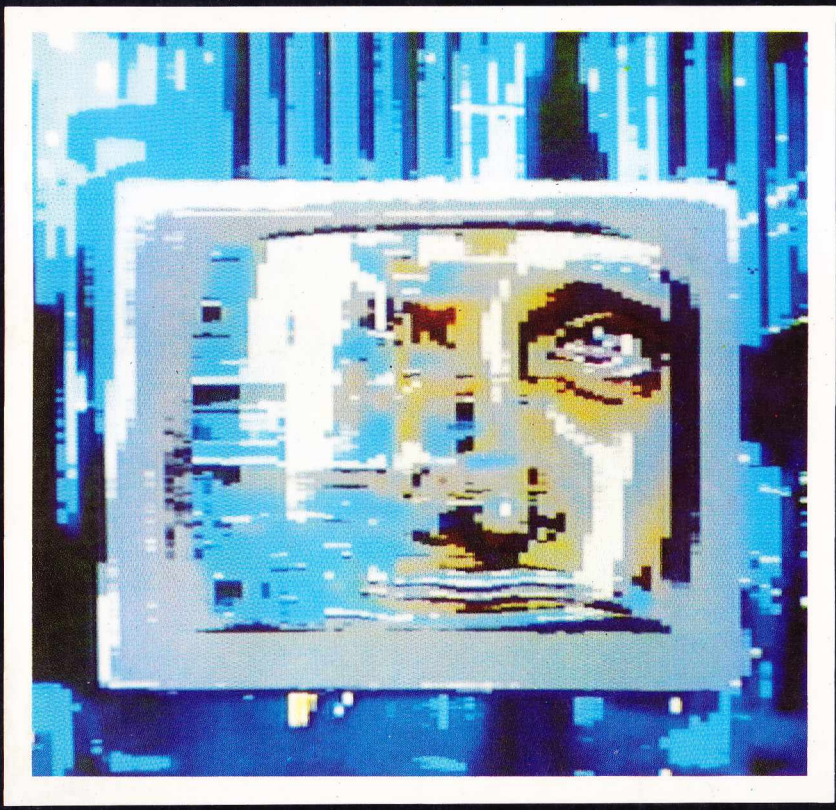
BIBLIOTECA BASICA

INFORMÁTICA

DENTRO E FORA DO COMPUTADOR 1



SECULO FUTURO

BIBLIOTECA BÁSICA

INFORMÁTICA

DENTRO E FORA DO COMPUTADOR 1

SÉCULO FUTURO

Diretor editor:
M.A.Nieto
Coordenação e supervisão técnica:
Eng.º Sergio Rocha Paggioli
Tradução:
Ideli Novo
Projetos especiais:
Rainer K.E. Ladewig
Diretor de arte:
Duilio Sarto F.º
Studio editorial:
Auro Pereira da Silva (chefe), Susana M.Amaral Couto (revisão), Luiz Carlos Siqueira Lago (prod. gráfica), Antonio Carlos Martins, Rubens Tadeu Benedito
Fotocomposição, fotolito:
Omnicolor Gráfica e Propaganda Ltda - Rua Dr. Virgilio de Carvalho Pinto, 619 Pinheiros - CEP 05415 - São Paulo
Impressão
Editora Antártica S.A. - Av. Ramon Freire, 6920 (Pajaritos) - Santiago - Chile

ÍNDICE

PREFÁCIO

A rápida evolução da microeletrônica tornou possível que a aquisição de um computador pessoal seja algo praticamente, ao alcance de todos. A maioria destes equipamentos, apesar de seu tamanho reduzido, esta dotada de uma potência de cálculo igual ou superior à de um enorme computador dos anos 50, cuja instalação exigia enormes espaços.

Mas a evolução tecnológica não influenciou unicamente nas dimensões da máquina. De fato, basta olhar qualquer revista especializada para ver-se "cercado" por páginas e páginas com publicidade de computadores, acessórios, periféricos cada vez mais aperfeiçoados e, também, por montanhas de software. Nesta verdadeira selva de opções, o usuário (ou aquele que pensa ser no futuro) pode perder-se com facilidade, se não tiver um mínimo de conhecimento sobre o que contém na realidade estas caixas tão bem apresentadas. É necessário saber o que tem dentro e fora deste que pode ser um computador. Algo assim acontece com aqueles automóveis que tem carrocerias muito parecidas, mas cujos motores e mecânica são muito diferentes. Não devemos nos deixar enganar.

O conjunto de elementos físicos que constituem um computador é conhecido como hardware, enquanto que o software é a série de programas que lhes permitem funcionar de modo correto. Ambas as partes são interdependentes: se o hardware é pouco potente, será difícil obter o máximo rendimento inclusive do

melhor software disponível e vice-versa. Por isto, uma das intenções deste livro é precisamente introduzir o leitor no mundo dos chips, que constituem o "coração" do sistema (CPU, RAM, ROM, etc.) e àqueles que vão por fora (periféricos). Conhecer o hardware ajuda muito na hora de selecionar um computador ou, se já possui um, permite tirar o máximo proveito e obter sólidos critérios para a posterior aquisição de impressora, memórias de massa e periféricos diversos. Aliás, como é possível "conviver" com nosso próprio computador pessoal sem ter mínimo de desejo de conhecer o seu interior?

Se conseguimos transmitir-lhe "o vírus" acompanhe-nos em nossa viagem. Trataremos de adentrarmos juntos no mundo do hardware: cartões principais, de expansão, de aplicação, de comunicação... e, como não! Veremos os chips, esses circuitos eletrônicos, integrados numa pastilha, que cobrem a superfície de qualquer cartão.

Nossa forma de expor-lhe os temas será séria e clara, sem ser massante (pelo menos é o que esperamos), com um progressivo aprofundamento para evitar "sustos". Com esta mesma finalidade, incorporaremos a cada capítulo um glossário, feito para ajudá-lo na compreensão das terminologias mais difíceis ou menos conhecidas, e que em muitos casos poderá ajudá-lo quando aparecer em algum texto um novo conceito para você.

Para concluir, faremos um apanhado do conteúdo dos diversos capítulos. Começaremos pela abertura do próprio computador pessoal, operação imprescindível se desejamos examinar o que observamos à primeira vista (cartões, elementos mecânicos, disposição dos diversos componentes). No segundo capítulo, falaremos dos circuitos digitais elementares que, integrados a milhares, darão lugar ao uP ("microprocessador") e a seus diversos chips de apoio, assim como o funcionamento de um microprocessador. No terceiro capítulo veremos o que é e qual a missão de cada chip associado ao CPU, estabeleceremos contato com as memórias RAM e ROM, os dispositivos de entrada/saída e suas características principais. No quarto capítulo falaremos do software, considerado como elemento aglutinante e controlador do hardware anteriormente descrito, com o que chegaremos à metade da exposição. No quinto capítulo examinaremos, de forma mais amena possível, os "zeros", "uns", códigos hexadecimais, código ASCII, etc., com a finalidade de poder aprofundar um nível a mais no hardware. Os capítulos seguintes, sexto e sétimo, são os mais longos e descritivos, pois, neles conheceremos e analisaremos todos os tipos de periféricos de um computador pessoal, posto que, chegando a este ponto, dis-

poremos de todos os instrumentos necessários para fazê-lo com a máxima objetividade. No oitavo capítulo, de modo a concluir, trataremos de fixar as idéias e dar algumas recomendações finais ao leitor.

Confiamos que nosso empenho de fazer assimiláveis algumas idéias e conceitos tão difíceis não nos leve a perder essa rigorosidade também necessária. Você, amigo leitor, será o juiz.

CAPÍTULO I

DENTRO DA CAIXA

Já possuimos um computador colocado em nosso escritório e disposto a nos servir tanto mais fielmente, quanto mais tempo lhe dedicarmos. Dispomos dos manuais, sobretudo o do usuário, que leremos primeiramente, e uma boa variedade de discos e cassetes que contém programas concebidos para ajudar-nos a resolver muitos problemas cotidianos. Por hipótese, também temos um cabo de conexão à rede, que se introduzirmos na tomada de corrente e ligarmos o interruptor do sistema, este tomará vida como por arte de magia e, desde este preciso momento em diante, será "todo nosso". Neste ponto teremos uma dúvida: Como é possível que tudo funcione conforme as descrições do manual, quando é uma caixa tão pequena a que temos diante de nós? O que tem dentro dela? E sobretudo, é possível que, conhecendo também o interior do computador compreendamos melhor a forma de utilizálo?

Casualmente, no escritório encontra-se também o primeiro livro da coleção "Biblioteca Básica de Informática" que, não somente não desaconselha a abertura da caixa, como convida o proprietário a olhar com a máxima atenção seu conteúdo. Assim, estamos aqui com o manual técnico, com a chave de fenda e uma boa dose de espírito de aventura, dispostos a abrir a caixa do nosso computador pessoal.

Algumas considerações sobre o aspecto externo antes de entrarmos no interior

A fotografia 1 mostra alguns dos computadores pessoais mais difundidos. Talvez o seu se encontre entre eles. De imediato você observará uma característica comum em sua estrutura externa: todos têm uma caixa que contém a principal parte do sistema eletrônico e que, em alguns modelos, inclui também o teclado (Apple, Spectrum,...) e/ou as unidades de disco flexível (IBM-PC). Existe ainda um aparelho com forma de tela de televisão, que recebe o nome de monitor de vídeo. Em sua tela poderemos visualizar caracteres alfanuméricos (letras e números) ou gráficos. Na maior parte dos casos o monitor é externo e separado da estrutura principal, ainda que existam excessões que o incorporam na própria caixa. Nos computadores pessoais, denominados "portáteis" (por suas dimensões reduzidas) é comum substituir o monitor por um visor ("display") de cristal líquido.

A escolha do computador pessoal efetuada sobre a base de considerações exclusivamente estéticas não costuma ser conveniente, ainda que sua apresentação seja um detalhe que deve ser

Foto 1 - Estes são alguns dos computadores pessoais existentes.

levado em conta. As dimensões, peso, facilidade de manuseio, portatibilidade e durabilidade de seu teclado influenciam, às vezes de forma determinante, na boa relação entre usuário e sistema. Ao abrir a tampa, poderemos também avaliar a mecânica, distribuição e ordem do conjunto e isto é importante não porque temos curiosidade do seu interior como atividade cotidiana, mas sim porque devemos avaliar antecipadamente os problemas que apresentarão à conservação e inclusive os consertos.

Por outro lado, a partir do exterior podemos avaliar melhor a robustez do aparelho. As caixas de materiais plásticos são mais resistentes aos golpes do que as caixas de metal, e melhor são dispostos os reforços internos metálicos. Assim teremos caixas mais leves, resistentes e, normalmente, mais estéticas. Uma boa caixa assegura uma vida mais longa aos componentes internos.

Nas fotografias 2 e 3 aparecem, a título de exemplo, o famoso Apple II (na versão "e"). Na primeira vemos seu aspecto externo, enquanto que a segunda nos permite observar com clareza seu interior, onde podemos ver o cartão principal (único) que serve de suporte a toda a parte eletrônica e a fonte de alimentação, protegida por uma caixa metálica.

Em um bom computador pessoal, a estrutura interna deve ser simples e cada componente ter um fácil acesso. Por isto, vale a pena abrir a famosa tampa, antes de comprá-lo. Em qualquer caso, é fácil indicar algumas normas que servem de base na avaliação. Antes de tudo, quanto menos componentes tiver, menores serão as possibilidades de avaria. Nas máquinas de recente fabricação, um número reduzido de componentes indica que foram empregados circuitos integrados "feitos sob medida", o que nos faz pensar que ajustar-se-á melhor ao que se espera dela e que sob o ponto de vista tecnológico, será mais moderna.

Em segundo lugar devemos avaliar o desenho e a fabricação das placas eletrônicas existentes. O emprego de uma base (naturalmente se são de qualidade) é uma vantagem, pois facilita enormemente as tarefas de manutenção e reparo, em contraposição com aquelas placas em que os componentes estão soldados. Se existirem conectores, deverão ser facilmente acessíveis, ter uma conexão robusta e segura e, se possível, normalizadas (standard), pois assim será mais fácil encontrá-los se alguma vez precisarmos. O interior deve ser bem ventilado e a presença de um ventilador (silencioso) é, sem dúvida, garantia de uma vida mais prolongada para os componentes.

Feitas estas considerações, podemos centrar-nos na eletrônica contida no interior, no cartão principal.

Foto 2 - Unidade principal do Apple II-e.

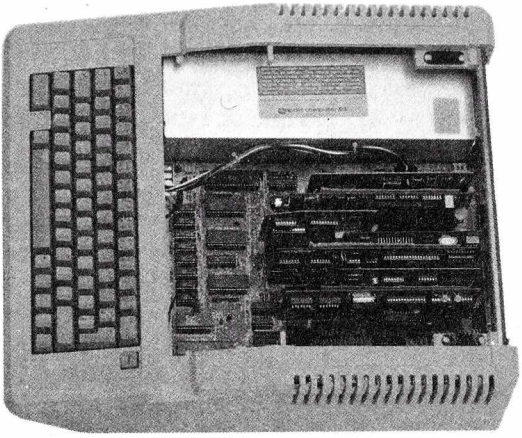


Foto 3 - Interior do Apple II, mostrando o cartão principal e a caixa da fonte de alimentação.

O cartão principal, às vezes o único e necessário para o funcionamento completo da máquina, costuma mostrar uma disposição de componentes que indica a diversidade de funções dos chips utilizados. Antes de tudo, o "cabeça de família", isto é, o microprocessador, sempre será facilmente identificável, pois é um circuito integrado grande, com uma cápsula que costuma ter 40 patilhas como é o caso do 6502 (microprocessador do Apple) ou o 8088 (microprocessador da IBM-PC). Nas máquinas mais modernas, tais como o "Mac" da Apple, o uP está contido em uma cápsula de 64 patilhas. Trata-se do 68000, muito mais potente que os outros dois e que, em conseqüência, precisa mais patilhas (pinos).

Mais adiante veremos o funcionamento e como está construído um microprocessador. Agora, basta dizer que o microprocessador recebe também a denominação de "Unidade Central de Processamento" (a CPU) porque sua missão é dirigir a execução dos programas e processar os dados introduzidos pelo usuário.

Ao redor da CPU encontram-se outros circuitos integrados, que costumam ser menores. Uma boa parte deles são de memória, atuando, em relação com a CPU, como uma espécie de "quarto de despejo" para os dados e as ordens em curso da execução. Finalmente, outros circuitos permitem conectar o cartão principal com outros cartões externos. Estes são conhecidos como "integrados de entrada/saída", pois controlam a entrada (e/ou saída) de dados para (ou desde) o cartão principal. Nos capítulos posteriores veremos com detalhes as funções de tais componentes e também quais são os dispositivos que nele podem ser conectados.

De modo "aglutinante" que permite unir os três tipos de circuitos integrados, anteriormente citados, existe no cartão uma ampla gama de outros integrados menores, cuja função é a de adaptar os sinais que fluem entre os circuitos integrados principais, facilitando assim sua conexão.

A necessidade destes pequenos circuitos auxiliares deriva de que tanto o microprocessador, a memória e os dispositivos de entrada/saída estão desenhados de forma que podem dar lugar a configurações muito distintas em suas necessidades e desenho.

Desta forma se tem a desejada versatilidade, que se torna onerosa, à medida que se vai acrescentando, em cada caso, um

maior número destes componentes auxiliares.

É o momento de pegar uma folha de papel e tentar desenhar a disposição do cartão principal, agrupando as funções existentes nos blocos lógicos. Assim obteremos, no esquema elétrico (que é algo sempre bastante complexo, variável de uma máquina para outra, e que não entra dentro dos objetivos deste livro), somente um diagrama de blocos funcional da parte eletrônica. Deste modo chegaremos ao que está ilustrado na figura 1. No retângulo que assinala os limites da placa principal estão contidos outros três retângulos que representam a cada uma das funções antes citadas: MEM identifica a memória e está conectada à CPU; E/S significa entrada/saída e também está conectada à CPU. As conexões são bidirecionais, o que quer dizer que o intercâmbio de informação pode ser produzido em ambos sentidos (desde e para a CPU). Certamente, uma fonte de alimentação fornece a energia necessária ao sistema. Observe que não aparece nenhuma referência ao suporte lógico e isto é devido a que, pelo menos aqui, não se faz necessário nem serve para nada em particular.

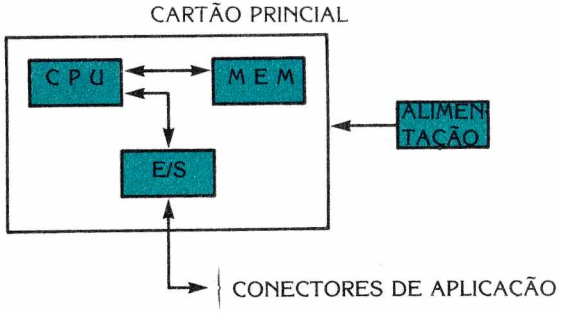


Fig. 1 - Blocos que constituem o cartão principal do computador pessoal.

Com o seguinte passo chegamos à figura 2. Nela a visão é muito mais ampla, pois o cartão principal mostra com mais detalhes seu conteúdo, ainda que, naturalmente, não sabemos qual é. Aparecem três unidades externas ao cartão, que em seguida identificaremos por menos que tenhamos trabalhado com nosso computador pessoal: abaixo temos um teclado, à direita uma unidade de apresentação e à esquerda uma memória de massa, indicada de forma simbólica por uma unidade de disco flexível.

O teclado é um acessório externo muito importante, já que, em princípio, é o único meio que o usuário tem para comunicar-se com o cartão principal proporcionando-lhe dados e ordens. Na figura 2 vemos que o sentido da conexão vai desde o teclado ao cartão principal. Conclusão: o teclado é um periférico e, mais concretamente, um periférico de entrada para o uP. Pelo contrário, a tela serve para o computador visualizar as respostas das nossas solicitações de forma "inteligível" pelo usuário, mediante frases ou figuras pré-programadas na memória do próprio computador.

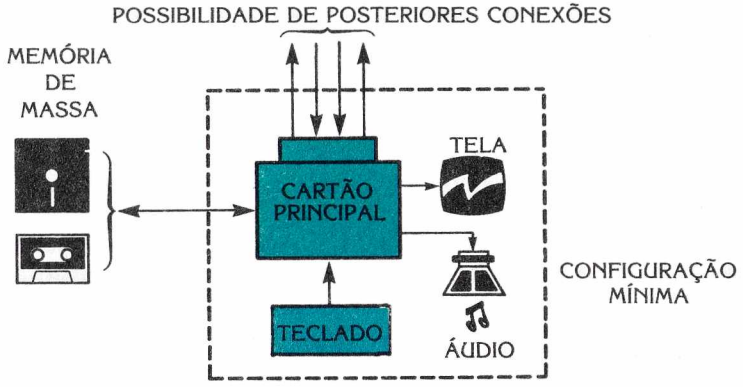


Fig. 2 - Estrutura de um computdor pessoal (configuração mínima e periféricos de memória de massa).

Assim, por exemplo, se batermos PRONT em lugar de PRINT obteremos a resposta: SYNTAX ERROR. A unidade de apresentação visual é, pois, um periférico exclusivamente de saída e o sentido de conexão vai desde o cartão principal até a própria unidade. Com frequência existe uma unidade de audio, reduzida a sua mínima expressão, mas suficiente para conseguir ao menos um assobio ("beep") que possa avisar ao programador, por acaso adormecido sobre o teclado às três da madrugada. É outro caso de periférico de saída.

O teclado, o monitor e o dispositivo de audio, juntos com o cartão principal, constituem o que costuma ser definido como "configuração mínima" de um computador pessoal. Também na figura 2 observamos uma linha de traços que rodeia estes elementos fundamentais, com o que indica-se de forma simbólica a caixa que os contêm. A unidade de memória de massa é um periférico importante. No entanto, e sobretudo para não aumentar o pre-

ço da configuração mínima, costuma-se oferecer como um acessório adicional. Apesar disto, computadores como o Apple II-c, o Macintosh, ou o IBM-PC podem ter todos eles exceto uma unidade de disco flexível incorporada na estrutura de base. A memória de massa, como seu nome indica, permite armazenar de modo permanente grandes quantidades de dados, que seria pouco adequado e demasiadamente custoso manter na memória interna hospedada, como vimos, no cartão principal. O armazenamento é produzido, sob ponto de vista conceitual, da mesma forma que quando gravamos em uma fita cassete uma música. De fato, nos computadores pessoais mais econômicos, são utilizados suportes magnéticos que são precisamente fitas de cassete. Mais adiante, no capítulo dedicado a estes periféricos analizaremos a fundo seu funcionamento.

Complementando o exame da figura 2, indicamos a existência de uma zona dedicada a conexãoes posteriores e que costumam denominar-se "back-panel". Na realidade, este painel contém os conectores de entrada, de saída e de entrada/saída disponíveis para futuras ampliações do computador pessoal. Na realização das conexões necessárias, podem intervir outras unidades periféricas e, inclusive, outros computadores, de modo que constitua-se uma rede de microprocessadores intercomunicados.

Intencionalmente indicamos os diversos tipos de conexões possíveis (unidirecionais e bidirecionais), úteis cada uma para determinadas classes de periféricos. Todas estas conexões, incluindo as que afetam as unidades descritas anteriormente, estão controladas por circuitos especializados de entrada/saída existentes no cartão principal, o qual torna-se mais importante ainda, se considerada a potência e a versatilidade desta parte do hardware antes da compra, pois um computador pessoal com escassos circuitos de entrada/saída não permitirá um controle ágil dos periféricos que a ele forem conectados, e limitará o número possível destes.

Personalização do hardware (as expansões)

Muitos e famosos computadores pessoais possuem um cartão principal preparado para admitir hardware adicional. Este apresenta-se sob a forma de cartão que é inserido nos conectores adequados (os famosos "slots") do cartão principal. Se este é o sistema empregado, costumam ser menos os dispositivos de entra-

da/saída inclusos na configuração básica. Desta forma ficamos limitados ao dispositivo de audio e à conexão para a unidade de fita de cassete. Ainda que possa parecer um sistema restrito, costuma ser, ao contrário, uma boa opção por parte do fabricante. Desse modo, o cartão principal e, por conseguinte, o computador pessoal na configuração básica, custa menos e o usuário não se vê obrigado também a pagar o que não necessita. Assim nasceu um florescente mercado de pequenos cartões de ampliação que entram, por direito próprio, dentro do conceito de hardware de um computador pessoal.

Naturalmente, entre tantos modelos disponíveis, é preciso orientar-se com bom critério para não correr o risco de adquirir um produto que não seja verdadeiramente compatível como o resto dos circuitos do cartão principal. Em tal finalidade, recorde que a inserção de um cartão em um encaixe equivale a ampliar, do ponto de vista eletrônico, o cartão principal, o que é quase como realizar modificações no circuito deste último. Por este motivo corremos o risco de deteriorar o núcleo da configuração principal, se inserirmos um cartão inadequado.

Deixando de lado alguns cartões "piratas", é conveniente dirigir-se aos representantes autorizados pelo fabricante, quando não for este último quem coloca à venda tais cartões adicionais. Quais deles são verdadeiramente necessários? A resposta depende exclusivamente de nossas previsões de trabalho: existem cartões de E/S em paralelo e em série (veremos mais adiante o que isto significa), para conectar uma impressora (talvez o primeiro periférico em importância, se descontarmos a presença de uma memória de massa), cartões de controle da unidade de apresentação visual (as dos gráficos de alta resolução e as coloridas estão na ordem do dia); cartões de controle para todas as classes de memórias de massa (discos flexíveis e discos rígidos) nos diversos formatos existentes; cartões para comunicações, para controle de sinais, etc. As regras de ouro aplicáveis sempre que um cartão de ampliação for adquirido, são as seguintes 1) o cartão deve ser original para assegurar a existência de componentes de qualidade; 2) não deve trazer problemas em seu contrôle e, na prática, deve funcionar de forma imediata uma vez inserido em seu encaixe (slot), sem ter que efetuar ações complicadas para sua ativação cada vez que for conectado ao computador, e 3) não deve absorver uma potência superior a que põe à sua disposição a fonte de alimentação do sistema.

Esta última regra é importante porque facilmente corre-se o risco de sobrecarregar a alimentação do próprio computador pes-

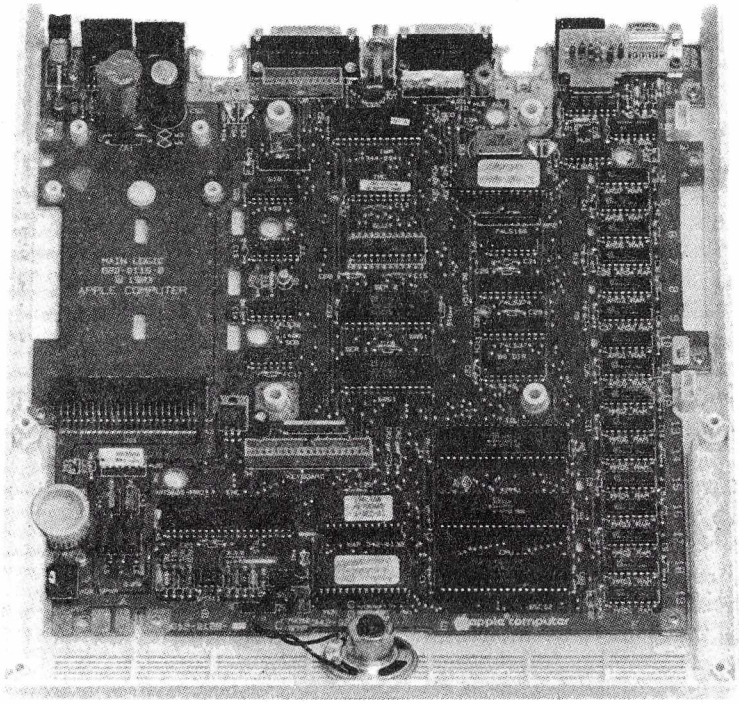


Foto 4 - Interior do Apple II-c.

soal (posto que nos cartões adicionais também tem um número considerável de integrados que consomem corrente), que encontrar-se-á então, em dificuldades. Sem estas condições, e produzida uma pequena falha (caída) na rede, todo o trabalho realizado até o momento perder-se-á pois a alimentação não logrará em subadministrar a energia necessária. À parte destas normas obrigatórias com respeito ao hardware, o usuário poderá escolher as opções adicionais que desejar, ainda que as mais comuns sejam: memória suplementar e interfaces (sistemas de interconexão) para impressora, vídeo e unidade de disco (flexível ou rígido).

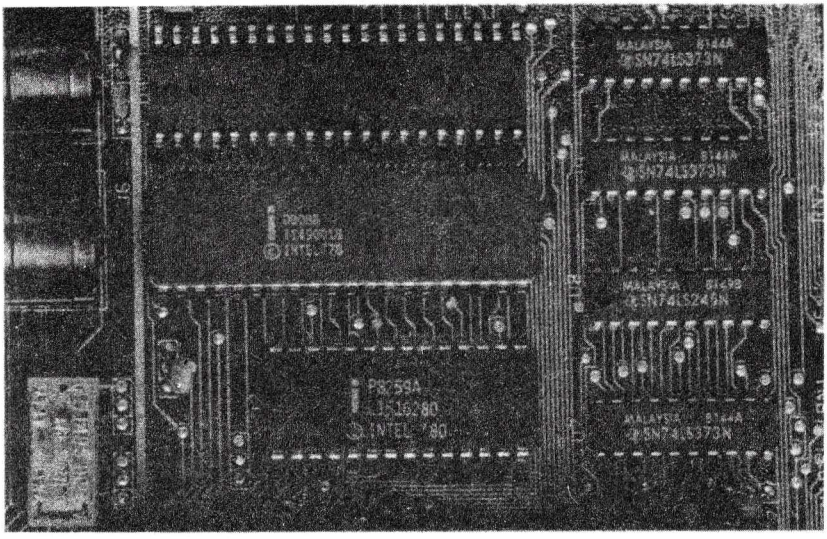


Foto 5 - Placa principal do IBM-PC

Outros computadores, que poderíamos chamar "fechados", não têm encaixes (slots) no cartão principal, mas incluem nele todos as interfaces clássicas citadas. Em tal caso, o cartão principal custará mais, mas o fato de ser dispensado de conectores e de espaço para os cartões adicionais proporcionará um conjunto mais compacto, e à miúdo, o custo total do computador pessoal será menor. Este é o caso das opções portáteis. Na fotografia 4 é mostrado o aspecto interno do Apple II-c; na 5 o de um IBM-PC, computadores pessoais clássicos, e na fotografia 6 aparecem alguns computadores pessoais portáteis.

Faça o estudo dos chips

Até agora não exigimos grandes esforços, apenas nos limitamos a abrir o computador pessoal e a explicar seu conteúdo em grandes passadas.

Neste ponto, podemos "visualizar" a arquitetura como um diagrama de blocos formado pelos diversos componentes que constituem o hardware, e, temos também, alguns critérios de escolha

Foto 6 - Exemplos de computadores pessoais portáteis.

essenciais, tais como a robustez, a facilidade de manuseio, durabilidade, fácil manutenção e capacidade de ampliação.

Para aprofundar-se de um modo mais amplo no hardware é necessário conhecer os conceitos funcionais de um computador, partindo daqueles que serviram de base no desenho dos próprios circuitos integrados. Veremos logo como a teoria digital é a base de todos os chips presentes em um computador pessoal, assim como do funcionamento da CPU e a forma em que logra "animar" o cartão principal para que este, por sua vez, "dê vida" ao nosso computador pessoal.

SISTEMA: Indica-se com este termo genérico a configuração constituída pelo computador pessoal com seus periféricos conectados (impressoras, discos, etc).

DISQUETE (DISCO FLEXÍVEL): meio de armazenamento massivo permanente, constituído por um disco de material magnetizável protegido dentro de um invólucro de plástico. Uma fenda no citado invólucro protetor permite que uma cabeça de gravação introduza os dados no disco. O disco flexível, universalmente denominado "floppy" ("mole"), necessita de um dispositivo, conhecido como "unidade de disco flexível", onde é inserido, para poder ser objeto de leitura ou "escrita", por parte do computador.

MONITOR: É o terminal de imagem usual do computador. Em uma apresentação que parece a tela de uma televisão aparecem nomes, dados, ordens e gráficos, tal como é introduzido a partir do teclado pelo usuário, assim como as respostas que proporciona o computador.

CARACTER ALFANUMÉRICO: Trata-se de uma letra de A a Z, um número ou qualquer caracter simbólico como: *, .:;$%&, etc.

VISOR ("DISPLAY") DE CRISTAL LÍQUIDO: Painel plano e delgado no qual a tela de apresentação visual (isto é, o tubo de raios catódicos) está substituída por uma placa de cristal líquido. A vantagem (frente a um custo muito mais elevado) é o consumo de corrente, praticamente nulo, que o faz aconselhável para os computadores portáteis.

PERIFÉRICO: Trata-se de um subsistema, habitualmente controlado por microprocessador, que é conectado ao computador pessoal mas que está fisicamente separado. Um exemplo clássico é a impressora.

SLOT (ENCAIXE): Conector situado no cartão principal no qual pode ser inserido um cartão de hardware adicional, também denominado de "expansão".

BACK-PANEL: Painel posterior do computador pessoal, que serve de receptáculo aos cabos e às conexões desde e para os periféricos. Os conectores que aparecem em "back-panel" (painel posterior) estão conectados, no interior do computador pessoal, com os circuitos de E/S (entrada/saída) que fazem parte integrante do cartão principal ou dos ocasionais cartões de expansão.

PATILHAS: Fios achatados de metal. No caso dos circuitos integrados estes fios são utilizados como contatos externos (pinos) que se encaixam às bases das cápsulas fixadas nas placas de circuito impresso.

CAPÍTULO II

ESTRUTURA E FUNCIONAMENTO DA CPU

Amigo leitor, não se assuste ao ler o título das primeiras divisões do capítulo. Nossa intenção não é a de esmagar-lhe com toda a história da microeletrônica, desde a invenção do transistor até o microprocessador, mas somente fazê-lo compreender melhor como milhares de transistores, integrados em um só chip de silício, podem cooperar no estabelecimento das características de potência e cálculo de uma CPU, "coração" do nosso computador pessoal.

Fundamentos da eletrônica digital

Um transistor é em essência um interruptor controlado eletricamente. Na figura 1a é mostrado um circuito elementar com um só transistor que, na figura 1b, está disposto com o terminal de entrada I a um potencial positivo. Sem aprofundar-nos na física dos semicondutores, vamos partir da hipótese de que na situação mostrada, o transistor comporta-se como um interruptor fechado e o terminal U está em tensão zero. O caso oposto é o da figura 1c, com o terminal I colocado em potencial 0 (zero), que é como se o interruptor estivesse aberto, com o terminal U ao potencial positivo + V. Assim pois, sempre estaremos manuseando duas situações (estados) distintas: 0 (zero) por um lado e + V por

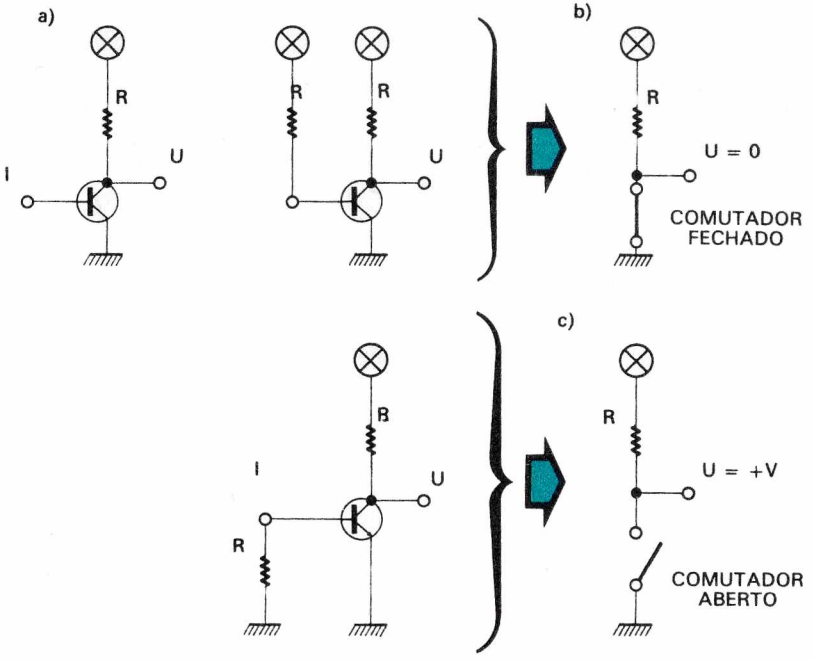


Fig. 1 - Um transistor (como o da figura a) é uma espécie de interruptor: se aplicarmos um sinal positivo à sua entrada de base, a saída passará a 0 Volts (b), enquanto que se a entrada estiver conectada à massa, a saída estará + V(c).

outro. É portanto um fucionamento binário (de dois estados).

Estas breves considerações fazem-nos refletir sobre o princípio fundamental básico do funcionamento de qualquer computador, do menor ao maior.

Qualquer ação realizada pelos circuitos que constituem a máquina, sempre será um tratamento mais ou menos complexo de sinais binários. Em poucas palavras, toda a eletrônica digital está baseada em comutações de tensão do tipo "aceso/apagado" ou "presença/ausência de tensão de um fio". Todos os dados tratados no computador são única e exclusivamente impulsos de tensão. Os denominados circuitos "lógicos", somente podem distingüir os estados de presença ou ausência de tensão em suas entradas e em função dos ditos estados dar um valor binário à sua saída.

Inclusive o mais complexo dos circuitos digitais podem decompor-se em blocos secundários cada vez mais simples, até sua "dissecação" completa em funções lógicas elementares. Estas últimas são as famosas NOT, AND e OR. Pelo que diz respeito ao

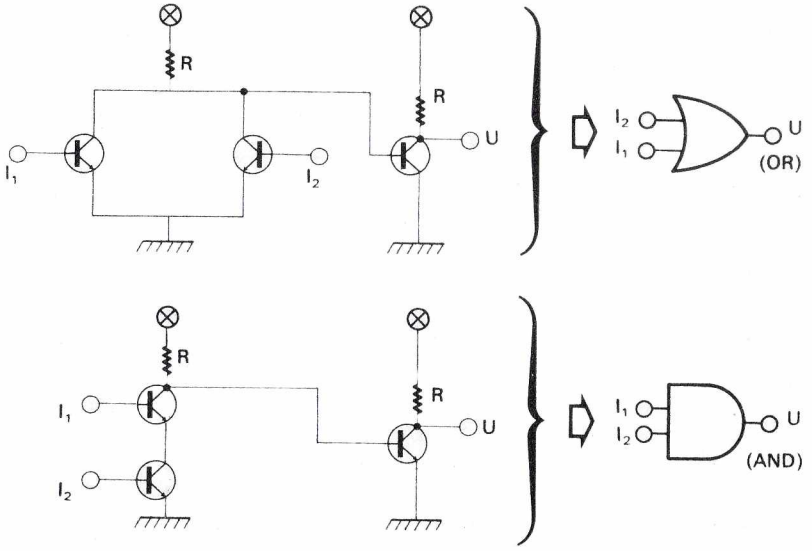


Fig. 2 - Funções lógicas simples (OR e AND) feitas com transistores.

hardware os nomes destas funções vão associadas às "portas" (conjunto de transistores e resistências) que as originam. Resulta o mesmo, por exemplo, falar de OR ou de uma porta OR. Observe a figura 2. Com ela pretendemos satisfazer a curiosidade daqueles leitores que podem perguntar-se como poderia obter-se, com simples transistores como o da figura 1, um circuito lógico do dipo AND ou OR. Se nos fixarmos também na figura 1 veremos que o que ela mostra é a realização de uma porta NOT. A figura 3 ilustra, com clássicos exemplos de eletricidade, o comportamento das funções AND e OR. O circuito AND acende a lâmpada se, e somente se, os 2 interruptores estiverem fechados, isto é, estão a nível de tensão positiva, o que falando em termos digitais equivale a dizer que as duas entradas são "verdadeiras", ou também que estão a "nível lógico 1" (veja a tabela 1 para esclarecer estes termos). Basta que uma só das duas entradas esteja ao nível lógico "0" para ter à saída um nível lógico "0". Pelo contrário, a função OR mostra que basta um só interruptor fechado ("1") para ter a lâmpada acesa e as duas entradas devem estar em "0" para ter também "0" à saída.

