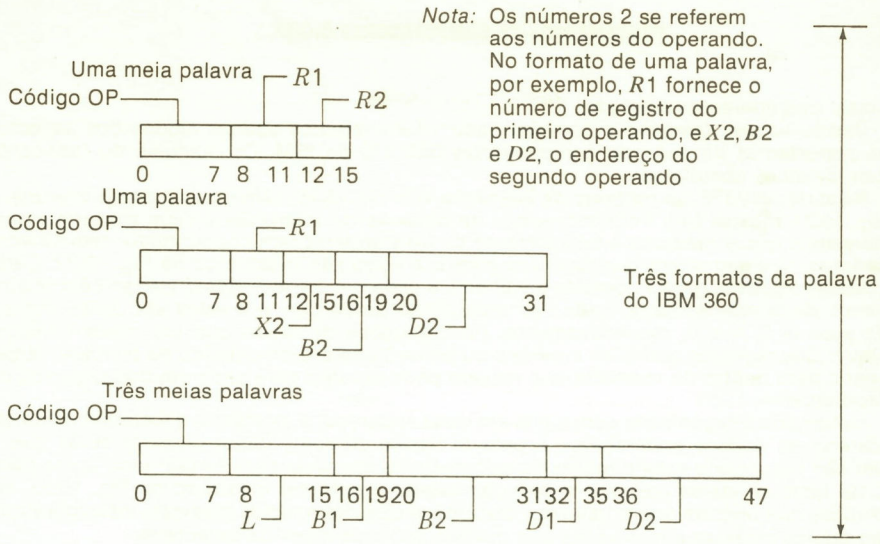


Fig. 10.31 Formatos típicos da palavra de instrução.

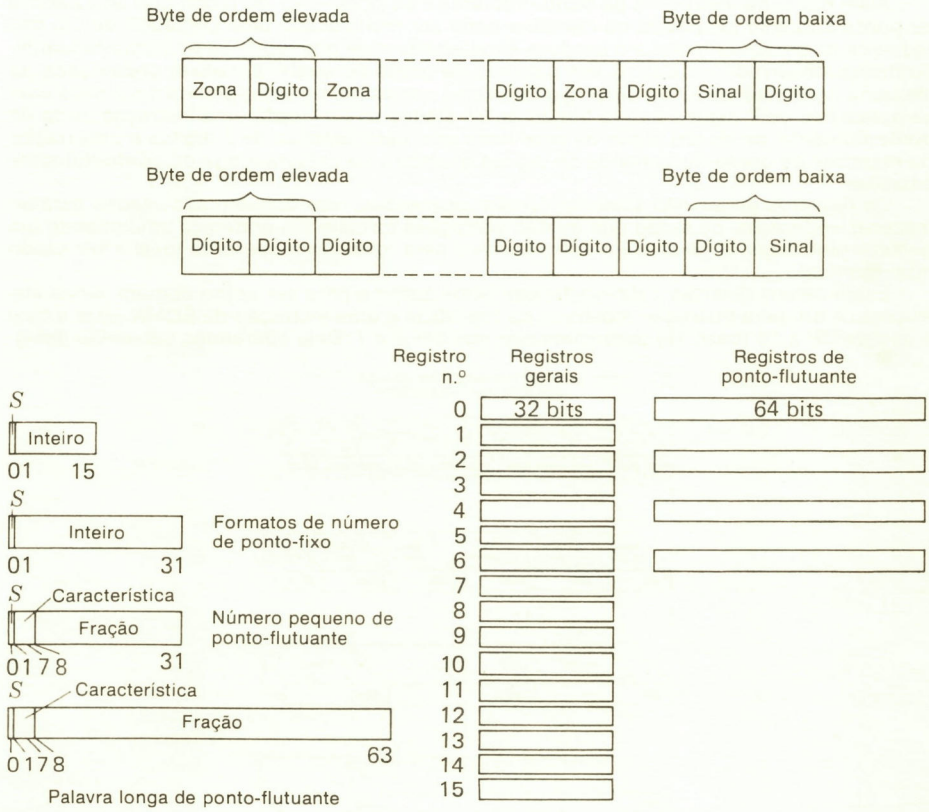


Fig. 10.32 Registros gerais.

gências, operadores de sinal etc.

Devido ao seu tamanho e complexidade, examinaremos apenas alguns dos aspectos mais importantes da série de computadores 360/370 da IBM. Os manuais do fabricante trazem detalhes completos.

Na série 360/370, as palavras de instrução vêm em vários tamanhos e comprimentos, e a Fig. 10.31 mostra três tipos diferentes de palavras de instrução e seus comprimentos. Observemos que a máquina é basicamente de dois endereços. O computador tem 16 acumuladores, que são chamados *registros gerais,* e estes são mostrados na Fig. 10.32. Cada um desses registros gerais consiste de 32 bits, e cada um é endereçável. Há também quatro *registros de ponto-flutuante,* cada um consistindo de 64 bits, e a estes são atribuídos os endereços 0, 2, 4 e 6, respectivamente. Há instruções de modo que um possa carregar qualquer dos registros gerais da memória ou possa transferir o conteúdo de qualquer registro geral para dentro da memória, e o mesmo pode ser dito para cada um dos registros de ponto-flutuante.

A memória é organizada com 8 bits em cada endereço, e, portanto, a memória é dita ser de *caráter* ou de *byte endereçável.* A palavra básica do computador consiste de 32 bits, e assim são requeridos quatro locais consecutivos de memória para formar uma única palavra. Há também *meias palavras* e *palavras duplas,* conforme mostrado na Fig. 10.33. Há instruções que operam sobre palavras, instruções que usam meias palavras, instruções que usam duplas palavras e instruções que operam sobre cordões de caracteres.

Em geral, quando adicionamos uma palavra de 32 bits a uma outra palavra, podemos (1) adicionar um dos registros gerais a um outro registro geral e armazenar a soma num dos registros gerais, ou (2) adicionar um dos registros gerais a uma palavra localizada na memória e armazenar a soma no registro geral.

Além disso, há instruções de ponto-flutuante e de ponto-fixo, de modo que uma palavra de ponto-flutuante localizada na memória pode ser multiplicada pelo conteúdo de um dos registros de ponto-flutuante e o produto armazenado num dos registros de ponto-flutuante. Conforme observado no Cap. 6, em palavras de ponto-flutuante, a palavra endereçada na memória consistirá de 32 ou de 64 dígitos. Estes consistem de um dígito de sinal mais uma característica de 7 dígitos e uma fração de 24 dígitos se for usada uma instrução curta de ponto-flutuante, ou de um dígito de sinal mais uma característica de 7 dígitos e uma fração ou mantissa de ponto-flutuante de 56 dígitos quando uma palavra longa de ponto-flutuante é usada.

Os registros gerais não somente contêm operandos, mas também são usados para armazenar endereços, de modo que muitas instruções formam um endereço adicionando um número num registro geral a um outro número para formar o número do local a ser usado num operando.

Examinemos diversas palavras típicas neste sistema para ver como operam várias instruções. A primeira instrução mostrada na Fig. 10.34 é uma instrução de SOMA, para a qual o código OP é 1*A* (base 16) permanecendo nos bits 0 a 7. Dois operandos são então desig-

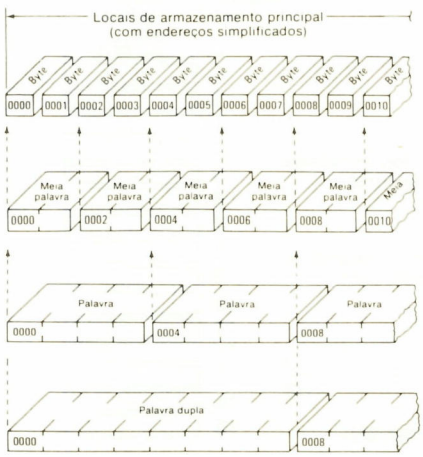


Fig. 10.33 Organização da memória — meias palavras, palavras, palavras duplas e bytes.

nados. Estes operandos são localizados no registro geral 7 e no registro geral 9, de modo que o número no registro geral 7 será adicionado ao número no registro geral 9, e a soma será armazenada no registro geral 7. Observemos que os dois endereços fornecidos neste tipo particular de instrução na realidade indicam diretamente os locais das duas palavras a serem adicionadas.

A segunda instrução mostrada, que é uma instrução de ARMAZENAMENTO, é mais complicada. O código de operação de armazenamento neste sistema é 50. Este código OP é acompanhado na palavra de instrução por quatro números no sistema. O primeiro número, que permanece nos bits 8 a 11, dá o número de um registro geral a ser usado; os conteúdos deste registro devem ser armazenados na memória. O segundo conjunto de três números ocupa os bits 12 a 31 na palavra de instrução e dá o endereço na memória onde este número deve ser armazenado. No entanto, este endereço não é fornecido diretamente. O endereço é, em vez disso, formado adicionando-se o número no registro geral 10 ao número no registro geral 14, tomando-se os 24 dígitos menos significativos deste número, e adicionando estes ao número armazenado nos bits 20 a 31 na palavra de instrução, que neste caso é 310. Por exemplo, se o registro geral 10 contivesse o número 400 e se o registro geral 14 contivesse o número 200, adicionaríamos estes dois números, obtendo 600, e adicionaríamos isto a 310, obtendo o número 910, de modo que a palavra no registro geral 3 seria armazenada no local 910 na memória.

As instruções de subtrair operam um tanto similarmente. A instrução de subtrair é indicada pelo código OP *5B*, e o tipo particular de instrução mostrado na figura subtrai o número armazenado num endereço que é determinado pela adição dos números armazenados no registro geral 9 ao número armazenado no registro geral 8, e a isto adicionando-se o número que permanece nos bits 20 a 31 da instrução de subtrair. Quando estes três números forem combinados para formar um endereço na memória, a palavra localizada naquele endereço é então subtraída da palavra localizada no registro geral 4, e a diferença é armazenada no registro geral 4.

Por exemplo, se o registro geral 4 contiver 300, o registro geral 9 contiver 100, o registro geral 8 contiver 200 e, na memória, o local 800 contiver a palavra 1000, formaremos o número de endereço 800 adicionando o número 200 que permanece no registro geral 8 ao número 100 que permanece no registro geral 9, e depois adicionando isto ao 500 fornecido nos bits 20 a 31 na palavra de instrução, obtendo o número 800. Olhamos então a memória no local 800 e achamos a palavra 1000. Se este número for subtraído de 300, que permanece no registro geral 4, a diferença −700 será armazenada no registro geral 4 quando a instrução estiver concluída.

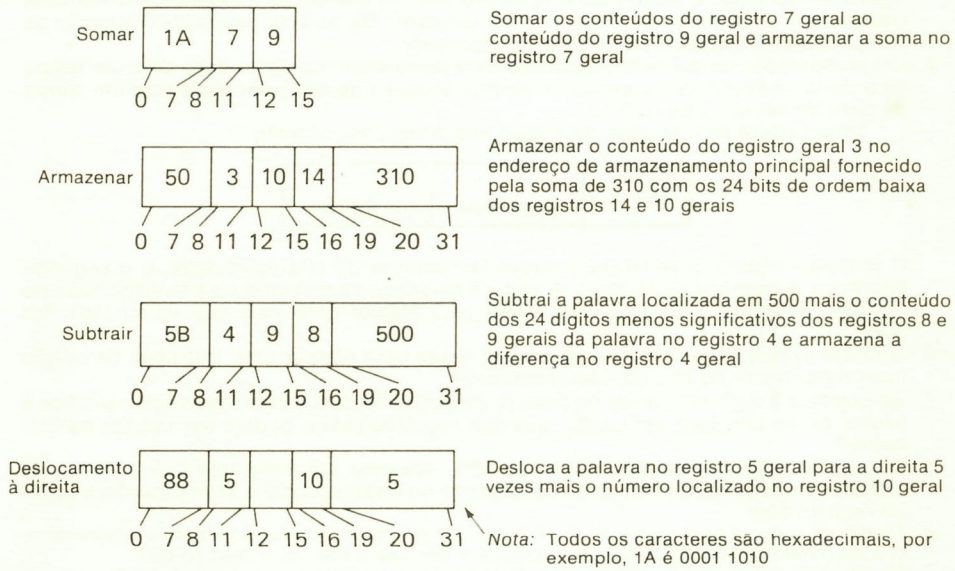


Fig. 10.34 Palavras de instrução típicas.

Como um último exemplo do código de instrução desta máquina, examinemos uma instrução de DESLOCAMENTO À DIREITA, para a qual o código OP é 88. Neste caso, o registro geral indicado pelos bits 8 a 11 na palavra de instrução deve ser deslocado à direita. O número de vezes que a palavra deve ser deslocada é calculado pela adição dos números nos bits 20 a 31 da palavra de instrução ao número no registro designado pelos bits 16 a 19 da palavra de instrução.

No exemplo mostrado, o registro geral 10 é convocado pelos bits 16 a 19; assim, o número armazenado no registro geral 10 será adicionado ao 5 armazenado nos bits 20 a 31, e o registro geral 5 será deslocado à direita, isto muitas vezes. Se, por exemplo, o registro geral 10 contiver o número 7, então devemos somar 7 a 5, dizendo que a palavra no registro geral 5 deve ser deslocada à direita num total de 12 vezes.

Há diversas razões para a técnica de endereçamento complicado usada aqui. O uso de dois registros de propósitos gerais para formar cada endereço é (1) permitir que os programas do sistema operacional relocalizem os programas dos usuários na memória, (2) permitir ao programador usar um dos registros como um registro-índice. A prática comum no 360/370 é para o segundo registro em cada endereço (o especificado nos bits 12 a 15) ser usado apenas pelo sistema operacional. Este registro é então chamado *registro base*, e, se o registro base for o mesmo — seja o registro 5 — em cada palavra de instrução num programa, então o número neste registro será adicionado a cada endereço gerado. Colocando um número apropriado neste registro, o sistema operacional pode fazer com que um certo programa seja carregado em locais de sua escolha na memória, e o programa irá operar ali porque cada endereço de palavra de dados e de palavras de instrução será apropriadamente localizado. Isso então deixa o registro restante — o especificado pelos bits 11 a 13 — para ser usado pelo programador como um registro-índice.

