MICROPROCESSADOR

Z-80

HARDWARE

NEY ACYR R. DE OLIVEIRA

ANDRÉ GIL RUBENS

AeN Interface

MICROPROCESSADOR

Z-80

HARDWARE

NEY ACYR R. DE OLIVEIRA *ANDRÉ GIL RUBENS*

Interface

MICROPROCESSADOR Z-80
HARDWARE

AUTORES: **ANDRÉ GIL RUBENS**
NEY ACYR R. DE OLIVEIRA

1º edição – 1983

Composição:
Studio Moraes
256-0939/257-3478

Revisão Gramatical:
Jussara Sant'anna de Oliveira

Capa:
Nilton Luiz Côrtes

Diagramação e Arte Final:
João R. Reis Jr.

Editado pela
A e N – Consultoria Projetos e Publicações Ltda.
Rua da Conceição 105 Sala 1906
Rio de Janeiro - RJ CEP 20051
1983
Impresso no Brasil

"O traço mais novo e mais marcante do século XXI, para o futuro da educação, será a intelectualização do trabalho, que exige a aquisição de uma base mínima de conhecimentos científicos em dia. Deverão ser "instruídas" não só as "mãos" do trabalhador, mas também, e principalmente, a sua "cabeça". Só esse dado já vai influir um pouco no vasto conjunto de exigências futuras da coletividade, no domínio da educação".

Arthur V. Petrovski
O Correio da Unesco
OUT/NOV 1982

Aos meus pais
à Luzia e
a nossa filha Fabiana.

N. A. R. O

Aos meus pais
e à tia Delza.

A. G. R.

Aos amigos Naila e Cesar, diretores da revista INTERFACE, os nossos agradecimentos pelo incentivo e apoio durante a realização deste livro.

Os autores

PREFÁCIO

A primeira versão deste texto foi redigida para cursos de extensão lecionados por nós, em faculdades e empresas privadas e estatais. Com a criação da revista Interface foi escrita uma segunda versão, dirigida para um público mais abrangente e editada sob a forma de lições.

O presente texto é dirigido aos estudantes, profissionais e auto-didatas das áreas de eletrônica e informática.

A filosofia que seguimos no desenvolvimento deste trabalho foi a de permitir uma boa assimilação do assunto, tanto para os profissionais e estudantes que já conhecem algum outro microprocessador, como para os iniciantes nas áreas de eletrônica e informática. Para tanto, apresentamos nos dois capítulos iniciais inúmeros conceitos indispensáveis ao entendimento dos microprocessadores. Acreditamos que o leitor que possua conhecimentos básicos de eletrônica digital e de álgebra booleana, tenha mais facilidade para assimilar toda a plenitude deste trabalho.

O leitor encontrará no decorrer dos capítulos uma enorme preocupação com os aspectos didáticos dos diversos conceitos envolvidos. Isto porque, uma vez dominado o hardware de um microprocessador como o Z-80, torna-se bem mais fácil compreender e utilizar outros microprocessadores, pois os conceitos básicos de funcionamento permanecem inalterados.

Vejamos agora um resumo de cada um dos capítulos e apêndices deste livro.

O Capítulo 1 é eminentemente conceitual. Ele é muito importante para o leitor que está iniciando na área de informática, pois nele é feito um nivelamento dos conhecimentos básicos de computadores. Apresentamos um histórico das máquinas de computação, dando ênfase aos fatos que influenciaram nas características das modernas máquinas hoje existentes no mercado, em especial o microcomputador. Em seguida, apresentamos a organização básica de um computador digital e sua seqüência operacional.

O Capítulo 2 conceitua microprocessador e microcomputador. Apresentamos a arquitetura básica, memórias, periféricos e as principais aplicações dos microcomputadores. Finalizamos com um breve histórico dos microprocessadores, o que permite ao leitor situar-se em relação ao Z-80.

O Capítulo 3 é dedicado ao estudo da arquitetura específica do Z-80, mostrando a sua organização lógica e o seu mapa de registradores.

No Capítulo 4 descrevemos as características elétricas de todos os sinais do microprocessador Z-80 e a sua compatibilidade com as famílias TTL e CMOS.

No Capítulo 5 apresentamos os estados, os diversos ciclos de máquina e os diagramas de tempo de cada um destes ciclos do Z-80.

Reservamos o Capítulo 6 exclusivamente para o estudo das interrupções no Z-80. Primeiramente, fazemos um estudo conceitual do assunto para, em seguida, apresentarmos separadamente os modos de interrupção do Z-80. Dedicamos um capítulo a este assunto dada a sua importância nos sistemas em tempo real.

O Capítulo 7 apresenta exemplos de projetos didáticos de módulos de memória e de interfaces de entrada e saída.

No Capítulo 8 procuramos reunir no projeto de um microcomputador todos os conhecimentos adquiridos nos capítulos anteriores. Aproveitamos, também, para descrever a interface paralela PIO-Z80, que faz parte do referido projeto.

No Apêndice A fornecemos as características elétricas detalhadas do microprocessador Z-80.

O Apêndice B apresenta o repertório de instruções do Z-80.

O Apêndice C é uma tabela de códigos ASCII, que é o mais utilizado na comunicação entre microcomputadores e periféricos.

No final de cada capítulo apresentamos diversos exercícios de fixação. Estes exercícios fazem parte integrante do aprendizado do texto, pois ao tentar resolvê-los, o leitor irá consolidar e ampliar os conhecimentos adquiridos. Temos um total de cento e quarenta exercícios, incluindo desde perguntas objetivas sobre tópicos do texto até projetos de interfaces e módulos de memória.

Procuramos escrever este livro de uma maneira seqüencial e organizada. O entendimento de um capítulo exige o conhecimento dos anteriores. Contudo, os leitores que já conhecem outro microprocessador podem iniciar a leitura a partir do capítulo 3.