O emprego de "1" e "0" para indicar que uma entrada está a um potencial + V (+ V é o valor genérico da tensão de alimentação, que costuma ser de + 5V) ou em potencial 0 é algo característico da lógica digital, porque simplifica o tratamento e é de com-

interruptor	tensão	valor lógico	nível lógico
fechado	positiva	verdadeiro	alto (1)
aberto	negativa	falso	baixo (0)

Tabela 1 - Equivalência das distintas terminologias em um sistema binário.

preensão imediata. No sucessivo, seguiremos sempre dita nomenclatura.

A figura 4 mostra as denominadas "tabelas verdade" das funções AND, OR e NOT. Estas tabelas representam todas as combinações possíveis das entradas de porta, e o resultado correspondente na saída. Debaixo de cada uma das tabelas está desenhado o símbolo lógico da porta correspondente. Isto costuma ser empregado em todos os esquemas digitais, como se vê na figura 5a que, a título de curiosidade, ilustra o esquema de algumas funções lógicas complexas que são realizáveis com as portas que acabamos de descrever. Estas portas não apresentam-se sozinhas, mas várias agrupadas em um só chip, com características normalizadas (circuitos integrados TTL).

Outros exemplos de blocos lógicos mais complicados podemos encontrar, por exemplo, no manual técnico ("Hardware Manual") de seu próprio computador pessoal.

O percurso a realizar para chegar ao microprocessador a partir das portas lógicas OR, AND e NOT é bem mais longo e não dispomos de muito espaço para abordá-lo. Consideramos suficien-

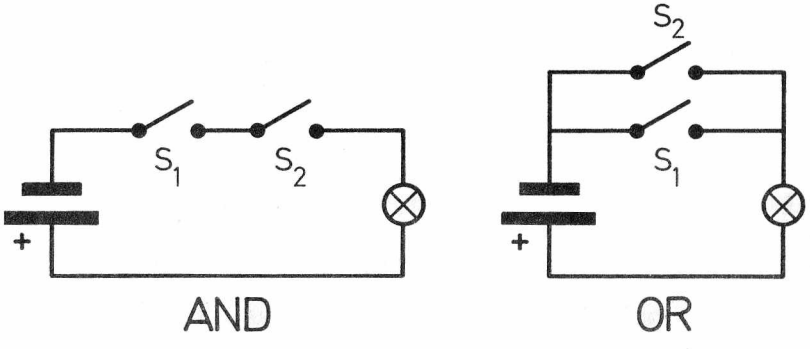


Fig. 3 - Representação funcional com interruptores das funções lógicas AND e OR.

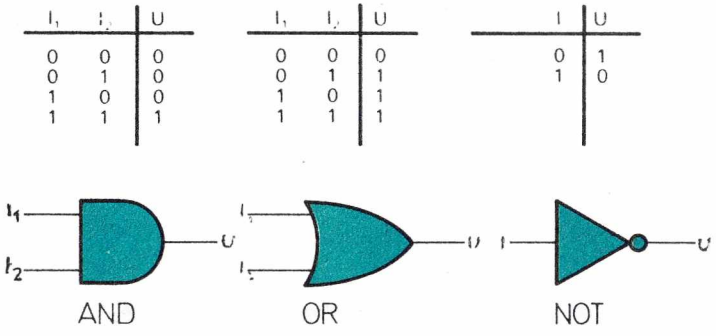


Fig. 4 - Tabelas verdade das funções lógicas AND, OR e NOT.

te ter adquirido pelo menos estes conceitos básicos. Se você desejar aprofundar-se mais nestes conceitos, recomendamos que consulte os livros da bibliografia ou as revistas especializadas.

LSI: A tecnologia que permitiu a informática pessoal

Se, como todos sabemos, um transistor é um "bicho" pequeno mas "visível" imagine o espaço que ocupariam cinqüenta mil deles. Afortunadamente, a tecnologia dos últimos quinze anos nos tem dado a mão e, com a integração cada vez mais desenvolvida, tornou-se possível a realização de dispositivos muito complexos, mantendo dimensões muito reduzidas.

Como é possível? A resposta já está diante de nossos olhos, no cartão principal do computador pessoal que acabamos de abrir e que está cheio desses dispositivos. Cada um desses circuitos, denominados "integrados"(C.I.), é uma pastilha, seja de plástico ou de material cerâmico, que contém uma diminuta placa de silício, material semimetálico (ou semicondutor) com o qual estão construídos os transistores. O curioso desses transistores reside no fato de que em lugar de utilizar fragmento de silício para cada transistor e conectá-los todos juntos externamente, utiliza-se uma placa fotografada que contém a disposição de todos os transistores, inclusive as conexões correspondentes. É algo assim como o gravador que obtém uma mesa artística a partir de um só fragmento de madeira, representando todas as figuras juntas, em lugar de realizar as cenas em separado, uma a uma e logo uní-las. No processo de

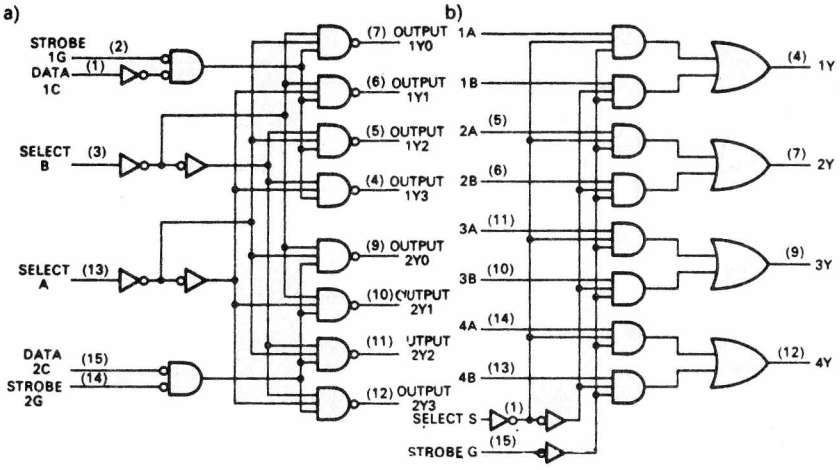


Fig. 5 - Alguns exemplos de circuitos digitais complexos, feitos com circuitos integrados TTL.

desenho dos circuitos integrados, umas máquinas especiais "traduzem" os esquemas lógicos (realizados em papel com os elementos AND-OR-NOT antes considerados) a seus esquemas elétricos (baseados em transistores) e estes a traçados adequados que serão praticados no chip de silício com um gravador fotolitográfico. A vantagem é enorme, posto que o espaço que ocupam os elementos ativos se reduz, por meios fotográficos, a dimensões de ordem das micras e isto torna possível a integração de milhares de funções lógicas. Para as portas simples que examinamos anteriormente, usa-se a integração em baixa escala (SSI = Small Scale Integration), posto que não são muitos circuitos os que serão colocados no chip. No cartão principal do computador pessoal as portas lógicas costumam estar incluídas na forma de circuitos integrados de 14 patilhas. Para as funções complexas, características dos circuitos integrados maiores, emprega-se uma integração a média escala (MSI) que permite entre 10 a 1000 portas por chip e os integrados mais complexos, grandes e custosos, tais como as memórias, os controladores de E/S e, naturalmente, a própria CPU, são fabricados com técnicas de integração em grande escala ou em maiores escalas (LSI e VLSI, respectivamente) que permitem até 10.000 portas lógicas para a 1.ª e mais para a 2.ª.

Um detalhe que deve ser levado em conta é que as dimensões das cápsulas não dependem tanto da complexidade do chip, mas do número de funções que necessitem comunicar-se com o exte-

rior como entradas ou saídas, isto é, do ponto de vista prático o tamanho depende do número de patilhas (pinos) necessário. Um microprocessador costuma estar confinado em uma cápsula de 40 patilhas, porque é elevado o número de sinais que deve controlar, tanto de entrada como de saída (brevemente veremos como o fazem). Na figura 6 pode-se ver a fotografia do chip de silício no qual está integrado o conhecido microprocessador Z80. Na realidade, a placa mede aproximadamente meio centímetro de lado.

Como funciona um microprocessador?

Agora que estamos familizarizados com as funções lógicas fundamentais e que nos introduzimos na tecnologia de fabricação dos dispositivos digitais, podemos pensar na construção da nossa CPU. O "truque" consiste em esquecermos das portas AND, OR e NOT, fazendo o raciocínio seguinte: "utilizaremos somente blocos lógicos cujo nome coincidirá com o da função que realizarão e formados pela união de tantas portas elementares quantas forem necessárias".

Talvez não seja um método habitual construir-se uma máquina para compreender como funciona mas, neste caso, veremos como o estar obrigados a pensar nos dispositivos necessários na CPU, nos ajudará muito à compreensão do funcionamento do conjunto.

Com este pensamento prévio, desenhamos o que aparece na figura 7a. O retângulo define os limites do microprocessador, que agora está vazio, mas iremos preenchendo-o pouco a pouco.

Começaremos por examinar as conexões externas, que serão traduzidas em outros tantos terminais ou patilhas. Pegamos oito fios de conexão e os agrupamos, criando o denominado "Bus de dados" (em inglês, "Data Bus"). Através do bus de dados poderão chegar à CPU sinais (dados) procedentes dos dispositivos que conectar-lhe-emos e, reciprocamente, a CPU poderá, através dele emitir sinais.

Na figura 7b ilustra-se, esquematicamente, a transmissão de um dado no bus. Para uma maior simplicidade, consideramos que o bus tem somente dois fios, mas a explicação pode estender-se a oito fios sem problema algum. Pois bem, o microprocessador seria o homem que está no recinto A, e o dispositivo externo conectado ao microprocessador, que pode ser RAM, ROM ou E/S, é o que está no recinto B. Cada linha pode dar tensão a uma lâmpada do recinto B, cuja chave está controlada desde o recinto A, isto é, desde o microprocessador. As referências de massa são comuns, tal como sucede na realidade. Se o microprocessador(μP)

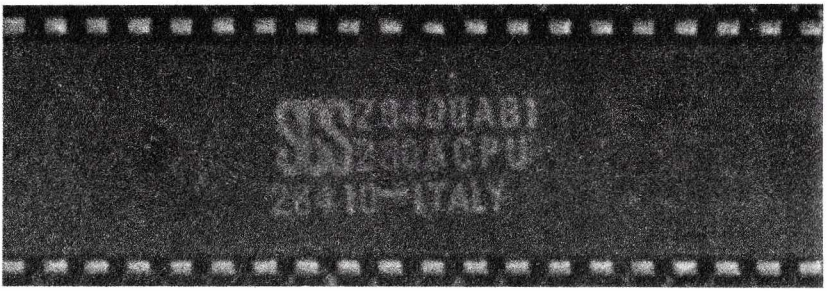
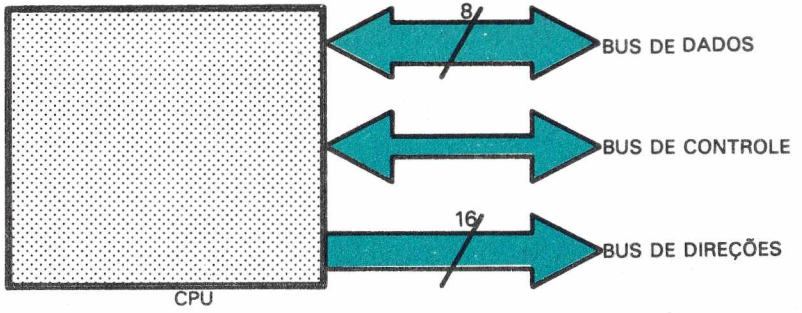


Fig. 6 - Fotografia do chip de silício que constitui o microprocessador Z80.

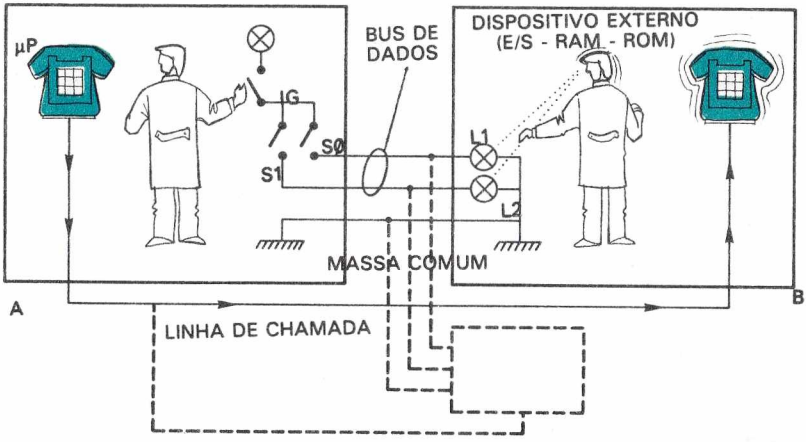
quer enviar um dado ao dispositivo, mediante uma operação que denomina-se "escritura do dado", o dispositivo deverá, antes de tudo, ser avisado a fim de que esteja preparado para receber o dado. Esta operação de aviso realiza-se com uma linha adicional, representada em nossa ilustração pelo telefone.

O telefone está tocando em B e este encontra-se preparado para receber o dado, operação que, em nosso caso, consiste em observar o estado das lâmpadas enquanto a chamada continua sendo produzida (isto é, basta deixar o telefone tocar). Agora A deverá posicionar, à sua escolha, os interruptores de dados S0 e S1 e, logo, acionar o interruptor geral IG para dar tensão simultaneamente a todos os interruptores. De forma imediata, B "verá" uma das quatro combinações possíveis de "aceso/apagado" das duas lâmpadas e tomará imediatamente nota do dito estado porque, transcorrido um curto período de tempo, A cortará a chamada e deixará livre a B para dedicar-se a outra atividade.

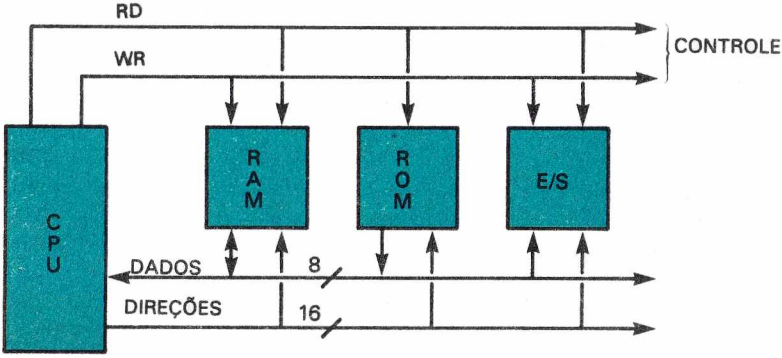
Naturalmente, nada impede ao microprocessador chamar imediatamente depois a outros dispositivos conectados (indicados na figura com linhas tracejadas) ou voltar a chamar ao próprio B. Se prestou atenção até aqui sentiu-se "chocado" que no exemplo da figura 7b mostre-se um bus de dados unidirecional. Pelo contrário, na realidade, o bus de dados é bidirecional e isto significa que também o dispositivo B tem seus interruptores de pré-escolha e A suas lâmpadas de visualização do dado. A única limitação a esta "bidirecionalidade" está em que, inclusive para ler um dado, será sempre o microprocessador quem chamará ao dispositivo externo, quem controlará todas as transferências e, em qualquer caso, quem terá sob seu controle todo o sistema eletrônico. Como sabe, o dispositivo externo, no momento em que é chama-


8
BUS DE DADOS
BUS DE CONTROLE
16
BUS DE DIREÇÕES
CPU

a)


μP
G
SØ
S1
BUS DE DADOS
DISPOSITIVO EXTERNO
(E/S - RAM - ROM)
L1
L2
MASSA COMUM
A
B
LINHA DE CHAMADA

b)


RD
WR
CONTROLE
CPU
RAM
ROM
E/S
DADOS
8
DIREÇÕES
16

c)

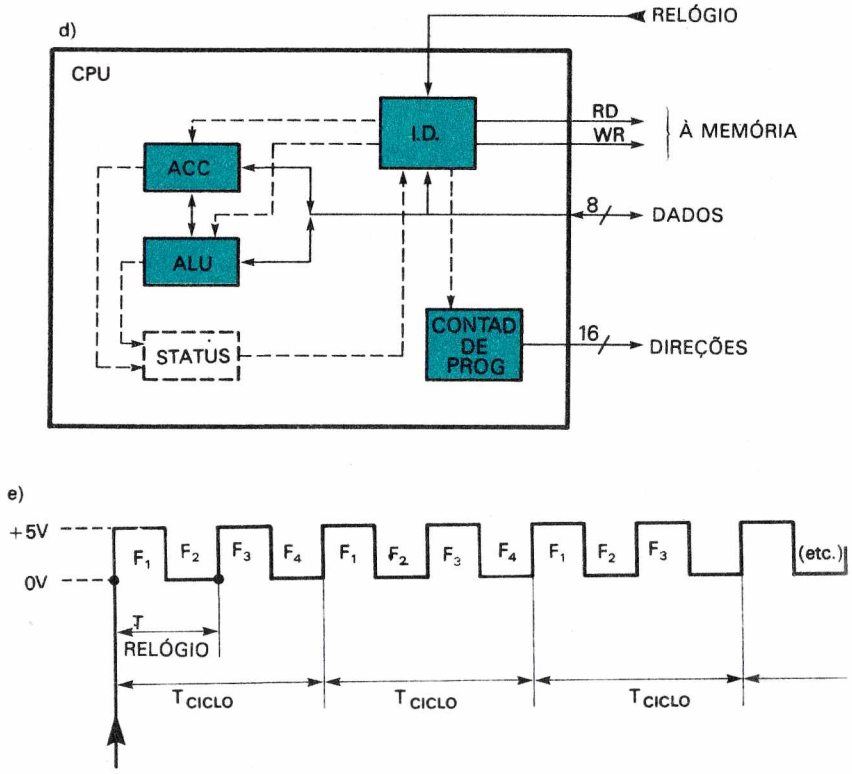


Fig. 7 (a, b, c, d e e) - Construção de uma CPU a partir das funções necessárias e dos principais blocos internos.

do, se a CPU solicita uma leitura ou uma escrita? Simplesmente mediante duas linhas de controle, denominadas RD e WR (que correspondem, respectivamente, à petição ou habilitação de leitura e escrita). A CPU, enquanto avisa a um dispositivo, põe uma das duas linhas a "1", o dispositivo saberá então o que deve fazer: se proporcionar o dado que tem a CPU para que este o leia ou admitir o presente no bus de dados para uma escrita. As linhas de controle estão normalmente agrupadas, como os dados, no bus chamado Bus de Control.

Quando existem inúmeros dispositivos conectados, resultaria incômodo e pouco prático usar uma linha individual no micro-

processador para cada dispositivo. Voltando ao exemplo anterior, podemos fazer que cada dispositivo tenha seu número de telefone, distinto dos demais. O número de telefone denomina-se "direção" e esta é a razão pelo qual já na figura 7a, desenhamos outros 16 fios à saída do microprocessador. Estes formam outro Bus que denomina-se "Bus de direção" (em inglês "Address Bus").

Quando o microprocessador chama a um dispositivo, para escrever ou ler um dado, emite o código desse dispositivo, através das 16 linhas do Bus de direções que estão conectadas com todos os dispositivos externos. O código é lido por todos os dispositivos, mas somente um "dar-se-á por aludido" ao coincidir o valor transmitido com seu "número".

Façamos algumas contas: Por uma linha podemos transmitir um "0" (lâmpada apagada) ou um "1" (lâmpada acesa); em duas linhas, as combinações são 00, 01, 10 e 11; em três linhas, desde 000 à 111, temos oito combinações, e assim sucessivamente. Portanto o número de combinações possíveis depende (na razão da potência de 2) das linhas disponíveis. Assim, com 8 linhas teremos $2^8 = 256$ combinações diversas e em 16 linhas poderemos transmitir até $2^{16} = 65536$ códigos diferentes. Este será, pois, o número máximo de dispositivos que podem ser conectados externamente à CPU.

Na realidade, por razões de economia e rapidez, muitos dos circuitos integrados do mercado (as memórias, por exemplo) contém em si milhares de dispositivos simples, tornando mais simples configurar portanto o circuito.

O μP é todavia uma caixa vazia, mas já sabemos que deve estar conectada ao exterior mediante linhas dispostas em grupos (Buses).

O bus de dados permite o intercâmbio recíproco (bidirecional de informação, através de 8 fios, entre o microprocessador e os dispositivos externos. O bus de direções sai do microprocessador, é unidirecional e serve para que o microprocessador emita o código de dispositivo com o qual quer dialogar, denominando-se dito código direção. Finalmente, as linhas do bus de controle servem para que os dispositivos identifiquem o tipo de operação que a CPU quer desenvolver.

Uma observação: as transferências de dados desde e para a CPU e a transmissão das direções, desde a CPU, realizam-se em paralelo, isto é, todas as linhas assumem simultaneamente a configuração pré-estabelecida de "uns" e de "zeros" (você recorda o interruptor geral do exemplo anterior?), com o que evitam-se confusões e atrasos.

Em informática é dito que uma informação necessita (ou "é" de) n bits quando para transmití-la são necessárias n linhas. Assim o Bus de dados é de 8 bits e o de direções de 16. A palavra bit provém de Binary digIT (dígito binário) e está universalmente aceita. Igualmente o está o termo byte que refere-se a um conjunto de 8 bits (o Bus de dados será de 1 byte e o de direções de dois).

A figura 7c representa a arquitetura genérica de um computador construído sobre a base da CPU. Os dois buses chegam a todos os dispositivos e o Bus de dados é bidirecional, enquanto que o de direções não o é. Observe que também as linhas RD e WR estão conectadas em paralelo a todos os dispositivos, ainda que um deles receba somente a linha de leitura RD (trata-se da memória ROM e mais adiante veremos a razão dele).

Com o que sabemos podemos acrescentar algumas coisas ao interior da caixa vazia. Na figura 7d mostram-se cinco blocos, o "decodificador de instruções" (denominado ID na figura), um contador de 16 bits que é o contador de programa ou contador de passos de programa, um dispositivo denominado "acumulador" e uma unidade denominado "ALU" que é a abreviação inglesa de "Unidade Aritmético-Lógica". O último bloco é um registro chamado de "estado" (status).

O ID (decodificador de instruções) é verdadeiramente importante pois controla o funcionamento interno do microprocessador. O bus de dados que está conectado ao microprocessador chega também ao ID, mas apenas em um sentido, pois o ID somente pode ler o código de 8 bits que chega procedente do bus de dados e não escrever algo neste. Vimos anteriormente que com oito linhas se pode representar 256 combinações de "aceso/apagado" ou de 1/0, desde 00000000 à 11111111. Estes 256 códigos são utilizados para enviar ordens "codificadas" desde o exterior ao nosso ID.

Essencialmente, o ID é um bloco lógico capaz de distingüir cada combinação individual dentre as 256 possíveis e convertê-la em uma seqüência precisa de ordens para os demais blocos existentes no microprocessador. Quanto mais complexo, versátil e podemos dizer que "virtuoso", for o ID, tanto melhor funcionará o microprocessador. O fabricante deste, fornece normalmente um manual de programação na linguagem de máquina, que não é outra coisa que uma tabela que associa cada um dos códigos que podemos enviar ao microprocessador com seu efeito dentro da CPU.

Exemplo: o manual da CPU 6502 indica que para o código 10101001, o ID obrigará o acumulador a guardar o dado que apareça imediatamente depois no bus de dados.

O contador de programa (PC = Program Counter) tem a fun-

ção de escrever no bus de direções um código de 16 bits que abrirá o acesso a um dispositivo determinado. O PC não emite estes códigos de direções de forma aleatória, segue somente as ordens do ID, que lhe comunica no momento oportuno qual será a direção de partida e quando e em que magnitude deverá variar citada direção. Assim mesmo recebe também a ordem de quando escrever seu conteúdo no bus de direções.

O acumulador é um bloco bastante simples: consiste em uma memória que lê os dados procedentes do bus e os armazena para que possam ser utilizados em operações posteriores. Estes tipos de memórias simples que fazem parte da CPU denominam-se "registros" da CPU. Em nosso microprocessador-exemplo somente existe um registro, mas na realidade são vários os registros que contém a CPU, o que aumenta sua versatilidade e potência.
Finalmente, somente nos resta o bloco denominado ALU. No ALU realizam-se "coisas de loucos" com os "uns" e "zeros" do dado que chega pelo bus desde o exterior, ou melhor, desde o acumulador: os "uns" podem transformar-se "sadicamente" em "zeros" e vice-versa; duas combinações de 8 bits podem somar-se ou diminuir-se segundo as regras da Álgebra de Boole ou da Aritmética; um dado pode comparar-se com o conteúdo no acumulador, etc.

Naturalmente, também neste caso todas as operações aritméticas (somas, diminuições, etc) ou lógicas (comparações, comutações 0/1, etc) estão estreitamente supervisionadas pelo ID que, dentro da CPU, tem um grande trabalho a fazer.

O sinal do relógio (clock)

As tarefas realizadas por todos os blocos internos da CPU, sob o controle do ID, sincronizam-se desde o exterior com um rítmo imposto por ele denominado "relógio do sistema" (em inglês, "clock") Trata-se de um sinal que é empregado como padrão de tempo e sincroniza todas as operações realizadas pelos diversos blocos integrados na CPU. Na mesma figura 7d, o sinal do relógio chega ao ID e este atua como se estivesse na mão de um cronômetro com o qual controlará a sincronização das fases de trabalho de cada bloco supervisado. Um sinal de relógio clássico é uma tensão que varia, de forma cíclica, entre 0 e + 5V, com uma forma de onda quadrada, como na figura 7e. Sua freqüência é calculada observando-se quantas vezes por segundo repete-se a forma básica denominada "ciclo" e representada por "Tclock" na figura 7e. Se o relógio tem uma freqüência de 4 MHz, em cada segundo

proporcionará quatro milhões de ciclos, pelo que um ciclo durará exatamente 1/4.000.000 = 250.10^{-9} seg. = 250 ns (de nano segundos = mil milionésimos de segundo). Quanto mais elevada é a freqüência do relógio, maior é a velocidade de execução, pois o ciclo durará menos.

Funcionamento interno da CPU durante um ciclo

Agora sabemos que o relógio fornece uma temporização de referência. Vejamos como se desenvolve o trabalho durante um "ciclo de máquina" (não tem porque coincidir com um "ciclo de relógio"). Vamos examinar as figuras 7d e 7e: a primeira para as conexões entre os blocos da CPU e para o exterior, e a segunda para o relógio. Observará que consideramos os ciclos contíguos de relógio, chamando F1, F2, F3 e F4 aos quatro semiciclos consecutivos. O ciclo de máquina é "Tcycle" e dura, em nosso exemplo, o dobro do período de relógio, o que significa 500 ns para um relógio de 4 MHz.

Fixemos um começo para a base de tempos onde supomos que o microprocessdor toma "vida" e inicia o trabalho. Chamamos "I" este instante inicial, que coincide com o começo da primeira fase, F1. O ID estabelece imediatamente o contador de programa com a direção de partida, cujo valor característico varia de uma CPU para outra (na 6502 seria a 65.529). Na fase F2, o ID carrega o valor do contador de programa no bus de direções e, ao mesmo tempo, ativa a linha RD para solicitar uma leitura do dispositivo selecionado. Este último, reconhecida sua direção no bus, reage imediatamente e emprega o período F3 para encontrar o dado solicitado. No F4, o dado é situado pelo dispositivo no bus de dados e lido pelo ID.

Este dado procedente do exterior é decodificado pelo ID para saber que operação deve executar; na fase F1 do ciclo de máquina seguinte começa a controlar todas as operações internas relacionadas com essa execução em particular. Pode ocorrer o caso em que bastam dois ciclos de máquina para completar todas as operações, então, o terceiro ciclo será na realidade o primeiro dedicado à seguinte operação. Para muitas operações no entanto, não bastam dois ciclos de máquina, podendo-se precisar de três, quatro e inclusive mais. Em qualquer caso, sempre é o primeiro ciclo de máquina o dedicado à aquisição do dado chave para o ID. Tal dado toma o nome de Código de Operação ("OP.CODE") e serve para selecionar a série de operações internas necessárias. Em continuação, fizemos uma pequena lista de algumas operações co-

muns em todas as CPU's:

- Carregamento imediato de um valor em acumulador (o valor a carregar faz parte da própria operação).
- Deslocar um dado desde um dispositivo externo a outro (de uma direção a outra).
- Somar um valor que chega desde o externo a um segundo dado já existente no acumulador.
- Comparar um dado que chega desde o externo com um dado já existente no acumulador.

Proporcionar uma instrução ao microprocessador significa, antes de tudo, enviar o código de operação que a define. Para completar a instrução devemos fornecer todos os dados aos que faz referência enviando-os imediatamente depois do código de operação. Estes dados que completam a instrução são os denominados OPERANDOS. Em definitivo, uma instrução genérica de μP tem o formato seguinte:

CÓDIGO DE OPERAÇÃO [OPERANDO] . . . [OPERANDO]

onde o código de operação é a única parte sempre necessária.

Deixamos o ID após o primeiro ciclo de máquina preparado para controlar a execução da instrução especificada pelo código de operação enviado. Neste ponto, podemos seguir a execução de algumas das simples instruções antes citadas.

- Primeiro caso: "carregamento imediato no acumulador do dado que segue".

No primeiro ciclo de máquina, já examinado, o ID compreendeu a instrução e no ciclo seguinte sua tarefa consistirá em: ordenar ao contador de programa que emita uma nova direção igual à anterior + 1 (durante F1); ativar, em F2 a linha de controle RD para uma leitura do dispositivo que deve proporcionar o dado (F2); em F3 selecionar o dispositivo e depositar o dado no bus e em F4 carregá-lo no interior do acumulador que, entretanto, foi avisado pelo próprio ID. Conclusão: a instrução completa-se em dois ciclos de máquina.

- Segundo caso: "Somar ao conteúdo do acumulador o dado que segue imediatamente (o resultado cai no acumulador)".

Depois do primeiro ciclo de máquina, segue um segundo ciclo idêntico ao comentado pela instrução antes considerada, com a desculpa de que o dado que chega não se carrega no acumulador, e sim no ALU e que em F4, o ID ordena ao ALU que some seu conteúdo com o do acumulador. Ao final de F4 o resultado cai no acumulador.

Recapitulando: A execução de cada instrução está pois controlada pelo bloco do "decodificador de instruções", que precisa um código chave para identificar cada instrução.

Tal código chama-se "código de operação" e vimos que normalmente é seguido por outros dados, denominados operandos, que completam a instrução.

Quando conectamos a CPU, temos que proporcionar-lhe imediatamente um código de operação executável. Finalmente, vimos como uma instrução é levada e cabo em vários ciclos de máquinas; uma vez terminada sua execução a CPU pode iniciar imediatamente a seguinte.

Caráter sequencial da execução de operações pela CPU

Por tudo que vimos até agora, parece claro que a CPU é capaz de executar todas as instruções na condição de que seja respeitada sua natureza sequencial: a CPU executa uma instrução utilizando o número requerido de ciclos de máquina, e é preciso que termine a execução de uma instrucão antes de iniciar outra. Se fornecermos uma seqüência correta de códigos de operação seguidos cada um por seu operando (ou operandos) adequado, a CPU conseguirá executar de forma correta o "programa" que lhe proporcionamos.

Encontramo-nos na situação da fitura 8 que, apesar de ser um pouco festiva, indica claramente a relação entre a CPU, os dispositivos externos e as instruções. O bus de dados é uma fita transportadora que leva os dados de forma sequencial até a CPU. Os dados estão simbolicamente representados por grupos de cubos, o primeiro dos quais é sempre o código de operação, enquanto que os demais são (se existirem) os operandos. Cada grupo é uma instrução, que se vê absorvida de forma sequencial pela CPU. Quem introduz os dados no correspondente bus de dados? São os dispositivos externos, depois de terem sido selecionados mediante o bus de direções e o de controle. Todas as operações são "implacavelmente" controladas pelo relógio (veja o "despertador da avó"). Para não complicar o desenho, o bus de dados foi representado como unidirecional.

Pelo que foi dito até agora, poderia parecer que a execução de toda instrução traz consigo exclusivamente a leitura de dados nos dispositivos externos até a CPU, mas não é certo.

Podemos dar-nos conta, por exemplo, seguindo a execução da seqüência "desloca um dado do dispositivo 1 ao dispositivo 2".

Assim veremos que é necessário a bidirecionalidade do Bus de dados.

No primeiro ciclo de máquina, o decodificador de instruções (ID) reconhece o código de operação que corresponde à instrução "desloca um dado de um dispositivo a outro". Para executá-la a CPU terá que conhecer tanto o dispositivo no qual irá realizar a leitura como aquele no qual irá escrever o dado. Sabemos que cada dispositivo tem uma direção. Pois bem, a CPU deve identificar a direção de "leitura" e de "escrita", que denominam-se "fonte" e "destino". Estas duas direções devem especificar-se como operandos da mesma instrução de deslocamento (movimento). Nossa instrução deverá estar então constituída por:

- Um dado inicial de 8 bits (código de operação).
- Dois dados de 8 bits, isto é, um dado completo de 16 bits, que defina a direção fonte (primeiro operando).
- Dois dados de 8 bits, para a direção de 16 bits de destino (segundo operando).

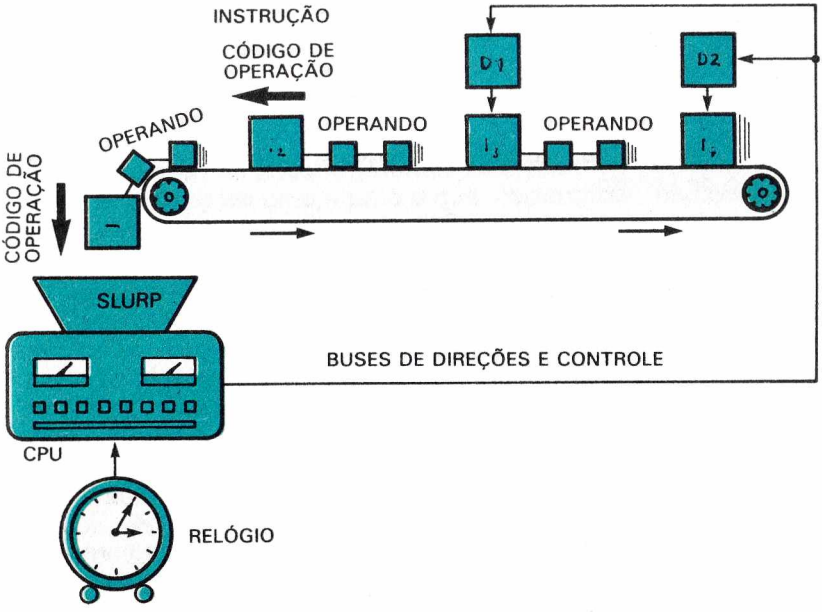


Fig. 8 - A CPU absorve de forma sequencial as instruções que chegam procedentes dos dispositivos externos (memória, E/S).

É fácil compreender que no segundo ciclo de máquina, a CPU adquire a metade do primeiro operando. Neste ponto, é bom recordar que os dispositivos externos trabalham todos eles com dados que têm uma longitude de somente 8 bits, enquanto que as direções são de 16 bits.

No terceiro ciclo, a CPU adquire a segunda parte da primeira direção e o ID a capta e a coloca junto a primeira metade.

No quarto ciclo, o ID situa esta direção completa de 16 bits no bus de direções, chamando ao dispositivo externo 1 e atuando o modo de leitura para obter o dado. No final do quarto ciclo, o dispositivo fonte enviou o dado que é captado pela CPU e guardado no acumulador.

Os ciclos quinto e sexto dedicam-se ao compilador do segundo operando, pelo que ao final do sexto ciclo o ID "tem no bolso" a nova direção de destino do dado. Finalmente, no sétimo e último ciclo de máquina, o ID, com esta direção, chama, em modo escrito, o dispositivo no qual irá guardar o dado.

Neste caso, o bus de dados está invertido e o fluxo de dados é fornecido desde o acumulador da CPU ao dispositivo selecionado para a escrita.

Conclusão: a instrução de "movimento" tem uma longitude de 5 bytes, isto é, utiliza 5 dados (inclusive o código de operações) para ficar definida por completo, e sua execução requer 7 ciclos de máquina por parte da CPU, dos quais 6 são de leitura e 1, o último, é de escrita.

Registro de estado e decisões da CPU

Na CPU da figura 7d está indicado, em linhas tracejadas, um registro muito importante: o registro de estado da CPU (em inglês, Status Register ou Flag Register), do qual prescindimos até aqui para que a explicação não ficasse complicada demais.