Há um conjunto de instruções, neste computador, similar aos mostrados para a série 200 da Minneapolis-Honeywell. Esta série de computadores é típica dos computadores usados nos grandes sistemas quanto a que ela tem um rico repertório de instruções e muitas características que os projetistas sentem serem úteis ao usuário do computador. Algumas linhas de computadores são muito mais orientadas para as aplicações científicas, e algumas outras são quase que exclusivamente orientadas para as atividades comerciais. No entanto, várias firmas se dedicam à fabricação de linhas de computadores destinadas às aplicações científicas e também às aplicações comerciais.

EXERCÍCIOS

1. Um computador de três endereços tem um comprimento de palavra de 11 dígitos decimais e mais o sinal. O computador pode executar 50 instruções diferentes, e a memória interna tem 1.000 palavras. Desenhar um diagrama da palavra básica da instrução do computador, deixando o bit de sinal abandonado.
2. Um computador de dois endereços tem uma única memória de núcleos com um tempo de ciclo de memória de 1 μs e uma memória pequena de alta velocidade com um tempo de ciclo de memória de 0,25 μs.

   Uma palavra de instrução de adição tem o seguinte aspecto:

ADICIONAR	1.º endereço	2.º endereço

   O primeiro endereço se refere à pequena memória de alta velocidade, e o segundo endereço, à memória de núcleos. A soma é colocada na memória de alta velocidade no primeiro endereço. Quanto tempo levará para efetuar uma instrução de adição? Por quê?
3. Quantas excursões à memória devem ser feitas para efetuar uma instrução de adição num computador do tipo de três endereços?
4. Se usarmos 3 dígitos binários na palavra de instrução para indicar que registro-índice é usado, ou se um deve ser usado, quantos registros-índice podem ser usados na máquina?
5. Usando as instruções-índice na Seção 10.4, escrever um programa que adiciona 40 números localizados na memória, começando no endereço 200 e armazenando a soma no registro 300.
6. Modificar o programa na Seção 10.4 de modo que os números localizados nos endereços da memória 353 a 546 sejam somados e armazenados no endereço 600.
7. Modificar o programa na Seção 10.4 de modo que os números localizados nos endereços 300 a 305 sejam multiplicados e o produto armazenado no endereço 310.

8. O uso de paginação permite a relocalização ou movimentação ao redor de programas na memória sem modificação extensa dos endereços no programa. Explicar por quê.
9. No computador Varian na Seção 10.5, o uso de um modo 100 envolve uma forma de paginação chamada *endereçamento relativo*, onde o endereço real é formado adicionando-se o endereço no qual a instrução permanece ao número fornecido na porção de endereço da palavra de instrução. Se a palavra de instrução estiver no local 293 na memória, uma instrução de SOMAR for usada com $m = 100$ e a porção de endereço contiver 63 (decimal), que endereço é realmente usado na adição?
10. Comparar a organização de 4 acumuladores no Nova com a organização de 16 acumuladores na série 360 da IBM, observando quaisquer diferenças no uso dos dois endereços nas palavras da instrução.
11. Explicar como a paginação e o endereçamento indireto podem ser úteis na relocalização de subprogramas quando um programa é reescrito. Quais são algumas desvantagens da paginação e do endereçamento indireto?
12. Explicar a "confirmação de conexão" num barramento quando os dados forem transferidos de um transmissor para um receptor. Como pode este processo ser usado para evitar erros devidos ao desvio de sinal causado por sinais nos diferentes fios que chegam nos diferentes instantes de tempo (desviados) por causa das diferenças no comprimento da linha e das características e diferenças nos retardos através dos excitadores de linha de CI etc.
13. Descrever algumas vantagens e desvantagens da arquitetura de computador com acumulador múltiplo (registros de propósitos gerais) *versus* acumulador único. Incluir os efeitos sobre o comprimento da palavra de instrução, sobre a conveniência da programação, sobre o uso de registros-índice ou registros base etc.
14. O pequeno programa que se segue está em linguagem de montador para o PDP-8:

```
*700
        CLA   CLL
        TAD   DAT1
GO,     ISZ   DAT2
        JMP   GO
        RAL
        JMP   GO
DAT1,   0077
DAT2,   0
$
```

Após este programa ter sido montado e carregado, ele aparece como se segue na memória (todos os dígitos são octais), exceto que os conteúdos dos dois endereços na memória necessitam ser preenchidos. Fornecer os conteúdos dos locais (octal) 700 e 701 na memória.

NÚMERO DE ENDEREÇO	CONTEÚDOS
0700	
0701	
0702	2307
0703	5302
0704	7004
0705	5302
0706	0077
0707	0000

15. Avaliar o uso de registros gerais múltiplos e um formato de IBM 360 no endereçamento *versus* o uso de registros-índice no DDP-24 para adicionar números arranjados em três tabelas.
16. Explicar como a paginação, que é uma técnica de endereçamento, pode ser útil na relocalização de programas e explicar depois como pequenas páginas podem, às vezes, forçar os programadores a "pensar em segmentos". As características acima são dese-

jáveis ou indesejáveis?

17. Na interface de instrumentação padronizada, desenhar os sinais DAV, NRFD, NDAC para uma transferência de dados de um locutor a um ouvinte. Admitir que não haja problemas na transferência de dados e indicar quem está elevando e abaixando cada sinal.
18. Para a interface de instrumentação padronizada indicar como o controlador seleciona um "locutor" e um "ouvinte".
19. Como é manipulado o "desvio de sinal" na interface de instrumentacão padronizada?
20. Os detalhes das instruções do Nova são fornecidos na Fig. 10.30. Dada a configuração inicial

AC1	0	100	011	010	001	000
AC2	0	000	000	000	000	001

fornecer os conteúdos dos dois registros (AC1, AC2) e o bit de transporte depois de as seguintes instruções serem executadas. Descrever o que acontece, e onde, durante a execução das primeiras *duas* instruções.

	ACS	ACD	FCTN	SH	C'Y		SKIP
1	01	10	000	00	01	0	000
1	10	01	110	01	00	1	010
1	10	10	011	00	11	0	000
1	10	01	110	10	00	0	011
1	01	01	011	00	11	0	000

21. Explicar o endereçamento de páginas na série PDP-8 e a técnica de endereçamento de índice-mais-deslocamento usada pelos computadores Nova da DGC. Apresentar razões pelas quais você pensa que os projetistas (arquitetos) dos sistemas escolheram este sistema.
22. A seguir, temos uma seção de programa e conteúdo da memória de uma listagem de montador para um PDP-8. O programador que escreveu isto argumenta que, após a instrução no local 255_8 ser executada, o local 256_8 na memória conterá a diferença $A - B$ dos números A e B nos locais 260_8 e 2053_8. O programador está certo? Justificar sua resposta, explicando a operação do programa.

```
                *250
0250  7300      CLA  CLL
0251  1657      TAD  I    F
0252  7040      CMA
0253  1260      TAD  A
0254  3256      DCA  C
0255  7402      HLT
0256  0004      C,   Y
0257  2053      F,   2053
0260  0005      A,   5
                *2053
2053  0006      B,   6
```

Endereços da memória — Conteúdo da memória — Programa escrito em linguagem de montador

23. Escrever a seqüência de código binário, em código de máquina Nova, para comparação de dois números, um no segundo acumulador e um outro na palavra que começa no endereço especificado no terceiro acumulador. O programa é para saltar a próxima

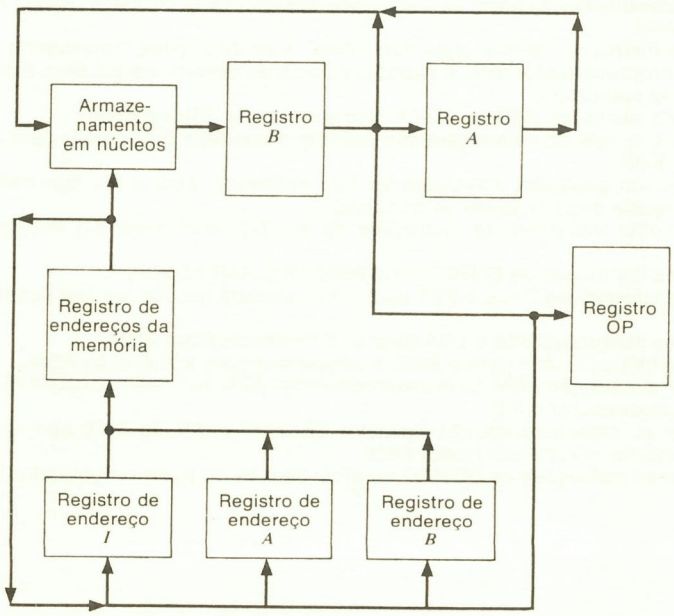
palavra de instrução se o conteúdo do acumulador (segundo) for maior do que o do local da memória.

24. Esboçar, descrever e discutir os méritos de uma das arquiteturas de máquina que foram apresentadas, ou de qualquer outra máquina com a qual esteja familiarizado (ou com a qual gostaria de estar familiarizado — incluindo qualquer um de seus próprios projetos "ideais", se tiver quaisquer de tais noções).
25. Um sistema de tempo real para controle de fabricação deve ser construído usando um computador. O sistema deve realizar duas funções:
    (a) Testar câmeras automaticamente quando elas forem fabricadas. Isto envolve, entre outras coisas, a leitura de 1.000 valores em cada segundo de diversos conversores A/D e a verificação ou teste para comprovar se os valores estão dentro dos limites prescritos.
    (b) O computador deve servir a quatro terminais que passam consultas contra a base de dados mantida nas câmaras, e também o computador deve passar alguns programas em Fortran e em Cobol.

    O projetista decide comprar um computador com um só barramento que é usado para interligar todos os conversores A/D com o computador bem como aos terminais, às unidades de disco, às memórias de núcleo e de CI etc. Discutir os méritos e as desvantagens deste uso de um só barramento em tal sistema.
26. No 200 da Minneapolis-Honeywell, quantos dígitos binários são necessários para armazenar os seguintes números decimais quando é usado o formato alfanumérico?

    (a) 1924 *(b)* 22 *(c)* 87 *(d)* 10.924

27. O diagrama-bloco que segue é dos registros de controle e aritmético do 200 da Honeywell. O registro de endereço *I* contém o endereço do caráter de instrução seguinte a ser usado pelo programa armazenado. O número neste registro é aumentado cada vez que um caráter é lido. Os registros de endereço *A* e *B* contêm os endereços dos dois caracteres alfanuméricos dos operandos que normalmente são usados. Estes registros são diminuídos de 1 cada vez que um caráter é usado. O registro de endereços da memória endereça a memória de núcleos; os registros *A* e *B* são registros de 9 bits que contêm os caracteres lidos da memória com *B*, recebendo sempre o que é lido diretamente da memória. O registro OP também tem 8 bits e é projetado para reter o caráter de código OP de cada palavra de instrução. Discutir a seqüência da operação destes registros durante a instrução de SUBTRAIR descrita na Seção 10.11.



28. Usando o mesmo diagrama-bloco da instrução precedente, descrever a seqüência destes registros no 220 da Minneapolis-Honeywell durante a instrução de adição na Seção 10.11.
29. No IBM 360, quantos dígitos binários são necessários para armazenar os seguintes números decimais em BCD e em binário. (Em BCD, o sinal requer 4 bits, e os números BCD são sempre organizados em bytes.)

    (a) 20 *(b)* 10.241 *(c)* 984 *(d)* 6

30. Descrever o cálculo de um endereço efetivo para o IBM 360 e esboçar o *hardware* da lógica que implementará este cálculo (sinais de controle, trajetos de dados, registros etc.).
31. Explicar o endereçamento de páginas na série PDP-8 e a técnica usada pelos computadores Nova da DGC. Discutir as vantagens e as desvantagens destas técnicas, umas em relação às outras.
32. Admitir que empilhamos *A*, *B*, *D* e *F* numa pilha, desempilhamos duas vezes e depois empilhamos *M* e em seguida *N*. Se depois desempilharmos três vezes as letras, que letras e em que ordem serão desempilhadas?
33. Explicar como os dispositivos periféricos interrompem um computador com organização de barramento único.
34. Explicar o significado de *acesso direto de memória* (DMA) e por que ele é desejável em alguns casos.
35. É possível imaginar quaisquer problemas que possam surgir nos sistemas de multiprocessadores?
36. Comparar uma organização de computadores de canalização com uma organização de múltiplas unidades. Onde surgem problemas quando são executadas, nestes computadores, as instruções de salto ou desvio?
37. Discutir a atribuição de locais no barramento de um microcomputador com duas RAM de 4K, uma ROM de 4K e um teletipo.
38. Explicar endereçamento estendido ou ampliado, endereçamento implícito ou subentendido e endereçamento relativo no microprocessador 6800.
39. Explicar como trabalha a instrução "desviar se zero" no microprocessador 6800.
40. Explicar a diferença entre uma instrução SOMAR acumulador (ABA) e uma instrução SOMAR (ADDA ou ADDB) no microprocessador 6800.
41. Explicar uma instrução de GIRAR À DIREITA no 6800.
42. Quais são as diferenças entre as instruções GIRAR e DESLOCAR no 6800? Por que são ambas úteis?
43. Explicar a instrução "desviar para sub-rotina" e as instruções "retornar de sub-rotina" para o microprocessador 6800 e explicar como elas seriam usadas para entrar e depois sair de uma sub-rotina.
44. Comparar a instrução JMS no PDP-8 com a instrução JSR no 6800.
45. Explicar os modos de endereçamento indireto e imediato do registro para o microprocessador 8080.
46. Explicar como duas das instruções do tipo aritmético para o microprocessador 8080 afetam o ajuste das bandeiras de condição.
47. Explicar como trabalham as instruções de SALTO condicional no microprocessador 8080.
48. Explicar as instruções de EMPILHAR e DESEMPILHAR no 8080.
49. Explicar as instruções CALL e RET no 8080 e compará-las com as instruções JSR e RTS do 6800.
50. Explicar as instruções STA e LDA para o microprocessador 8080.
51. Explicar a instrução ADI para o 8080 e compará-la com a instrução ADM.
52. Comparar a instrução ADM no microprocessador 8080 com as instruções ADDA e ADDB no microprocessador 6800.
53. Comparar as instruções de EMPILHAR e DESEMPILHAR no 8080 com as instruções DOES e INS no microprocessador 6800.
54. Comparar as instruções de DESVIO no 6800 com as do microprocessador 8080.