A necessidade de se escrever em português, sobre um assunto cuja origem é um inglês técnico específico, foi para nós um desafio. É fato sabido que os profissionais evitam traduzir os nomes dos elementos e conceitos introduzidos em inglês. Isto ocorre, principalmente, devido ao receio de não se fazerem entender, caso se aventurem a traduzí-los. Assim, os termos técnicos cuja tradução não é utilizada no dia a dia dos profissionais foram deixados na versão original. Por outro lado, aqueles cuja tradução já começa a se popularizar foram traduzidos, e a versão original encontra-se entre aspas e parênteses.

Com relação às figuras, alertamos que usamos o zero com barra (ϕ) quando este representa níveis lógicos. No restante do texto, usamos o zero na sua representação tradicional (0).

Gostaríamos de agradecer a todos que participaram e colaboraram nesta obra e, em particular, ao Engº Luiz Antonio Maron Fonseca pelas sugestões apresentadas.

Os autores
Rio de Janeiro, setembro de 1983

ÍNDICE

PARTE I – CONCEITOS BÁSICOS

CAPÍTULO 1 – AS MÁQUINAS DE COMPUTAÇÃO

CAPÍTULO 2 – MICROPROCESSADORES E MICROCOMPUTADORES

PARTE II – O ESTUDO DO Z-80

CAPÍTULO 3 – A ARQUITETURA

CAPÍTULO 4 – IDENTIFICAÇÃO DOS PINOS

CAPÍTULO 5 – O FUNCIONAMENTO

CAPÍTULO 8 – PROJETO DE UM MICROCOMPUTADOR

PARTE IV – APÊNDICES

PARTE I
CONCEITOS BÁSICOS

1 AS MÁQUINAS DE COMPUTAÇÃO

1.1 – APRESENTAÇÃO

Neste capítulo, faremos uma revisão dos princípios funcionais dos computadores, os quais utilizaremos como base para a conceituação dos microprocessadores e microcomputadores.

Iniciaremos apresentando a evolução das máquinas de computação, desde o ábaco até os computadores eletrônicos, dando ênfase aos marcos que estabeleceram os conceitos fundamentais ao desenvolvimento da ciência dos computadores.

Estudaremos a arquitetura básica e a seqüência operacional dos computadores digitais.

1.2 – EVOLUÇÃO

1.2.1 – As Máquinas Mecânicas

Grande parte das invenções foram motivadas pelo desejo do homem de proporcionar meios para automatizar trabalhos em que se observam procedimentos repetitivos. O ábaco que data de 450 AC, foi um dos primeiros instrumentos inventados pelo homem para auxiliá-lo nas operações aritméticas.

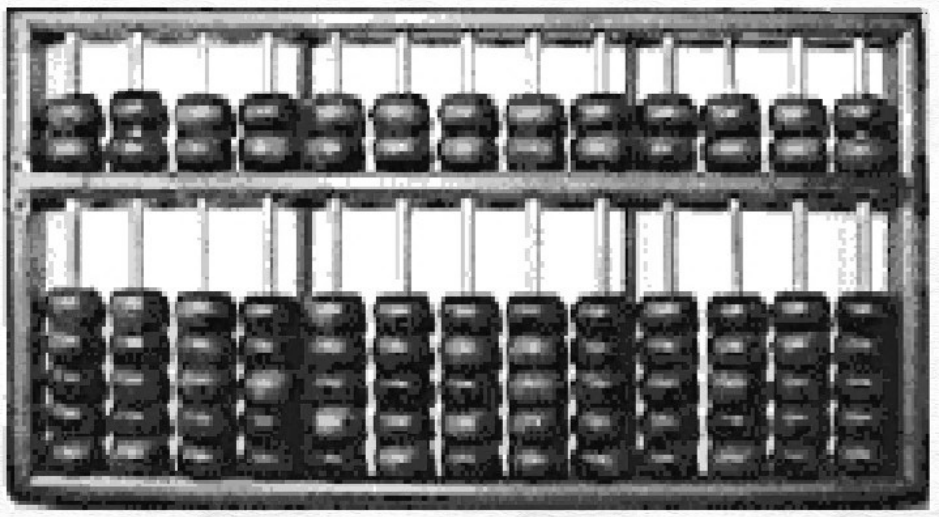


Fig. 1.1 – O Ábaco

Em 1642, Pascal desenvolveu a primeira calculadora mecânica que realizava somas e subtrações na base numérica decimal, que é a que utilizamos normalmente.

Leibniz, em 1671, projetou uma calculadora mecânica mais versátil, que tinha a característica importante de operar na base *numérica binária;* e incluía, ainda, a capacidade de fazer multiplicações e divisões.

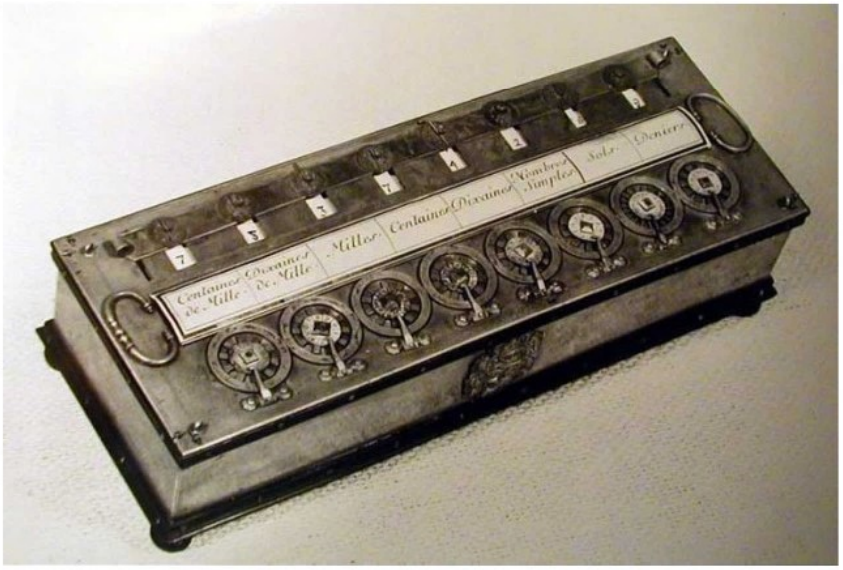


Fig. 1.2 – A Calculadora de Pascal

Charles Babbage, matemático inglês, foi incumbido de desenvolver métodos para computar tabelas de logarítmos. Observando que tal tabela seria muito trabalhosa com os recursos disponíveis, desenvolveu uma estrutura de máquina que executava processamentos complexos e emitia resultados sem a intervenção do homem. Aplicando tal idéia, em 1822, concebeu uma máquina que computava tabelas matemáticas e fornecia resultados com até cinco dígitos significativos.