Tal registro põe de manifesto, em cada momento, qual é a situação dentro da CPU. Sabemos que todas as instruções tratam os dados sob a forma de "zeros" e "uns". Pois bem, pode haver alguma combinação que nos interesse recordar com a finalidade de executar de forma correta a seguinte instrução. Por exemplo, o registro de estado recorda se o último resultado que obtivemos era um dado com todos os bits "zeros", se o bit 7 foi colocada em "1" (número negativo), se existe um acarretamento procedente da soma que acabou-se de executar, etc. Cada uma destas situações, se for produzida, "acende" um bit de registro de estado, intimamente conectado com o ID. Desta forma é possível que o ID possa "to-

mar decisões" sobre como executar corretamente a seguinte instrução, baseando-se nos resultados da instrução anterior, refletidos no registro de estado. A faculdade de tomar decisões é uma das características mais importantes de um microprocessador e permite a escrita de programas complexos e versáteis, com instruções do tipo "se o resultado da instrução é zero, faça isto; se não, faça este outro...".

Este sim que é um programa!

Aos "sobreviventes" da maratona precedente, agradará saber que a seqüência de instruções que até agora aplicamos "brutalmente" à entrada da CPU, são realmente parte de um programa. Esta é pois, também a definição de "programa": seqüência de instruções executáveis pela CPU. Certamente tem uma grande diferença entre um programa entendido como uma seqüência de códigos máquina (OP.CODE + operandos) e um programa escrito "de forma inteligível", por exemplo, em BASIC. No segundo caso também trata-se de uma seqüência de instruções executadas pelo computador pessoal, mas tem uma série de passos intermediários que facilitam a compreensão humana. Por hora, contentamo-nos em saber que a CPU só funciona perfeitamente se lhe proporcionarmos instruções "sensatas" uma após a outra. É evidente que, na realidade não nos encarregaremos de transmitir pessoalmente à CPU as instruções uma a uma e que, certamente, não conseguiremos seguir o rítmo do relógio. Por conseguinte, as instruções procederão de alguma forma dos dispositivos externos conectados à CPU.

Intencionalmente não nos aprofundamos até agora nestes misteriosos "dispositivos", porque não achamos conveniente colocar "muita carne para assar" em um capítulo por si muito árido para o principiante. Chegou o momento de revelar suas identidades: tais dispositivos são a memória (RAM ou ROM) e os circuitos de interconexão ("interface") com o exterior, isto é, os circuitos de entrada/saída (E/S, I/O em inglês). Sem estes circuitos integrados, a CPU não poderá funcionar. No capítulo seguinte serão estudados com mais detalhes estes importantes dispositivos.

Antes de terminar o capítulo, uma observação: a exposição realizada sobre o funcionamento da CPU está baseada em considerações gerais e justificadas, ainda que para os "Sherlock Holmes" desta matéria, esclarecemos que as referências pesquisadas, um pouco aqui e um pouco acolá, concentram-se nos microprocessa-

dores Z80 (Zilog) e 6502 (Rockwell).

GLOSSÁRIO

SILÍCIO: Elemento dos mais comuns na terra. Seu símbolo químico é Si e emprega-se como material básico para a construção de transistores e circuitos integrados. O silício para tais usos, apresenta-se em cristais cilíndricos homogênos e puríssimos, que são cortados em fatias. Cada uma destas rodelas, toma o nome inglês de "wafer". Nestes suportes imprimem-se por meios microlitográficos e de fotogravação o interior dos chips; finalmente, separam-se cortando a pastilha suporte e realizando sua montagem em cápsulas, para chegar ao chip definitivo.

VLSI: abreviatura da expressão inglesa "Very Large Scale Integration" (Integração em larga escala). É o nível de complexidade de circuitos integrados tais como microprocessadores, memórias e interfaces de E/S. (A integração em pequena, média e grande escala tem as abreviações SSI, MSI e LSI).

CHIP: pastilha de silício na qual estão fotografados os circuitos lógicos, preparados para uso imediato, e alojada em cápsulas integradas.

ÁLGEBRA DE BOOLE: Série de regras que regem as relações e operações entre valores lógicos, isto é, entre fatos que podem ser representados por "zeros" e "uns". As operações lógicas fundamentais são: AND, OR e NOT. Com estes circuitos lógicos básicos são feitos circuitos mais complexos, tais como o ALU (Unidade Aritmético-Lógica).

CICLO DE MÁQUINA: É o período mínimo de tempo empregado pela CPU para adquirir um dado de memória ou escrevê-lo na mesma. Mede-se em número de ciclos de relógio.

CICLO DE RELÓGIO: Período da onda quadrada procedente de um dispositivo externo, que serve para sincronizar as funções do μP.

BIT: Unidade de informação. Pode tomar os valores "0" e "1".

BYTE: Agrupamento de 8 bits. Os microprocessadores mais conhecidos são de 8 bits porque têm o Bus de dados desse tamanho. Outras unidades de CPU mais modernas são de 16 bits ao ser seu bus de dados de 16 linhas (16 bits).

WORD (palavra): Conjunto de bits que podem ser manipulados pelo μP em uma só operação. Segundo a CPU a palavra pode ser de 1, 2 ou 4 bytes.

Z80: Conhecida CPU produzida por Zilog. Utiliza-se com grande freqüência nos computadores pessoais de gestão que adotam o sistema operacional CP/M.

6502: Clássica CPU adotada por Apple, que a empregou em todos os seus computadores pessoais, com exceção do Macintosh.

8086/8088: CPU de 16 bits da Intel adotada pela IBM para seu PC e por outros fabricantes de microsistemas de 16 bits. O microprocessador 8088 (usado no IBM-PC) tem o bus de dados exterior de 8 bits, enquanto que outros computadores empregam a mais potente CPU 8086, compatível (em software) com o 8088.

68000: CPU muito potente, produzida pela Motorola, que é utilizada nos computadores pessoais mais modernos. Tem o Bus de dados externo de 16 bits, mas internamente os registros são de 32 bits, o que permite um tratamento muito rápido dos valores armazenados em memória (utilizado no Macintosh).

CAPÍTULO III

CIRCUITOS DE APOIO À CPU

No capítulo anterior tratamos de assimilar como trabalha uma CPU. No entanto, pudemos dar-nos conta também de que uma CPU não funciona por si so; de fato, cada uma das suas ações está em estreita correlação com a existencia de circuitos exteriores de apoio, os denominados "chips periféricos". Estes têm muitas funções, mas as principais são proporcionar dados à CPU e aceitar os mesmos dela.

No capítulo anterior, também classificamos em três categorias este tipo de chips: RAM, ROM e dispositivos de E/S. Veremos agora, com detalhes, suas funções.

Começaremos com as memórias RAM, que podem ser lidas e escritas pela CPU, que prescindindo de sua função, tudo quanto dissermos sobre suas conexões e arquitetura terá validade também para a memória ROM e para os dispositivos de E/S.

RAM: "Random Access Memory" Memória de Leitura e Escrita

Num sistema de microprocessador, a memória tem a função de receber a guardar informações e fornecê-las de novo sob a forma de bloques contíguos de bytes.

Com freqüência fala-se em termos de bytes, se bem que, como explicamos anteriormente, as CPU's de 8 bits têm um bus de dados precisamente de 8 bits e, conseqüentemente, os chips periféricos conectados a elas, poderão fornecer, ou admitir, dados completos desta longitude.

Antes de examinar as conexões do hardware propriamente ditas, concentrar-nos-emos em dar uma idéia clara da função que desempenha a memória. Podemos compará-la com um armário, provido de uma porta principal com fechadura: se for aberta, deixa a descoberto uma configuração interna constituída por diversas filas de caixas, cada um dos quais tem sua própria etiqueta. Para chegar a uma caixa, primeiro é preciso conhecer o armário ao qual pertence e selecioná-la, isto é, tem que possuir a chave correspondente à porta do armário. Além disso, devemos saber qual é a caixa, dentro desse armário, que queremos abrir para obter o dado que contém. Uma vez aberto a caixa vemos seu interior para ler o dado; então tornaremos a fechá-la deixando seu conteúdo tal e como estava. Evidentemente esta operação é uma "leitura".

Se, pelo contrário, uma vez aberto a caixa, esvaziarmos seu conteúdo e logo introduzirmos um novo dado, efetuaremos uma operação de "escrita" (o novo dado substituiu o anterior). O processo pode ser acompanhado na figura 1.

Do anteriormente exposto deduzimos que, uma vez aberto o armário, cada caixa pode ser aberta sem seguir nenhuma regra nem respeitar nenhuma prioridade. Esta é a razão pela qual este dispositivo recebeu a denominação de memória de acesso aleatório ("random access memory").

Se neste preciso momento, mediante uma operação mental, mudarmos o mundo mecânico pelo do hardware microeletrônico, passaremos do armário ao chip de memória (por exemplo, uma RAM de 2048 bytes) e a cada caixa corresponderá uma célula individual de memória entre as 2048 disponíveis. A memória será

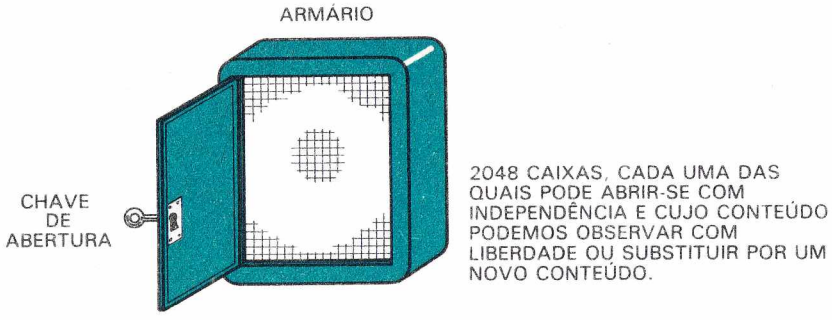


Fig. 1 - Uma memória pode ser comparada com um armário que contém muitas caixas, sendo cada uma acessível de maneira independente.

impenetrável quando não possuirmos a chave de acesso que no armário abrimos a porta e no chip enviamos o sinal de "seleção de chip" (CS = chip select). Agora que estamos nele é conveniente esclarecermos algo sobre a terminologia informática: uma memória de 2048 bytes fala-se que tem 2 kbytes. Aqui o K não indica x1000 como na física (kg = 1000 gramas) mas sim, x1.024. A razão é que é o valor da potência 2 mais próxima a 1.000 a utilizada (2^{10} = 1024) e não devemos esquecer que os valores que podemos obter em aritmética binária são, forçosamente, soma de potências de 2.

Quem gera o sinal "chip select" (no seguinte, denominada CS)? Naturalmente a CPU do sistema, ainda que a operação não se produza de forma direta, mas através de um hardware externo que toma o nome de "decodificador". Se o microcomputador é a oficina, a CPU é o chefe, as transferências dos dados são realizadas fisicamente pelo inefável Sr. Luis (de forma muito rápida, logicamente) e a memória é o arquivista, uma operação de leitura pode ter lugar tal como se indica em seguida:

Chefe: "Luis, traga-me uma cópia do capítulo 23 do expediente Roberto (dado a buscar)".

Luis: Pega o livro maior (decodificador) e encontra: "expediente Roberto: armário 3". Pega imediatamente a chave correspondente.

Luis: Abre o armário (Chip Select) e vai à caixa 23, pega o expediente, fotocopia-o e torna a colocar o original onde estava (operação de leitura completada).

Luis: Fecha o armário (cancela a seleção da memória) e com as cópias na mão dirige-se ao despacho do Chefe (dados que chegam finalmente à CPU).

Naturalmente, como costuma ocorrer fora do mundo do hardware, não há nenhum sinal de agradecimento por parte da CPU.

A operação de escrita é produzida de forma análoga, com a desculpa de que a ordem da CPU é: "coloque este expediente no arquivo". Cuidado com isto, pois a operação de escrita tem caráter destrutivo, isto é, o conteúdo anterior da memória perder-se-á para sempre.

Passando esta analogia ao processo real, a questão seria: a CPU coloca no bus de direções a correspondente à posição de memória que quer ler (ou escrever) e ativa o sinal RD (ou WR) do bus de controle; o decodificador lê este valor e ativa (CS) tão somente o chip de RAM que contém tal direção. Então, a célula correspondente copia seus dados no bus de dados se estiver ativado RD,

ou então, armazena os dados do dito bus nela se estiver ativado WR.

Neste momento estamos em condições de analisar a figura 2. (Nela a parte|a) mostra a conexão de uma típica memória RAM de 2K x 8 e a parte b) indica a "patilhagem" de circuitos integrados comerciais (recordemos que K é uma constante que equivale a 1024). Observa-se imediatamente que tem 8 linhas para a transmissão do dado desde e para a memória. De fato, cada memória RAM está conectada "em paralelo" ao bus de dados. Também está em paralelo a conexão para parte do bus de direções e a memória somente está conectada ao número de linhas de direção necessárias para gerar todas as direções internas de suas células, o que é possível graças à existência de uma decodificação posterior que é produzida no interior da própria memória.

Para distingüir fisicamente 2048 células precisa-se 11 linhas (2^{11} = 2048), de A0 à A10. No entanto o bus de direções é de 16 linhas (de A0 à A15, portanto, sobram 5 sem unir à RAM). Devemos assinalar que à memória RAM estão conectadas as linhas "menos significativas" (de menor valor), enquanto que as linhas "mais significativas", de A11 à A15, estão conectadas ao decodificador do sistema. Posto que temos cinco linhas que controlam dita decodificação, as combinações à entrada poderiam ter como máximo 2^5 = 32, conseqüentemente, o decodificador teria um máximo de 32 saídas de CS e poderemos escolher uma entre essas 32 saídas independentes para utilizá-la como seleção do nosso chip de memória RAM. Assim, por exemplo, podemos fazer que a memória RAM "responda" somente quando o bus de direções à CPU situe uma combinação de "zeros" e "uns" compreendida entre "0000 0000 0000 0000" ($0000 em hexadecimal — ver glossário —) e "0000 0111 1111 1111" ($07 FF). Pode observar que, de forma intencional, os cinco bits mais significativos (mais à esquerda) da direção permanecem sempre fixos a "00000". Com qualquer destas disposições válidas o decodificador do sistema manterá ativa uma linha de saída que chamaremos "chip select 0" (CS0 que corresponderá às linhas A11-A15 à 0). Para qualquer direção compreendida entre estas 2048 (de $0000 a $07FF) a memória RAM será habilitada através da linha CS0. Como é evidente, se CS0 direciona 2 Kbytes o mesmo acontecerá com as outras linhas de CS, logo poderemos abranger 32 x 2048 = 65536 bytes, que é toda a memória direcionável com um bus de direções de 16 bits (2^{16} = 65.536).

Conclusão: a CPU poderá dialogar com um máximo de 32

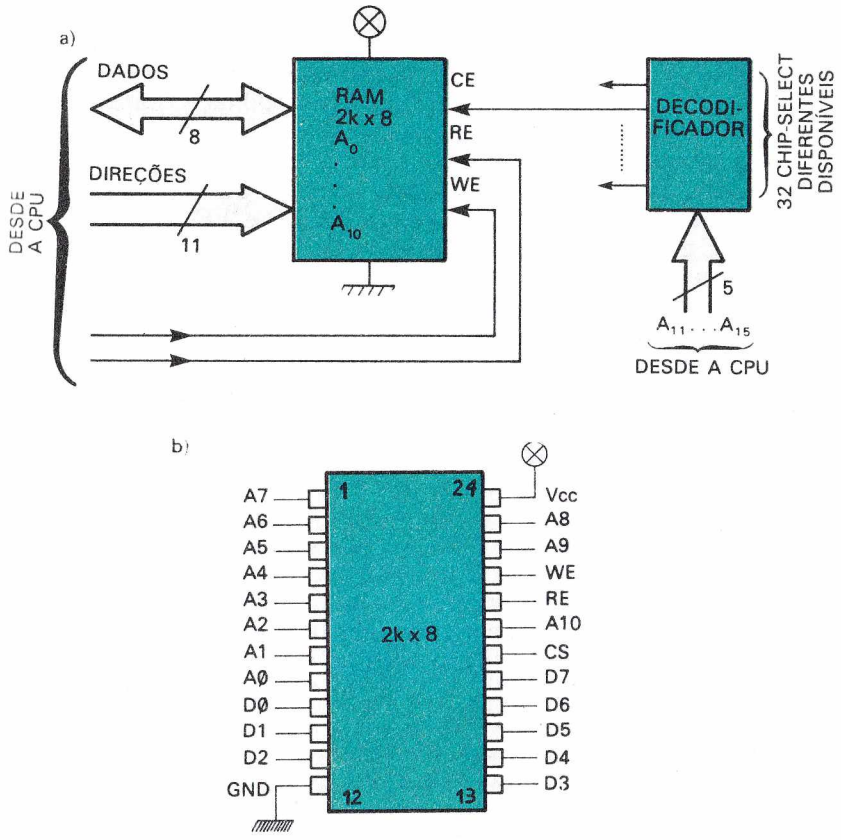


Fig. 2 - Linhas de controle, dados e direções de um chip de RAM (2K × 8) (a) e patilhagem de um componente comercial (b)

chips se cada um é de 2K x 8 bytes. Evidentemente, nada impede de utilizar chips de outra capacidade e colocar em prática uma decodificação mais complexa. Por outro lado, podem encontrar-se chips conectados à CPU com capacidades de 4K, 8K e 16K pelo alto e, por baixo, de até alguns bytes, pois existem circuitos de E/S que contém muito poucas células de memória (normalmente 4, 8 ou 16). (Seguramente foi observado que, às vezes "cortamos" o zero. Esta é uma prática normal em informática para diferenciá-lo perfeitamente da letra O. Fazemos isto somente quando, pelo

contexto, possa prestar-se a confusão).

Voltando à figura 2, além das linhas de alimentação (+ e massa) observamos duas linhas muito importantes no controle do chip: a linha de habilitação para leitura (RD) e de habilitação para escrita (WR), ambas já conhecidas pelo leitor. Recordemos somente que a CPU é o componente que sabe quando a operação a executar é uma leitura ou uma escrita, conseqüentemente, será sempre a CPU quem ativa a linha de controle necessária. RD e WR (às vezes chamadas RE e WE por Enable = habilitar, ou somente R e W) podem ser ambas inativas, ou melhor, uma ativa, e a outra inativa, mas nunca podem ser ambas ativas de uma vez. Para evitar que isso possa ocorrer é muito normal introduzir ambas na mesma linha de controle (chamada então R/$\overline{W}$ ou $\overline{R}$/W) de forma que cada uma considere-se ativa com um valor da linha (a que leva o guiador em cima será ativa quando tivermos um 0).

As figuras 3 e 4 mostram os diagramas de tempos, ou cronogramas, do desenvolvimento de um ciclo de leitura e de um ciclo de escrita entre a CPU e a RAM. Vamos centrar-nos no primeiro (figura 3), correspondente a leitura. O primeiro sinal que observa-se é a de relógio, controlando um ciclo qualquer da máquina. A CPU estabelece o valor correspondente no bus de direções uma vez que ativa o sinal RE para indicar que irá realizar uma leitura. O decodificador lê o valor do bus de direções e, transcorrido um pequeno atraso Td, responde ativando a linha CS correspondente que então, habilita a memória direcionada e esta sabe que a operação é de leitura ao observar que está ativada RE. A memória RAM decifra o valor das linhas do bus de direções aplicadas a suas entradas e, transcorrido um tempo de acesso Tacc (varia segundo a tecnologia de contrução, mas costuma estar entre 150 e 400 ns para as memórias RAM mais comuns) situa o dado localizado nessa célula no bus de dados. Ao finalizar o ciclo de máquina, a CPU pode "engolir" o mesmo sem problema algum. Imediatamente apagará o bus de direções e o sinal RE; o decodificador desabilitará o sinal CS, desativando a memória e ficando pronta para reativar-se durante o ciclo seguinte.

Se tratarmos de uma escrita, tudo desenvolver-se-á como indicado no cronograma da figura 4. A CPU emite a direção e o decodificador gera, depois de um tempo de atraso Td, o sinal de Chip Select. Ao mesmo tempo que a direção, o sinal WE torna-se ativo, pelo que a memória RAM saberá que "está por chegar" um dado procedente da CPU. Esta última deverá manter um certo tempo o dado no bus de forma estável uma vez identificada a célula que deve modificar-se. Tal tempo denomina-se Tds (Data-Setup).

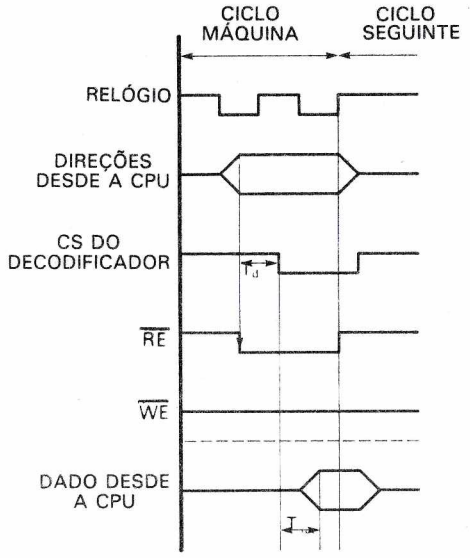


Fig. 3 - Cronograma de um ciclo de leitura genérico.

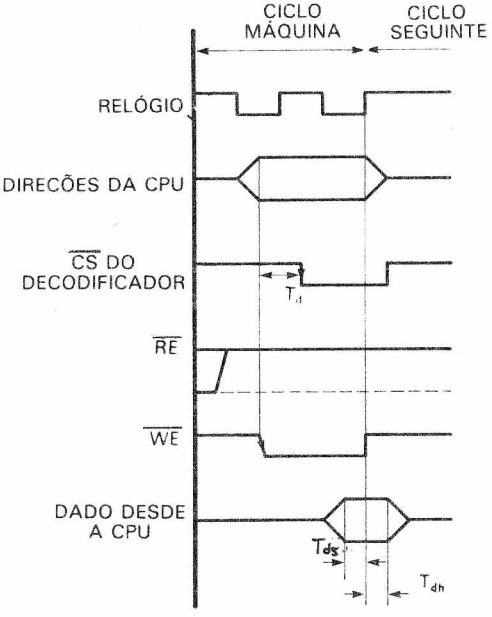


Fig. 4 - Cronograma de um ciclo de escrita genérico.

Eliminado então o sinal WE o dado é gravado na RAM permanecendo todavia no bus durante um breve período de tempo (Tdh = Data Hold — manutenção do dado —). Uma vez terminado o ciclo, a CPU quita a direção do bus correspondente, desaparecendo CS, a memória RAM se desativará e a célula direcionada terá um dado novo.

Se você seguiu a explicação fixando-se nas figuras, seguramente surpreendeu-lhe que em vários sinais quando nós dissemos que se ativavam no cronograma passavam do valor 1 ao 0. Vejamos quais são as razões desta conduta.

Sinais ativos: a nível alto (1) ou a nível baixo (0)?

Até agora tratamos os sinais de controle (por exemplo CS, WE, RE, etc.), com seu nome genérico, sem especificar o fato de que o sinal ser ativo quando estiver em "1", ou em "0". Ainda que aqui continuemos com esta nomenclatura (os gráficos e as ilustrações servirão para dissipar as dúvidas) achamos conveniente comentar o que costuma empregar-se em catálogos e manuais técnicos: quando um sinal é ativo no nível 1 emprega-se seu nome, no entato quando é ativo no nível 0 situa-se em cima de seu nome um guiador (por exemplo, $\overline{CS}$).

O atento leitor terá observado que os sinais de controle mais comuns, tais como WE, RE, etc. estão "ativos" quando seu nível lógico é "0", e inativos se seu nível lógico é "1". A razão fundamenta-se no uso de tecnologias (TTL, N-MOS) nas quais os chips consomem menos energia se as entradas estão ao nível lógico "1". Posto que os sinais que são utilizados em um microcomputador são ativos durante um tempo muito pequeno frente ao total, torna-se conveniente utilizar sinais ativos com o nível lógico "0" e inativos com o nível lógico "1", já que o consumo assim será inferior. No entanto, para os fins que solicitamos, esta questão resulta irrelevante e poderia complicar a explicação. Por outro lado, no capítulo anterior decidimos tratar os diversos componentes do hardware como "caixas negras", ignorando por completo a estrutura interna. Assim, continuaremos falando dos diversos sinais sem especificar constantemente se são ativos com o nível lógico "0" ou "1", simplesmente explicitando quando for oportuno, sem maiores complicações.

Algo mais sobre a decodificação

Para não complicar neste momento a exposição, passamos

por alto a explicação do funcionamento de um decodificador. Limitemo-nos agora a esse cabo solto.

Na figura 5 mostramos um decodificador muito simples, constituído por 4 portas AND e 8 portas inversoras NOT (por simplificação, no esquema as negociações ou inversões estão indicadas, segundo o convênio universal, por pequenos círculos desenhados nas entradas ou nas saídas das portas). O decodificador é definido fisicamente por tudo o que está dentro do retângulo de linhas tracejadas e estará integrado em um só chip. Este circuito distingue as quatro combinações possíveis em suas duas entradas, ativando para cada combinação a linha de saída correspondente.

No capítulo dois descrevemos o funcionamento de uma porta lógica AND e, conseqüentemente, sabemos que sua saída está ao nível lógico "1" somente se todas as entradas estão a tal nível. Torna-se fácil comprovar, por exemplo, que a saída CS0 é ativada somente se as duas entradas do decodificador, A14 e A15, estão ao nível lógico "0", já que logo serão objeto de negação e, conseqüentemente, passam ambas ao nível lógico "1", tal como era requerido. A saída está, pois, a "1", mas passará a "0" (observe o pequeno círculo de negação que transforma a porta AND em uma porta NAND, isto é AND-NOT) e assim obteremos nosso desejado sinal de seleção.

Por que utilizam-se precisamente A14 e A15 como entradas ao decodificador? A resposta é simples: A14 e A15 são as duas linhas de direção mais significativas do bus de direções, assim, ao utilizá-las, podemos subdividir fisicamente, graças a nosso decodificador, toda a memória direcionável em quatro cômodos "bancos" de 16k x 8 bits (16 Kbytes) cada um. Com o emprego de 4 chips de 16K bytes implantamos com facilidade toda a memória (RAM e ROM) no nosso computador, cujo diagrama de blocos coincidirá com o da figura 6. Nesta figura, os quatro chips estão conectados em paralelo ao bus de dados de 8 bits e ao bus de direções (com exceção das duas linhas mais significativas); 14 linhas bastam para identificar cada célula objeto de chamada entre as 16.384 disponíveis em cada chip (2^{14}). A seleção é confiada aos sinais de CS que chegam procedentes do decodificador e este último, por sua vez, está conectado às duas linhas sem usar o bus de direções (A14 e A15).

A figura 7a ilustra uma representação mecânica de um decodificador. A configuração presente na entrada posiciona o contato do seletor, ativando a única saída válida. É importante destacar a existência de uma entrada adicional que permite levar o comutador a uma posição "neutra", na qual todas as saídas são ina-

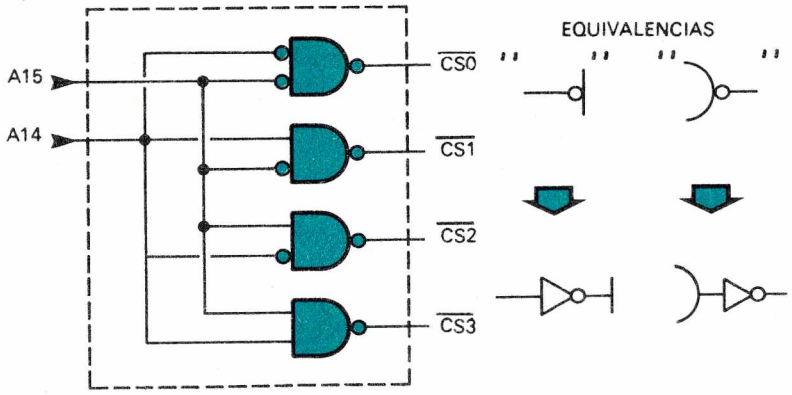


Fig. 5 - Simples decodificador de 2 a 4, feito com portas AND e inversores.

tivas (nível lógico "1"). Tal entrada denomina-se "habilitação do decodificador" e costuma estar ativo para um nível lógico "0". Com a não habilitação do decodificador, a situação das entradas já não tem nenhuma influência sobre as saídas, que permanecerão segundo foi dito, todas elas a nível lógico "1".

Podemos criar muitos tipos de decodificadores, mas os modelos clássicos da tecnologia TTL são os de 2 a 4, 3 a 8 e 4 a 16, o que mostra que as saídas são, em qualquer caso, igual a dois elevado ao número de entradas ($4 = 2^2$, $8 = 2^3$ e $16 = 2^4$). Naturalmente, também o chip de memória tem em seu interior um decodificador, se bem que muito mais complexo, levando-se em conta de que tem que selecionar uma célula concreta entre milhares. No caso de uma memória RAM de 2K por exemplo, a decodificação interna será do tipo 11 a 2048.

Como cada memória tem seu próprio decodificador devendo conectar ao chip tantas linhas de direção (de A0 em diante) quantas entradas necessite o decodificador interno, as linhas não usadas do bus de direcões serão empregadas nos decodificadores externos.

Assim para uma memória de M bytes farão falta: $2^N = M$ — n (nº linhas necessárias) = log M/log2. Posto isto para M = 2048 bytes — n = 11, tal como se viu nos exemplos ilustrados.

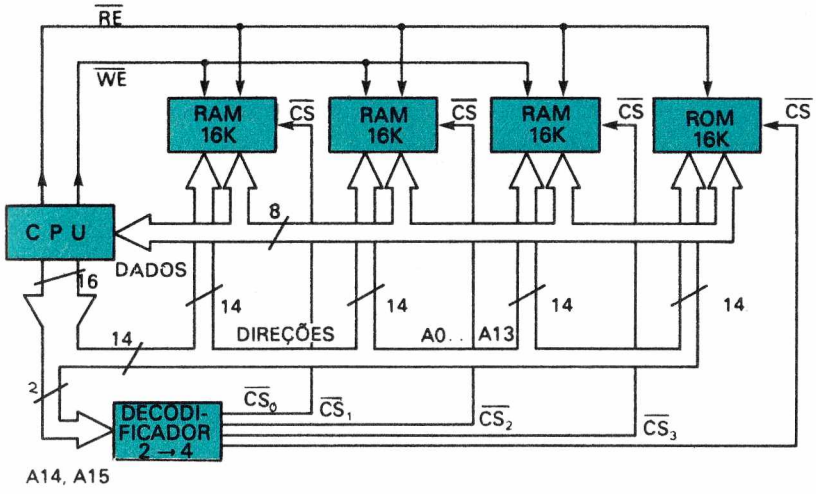
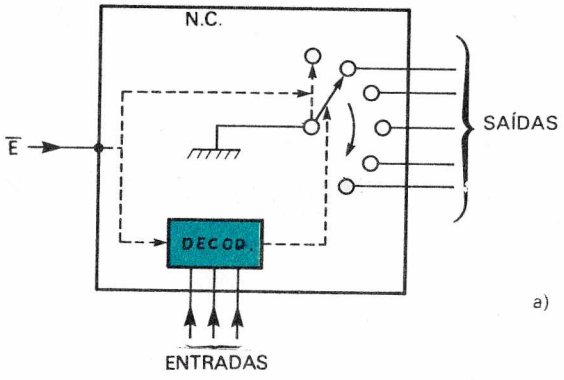


Fig. 6 - Esquema completo de um desenho hardware com CPU, RAM, ROM e decodificador correspondente para selecionar, de forma correta, 4 chips de 16 k cada um.

Na figura 7b podemos ver o diagrama de blocos de um chip de memória, com as células e o decodificador interno para sua seleção. O sinal "Chip Select" serve, precisamente, para habilitar este decodificador interno que, se não fosse assim, manter-se-ia inerte inclusive quando fossem ativadas as "n" direções em sua entrada.

No computador da figura 6 temos 48K de RAM, cujos chips são reconhecíveis pelo fato de que tem duas entradas de controle RE e WE (ativas no nível lógico "0") e outros 16K que estão destinados à memória ROM, que será matéria do seguinte item. Parece-lhes que uma estrutura desta forma está computada e é simples? Pois bem, acredite ou não, salvo poucas modificações, também seu computador pessoal tem este tipo de organização. Tudo quanto lhe pareça misterioso, ou complicado, nos esquemas inclusos em seu manual, é somente resultado das características acrescentadas a esta base de partida comum (por exemplo, conexões e interfaces diversos, discos, controlador de vídeo, etc.).



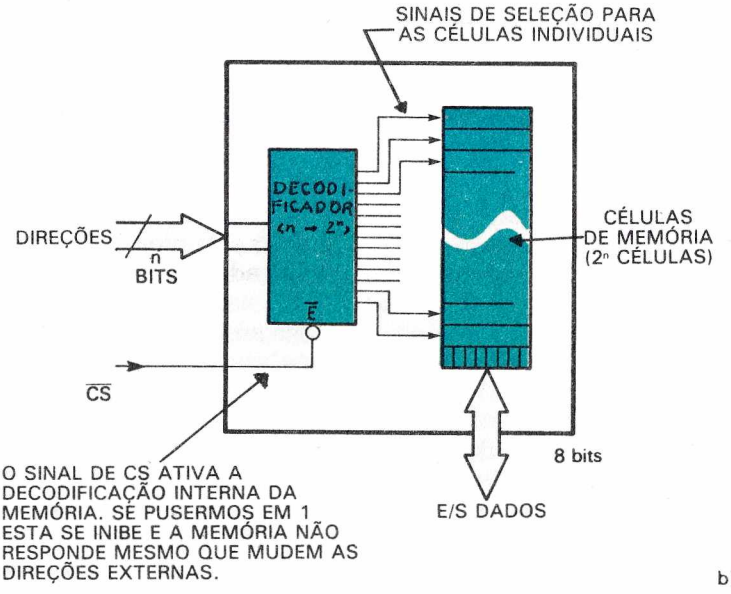


Fig. 7 - (a) Um decodificador é como um seletor mecânico cujo cursor está posicionado, de forma unívoca, pela combinação aplicada às linhas de entrada. A entrada de habilitação, se é inativa (nível lógico "1"), desliga o comutador colocando-o na posição "não conectado" (NC); (b) estrutura interna de uma memória RAM (ou ROM).

Rom: "ready only memory". A memória com dados fixos

A memória RAM tem a peculiaridade de poder alterar o conteúdo de cada uma de suas células à vontade da CPU. Por isto, o principal uso de uma memória RAM é o armazenamento provisório de dados (resultados intermediários, valores de variáveis, texto a processar) e de programas. Em resumo, de tudo aquilo que se pode refazer, voltar a escrever ou mudar. Existe somente um "pequeno" inconveniente: uma memória RAM trabalha perfeitamente enquanto recebe energia, mas se cortamos a alimentação (desligarmos o sistema) e logo tornarmos a ligar, encontrar-nos-emos com uma situação que podemos qualificar de "desastrosa", ou seja, o conteúdo anterior à interrupção de corrente desaparecerá na memória RAM e será substituído por configurações totalmente aleatórias de códigos binários de 8 bits que não nos servirão para nada.

Conclusão: em uma memória RAM não podem armazenar-se dados de modo "permanente". Como pode então nosso computador pessoal "despertar", apenas sendo ligado, e saber imediatamente o que deve fazer? Tudo se deve à existência das memórias ROM que, afortunadamente, estão presentes nele. São tipos de memórias que não perdem os dados, mesmo que falte a alimentação.

No entanto, como contrapartida, uma memória ROM não pode ser escrita como sucede com uma RAM, isto é, pode ser lida, mas seu conteúdo não pode modificar-se mediante uma operação de escrita normal.

O único modo de mudar inclusive um simples bit de uma memória ROM, é substituí-la por outra adequadamente programada durante sua fabricação. A tecnologia atual está investindo muito dinheiro na pesquisa de dispositivos de memória não voláteis que funcionem como a memória RAM (isto é, que podem ser escritas além de lidas) mas, até agora, não encontraram o substituto ideal. Existem algumas "aproximações", como as memórias EEPROM e NOVRAM que funcionam muito bem, mas somente servem para aplicações determinadas e, além disso, seu custo é muito elevado. As memórias CMOS, mais que "não voláteis", o que fazem é necessitar muito pouca energia para conservar o dado, com o qual uma simples pilha evita a perda da informação. Na prática, nosso computador pessoal deverá apresentar uma mescla adequada de chips de memória RAM e ROM, confiando à primeira o tra-

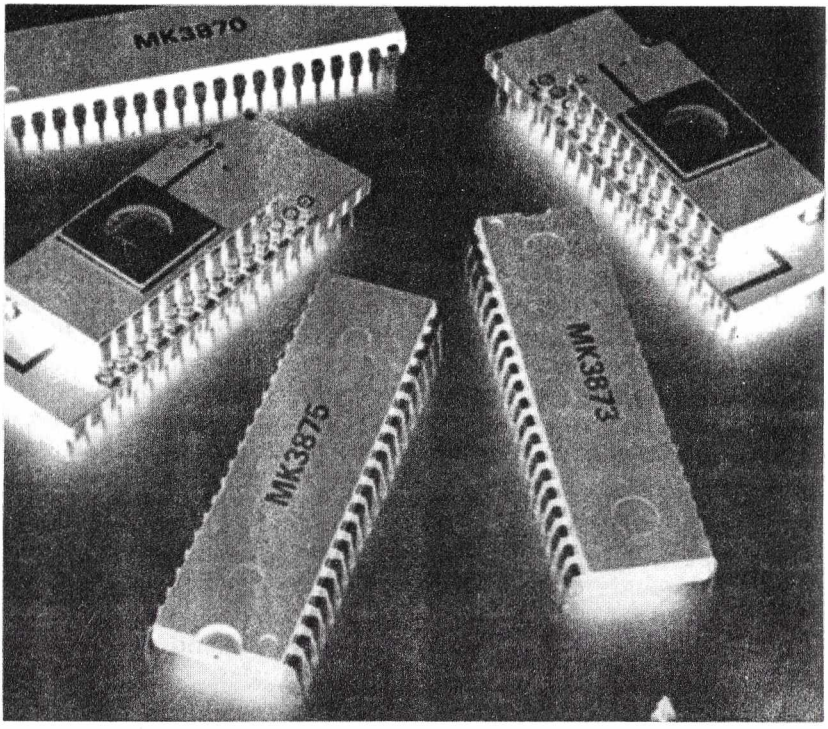


Foto 1 - Alguns dos circuitos integrados "monochips" mais difundidos no mercado. Cada cápsula contém um único chip no qual estão integrados, além da CPU, uma certa quantidade de ROM, RAM e alguns dispositivos de E/S. O uso destes circuitos de "um só chip" é recomendado para todas aquelas aplicações nas quais é importante economizar espaço reduzindo o número de chips utilizados.

balho de armazenamento provisório (com o computador pessoal ligado, para entender-nos) e, à segunda, a missão de reter aqueles dados necessários para que arranque sem problemas o sistema no momento de ser ligado e funcione corretamente depois.