BIBLIOGRAFIA

CAP. 1

Allan, R.: Microprocessors Galore, *IEEE Spectrum*, vol. 13, no. 1, January 1976.

Barron, D. W.: *Assemblers and Loaders*, MacDonald American Elsevier, New York, 1975.

Bartee, T. C.: *Introduction to Computer Science*, McGraw-Hill Book Company, New York, 1975.

Bell, G., and A. Newell: *Computer Structures: Readings and Examples*, McGraw-Hill Book Company, New York, 1971.

Chapin, N.: *360/370 Programming in Assembly Language*, McGraw-Hill Book Company, New York, 1973.

Cole, R. W.: *Introduction to Computing*, McGraw-Hill Book Company, New York, 1972.

Colin, A. J. T.: *Introduction to Operating Systems*, MacDonald American Elsevier, New York, 1972.

Donovan, J. J.: *Systems Programming*, McGraw-Hill Book Company, New York, 1972.

Estes, J. W., and B. R. Ellis: *Elements of Computer Science*, Canfield Press, 1972.

Gear, C. W.: *Computer Organization and Programming*, McGraw-Hill Book Company, New York, 1974.

Gschwind, H. W., and E. J. McCluskey: *Design of Digital Computers*, Springer-Verlag, New York, 1975.

Hellerman, H.: *Digital Computer System Principles*, McGraw-Hill Book Company, New York, 1973.

Maurer, W. D.: *Programming*, Holden-Day, Inc., Publisher, San Francisco, 1970.

Peatman, J. B.: *The Design of Digital Systems*, McGraw-Hill Book Company, New York, 1972.

Ralston, A.: *Introduction to Programming and Computer Science*, McGraw-Hill Book Company, New York, 1972.

Rice, J. K., and R. R. Rice: *Introduction to Computer Science*, Holt, Rinehart and Winston, Inc., New York, 1969.

Souček, B.: *Minicomputers in Data Processing and Simulation*, John Wiley & Sons, Inc., New York, 1972.

Stark, R. H., and D. W. Dearhold: *Computer Concepts and Assembler Programming: 360/370 Systems*, Academic Press, New York, 1975.

Stone, H. S.: *Introduction to Computer Science and Data Structures*, McGraw-Hill Book Company, New York, 1972.

Watson, R. W.: *Timesharing System Design Concepts*, McGraw-Hill Book Company, New York, 1970.

Weiss, E.: *The PL/1 Converter*, McGraw-Hill Book Company, New York, 1969.

CAP. 2

Booth, T. L., and Y. T. Chien: *Computing*, Hamilton, Santa Barbara, Calif., 1974.

Estes, J. W., and B. R. Ellis: *Elements of Computer Science*, Canfield Press, 1972.

Gschwind, H. W., and E. J. McCluskey: *Design of Digital Computers*, Springer-Verlag, New York, 1975.
Hellerman, H.: *Digital Computer System Principles*, McGraw-Hill Book Company, New York, 1973.
O'Mally, J.: *Introduction to the Digital Computer*, Holt, Rinehart and Winston, Inc., New York, 1971.
Scott, N. R.: *Electronic Computer Technology*, McGraw-Hill Book Company, New York, 1970.

CAP. 3

Bartee, T. C., I. L. Lebow, and I. S. Reed: *Theory and Design of Digital Machines*, McGraw-Hill Book Company, New York, 1962.
Birkhoff, G., and T. C. Bartee: *Modern Applied Algebra*, McGraw-Hill Book Company, New York, 1970.
Boole, G.: *An Investigation of the Laws of Thought*, Dover Publications, Inc., New York, 1966. First published 1854.
Brzozowski, J. A., and M. Yoeli: *Digital Networks*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975.
Harrison, M. A.: *Introduction to Switching and Automata Theory*, McGraw-Hill Book Company, New York, 1965.
McCluskey, E. J., and T. C. Bartee: *A Survey of Switching Circuit Theory*, McGraw-Hill Book Company, New York, 1962.
McCluskey, E. J., Jr.: *Introduction to the Theory of Switching Circuits*, McGraw-Hill Book Company, New York, 1965.
Oberman, R. M. M.: *Discipline in Combinational and Sequential Circuit Designs*, McGraw-Hill Book Company, New York, 1970.
Wood, P. E.: *Switching Theory*, McGraw-Hill Book Company, New York, 1968.

CAP. 4

American Microsystems, Inc.: *Guide to Standard MOS Products*, Santa Clara, Calif., 1975.
Birkhoff, G., and T. C. Bartee: *Modern Applied Algebra*, McGraw-Hill Book Company, New York, 1970.
Brzozowski, J. A., and M. Yoeli: *Digital Networks*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975.
Fairchild Semiconductor: *TTL Data Book*, Mountain View, Calif., 1976.
Gschwind, H. W., and E. J. McCluskey: *Design of Digital Computers*, Springer-Verlag, New York, 1975.
Hodges, D. A.: Trends in Computer Hardware Technology, *Computer Design*, vol. 15, no. 2, February 1976.
Intel Corporation: *Intel 8080 Microcomputer System User's Manual*, Santa Clara, Calif., 1975.
Motorola Corporation: *M6800 Microprocessor Applications Manual*, Phoenix, Ariz., 1975.
Rottner, J., J. Cornet, and M. E. Hoff: Bipolar LSI Computing Elements Usher in New Era of Digital Design, *Electronics*, September 1976.
Texas Instruments, Inc.: *The TTL Data Book*, Dallas, Tex., 1976.

CAP. 5

American Microsystems, Inc.: *Guide to Standard MOS Products*, Santa Clara, Calif., 1975.
Deem, W., K. Muchow, and A. Zeppa: *Digital Computers, Circuits and Concepts*, Reston

Publishing Company, Inc., Reston, Va., 1974.
Fairchild Semiconductor: *TTL Data Book*, Mountain View, Calif., 1976.
Hodges, D. A., Trends in Computer Hardware Technology, *Computer Design*, vol. 15, no. 2, February 1976.
Intel Corporation: *Intel 8080 Microcomputer System User's Manual*, Santa Clara, Calif., 1975.
Malvino, A. P., and D. P. Leach: *Digital Principles and Applications*, McGraw-Hill Book Company, New York, 1974.
Motorola Corporation: *M6800 Microprocessor Applications Manual*, Phoenix, Ariz., 1975.
Motorola Semiconductor Products, Inc.: Semiconductor Data Library, Phoenix, Ariz., 1975.
Rottner, J., J. Cornet, and M. E. Hoff: Bipolar LSI Computing Elements Usher in New Era of Digital Design, *Electronics*, September 1976.
Texas Instruments, Inc.: *The TTL Data Book*, Dallas, Tex., 1976.

CAP. 6

Bartee, T. C., I. L. Lebow, and I. S. Reed: *Theory and Design of Digital Machines*, McGraw-Hill Book Company, New York, 1962.
Flores, I.: *The Logic of Computer Arithmetic*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1963.
Foster, C. C.: *Computer Architecture*, D. Van Nostrand Company, Inc., Princeton, N.J., 1971.
Freiman, C. V.: Statistical Analysis of Certain Binary Division Algorithms, *Proceedings of the IRE*, vol 49, no. 1, January 1961.
Gear, C. W.: *Computer Organization and Programming*, McGraw-Hill Book Company, New York, 1974.
MacSorley, O. L.: High Speed Arithmetic in Binary Computers, *Proceedings of the IRE*, vol. 49, no. 1, January 1961.
Rice, J. K., and R. R. Rice: *Introduction to Computer Science*, Holt, Rinehart and Winston, Inc., New York, 1969.
Stone, H. S.: *Introduction to Computer Science and Data Structures*, McGraw-Hill Book Company, New York, 1972.

CAP. 7

American Microsystems, Inc.: *Guide to Standard MOS Products*, Santa Clara, Calif., 1975.
Carothers, J. D.: A New High Density Recording System: The IBM 1311 Disc Storage Drive with Interchangeable Disc Packs, *Proceedings of the 1963 Fall Joint Computer Conference*, American Federation of Information Processing Societies.
Cole, R. W.: *Introduction to Computing*, McGraw-Hill Book Company, New York, 1972.
Davis, S.: Selection and Application of Semiconductor Memories, *Computer Design*, January 1976.
Deem, W., K. Muchow, and A. Zeppa: *Digital Computers Circuits and Concepts*, Reston Publishing Company, Inc., Reston, Va., 1974.
Intel Corporation: *Intel 8080 Microcomputer System User's Manual*, Santa Clara, Calif., 1975.
Motorola Corporation: *M6800 Microprocessor Applications Manual*, Phoenix, Ariz., 1975.
Motorola Semiconductor Products, Inc.: Semiconductor Data Library, Phoenix, Ariz., 1975.

Postoriza, J.: Tradeoffs among Binary Codes in Magnetic Tape Cassettes, *Computer Design*, vol. 15, no. 1, 1976.
Rottner, J., J. Cornet, and M. E. Hoff: Bipolar LSI Computing Elements Usher in New Era of Digital Design, *Electronics*, September 1976.
Smith, C. H., and L. D. Wittie: Memory Hardware for High Speed Job Selection, *IEEE Transactions on Computers*, vol. C-25, no. 2, February 1976.
Thomas, A. T.: Design Techniques for Microprocessor Memory Systems, *Computer Design*, August 1975.
Walther, T. R.: Dynamic N-MOS RAM with Simplified Refresh, *Computer Design*, February 1973.

CAP. 8

Anderson, T. O.: Code Conversion for Parallel A-D Converters, *Computer Design*, July 1973.
Booth, T. L., and Y. T. Chien: *Computing*, Hamilton, Santa Barbara, Calif., 1974.
Davis, S.: Printer Selection Factors, *Computer Design*, December 1972.
Deem, W., K. Muchow, and A. Zeppa: *Digital Computers Circuits and Concepts*, Reston Publishing Company, Inc., Reston, Va., 1974.
Foster, C. C.: *Computer Architecture*, D. Van Nostrand Company, Inc., Princeton, N.J., 1977.
Hodges, D. A.: Trends in Computer Hardware Technology, *Computer Design*, vol. 15, no. 2, February 1976.
O'Mally, J.: *Introduction to the Digital Computer*, Holt, Rinehart and Winston, New York, 1971.
Postoriza, J.: Tradeoffs among Binary Codes in Magnetic Tape Cassettes, *Computer Design*, vol. 15, no. 1, 1976.
Scott, N. R.: *Electronic Computer Technology*, McGraw-Hill Book Company, New York, 1970.
Talambiras, P. P.: Digital-to-Analog Converters: Some Problems in Producing High-fidelity Systems, *Computer Design*, vol. 15, no. 1, January 1975.

CAP. 9

Bartee, T. C.: *Introduction to Computer Science*, McGraw-Hill Book Company, New York, 1975.
Bell, G., and A. Newell: *Computer Structures: Readings and Examples*, McGraw-Hill Book Company, New York, 1971.
Booth, T. L., and Y. T. Chien: *Computing*, Hamilton, Santa Barbara, Calif., 1974.
Boulage, G., and J. Mermet (eds.): *Microprogramming*, Hermann & Cie, Paris, 1972.
Estes, J. W., and B. R. Ellis: *Elements of Computer Science*, Canfield Press, 1972.
Fagg, P., et al.: IBM System 360 Engineering, *Proceedings of the 1964 Fall Joint Computer Conference*, American Federation of Information Processing Societies.
Foster, C. C.: *Computer Architecture*, D. Van Nostrand Company, Inc., Princeton, N.J., 1971.
Hellerman, H.: *Digital Computer System Principles*, McGraw-Hill Book Company, New York, 1973.
Logan, J. D., and P. S. Kreager: Using a Microprocessor, *Computer Design*, September 1975.
Marchin, P.: Multi-level Nesting of Subroutines in a One-Level Microprocessor, *Computer Design*, vol. 15, no. 2, February 1976.
Peatman, J. B.: *The Design of Digital Systems*, McGraw-Hill Book Company, New York, 1972.

Ralston, A.: *Introduction to Programming and Computer Science*, McGraw-Hill Book Company, New York, 1972.
Souček, B.: *Minicomputers in Data Processing and Simulation*, John Wiley & Sons, Inc., New York, 1972.
Stone, H. S.: *Introduction to Computer Science and Data Structures*, McGraw-Hill Book Company, New York, 1972.
Waldecker, D. E.: Comparison of a Microprogrammed and a Non-microprogrammed Computer, *Computer Design*, June 1970.