Fig. 1.3 – A Máquina de Babbage de 1822

Com o objetivo de aperfeiçoar esta máquina, Babbage propôs a construção de um dispositivo que denominou de *"Máquina Analítica"*. Esta máquina é considerada a predecessora do computador, pois sua *arquitetura*, isto é, sua organização lógica, contém basicamente as mesmas unidades funcionais que são encontradas nos computadores mais modernos.

Babbage estabeleceu os seguintes princípios de funcionamento para sua máquina analítica:

1 – Ser equipada de uma *unidade de cálculo* capaz de realizar *operações lógicas* e *aritméticas;*

2 – Ser dotada de um meio de *entrada,* pelo qual um número ilimitado de operandos ou *instruções* possa ser inserido na máquina;

3 – Possuir uma *memória,* da qual operandos possam ser obtidos, e na qual resultados possam ser armazenados em qualquer ordem desejada;

4 – Ser dotada de um meio de *saída,* pelo qual um número ilimitado de resultados possa ser emitido para o usuário;

5 – Dispor da *capacidade de decisão,* através da qual possam ser escolhidos cursos de ação alternativos, com base nos resultados computados.

As informações operadas pela unidade de cálculo são chamadas de *operandos.* As instruções são comandos elementares capazes de serem interpretados pela máquina, e que se propõem a automatizar a intervenção manual do homem. Uma seqüência predeterminada de instruções forma um *programa,* que define as atividades da máquina para a consecução das tarefas desejadas pelo usuário.

Babbage iniciou os trabalhos de concepção da máquina analítica em 1830, e passou o restante da sua vida num esforço infrutífero para construí-la. Na verdade, suas idéias estavam cem anos avançadas em relação ao estado da técnica. A tecnologia da época era inadequada para a elaboração de um engenho com as exigências de sua especificação, e é até duvidoso que pudesse ser implementada com a tecnologia mecânica dos dias de hoje. De fato, os sonhos de Babbage tiveram que esperar o desenvolvimento da eletrônica.

1.2.2 – As Máquinas Eletrônicas

Em 1937, Howard Aiken, da Universidade de Harvard propôs a concepção de uma calculadora automática baseada na combinação dos princípios de Babbage e na tecnologia de calculadoras eletromecânicas da IBM. O projeto teve início em 1937, levou o nome de Mark I e foi terminado em 1944, ano que é considerado por muitos como o início da era dos computadores.

O Mark I era uma máquina eletromecânica construída essencialmente com *relés,* fator limitante de sua velocidade. Em 1945, a Universidade de Pensilvânia iniciou o desenvolvimento do ENIAC, à base de válvulas, resultando no primeiro *computador digital eletrônico.*

Fig. 1.4 – O Mark I

Nessas calculadoras automáticas as instruções eram programadas através de *lógica cabeada,* isto é, pela conexão de fios em um painel de controle, ou a partir da leitura de cartões perfurados contendo as instruções. Esta leitura era realizada à medida que o programa ia sendo executado.

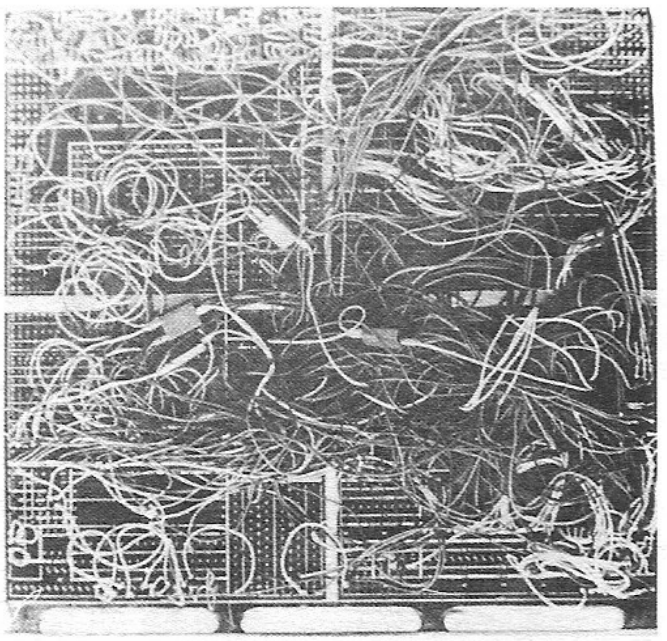


Fig. 1.5 – A Lógica Cabeada

Esses métodos limitavam a velocidade de processamento, porque o início de cada programa tinha que esperar a intervenção do operador da máquina, a reprogramação das conexões dos fios na lógica cabeada ou ainda, a introdução de novos cartões.

Por outro lado, uma vez que estas máquinas processavam operandos, por que não poderiam processar suas próprias instruções? Desta forma, auxiliariam o homem na reprogramação das instruções.

Von Neumann, em 1947, enunciou o conceito de *programa armazenado*, que resultou de uma conseqüência natural das dificuldades encontradas para se efetuar modificações nos programas das primeiras calculadoras automáticas. O enunciado do seu princípio é o seguinte:

> Operandos e instruções devem ser armazenados da mesma forma na memória, e serem igualmente acessíveis, de tal forma que as instruções possam ser tratadas como operandos, e assim, serem facilmente modificadas pela máquina.