A estrutura interna (tecnologias em separado) de uma memória ROM é idêntica à de uma memória RAM, assim como suas conexões, com a diferença de que na ROM falta a linha de habilitação para a escrita posto que, como é evidente, não serviria para nada.

O que ocorre quando conectamos o computador?
Uma nova linha de controle: Reset

Até que a tensão da rêde não chegue à fonte de alimentação de nosso computador, este está mudo e inerte sobre a mesa, no entanto seu hardware está preparado para despertar de imediato assim que acionarmos o interruptor.

A CPU, como sabemos, é uma executora obediente de instruções, que não são outra coisa que códigos de "uns" e "zeros" que extrai da memória, sucessivamente. Quando ligamos o sistema, um pequeno circuito externo à CPU põe, por uns instantes, ao nível lógico "zero" uma entrada da CPU, denominada de "RESET" (colocada a zero ou iniciação). Um nível lógico "zero" nesta entrada, é como o timbre que soa exatamente antes do começo de uma classe: ao ouví-lo, todos os alunos permanecerão atentos esperando a chegada do professor. No nosso caso todos os dispositivos internos permanecem à espera para começar a trabalhar. Inclusive o ID (codificador de instruções) fica esperando que o sinal de RESET volte ao nível lógico "um", depois do qual, com sua acostumada exatidão, começará a dirigir as tarefas no interior da CPU.

E é aqui que o RESET voltou ao nível lógico "1". Ao observar esta mudança o ID ordena ao contador de programa que inicie a conta a partir de um valor determinado e característico de cada μP. Vamos supor que este valor, naturalmente de 16 bits, seja o número binário 1111 1111 1111 1100 ($FFFC em hexadecimal). Tal direção, com a qual o contador de programas será iniciado depois do RESET denomina-se "Vetor de Reset". Mas o trabalho não está todavia terminado. O ID ordena imediatamente que sejam lidos os dados contidos nesta direção de RESET e na imediatamente superior. Estes dois valores de 8 bits são "capturados" pelo ID, que deposita-os no contador de programa de forma que, finalmente, formam o valor da direção de memória, a partir do qual começa o programa de iniciação propriamente dito do computador. É essa a razão do nome "Vetor de Reset", pois seu conteúdo "aponta" ao lugar onde começam as tarefas de Reset.

Os mais astutos devem ter suspeitado que as duas direções consideradas (de Reset) devem corresponder a um chip de ROM, pois é fundamental que seu conteúdo seja sempre idêntico, para que em qualquer "ligada" o computador proceda da mesma forma. Pelas mesmas razões as instruções do programa de iniciação deverão estar também na memória ROM.

Em poucos instantes a máquina estará completamente

iniciada.

A partir desse momento será fundamentalmente a memória RAM a que, será colocada à disposição do usuário, quem poderá escrever seus programas, lê-los e mudá-los utilizando a capacidade de armazenamento da memória RAM. Mas se houver um "corte" na alimentação... adeus!

Agora fica mais fácil compreender por que razão existem os dispositivos de armazenamento em suportes magnéticos (cassete, discos). Eles nos permitem copiar o conteúdo da parte de RAM que nos interessa de forma que possamos desligar sem medo o computador e logo poder recuperar toda essa informação.

Na figura 8 ilustra-se a relação ROM/CPU durante a seqüência imediatamente posterior a um RESET. Supõe-se que o denominado Vetor de Reset está em $FFFE e $FFFF e que o programa de iniciação propriamente dito está na memória ROM, a partir da direção $F000 em diante. O processo será o seguinte: fase 0) liga-se o sistema e à CPU chega o impulso de Reset, passando esta linha ao nível lógico "1"; fase 1) situa-se o valor da direção "baixa" de Reset ($FFFE); fase 2) lê-se o dado contido nesta direção ($F0); fase 3) emite-se o valor da direção "alta" de Reset e que equivale à anterior + 1; fase 4) lê-se o conteúdo da célula de memória ROM correspondente ($00); fase 5) no interior da CPU juntam-se os dois de 8 bits em uma direção de 16 bits ($F000), que indica o verdadeiro começo, também em memória ROM, do programa de iniciação; e fase 6) passa-se ao contador de programa dita di-

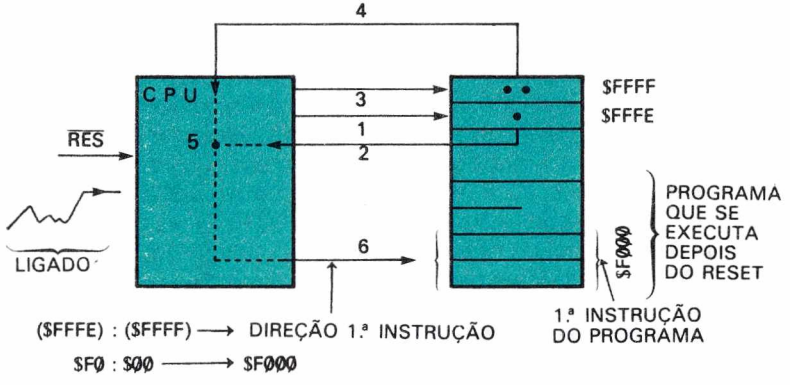


Fig. 8 - Seqüência de reposição ("reset") após ligar o computador.

reção e inicia-se a execução do programa de iniciação.

Para a explicação baseamo-nos na seqüência de Reset de uma conhecida CPU (6809), mas qualquer outra segue uma seqüência similar, ainda que as direções de Reset sejam diferentes e seus conteúdos também.

Esperamos que a função de uma memória ROM tenha ficado clara. No Glossário colocamos informação adicional sobre os diversos tipos disponíveis no mercado.

Já temos uma arquitetura baseada em CPU, RAM e ROM, mas sabemos como utilizá-la de maneira útil? Todavia não. Falta algo tão importante como para um automóvel podem ser as rodas: os dispositivos de entrada/saída.

Dispositivos de entrada/saída: a conexão com o mundo externo

Os dispositivos de entrada/saída (E/S) são circuitos através dos quais a CPU pode emitir ou receber sinais para ou desde o exterior. Tais sinais costumam ser do tipo binário porque são inúmeros, na prática cotidiana, os dispositivos externos ao computador que podem ser controlados, com facilidade, mediante sinais do tipo "tudo ou nada". Assim, por exemplo, como dispositivos de saída podemos citar os relés, eletro-válvulas e diodos LED; como dispositivos de entradas, os comutadores, interruptores de fim de carreira, sensores, etc.

Para compreender como funciona e como está constituído um dispositivo E/S, podemos estender a imagem que tínhamos de nossa conhecida memória RAM, com uma única diferença fundamental: quando escrevemos em uma direção pertencente a um dispositivo de E/S, um circuito particular do dispositivo (não presente na memória RAM normal) memoriza o conteúdo da célula e o reproduz nas linhas de sinal correspondentes; estas são as que saem fora da cápsula e as que podem ser conectadas a todos os dispositivos externos, tais como diodos LED, relés ou qualquer outro que possa ser controlado por sinais digitais do tipo "tudo ou nada".

Na Figura 9 ilustra-se esquematicamente o interior de um chip de saída com uma única célula para conter os valores dos 8 sinais de saída. Reconhecemos o bus de dados, as linhas de direções que controlam o decodificador interno (habilitado por sua vez pelo sinal Chip Select) e finalmente, a única célula existente que, igual a um RAM de um só byte, pode ser escrita e posteriormente lida (assim poderemos revisar o que já escrevemos). O homenzinho simboliza o circuito que extrai o conteúdo da célula e o reproduz nas pa-

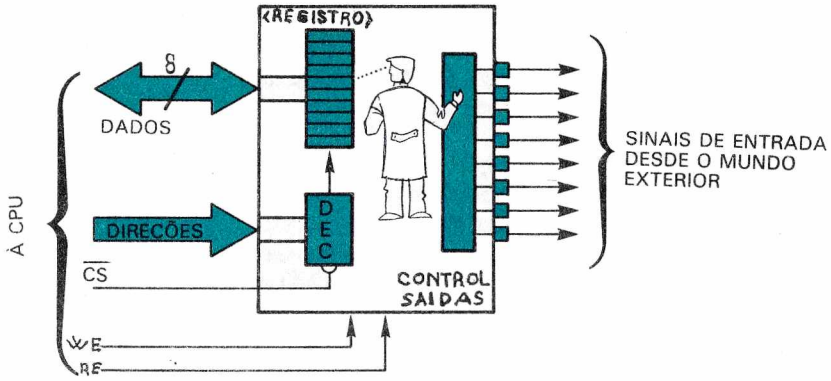


Fig. 9 - Representação esquemática do interior de um chip de saída. O homenzinho que copia o dado desde o registro às linhas de saída em paralelo, sintetiza as funções realmente desenvolvidas por uma lógica digital sofisticada.

tilhas de saída, cujo estado mudará cada vez que a CPU escrever um novo dado no chip.

A Figura 10 mostra um dispositivo de entrada, análogo ao de saída visto com a ressalva de que agora os terminais recebem sinais (também do tipo "tudo ou nada") do mundo exterior. Quando a CPU dá ordem de leitura ao chip, o homenzinho copia o estado daś patilhas na única célula existente, e é o conteúdo desta o que passa ao bus de dados.

As células dos dispositivos E/S costumam ser pouco numerosas já que a cada uma corresponde, pelo geral, um conjunto de 8 terminais ("Port") e a cápsula não costuma ter mais de 40. O nome clássico das células dos chips de E/S é "registro". Assim, para um dispositivo de saída teremos pelo menos um "registro de saída" e para um dispositivo de entrada haverá um "registro de entrada".

É muito comum encontrar dispositivos de entrada e saída em um só chip. Então, o valor que introduzirmos no registro "A" indicará qual das linhas é entrada e qual saída conforme o que seu bit associado indica (1 ou 0). Em outra célula poderemos escrever dados nas posições associadas à linhas definidas como saídas ou então ler os dados existentes nas linhas definidas como entradas. O funcionamento concreto, em todo caso, depende do tipo de dispositivo e do fabricante.

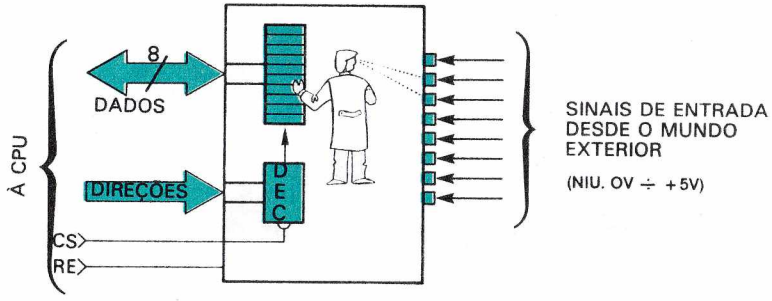


Fig. 10 - Ilustração do interior de um chip de entrada.

Duas opções: em paralelo ou em série

Pelas descrições do item anterior pode-se deduzir que os "zeros" e os "uns" do dado escrito na saída (ou lido na entrada) chegam agrupados de 8 em 8 (em paralelo). Uma operação de escrita do dado binário 10101010, colocará os oito terminais de saída do circuito integrado, respectivamente, às tensões de +5V, 0V, +5V, etc., em uma só operação. Ao "Port" (recordemos que Port = conjunto de 8 linhas de E/S) de saída poderemos conectar simultaneamente 8 fios controlando 8 dispositivos diferentes, ou então conectá-lo ao "Port" de entrada de outro computador e assim as duas máquinas poderão comunicar-se entre si...

No entanto, manejar tantos cabos pode ser incômodo e caro. Por este fato a aplicação dos dispositivos de E/S munidos de "Ports" em paralelo estão limitadas às transmissões de sinais a curta distância e ao controle de sensores em aplicações industriais.

Além dos 3 ou 5 metros se preferem normalmente os dispositivos de E/S em série, que permitem a comunicação com um só fio (mais a massa comum, claro). Desta forma limita-se o custo do cabo de conexão e por ser só um podemos garantir uma boa proteção frente às perturbações elétricas, coisa que, no caso anterior por ser 8 ou mais os fios, resultaria num custo muito alto. Na figura 11 se ilustra a forma em que se produz a transmissão em série. No dispositivo tem uma célula (registro "TX"), de onde a CPU somente pode escrever. Uma vez terminada a escrita (que se realiza de forma análoga à de uma memória RAM normal), nosso homenzinho acionará o interruptor "S", que controla o único terminal de saída do circuito integrado. A existência do "homenzinho"

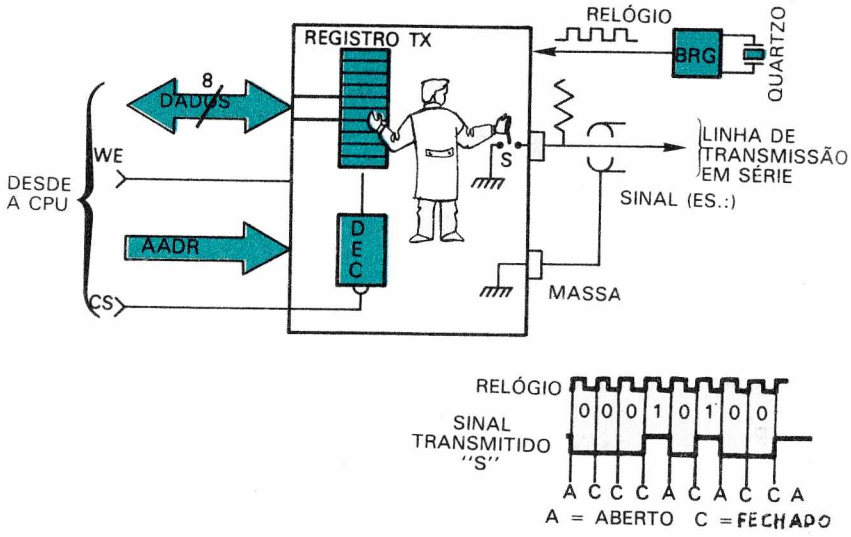


Fig. 11 - Esquema do interior de um dispositivo de transmissão em série

se deve ao desejo de evitar falar desnecessariamente neste momento dos registros de deslocação, circuitos flip-flop, contadores, etc., circuitos que são usados para lograr a transmissão, mas cuja missão somente entorpeceria a compreensão do processo.

Voltando ao exemplo, observará que se S está aberto, a linha vai a "1" (tem uma resistência, chamada de pull-up = manter a tensão, conectada ao +), enquanto que ao fechar S a linha vai a "0". Conseqüentemente, mediante a comutação adequada de S é possível gerar formas de onda quadrada à vontade. O sinal fundamental nos chips de E/S em série e que não existe nos demais, é um relógio. Procede de um circuito externo especial e consiste em um oscilador de quartzo denominado "Baud Rate Generator" (gerador de velocidade de transmissão ou de baud). Sendo assim, o baud equivale a um bit por segundo (bps), o qual significa que a velocidade (Baud rate) a que nos referimos é a de transmissão em série do dado no único fio, bit após bit. O sinal de relógio do gerador obriga a que a lógica de controle se sincronize com a máxima precisão, comutando o interruptor S (quando responde) somente em correspondência com as fases descendentes do próprio relógio. Ve-

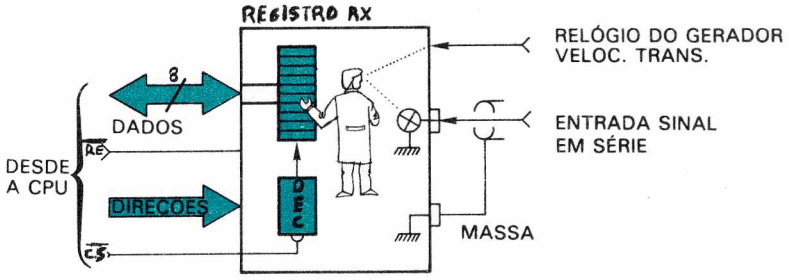


Fig. 12 - Representação do interior de um dispositivo de recepção em série.

remos um exemplo para esclarecer as coisas.

Se a CPU escreve no registro TX o dado binário 00010100 ($14), o comutador S deverá acionar-se de modo que coloque a linha de saída a "zero" durante dois períodos de relógio, logo a "um" durante um período, a "zero" durante outro período, a "um" durante um período e, finalmente, a "zero" durante três períodos. Depois de oito ciclos de relógio, o dado foi transmitido completamente através da única linha de saída. (Como foi vista, a transmissão começa pelo bit menos significativo, ou de menos peso, que é o situado mais à direita).

O dispositivo que pode receber e interpretar de forma correta um dado transmitido em série, se representa de maneira esquemática na Figura 12. O sinal de entrada ativa uma lâmpada que nosso homenzinho observa atentamente uma vez que verifica o relógio de velocidade de transmissão. Quando tem uma mudança neste, o homenzinho aponta no registro de recepção (RX) um 1 (lâmpada acesa) ou um 0 (apagado) no kit correspondente, enchendo o registro da direita (bit menos significativo) à esquerda (mais significativo).

Para que a interpretação do dado seja a correta é preciso que o relógio do gerador BRG seja idêntico para ambos dispositivos (emissor e receptor), aqui há necessidade de empregar osciladores estáveis e, por isto, de quartzo. À medida que se obtém "uns" e "zeros" na entrada, no sincronismo com o relógio, o registro de recepção RX se enche, como explicamos anteriormente, começando pelo bit 0 (o menos significativo) que, por sua vez, se transmitiu primeiro. Transcorridos oito ciclos de relógio de transmissão/recepção o registro está completo e a CPU como ler o mesmo como uma célula normal. As velocidades de transmissão normais va-

riam de 300 a 9600 bits por segundo.

Alguns destes dispositivos de E/S também incluem vários integrados no mesmo chip, especialmente o gerador de BRG, pelo que o desenhador somente tem que acrescentar o oscilador de quartzo e conectar o chip à CPU, sem mais problemas.

Algo mais faz falta?
Do hardware ao software

Se você até aqui seguiu sem problemas a leitura do livro poderá afirmar que já conhece tanto o funcionamento geral da CPU como o dos circuitos integrados de apoio: RAM, ROM e E/S em série e em paralelo. Saberá avaliar a importância das memórias ROM colocadas em marcha no computador pessoal, das memórias RAM na retenção temporial de dados e a dos dispositivos de E/S na comunicação em ambos sentidos com o mundo exterior.

Ainda assim, todo este hardware, cuja descrição (a um primeiro nível) consideramos concluída, continuará sendo uma grande massa inerte de custoso silício enquanto não tiver um programa armazenado no lugar oportuno da memória ROM. Efetivamente, para tirar proveito do hardware, é preciso dispor de um programa (software) que lhe indique o que nós queremos que faça: por onde pegar os dados, o que fazer com eles, como apresentar-nos os resultados...

Nos capítulos seguintes iremos ver de que forma será que o software nos ajudará a "personalizar" nosso computador pessoal.

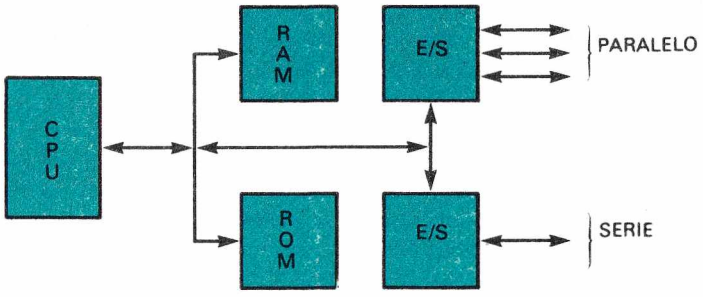


Fig. 13 - Estrutura completa do nosso microcomputador.

KBYTES: Unidade de medida da quantidade de memória de um computador. Cada K (se não se diz ao contrário K pode-se ler Kbytes) equivale a 1024 bytes (grupos contíguos de 8 bits). Uma CPU normal de 8 bits, com um bus de direções de 16 linhas, pode direcionar um máximo de 64K (65.536 bytes).

HEXADECIMAL: Anotação que se emprega ao manejar dados binários para não "marearmos" com tantos "zeros" e "uns". A anotação hexadecimal faz uso de 16 dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E e F. Cada dígito hexadecimal representa uma das 16 combinações possíveis com quatro bits. Por exemplo: 10100010 equivaleria ao número hexadecimal A2 (normalmente se põe o sinal $ diante — $A2 — para indicar que se trata de um número hexadecimal). Para vê-lo, simplesmente dividimos em grupos de 4 bits o número binário (começando pela direita) e substituimos cada um por seu equivalente hexadecimal. No capítulo 5 daremos uma descrição mais detalhada deste sistema de numeração.

DECODIFICADOR: Bloco lógico com um número de saídas que, como máximo, é a potência de dois do número de entradas. Por exemplo, se tem 4 entradas, o máximo número de saídas será $2^4 = 16$. Sua função é reconhecer, de forma inequívoca, qualquer combinação das entradas, ativando a única saída correspondente a tal combinação. Os decodificadores se utilizam, fora da CPU, para obter os sinais de habilitação dos diversos chips de apoio (RAM, ROM, etc.).

VOLATIL: Memória que perde todo seu conteúdo quando deixa de aplicar-se a alimentação. Típico exemplo são as memórias RAM.

NÃO VOLATIL: É um tipo de memória como a RAM, mas que, em determinadas condições, e inclusive sem alimentação, não perde o conteúdo de suas células (veja NOVRAM).

ROM: Abreviatura de Read Only Memory (Memória só de leitura). Nome genérico de uma memória que somente pode ser lida, não escrita, pelo usuário. Seu conteúdo se determina durante o processo de fabricação mediante gravação "por máscaras".

PROM: É como a anterior, com a ressalva de ser programável pelo usuário mediante umas máquinas denominadas, logicamente, "programadores de PROM". Uma vez programada não pode modificar-se seu conteúdo e teremos de substituí-la por outra PROM se precisarmos modificar o programa.

EPROM: Memória PROM apagável (Erasable-PROM). É como a

PROM, mas pode se apagar expondo a certa quantidade de raios ultra-violetas em uma janelinha transparente da cápsula. Conseqüentemente podemos reprogramá-la. É mais cara que uma ROM, mas permite uma notável economia na fase de colocação em ponto e comprovação (com as modificações correspondentes) de um programa.

EEPROM: "Electrically-Erasable-PROM" (PROM apagável por meios elétricos). De características similares à EPROM, se pode apagar com maior comodidade do que esta, já que basta aplicar impulsos elétricos em um de seus terminais para apagar seu conteúdo. Lamentavelmente trata-se de um chip de alto custo e seu uso está limitado a certos fins específicos.

NOVRAM: "NOn-Volatile-RAM" (Memória RAM não volátil). Trata-se de uma memória RAM que contém também uma quantidade idêntica de EEPROM. Quando cai a alimentação, o conteúdo da memória RAM se copia na EEPROM e quando volta a alimentação ocorre o contrário: o conteúdo da EEPROM se copia na RAM e tudo fica como antes. São chips ideais contra as perdas de dados, mas seu custo é muito elevado.

SISTEMA OPERACIONAL: Programa original, incluindo normalmente nas memórias ROM do computador pelo fabricante, que controla o funcionamento correto do computador em suas relações com o usuário e com os periféricos.

VETOR DE RESET: Se denomina "vetor" a um par de posições contíguas de memória cujos valores formam outra direção. No UP o vetor de Reset é o par de células nas quais a CPU buscará, imediatamente depois do Reset, a direção efetiva onde começa o programa de iniciação.

REGISTRO: Nome clássico de uma célula genérica de um dispositivo de E/S ou similar. Se aplica especialmente a células muito específicas da CPU ou dos dispositivos de E/S (por exemplo, "registro de dados" na CPU).

PORT: Nome que se dá ao bloco de 8 terminais de entrada (port de entrada) ou saída (port de saída) de um chip de E/S.

BAUD: Unidade de medida da velocidade de uma transmissão em série e equivalente a um bit por segundo.

SINGLE-CHIP (MICROCOMPUTER): Microcomputador em um só

chip. Se trata de um circuito integrado que contém uma CPU, dispositivos de E/S e memória RAM e ROM, todo ele integrado no mesmo chip de silício. Se utiliza, sobretudo, em aplicações industriais nas quais se construam muitas unidades e que precisam uma grande quantidade de memória ROM (por exemplo, os microcomputadores que controlam lavadoras, elevadores, etc.).

CAPÍTULO IV

SOFTWARE: O COMBUSTÍVEL DO COMPUTADOR

Se amplo é o tema do hardware, maior é o do software. Podemos dizer inclusive que, na medida certa, o segundo é mais importante que o primeiro, porque enquanto as tecnologias de construção melhoram dia a dia, não está tão fácil conseguir "experts" que saibam aproveitar ao máximo toda a potência destas máquinas tão complexas. A título de exemplo, empregar um mau programador com um computador bom seria o mesmo que confiar a um automobilista comum a direção de uma Ferrari de Fórmula 1.

Os primeiros computadores eram máquinas mostruosas concebidas para facilitar o maior número de realizações de cálculos científicos complexos e sua capacidade era comparável a de algumas das atuais calculadoras de bolso. O programa, isto é, a seqüência de operações a executar, eram introduzidas por meios técnicos manuais especializados dada a complexidade do processo. Em resumo, era um ambiente reservado a alguns poucos eleitos e, certamente, tanto no fundo como na forma, muito pouco "user friendly" (amistoso com o usuário) como se diz na atualidade, pois entre outras coisas trabalhava-se quase constantemente em binário.

Mais adiante, o célebre cientista Von Neumann teve a brilhante idéia de considerar os códigos de operação da máquina equivalentes, senão no fundo ao menos na forma aos próprios dados.

Então se levantou a hipótese de porque não podiam estar ambos (códigos e dados) juntos na memória do computador "monstro". Deste modo, proporcionavam de imediato ao computador junto aos dados, a lista de operações, deixando-o livre para prosseguir de forma automática com a execução dos cálculos. Ainda que isto nos soe hoje como "os contos da vovozinha", não faz muito tempo que ocorreu e nos mostra como era difícil, e continua sendo, dialogar com uma máquina que carece de programa adequado que a transforme de um monstro inerte e estúpido em um colaborador sempre a nossa disposição.

Sem aprofundar demasiadamente nos conceitos de programação (próximos volumes desta coleção), este livro quer mostrar como é possível, graças ao software, fazer funcionar o hardware descrito anteriormente e conseguir finalmente uma máquina disposta a prestar toda classe de serviços e com toda obediência. Tomaremos, pois, como base, uma arquitetura muito simples tal como a ilustrada na figura 1, com um dispositivo de entrada ao qual está conectado um teclado e um dispositivo de saída unido a uma

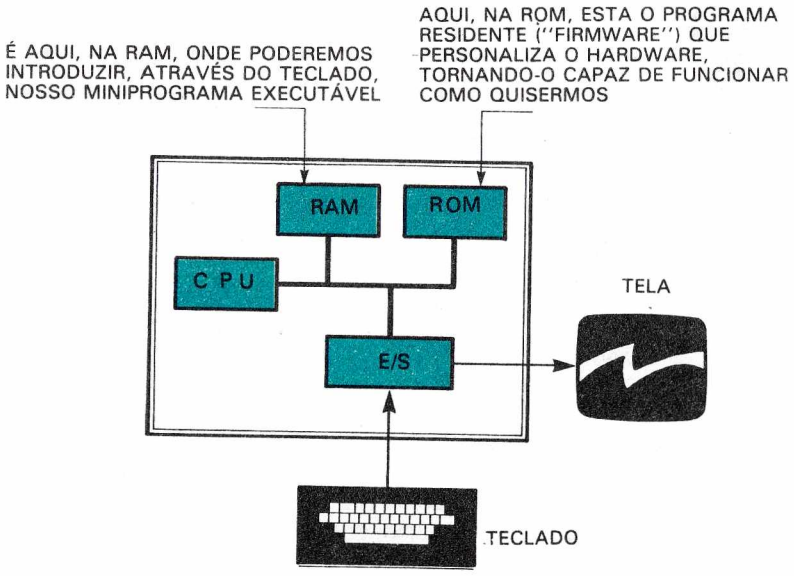


Fig. 1 - O simples computador pessoal que estamos construindo. Controla um teclado de 8 teclas e uma tela. Pode compreender e executar somente uma única instrução:< ?> (PRINT)

tela, além de uma determinada quantidade de memória RAM e ROM. Na ROM deveremos introduzir o programa que fará funcionar o computador no momento que o ligármos de modo que possamos dialogar com o mesmo permitindo-lhe compreender uma simples instrução de "?" (como a instrução "PRINT" do BASIC), e que indique a existência de um erro se introduzirmos o programa de modo ambíguo. É um exemplo muito simples e com muitas limitações, evidentemente, mas servirá para fazer-nos compreender como, a partir do hardware e graças ao software, se chega a uma máquina perfeitamente "funcional".

Os próximos itens serão, indubitavelmente, mais difíceis de seguir para aqueles que não tenham experiência alguma em programação, nem sequer em BASIC, mas tentaremos fazer com que, inclusive nesses casos, o texto seja inteligível. Quaisquer dúvidas que surgirem poderão ser dissipadas nos próximos livros desta coleção ou consultando os livros indicados na bibliografia.

O SOFTWARE DO HARDWARE

Pedimos perdão pelo jogo de palavras, mas desejamos insistir uma vez mais em que nosso hardware deve possuir seus próprios recursos de software, isto é, de programas. Isto significa que as instruções dos programas devem estar em um dos dispositivos de apoio que descrevemos no capítulo anterior: em memória ROM e RAM, cada uma para coisas diferentes.

Na memória ROM introduziremos o "software residente", denominado também "firmware", programado sobre ela na fábrica. O chip será colocado no cartão jutamente com o resto do hardware do computador. Na memória RAM, pelo contrário, o usuário irá introduzir seu próprio programa, uma vez ligada a máquina e executado o programa de iniciação.

Recordemos que programa é o nome genérico dado a uma seqüência de instruções, cada uma das quais não é outra coisa que um grupo de códigos binários que podem ser compreendidos pela CPU. Posto que a CPU executa as instruções uma pós a outra, o programa deve ser necessariamente compacto, isto é, os códigos de todas as instruções devem estar em células contíguas na memória. No caso do programa ter distintas partes separadas e ficar "em pedaços", ao final de cada segmento de programa deverão existir as convenientes instruções que digam à CPU "não continue na direção seguinte e veja a xyzw, onde está o próximo segmento do programa". (São as chamadas instruções "de salto").

Vejamos novamente o que acontece quando conectamos o computador. Em primeiro lugar se executa a seqüência de "Reset", descrita no capítulo anterior, o que exige a presença de, no mínimo, uma memória ROM com as primeiras instruções de tal seqüência. E depois, o que acontece?

A resposta é simples: a CPU prossegue a execução de outro programa também residente na memória ROM, que "personaliza" o hardware obrigando-o a comunicar-se conosco tal e como se descreve em seu manual de manuseio. Não faz sentido explicar agora em detalhes, as operações que podem ser realizadas em uma clássica seqüência de iniciação. Bastará dizer que costumam ser uma série de leituras e escritas em algumas zonas da memória e dos registros dos chips de E/S, com o objetivo de que todos os pontos neurálgicos do hardware fiquem com valores conhecidos, sobre os quais se baseiam os futuros processos. Vejamos um exemplo: no capítulo anterior falamos de dispositivos com linhas que poderiam ser entradas ou saídas de acordo com o valor 1 ou 0 de seu bit associado dentro de um registro. Pois bem, se esse dispositivo é usado pelo nosso computador, por exemplo, para controlar um teclado e uma tela, o programa de iniciação deverá carregar nesse registro, os valores adequados para que as linhas que, por hardware, unidas à tela sejam saídas e as conectadas ao teclado sejam entradas.

O fato de que normalmente o computador admita dados introduzidos através de um teclado e visualize os resultados em uma tela, nos dá idéia da importância destes, e é tanta que muitas vezes seu controle se realiza mediante chips de E/S especializados.

A figura 2, por exemplo, mostra a forma mais simples de "ler" um teclado alfanumérico. Na realidade, tal realização seria "desperdiçadora", pois não se dedica uma entrada individual do port de E/S para cada tecla, a não ser que se prefiram conexões matriciais mais complexas. Não obstante, para fazer o exemplo o mais simples possível admitiremos ter tantas entradas quantas teclas existam no teclado, limitando este número a um total de 8, visando assim a máxima simplicidade. Cada tecla é um pulsador que, pressionado, une a entrada do port à massa e ao soltar a tecla, a entrada voltará ao nível lógico "1" obrigada pela visível resistência de pull-up (colocada em tensão). As entradas estão agrupadas em grupos de 8, já que se supõe que cada port de entrada é de 8 bits.

Naturalmente, o chip de entrada ao que está conectado o miniteclado deve estar em alguma parte do mapa de memória, posto que a CPU deve ter a possibilidade de ler o port efetuando uma operação de leitura normal. Por comodidade, a direção do port pa-

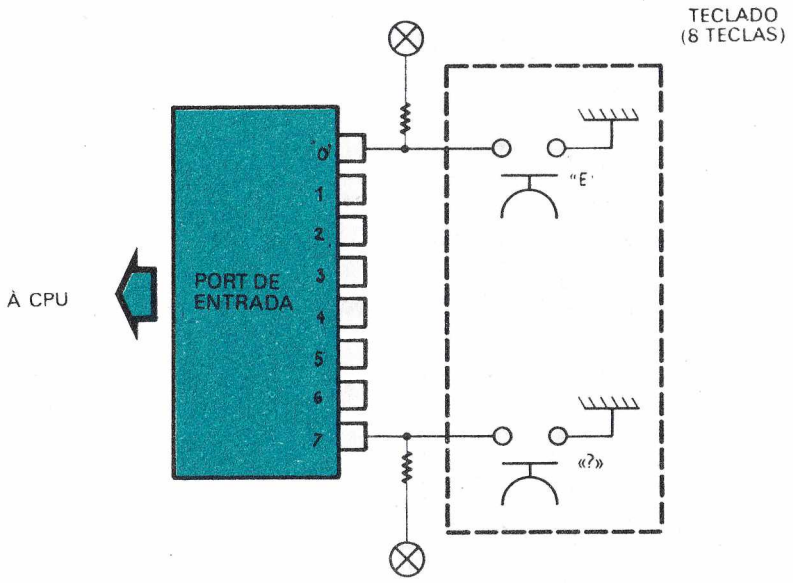


Fig. 2 - Conexão de um teclado no port de entrada do computador.

ra o teclado é $E000, e será $E100 a direção do port de saída do chip que controla a tela.

A figura 3 ilustra a disposição do mapa de memória no nosso computador. Observa-se que a memória RAM tem uma amplitude de 1.024 bytes (estamos com escassez de recursos...), de $0000 a $03FF, enquanto que mais acima estão os chips de E/S e logo, de $F000 até o final — isto é $FFFF —, tem 4.096 bytes de memória ROM (mais que suficiente, como será visto mais adiante). Voltando ao nosso pequeno teclado, na Tabela 1 se indica os nomes de suas teclas e o código resultante nos terminais de port quando se pressionam, tanto em forma binária como hexadecimal.

Colocamo-nos agora a imaginar qual pode ser a série de instruções que permite à CPU reconhecer a tecla que pressionamos. Temos presente que é, realmente, uma operação de leitura: a CPU emite uma direção e solicita ao dispositivo que se encontra nessa direção (ROM, RAM ou então E/S, como neste caso) que transmita o dado de 8 bits que contém. A sentença: LDA $E000 indica, no código "mneumônico" que resume a instrução, uma ope-

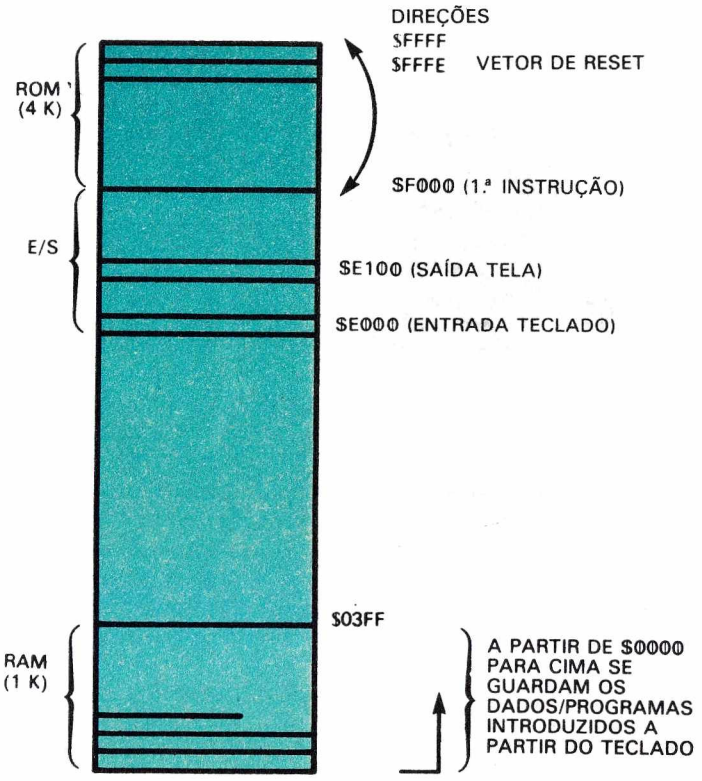


Fig. 3 - Mapa de memória correspondente ao nosso microcomputador.

ração de leitura do conteúdo de uma certa direção ($E000, neste caso). Os "mneumônicos" são as instruções de uma linguagem chamada Assembler, cuja única finalidade é estabelecer uma correspondência direta e única de cada código máquina com um "mneumônico", isto é, com uma palavra que nos recorda que é o que faz esse código com o qual vai associado; desta forma o usuário pode escrever primeiro o programa em assembler, sem ter que memorizar o significado de cada código e logo traduzir o programa assim escrito aos correspondentes códigos máquina. (Veja o Glossário para mais informações).