CAP. 10

Allan, R.: Microprocessors Galore, *IEEE Spectrum*, vol. 13, no. 1, January 1976.
Anderson, H. A., M. Reiser, and G. L. Galati: Tuning a Virtual Storage System, *IBM Systems Journal*, vol. 14, no. 3, 1975.
Ashenhurst, R. L., and R. H. Vonderohe: A Hierarchical Network (of Minicomputers), *Datamation*, vol. 21, no. 2, February 1975.
Barron, D. W.: "Assemblers and Loaders," MacDonald American Elsevier, New York, 1975.
Bowra, J. W., and H. C. Torng: The Modeling and Design of Multiple Function Unit Processors, *IEEE Transactions on Computers*, vol. C-25, no. 3, March 1976.
Chapin, N.: *360/370 Programming in Assembly Language*, McGraw-Hill Book Company, New York, 1973.
Colin, A. J. T.: *Introduction to Operating Systems*, MacDonald American Elsevier, New York, 1972.
Cowry, F. F.: Advanced Architecture and Applications of Microcomputers, *Computer*, vol. 9, no. 1, January 1976.
Data General Corporation: *How to Use the Nova Computers*, Southboro, Mass., 1975.
Digital Equipment Corporation: *Introduction to Programming*, Maynard, Mass., 1972.
Digital Equipment Corporation: *PDP 11/45 Processor Handbook*, 1976.
Donovan, J. J.: *Systems Programming*, McGraw-Hill Book Company, New York, 1972.
Estes, J. W., and B. R. Ellis: *Elements of Computer Science*, Canfield Press, 1972.
Farber, D. J.: A Ring Network, *Datamation*, vol. 21, no. 2, February 1975.
Fleisher, H., and L. I. Maissel: An Introduction to Array Logic, *IBM Journal of Research and Development*, vol. 19, no. 2, March 1975.
Foster, C. C.: *Computer Architecture*, D. Van Nostrand Company, Inc., Princeton, N.J., 1971.
Gear, C. W.: *Computer Organization and Programming*, McGraw-Hill Book Company, New York, 1974.
Gschwind, H. W., and E. J. McCluskey: *Design of Digital Computers*, Springer-Verlag, New York, 1975.
Intel Corporation: *Intel 8080 Microcomputer System User's Manual*, Santa Clara, Calif., 1975.
Kinoshito, K., T. Sasao, and J. Matsuda: On Magnetic Bubble Logic Circuits, *IEEE Transactions on Computers*, vol. C-25, no. 3, March 1976.
Lowrie, D. H.: Access and Alignment of Data in an Array Processor, *IEEE Transactions on Computers*, vol. C-24, no. 12, December 1975.
Madnick, S., and J. Donovan: *Operating Systems*, McGraw-Hill Book Company, New York, 1973.
Motorola Corporation: *M6800 Microprocessor Applications Manual*, Phoenix, Ariz., 1975.
Salisbury, A. B., *Microprogrammable Computer Architecture*, MacDonald American Elsevier, New York, 1976.
Souček, B.: *Minicomputers in Data Processing and Simulation*, John Wiley & Sons, Inc., New York, 1972.

Stark, R. H., and D. W. Dearholt: *Computer Concepts and Assembler Programming: 360/370 Systems*, Academic Press, New York, 1975.

Thomas, A. T.: Design Techniques for Microprocessor Memory Systems, *Computer Design*, August 1975.

Ward, A. R.: LSI Microprocessors and Microcomputers: A Bibliography, *Computer*, vol. 9, no. 1, January 1976.

Weissberger, A. J.: Microprocessors Expand Industry Applications of Data Acquisitions, *Electronics*, September 1974.

RESPOSTAS AOS EXERCÍCIOS SELECIONADOS COM NUMERAÇÃO ÍMPAR

CAPÍTULO 1

5.

Endereço	Código de operação	Parte de endereço
1	CLA	40
2	ADD	41
3	ADD	42
4	STO	43
5	HLT	000

40 contém X
41 contém Y
42 contém Z
43 contém O

7.

Endereço	Operação	Operando
1	CLA	20
2	MUL	20
3	STO	40
4	CLA	21
5	MUL	21
6	ADD	40
7	STO	40
8	CLA	22
9	MUL	22
10	ADD	40
11	STO	40
12	HLT	
20	contém	X
21	contém	Y
22	contém	Z
40	contém	0

11.

Endereço	Operação	Operando
1	CLA	20
2	STO	21
3	CLA	21
4	MUL	20
5	STO	21
6	CLA	50
7	ADD	51
8	STO	50
9	BRM	3
10	CLA	21
11	ADD	20
12	STO	21
13	HLT	
20	contém	X
21	contém	0
50	contém	−5
51	contém	1

Armazenaremos nossa soma $X^5 + X$ no endereço atribuído pelo montador à variável *D*.

Endereço	Operação	Operando
A	DEC	0
B	DEC	−5
C	DEC	1
D	DEC	0
	CLA	X
	STO	D
E	CLA	D
	MUL	X
	STO	D
	CLA	B
	ADD	C
	STO	B
	BRM	E
	CLA	D
	ADD	X
	STO	D
	HLT	

17.

Se admitirmos somente que os inteiros *estão* em ordem crescente ou decrescente:

Endereço	Operação	Operando
1	CLA	30
2	SUB	31
3	BRM	5
4	HLT	
5	HLT	

O computador pára no endereço 4 se os números estiverem em ordem crescente e, no 5, se os números estiverem em ordem decrescente.

Devemos verificar se os números estão em ordem crescente ou decrescente.

Endereço	Operação	Operando
1	CLA	30
2	SUB	31
3	BRM	10
4	CLA	31
5	SUB	32
6	BRM	13
7	HLT	
10	CLA	31
11	SUB	32
12	BRM	15
13	HLT	
15	HLT	

Este programa pára no endereço 13 se os números não estiverem nem em ordem crescente nem decrescente; no endereço 7, se os números estiverem em ordem crescente; e, no endereço 15, se os números estiverem em ordem decrescente.

21.

Endereço	Operação	Operando
1	CLA	26
2	SUB	25
3	BRM	300
4	BRA	400

23.

Endereço	Operação	Operando
P	CLA	A
	SUB	B
	BRM	M
	CLA	A
	SUB	C
	BRM	M
	CLA	A
	STO	X
	HLT	
M	CLA	B
	SUB	C
	BRM	N
	CLA	B
	STO	X
	HLT	
N	CLA	C
	STO	X
	HLT	

25.
```
A = 10;
B = 20;
C = 5;
D = (A * B) ** C;
END;
```

27. 72

29.
```
Z = 0
A = 0
LOOP. DO WHILE Z LE A
X = -X
Y = -Y
Z = X - Y
END LOOP
PRINT (Z)
END
```

31.

Endereço	Operação	Operando	Endereço	Operação	Operando
1	CLA	300	13	CLA	6
2	BRM	5	14	ADD	400
3	STO	300	15	STO	6
4	BRA	7	16	CLA	401
5	SUB	300	17	ADD	400
6	SUB	300	18	STO	401
7	CLA	3	19	BRM	1
8	ADD	400	20	HLT	
9	STO	3	300 }	contém números	
10	CLA	5	329 }		
11	ADD	400	400	contém 1	
12	STO	5	401	contém −30	

33. *(a)* 45 *(b)* 2000 *(c)* 55 *(d)* 70

35. *(a)* A ** C / B ** D *(b)* A ** 2 + B ** 2 + C * D *(c)* A/B + (C/D) ** E *(d)* (A * B) ** C + (B * D)/E

CAPÍTULO 2

1. *(a)* 100111 *(b)* 111011 *(c)* 1000000000 *(d)* 0,011100
(e) 0,11001 *(f)* 0,110011 *(g)* 100.000.000,11
(h) 10.000.001,1001 *(i)* 1.000.000.000.000,11101

3. *(a)* 101011 *(b)* 1000000 *(c)* 100000000000 *(d)* 0,011
(e) 0,11011 *(f)* 0,0111 *(g)* 1000000000,1 *(h)* 10000011,1001
(i) 10000000000,0001

5. *(a)* 13 *(b)* 27 *(c)* 23 *(d)* 0,6875 *(e)* 0,203125 *(f)* 0,21288065
(g) 59,6875 *(h)* 91,203125 *(i)* 22,3408203125

7. *(a)* 11 *(b)* 36 *(c)* 19 *(d)* 0,8125 *(e)* 0,5625 *(f)* 0,3125
(g) 11,1875 *(h)* 9,5625 *(i)* 5,375

9. *(a)* 10100,11 = 20,75 *(b)* 1001010 = 74 *(c)* 1,1 = 1,5
(d) 10101 = 21

11. *(a)*

1101,1	13,5
+ 1011,1	+ 11,5
11001,0	25

(b)

101101	45
+ 110110	+ 54
1100011	99

(c)

0,0011	0,1875
+ 0,1110	+ 0,875
1,0001	1,0625

(d)

1100,011	12,375
+ 1011,011	+ 11,375
10111,110	23,750

13. *(a)*

```
  1000000
- 100000
  100000
```

(b)

```
  1111111
- 111111
  1000000
```

(c)

```
  1011101,1
- 101010,11
  110010,11
```

(d)

```
  1010100,01001
- 110000,01010
  100011,11111
```

15. *(a)*

```
  100101
- 100011
  000010
```

(b)

```
  100000000
  010000000
  010000000
```

(c)

```
  1011110,1
+ 101011,11
  110010,11
```

(d)

```
  111111111
+ 111111111
  100000000
```

17. *(a)* 100100000 *(b)* 11111100 *(c)* 100 *(d)* 101
(e) 10100000001,101 *(f)* 10,1

19. *(a)*

```
    1111
    1101
    1111
   11110
  1111
 11000011
```

(b)

```
    1111
    1010
   11110
  11110
 100101100
```

(c)

```
    101100
      1011
    101100
   101100
  1011000
 111100100
```

(d)

```
    101010
      1100
   101010
  101010
 1111111000
```

(e)

```
     111,11
       10,1
  ---------
     11111
   111110
  ---------
  10011,011
```

(f)

```
      10110,1
        100,11
  ----------
       101101
      101101
   10110100
  ----------
  1101010,111
```

21.

	COMPLEMENTO DE 9	COMPLEMENTO DE 10
(a)	4563	4564
(b)	8067	8068
(c)	54,84	54,85
(d)	81,706	81,707

23.

	COMPLEMENTO DE 9	COMPLEMENTO DE 10
(a)	6345	6346
(b)	7877	7878
(c)	45,80	45,81
(d)	62,736	62,737

25.

	COMPLEMENTO DE 1	COMPLEMENTO DE 2
(a)	0100	0101
(b)	00100	00101
(c)	0100,10	0100,11
(d)	00100,10	00100,11

27.

	COMPLEMENTO DE 1	COMPLEMENTO DE 2
(a)	01000	01001
(b)	011011	011100
(c)	01000,01	01000,10
(d)	01100,00	01100,01

29.

(a)

```
     948
     765
  1  713
  └──→ 1
     714
```

```
  948
  766
  714
```

(b)

```
     347
     736
  1  083
  └──→ 1
     084
```

```
  347
  737
  084
```

(c)

```
     349,5
     754,6
  1  104,1
  └──→ 1
     104,2
```

```
  349,5
  754,7
  104,2
```

(d)

```
     412,7
     590,7
  1  003,4
  └──→ 1
       3,5
```

```
  412,7
  590,8
    3,5
```

31.

	COMPLEMENTO DE 9	COMPLEMENTO DE 10
(a)	1024 9086 0110 → 1 111	1024 9087 0111
(b)	249 862 111 → 1 112	249 863 112
(c)	24,1 86,5 10,6 → 1 10,7	24,1 86,6 10,7
(d)	239,3 880,5 119,8 → 1 119,9	239,3 880,6 119,9

33.

	COMPLEMENTO DE 1	COMPLEMENTO DE 2
(a)	1011 1010 1 0101 → 1 0110	1011 1011 0110
(b)	11011 00110 1 00001 → 1 10	11011 00111 00010
(c)	10111,1 01100,0 1 00011,1 → 1 100,0	10111,1 01100,1 100,0
(d)	11011,00 01100,00 1 00111,00 → 1 111,01	11011,00 01100,01 111,01

35. $2^6 = 64$

37. 1.000 números diferentes em cada caso (de 0 a 999, por exemplo)

39. 0, 1, 2, 3, 10, 11, 12, 13, 20, 21

41. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 20, 21, 22

43. *(a)*
$$\begin{array}{r} 0{,}1001 \\ 0{,}1001 \\ \hline 0010 \\ \hline \rightarrow 1 \\ \hline 0{,}0011 \end{array}$$
(b)
$$\begin{array}{r} 0{,}1110 \\ 0{,}1001 \\ \hline 0111 \\ \hline \rightarrow 1 \\ \hline 0{,}1000 \end{array}$$
(c)
$$\begin{array}{r} 01111 \\ 10110 \\ \hline 00101 \\ \hline \rightarrow 1 \\ \hline 0{,}00110 \end{array}$$
(d)
$$\begin{array}{r} 11011 \\ 00110 \\ \hline 00001 \\ \hline \rightarrow 1 \\ \hline 00010 \end{array}$$
(e)
$$\begin{array}{r} 1110101 \\ 0101101 \\ \hline 1\ 0100010 \\ \hline \rightarrow 1 \\ \hline 0100011 \end{array}$$

47. *(a)*
$$\begin{array}{r} 45056 \\ 1536 \\ 192 \\ 7 \\ \hline 46791 \end{array}$$
(b)
$$\begin{array}{r} 24576 \\ 1024 \\ 160 \\ 12 \\ \hline 25772 \end{array}$$
(c)
$$\begin{array}{r} 40960 \\ 1024 \\ 144 \\ 2 \\ \hline 42130 \end{array}$$
(d)
$$\begin{array}{r} 851968 \\ 8192 \\ 1792 \\ 96 \\ 3 \\ \hline 862051 \end{array}$$

51. *(a)* 55 *(b)* 556 *(c)* 267 *(d)* 66,3 *(e)* 3,554

55. *(a)* 1644,0 *(b)* 514 *(c)* 1041,3 *(d)* 1170.76051,0 *(e)* 10515,5

59. *(a)* B7 *(b)* 9C *(c)* 5F *(d)* 0,7E *(e)* B7A

63. *(a)*
$$\begin{array}{r} {\scriptstyle 1}\ \ \\ 15_8 \\ +14_8 \\ \hline 31_8 \end{array}$$
(b)
$$\begin{array}{r} {\scriptstyle 1}\ \ \\ 24_8 \\ +36_8 \\ \hline 62_8 \end{array}$$
(c)
$$\begin{array}{r} {\scriptstyle 1}\ \ \\ 126_8 \\ 347_8 \\ \hline 475_8 \end{array}$$
(d)
$$\begin{array}{r} {\scriptstyle 1}\ \ \\ 67_8 \\ 45_8 \\ \hline 134_8 \end{array}$$
(e)
$$\begin{array}{r} {\scriptstyle 1}\ \ \\ 136_8 \\ 636_8 \\ \hline 774_8 \end{array}$$

CAPÍTULO 3

1. *(a)*

X	Y	$\overline{Y}$	XY	$X\overline{Y}$	$XY + X\overline{Y}$
0	0	1	0	0	0
0	1	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1

(b)