Neste caso, a memória é usada para armazenar instruções e operandos indistintamente, sem diferença na forma. Para tal, a máquina de programa armazenado tem que ser capaz de buscar as instruções na memória, discerní-las de operandos, interpretá-las e executá-las. Instruções e operandos armazenados na memória são genericamente chamados de *dados*.

O conceito de programa armazenado resultou em um grande impulso no desenvolvimento de grandes memórias, pois agora elas têm o propósito de armazenar operandos e instruções; e deu origem às expressões *HARDWARE* e *SOFTWARE*. Hardware designa os componentes físicos do computador e software compreende as instruções armazenadas na memória, as quais definem as tarefas da máquina.

O conceito de Von Neumann e os cinco critérios que definem a máquina de Babbage formam os seis princípios fundamentais dos computadores de nossos dias. Mais adiante, veremos como estes princípios conduziram a organização básica dos modernos sistemas de computação.

O ano de 1948 marcou a realização da primeira máquina de programa armazenado, o *EDVAC*, dando início a uma indústria caracterizada pela nova filosofia de implementação de recursos de sistema através de uma clara distinção entre hardware e software. Esse mesmo ano de 1948 marcou o desenvolvimento do primeiro *transistor* nos laboratórios da Bell. O transistor é um dispositivo semicondutor que veio substituir a válvula com baixo custo e menores dimensões. Ele tornou economicamente viável a construção de *computadores digitais de grande porte*, sendo que o primeiro computador comercial transistorizado foi lançado no mercado dez anos mais tarde, em 1958.

Neste mesmo ano de 1958, ocorreu o desenvolvimento do primeiro *circuito integrado,* que consiste de um cristal monolítico de silício denominado *"chip"*, encapsulado num envoltório de cerâmica ou plástico, contendo elementos eletrônicos ativos (transistores) e passivos (resistores) ligados entre si.

A importância do impacto da tecnologia dos semicondutores no desenvolvimento de máquinas de computação acentuou-se a partir do advento do transistor, a tal ponto, que os avanços tecnológicos das referidas máquinas não puderam ser mais dissociados das pesquisas no campo da física dos semicondutores.

No ano de 1964, a Digital Equipment Corp. colocou no mercado o primeiro *minicomputador.* Tal classe de máquina tem essencialmente um baixo custo e um tamanho reduzido em relação aos computadores de grande porte. Estas características foram conseguidas pela redução da capacidade da memória e do repertório de instruções e, principalmente, pela utilização da tecnologia de circuitos integrados. A capacidade de realizar cálculos precisos e tomar decisões em alta velocidade, tornou viável a utilização do minicomputador como controlador de sistemas em fábricas, refinarias e empresas comerciais de menor porte.

Os progressivos aumentos na *densidade de integração* de circuitos semicondutores precipitou, em 1964, o desenvolvimento da tecnologia de *integração em grande escala,* denominada *"LSI"*, que permite a construção de "chips" contendo mais de 1000 componentes eletrônicos.

A revolução provocada na eletrônica digital pelos constantes progressos alcançados nas técnicas de integração, combinada com o conceito de programa armazenado, culminou com o aparecimento em 1970 da primeira memória semicondutora LSI. Sucedendo-se, em 1971, o lançamento do primeiro *microprocessador.*

O microprocessador é um circuito integrado LSI que reúne a unidade de cálculos lógicos e aritméticos, os elementos de controle e seqüência, bem como unidades auxiliares de processamento. Ele substitui todos os circuitos contidos em múltiplas placas de circuito impresso instaladas em diversos bastidores, que eram necessários para constituir uma unidade de processamento de um minicomputador.

Para conhecermos melhor os microprocessadores, estudaremos em seguida alguns conceitos fundamentais de computadores. O que podemos adiantar, ainda, é que se foi grande o impacto tecnológico, mais importante foi a redução que a introdução dos microprocessadores causou na composição de preços dos sistemas de controle de processo e de processamento de dados. A primeira, e talvez mais notável conseqüência deste fato, foi o ingresso dos microprocessadores na área dos equipamentos eletrônicos profissionais de me-

dição e controle, dos sistemas de comutação eletrônica, das máquinas industriais e gráficas, das balanças e caixas registradoras de supermercados, além de muitas outras aplicações que se multiplicam ano a ano.

As reduções subseqüentes nos preços dos microprocessadores fizeram com que o seu uso se expandisse além do campo dos dispositivos de medição e controle, surgindo máquinas que colocaram a informática ao alcance das empresas de pequeno porte, e acabaram, finalmente, por ingressar nos lares como valiosa ferramenta de estudo, de controle orçamentário doméstico e até de lazer e entretenimento. Atualmente, existem microprocessadores tão baratos, que até mesmo sua utilização em jogos e brinquedos tornou-se viável e competitiva.

1.3 – O COMPUTADOR DIGITAL

1.3.1 – A Organização Básica

Analisaremos a organização básica de um computador digital, acrescentando assim alguns subsídios indispensáveis a uma melhor compreensão do assunto.

Qualquer sistema de computação que satisfaça aos critérios da máquina analítica de Babbage e ao conceito de programa armazenado de Von Neumann, pode ter a sua organização lógica básica conforme a Figura 1.6.