Por suposição, sabemos que o decodificar de instruções (ID) que existe dentro da CPU e que tem como função supervisionar a execução das instruções, não compreende a instrução "LDA", so-

TECLA	CODIGO BINARIO	CODIGO HEXADECIMAL
E	1 1 1 1 1 1 1 0	F E
R	1 1 1 1 1 1 0 1	F D
O	1 1 1 1 1 0 1 1	F B
A	1 1 1 1 0 1 1 1	F 7
B	1 1 1 0 1 1 1 1	E F
"	1 1 0 1 1 1 1 1	D F
«CR»	1 0 1 1 1 1 1 1	B F
?	0 1 1 1 1 1 1 1	7 F

Tabela 1 - Codificação das teclas do teclado.

mente códigos binários bem precisos. Ante esta circunstância, pegaremos o manual da CPU e, buscando o código que corresponde à instrução LDA, encontramos, por exemplo, 10101101 (ou melhor $AD). Escrevemos então a instrução, e o programa que ao final introduziremos na memória ROM vai convertendo-se em:

INICIALIZAÇÃO (seqüência de códigos que passamos por ato)

AD E0 00

Uma vez lido o Port, é preciso comprovar se tem uma tecla pressionada e esta operação é fácil, posto que se não há o dado lido será $FF, isto é 11111111 em binário. Se, pelo contrário, tem alguma tecla pressionada, haverá um "zero" em uma posição qualquer das oito lidas. Então, a solução mais simples e rápida consiste em admitir como código da tecla o valor lido (segundo se indica na Tabela 1). Para simplificar consideramos a priori que não é possível pressionar simultaneamente várias teclas, conseqüentemente, estamos seguros de que os códigos lidos são verdadeiramente os correspondentes das teclas pressionadas. A comprovação do valor lido pela CPU é facilmente realizável mediante uma instrução de comparação do tipo <CMPA#$FF>, que significa "compara o dado que acabou de ler (na instrução anterior) e que tem no acumulador com o valor que segue, isto é, $FF (11111111 em binário)". Consultando novamente o manual da CPU, encontramos que isto equivale para a CPU a:

C9 FF

e, conseqüentemente, nosso programa se converte agora em:

INICIALIZAÇÃO
AD E0 00
C9 FF

Depois da comparação é preciso ver qual foi o resultado. Em nosso caso, voltamos a tentar uma leitura se não encontramos nenhuma tecla pressionada, isto é, se o dado lido era igual a $FF. A instrução pode ser "JEQ-5", ou melhor "Se, e somente se, o valor lido é igual (EQual) ao de comparação, pula (Jump) cinco passos para trás (-5)". O resultado é que a CPU se veria obrigada a repetir a instrução de leitura que se inicia com "AD".

Deste modo, criamos um "loop" ou ciclo em nosso programa. Se repetirá indefinidamente até que pressionemos uma tecla. O programa é agora:

INICIALIZAÇÃO
AD E0 00 (Carrega o valor de E000)
C9 FF (Compara-o com $FF)
F0 -5 (Se é igual pula de novo, a $AD. Reservamos para mais adiante a tradução em binário de —5).

Como pode obervar, com somente três instruções traduzimos em códigos interpretáveis pela CPU este raciocínio: "continue lendo a direção do Port de entrada, ao qual está conectado o teclado, até que encontre uma tecla pressionada". É possível, seguindo com este procedimento (RACIOCÍNIO — OPERAÇÕES NECESSÁRIAS — CODIFICAÇÃO BINÁRIA DAS INSTRUÇÕES) desenvolver todo o programa na forma requerida?

Sem dúvida que sim, mas também é alta a probabilidade de introduzir erros, porque se perde de vista a "lógica" que o programa deve seguir. Chegou o momento, antes de completar nossa obra, de tratarmos de analisar a figura 4 e compreender sua utilidade.

Diagramas de fluxo

Como muitos de nossos leitores possivelmente saibam, um diagrama de fluxo é uma representação gráfica que expressa com os símbolos adequados, a sucessão de operações necessárias em um programa, simplificando sua compreensão e facilitando as tarefas de conservação e modificação posteriores.

A figura 4 ilustra um diagrama de fluxo, relativo ao miniprograma que estamos escrevendo. Observará que em uma só figura está condensada toda a série de operações necessárias para o bom funcionamento do programa, descritas sem que apareça nenhuma instrução em código máquina (pois não pretendemos em

absoluto utilizar este esquema para codificar as instruções na linguagem máquina de uma CPU específica). Passemos a ler o diagrama.

Se inicia com um retângulo, um símbolo freqüentemente utilizado para reagrupar uma série de instruções que serão executadas de forma seqüencial. De fato, o primeiro retângulo simboliza a totalidade das instruções preliminares dedicadas à iniciação da máquina, situação que tomamos como ponto de partida.

A leitura prosegue no sentido das flexas e se chega ao segundo retângulo, que indica uma operação de leitura, precisamente do teclado.

A terceira figura é um losango, símbolo "de decisão" que indica uma "prova", isto é, a comprovação de um valor. O bloco tem duas saídas: uma se a prova deu resultado afirmativo e a outra em caso contrário. Em nosso exemplo, até que não haja alguma tecla pressionada o caminho a seguir é sempre o da flecha que sai por "não", pelo que voltaremos ao retângulo que indica a leitura do teclado. É muito clara a existência do anel de repetição ("loop") ao qual fizemos referência enquanto escrevíamos as primeiras instruções do programa.

O que acontece se pressionármos uma tecla? Continuando com a interpretação do diagrama, se observa que o código da tecla se copia na memória RAM, a partir da direção $0000 para cima, e se reproduz na tela. Depois o programa solicita uma segunda prova para comprovar se a tecla pressionada era um "retorno de carro" ("Return"). Se não era, a saída "Não" do bloco de prova levanos novamente a uma nova leitura do teclado, com o que se cria um segundo "loop" que envolve ao anterior, menor. Se a tecla pressionada coincidir com o retorno do carro, considera-se que o usuário quer indicar que terminou de introduzir o programa e, por isto, o "loop" se encerra executando-se o bloco seguinte. Mas antes de continuar, podemos ter a curiosidade de saber onde irá parar o programa escrito pelo usuário e como chegou ali.

Pois bem, voltaremos a analisar o comportamento dos "loops" anteriores. Para cada pressionada de uma tecla, incluindo a de retorno de carro, os códigos correspondentes se guardaram sucessivamente na memória RAM a partir da direção $0000 em diante. O programa introduzido pelo usuário se lerá posteriormente nesta parte da memória do sistema, e estará constituído por uma sequência de códigos, que são os que atribuimos arbitrariamente às 8 teclas (ver Tabela 1).

O programa escrito pelo usuário pode estar constituído por códigos que não tenham sentido para a CPU. Para que possam ser

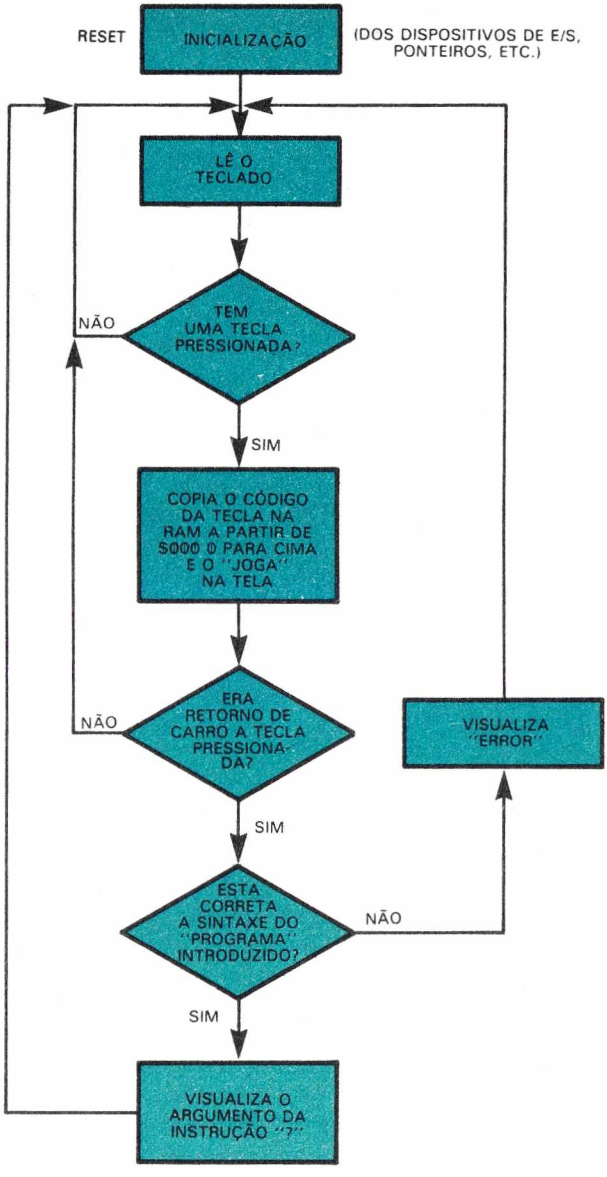


Fig. 4 - Diagrama de fluxo do programa que controla nosso microcomputador.

compreendidos por tal unidade é preciso que ela mesma esteja executando outro programa (mais "profundo") que lhe sirva de intérprete entre os códigos empregados pelo usuário e os que a CPU maneja.

Em outros termos, isto significa que deveremos escrever utilizando o código de máquina próprio da CPU, um programa "intérprete" graças ao qual poderemos escrever outros programas cujas instruções utilizarão códigos completamente diferentes aos da máquina. A razão disto é simplesmente poder usar códigos que "nos digam algo" sobre a operação a realizar, coisa que não ocorre com os que emprega a máquina (AD, C9, F0,...). Ditos programas "intérpretes" denomina-se, em geral, "linguaguens". Se estabelece assim um "duplo nível" na execução dos programas que escrevemos quando trabalhamos com nosso computador pessoal. Por exemplo, se nós programamos empregando uma linguagem simples, a BASIC, cujas instruções estão constituidas por palavras fáceis de escrever e recordar, este programa, escrito em BASIC e introduzido na memória do computador pessoal através do teclado, é "interpretado" por outro programa residente no computador, geralmente em uma memória ROM produzida pelo mesmo fabricante do computador pessoal (em outros casos, se carrega em memória RAM a partir do disco ou da fita). O resultado desta interpretação é, finalmente, a seqüência de instruções em código máquina que são executáveis pela CPU.

Os últimos toques

A figura 5 mostra a interface de vídeo que necessitamos. O trataremos como um circuito integrado normal de E/S. Mais adiante nos aprofundaremos no funcionamento dos terminais, mas agora nos contentaremos em saber que a CPU, para apresentar um caracter na tela, simplesmente efetua uma operação de escrita em um registro concreto do chip de E/S dedicado ao controle da tela. As complexas funções internas do chip de controle supomos se preestabelecerão durante a fase de iniciação. Se agora revisarmos no diagrama de fluxo as operações realizadas pelo bloco sinalizado com o asterisco, observamos que, apenas se detectou o acionamento de uma tecla. O código que a descreve não somente se copia em outra parte da memória, como também se introduz no chip de vídeo. Desta maneira teremos imediatamente a visualização de todas as teclas pressionadas. Quando pressionamos a tecla "Return" (CR, Retorno do carro) indicamos que acabamos a

introdução de códigos, portanto a máquina pode começar a traduzir (se puder) as ordens distribuidas. Em nosso caso definiremos uma só instrução, a "?", para a qual estabelecemos que a sintexe válida é a seguinte:

<?"xyzw..wxz(CR)>. Dito de outro modo, depois da tecla<?> deve pressionar-se a tecla<">, que indica o começo da cadeia alfanumérica que deverá ser escrita na tela. A cadeia termina com o CR, isto é, com o retorno do carro ou "Return" do teclado. Qualquer outra combinação de teclas pressionadas, a consideraremos não válida, e o computador deverá indicar-lhe com a presença visual de mensagem <ERROR>.

Nosso microsistema operacional é muito simples e não leva em conta muitas das situações que poderiam ocorrer, portanto poderíamos dizer que não está todavia "purificado". Porém também não pretendemos chegar mais além do que é o exemplo, pois então consideraremos o diagrama de fluxo e sua tradução em linguagem de máquina (Tabela 2) tal como está, sem colocar-nos em complicações.

Na Tabela 2 completamos a escrita do nosso software residente, utilizando uma linguagem de máquina "semi-inventada", com o objetivo de dar às instruções nomes fáceis de compreender. Vamos tentar ler estas instruções.

A fase de INICIAÇÃO foi examinada anteriormente. O "verdadeiro" programa se inicia com a colocação de um ponteiro a zero (instrução CLX, que se verá dentro em pouco). Encontramos logo no começo do primeiro "loop" dedicado a vigiar o teclado até que se pressione uma tecla. Quando pressionar o "loop" se interromperá e o código da tecla se armazenará na memória, "a partir da posição $0000 para cima".

Essa operação se executa com uma instrução de "escrita com ponteiro". O ponteiro (X) é outro registro interno da CPU (normalmente existe mais de um), que pode colocar-se a zero e incrementar-se de 1 em 1 com uma instrução INX (Incrementa o ponteiro X). A instrução de escrita com ponteiro, a "STAX", comunica à CPU o seguinte: "guarda o valor que tem no acumulador na célula de memória, cuja direção será dada ao somar o valor que dou em seguida — direção "base" — com o conteúdo do registro ponteiro "X".

Em nosso programa, vemos que a direção "base" é sempre $0000, pelo que X é 0, como ocorre depois de uma instrução de colocada a zero (CLX), a instrução STAX $0000 equivale a STA $0000; se, pelo contrário, X contém outro valor (1, 2, 3, etc.), a mesma instrução STAX $0000 equivalerá respectivamente a: STA

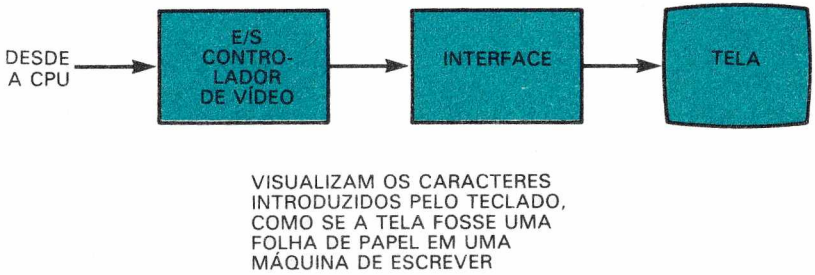


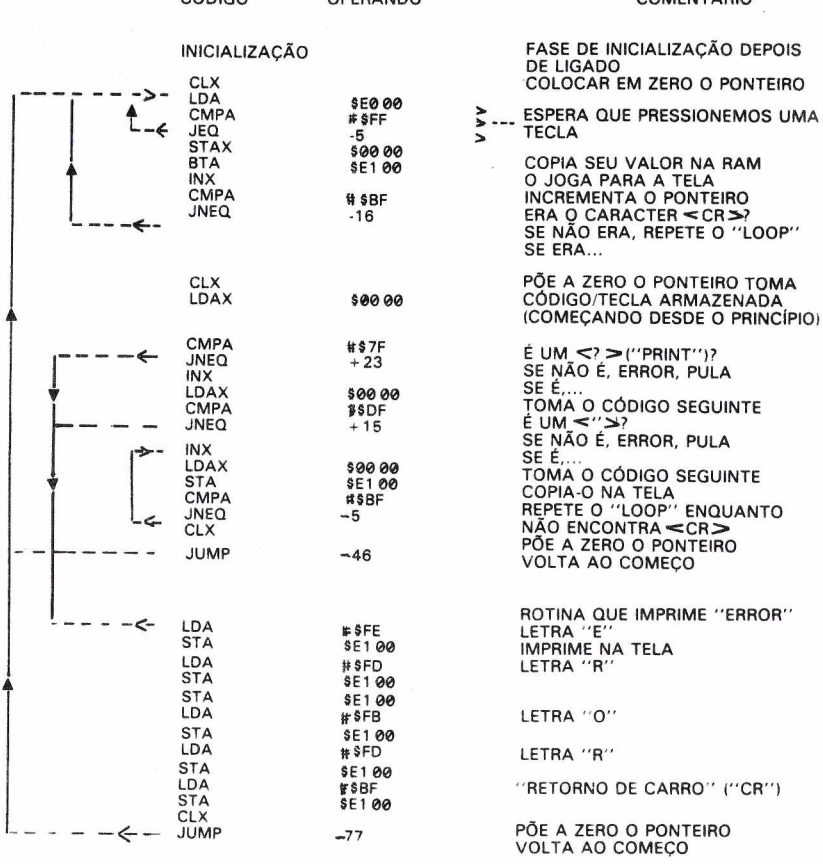
Fig. 5 - Conexão da tela, manejada por um chip controlador de vídeo especial que, ante a CPU, aparece como um port de saída normal.

$0001, STA $0002, STA $0003, etc. Deste modo, se pode simplificar notavelmente a escrita de um programa. Agora podemos compreender para que servia a instrução CLX situada no começo.

Depois de copiar o código na direção de RAM correspondente (neste primeiro momento seria $0000), copia-o em $E100 para tirá-lo da tela. Depois faz INX para elevar em 1 o conteúdo de X (agora STAX $0000 equivalerá a STA $0001, com o qual não apagou o dado que antes guardou em $0000).

Este processo irá repetindo-se enquanto pressionármos qualquer tecla que não seja um retorno de carro <CR>. Este fato é comprovado na instrução CMPA # $BF. Dita comparação vai seguida por um salto condicional, que produz a repetição do "loop" se a tecla não foi um retorno de carro, pois se o código guardado no acumulador não é $BF salta 16 bytes para cima no programa. Se chegar dito retorno de carro põe novamente a zero o ponteiro (CLX) e logo começará a ler o conteúdo da memória RAM a partir de $0000 para cima (instrução LDAX $0000; equivalente em seu mecanismo à STAX, mas em modo leitura), comprovando se a sintaxe era correta. Esta é a fase de interpretação propriamente dita do programa introduzido pelo teclado.

Conforme o previsto, deveremos encontrar em primeiro lugar o código da única instrução admitida, a <?>, que é $7F. Se não for assim, cometemos um erro e se produzirá o salto JNEQ, que significa "salta" (Jump) se os valores comparados antes NÃO são iguais (EQual)". Aqui vemos que se salta 23 passos para adiante. Se o resultado da prova for positivo (tem um $7F) se incrementa o ponteiro para poder ter acesso à direção seguinte ($0001) utilizando também a instrução LDAX $0000. O valor deste segundo



CÓDIGO	OPERANDO	COMENTÁRIO
INICIALIZAÇÃO		FASE DE INICIALIZAÇÃO DEPOIS DE LIGADO
CLX		COLOCAR EM ZERO O PONTEIRO
LDA	$E0 00	
CMPA	#$FF	ESPERA QUE PRESSIONEMOS UMA TECLA
JEQ	-5	
STAX	$00 00	
BTA	$E1 00	COPIA SEU VALOR NA RAM
INX		O JOGA PARA A TELA
CMPA	#$BF	INCREMENTA O PONTEIRO
JNEQ	-16	ERA O CARACTER <CR>?
		SE NÃO ERA, REPETE O "LOOP"
		SE ERA...
CLX		PÕE A ZERO O PONTEIRO TOMA
LDAX	$00 00	CÓDIGO/TECLA ARMAZENADA
		(COMEÇANDO DESDE O PRINCÍPIO)
CMPA	#$7F	É UM <?>("PRINT")?
JNEQ	+23	SE NÃO É, ERROR, PULA
INX		SE É,...
LDAX	$00 00	TOMA O CÓDIGO SEGUINTE
CMPA	#$DF	É UM <">?
JNEQ	+15	SE NÃO É, ERROR, PULA
INX		SE É,...
LDAX	$00 00	TOMA O CÓDIGO SEGUINTE
STA	$E1 00	COPIA-O NA TELA
CMPA	#$BF	REPETE O "LOOP" ENQUANTO
JNEQ	-5	NÃO ENCONTRA <CR>
CLX		PÕE A ZERO O PONTEIRO
JUMP	-46	VOLTA AO COMEÇO
LDA	#$FE	ROTINA QUE IMPRIME "ERROR"
STA	$E1 00	LETRA "E"
LDA	#$FD	IMPRIME NA TELA
STA	$E1 00	LETRA "R"
STA	$E1 00	
LDA	#$FB	LETRA "O"
STA	$E1 00	
LDA	#$FD	LETRA "R"
STA	$E1 00	
LDA	#$BF	"RETORNO DE CARRO" ("CR")
STA	$E1 00	
CLX		PÕE A ZERO O PONTEIRO
JUMP	-77	VOLTA AO COMEÇO

Tabela 2 - Programa que controla nosso hardware, escritonos mneumônicos da linguagem do assemble de uma CPU imaginária (que se parece um pouco às 6502, 6809, etc.). Junto às instruções está o comentário correspondente.

código deve ser<">, que indica o começo da cadeia a visualizar. Outra prova, e outro salto JNEQ, se não é o código requerido.

Pelo contrário, se tudo é correto, se iniciará um novo "loop" que tem por objetivo passar ao chip de E/S correspondente à apresentação visual, toda a seqüência de códigos armazenados em RAM, que se encontram depois dos primeiros códigos de instru-

ção, para sua visualização na tela. O "loop" termina quando se encontra o código de retorno de carro (CR), depois do qual se põe a zero o ponteiro e o programa se reinicia desde o começo (observe que seria o mesmo não colocar o CLX, pois nesse caso JUMP-46 produziria um saldo ao primeiro CLX do programa). A máquina, conseqüentemente, ficará à espera de um novo ciclo de acionamento de teclas.

As últimas linhas do programa procedem à visualização de mensagem "ERROR", ao término do qual voltará ao começo com o ponteiro colocado a zero.

Parece complicado? Prove comparando a figura 4 (diagrama de fluxo) e a Tabela 2 (programa escrito em linguagem Assembler), que verdadeiramente, não são tão diferentes.

Até agora não fizemos outra coisa senão colocar em prática o seguinte procedimento:

1) Ter as idéias claras sobre o que exigir ao hardware incorporado.

2) Desenvolver um memorando de funcionamento global da máquina.

3) Realizar o diagrama de fluxo das operações a efetuar.

4) Traduzir o diagrama a uma seqüência de instruções escritas em linguagem Assembler de uma suposta CPU.

Os passos que nos faltam são:

5) Codificar o programa, instrução por instrução, para obter a seqüência de códigos máquina (binários), executáveis pela CPU.

6) Programar uma memória ROM com os códigos antes citados, de forma que constituirá nosso programa residente (firmware) e inserí-la em seu lugar dentro da placa fabricada.

7) Ligar a máquina e verificar que o programa funciona de forma correta. Se não for assim, será preciso encontrar os erros e repetir o processo desde o ponto 3.

Mediante o manual de programação da CPU utilizada podemos traduzir cada instrução da Tabela 2 em seu código binário correspondentes (hexadecimal, para ter mais comodidade na escrita) e como resultado obteremos algo "parecido" (tudo dependerá de quais sejam os códigos da nossa CPU) ao exposto na Tabela 3. Para que siga a "conversão", no final da tabela indicamos as correspondências que marca nosso "inexistente" manual. Entre outras coisas, os valores negativos e positivos dos saltos (instruções JUMP) se converteram em códigos hexadecimais (como é uma questão a que responderemos no capítulo seguinte).

O programa deverá gravar-se (veja o mapa de memória da fi-

gura 3) desde a direção $F000 em diante incluindo a iniciação. Na Tabela 3 supomos que esta ocupa um espaço de $89 bytes (137 em decimal), desde $F000 a $F088, pelo que a primeira instrução de nosso programa "de verdade" (CLX) se escreve na direção $F089; como ocupa um byte de memória a segunda instrução (LDA $E000) se escreverá na direção $F08A e ao ocupar três bytes, a terceira instrução se escreverá (dois bytes) em $F08D e em $F08E, a quarta em $F08F e $F090, e assim sucessivamente. Na mesma tabela, junto a cada instrução codificada, se indica a direção onde guardamos o primeiro byte da instrução correspondente. A longitude total do programa é de $D9 bytes (em decimal 217), desde $F000 a $F0D8.

Para executar o passo 6, se utiliza uma máquina especial denominada "Programador de PROM", e se programa uma PROM (ou uma EPROM ou EEPROM), com os códigos antes citados. Mas a programação não acaba aí, posto que nas duas últimas células desta memória (que supomos de 4.096 bytes) é preciso programar o famoso "Vetor de Reset". Seu valor deve ser $F000 por quanto que, como se viu no capítulo anterior, o vetor deve apontar ao começo do programa residente a fim de que ao realizar a conexão (liga) a CPU se dirija imediatamente a essa direção. Conseqüentemente, introduziremos o dado $F0 na penúltima célula da memória e $00 na seguinte. Uma vez instalada a memória no cartão estas posições corresponderão às direções $FFFE e $FFFF.

Dito e feito. Agora temos uma memória PROM de 4K bytes, que montamos exatamente no local correspondente de nosso cartão de microcomputador. Conectamos a tela e o teclado, os cabos de alimentação e acionamos o interruptor.Se tudo se realizou sem erros, a tela estará "limpa" e poderemos começar "a programar" em nossa mini-linguagem:

?"AB <return>

e na tela aparecerá:

AB

Pelo contrário, se teclamos:

EEEEEE <return>

obteremos, de forma implacável:

ERROR

e assim sucessivamente.

Verdadeiramente, temos de admitir que não se trata de um grande logro, mas o processo seguido para chegar a uma aplicação tão microscópica e limitada pode fazer-lhe compreender como é complexa a escrita de um verdadeiro programa intérprete como, por exemplo, um BASIC da Apple ou do Spectrum. Portan-

CÓDIGO HEXADECIMAL COD. OP.	OPERANDO		DIRECÃO DO PRIMERO BYTE DA INSTRUÇÃO
INICIALIZAÇÃO			DESDE $F000 A $F088
E9			$F089
AD	E0 00		$F08A
C9	FF		$F08D
F0	FB	(-5)	$F08F
9D	00 00		$F091
8D	E1 00		$F094
E8			$F097
C9	BF		$F098
D0	F0	(-16)	$F09A
E9			$F09C
8D	00 00		$F09D
C9	7F		$F0A0
D0	17	(+23)	$F0A2
E8			$F0A4
8D	00 00		$F0A5
C9	DF		$F0A8
D0	0F	(+15)	$F0AA
E8			$F0AC
8D	00 00		$F0AD
8D	E1 00		$F080
C9	BF		$F083
D0	F7	(-9)	$F085
E9			$F087
4C	D2	(-45)	$F088
A9	FE		$F08A
8D	E1 00		$F08C
A9	FD		$F08F
8D	E1 00		$F0C1
8D	E1 00		$F0C4
A9	F8		$F0C7
8D	E1 00		$F0C9
A9	FD		$F0CC
8D	E1 00		$F0CE
A9	BF		$F0D1
8D	E1 00		$F0D3
E9			$F0D6
4C	B3	(-77)	$F0D7

DIREÇÃO DO ÚLTIMO BYTE $F0D8

Tabela 3 - Codificação, em hexadecimal, do programa escrito na tabela 2. Se observa o caracter biunívoco entre a versão mneumônica e a codificação hexadecimal de cada instrução. À direita, se indica a direção de memória onde está situado o primeiro byte (o código operativo) de cada instrução. Para realizar a codificação, usamos um manual de programação imaginário de nossa "inventada" CPU, onde estariam indicadas as correspondências seguintes (mneumônico — código máquina em hexadecimal).

CLX	*E9*		*STA*	*8D*
LDA	*AD*	(DESDE A MEMÓRIA)	*INX*	*E8*
LDA	*A9*	(IMEDIATO)	*JNEQ*	*D0*
CMPA	*C9*		*LDAX*	*BD*
JEQ	*F0*		*JUMP*	*4C*
STAX	*9D*			

to, no fundo, qualquer BASIC não é outra coisa que uma seqüência de instruções em linguagem máquina, que obriga ao hardware a esperar as ordens procedentes do teclado e executá-las se, e somente se, não tem erros de sintaxe.

Em uma verdadeira linguagem "intérprete" existirão milhares de provas e comparações e centenas de pesquisas de palavras chave em lugar das duas ou três que nós implantamos, mas o resultado é sempre o mesmo: a máquina, com as instruções proporcionadas pelo software, "ganha vida" e se põe "a nossas ordens".

Naturalmente, teremos um "manual" bastante complexo e deveremos estudar a sintaxe própria da linguagem que quisermos utilizar, seja uma versão de BASIC, PASCAL, FORTH, FORTRAN ou "C". Levará bastante tempo a aprendizagem e não poucos segundos como para nosso microintérprete. Ao final, poderemos utilizar 100% a potência do harware, graças ao sofisticado software que, como se diz na gíria "corre" dentro do computador.

Conclusões: a importância do software

A árdua dissertação anterior se aplica a todo computador, desde o mais elementar dos microcomputadores até o mais complicado "monstro". Sempre tem um hardware, tão complexo como se queira, mas será o software quem o faz funcionar. Em separado das possibilidades funcionais do hardware, unicamente as boas qualidades do software fornecido com a máquina, ou que você desenvolve, permitirão obter prestações a alto nível (como aproveitar ao máximo a potência do hardware). Quanto mais potente é a CPU utilizada mais complexo deverá ser o software de controle, normalmente denominado "sistema operacional". Instalando um software residente adequado em um hardware este poderá realizar as funções que lhe forem encomendadas com o máximo proveito, sobretudo se lhe dedicar a uma aplicação concreta.

Este é o caso dos periféricos mais importantes do computador pessoal, começando pela impressora, e que precisamente por isto se denominam "inteligentes". Conseqüentemente, as impressoras, os dispositivos de traçado ("plotters"), as telas de texto e de gráficos, as unidades de disco e demais periféricos, são máquinas nas quais existe um hardware com a habitual estrutura (CPU, RAM, ROM e E/S) e um programa residente em ROM, que as faz funcionar na forma requerida. Na prática, os modernos periféricos são realmente computadores "especializados" preparados para conectar-se ao nosso computador pessoal. Por isto, quando fala-

mos deles não teremos necessidade de descrever seu hardware com detalhe porque, em essencia, já conhecemos. Conseqüentemente, haverá mais espaço para analisar as funções, conexões, manuseio, virtudes e defeitos. E isto é exatamente o que nos propomos fazer nos capítulos seguintes.

Glossário

USER-FRIENDLY: Este termo inglês se aplica ao software que controla o funcionamento de um computador tornando-o fácil para o usuário empregar (literalmente "amigável"). Um sistema operacional terá esta natureza quando todas suas ordens estiverem organizadas de tal modo que seja simples para quem o utiliza (introduzí-las no terminal, manejar suas operações e aprendê-las).

FIRMWARE / SISTEMA OPERACIONAL / SOFTWARE DO SISTEMA: Todos estes termos tem significado parecido. Em geral se utilizam para definir os programas que "personalizam" o hardware, conferindo à máquina um modo de funcionamento ajustado aos desejos do usuário. O primeiro termo se costuma empregar para definir todo o software armazenado em memória ROM ou similar (PROM, EPROM...).

O segundo se refere ao software que controla o conjunto das tarefas efetuadas pelo computador: atribuição de E/S, de memória, gestão da informação... e põe à disposição do programador uma série de ordens simples que lhe permitem acceder a zonas deste controle se o desejar.

Por último, o software do sistema engloba os dois termos anteriores estendendo-se também à linguagem de programação. Nas máquinas modernas somente uma pequena parte do software do sistema está em memória ROM e com ele basta para iniciar corretamente a máquina. Assim, depois de ligado, são quase sempre necessários periféricos de memória de massa conectados ao computador (por exemplo, discos) dos quais se extraem, de vez em quando, os blocos de software necessários para realizar as diversas operações. Desse modo, se encomiza muito espaço na memória central do computador.

LINGUAGEM: Programa executado pela CPU do computador que permite ao usuário escrever outros programas ou introduzir dados de uma maneira muito mais simples e compreensível que com o emprego da linguagem de máquina. Um dos mais

populares é o BASIC, que tem a particularidade de que quando se "arranca" — se executa — é como se o usuário possuísse uma nova máquina, pois é capaz de compreender instruções, ordens e programas escritos em linguagem BASIC. Existem muitas linguagens para cada um dos computadores presentes no mercado, além do citado BASIC. Entre eles estão os conhecidos, Pascal, FORTH, FORTRAN, "C", Cobol, Logo, etc. Cada um está recomendado para um tipo de aplicação, destacando-se em aspectos distintos.

LOOP: Significa "anel" ou "ciclo". É uma parte integrante do programa, que se executa de forma repetida até que se cumpra uma determinada condição. Um "loop" em essencia está constituído por:

1) execução de instruções, 2) provas para verificar se uma determinada condição "imposta antes de entrar no loop" é válida e 3) se for válida, se voltará ao item 1; se não for, terminará o "loop" e a execução prosseguirá com a instrução imediatamente posterior ao do item 3.

O emprego em um programa de um "loop", com "ponteiros" ou "contadores" que indicam o número de vezes que se executa o "loop", simplifica notavelmente a execução de muitas operações.

SINTAXE: Conjunto de normas que regem um idioma e também, por extensão, uma linguagem para computador. Ao programar, além dos erros de sintaxe, existem, naturalmente, os de ortografia. Em uma linguagem, ambos os erros impedem ao programa intérprete compreender que "diabo" de instruções lhe está transmitindo o usuário, pelo que o sistema operacional se refugiará atrás de uma mensagem de erros — "SYNTAX ERROR" — e corresponderá ao programador a tarefa de encontrar o engano no programa e corrigí-lo.

BUG (ERROS DE PROGRAMAÇÃO): Literalmente, este termo significa "percevejo" e se refere aos erros que "infestam" os programas, inclusive dos mais experts programadores. Se elimina com uma operação adequada de "desinfecção" que toma o nome de DEBUG (DEPURAÇÃO). Esta depuração, sobretudo nos sistemas evoluídos para o desenvolvimento de software, se realiza por meio de um programa especial, que curiosamente, costuma denominar-se "DDT"; ainda que o pareça não se refere ao conhecido inseticida, mas sim à abreviação de "Dynamic Debugging Tool" (literalmente ferramenta de depuração dinâmica) que é um rápido e potente "busca erros" utilizando, sobretudo, com programas em lingua-

gem de máquina.

MNEUMÔNICO/ ASSEMBLER : Cada instrução do conjunto executável pela CPU tem seu exclusivo código binário que é o único que a faz compreensível por parte do decodificador de instruções interno à CPU. No entanto, seria "pouco humano" obrigar ao programador em linguagem máquina a recordar todos os códigos binários (além de estúpido e inútil), pelo que o mesmo fabricante da CPU fornece uma lista das instruções executáveis, atribuindo a cada um dos códigos binários um nome fácil de recordar e que costuma ter relação com o que faz a própria instrução. Por exemplo: LDA por LoaD Accumulator (carregar acumulador), CLX por CLear register X (apagar registro X), STA por STore Accumulator (armazenar acumulador), STAX por STore Accumulator indexed X (armazenar acumulador indicado X), etc.

O programador escreve seu programa empregando estes códigos mneumônicos, de forma que as instruções coincidem, uma a uma, com as executáveis pela CPU. A linguagem utilizada se demonina linguagem de Assembler . Uma vez escrito o programa em linguagem de Assembler ("programa fonte"), se deve consultar o manual da CPU e traduzir cada código mneumônico ao código binário correspondente. Afortunadamente, esta operação se realiza de forma automática em muitas máquinas, com programas de tradução denominados "compiladores", que realizam justamente esta tradução, de forma que no final se tem uma seqüência de códigos binários que representam o programa em formato executável pela CPU ("programa objeto"). Para que se possa finalmente executar deve introduzir-se tal programa em memória: seja RAM ou ROM (EPROM, ...) e posicionar o contador de programa na direção de começo.

CAPÍTULO V

SISTEMAS DE NUMERAÇÃO E CODIFICAÇÃO E ALGUNS TRUQUES ARITMÉTICOS

Nos capítulos anteriores manejamos com despreparo números binários e hexadecimais, mas existem muitas coisas que precisamos saber sobre eles: como se representa um número negativo, como se opera com eles, de que forma passar de um a outro... Também devemos tratar os sistemas de codificação, dos que não falamos nada até agora e que, sem dúvida, tem bastante interesse.

Números binários

Vamos começar pelo princípio: em um computador todos os sinais são elétricos, isto é, do tipo tem ou não tensão. E onde? Naturalmente, em um fio que une dois dispositivos, dos quais um envia e o outro recebe o sinal. Chamamos "1" ao estado de "tensão no fio" e "0" ao estado contrário, conforme um convênio universalmente aceito. Como vimos, em muitos casos se reunem fios que tem a mesma função, até formar um feixe ao qual estão conectadas a CPU e os diversos dispositivos, que desta forma podem dialogar através desta via comum que se denomina bus.