X	Y	Z	$\overline{Y}$	$\overline{Z}$	$X\overline{Y}Z$	$\overline{YZ}$	$X\overline{Y}Z + \overline{YZ}$
0	0	0	1	1	0	1	1
0	0	1	1	0	0	0	0
0	1	0	0	1	0	0	0
0	1	1	0	0	0	0	0
1	0	0	1	1	0	1	1
1	0	1	1	0	1	0	1
1	1	0	0	1	0	0	0
1	1	1	0	0	0	0	0

3. (a)

X	Y	Z	XYZ	$X\overline{Y}\overline{Z}$	$XYZ + X\overline{Y}\overline{Z}$
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	1	0	1

(b)

A	B	C	ABC	$A\overline{B}\overline{C}$	$\overline{A}\overline{B}\overline{C}$	$ABC + A\overline{B}\overline{C} + \overline{A}\overline{B}\overline{C}$
0	0	0	0	0	1	1
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	1	0	1
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	1	0	0	1

(c)

A	B	C	$B\overline{C}$	$\overline{B}C$	$B\overline{C} + \overline{B}C$	$A(B\overline{C} + \overline{B}C)$
0	0	0	0	0	0	0
0	0	1	0	1	1	0
0	1	0	1	0	1	0
0	1	1	0	0	0	0
1	0	0	0	0	0	0
1	0	1	0	1	1	1
1	1	0	1	0	1	1
1	1	1	0	0	0	0

(d)

A	B	C	$A+B$	$A+C$	$\overline{A}+\overline{B}$	$(A+B)(A+C)(\overline{A}+\overline{B})$
0	0	0	0	0	1	0
0	0	1	0	1	1	0
0	1	0	1	0	1	0
0	1	1	1	1	1	1
1	0	0	1	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	0	0
1	1	1	1	1	0	0

5. Somente os valores das expressões são listados:

(a)

A	B	$A\overline{B}+\overline{A}B$
0	0	0
0	1	1
1	0	1
1	1	0

(b)

A	B	C	$A\overline{B}+B\overline{C}$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

(c)

A	C	$A\overline{C}+AC$
0	0	0
0	1	0
1	0	1
1	1	1

(d)

A	B	C	$A\overline{B}C+AB\overline{C}+\overline{A}BC$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

(e)

A	B	C	$A(A\overline{B}C+A\overline{B}\overline{C}+AB\overline{C})$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

7. *(a)* $\bar{B}\bar{C} + \bar{A}\bar{C} + \bar{A}\bar{B}$ *(b)* $\bar{C} + \bar{B} + \bar{A}$
(c) $A(B + C)$ *(d)* A é uma expressão mínima.

9. *(a)* $ABC(AB\bar{C} + A\bar{B}C + \bar{A}BC) = 0$, nenhuma designação de valores binários fará esta expressão assumir o valor 1.
(b) $AB + A\bar{B} + \bar{A}C + \bar{A}\bar{C} = 1$, cada designação de valores dará a esta expressão o valor 1.
(c) $XY + XYZ + XY\bar{Z} + \bar{X}YZ = Y$
(d) $XY(\bar{X}YZ + X\bar{Y}\bar{Z} + \bar{X}\bar{Y}\bar{Z}) = 0$

11. *(a)* $\bar{A}(\bar{B} + \bar{C})(\bar{A} + \bar{B}) = \bar{A}\bar{B} + \bar{A}\bar{C}$ ou $\bar{A}(\bar{B} + \bar{C})$
(b) $\bar{A}\bar{B} + \bar{B}\bar{C} + \bar{A}\bar{C}$
(c) $(\bar{A} + \bar{B})(B + \bar{C})(\bar{C} + D) = \bar{A}B\bar{C} + \bar{A}\bar{C} + \bar{A}BD + \bar{A}\bar{C}D =$
$\bar{A}\bar{C} + \bar{B}\bar{C} + \bar{A}BD$
(d) $(\bar{A} + \bar{B}) + (C + \bar{D})(B + \bar{C}) = \bar{A} + \bar{B} + BC + B\bar{D} + \bar{C}\bar{D} =$
$\bar{A}\bar{B} + C + \bar{D}$
(e) $\bar{A} + \bar{B}\bar{C} + CD$

13. As colunas importantes nestes quadros são como segue:

X	Y	Z	$\overline{(X+Y+Z)}$	$\bar{X}\bar{Y}\bar{Z}$
0	0	0	1	1
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

X	Y	Z	$\overline{(XYZ)}$	$\bar{X} + \bar{Y} + \bar{Z}$
0	0	0	1	1
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

15. *(a)* $\bar{A}BC + A\bar{B}C + AC + BC$
(b) $A\bar{B} + A\bar{C} + ABC + AB\bar{C} + \bar{A}BC + \bar{B}C$ (Isto pode ser simplificado.)
(c) $AB + ABC$

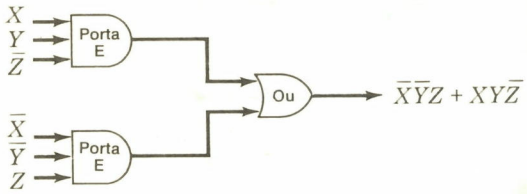
17. Regra 13

19. *(a)* $AB + AD + BC + CD$
(b) $AB + AD + AC + BC + CD + C + BD + D + DC = C + D + AB$

(c) $ABC + ABD + AC + BC + DC + ADC + BDC + DC = AC + BC + DC + ABD$

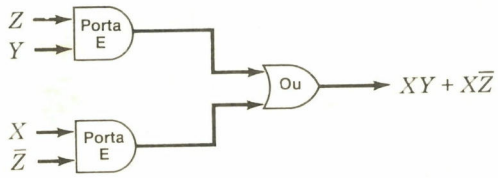
(d) $A\overline{B}\overline{C} + ABC$

21. $\overline{X}YZ + XY\overline{Z}$



23. $(X + Y)(X + Z) = X + XZ + YZ$
$= X + Z + YZ$
$= X + Z$

25. $XY + X\overline{Z}$



27.

X	Y	$X + \overline{X}Y$	$X + Y$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

29. $Y\overline{Z} + \overline{Y}Z$ é a expressão da soma-de-produtos e $(Y + Z)(\overline{Y} + \overline{Z})$ é a expressão do produto-de-somas.

35. (a)

	$\overline{X}$	X
$\overline{Y}$	1	
Y	1	

$\overline{X}\overline{Y} + \overline{X}Y$

(b)

	$\overline{X}\overline{Y}$	$\overline{X}Y$	XY	$X\overline{Y}$
$\overline{Z}$	1		1	
Z				

$\overline{X}\overline{Y}\overline{Z} + XY\overline{Z}$

(c)

	$\overline{X}\overline{Y}$	$\overline{X}Y$	XY	$X\overline{Y}$
$\overline{Z}$	1	1		
Z	1			

$\overline{X}Y\overline{Z} + \overline{X}\overline{Y}$

(d)

	$\bar{X}\bar{Y}$	$\bar{X}Y$	XY	$X\bar{Y}$
$\bar{Z}$	1	1		1
Z				

$\bar{X}\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z}$

(e)

	$\bar{X}\bar{Y}$	XY	XY	$X\bar{Y}$
$\bar{Z}$	1	1		1
Z				

$\bar{X}\bar{Z} + \bar{Y}\bar{Z}$

(f)

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
C				
C				

$AB(\bar{A}\bar{B}\bar{C} + \bar{B}C)$

39. (a)

	$\bar{X}$	X
$\bar{Y}$	1	1
Y		

$X\bar{Y} + \bar{X}\bar{Y}$

(b)

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}$	1			1
C				

$\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C}$

(c)

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}$				
C			1	1

$A\bar{B}C + ABC$

(d)

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}$			1	1
C			1	1

$ABC + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C}$

(e)

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
C				1
C			1	1

$ABC + A\bar{B}\bar{C} + A\bar{B}C$

(f)

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}$	1			
C		1	1	

$ABC + \bar{A}BC + \bar{A}\bar{B}\bar{C}$

41. *(a)*

	$\bar{X}\bar{Y}$	$\bar{X}Y$	XY	$X\bar{Y}$
$\bar{Z}$	1			
Z	1		1	1

$m_0 + m_1 + m_5 + m_7$

(b)

	$\bar{X}\bar{Y}$	$\bar{X}Y$	XY	$X\bar{Y}$
$\bar{Z}$				1
Z	1	1		1

$m_1 + m_3 + m_4 + m_5$

(c)

	$\bar{X}\bar{Y}$	$\bar{X}Y$	XY	$X\bar{Y}$
$\bar{Z}$		1		
Z	1	1		1

$m_1 + m_2 + m_3 + m_5$

(d)

	$\bar{X}\bar{Y}$	$\bar{X}Y$	XY	$X\bar{Y}$
$\bar{Z}$	1			
Z			1	1

$m_0 + m_5 + m_7$

45. *(b)*

	$\bar{W}\bar{X}$	$\bar{W}X$	WX	$W\bar{X}$
$\bar{Y}\bar{Z}$	1	1	1	1
$\bar{Y}Z$			1	1
YZ				1
$Y\bar{Z}$	1			1

$m_0 = \bar{W}\bar{X}\bar{Y}\bar{Z}$ $\quad m_{10} = W\bar{X}Y\bar{Z}$
$m_2 = \bar{W}\bar{X}Y\bar{Z}$ $\quad m_{11} = W\bar{X}YZ$
$m_4 = \bar{W}X\bar{Y}\bar{Z}$ $\quad m_{12} = WX\bar{Y}\bar{Z}$
$m_8 = W\bar{X}\bar{Y}\bar{Z}$ $\quad m_{13} = WX\bar{Y}Z$
$m_9 = W\bar{X}\bar{Y}Z$

$m_0 + m_2 + m_4 + m_8 + m_9 + m_{10} + m_{11} + m_{12} + m_{13}$
$\bar{X}\bar{Z} + \bar{Y}\bar{Z} + W\bar{X} + W\bar{Y}$
$\bar{Z}(\bar{X} + \bar{Y}) + W(\bar{X} + \bar{Y})$
$(\bar{Z} + W)(\bar{X} + \bar{Y})$

47. *(a)* $m_1 + m_3 + m_5 + m_7 + m_{12} + m_{13} + m_8 + m_9$

	WX	$W\bar{X}$	$\bar{W}\bar{X}$	$\bar{W}X$
YZ	0	0	m_3	m_7
$Y\bar{Z}$	0	0	0	0
$\bar{Y}\bar{Z}$	m_{12}	m_8	0	0
$\bar{Y}Z$	m_{13}	m_0	m_1	m_5

Produto-de-somas:

$\overline{(WY + W\bar{Z})} =$
$(\bar{W} + \bar{Y})(W + Z)$

Soma-de-produtos:

$W\bar{Y} + \bar{W}Z$

(b) $m_0 + m_5 + m_7 + m_8 + m_{11} + m_{13} + m_{15}$

	WX	$W\bar{X}$	$\bar{W}\bar{X}$	$\bar{W}X$
YZ	m_{15}	m_{11}		m_7
$Y\bar{Z}$	0			0
$\bar{Y}\bar{Z}$	0	m_8	m_0	0
$\bar{Y}Z$	m_{13}			m_5

Produto-de-somas:

$(X\bar{Z} + Y\bar{Z} + \bar{W}\bar{X}Z + \bar{X}\bar{Y}Z) =$
$(\bar{X} + Z)(\bar{Y} + Z)(W + X + \bar{Z})$
$(X + Y + \bar{Z})$

não-especificadas

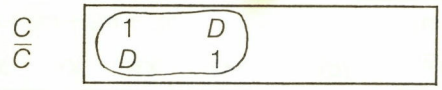
49. *(a)* $\bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + \overbrace{ABC + \bar{A}B\bar{C} + \bar{A}BC}^{} = B\bar{C}$

	AB	$A\bar{B}$	$\bar{A}\bar{B}$	$\bar{A}B$
C	D		D	
$\bar{C}$		1	1	D

não-especificadas

(b) $ABC + \bar{A}\bar{B}\bar{C} + AB\bar{C} + A\bar{B}C = A$

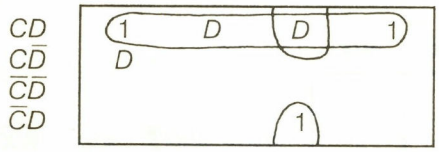
$AB \quad A\bar{B} \quad \bar{A}\bar{B} \quad \bar{A}B$



não-especificadas

(c) $ABCD + \bar{A}\bar{B}\bar{C}D + \bar{A}BCD + A\bar{B}CD + \bar{A}\bar{B}CD + AB\bar{C}\bar{D}$

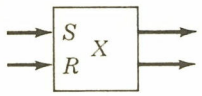
$AB \quad A\bar{B} \quad \bar{A}\bar{B} \quad \bar{A}B$

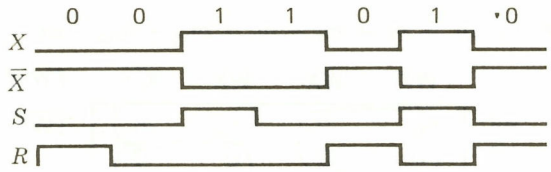


$CD + \bar{A}\bar{B}D$

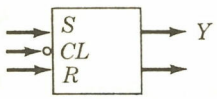
CAPÍTULO 4

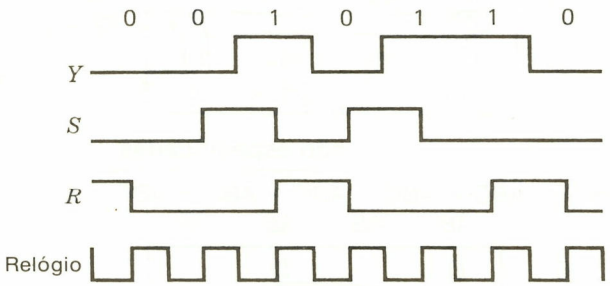
1.