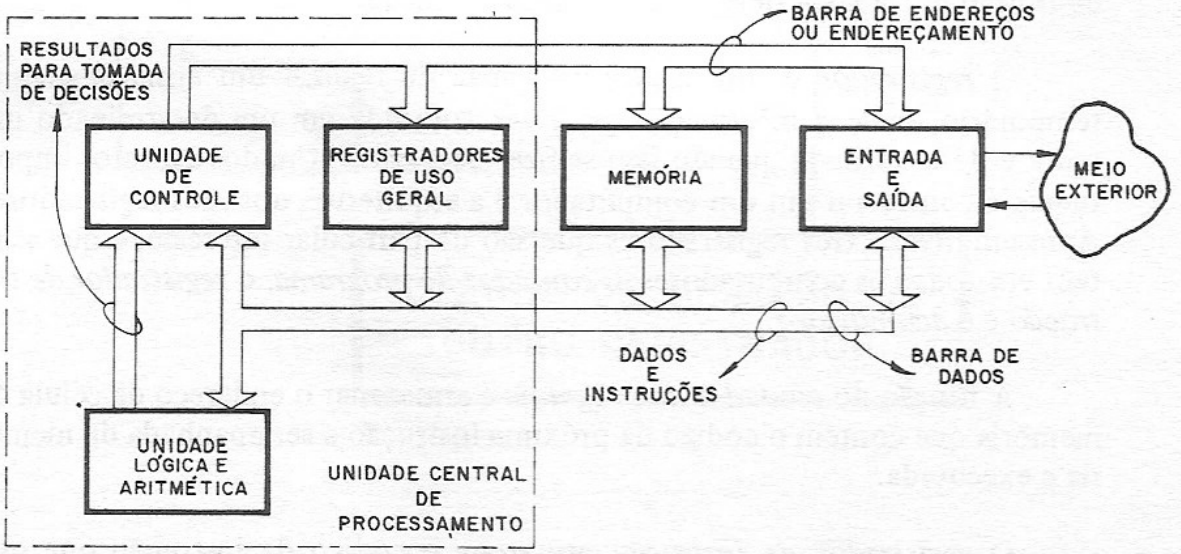


Fig. 1.6 – Organização Básica de um Computador

As cinco funções associadas aos módulos da figura 1.6 podem ser identificadas em qualquer computador digital, independente da natureza tecnológica dos componentes eletrônicos empregados, da existência entre os módulos de alguma superposição de funções e, mesmo que os componentes sejam compartilhados.

A *unidade de controle* busca as instruções na memória, as interpreta e gera os sinais de controle e sincronização apropriados, que respondem pelo automatismo do processamento. Os referidos sinais exercem as funções básicas de coordenar os seguintes eventos: as transferências de informações entre os módulos do computador, as operações lógicas e aritméticas, e a opção entre cursos alternativos de ação, com base nos resultados das operações lógicas e artiméticas.

A *memória* é um conjunto de células de armazenamento de operandos e instruções. A cada célula está associado um endereço, segundo uma relação biunívoca. Através do endereço, determinada célula pode ser acessada pela unidade de controle. Para a memória, é indiferente se o conteúdo de cada célula encerra um operando ou uma instrução, pois esta identificação é feita pela unidade de controle.

A função básica do *módulo de entrada e saída* é oferecer os recursos de comunicação do computador com o *meio exterior,* através das *unidades periféricas,* tais como: uma impressora, um terminal, outra memória, ou até mesmo um outro computador.

A *unidade lógica e aritmética* (ULA) efetua todas as operações lógicas e aritméticas, tendo como operando dados provenientes da memória, do módulo de entrada e saída ou dos registradores. A unidade de controle designa o tipo de função que a ULA deve realizar a cada operação.

O *registrador* é um dispositivo capaz de realizar um armazenamento temporário, onde a informação pode ser colocada em um determinado instante e de lá retirada quando isso se fizer necessário. Um dos aspectos importantes a considerar em um computador é a arquitetura dos seus registradores. Apresentaremos três registradores que são de particular interesse, e que existem em todos os computadores: o *contador de programa,* o *registrador de instrução* e o *acumulador.*

A função do *contador de programa* é armazenar o endereço da célula de memória que contém o código da próxima instrução a ser apanhada da memória e executada.

O *registrador de instrução* armazena o código da instrução que está em processo de execução. Tal código gera sinais de sincronização e controle, precisamente distribuídos no tempo e no hardware, que efetuam as ações necessárias para a execução da instrução.

As operações lógicas e aritméticas só podem ser realizadas usando-se como operandos os conteúdos do *acumulador*, de uma célula de memória ou de um outro registrador. O resultado da operação é sempre depositado no acumulador conforme mostra a figura 1.7, daí este registrador ser chamado de acumulador.