Assim, podemos definir simultaneamente o estado de sinal em todos os fios ou linhas do bus definindo um dado, expresso

com "zeros" e "uns", formado por tantos dígitos quantos fios ou linhas tenha no bus. Para distingüir entre si estas linhas, se atribui a cada uma delas um nome e uma prioridade. À primeira linha corresponderá o chamado "bit número 0", à segunda o "bit número 1" e assim sucessivamente. A atribuição de prioridade se resolve conferindo maior "peso" (importância) aos bits cuja denominação seja mais "alta". O bit 0 será o de menor peso, ou o que significa o mesmo, o bit "menos significativo" (LSB = Least Significant Bit), enquanto que o bit 7 (sobre 8) ou o bit 15 (sobre 16) são os bits "mais significativos" (MSB = Most Significant Bit).

Disto se deduz que para um bus de 8 linhas (por exemplo o bus de dados), o número resultante pode ser uma combinação qualquer de "uns" e de "zeros" de 0000.0000 até 1111.1111, enquanto que para um bus de 16 linhas (por exemplo o bus de direções) as combinações são todas as compreendidas entre 0000.0000.0000.0000 e 1111.1111.1111.1111 (acrescentamos pontos subdividindo os números em grupos de quatro dígitos, simplesmente por comodidade de leitura). No primeiro caso teremos somente 256 combinações possíveis, enquanto que no segundo caso as combinações serão 65.536. Todos estes números se denominam "binários", já que cada dígito só pode ser 0 ou 1. O dígito simples toma o nome de "bit" (unidade elementar de informação), contração da denominação inglesa "Binary digit" (dígito binário). Agora, finalmente, estamos preparados para traduzir os números binários a uma numeração que todos devemos conhecer: a numeração decimal, isto é, realizar conversões entre números expressos em base 2 ou 10.

Para fazê-lo, basta construir uma tabela de correspondência como a seguinte:

número binário	número decimal corresponde
0000.0000	0
0000.0001	1
0000.0010	2
0000.0011	3
...	...
1111.1111	255
...	...

e assim sucessivamente. Se poderia continuar até o "infinito" aumentando o número de bits, mas um procedimento deste tipo é inútil. O mais prático (a não ser que você "invente" outra coisa) é efetuar em cada ocasião, o cálculo seguinte:

Valor decimal = bit 0×2^0 + bit 1×2^1 + bit 2×2^2 + ... + bit $n \times 2^n$.

onde "2^n" significa "2 elevado a enésima potência"; o bit 0 é o valor do bit menos significativo (LSB) e o bit n é o bit mais significativo (MSB). Por exemplo, o número binário 0101.0000 correspondente ao valor decimal 80, já que se tem:

$$0.2^0 + 0.2^1 + 0.2^2 + 0.2^3 + 1.2^4 + 0.2^5 + 1.2^6 + 0.2^7 = 16 + 64 = 80$$

De agora em diante, e para não haver confusão usaremos a seguinte anotação para determinar em que base está o número que escrevemos:

	Decimal	Hexadecimal	Binário
Expr. equivalentes	10	$0A	%00001010
	10d	0Ah	00001010b

Sistema de numeração hexadecimal

A numeração binária, tão cômoda para o computador, é notavelmente incômoda para os usuários. Esta é a razão do uso dos números hexadecimais.

A numeração hexadecimal não é mais que uma "fritura" da numeração binária, mediante a qual podemos tratar números equivalentes aos binários sem ter que manusear grandes quantidades de "zeros" e de "uns".

O truque (que, em definitivo, se baseia no fato de que 16 — base da numeração hexadecimal — é a quarta potência de 2 — base da binária —) é o seguinte: se agrupam os "zeros" e os "uns" do número binário de quatro em quatro bits partindo, sempre do bit menos significativo. Tendo em conta de que com quatro bits se obtem até 16 combinações, podemos utilizar 16 símbolos para representar sem confusão todas essas combinações. Deste modo, teremos "comprimido" 4 dígitos binários (bits) em um só dígito hexadecimal. Os 16 símbolos eleitos foram as 10 cifras decimais (0,1,..,9) mais as 6 primeiras letras do alfabeto A, B, C, D, E, F. Deste modo cada grupo de 4 bits (denominado também "nibble") se corresponde com um dígito hexadecimal, como se pode ver na seguinte tabela:

DECIMAL	HEXADECIMAL	BINÁRIO
0	0	0000
1	1	0001

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

Com a notação hexadecimal uma direção como ..... 1111.1111.1111.1110b adota a forma mais humana e compreensível de FFFEh (ou então $FFFE).

Com 8 bits os números hexadecimais correspondentes terão dois dígitos, de 00h a FFh; com 16 bits terão quatro dígitos, de 0000h a FFFh, etc. Mediante o emprego dos números hexadecimais torna-se também mais fácil encontrar o valor decimal correspondente, já que cada cifra hexadecimal representa a quatro bits (os pesos são potências de 16 e não de 2) o valor decimal que se calculará aplicando a fórmula seguinte:

valor decimal = dígito 0×16^0 + dígito 1×16^1 + ... + dígito $n \times 16^n$.

Se o número é, por exemplo, 50h, o valor é $0 \times 1 + 5 \times 16 = 80$ (recordemos que a potência "zero" de qualquer número é 1; $16^0 = 1$). É conveniente habituar-se, desde o princípio, a manusear os números em suas distintas representações (decimal), binária, hexadecimal, ou como veremos mais adiante, em BCD), pois o bom programador deverá usá-las a miúdo.

Talvez queira saber como se lêem os números! Portanto se é assim como se modesta e atinadamente, acredita que não, vejamos:

- Um número decimal se lê como estamos acostumados: <um>, <dez>, <cem>..., etc.
- Um número binário se lê dígito a dígito. Assim, 1001.0111b, por exemplo, se lê "um.zero.zero.um.zero.um.um.um", e 101b "um.zero.um" e não "cento e um".
- Um número hexadecimal também se lê dígito a dígito, inclusi-

ve se parecer a um número decimal. Conseqüentemente, AF3Ch se lê "a.efe.três.ce" e, por exemplo, "1000" se lê "um.zero.zero.zero" e não "mil".

Um pouco de aritmética binária

Também os números binários ou hexadecimais, como todos os números, se podem somar, diminuir, multiplicar, dividir. As regras são idênticas às que se aplicam à aritmética decimal, com a única diferença dos "pesos", que são potências de 10 no sistema-decimal, de 2 no binário e de 16 no hexadecimal. Levando em conta os "transportes" ("carry"), isto é, o excesso sobre o maior dígito da base, as contas sempre serão exatas. Por exemplo: 12d + 19d = 31d tem um transporte desde as unidades às dezena. pelo contrário, no binário se teria para a mesma operação 0000.1100b = 0001.0011b que não produz nenhum transporte; também em hexadecimal se teria 0Ch + 13h = 1Fh, que não gera transportes de um peso a outro.

Em binário se produz um transporte quando se somam dois dígitos iguais a um, dado que 1b + 1b = 10b. Assim, por exemplo, 7d + 12d = 19d (sem transporte), mas seu equivalente em binário 0000.0111b + 0000.1100b = 0001.0011b e também 7 + Ch = 13h, tem um transporte desde o grupo de 4 bytes ("nibble") menos significativo ao imediato superior.

Para praticar pode utilizar a roda da figura 1. Cada vez que se passa pelo zero se tem um transporte. De forma análoga se podem calcular as subtrações, tendo em conta que, neste caso o transporte é negativo (em inglês "borrow", que vale -1).

Números negativos

Vimos que todos os números decimais podem representar-se, de forma biunívoca, em binário (ou em hexadecimal). Para isto bastará dispor do adequado número de bits. Com 8 bits se podem representar todos os números decimais de 0d a 255d, enquanto que com 16 bits se tem todos os valores compreendidos entre 0d e 65535d e como se observará todos os valores mencionados são positivos. No entanto, a miúdo temos que trabalhar com valores negativos. Em tal caso, o artifício mais difundido consiste em utilizar o bit de maior peso (o bit 7 em 8 bits e o 15 em 16) — recordemos que o menos significativo é o bit 0 — para indicar o sinal, de modo que se é zero o número é positivo e se é 1 o número é

negativo. Utilizando esta convenção, com 8 bits podemos representar os números compreendidos entre -128d e +127d, e com16 bits se podem representar os números compreendidos entre

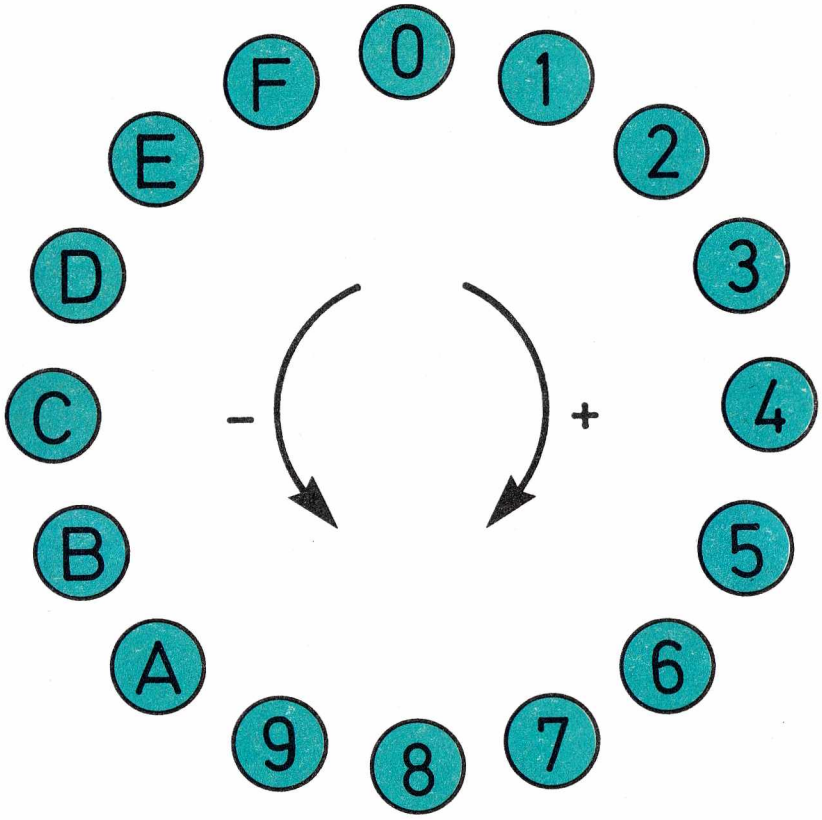


Fig. 1 - Roda para o cálculo de "transportes" nas somas ou subtrações de dígitos hexadecimais. Para uma soma, se parte do dígito do primeiro têrmo e se gira em sentido horário tantas posições quantas indique o dígito do segundo têrmo, com um transporte de 1 cada vez que se passa pelo zero: por exemplo, 1Ch + 9h = 5h com um transporte de 1 a somar ao bit de maior peso, quer dizer, no total 25h. Para uma subtração se gira em sentido contrário, com um transporte negativo de 1 cada vez que se passa por zero: 25h—9h se calcula tomando 5h, retrocedendo em 9 posições até Ch e contando -1 de transporte negativo a tirar do dígito de peso superior. O resultado definitivo é 1Ch.

-32768d e +32767d.

Por comodidade, o tratamento dos números inteiros negativos se realiza representando-os mediante uma técnica especial denominada "em complemento a 2". À primeira vista, se poderá pensar que, se 000.0001b é o valor 1d e o valor -1d estaria representado por 1000.000 1b. Na verdade, se utiliza um método menos evidente, porém mais produtivo, que consiste em que o número negativo se obtém, tomando-se o número positivo correspondente, substituindo os "zeros" por "uns" e vice-versa, e somando-lhe, em seguida, ao resultado, o valor 1 (levando em consideração os possíveis transportes aos pesos superiores). Consequentemente, "-1d" se faz 0000.0001b (+ 1d) —►1111.1110b (complemento a 1) —► 111.111b (-1d). Este sistema não é muito complexo e, em compensação, facilita muito a execução de qualquer operação na CPU.

Para simplificar o circuito da unidade aritmético-lógica (ALU), dentro da unidade central de processo (CPU) não existe uma lógica de subtração, mas somente um bloco somador.

Quando se tem que somar dois números binários não tem problema algum, mas quando se tem que efetuar uma diminuição, o subtraendo se põe antes, automaticamente, em complemento a dois de forma que a diminuição se transforma em uma simples soma. A modo de prova vejamos um exemplo: 1d - 1d = 0d (sobre este resultado não há dúvida alguma) se converte em 0000 .0001b (1d) + 1111.1111b (1-d) = 0000 .0000 b como se quer demonstrar. O transporte desde o bit mais significativo não se tem em conta posto que o segundo termo da soma é um número negativo e a CPU "o sabe").

Se quer convencer-se, pode consultar qualquer livro da bibliografia. Encontrará, na descrição dos circuitos digitais todos os somadores e "negadores" que quiser, mas nenhum "diminuidor". Em resumo: qualquer soma em binário (ou em hexadecimal) se calcula de modo análogo a como se efetua com os números decimais, levando em consideração os transportes de um peso a outro; pelo contrário, toda diminuição se efetua substituindo o subtraendo pelo seu complemento a 2, e somando-o logo ao minuendo em lugar de diminuí-lo. Esta operação é realizada de forma automática pela ALU da CPU, ou, deveremos prepará-la (veja Tabela 1), dependendo do μP que usemos.

Conversão de um número decimal a seu equivalente binário (ou hexadecimal)

Nos itens anteriores se constatou como qualquer número bi-

nário, ou hexadecimal, é facilmente conversível em seu valor decimal. Um pouco mais complexa resulta a operação inversa, embora seja sempre possível. Basta tomar o valor decimal que se quer converter em binário, dividí-lo por dois e apontar o resultado divisão, que será naturalmente 1 ou 0. O quociente se divide novamente por dois, marcando também o resultado e assim se continua até que o quociente seja igual a 0. O último resultado, será o bit mais significativo e os resultados anteriores são, por ordem, os bits de pesos decrescentes.

Na figura 2 se ilustra este procedimento com dois exemplos simples que serão lidos como segue: a) "251d dividido por 2 é 125d com resultado 1; 125d dividido por 2 é 62d com resultado 1; 62d dividido por 2 é 31d com resultdo 0; 31d dividido por 2 é 15d com resultado 1; 15d dividido por 2 é 7d com resultado 1; 7d dividido por 2 é 3d com resultado 1; 3d divido por 2 é 1d com resultado 1; 1d entre 2 é 0 com resultado — último resultado — 1". O número binário resultante é 1111.1011b = FBh = 251d, como se pode comprovar com facilidade.

Um exemplo análogo é o da figura 2b, onde o número a converter é par. Para converter um número negativo se transforma previamente sem levar em consideração o sinal e logo se realiza seu complemento a dois, tal como se viu no item anterior.

Números negativos ← 2.° dígito hexad.

1.° dígito hex.	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
F	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
E	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
D	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	44
C	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
B	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
A	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
9	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
8	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128

Tabela 1 - Tabela para o cálculo dos números negativos de 8 bits (dois dígitos hexadecimais). Por comodidade, não colocamos o sinal "-". Para obter, por exemplo, -20 se procura 20 (para simplificar se evita o sinal menos) e se obtém ECh. Se pode verificar o resultado no modo habitual: 20d = 0001.0100b; o complemento a 1 é 1110.1011b e o complemento a 2 (se soma 1) é 1110.1100b = ECh, como se esperava.

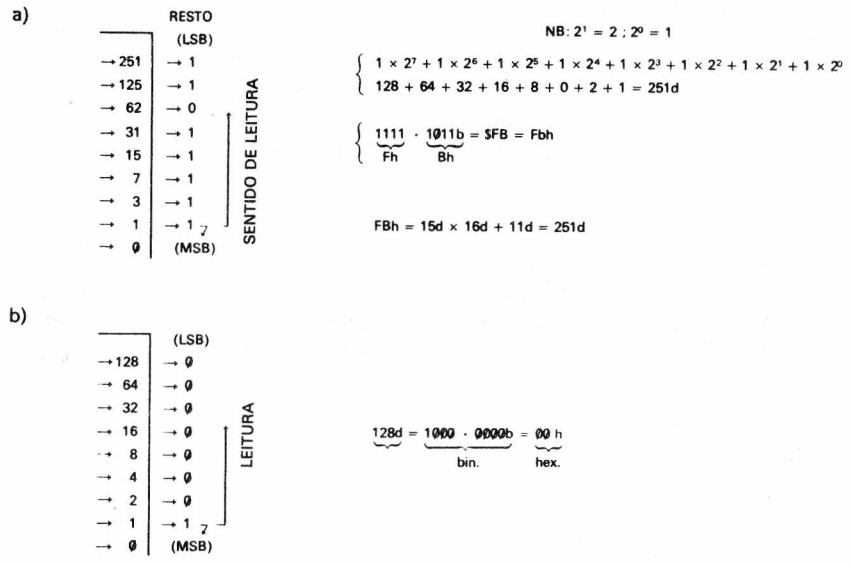


Fig. 2 - Exemplo de conversão de decimal a binário a) de um número ímpar, b) de um número par.

Código BCD: decimal codificado em binário

Há ocasiões nas quais em lugar de converter, por exemplo, o número "9781d", aplicando o procedimento antes indicado, nos convém converter diretamente cada dígito decimal em seu equivalente binário (já que com 4 bits podemos representar todos os dígitos decimais do 0 ao 9). Um caso típico desta necessidade é a apresentação em um display de 7 segmentos (7 leds colocados como um 8:8) de um número decimal qualquer.

Para nosso 9781d obteremos o número binário ..... 1001.0111.10000.0001 que, evidentemente, não é o número binário correspondente. De fato, ao coverter este número a decimal aplicando as fórmulas que vimos anteriormente se teria 39025d, bem diferente de 9871d. Tudo isto acontece porque convertemos o número decimal de partida utilizando a notação "decimal condificado em binário" (Binary Coded Decimal = BCD). Se trata de uma notação muito cômoda para o usuário que, mesmo estando na era do computador, segue utilizando dez dedos e, conseqüentemente, o sistema decimal.

Realmente os computadores, para muitas aplicações de entrada/saída dirigidas a nós e inclusive no cálculo científico ou comercial, se adaptam a nossas exigências e adotam o sistema BCD (para as quantidades muito grandes ou muito pequenas se utilizará a denominada representação em "ponto flutuante", mas este tema será tratado ao vermos o BASIC). É evidente que com este sistema esbanjaremos bits relacionados ao sistema binário puro (uns 20% como mínimo). Por exemplo, 999d requer três dígitos BCD (1001.1001.1001), isto é 12 bits, frente aos 10 bits necessários no sistema binário puro ou hexadecimal.

Ao efetuar operações com dígitos em BCD, temos de ter em mente que já não se produzem transportes ao passar pelo 0 depois do valor F(1111b), se não depois do valor 9d(1001 BCD). Uma boa ALU permite ao menos efetuar somas em BCD.

A codificação ASCII

No computador as configurações de "zeros" e "uns" deixam poucas oportunidades à fantasia do programador. Não somente os bytes tem um significado bem preciso (seja em binário/hexadecimal ou em BCD), como também, quando queremos representar caracteres alfanuméricos, estaremos ligados a codificações standar universalmente utilizadas. Uma delas, a mais conhecida, é a condificação ASCII, abreviatura de "American Standar Code for Information Interchange" e que utiliza 7 bits para 128 códigos diferentes (o oitavo bit é sempre 0 para os caracteres standar, enquanto que vale 1 quando se refere a caracteres semigráficos, particulares de algumas máquinas). Cada um destes 128 códigos corresponde a uma letra do alfabeto, a um número de 0 a 9, a um sinal de pontuação, a um código de controle, etc., tal como se mostra na Tabela 2, onde para cada caracter se indica sua codificação ASCII expressa em decimal e hexadecimal.

Quando enviarmos, por exemplo, o código ASCII 41h, estaremos representando a letra "A". Onde utilizamos os códigos ASCII para codificar letras, números e símbolos diversos? Quando um "port" de entrada "lê" o teclado de nosso computador pessoal, por exemplo, ao pressionar a tecla "A" um circuito particular acoplado ao teclado (conhecido como "codificador"), situa o código ASCII 0100.0001b, isto é, 41h, nas 7 linhas que estão conectadas ao citado "port" de entrada. A partir desse momento, dentro do computador, o dado 41h indicará a letra "A", e todos os dispositivos capazes de compreender a codificação ASCII o considerarão au-

tomaticamente como tal. Por exemplo, um controlador de vídeo ao que a CPU envia 41h visualizará na tela "A", uma impressora imprimirá "A", etc.

Com 7 bits, e conseqüentemente com um conjunto de 128 caracteres diferentes, se podem "traduzir" todos os símbolos alfanuméricos junto com muitos outros caracteres especiais ..... <!,.?."# $&'()★:=-> e diversos sinais de controle.

Trata-se de códigos importantíssimos que servem para o intercâmbio de informação inclusive sob a forma de códigos de controle que, como se vê na Tabela 2, correspondem aos que vão de 1d a 31d e ao 127d, incluindo sinais de controle tais como BEL (Beep), CR (Carriage Return — retorno de carro), BS ("Back Space", retrocede um espaço), DEL (apagar) e outros controles mais "comunicativos" tais como ACK (Acknow-ledgement", reconhecimento), NACK (Non-Acknow-ledgement — não reconhecimento), etc.

Os primeiros servem para transferir ao periférico, que costuma ser uma impressora (mas que também pode ser um terminal de vídeo ou dispositivos similares), a ordem de fazer tocar algum sinal acústico (se houver), do retorno do carro de impressão, ou também gerar um salto de linha (LF, Line Feed) ou um salto de página (FF, Form Feed). Pelo contrário, os códigos de controle tais como ACK e NACK servem para possibilitar o intercâmbio de informação entre os periféricos e o computador, com sinais do tipo "recebido" ou "não recebido", a clássica de "curto e fechado", etc.

Os caracteres de controle são gerados no teclado de um computador pessoal, pressionando-se simultaneamente uma tecla denominada de controle (abreviada como CTRL com frequência) e uma letra. As letras utilizadas vão desde o A em diante. Por exemplo, Control (abreviamos com o sinal^) <A> proporciona o código ASCII 001d, que significa "Stard of Header" Começo de cabeceira (SOH),^F (006d) da ACK,^J(010d) proporciona LF, M(013d) da CR, etc. Se, em BASIC, se tecla:

```
PRINT CHR$ (7)
```

que significa "escreva na tela o valor do código ASCII 7", na realidade não se escreve nada, pois neste caso o computador apenas emite um som de curta duração ("beep"), que corresponde ao caracter BEL, conforme indica a Tabela 2.

O código 027d (Escape) indica que a seqüência de caracteres que vai depois do mesmo deve considerar-se de um modo par-

CODIGOS ASCII NORMALIZADOS

Dec.	Hex.	CAR.	Dec.	Hex.	CAR.	Dec.	Hex.	CAR.
000	00	NUL	043	2B	+	086	56	V
001	01	SOH	044	2C	,	087	57	W
002	02	STX	045	2D	-	088	58	X
003	03	ETX	046	2E	.	089	59	Y
004	04	EOT	047	2F	/	090	5A	Z
005	05	ENQ	048	30	0	081	5B	[
006	06	ACK	049	31	1	092	5C	
007	07	BEL	050	32	2	093	5D	
008	08	BS	051	33	3	094	5E	^
009	09	HT	052	34	4	095	5F	—
010	0A	LF	053	35	5	096	60	'
011	0B	VT	054	36	6	097	61	a
012	0C	FF	055	37	7	098	62	b
013	OD	CR	056	38	8	099	63	c
014	0E	SO	057	39	9	100	64	d
015	0F	SI	058	3A	:	101	65	e
016	10	DLE	059	3B	;	102	66	f
017	11	DC1	060	3C	<	103	67	g
018	12	DC2	061	3D	=	104	68	h
019	13	DC3	062	3E	>	105	69	i
020	14	DC4	063	3F	?	106	6A	j
021	15	NAK	064	40		107	6B	k
022	16	SYN	065	41	A	108	6C	l
023	17	ETB	066	42	B	109	6D	m
024	18	CAN	067	43	C	110	6E	n
025	19	EM	068	44	D	111	6F	o
026	1A	SUB	069	45	E	112	70	p
027	1B	ESCAPE	070	46	F	113	71	q
028	1C	FS	071	47	G	114	72	r
029	1D	GS	072	48	H	115	73	s
030	1E	RS	073	49	I	116	74	t
031	1F	US	074	4A	J	117	75	u
032	20	SPACE	075	4B	K	118	76	v
033	21	!	076	4C	L	119	77	w
034	22	"	077	4D	M	120	78	x
035	23	#	078	4E	N	121	79	y
036	24	$	079	4F	O	122	7A	z
037	25	%	080	50	P	123	7B	{
038	26	&	081	51	Q	124	7C	
039	27	'	082	52	R	125	7D	}
040	28	(	083	53	S	126	7E	~
041	29	)	084	54	T	127	7F	DEL
042	2A	*	085	55	U			

Dec = decimal, Hex = hexadecimal, CAR = caracter LF = Line Feed (avance linha), FF = Form Feed (avance uma folha papel), CR = Carriage Return (retorno de carro), DEL = apagar.

Tabela 2 - Tabela dos 128 códigos ASCII normalizados (bit 7 = 0), incluindo os códigos de controle.

ticular (se afastado do standar) e permite aos dois interlocutores entenderem-se conforme conveções particulares.

Não há nenhuma razão para usar outra codificação que não seja a ASCII, pois todos os dispositivos, e sobretudo os periféricos de uso mais freqüente, estão desenhados de modo que admitam seus códigos. Compreenderão perfeitamente os dados que se lhes transmitam sempre que estejam expressos, precisamente, nos correspondentes códigos ASCII.

No nosso exemplo do capítulo anterior, pelo contrário, utilizamos uma codificacão "espartana" e imediata para as teclas de entrada, o qual não importa muito porque supusemos que se disporia de todo um hardware "feito sob medida" e o exemplo era meramente didático. Na realidade, se houvéssemos usado um verdadeiro teclado comercial, e um monitor de vídeo ordinário, as teclas E, R, O, A, B," e?, se haveriam codificado, em ASCII, como 45h, 52h, 4Fh, 41h, 42h, 22h e 3Fh, correspondentes aos valores decimais 69d, 82d, 79d, 65d, 66d, 34d e 63d.

Qualquer um que tenha um computador pessoal com uma versão de linguagem BASIC, se não confiar, pode comprovar ditas codificações internas provando a instrução PRINT ASC ("x"), colocando em lugar de <x>, sucessivamente (o que constitui um exercício banal, mas instrutivo) todos os caracteres do teclado.

E já é hora de falar em profundidade de um dos periféricos "por excelência" de um sistema informático: a impressora.

CAPÍTULO VI

A IMPRESSORA

Finalmente podemos falar dos periféricos de um computador, isto é, de todas aquelas máquinas e/ou acessórios que se conectam à estrutura base de nossa unidade de processamento. O mercado oferece inúmeros tipos, que desenvolvem as funções mais variadas e especializadas, mas nos limitaremos a dar-lhes uma descrição resumida da maioria concentrando nossa atenção nos mais importantes. Neste capítulo falaremos das impressoras que, depois da tela e do teclado (que constituem um conjunto de E/S considerado normalmente parte integrante do computador pessoal) é o periférico "número um" do nosso sistema. Sua importante função é registrar de maneira permanente sobre papel o fruto dos processos de nosso computador, tais como a listagem de um programa, seus resultados numéricos ou gráficos, ou um conjunto de dados extraído de qualquer dispositivo de armazenamento (seja ou não de memória massiva).

A importância de ter uma impressão dos dados é evidente: em papel se pode raciocinar melhor, se pode ter uma visão de conjunto dos dados, como quando se lê um livro, e se podem efetuar também mais facilmente correções tendo sempre à vista todas as variações possíveis com relação ao original tirado pela impressora. Todas estas variações são impossíveis, ou pelo menos bastante difíceis de efetuar, com a ajuda de uma tela, que somente permite a visão de uma parte limitada do programa ou de uma página de

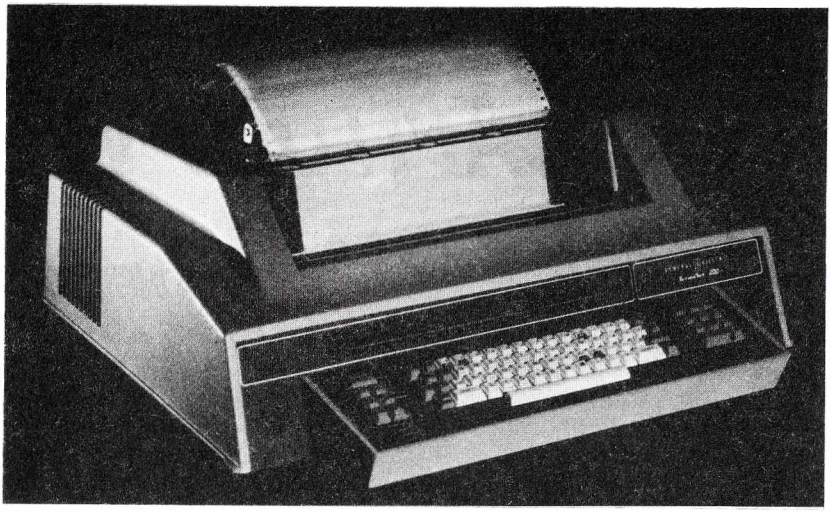


Foto 1 - Exemplo de um teletipo.

cada vez, para não falar do incômodo que acarreta ter que voltar a ligar a máquina e reinserir os disquetes, depois de desligá-la, se quisermos ver novamente os dados. Em relação à tela, as vantagens são notáveis enquanto que os inconvenientes são exclusivamente a relativa lentidão de impressão e o consumo do papel.

De qualquer forma, precisamente por isso, a impressora não pode substituir por completo a tela, pois a esta última corresponde desempenhar as funções simples e rápidas de edição dos dados introduzidos pelo teclado, pelo que são válidas as regras de ouro seguintes: 1) utilizar a tela para trabalhar durante a introdução dos dados; 2) empregar a impressora uma vez terminada a introdução para detectar com tranqüilidade, sentados à mesa, os erros que cometemos ou então, ter uma listagem por escrito.

No entanto, insistimos em que, por regra geral, qualquer computador pessoal que se preze terá um teclado e uma tela, bem integrados na estrutura de base ou externos (as funções e características gerais das telas serão examinadas mais adiante).

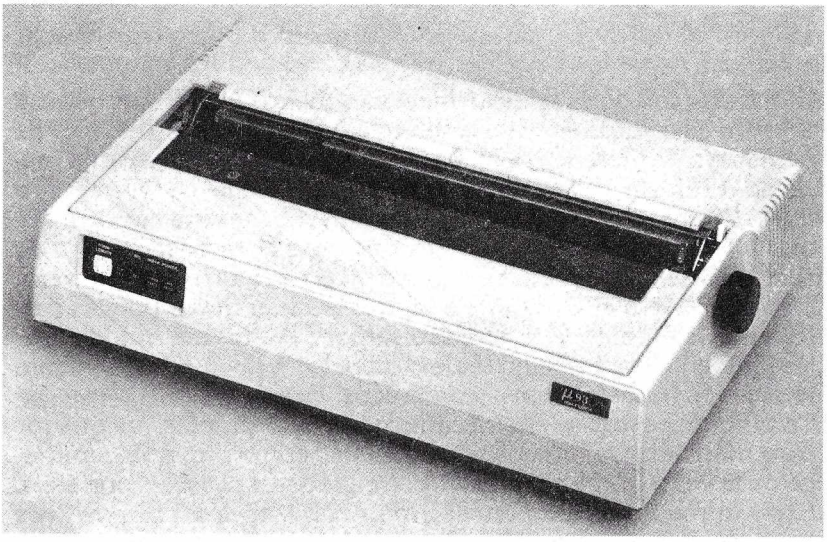


Foto 2 - Impressora de agulhas.

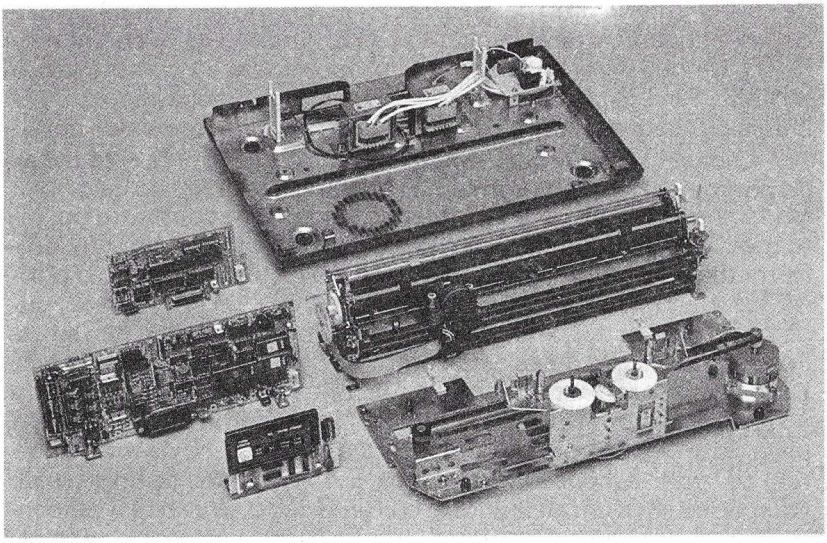


Foto 3 - Peças de uma impressora de agulhas.

Nesta categoria se incluem todas aquelas impressoras que imprimem os caracteres em papel sem ajuda de meios mecânicos de percussão, à diferença do que sucede, por exemplo, em uma máquina de escrever normal, na qual é uma espécie de "martelo" o que marca o símbolo sobre o papel, ao golpear uma fita com tinta. As tecnologias de "não percussão" são numerosas e uma de suas melhores qualidades é o baixo nível de ruído que produzem, o que se deve a que durante a impressão tem muito poucas partes em movimento. Por isto, são sempre aceitas em todos aqueles ambientes de trabalho onde o silêncio "vale ouro". Começaremos pelos modelos mais econômicos: as impressoras térmicas. Costumam basear-se em mecanismos muito simples, adequados para máquinas de dimensões reduzidas, como prova seu emprego cada vez mais freqüente nos computadores pessoais portáteis. Estas impressoras também são muito utilizadas em dispositivos baseados em microprocessadores e dedicados a funções específicas, tais como caixas registradoras, balanças e registradoras de dados ("data-logger"). Um denominador comum das impressoras térmicas é o reduzido comprimento da linha de impressão, que pode conter um máximo de 80 caracteres nos modelos mais sofisticados, enquanto que a velocidade da impressão varia de um modelo para outro e é inversamente proporcional ao comprimento da linha. Pelo geral, são velocidades bastante freqüentes de 2 a 5 linhas por segundo para os modelos de 20 caracteres por linha.

O papel sobre o qual se imprimem os caracteres está tratado por meios químicos com uma substância sensível ao calor (papel termosensível). Este se arrasta à velocidade constante ou, dito mais adequadamente, com pequenos saltos para adiante, uniformemente separados. Enquanto se desloca, um cilindro o oprime contra um suporte, denominado "cabeça de impressão", onde há uma série de diminutos elementos que podem ou não se esquentar. Cada um deles está controlado por uma das linhas procedentes de um "port" de saída que, por sua vez, está conectado ao resto do hardware do computador. Neste último deve existir um programa de controle que envia todos os impulsos de forma correta ao "port" e, conseqüentemente, aos elementos de impressão. Quando se ativa um elemento, a corrente que o atravessa o esquenta, já que na prática dito elemento não é outra coisa que uma resistência muito pequena. O calor gerado, ainda que o tempo de ativação seja bastante breve, é suficiente para fazer obscurecer o

papel no ponto que, nesse preciso instante, está em contato com o elemento de impressão, e assim no papel ficará marcado um minúsculo ponto. Com o deslocamento da cabeça perpendicularmente ao sentido de arrasto do papel, consegue-se "incidir" em uma série ordenada de pontos e, conseqüentemente, criar uma série de caracteres cuja definição depende exclusivamente das dimensões do ponto gerado e da proximidade dos elementos de caldeu entre si.

Dentro de certos limites, quantos mais forem os elementos de caldeu da cabeça tanto mais alta pode ser a velocidade de impressão, já que nada impede ativar simultaneamente mais elementos para formar no papel mais caracteres em uma só passada.

Um segundo tipo de impressora de "não percussão" é a que utiliza um papel que não é sensível ao calor, mas às descargas de eletrecidade estática. Em tal caso, se fala de tecnologia de impressão eletrostática. O papel tem uma superfície prateada e se enegrece onde se produz a descarga elétrica, isto é, entre o elemento da cabeça e o próprio papel. A mecânica e a técnica de impressão (movimento da cabeça e arrasto do papel) são as mesmas que os da impressora térmica. Na figura 1 se ilustra, de forma simplificada, a mecânica destes dois tipos de impressora.

Um terceiro tipo de impressora tem uma cabeça móvel, que se desloca perpendicularmente ao sentido de arrasto do papel que, na prática é um funil muito delgado pelo qual, a partir de um depósito especial, se fazem passar microgotas de tinta que logo "mancham", naturalmente de forma controlada, o papel situado na frente. Uma impressora que utiliza esta tecnologia de impressão toma o nome de "Ink-jet" (isto é, jorro de tinta).

É evidente que, independente do método empregado, o objetivo é sempre o mesmo: gerar os caracteres no papel compondo-os por meio de pontos.