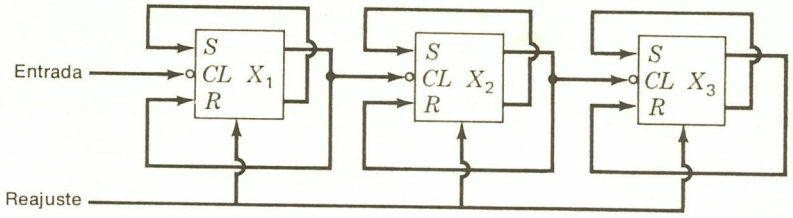


5.

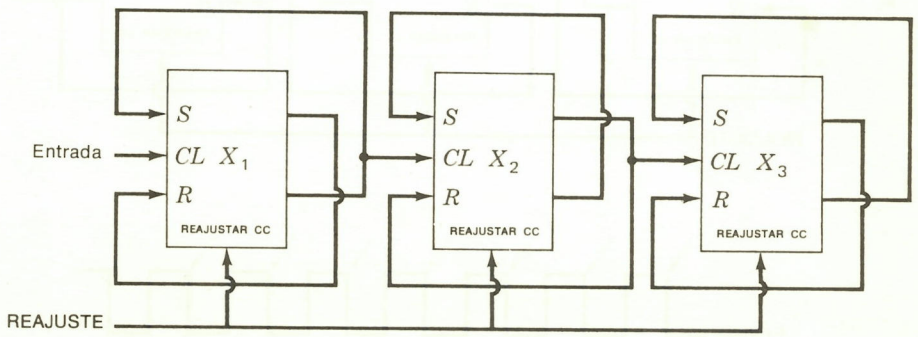


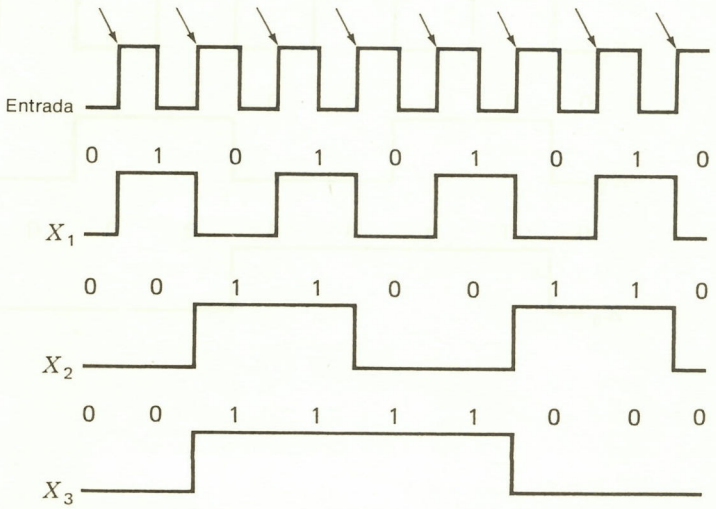


9.

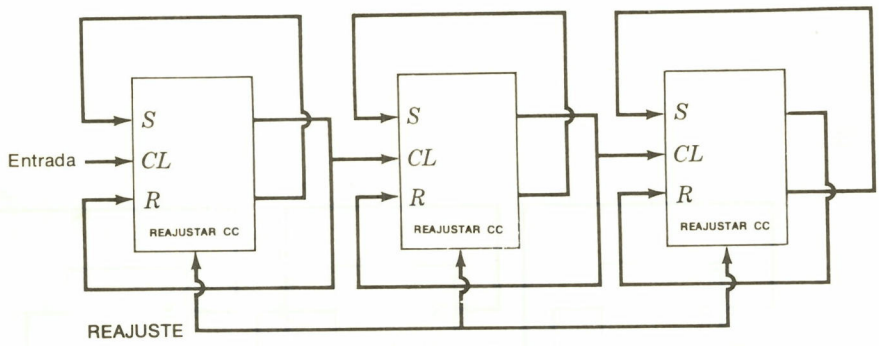

Entrada
S
CL X1
R
S
CL X2
R
S
CL X3
R
Reajuste

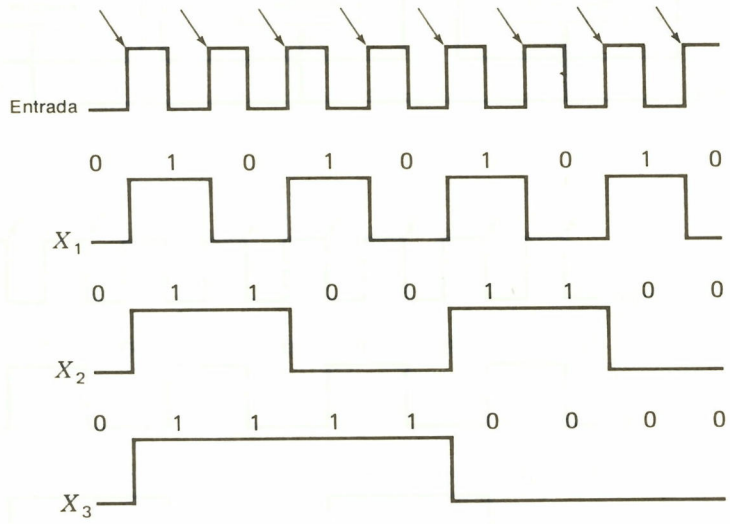
13.


Entrada
S
CL X1
R
REAJUSTAR CC
S
CL X2
R
REAJUSTAR CC
S
CL X3
R
REAJUSTAR CC
REAJUSTE

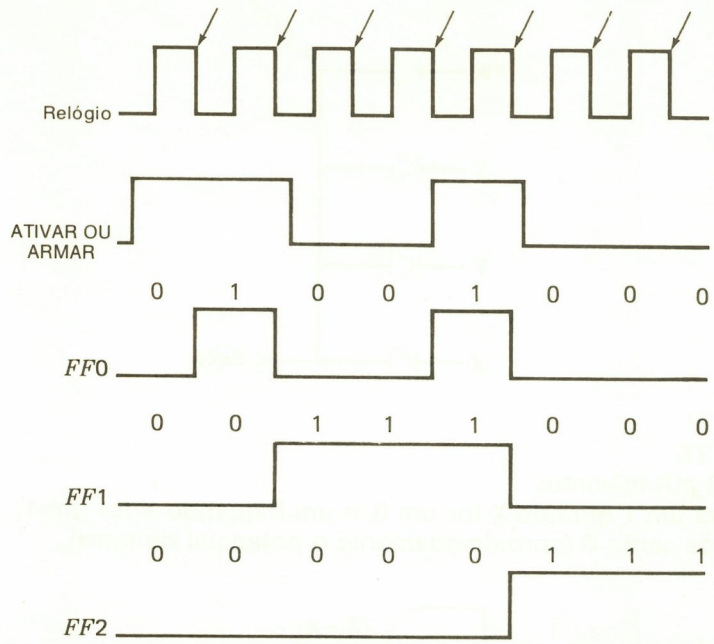

Entrada
0 1 0 1 0 1 0 1 0
X1
0 0 1 1 0 0 1 1 0
X2
0 0 1 1 1 1 0 0 0
X3

15.


S
CL
R
REAJUSTAR CC
Entrada
REAJUSTE

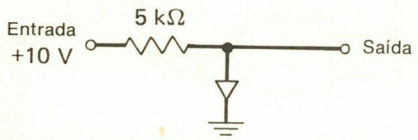

Entrada
0 1 0 1 0 1 0 1 0
X_1
0 1 1 0 0 1 1 0 0
X_2
0 1 1 1 1 0 0 0 0
X_3

17.



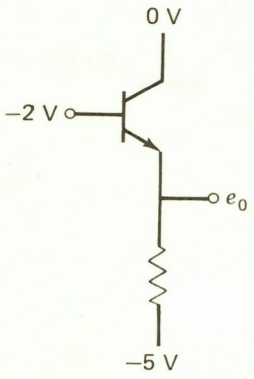
CAPÍTULO 5

1.



	Germânio	Silício
(a)	$e_0 = 0{,}3$ V	0,7 V
(b)	Aproximadamente 2 mA	

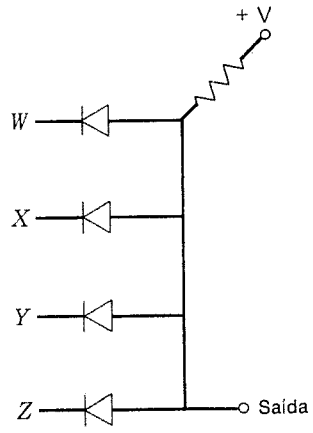
9.



(a) $e_o = -2{,}75$ V

(b) Sim; a junção emissor-base é polarizada diretamente (a base do *pnp* mais positiva do que o emissor).

13.



15. +0,1 V

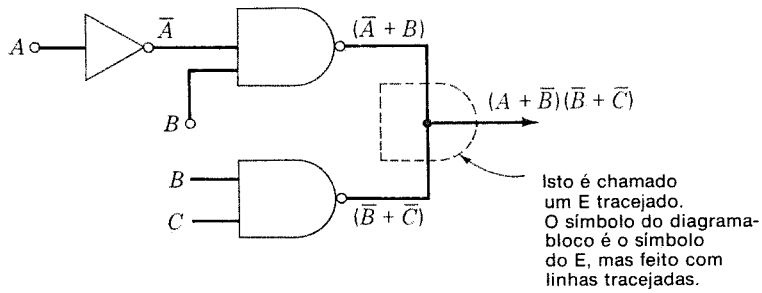
25. T3 e T5

31. 8,333 mA máximos

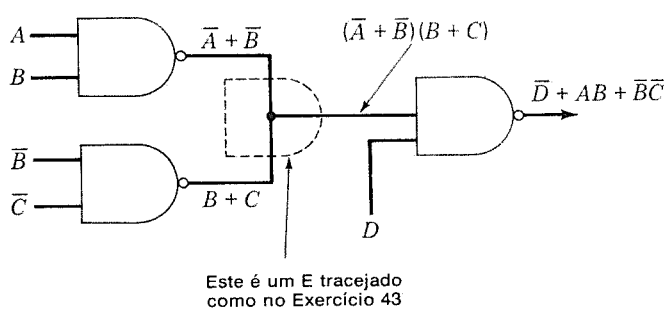
35. $\bar{X}$ será um 1 quando X for um 0, e um 0 quando X for um 1.

41. Ambos serão 0 (aproximadamente o potencial de terra).

43.



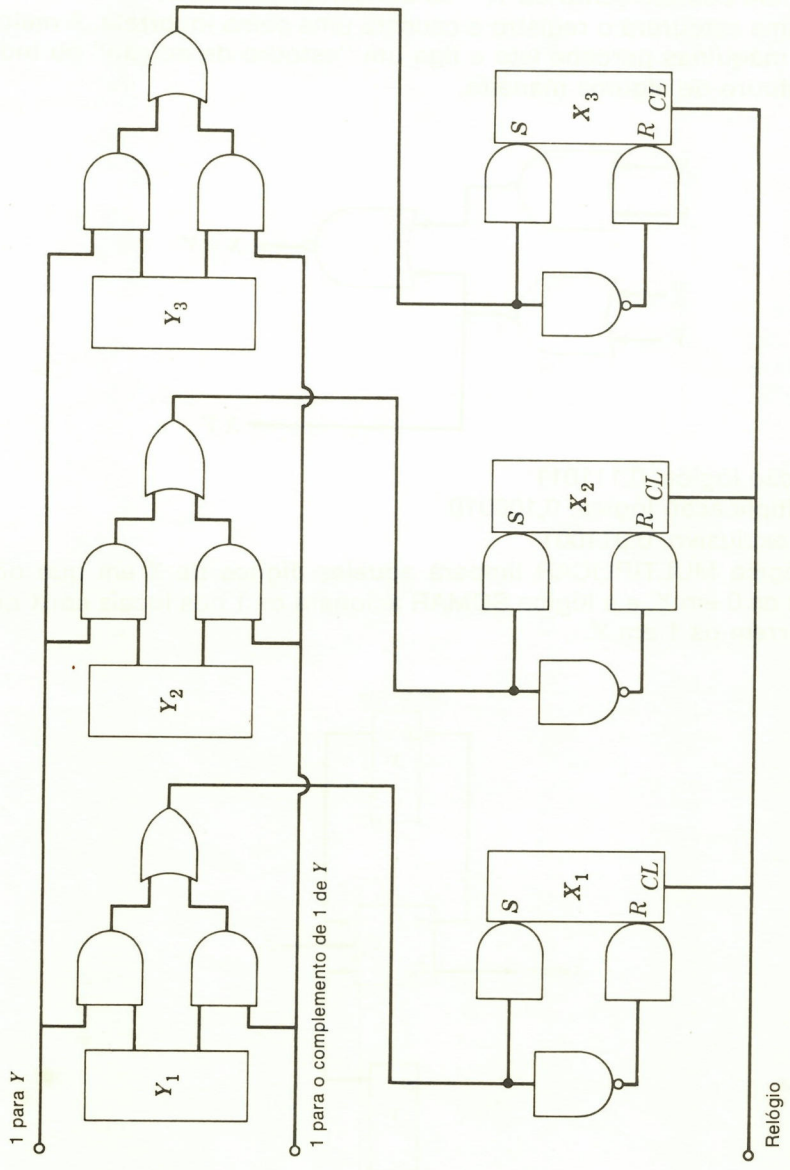
45.



49. 200 μA

CAPÍTULO 6

1.



3. *(a)* 0,0110 *(b)* 0,1010 *(c)* 1,1100 *(d)* Não representável

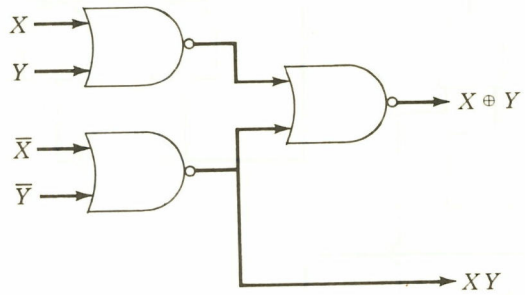
5. −4 deve ser armazenado 1,0100 no sistema de magnitude, 1,1011 no sistema de complemento de 1 e 1,1100 no sistema de complemento de 2.

7. s = 1 e $c = 1$

9. −12 em complemento de 1; −13 em complemento de 2.