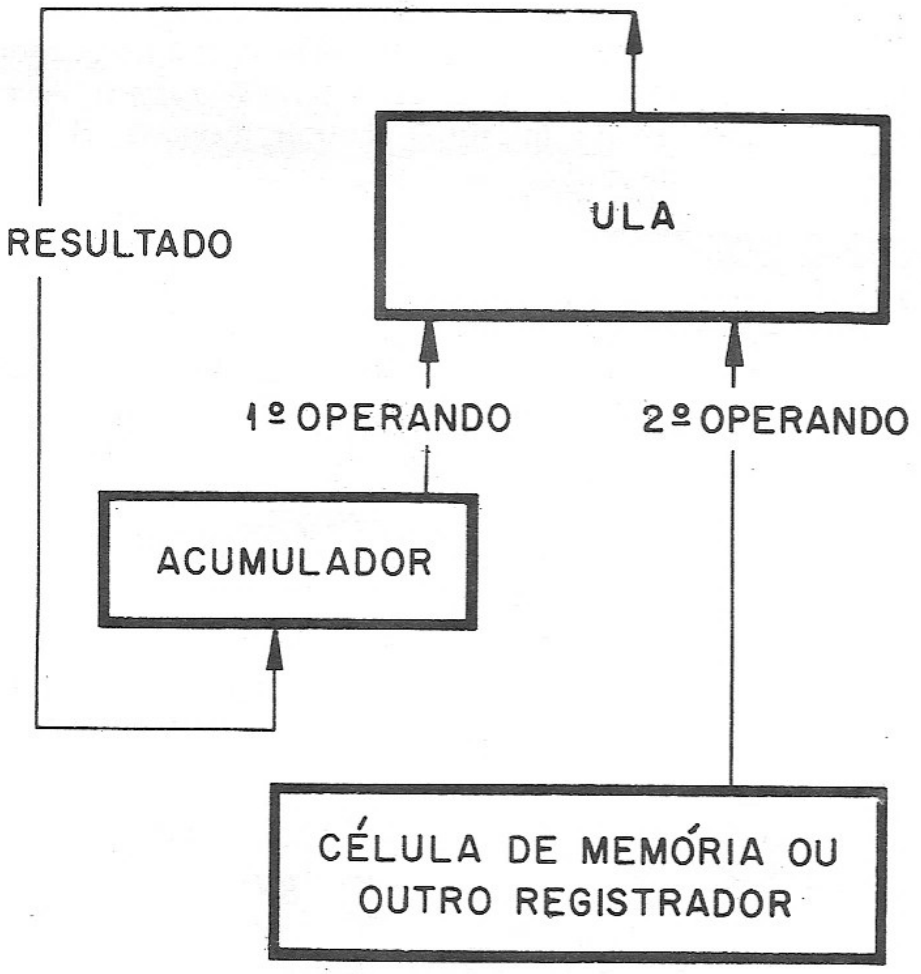


Fig. 1.7 – Operações Lógicas e Aritméticas

Além dos três registradores apresentados, via de regra, um computador inclui *registradores de uso geral* que podem ser usados como dispositivos de armazenamento de dados e de endereços de células de memória, e servir, também, de operandos nas operações lógicas e aritméticas.

A arquitetura básica da figura 1.6 satisfaz ao critério de Von Neumann, uma vez que a ULA pode processar qualquer informação originada da memória; indistintamente instruções e operandos. A combinação da ULA, da unidade de controle e dos registradores é denominada *unidade central de processamento* (UCP).

As informações procedentes da UCP ou a ela destinadas, trafegam através de grupos de vias de transporte de sinais elétricos denominados *barras ("BUS")*, que permitem a transferência de informações.

A organização básica da figura 1.6 tem duas barras comuns. A *barra de endereços* permite que a UCP selecione qualquer célula de memória ou dispositivos de entrada e saída. Esta barra é *unidirecional,* uma vez que somente a UCP escolhe a célula de memória desejada. A *barra de dados* estabelece a comunicação *bidirecional* entre a unidade de controle, memória, ULA e entrada e saída; e por ela trafegam, basicamente, operandos e instruções.

Um computador digital processa sinais que assumem somente dois estados lógicos, aos quais correspondem os algarismos binários '0' ou '1', denominados *bits,* contração de *"binary digit"*. O bit constitui a unidade elementar de informação, admitindo múltiplos usuais tais como o KILOBIT (1024 bits) e o MEGABIT (1.048.576 bits).

A utilização do elemento binário deve-se ao fato de ser mais fácil implementar um circuito eletrônico digital que reconheça dois níveis lógicos, do que um circuito que admite dez níveis, o que se aplicaria a um sistema decimal.

Os dados manipulados na operação de uma máquina de computação são formados por blocos de bits de extensão constante, baseados nas seguintes unidades:

> O *byte* que é composto de 8 bits e a *palavra* que é formada de um nº inteiro de bytes que varia de acordo com o porte do processador.

De acordo com esta quantização, os elementos constitutivos dos circuitos de processamento, tais como registradores, dispositivos de cálculo e células de memória, são feitos para comportarem, via de regra, números inteiros de bytes.

1.3.2. – Seqüência Operacional

A organização funcional de um computador pode ser melhor entendida por uma apreciação de sua dinâmica de processamento, isto é, pelo modo como se sucedem as ações principais que determinam a execução da seqüência de instruções de um programa.