Em qualquer caso, o arrasto do papel e da cabeça em sentido perpendicular, são controlados sempre por um microprocessador (naturalmente, com sua RAM, ROM e E/S). Finalmente, um dos mais recentes métodos de impressão é o "disparo de tinta", tecnologia utilizada em algumas máquinas produzidas pela Olivetti. Este método (na terminologia inglesa denominado "Ink-shot") combina a modalidade de impressão eletrostática com a de jorro de tinta, eliminando os problemas comuns a ambas. O papel eletrostático e o termosensível se alteram, sobretudo se expostos à ação da luz, e além disso as técnicas do tipo "ink-jet" produzem freqüentes problemas na conservação da cabeça de impressão. Pelo contrário, com o sistema "ink-shot" se utiliza papel normal, con-

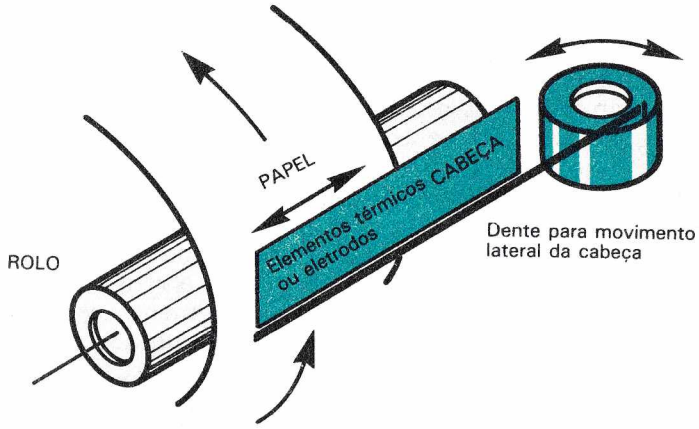


Fig. 1 - Estrutura simplificada de uma impressora térmica ou eletrostática.

seqüentemente não sujeito a deterioração e a tinta, sólida, esta contida em um cartucho especial alojado em uma cabeça móvel. O papel se desloca entre a ponta do cartucho (anódio) e uma placa de suporte (catódio). Quando chega o momento de gerar um ponto do caracter objeto de impressão se produz uma descarga entre os dois eletrodos, por causa da microexplosão, sendo arrastada até o papel em forma de microgotas que, ao golpear, formam o habitual ponto. Na figura 2 se ilustram ambos tipos de impressão.

Recursos similares a este (por exemplo, a técnica "Think jet" da Hewlett Packard, que projeta as gotas por esquentamento da cabeça) estão fazendo cada vez mais confiáveis e vantajosas as tecnologias de impressão por jorro de tinta.

Não queremos concluir o tema das impressoras de "não percussão", sem citar as novas e avançadas tecnologias que fazem uso do laser e que tem como princípio de funcionamento o utilizado nas máquinas fotocopiadoras normais. Neste caso, a impressão é muito rápida. Sua maior aplicação se encontra na reprodução rápida de páginas com gráficos de alta resolução, como as obtidas em sistemas para desenho assistido pelo computador (CAD). Nos modelos mais evoluídos, a impressão é à cores. Estas últimas são, até agora, objetos de luxo inclusive para um computador pessoal profissional.

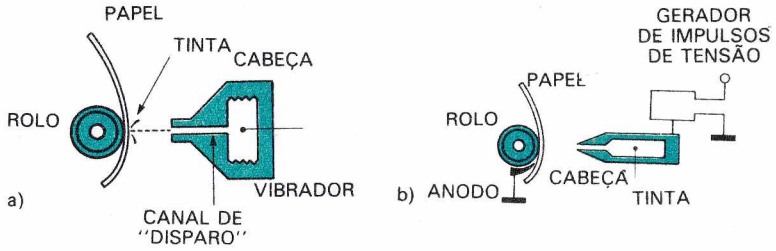


Fig. 2 - Estrutura simplificada de uma impressora de jorro de tinta ("ink-jet") (a) e de disparo de tinta ("ink-shot") (b).

Impressoras de impacto

Chegamos agora à tecnologia mais tradicional e difundida. Uma impressora de impacto proporciona documentos em papel (com possibilidade também de cópia automática, para obter um original e uma ou mais cópias imediatas), utilizando o provado principio da máquina de escrever que marca o caracter no papel golpeando o "martelo" com o relevo do caracter sobre a fita com tinta. Os primeiros modelos destas impressoras não eram outra coisa que máquinas de escrever controladas por um interface eletromecânico especial. É evidente que, levando em conta o conjunto dos martelos e do percurso que devem realizar cada vez (deslocamento até a fita e posterior retrocesso), a velocidade máxima admissível não pode ser elevada e um valor de 15 caracteres por segundo é bastante normal.

Se em lugar dos martelos se utiliza a bem conhecida esfera, como nas máquinas Olivetti ou IBM, por exemplo, a velocidade pode aumentar até mais do dobro, com a vantagem suplementar de poder substituir os tipos de caracteres quando se quiser e poder assim conseguir documentos com uma aparência verdadeiramente profissional.

Por último, se empregarmos a chamada "margarida", teremos uma nova possibilidade de aumentar a velocidade, já que a massa da cabeça de margarida é menor, porém em qualquer caso, não se pode superar, o limite de 50 cps (caracteres por segundo). Na figura 3 se mostra o sistema de impressão de margarida. Esta "flor" gira para levar o caracter a imprimir frente a alavanca de percus-

são. Com a finalidade de economizar tempo, os caracteres nas "pétalas" da margarida estão agrupados de modo que os mais utilizados estão próximos entre si. Além disso, a margarida gira durante o deslocamento horizontal do suporte que, por sua vez, se move com uma velocidade mais elevada quando tem espaços no texto, isto é, quando não se deve imprimir nada. Finalmente, a técnica de impressão por impacto incorpora uma otimização adicional: nas impressoras de margarida ou de esfera de alta qualidade não se chega ao final da linha se a última parte somente está constituída por espaços (espaço = nenhum caracter). Tudo isto faz economizar muito tempo e se podem imprimir até 70-90 cps (caracteres por segundo).

Trocando a margarida (ou a esfera) se obtém diferentes conjuntos e estilos de caracteres.

Os tipos de impressora que acabamos de descrever são utilizadas principalmente para trabalhos de escritório e, em geral, para se obter um documento de melhor qualidade. Como se diz comumente são consideradas impressora de "letter-quality" (qualidade de escrita). Esta denominação se deriva de que os caracteres impressos são bastante nítidos, posto que, cada um deles se consegue graças à ação de um relevo (martelo ou esfera) que incide sobre a fita com tinta. Se, além disso, consideramos o emprego, já muito difundido, de fitas de polyester do tipo "usar e tirar", se compreenderá que os caracteres levados sobre o papel são tão nítidos que parecem impressos em tipografia e, conseqüentemente, o resultado é altamente profissional. Como é lógico supor, também os custos são bastante elevados, portanto, estas máquinas estão destinadas sobretudo a profissionais, que as utilizarão em conjunto com sofisticados sistemas de tratamento de textos.

Quando, no entanto, o fator "velocidade de impressão" prevalece sobre a qualidade de escrita, é preciso passar a outras tecnologias de impressão, tal como a impressão de impacto com cabeça de agulhas que, como veremos mais adiante, permite novas prestações: semigráficas ou gráficas.

Na figura 4 se ilustra a disposição de uma típica cabeça de agulhas. Na prática, a cabeça é o suporte de onde costumam afluir 7 ou 9 micromartelos, cada um dos quais tem a forma de um pequeno cilindro, tão fino que costumam ser denominados agulha. Seu diâmetro é de uns décimos de milímetro. Cada agulha termina em um corpo mais consistente que é, na prática, o núcleo móvel de um eletroimã. Se excitarmos o eletroimã, a agulha será empurrada para fora do enrolamento e sobressaltará alguns milímetros da cabeça. Quando não se excita o eletroimã, uma mola faz retro-

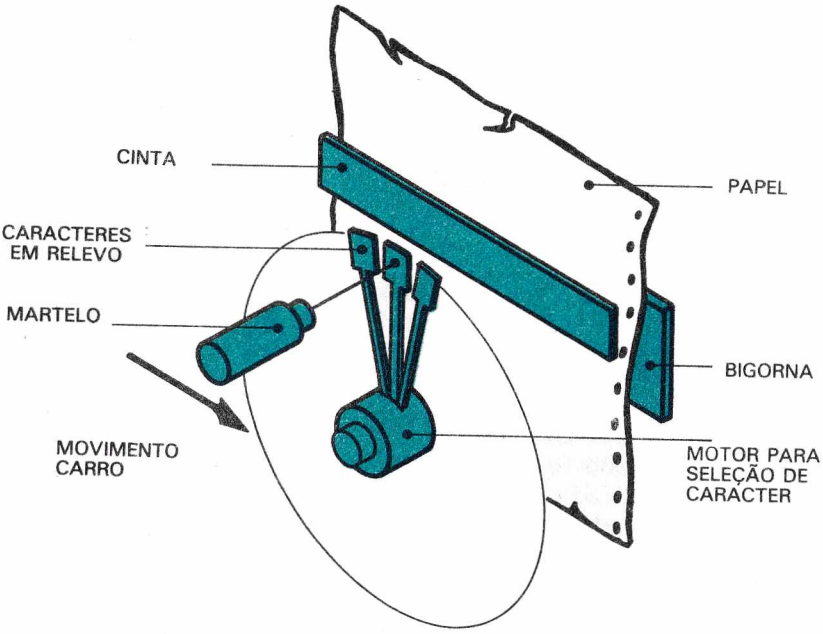


Fig. 3 - Mecânica de uma impressora de margarida.

ceder imediatamente a agulha até o interior de seu alojamento na cabeça.

Nove agulhas estão dispostas em fila vertical, uma em cima da outra, preparadas para golpear a fita com tinta para frente da qual se desloca a cabeça. Naturalmente, a percussão se poduz somente quando as agulhas se impulsionam para fora. No papel, arrastado por um rolo de borracha dura (denteado nos extremos, se utilizarmos papel contínuo), se imprimem o número de pontos correspondentes ao número de agulhas ativadas, dentre as 9 disponíveis. A fita costuma ser de nylon com uma tinta pouco gordurosa e a malha é tão fina que cada ponto resulta nítido e bem definido.

Agora estamos em condições de estudar como se realiza a impressão de um caracter completo e das diversas linhas.

A mecânica de arrasto é similar às das impressoras térmicas: a cabeça se desloca em sentido perpendicular ao deslizamento do papel e, durante seu movimento, as agulhas se lançam e se reco-

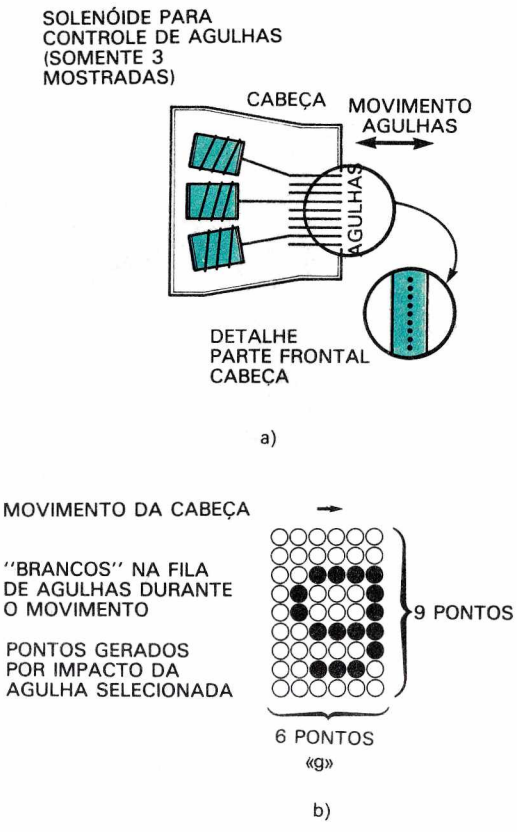


Fig. 4 - Cabeça de agulhas (a) e matriz de pontos (b) que se pode imprimir ao deslocar a cabeça sobre seu guia e acionar as agulhas.

lhem, repetidamente. Na figura 5 se ilustra o anteriormente exposto. Os caracteres se formam, fila após fila, segundo um esquema bem preciso que, por sua vez, emprega uma "matriz" de pontos armazenada no microcomputador que controla a impressora. Em cada passada da cabeça, da esquerda para a direita (e também da direita para a esquerda se a impressão é bidirecional), se imprime uma linha completa. O comprimento da linha está limitado somente pelo comprimento do suporte da cabeça.

Nos modelos mais sofisticados, as linhas de 132 caracteres são as mais habituais mas é evidente que, aumentando ou dimi-

nuindo a velocidade de arrasto da cabeça, se podem escrever mais ou menos caracteres fazendo-os mais largos ou mais estreitos.

No geral, a impressão "normal" é de 10 caracteres por polegada, a "expandida" (alongada ou evidenciada) de 7 ou 5 caracteres por polegada, e a "comprimida" de 16 caracteres e meio por polegada. A figura 6 mostra alguns exemplos de impressão, obtidos com uma impressora Epson LX-80.

Que velocidade se pode obter com uma impressora de agulhas? Também muito elevadas: de 300 caracteres por segundo, e inclusive maiores, se a impressão está otimizada, é bidirecional e o arrasto da cabeça se acelera em correspondência com os espaços. Não obstante, nestas impressoras nos encontramos com caracteres "pontilhados", pelo que a nitidez se reduzirá. A qualidade de escrita proporcionada por uma impressora de agulhas não poderá ser nunca comparável com a lograda máquina de esfera ou margarida, mas em aplicações tais como recibos, faturas, listas, etc., o usuário não busca certamente a beleza de um caracter em

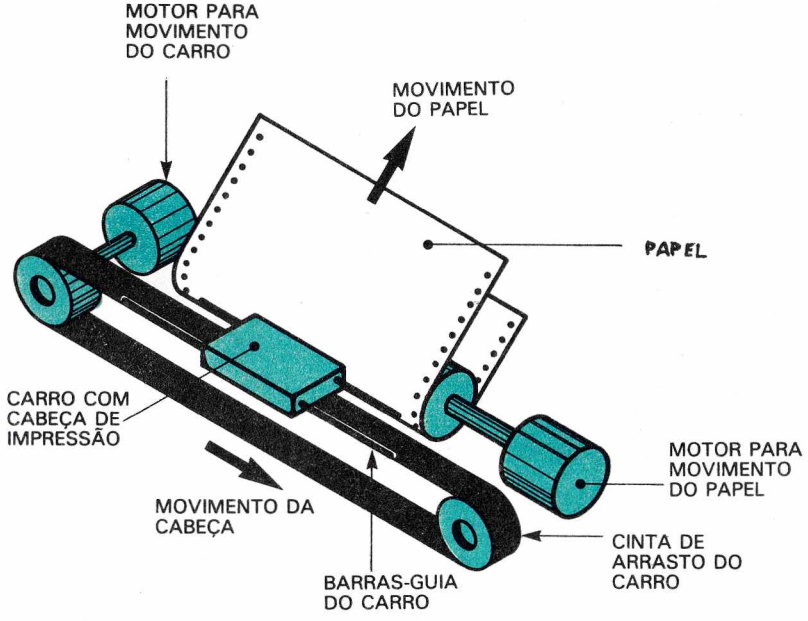


Fig. 5 - Estrutura de uma impressora de agulhas.

negrito ou em cursivo, além de conseguir que a escrita seja legível e que os impressos ou formulários se obtenham com rapidez. Além disso, tem uma vantagem que as de margarida ou esfera não tem: as impressoras de agulhas podem ser utilizadas como impressoras gráficas de pontos com uma alta resolução em algumas ocasiões. Com programas especiais é possível imprimir os mais diversos desenhos e figuras e não somente visualizá-los em tela.

Para concluir esta exposição, e antes de examinar a parte eletrônica, é obrigatório citar as impressoras que podemos ver nos "santuários" dos grandes computadores. Por exemplo, observa-se que muitos recibos de luz ou de gás tem caracteres impressos por martelo e não por agulhas. Isto significa que se empregou uma impressora de banda ou cadeia, capaz de proporcionar até 1.500-2.000 linhas de 132 caracteres por minuto.

O procedimento de impressão, segundo se ilustra na figura 7, é bastante simples. Uma cadeia, que gira com grande rapidez e sem interrupção, leva em relêvo inúmeras séries completas de caracteres alfanuméricos. O papel se descola em sentido perpendicular entre a cadeia, a fita com tinta e uma série de 132 alavan-

IMPRESION NORMAL, 10 C.P.I. (CAR. POR INCH):

ABCDEFGHIJKLMNOPQRSTUVWYYZ - 1234567890

IMPRESION COMPRIMIDA, 16,5 C.P.I.:

ABCDEFGHIJKLMNOPQRSTUVWXYZ - 1234567890

IMPRESION ENFATIZADA, 8,3 C.P.I.:

ABCDEFGHIJKLMNOPQRSTUVWXYZ - 1234567890

IMPRESION ENFATIZADA, 5 C.P.I. (CAR. POR INCH):

ABCDEFGHIJKLMN — 12345

Fig. 6 - Exemplos de impressão com cabeça de agulhas.

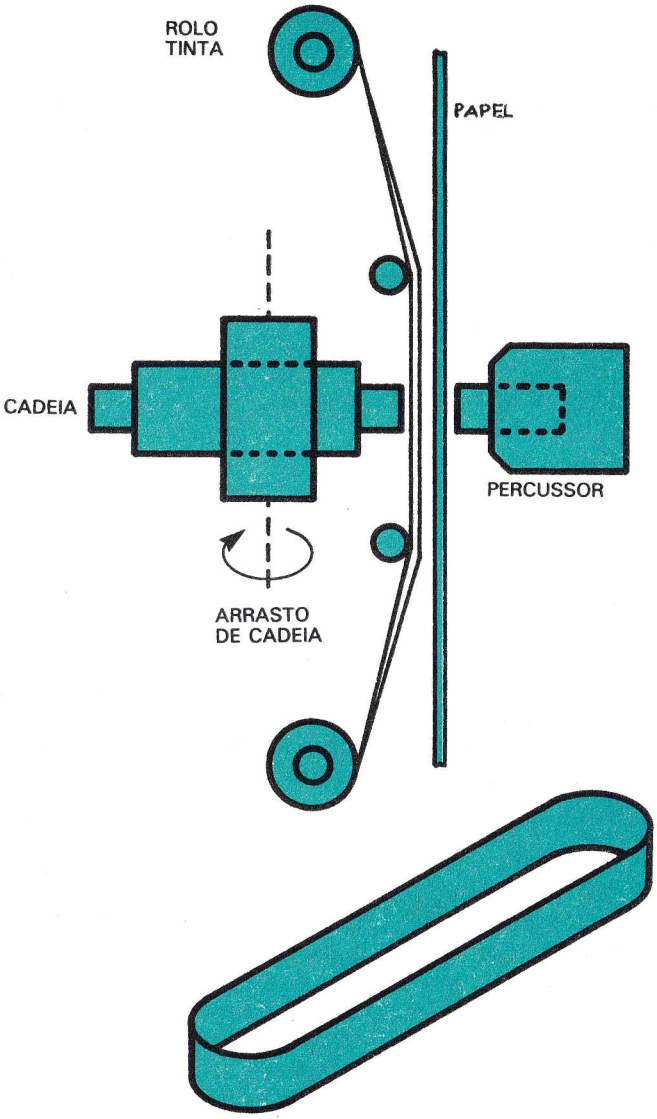


Fig. 7 - Mecanismo de uma impressora de cadeia.

cas percussoras contíguas, com uma separação de 1/10 de polegada. Quando um caracter deve estar em uma determinada posição, passa para diante do percussor correspondente, este último se dispara até o papel e o caracter fica impresso no mesmo. O procedimento se repete com rapidez ao longo da mesma linha, já que a cadeia gira de forma contínua e, tal como foi indicado, leva várias séries de caracteres alfabéticos em relevo, uma após a outra. Naturalmente, durante a impressão de cada linha estará bloqueado o arrasto do papel, mas os saltos são tão rápidos que o movimento parece contínuo. Um grande incoveniente das impressoras de cadeia é seu alto nível de ruído: é tão elevado que nos centros de processamento de dados se costumam instalar em recintos isolados acusticamente. No entanto, são tão caras que não é provável que você queira assumir o risco de despertar os vizinhos adquirindo uma para seu computador pessoal.

Conexão ao computador

A impressora é um dos periféricos denominados "inteligentes" porque sua complicada parte mecânica está controlada por um pequeno, mas potente, microcomputador. Este trabalha de modo que o programa que se executa depois da conexão é único e não modificável. Seu único encargo é controlar todas as funções de impressão e, naturalmente, aceitar as ordens e dados do computador, ao que está conectada a impressora.

Em definitivo, nosso computador pessoal, na fase de impressão, não faz outra coisa que transmitir ordens e dados, em uma sucessão muito rápida, ao microcomputador que se encontra no periférico.

Esta comunicação se produz fundamentalmente através de dois "ports": um de saída (em nosso computador pessoal) e outro de entrada (no periférico). Na figura 8 vemos com detalhe o diagrama de blocos de uma impressora de agulhas típicas. O quadrado central encerra praticamente toda a parte eletrônica, constituída pelo microcomputador e a parte de controle dos elementos de potência: motores e bobinas das agulhas. Os motores são dois, freqüentemente do tipo "passo a passo".

Um deles controla o avanço do papel, fazendo girar o rolo trator, e o outro atua sobre uma correia que arrasta a cabeça de impressão ao longo de um suporte horizontal. Uma vez que se liga a impressora, o programa armazenado na memória ROM de seu microcomputador produz uma iniciação geral: agulhas retraídas,

cabeça retida e finalmente o princípio de sua carreira, rolo parado. O μP da impressora fica então atento ao "port" de entrada esperando "novidades". Quando o computador pessoal inicia uma impressão começa a transmitir uma seqüência de códigos, cada um dos quais tem um significado bem preciso na codificação ASCII (veja o capítulo anterior). O μP existente na impressora está programado para reconhecer ditos códigos ASCII, de forma que pode ter lugar a transferência dos dados até o periférico e sua interpretação. Caracter após caracter, um programa que se está executando dentro do nosso computador pessoal lê o conteúdo da memória que queremos imprimir e transmite todo ao exterior através de seu "port" de saída, ao qual está conectado o cabo da impressora. Os dados, em paralelo ou em série (dependendo do tipo de conexão escolhida entre o computador pessoal e o periférico), chegam um após o outro ao "port" de entrada do μP que controla a impressora, a qual os lê e introduz na memória. No entanto, a comunicação não é contínua, já que a transmissão de um caracter entre o computador pessoal e o periférico se realiza de forma muito mais rápida que a posterior impressão. A dificuldade se resolve programando a impressora para que admita cada vez tantos caracteres quantos possam estar, como máximo, em uma linha de impressão.

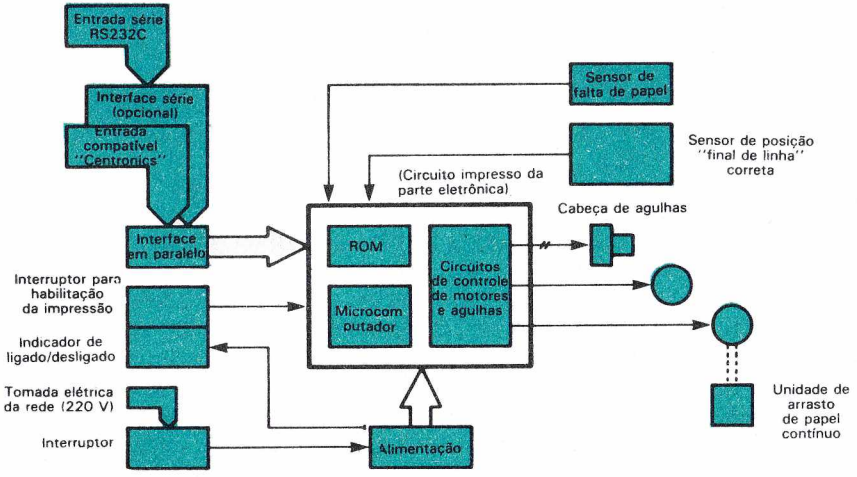


Fig. 8 - Diagrama de blocos dos elementos que constituem uma impressora de agulhas.

Em definitivo, a seqüência é a seguinte: 1) o computador pessoal toma um caracter de sua memória e o escreve no "port" de saída, 2) espera para que a impressora lhe dê a resposta ("OK, foi tomado o caracter, manda-me outro") mediante o envio de um sinal ACK e 3) volta ao ponto 1.

Para a impressora, a seqüência é: 1) espera um caracter procedente do computador pessoal, 2) quando chega o caracter (em ASCII), o lê no port de entrada e o introduz na memória, 3) avisa ao computador pessoal ("tomei o dado, manda-me outro"), 4) a seqüência se repete até completar o número de caracteres que tem em cada linha (40, 80, 132) enviando então ao computador um sinal de "ocupado", 5) imprime a linha completa (o computador pessoal fica esperando que a impressora fique livre e, portanto, não transmite) e 6) terminada a impressão da linha, a impressora avisa ao computador pessoal de que está novamente preparada e a seqüência volta ao ponto 1.

Quando a impressora tem em memória, sob a forma de um bloco de códigos ASCII, toda a linha a imprimir, o microcomputador que a controla se dedica completamente ao controle dos motores e das agulhas de cabeça. Esta última começa a ser arrastada da esquerda à direita e, durante seu percurso, as agulhas se lançam de modo que formem, no papel, mediante pontos, os caracteres correspondentes aos códigos anteriormente armazenados na memória.

Se você se fixar na figura 4, poderá ver como se forma um caracter. Basta golpear durante o movimento todos os pontos que compõem o contorno do caracter, selecionando-os no momento oportuno entre os 6 x 9 disponíveis. A linha se imprime em uma só passada, já que temos 9 agulhas em coluna, uma em cima da outra. Ao finalizar a impressão da linha, o microcomputador detém o deslocamento da cabeça e faz avançar o motor de arrasto do papel uma linha para cima. Assim é possível iniciar a impressão da linha seguinte, uma vez que esteja carregada na memória.

A impressão pode produzir-se ao ir da esquerda para a direita somente (em cujo caso, há de levar-se a cabeça ao final de seu percurso), ou também se imprime ao ir da direita para a esquerda (procedimento típico dos modelos mais aperfeiçoados). Em tal caso, a impressão se denomina bidirecional e, ao eliminar o retorno ao início de nova linha, a velocidade aumenta de forma considerável.

Também é fácil compreender que se ao invés de controlar as agulhas com o único objetivo de imprimir caracteres com pontos previamente determinados, se controlam de forma individual, se

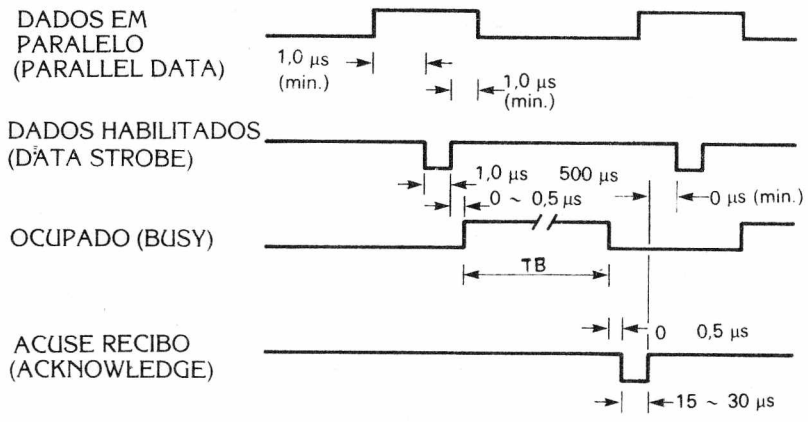


Fig. 9 - Sinais de comunicação entre o computador e a impressora (interface paralelo).

poderia gerar todas as figuras e desenhos que desejamos, inclusive os constituídos por seqüências de pontos. Naturalmente, serão necessárias mais passadas e a impressão será mais leta, mas poderemos conseguir ótimos resultados gráficos, representando em papel diagramas de barras, projeções ortogonais, desenhos em perspectiva, imagens, etc. Em tal caso, se diz que a impressora tem incorporada a opção gráfica.

Na figura 9 se mostra um exemplo dos sinais de comunicação adotados entre o computador pessoal e uma impressora, se a conexão se realiza através de uma interface paralela, enquanto que para o caso de que esta interface seja compatível com o Centronics (normalizado) a figura 10 contém a descrição de todas as linhas típicas de controle.

Como selecionar uma impressora

Para concluir esta visão panorâmica das impressoras, seria conveniente ver alguns critérios de seleção. Antes de tudo, deve basear-se no tipo de atividade a que se aplicará, ou seja, se o processador está destinado a utilizar-se, sobretudo, para desenvolver

Sinal	N.º * patilha	Tipo para a impressora	Descrição
Data Strobe	1	Entrada	Chega desde o computador para indicar à impressora que entrou um dado.
Data Bit 1 Data Bit 2 Data Bit 3 Data Bit 4 Data Bit 5 Data Bit 6 Data Bit 7 Data Bit 8	2 3 4 5 6 7 8 9	Entrada Entrada Entrada Entrada Entrada Entrada Entrada Entrada	8 bits do dado enviado pelo computador
Acknowledge	10	Saída	Indica ao computador que o caracter foi recebido de forma normal
Busy	11	Saída	Indica ao computador que a impressora está ocupada
Paper out (esgotam. papel)	12	Saída	Indica falta de papel
Select	13	Saída	Indica se a impressora está, ou não, ativada
OV	14 16 44	E/S	Massa de sinal
CHASSIS GROUND	17	E/S	Massa chassis
+ 5 V	18	Saída	Alimentação suplementar 50 mA
OV	19 a 30	E/S	Retorno massa sinais
* Do conector			

Fig. 10 - Linha de interface paralelo compatível com Centronics.

software de aplicação (programas concebidos para outros computadores ou para a automatização industrial, por exemplo), será útil uma impressora muito rápida, com possibilidades gráficas, sendo a mais indicada uma de impacto com cabeça de agulhas. Por

outro lado se a impressora está destinada para trabalhos de escritório, ou onde o computador pessoal tem funções tipicamente de gestão para a obtenção de documentos, pode ser conveniente uma impressora de boa qualidade de escrita (de esfera ou de margarida). O mesmo pode se dizer com respeito àquelas aplicações nas quais se imprimem os resultados de processamento de textos, correspondência automatizada, etc. Para trabalhar com um computador pessoal em uma fase na qual as inversões estejam limitadas pode ser conveniente a aquisição de uma impressora térmica ou uma eletrostática. Para aplicações no campo industrial são recomendáveis as impressoras térmicas ou modelos com cabeça de agulhas e um número limitado de colunas (um máximo de 20 a 25), porque o espaço ocupado deve ser mínimo.

Os preços variam não sempre de modo proporcional à utilização da máquina, ainda que se possa dizer que os custos mais moderados são características dos modelos de agulhas e esta é a razão de que seja o tipo de impressora mais utilizado como periférico para os computadores pessoais atuais.

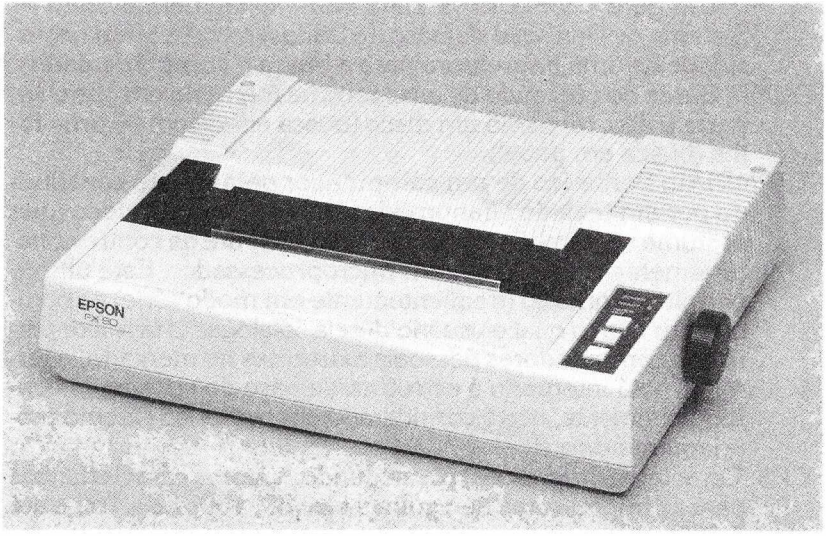


Foto 4 - Outro modelo de impressora de agulha.

PAPEL CONTÍNUO (OU FORMULÁRIO CONTÍNUO): Faixa de papel cujas bordas estão perfuradas a intervalos determinados e em correspondência com os dentes existentes nos extremos do rolo porta-papel da impressora, onde podem introduzir-se e "prender-se" para assegurar um arraste seguro de papel. Cada 28 cm. aproximadamente, o papel tem um picote, que facilita também a separação de folhas individuais. Assim dobrado e empacotado "na forma de acordeão" o papel contínuo é fornecido em pacotes de 1.000 folhas ou mais.

BANCO DE DADOS ("Data Base"): Sistema de dados, ordenados conforme seqüências bem determinadas, definidas a priori, pelo usuário, que reside na memória do computador (em disco ou em RAM). Com as oportunas ordens do sistema operacional, o usuário pode selecionar ou extrair os dados que possuem características similares, tais como: "direções de todos os clientes cujo sobrenome comece por [BRA]", "números de todos os recibos emitidos em 15/06/1984", etc. Aqueles que trabalham com bases de dados e com os programas que as gerenciam (DBMS, Data Base Managment System — Sistema de Gestação de Base de Dados) sempre terão necessidade de uma impressora para a busca ("dump") de dados.

DUMP: Busca do conteúdo de uma zona de memória em outro suporte físico, tal como um disco (busca em disco) ou uma folha (busca em papel).

TERMINAL: Periférico de um computador geralmente constituído por um teclado alfanumérico, um monitor de vídeo (que costuma ser também gráfico) e um hardware de controle, naturalmente baseado em um microprocessador. Este último permite a conexão (freqüentemente em modo série) ao computador com o qual o usuário deseja "dialogar". Na maior parte dos computadores pessoais existentes no mercado, o terminal está integrado à estrutura de base do sistema e, conseqüentemente, não é considerado um periférico externo propriamente dito.

CPS: Caracteres (impressos) por segundo. Valores característicos para as impressoras de agulhas são: 80, 100, 120, 160 e até 3.000 cps.

CPI: Caracteres (impressos) por polegada (inch). Medida da "densidade" da impressão. Quanto mais caracteres tem em uma polegada mais comprimida será a impressão e vice-versa. Os

distintos valores de compressão se obtém deslocando a cabeça a diversas velocidades, enquanto que as agulhas são lançadas à mesma cadência de uma impressão normal, ou seja, quanto mais rapidamente se move a cabeça, menos comprimida será a impressão e mais baixo será o valor de CPI.

MOTOR PASSO A PASSO: Se trata de um motor especial de corrente contínua que não tem um movimento giratório uniforme, além de avançar em saltos, ainda que pequeníssimos. Posto que depois de cada salto o motor pode bloquear-se em uma posição estável, seu emprego está especialmente recomendado em todas as aplicações para máquinas nas quais os avanços e os deslocamentos devem ser discretos, isto é, em saltos, e precisos.

CAPÍTULO VII

MEMÓRIAS DE MASSA E OUTROS PERIFÉRICOS DE ENTRADA/SAÍDA

Para seu correto funcionamento, um computador deve ter necessariamente, além da CPU, uma quantidade de memória adequada. Vimos nos capítulos anteriores que parte desta memória deve ser ROM (permanente) e outra parte RAM (alterável pela CPU). Quando se deixa de aplicar a alimentação, a memória RAM perderá todo seu conteúdo, por isto é preciso dotar sempre ao sistema de um meio para "salvar" os dados da memória de modo que se possam recuperar quando voltar a ligar a máquina. Se utilizam com este fim periféricos denominados "memórias de massa".

Existem muitas técnicas para copiar o conteúdo da RAM no exterior do computador, mas as mais empregadas utilizam suportes magnéticos tais como fitas de cassete e discos. Estes últimos podem ser flexíveis ou rígidos (também chamados duros). A busca dos dados da memória no suporte magnético se efetua através de uma interface especial que costuma estar constituída por chips do tipo de integração em alta escala (VLSI). Um programa, residente na memória ROM do computador, extrai da memória um dado de cada vez e o transmite ao chip da interface, que "se vê" desde a CPU como um port normal de E/S. Este chip, que costuma denominar-se controlador (de discos ou de fita) gera sinais elétricos modulados que dependem exatamente dos "uns" e dos "zeros" do dado binário recebido e que chegam, através de um cabo blindado especial, a uma cabeça de gravação sob a qual se desli-

za o suporte magnético (fita ou disco). Seja qual for, o suporte se magnetiza e, ao final da gravação, conserva uma cópia dos dados que estão na memória.

Para devolver os dados à memória central, basta que o chip controlador leia o suporte magnético com a cabeça de leitura (que é a mesma utilizada na fase de gravação), decifre os sinais elétricos extraídos e coloque o dado resultante no port para que o computador o leia.

Existem três tipos de suportes magnéticos muito difundidos. O primeiro é a fita de cassete, para a qual empregamos um gravador normal. O processo é lento, porém, muito econômico.

O segundo método emprega como suporte, discos magnéticos feitos do mesmo material que a fita. Se denominam discos flexíveis ou disquetes ("floppy-disk", em inglês). Para sua escrita e leitura se utilizam unidades especiais, denominadas unidades de disco, que são mais complexas que um gravador de cassete, mas muito mais rápida (até 200 vezes mais). Oferecem, além disso, a grande vantagem do denominado acesso direto, que descreveremos mais adiante. Naturalmente, o custo é bastante elevado, ainda que no conjunto se compensa se levarmos em conta a quantidade de informação que pode gravar-se em um disco flexível em comparação com uma fita de cassete normal.