11. A soma estourará o registro e causará uma soma incorreta. A maioria das máquinas percebe isto e liga um "estouro de adição" ou indica o estouro de alguma maneira.

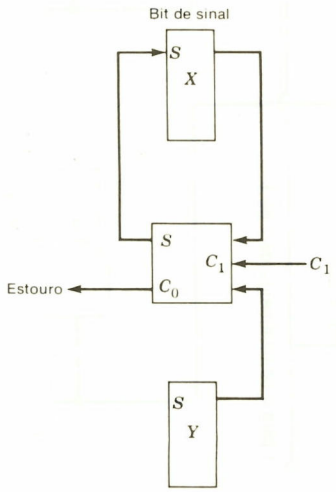
15.



27. Adição lógica, 0,111011
Multiplicação lógica, 0,100010
OU exclusivo, 0,011001

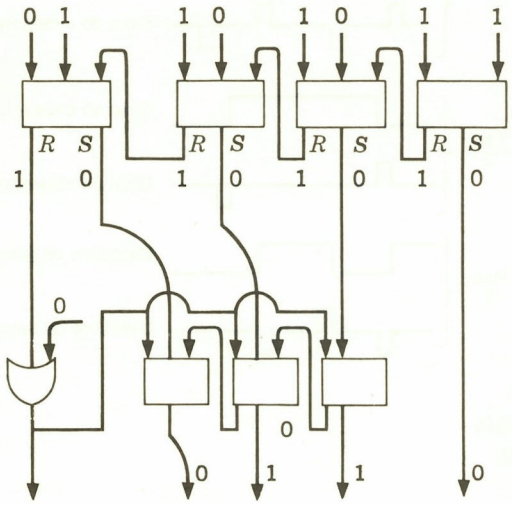
29. A lógica MULTIPLICAR limpará aqueles dígitos de X em que ocorrem os 0 em Y, e a lógica SOMAR acionará os 1 nos locais em X onde ocorrem os 1 em Y.

41.



Estouro $= (A > \emptyset) \cdot (B > \emptyset) \cdot (C_1 = 1) + (A < \emptyset) \cdot (B < 0) \cdot (C_1 = \emptyset)$
Estouro $= \overline{XY}C_1 + XY\overline{C_1}$

43. O esquema geral é:



53. *(a)* 0 0000 0000 0100 0010
(b) 1 0000 0001 0010 0010
(c) 0 0001 0000 0011 0100
(d) 1 0001 0000 1001 0111

57. *(a)* 0 0000 0000 0100 0010
(b) 1 1111 1110 1101 1110
(c) 0 0001 0000 0011 0100
(d) 1 1110 1111 0110 1001

CAPÍTULO 7

1. 143.360 núcleos
12 biestáveis para registro de endereços
35 biestáveis para o registro da memória intermediária

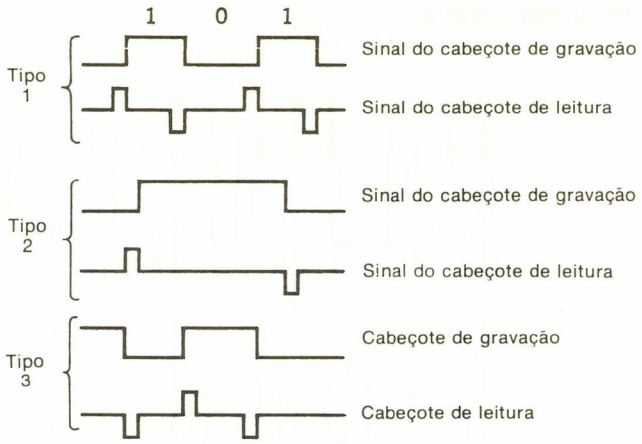
3. 147.456 núcleos; 36 amplificadores de leitura; 36 excitadores de inibição; 64 excitadores X e 64 excitadores Y; 36 planos e 4.096 núcleos por plano

5. Falso, porque todos os núcleos selecionados estão ajustados no estado 0, e os 1 devem ser regravados.

7. X_2Y_2, X_2Y_3

9. *(a) 1*

11.



41. 40 μs, 20 μs
43. 3.600.000 bits/s
45. 262.500 bits/s

CAPÍTULO 8

3. 01100001
01100100
01100100
00010000
00010110
00000100
00000001
10000000 Retorno do carro talvez seja opcional.

5. 27 orifícios nos três cartões. Mais fácil corrigir se forem usados cartões, pois os cartões errados simplesmente podem ser substituídos.

7. Nove orifícios

9.

Decimal	Código excesso de 3	Teste de paridade par	Teste de paridade ímpar
0	0011	0	1
1	0100	1	0
2	0101	0	1
3	0110	0	1
4	0111	1	0
5	1000	1	0
6	1001	0	1
7	1010	0	1
8	1011	1	0
9	1100	0	1

11. *(a)* Os erros estão nas décima sétima, oitava e nona fileiras; o sexto dígito em cada fileira está errado, e a mensagem é *"that's right"*.
(b) O erro está no quarto dígito na sexta fileira, e a mensagem é *"don't stop"*.

13. Um tal código pode ser formado acrescentando-se um 0 dianteiro em cada um dos grupos de código Gray de quatro dígitos binários listados no Cap. 8 e depois acrescentando-se outras 16 fileiras com os 4 dígitos binários mais à direita, tal como no Cap. 8, mas com a ordem invertida e com um 1 dianteiro acrescentado a cada uma dessas fileiras de código.

15. *(a)* Os erros estão nas segunda, terceira e quarta fileiras, o décimo sétimo dígito de entrada.
A mensagem é *"that's fine"*.
(b) Os erros estão nas segunda e terceira fileiras, o terceiro dígito de entrada, e a mensagem é *"bad tapes"*. Observemos que corrigimos um erro duplo numa coluna.

17. 0101 dá $+2^2/_3$ V.
1010 dá $+5^1/_3$ V.
1111 dá +8 V.

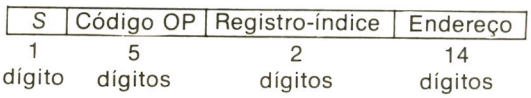
19. O código detectará mas não corrigirá erros que ocorram em duas colunas diferentes. Ele não detectará nem corrigirá quatro erros que ocorram nas duas fileiras e nas duas colunas, isto é, dois erros em cada uma das duas fileiras com erros nas mesmas duas colunas em cada fileira. Há muitos outros padrões destes tipos, e a análise destes códigos é muito sutil.

23. O código no Exercício 15 corrigirá todos os erros no Exercício 11. O código no Exercício 11 corrigirá os erros no Exercício 15*(a)*, mas somente detectará os do Exercício 15*(b)*.

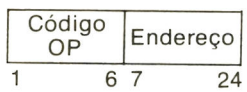
27. Erros duplos numa fileira conduzem à detecção através dos dígitos de teste. Uma vez que não há essencialmente testes com dígitos de teste, os erros nestes dígitos resultam em perdas de dados.

CAPÍTULO 9

1.

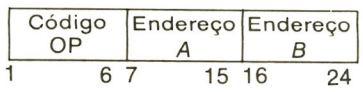


3. Endereço único



2^{18} diferentes endereços

Endereço duplo



2^9 diferentes endereços

5. Aqui está um possível programa:

```
CLA  25
MUL  24
MUL  29
STO  30
CLA  25
MUL  28
ADD  30
ADD  27
STO  30
```

25 contém *X*
27 contém *C*
28 contém *B*
29 contém *A*

7. Bit de sinal ⟷ 1112 ⟷ 151617 ⟷ 2324 ⟷ 30



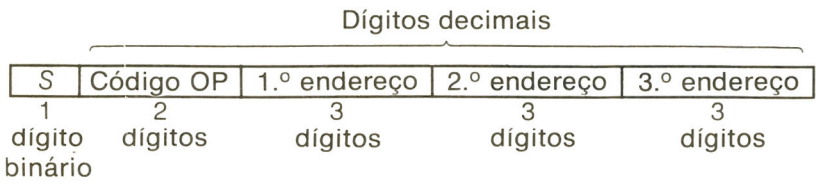
11.
```
CRA  302
MPX  302
ADD  301
DIV  303
STA  304
```

13. A instrução SBM subtrai a magnitude do número armazenado no local fornecido pela parte de endereço da palavra de instrução do número no acumulador, e coloca este valor da diferença no acumulador. A instrução é indexável e indiretamente endereçável e pode resultar num estouro.

15. Mudar cada AC_0 para $\bar{A}\bar{C}_0$, cada $\bar{A}\bar{C}_0$ para AC_0 e BRM para BRP.

CAPÍTULO 10

1.



3. Quatro viagens para uma instrução de adição

5.

Endereço	Registro-índice	Código OP	Parte de endereço
0	01	SIR	39
1	01	CAD	200
2	00	ADD	300
3	00	STO	300
4	01	BRI	1
5	00	HLT	0
Loca-	200		
lização	239		
de dados	300	contém 0 quando o programa é iniciado.	

ÍNDICE ALFABÉTICO

Os números em *itálico* referem-se a localizações fora do texto (legendas, quadros, dísticos, notas, etc.).

A

Acesso
- a uma trilha, 280
- aleatório, 231
- seqüencial, 231
Acoplador acústico, 326
Acumulador(es), 16, 186
- de destino, 425
- fonte, 425
- múltiplos, 372
Adaptadores de interface de periféricos, 393
Adição
- aritmética, 187
- lógica, 62, 218, 220
- no sistema de complemento de 1, 192
Álgebra
- Booleana, 61
- - conceitos fundamentais de, 62
- - dualidade básica da, 72
- - leis básicas da, 68-70
- de comutação, 61
- de conjuntos, 61
- Lógica, 61
- Moderna, 61
Alimentação de fita, 304
Amplificador(es)
- de ajuste de nível, 331
- de leitura, 238
Anodo, 143
Aritmética de ponto fixo, 220
Armazenamento
- de informação em núcleos magnéticos numa matriz bidimensional, 235-238
- de reserva, 230
- dinâmico, 231
- em núcleos magnéticos, 233-235
- em tambor magnético, 270-271
- estático, 231
- interno, 230
- secundário, 230
Arquitetura de computadores, 372

B

Barramento, 387
- assíncrono, 393
- de dados, 389
- de transferência, 389
- geral de controle de interface, 389
Base, 144
Biestável(is)
- a transistor, 156-157
- com circuitos lógicos, 129-131
- - projetos de, 129
- D, 114
- JK, 114
- mestre-escravo, 109, 131
- RS, 106
Binário(s)
- adição em, 40
- conversão de números decimais em, 42-43
- divisão em, 40
- multiplicação em, 40
- representação de número decimal codificado em, 46-48
- subtração em, 40
Bit
- de encadeamento, 404
- de máscara de interrupção, 403
- de página, 378
- de paridade
- - ímpar, 323
- - par, 323
- L, 404
Bordo
- crescente negativo, 108
- crescente positivo, 108
- de queda, 108
- de subida, 108
- dianteiro, 108

C

Cabeçote(s)
- de leitura-gravação, 280
- volantes, 276, *276*
Calculadoras, 1-3
Calculi, 1
Cálculo
- de expressões lógicas, 65-66
- de proposições, 61
- de uma expressão que contém parênteses, 67
Característica, 222
Carga padrão, 122
Carregamento, 153
Cassetes de fita, 285, 286, *290*
Catodo, 143
Ciclo
- de execução, 345
- de instrução, 345
- de memória, 240
Circuito(s)
- a semicondutor metal-óxido, 170-173
- basculadores, 105
- com repartição de alimentação, 162
- combinacionais, 75
- de chaveamento, 143
- de computador, 142
- de pólo totem, 162
- de transferência, 107
- de um quarto de soma, 188
- digitais, 142-184
- E a diodo, 151-152
- integrados, 4
- inversor, 153, *153*
- lógicos, 74
- lógicos de CMOS, 173-177
- meia-soma, 187
- porta, 61-104
- somador completo, 188-189, *189*
Classificadora, 307, *309*
Cobertura, 86
Codificação, 16
Codificador
- exatidão do, 328
- precisão do, 328
Código
- ASCII, 310, *312*
- Baudot, 304
- BCD, 310, *311*
- de operação, 16, 341
- EBCDIC, 310, *313*
- Gray, 329
- Hollerith, 306, *307*
- OP, 341
Coffin, 118
Coletor, 144
Complementação, 64
Complementador, 203
Complemento
- de base-menos-um, 45
- verdadeiro, 45
Comprimento da palavra, 231, 232
Computador(es), 1
- aplicação(ões)
- - aos problemas, 5-6, *5*
- - científicas, 7-8

- - no comércio e na indústria, 6-7
- com múltiplos acumuladores, 425
- compartilhado em tempo, 14
- compatíveis para cima, 427
- de canalização, 387
- de dois endereços, 422-427
- de múltiplos endereços, 16
- de programa armazenado, 4, 339
- de propósitos especiais, 339
- de propósitos gerais, 339
- de tempo compartilhado, 11
- de um só endereço, 16
- digital(is)
- - com propósitos especiais, 13
- - componentes básicos, 14-15, *14*
- - de propósitos gerais, 13
- - endereço, 15
- - local na memória, 15
- - uso de, nos sistemas de controle, 11-13
- eletrônicos digitais, 3-4
- indústria de, 1
- leitura do programa no, 20-25
- operação do, 1-35
- - construção da memória, 15
- - instrução(ões), 15
- - - de multiplicação, 18-19
- - linguagens de compilador, 29-30
- - linguagens de montador, 27-29
- - sistemas de programação, 25-27
- organização de, 371-437

Configuração Darlington, 161
Conjunto de circuitos integrados monolíticos, 126
Contador(es)
- binário, 112-115
- de BCD, 115-117
- de instrução, 345
- de módulo 8, 113
- de módulo 15, 113
- de ondulação, 114
- de quatro estágios, 113
- de três estágios, 113
- projeto de, 133-135

Controlador, 389
Conversão
- A/D, 328
- analógica/digital, 328

Conversor(es)
- analógico/digital, 11, 329
- digital/analógico, 12, 329
- eletrônicos analógicos/digitais de alta velocidade, 331-334
- instantâneos, 333
- simultâneos, 333