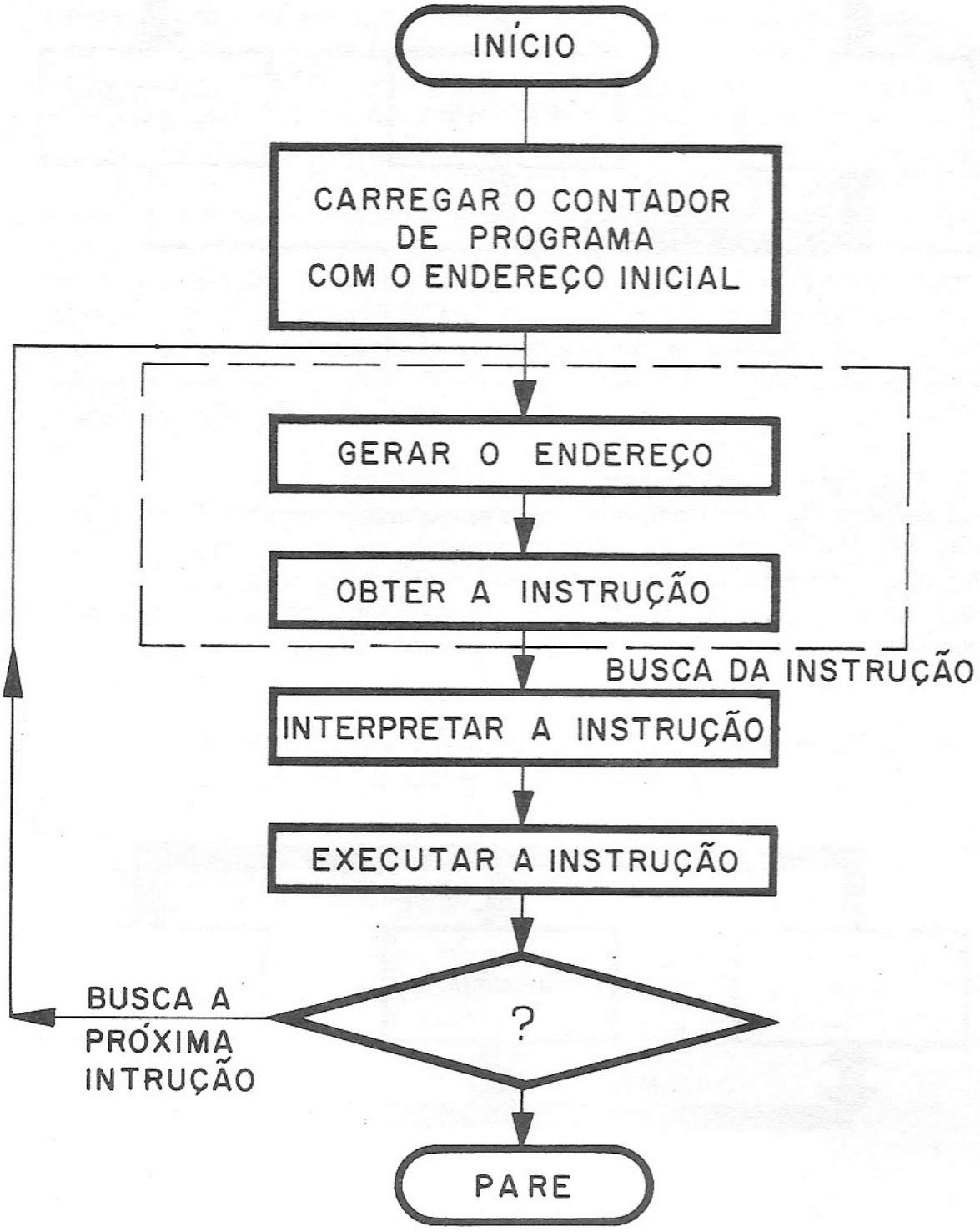


Fig. 1.8 – Transição de Estados da UCP

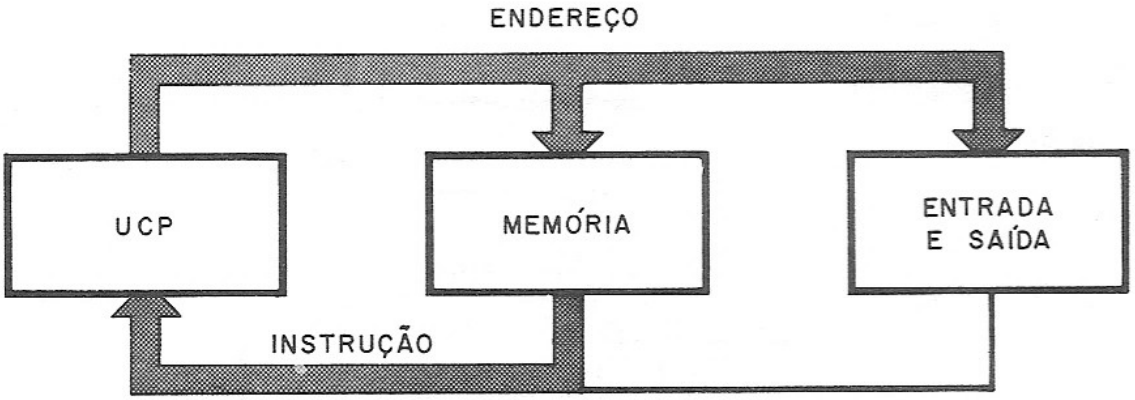


Fig. 1.9 – Busca de Instrução

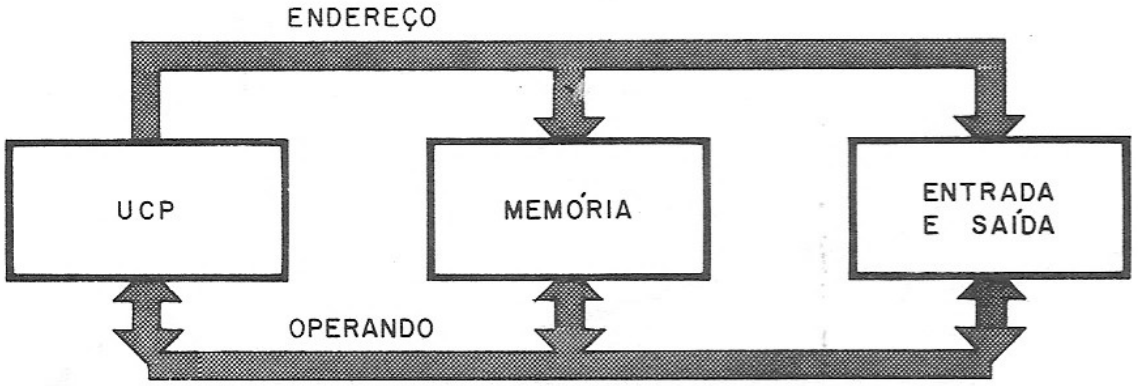


Fig. 1.10 – Busca do Operando

A figura 1.8 mostra um fluxograma que elucida a seqüência de transições dos principais estados durante o processamento de um programa. As figuras 1.9 e 1.10 mostram o tráfego das informações através das barras de endereços e dados, durante a busca da instrução e do operando, respectivamente.

As instruções do programa se acham armazenadas na memória, que se comunica com o microprocessador através das barras de dados e endereços. Antes do início da busca da primeira instrução na memória, o contador de programa é carregado com o *endereço inicial do programa,* isto é, o endereço da célula de memória que encerra o código da primeira instrução do programa. No instante adequado, determinado pelo cadenciamento dos sinais de controle e sincronização da máquina, este endereço é colocado na barra de endereços, indo ativar a célula de memória que contém a instrução a ser apanhada. Em seguida, o conteúdo da célula endereçada passa para a barra de dados, sendo através desta transferido para o registrador de instrução. Esta operação denomina-se *busca da instrução* na memória, e ao completá-la diz-se que a instrução foi apanhada. O contador de programa é então automaticamente atualizado, passando a conter o endereço da próxima instrução do programa a ser apanhada.

Do registrador de instrução, o código representativo da instrução passa ao processo de *interpretação,* que consiste em se fazer com que os níveis lógicos, associados aos bits armazenados no registrador de instrução, sejam usados para ativar sinais lógicos de sincronização e controle, necessários para a execução da instrução. Durante a execução de uma instrução pode-se fazer necessário outros acessos à memória ou referências à entrada e saída, para transferências de operandos.

A sistemática de busca, interpretação e execução de instruções prossegue indefinidamente, até que um comando "PARE" interrompa o programa.

1.4 – EXERCÍCIOS DE FIXAÇÃO

1.4.1 – Quais os princípios de funcionamento da máquina analítica de Babbage?