O terceiro sistema de armazenamento massivo utiliza os chamados discos duros rígidos ("hard-disk", em inglês). Se parecem em seu aspecto aos discos flexíveis, mas são de material metálico e estão envolvidos em invólucros especiais herméticos. Um disco rígido permite armazenar até 50 vezes mais dados que um disco flexível das mesmas dimensões, o que é possível pela ausência de poeira e a precisão do suporte metálico que permite gravar a informação de modo muito mais compacto.

Acessos seqüencial e aleatório

Uma distinção importante no campo das memórias se baseia no método de acesso, isto é, na forma mediante a qual o computador pode extrair delas a informação. Se fala de acesso "seqüencial" e de acesso "direto" ou "aleatório" (em inglês, "random"), métodos característicos de fitas e de discos, respectivamente.

No caso de acesso seqüencial se quisermos ler (ou escrever) os dados situados em uma posição, e nos encontramos em outra, teremos forçosamente que recorrer e ler todos os dados intermediários, mesmo quando não nos interessem em absoluto. Isto se

vê muito claramente no meio típico de acesso seqüencial: a fita.

Em caso contrário é, por exemplo, o dos discos flexíveis: sua superfície está subdividida em pistas circulares concêntricas (isto é, não se trata de uma só pista contínua em espiral, como nos discos musicais) e cada uma destas pistas está, por sua vez, dividida em setores (normalmente, em um disco flexível de 8 polegadas — 20,32 cm. — se tem 77 pistas por face, cada uma com 8 setores capazes de armazenar cada um, uns 250 bytes). Quando queremos aceder a um setor particular, o braço porta-cabeças se desloca até situar-se sobre a pista correspondente e espera a chegada do setor (evidentemente o disco se põe a girar cada vez que se realiza uma operação de leitura/escrita) passando assim sobre o mínimo e indispensável número de dados que não interessam. Com ele se aumenta enormemente a velocidade e, sobretudo, a flexibilidade com que se realizam as operações de E/S, o que é fundamental na gestão dos fichários (arquivos de dados) e no funcionamento dos sistemas operacionais. Grande parte destes sistemas são conhecidos com a abreviação DOS (Disk Operating System — Sistema Operacional de Disco) porque controlam a unidade de disco e se guardam em parte neste, até que se necessitem (momento no qual se passam à memória central).

A propósito dos discos, outro ponto importante é o denominado "formato". Se trata da escrita preliminar de sinais adequados no disco (antes de que o utilizemos). Servem fundamentalmente para duas coisas: 1) criar pontos de referência (como sulcos magnéticos) e de sincronismo na leitura/escrita, e 2) definir a posição de início dos dados propriamente ditos. Esta operação, lamentavelmente diferente de um computador a outro, é uma das primeiras coisas que se aprende a fazer com um computador pessoal. Rouba um pouco de espaço aos dados e por isto será preciso distingüir a capacidade de um disco flexível "formatado" ou "não formatado" (contando unicamente a primeira para os fins práticos).

Cada sistema deve ter o periférico de memória de massa que melhor se adapte às suas dimensões e a seu grau de sofisticação. Um computador pessoal para atividades caseiras, por exemplo, está provido de interface para gravadora de cassete na versão básica, e só com caráter opcional poderá dotar-lhe de uma unidade de disco flexível. Pelo contrário, um computador pessoal evoluído, utilizado em aplicações de gestão ou científicas, deverá ter uma estrutura de base que compreenderá ao menos uma unidade de disco, posto que, se não for assim, seria impossível trabalhar de um modo sério e, sobretudo, produtivo. Assim mesmo, os modelos mais sofisticados poderão ter como configuração básica duas uni-

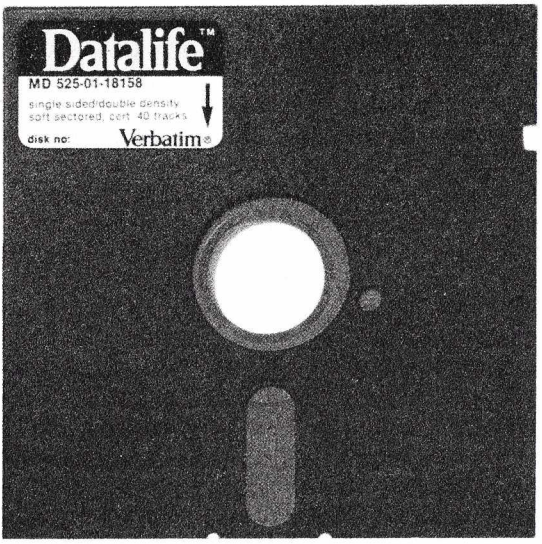


Foto 1 - Disco flexível de 5 1/4".

dades de disco, ou então, uma unidade de disco rígido, com grande capacidade e velocidade. A vantagem de uma memória de massa rápida não é somente a de poder "salvar" tudo o que tem no computador antes de sua desconexão, mas também a de permitir uma gestão ágil dos recursos do software do computador, utilizando os discos enquanto se deslocam blocos de dados, programas, processos parciais de cálculo, etc.

Para aproveitar as unidades de disco é necessário que o sistema operacional do computador seja sofisticado, rápido e potente ao mesmo tempo, e capaz de controlar qualquer transferência de dados, desde ou até os discos, de maneira mais simples possível.

Um bom sistema operacional para o controle dos discos é talvez em certas circunstâncias o que mais condiciona ao usuário na escolha da máquina. Sistemas operacionais de grande difusão e prestações comprovadas são: o CP/M (Control Program for Microcomputers) para Unidades Centrais de Processamento (CPU) tais como Z80, 8080, 8085 e 8086/8088 (para esta é o CP/M-86 da Digital Research), o DOS 3.3 e o Pro-DOS para Apple, o PC/MS-DOS (da Microsoft) para o 8088 do computador IBM-PC, etc.

Foto 2 - Microdisco flexível de 3 1/2" em cartucho de plástico (ainda não difundido no Brasil).

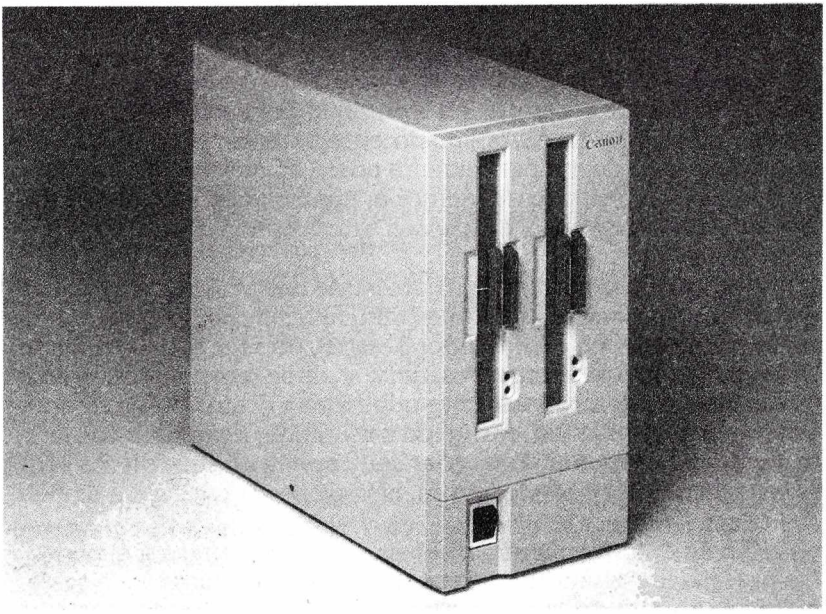


Foto 3 - Exemplo de um conjunto de duas unidades de disco flexível.

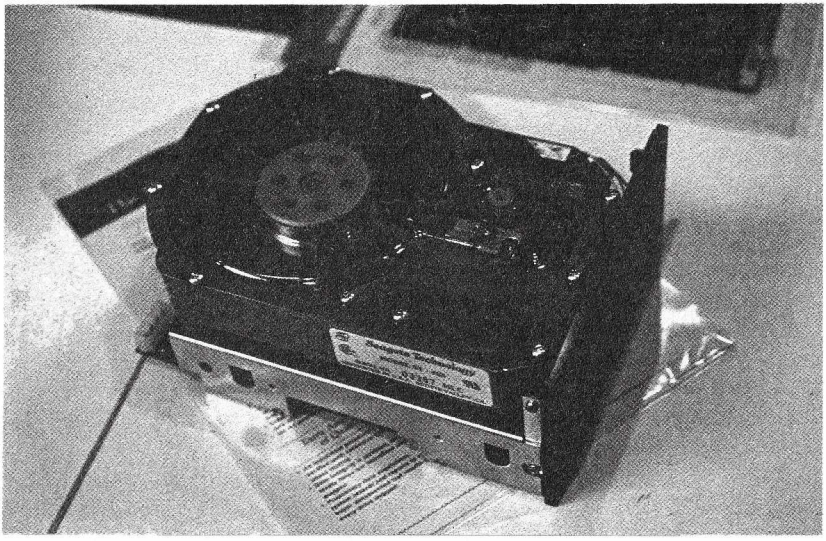


Foto 4 - Unidade de disco rígido sem a tampa, onde pode-se ver o disco.

Um Sistema Operacional de Disco (DOS) deve integrar uma série completa de ordens fáceis e potentes para o controle da memória de massa com outra, igualmente potente, série de ordens para manusear, da melhor maneira possível, os recursos internos do computador: espaço de memória, apresentação visual dos dados e/ou dos programas, execuções simultâneas de vários programas, etc.

Também é verdade que a fiabilidade dos próprios periféricos de memória de massa está em estreita relação com as boas características globais do computador pessoal, ou seja, quanto mais se desce até as atividades caseiras, tanto mais se tem que fazer frente a evidentes compromissos entre utilidades e custo. Tenha sempre em mente que uma vez desligada a máquina, deve ter a completa segurança de poder voltar a ler seu cassete ou seus discos sem erros, posto que, se não for assim, haveria perdido tempo e dinheiro.

Finalmente, lhe damos uma recomendação que, por experiência própria, lhe aconselhamos seguir ao pé da letra: se você se decidir pela compra de uma máquina fiável e de boas prestações, não busque economizar alguns cruzados adquirindo suportes magnéticos de baixa qualidade, que ao final resultarão mais caros.

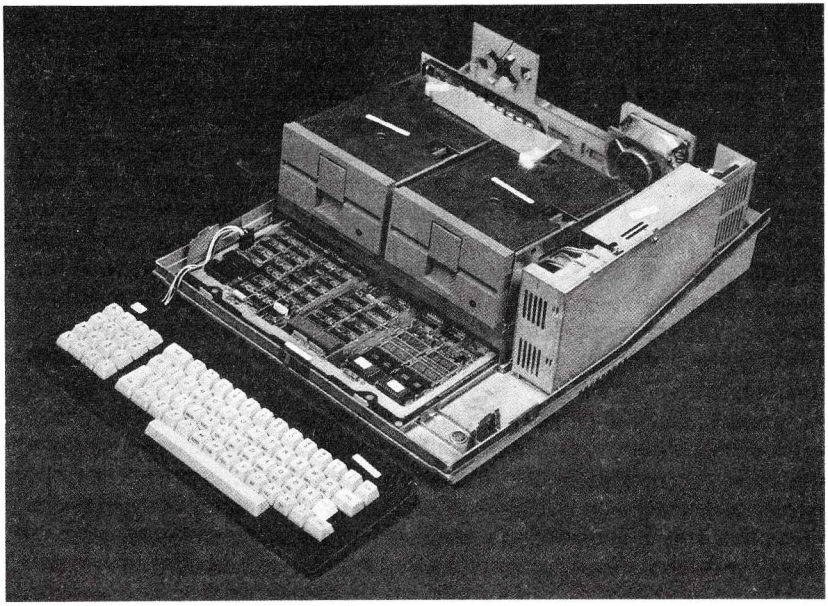


Foto 5 - Vista interna de um computador pessoal, onde pode-se observar as duas unidades de disco flexível de 5 1/4 polegadas, que estão integradas no sistema.

Como ilustração de nossa exposição sobre as memórias de massa, as fotografias 1, 2, 3, 4 e 5 mostram o aspecto que apresentam um disco flexível, tanto de 5 1/4 polegadas como de 3 polegadas e meia, um periférico típico que contém duas unidades de disco flexível, uma unidade para disco rígido e finalmente, duas unidades de disco integradas na estrutura básica de um computador pessoal.

Outros periféricos: A) Periféricos "De saída"

Agrupamos nesta categoria todos os periféricos que permitem ao sistema enviar seus dados até o mundo externo, pelo que a dita categoria pertencem: as impressoras, as telas de texto e/ou gráficos, os dispositivos traçadores ("plotters")e os interfaces musicais e vocais. Faremos uma descrição rápida das unidades mais significativas, excluindo as impressoras, que mereceram um item próprio.

Plotter

Um plotter ou traçador de gráficos é uma máquina capaz de desenhar qualquer tipo de figura por meio de uma espécie de lápis que se desloca sobre uma folha de papel, normalmente fixa. O lápis é controlado, em seu movimento, por dois motores: um que pode deslocá-lo ao longo da direção eixo x, fazendo que se mova o braço porta-lápis, e outro que se move na direção "y", perpendicular a "x", sobre o próprio braço, pelo que o lápis pode levar-se com finalidade a um ponto qualquer do plano x-y. Na fotografia 6 se mostra o aspecto de um moderno plotter, cuja parte eletrônica de controle está governada por um microprocessador incorporado.

Quando uma pessoa desenha, desloca sua mão controlando o movimento com os olhos, realizando uma comparação de cada posição com a imagem, praticamente acabada, que tem em seu cérebro. O mesmo acontece no caso do plotter: o lápis se leva, em cada momento, a uma posição da folha de papel, cujas coordenadas se enviam desde o computador ao qual está conectado, e que tem em sua memória o desenho traduzido já em pontos. O desenho se consegue, pois, com uma série de traços infinitesi mais, que correspondem cada um ao rastro desejado pelo lápis em seu percurso de um ponto a outro muito próximo. Para uma realização "limpa" dos desenhos é necessário que o computador que trans-

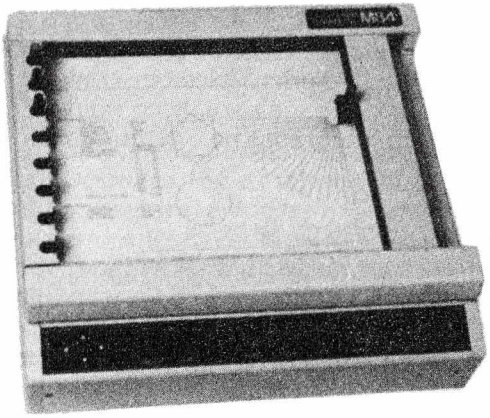


Foto 6 - Exemplo de um plotter.

mite as ordens ao plotter possua um software bastante sofisticado.

O plotter é um periférico com prestações espetaculares e que põe à disposição do usuário do computador pessoal grandes possibilidades gráficas que não podem proporcionar a tela nem a impressora por si só. Lamentavelmente, é todavia uma unidade de relativamente alto custo, ainda que, com a evolução da tecnologia e a conseqüente redução dos preços, existe a expectativa de que, em um futuro não muito distante, se produza a aparição no mercado de dispositivos com grande utilidade e preços mais moderados. Na fotografia 7 se ilustra alguns exemplos de desenhos realizados com o plotter.

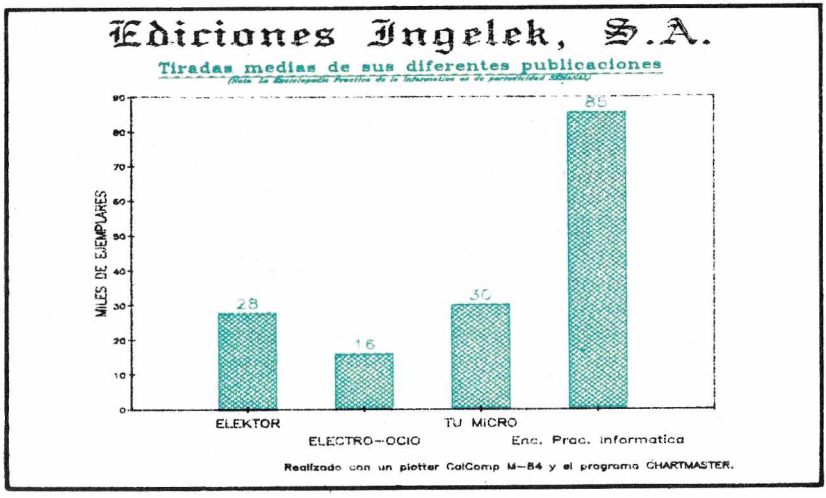
Terminais

A visualização de dados em forma de gráficos ou, mais simplesmente, de caracteres alfanuméricos, imprescindível nas aplicações informatizadas, se logra mediante periféricos sofisticados que se denominam "terminais". Um terminal é um periférico tanto de entrada como de saída, posto que admite também dados através do teclado alfanumérico.

A parte do terminal que se encarrega da entrada desde o teclado é banal e pouco sofisticada (como se torna evidente no simples exemplo de microcomputador pessoal descrito no capítulo quarto), enquanto que a visualização é uma operação delicada e que merece algumas observações complementares.

Recorda como funciona um televisor? Pois bem, um terminal é algo muito similar. Também neste caso tem um feixe de eléctrons que, ao percorrer de uma maneira pré-estabelecida a tela, excita o fósfor que a recobre e desenha as figuras. O feixe de eléctrons, que explora ou varre a tela em 1/50 de segundo, a percorre em sua totalidade mediante uns traços, que se dispõem horizontalmente uma debaixo da outra (o que se denominam linhas), que cria no sentido da esquerda à direita e de cima para baixo (como lemos a página de um livro). Na figura 1 se mostra uma representação esquemática do "pincel" de eléctrons.

Ao final do quadro, depois de ter percorrido 625 linhas, o pincel eletrônico volta ao ponto denominado "I", e começa desde o princípio seu zaguezagueante percurso. Durante os finais de linha e de quadro, o pincel se apaga, não aparecendo estes traços "de retorno" no fósfor que recobre a tela, pois se o pincel eletrônico estivesse sempre aceso, veríamos todas estas 625 linhas e os tra-



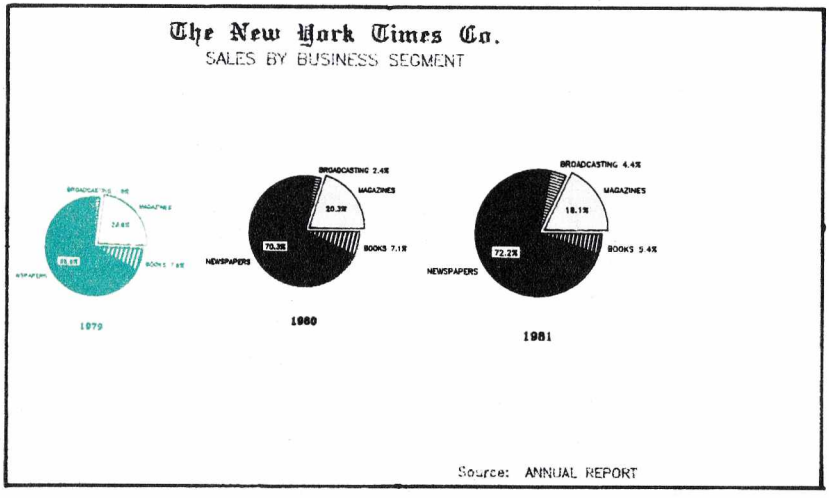


Foto 7 - Alguns gráficos obtidos com um plotter.

ços de retorno nítidos, brancos e tão próximos uns dos outros que a impressão seria a de um único retângulo branco. Mas o pincel pode tomar dois estados distintos (aceso e apagado), além de ter uma intensidade regulável.

Se um circuito controla o percurso do pincel eletrônico, e, enquanto isso, em sincronismo com todas as demais fases da apresentação visual, o acende e o apaga, se gerarão na tela pontos u, ocasionalmente, segmentos. Aproximando de modo controlado estes pontos e/ou segmentos, se podem realizar com facilidade contornos de caracteres alfanuméricos ou gráficos. O controle de aceso e de apagado do pincel eletrônico se realiza por meio de um só chip integrado VLSI, denominado "controlador de CRT" (Cathode Ray Tube = tubo de raios catódicos).

Deste modo a CPU não tem que preocupar-se em absoluto de seguir as fases da apresentação visual e de sincronizar-se com as evo-

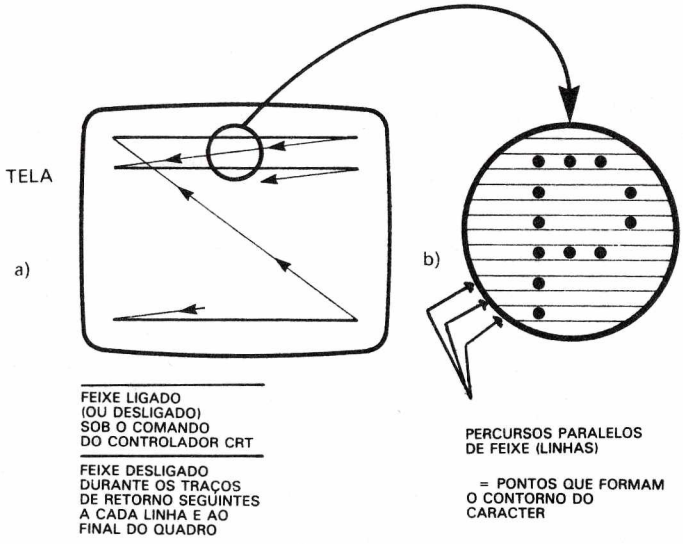
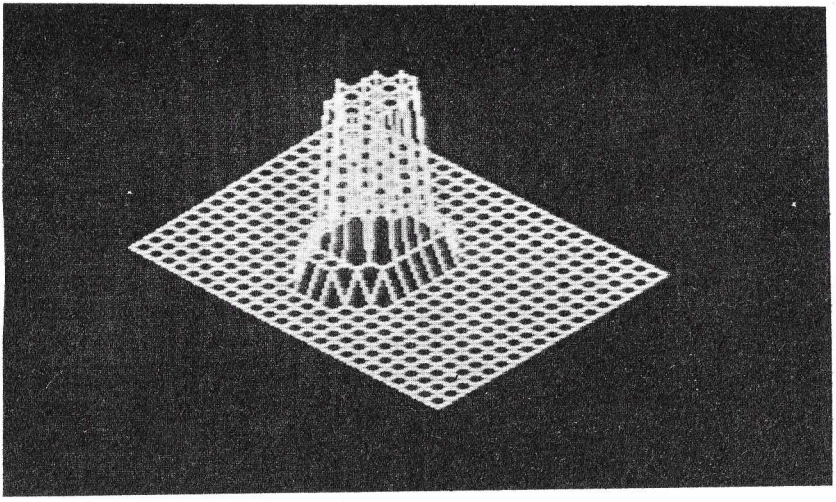


Fig. 1 - a) Percurso de feixe de eléctrons na tela durante um quadro. Cada segundo se realizam 50 quadros, constituído cada um por 625 linhas. b) Detalhe da geração de um caracter em uma tela de CRT. Os pontos são as posições das diversas linhas onde o chip controlador de CRT liga o feixe de eléctrons, pelo qual pode-se ver pontos luminosos. O efeito total é o de delimitar o contorno de um caracter alfanumérico ou gráfico. No exemplo foi gerada a letra "P".



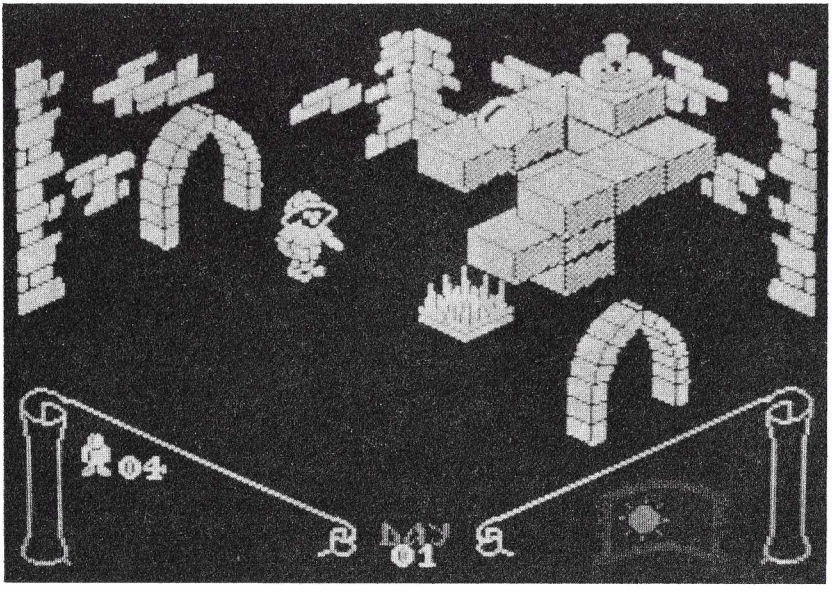


Foto 8 - Alguns gráficos realizados em telas de média resolução.

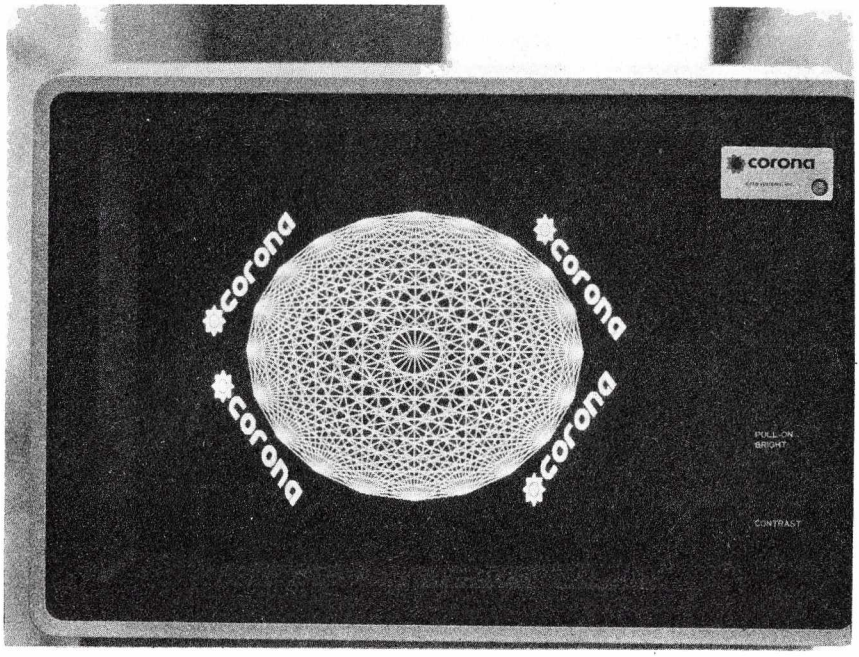


Foto 9 - Imagem de uma tela com gráficos de alta resolução.

luções da face eletrônica, porque de todo ele se encarrega o controlador da CRT, que a CPU "vê" como um port de saída normal. Controladores muito usados são o 6545 da Rockwell (utilizados nos equipamentos com CPU 6502 como o AIM-65 e o Apple) e o 8275 da Intel (utilizado no IBM-PC).

Na figura 1b se mostra o efeito que se obtém acendendo o pincel somente em determinadas posições de seu percurso. No exemplo se gera o caracter "P". Se a CPU continua transmitindo novos dados ao chip controlador, este gerará novos caracteres ou sinais gráficos até que toda a tela esteja cheia. Então, para que os novos caracteres não substituam, por superposição, aos antigos, se pode mandar a ordem oportuna ao chip controlador de CRT, que irá deslocando para cima as linhas. Alguns terminais gráficos podem controlar até 1024 x 1024 pontos independentes. Pode imaginar-se a incrível resolução (definição) das figuras geradas em terminais desta natureza. Por curiosidade, há somente três anos necessitava-se um mínimo de 50 circuitos integrados diferentes para realizar

Foto 10 - Exemplo de uma interface de voz.

o que atualmente se consegue com dois ou três chips e, natural mente, são bem menores.

Na atualidade existem poucos computadores pessoais providos de terminal externo. Quase todos estão providos de teclado integrado na estrutura, e de interface interna de vídeo, bastanto portanto conectar um monitor (ou aparelho de TV normal) e tudo funcionará de maneira imediata.

Na fotografia 8 se mostram alguns exemplos de gráficos realizados em telas com uma resolução média, enquanto que na fotografia 9 vemos a nitidez das imagens de alta resolução.

Interfaces de som (voz e música)

O computador pessoal tem sido utilizado, desde sua introdução no mercado pelos apaixonados da música, devido a suas enormes possibilidades de controle tão rápido que é quase em tempo

real igual ao dos parâmetros dos novos instrumentos musicais: os sintetizadores. Sem entrar em muitos detalhes, somente diremos que atualmente muitos instrumentos comerciais estão dotados de uma interface normalizada para a conexão a qualquer computador. Citada interface não é outra coisa que um conjunto de "ports", tanto de entrada como de saída, que são conectados a outros idênticos do próprio computador.

Esquecíamos de dizer que os modernos sintetizadores estão providos de um potente sistema eletrônico digital com microprocessador, que controla os parâmetros de cada som, permite a polifonia, arpejos automáticos e uma grande gama de efeitos. No comércio existem acessórios, em forma de cartões para inserir em um ou vários encaixes do computador pessoal, que o transformam em um instrumento sofisticado: basta unir-lhe um teclado e poderemos "tocá-lo".

Com o mesmo princípio do cartão adicional existem interfaces de voz, baseados em chips VLSI muito aperfeiçoados, que traduzem um código binário, transmitido desde a CPU, em um fonema: oh, ah, b, ch, por exemplo. Uma série de ordens binárias proporciona assim uma sucessão de fonemas que permitem "ouvir" uma palavra ao unir-se (a fotografia 10 mostra o modelo deste periférico). Desta forma, o interface pode ler um texto toda vez que o imprimimos ou visualizamos. Lembra do filme "Jogos de Guerra" e a interface Votrax conectada ao IMSAI do protagonista? Os únicos defeitos (além do preço, claro) no estado atual da tecnologia, são a carência de inflexões na expressão e a falta de chips especializados em pronúncias diferentes: não existe outro remédio, se não habituar-se ao forte sotaque inglês.

Outros periféricos: B) Periféricos "De entrada"

Um periférico de entrada recolhe dados e/ou ordens do mundo externo e os transmite ao computador sob a forma de códigos binários. São periféricos de entrada, além dos teclados alfanuméricos, as tábuas gráficas, o joystick, o "mouse", os "paddles" e, para aplicações sofisticadas, todas as unidades de conversão analógica-digital tais como cartões de osciloscópio digital, analizadores de espectro, etc. Estes últimos produtos costumam estar diponíveis como acessórios inseríveis nos encaixes do cartão matriz do computador pessoal.

FLOPPY DISK: Disco flexível de material plástico, sobre o qual está depositado um extrato de óxido metálico magnetizável, contido em um estojo quadrado de proteção. Os primeiros exemplares foram produzidos pela IBM, com um diâmetro de 8 polegadas (8"). Quem deu o nome de "floppy" (flexível) ao primeiro disquete parece que foi um dos técnicos que teria como encargo a comprovação do novo produto. Ao tomá-lo em sua mão por um lado, este se curvou em sua totalidade e foi espontâneo o adjetivo "floppy" para o disco. Os diâmetros existentes, além dos antigos e volumosos de 8 polegadas, são 5 1/4, 3 e 3 1/2 polegadas. Os dois últimos estão contidos em um receptáculo mais rígido que o invólucro normal, portanto o disco em seu interior está mais protegido. O mercado mundial parece estar se orientando cada vez mais para a utilização destes novos discos flexíveis, denominados também "microfloppy".

CRT (TRC): Tubo de raios catódicos. Os termos derivados são: controlador CRT (ou TRC), terminal CRT, etc.

JOYSTICK: Se trata de um potenciômetro duplo situado em um suporte, giratório em duas direções perpendiculares entre si. Uma alavanca controla o mecanismo de rotação e permite acionar simultaneamente os dois potenciômetros, em proporções variáveis conforme a inclinação da própria alavanca. O joystick produz a saída das tensões, que podem enviar-se às entradas correspondentes do computador e que são, uma delas proporcional ao deslocamento da alavanca e na direção de "x", e a outra proporcional ao deslocamento da alavanca na direção "y". O joystick é idôneo para ajustar a posição de um cursor em um plano x-y, como é a tela de gráficos de um terminal de alta resolução.

MOUSE: Se trata de uma caixinha, com um ou mais pulsadores na parte superior, conectada mediante um cabo (a cauda) ao computador pessoal. Recebe este nome por seu aspecto, semelhante ao de um ratinho (lhe faltam as orelhas). Uma bola de borracha gira com componentes de rotação em x-y quando o usuário faz deslizar-se ao "mouse" sobre uma uperfície plana preparada. A bola atua sobre dois contatos giratórios perpendiculares entre si, que geram impulsos cuja freqüência é proporcional ao componente x-y do movimento do "mouse" . O computador pessoal, ao qual está conectado o "mouse" pode reconstruir assim o valor da coordenada so-

bre a que está o cursor, tal como acontece com um joystick.

PADDLE: Palheta. Acessório similar ao joystick, mas com um só potenciômetro, que gera uma tensão de referência única. Se utiliza muito nos videogames.

TÁBUA GRÁFICA (ou prancheta eletrônica): É uma prancheta plastificada sob a qual uma rede especial de sensores reconhece a presença de uma ponta que se move, desenhando, em uma folha de papel apoiada sobre a mesma tábua. Deste modo, se identifica a posição da ponta e se transforma em coordenadas x-y, que logo se transmitem ao computador em forma de dados binários. Assim, o computador pode armazenar por pontos um desenho enquanto que o usuário o está realizando com a tábua.

CAPÍTULO VIII

CONCLUSÕES

No momento presente, nos encontramos em uma situação extremamente confusa no que diz respeito ao conhecimento e emprego dos computadores, tanto pessoais como "domésticos". Existem aqueles que acreditam terem resolvido todos os seus problemas com a simples aquisição de um equipamento, talvez custoso, que permanecerá quase sempre desligado, ainda que, dando um imponente aspecto ao escritório.

Como em todas as coisas, é preciso ter (ou adquirir) conhecimentos e familiarizar-se pouco a pouco com o computador, pois se não for assim, corre-se o risco de dar-se conta de que não se sabe fazer nada ou quase nada com ele. Apesar disso, inclusive na atual confusão, o emprego destas máquinas maravilhosas em qualquer de suas formas e em qualquer ambiente, sempre fornece resultados positivos posto que a nova geração estará habituada a pensar e a trabalhar de um modo lógico, servindo-se do computador pessoal em casa, na escola ou no escritório.

Mas, para começar bem, é preciso refletir na hora de escolher o sistema e orientar-se até a aquisição de computadores e periféricos seguros e confiáveis. É indispensável evitar cair nas garras daqueles que especulam a ignorância (inicial) do comprador de seu primeiro computador e cuidar-se para não adquirir um equipamento que não tenha sentido nem justificativa em comparação com o uso que será dado ao mesmo. Em resumo, agenciar-se em um

bom sistema é sempre algo difícil. É preciso assessorar-se bem, sobretudo devido às inúmeras categorias e preços de cada um dos produtos existentes atualmente no mercado, incluindo os periféricos. Critérios mais detalhados que sirvam de guia na escolha de um computador pessoal serão objeto do próximo volume da BBI. No entanto, antes da aquisição, é indispensável conhecer como funcionam, de modo que se possa apreciar suas características mais importantes e, com freqüência, menos patentes. Este era nosso objetivo.

Ficaríamos satisfeitos se, depois do esforço que nós dispensamos para sua redação (e dispensamos para o leitor na sua compreensão), você tiver as idéias um pouco mais claras sobre o que há dentro e fora dessa incrível máquina que é o computador pessoal.

BIBLIOGRAFIA

1. Eletrônica digital - circuitos e tecnologia
   Garue
2. Computer aided structural design
   Clarke
3. Computer architecture
   Foster
4. From chips to systems - introd. to microprocessors
   Zaks
5. Microcomputer - based design
   Peatman
6. Microcomputer structures
   D'Angelo
7. Microprocessadores - conceitos básicos
   Osborne
8. Construa seu próprio microcomputador usando Z-80
   Ciarcia
9. Microcomputadores e microprocessadores
   Malvino
10. Circuitos digitais e microprocessadores
    Taub
11. Introdução aos microprocessadores
    Tokheim
12. 6502 Assembly Language Programming
    Leventhal
13. 6502 Programming & Hardware Manual
    Rockwell
14. Z80 Programming & Hardware Manual
    Zilog

NOTAS

Neste primeiro volume da Biblioteca Básica de Informática trata-se, fundamentalmente, de tudo que se relaciona com o hardware ou "parte física" de um computador, quer dizer o que podemos tocar. Assim, têrmos obscuros como CPU, microprocessador, periféricos, RAM, código ASCII, etc, e outros conceitos difíceis como a "arquitetura interna" de um computador ou comunicação homem-máquina são explicados com uma linguagem simples e desenfadonha que permite assimilá-los facilmente, inclusive por pessoas sem conhecimentos prévios de informática.

Por outro lado, como uma ajuda adicional ao leitor, ao longo dos textos, os conceitos mais importantes estão ressaltados em azul, o que facilita enormemente sua localização nas leituras posteriores desta obra, que certamente ocorrerão.

Cz$ 49,50