Cópia de leitura direta, 303
Corrente
- convencional, 143
- de corte, 149

D

Declaração, 27
Decodificador, 243, 246
- de sete segmentos, 126
- de uma ou várias funções, 243, 246

Dedução de uma expressão booleana, 72-74
Desempilhamento, 374
Desvios condicionais, 339
Diagramas de Veitch, 84
Dígito(s)
- de mão direita, 213
- de mão esquerda, 213
- de marca de item, 422

Diodo, 143
- polarizado diretamente, 143
- polarizado inversamente, 143
- Schottky, 163, *164*

DIP, 118
Disco *floppy*, 279, *280*
Dispositivo(s)
- de acesso em série, 231
- de armazenamento de acesso direto, 231
- de entrada-saída, 300-338
- - cartões perfurados, 306-308, *307*
- - codificadores de posição angular, 328
- - códigos
- - - alfanuméricos, 310
- - - de correção de erro, 323
- - - de detecção de erro, 323
- - conversores
- - - analógico/digital, 328
- - - digital/analógico, 329-331
- - - eletrônicos analógicos/digitais de alta velocidade, 331-334
- - de sistemas com componentes analógicos, 327-328
- - diferenças entre as aplicações comerciais e científicas, 301
- - equipamento de saída, 316-317
- - fita perfurada, 301-304, *301, 302, 303*
- - impressoras, 317-320
- - leitoras de cartões, 308-310
- - leitoras de fita, 304-306, *304, 305*
- - preparação de dados, 310-315
- - reconhecimento de caracteres, 315-316, *315*
- - terminais, 324-327
- - teste de paridade, 323
- de exposição visual, 10
- periféricos, 382

Divisão, 214-218, *215, 216, 217*

E

Elemento de memória, 230-299
- armazenamento em núcleos magnéticos, 233-235
- decodificadores, 246-249
- excitação das linhas de seleção X e Y, 243-244
- seqüência de sincronização, 240-241

Embalagem em linha dupla, 118
Emissor, 144
Empilhamento, 374
Emulação, 368
Endereçamento
- direto, 375
- indireto, 378

Endereço, 15, 16
Enrolamento
- de bit, 251
- de dígito, 251
- de leitura, 234, *236*
- inibidor, 240

Escovas de leitura, 308
Espaço, 325
Estojos de fita, 285, 286, 289, *290*
Estruturas de dois níveis, 77
Exatidão do codificador, 328
Excitadores de linha de três estados, 389
Expansão canônica, 73
Expressão
- de produto-de-somas, 76
- de soma-de-produtos, 76

F

Fator de carga
- de entrada, 153
- de saída, 153

Firmware, 368
Fita magnética, 280-285
- equipamentos de comutação e de memória intermediária, 284
- mecanismo de transmissão da, 282
- - características, 284
- sistema de leitura e gravação, 282

Flip-flop, 105
Forma complementada, 44
Fração, 222

G

Gerador
- de registro de deslocamento linear, 120
- de seqüência aleatória, 120
- de seqüência recurrente linear, 120

I

Impressora(s), 317-320
- de linhas, 318
- de uma linha de cada vez, 318, *320*
- eletromagnéticas, 318
- eletrostáticas, 318
- por pontos, 317
- sem impacto, 318
- térmicas, 318

Indexação, 376
Indicador de pilhas, 374
Indicadores visuais de osciloscópio, 320
Instruções
- do modo do usuário, 428
- do modo supervisor, 428
- de desvio, 19
- - condicionais, 19
- - incondicionais, 19
- de salto, 19

Integração

- em larga escala, 126
- em média escala, 126
Intercaladora, 307, *309*
Intervalo de blocos, 284
Inversor, 65, *65*, 148

J

Junção de semicondutor, 143

K

Karnaugh, mapa de, 84, *84*

L

Laço de histerese, 234, *235*, *236*
Latência, 276
Lei(s)
- associativa(s), 63, 69
- comutativas, 69
- distributiva, 69
Linguagem(ns)
- de alto nível, 29
- de compilador, 29
- de programação, 26
- orientadas para o problema, 29
- PL/1, 30
- de dígitos, 251
- de interrupção, 382
- de palavras, 251
Lógica
- acoplada pelo emissor, 167
- bipolar, 122
- de modo corrente, 167
- de orientação de corrente, 167
- diodo-transistor, 159
- não-saturada, 167
- positiva, 107
- resistor-transistor, 158
- transistor-transistor, 161

M

Mapa de Karnaugh, 84, *84*
Marca, 325
Matriz decodificadora, 246
Maxicomputadores, 371
Meias palavras, 430, *430*
Memória(s)
- auxiliar, 230
- bipolares, 255
- borrão, 255
- comprimento da palavra da, 232
- conteúdos, 232
- de acesso aleatório, 233
- - a semicondutor, 255
- de alta velocidade, 230
- de byte endereçável, 430
- de caráter, 430
- - endereçável, 422
- de controle, 363
- de disco magnético, 274, *275*
- de leitura exclusiva, 232, 264
- de leitura/gravação, 232, *233*
- de núcleos de quatro fios de acesso aleatório e de corrente coincidente, 249
- dinâmicas, 255, 262-264
- elemento de, 230-299
- estáticas, 255
- estáticas de MOS, 258-262
- fixas, 232
- interna, 230
- leitura não-destrutiva, 232
- principal, 230
- secundária, 230
Mensagens de interface, 389
Método
- das aproximações sucessivas, 333
- do mapeamento para simplificação de expressões, 83-86
Microcomputadores, 371, 377
Microoperações, 362, *362*
Microprocessador, 371
Microprograma, 362, *365*, 366
Microprogramação, 362
- horizontal, 364
- variações nas configurações de, 364
- vertical 364
Modem, 10, 326
Modo
- de multiprocessamento, 382
- de realce de corrente, 172
Módulo, 277
Mudança de escala, 220
Multiplicação
- binária, 211-213
- decimal, 213
- lógica, 62, 218
Multiprocessador, 384
Multiprogramação, 428
Multivibrador biestável, 105

N

Notação posicional, 37
Núcleo(s)
- características da matriz da memória de, 241-242
- características da memória de, 246
- completamente selecionado, 238
- magnético, 233, *234*
- semi-selecionados, 238
Número(s)
- binário
- complementos de, 45-46
- negativos, 43-44, 191
- - uso de complementos para representar, 44-45
- positivos, 191

O

Onda quadrada, 108
Operação(ões)
- assíncrona, 108
- binárias, 64
- condicionais, 209
- de deslocamento, 207
- de extração, 218
- de mascaramento, 218
- de soma na base 2, 218, 220
- de transferência, 108
- E, 218
- síncrona, 108
- singulares, 64
- unárias, 64
- unitárias, 64
Organização de computador(es), 371-437
- de endereço único, 404-409
- dispositivos de interface, 387-393
- endereçamento indireto, 377
- formatos da palavra de instrução, 372, *373*
- interligação dos componentes do sistema, 380-387
- microprocessador(es), 393
- - de endereço único, 409-422
- número de endereços, 372
- paginação, 377
- registros-índice, 376-377
- representação de instruções e dados, 374-375
- técnicas de endereçamento, 375
Osciloscópios de armazenamento, 321

P

Paginação, 378
Palavra(s), 15
- de computador, 340
- de instrução, 16, *17*
- - de um só endereço, 341
- duplas, 430, *430*
Parte de endereço, 341
Pilha, 373, 374
Plano de memória de núcleos, 238
Pontos remanentes, 234
Porta(s), 63
- combinação de, 74-75
- de transmissão, *174*, 176
- E, 63, *64*, 150-151
- NE, 80, 81, *81*, 82, *82*, 83
- NOU, 80, 81, *81*, 82, *82*, 83, 155-156
- OU, 63, *63*, 152
Precisão do codificador, 328
Prime implicant, 87
Problema
- da corrida, 112
- da propagação do "vai um", 199
Processamento em lotes, 8
Produto-de-somas, 75
- dedução das expressões de, 77-78
- técnica das expressões de, 90
Programa(s), 3
- de aplicações, 428
- de canal, 384
- de chamada, 406
- de serviço de interrupções, 408
- do usuário, 428
- fonte, 26
- objeto, 26
- relocalizável(is), 375, 428
Programação, 3
Programadores, 3
Projeto lógico, 105-141
Protocolos, 389
Prova por indução perfeita, 68, 70-71
Pulsos de relógio, 108
Pushing, 374

Q

Quantização, 328

R

Rede
- de decodificação tipo árvore, 248
- decodificadora multiplicativa balanceada, 248
Região de ambigüidade, 155
Registro(s), 44
- base, 432
- da memória intermediária, 232, 242, 244
- - e conjunto de circuitos associado, 244-245
- de deslocamento, 111
- - com realimentação, 120, *120*
- de endereço da memória, 232, 242, *242*
- de operação, 230
- de ponto-flutuante, 430
- de propósitos gerais, 372
- em teclado, 303
- gerais, 186, 430
- índice, 376, 432
Regra da mão direita, 233
Relógio, 108-111
- de múltiplas fases, 131
Repercussão do caráter, 325
Representável, 191
Retardo
- de latência, 275
- de tempo de armazenamento, 163, *164*
- rotacional, 275
ROM
- apagável, 267
- de campo programável, 267
- programável, 267
- reprogramável, 267
Rotinas de ponto-flutuante, 220
Roubo de ciclos, 382

S

Saídas
- *don't-cares*, 91
- não-especificadas, 91
Seguidor de emissor, 149-150
Segurança, 428
Seleção linear, 251
Semi-somador, 187
Separação entre registros, 284
Seqüência de sincronização, 240-241
Série-paralelo, 206
Simplificações de expressões, 71
Simulação, 367
Sistema(s)
- binário, 38-39
- - contagem no, 39-40
- - de inteiros com sinal, 186
- de armazenamento em disco flexível, 279-280
- de cabeçote fixo, 274
- de cabeçote movível, 274
- de caracteres endereçáveis, 15
- de computador em tempo compartilhado, 13
- de controle
- - de administração, 13
- - e comando, 13
- - em tempo real, 11
- de fase codificada, 295
- de freqüência dividida, 295
- de inteiros binários de magnitude com sinal, 186
- de interação em-linha, 8
- de marca bifásica, 295
- de notação posicional, 38
- de numeração, 37-60
- - base de, 38
- - hexadecimal, 48-52
- - octal, 48-52
- - raiz de, 38
- de palavra endereçável, 15
- de programação, 25-27
- - compiladores, 26
- - montadores, 26
- decimal, 37-38
- deslocado, 222
- Harvard, 295
- Manchester, 295
- multiprocessadores, 387
- operacional, 428
Smart, 325
Soma-de-produtos, 75
Somador
- binário
- - completo, 188
- - paralelo, 190-191
- completo, 197-199, *197*
- de decimal-codificado-em-binário, 199-200
Subcubo, 86
- máximo, 87
Subprograma
- chamado, 406
- recursivo, 408

T

Tabelas
- da verdade, 65
- de combinações, 65
- de contador, 121
- de projeto de contador, 133
Tambor magnético
- operação em série e em paralelo de um, 271-274, *274*
Tape feed, 304
Técnica(s)
- de gravação
- - de não-retorno a zero, 294, *294*, 295, *295*
- - de retorno à magnetização fundamental, 292, *293*
- - de retorno a zero, 290, 291, 292, *293*
- - digital, 289
- de rearmazenamento, 214
- de seleção, 235
- - de corrente de coincidência, 235
- sem armazenamento, 214
Tecnologia de circuitos integrados, 117
Teletipo, 9, *9*
Tempo
- de acesso, 15, 231, 275
- - de leitura, 246
- de busca, 275
- de gravação, 240
- de leitura, 240
- de retardo, 109
- de retorno, 13
Tensão de ruptura, 149
Teoremas de De Morgan, 71
Terminal(is), 324
- de computador, 10
Termo
- mínimo, 84
- produto, 76
- soma, 76
Transferência, 107-108
Transistor(es)
- biestável a, 156-157
- de efeito de campo, 170
- de junção, 144
- *npn*, 144, *146*
- *pnp*, 144
- - características do, num circuito de base-comum, 148-149
- região
- - ativa, 145
- - de corte, 145
- - saturada, 145
Transmissão
- assíncrona, 325
- síncrona, 325
Transposição, 149
Trilha, 270, *272*, 274
- externa, 274
- interna, 274
Tubos
- Charactron, 322, *322*
- de raios catódicos
- - dispositivos de saída em, 320-323

U

Unidade(s)
- aritmética e lógica, 185-229
- - adição
- - - no sistema de complemento de 1, 192
- - - no sistema de complemento de 2, 193-194
- - adição e subtração num elemento aritmético paralelo, 194-197, *196*
- - circuito meia-soma binário, 187-188
- - circuito somador completo, 188-189
- - divisão, 214-218, *215*, *216*, *217*
- - execução de operações aritméticas com números de ponto-flutuante, 224
- - multiplicação binária, 211-213
- - operação(ões)
- - - básicas, 207-211
- - - de deslocamento, 207
- - - lógicas, 218
- - representação de inteiros, 186-187
- - sistemas de numeração de ponto-flutuante, 220-224
- - somador
- - - binário completo, 190-191

- de tempo, 270
- periféricas de processamento, 384

V

Veitch, diagramas de, 84
Verificadora, 307, *309*
Via, 387
Voltímetros digitais, 334

ou de transferência, 356-358
- - microprogramação, 362-364
- - organização do ciclo de instrução e do ciclo de execução dos registros de controle, 342-347
- - seqüência de operação dos registros de controle, 347
- - seqüência típica das operações, 355-356
- - - completo, 197-199, *197*
- - - de decimal-codificado-em-binário, 199-200
- de controle, 340
- - construção da palavra da instrução, 340-342
- - controle das operações aritméticas, 347-355, *348*
- - instruções de deslocamento, 358-362
- - instruções de salto, de desvio

Impresso na
GRÁFICA EDITORA PRIMOR S.A.
Rodov Pres. Dutra, 2611
Rio de Janeiro — Brasil