1.4.2 – Como era realizada a programação no MARK 1 e no ENIAC?

1.4.3 – Enunciar o conceito de programa armazenado de Von Neumann.

1.4.4 – O que aconteceu com as máquinas de computação após o conceito de Von Neumann?

1.4.5 – Descrever as funções da unidade de controle da figura 1.6.

1.4.6 – Qual a função do contador de programa?

1.4.7 – Conceituar "byte" e "palavra".

1.4.8 – Descreva a busca da instrução na memória.

1.4.9 – Como é realizada a execução da instrução?

1.4.10 – O que é microprocessador?

1.4.11 – O que ocorreu na indústria eletrônica após o aparecimento do microprocessador?

2 MICROPROCESSADORES E MICROCOMPUTADORES

2.1 – APRESENTAÇÃO

Neste capítulo, apresentaremos o conceito de microprocessador e de microcomputador. Estudaremos a arquitetura e as aplicações básicas dos microcomputadores e conheceremos a evolução dos microprocessadores desde o 4004 até o Z-80.

2.2 – OS MICROPROCESSADORES

2. 2.1 – Conceituação

Agora que conhecemos a organização básica de uma máquina de computação temos os subsídios necessários para compreender os *microprocessadores.* Estes se constituem em verdadeiras unidades centrais de processamento concentradas em uma pastilha monolítica LSI, substituindo conjuntos numerosos de componentes discretos ou integrados, cuja presença era imprescindível antes do seu aparecimento.

O *microcomputador* é um sistema de computação constituído de memória, entrada e saída, e tendo como unidade central de processamento um *microprocessador.*

Alguns fabricantes desenvolveram pastilhas LSI que reúnem, além da unidade central de processamento, uma quantidade limitada de memória e de linhas de entrada e saída. São os chamados *microcomputadores contidos em uma única pastilha ("Single Chip Microcomputer").*

A capacidade de realização de cálculos e tomada de decisões dos microcomputadores os tornam excelentes controladores de uso geral, de baixo

custo e tamanho reduzido, que podem realizar qualquer função de processamento de dados e controle, desde que tenham memória suficiente para armazenar todos os programas e que a execução dos mesmos seja veloz o suficiente para satisfazer as especificações dos seus usuários.

2.2.2 – Parâmetros Principais

Torna-se importante conhecermos os principais parâmetros que definem a capacidade de realização de tarefas de um microprocessador, que são os seguintes:

1 – A freqüência do relógio de sincronização.
2 – O número de bits nas barras de dados e de endereços.

A velocidade de processamento é o tempo que o microprocessador leva para executar as suas tarefas. Os microprocessadores são dispositivos sincronizados por um relógio, que é constituído, na maioria das vezes, por um oscilador a cristal com saída(s) pulsada(s), cujo período determina a unidade de tempo fundamental para a realização das ações que ocorrem no processamento. Por exemplo, o tempo de execução de uma instrução é um número inteiro de períodos da referida unidade de tempo. Logo, a freqüência fundamental máxima permitida pelo fabricante, associada ao número de períodos necessários à execução das instruções, é um dos fatores que determinam a velocidade de processamento dos programas.

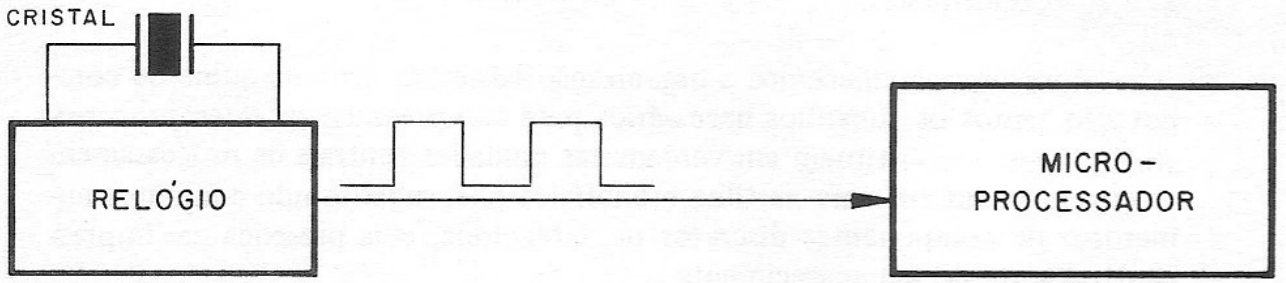


Fig. 2.1 – Microprocessador Sincronizado pelo Relógio

Outro fator determinante da velocidade de processamento é o número de bits da barra de dados. Cada acesso realizado à memória transfere, de uma só vez, uma informação codificada através dos bits desta barra. Logo, quanto maior o número de bits da barra de dados, maior será a quantidade de informação transferida de uma só vez.

O número de bits da barra de endereços determina o número máximo de células de memória e dispositivos de entrada e saída que o microprocessador pode acessar. A limitação do número de células, na verdade, tem implica-

ção direta na dimensão máxima possível dos programas. Para uma barra de endereços de *n* bits podemos ter até 2EXP(n) células de memória. Por exemplo, para uma barra de 16 bits teremos 2EXP(16), ou seja, 65536 células de memória (64K).

2.3. – OS MICROCOMPUTADORES

2.3.1 – Arquitetura Básica

Os microcomputadores, que são computadores cuja unidade central de processamento é implementada por um microprocessador, possuem uma arquitetura composta das mesmas unidades apresentadas no capítulo 1.

Portanto, antes de nos aprofundarmos no estudo do hardware do Z-80, é importante conhecermos com mais detalhes as partes que compõem os microcomputadores.

Na figura 2.2 temos a arquitetura básica de um microcomputador. Na verdade, esta é uma arquitetura mínima que, via de regra, pode ser identificada na maioria dos microcomputadores. No entanto, existem microcomputadores que possuem mais de uma UCP. Neste caso são utilizados mais de um microprocessador e a arquitetura é bem mais complexa.

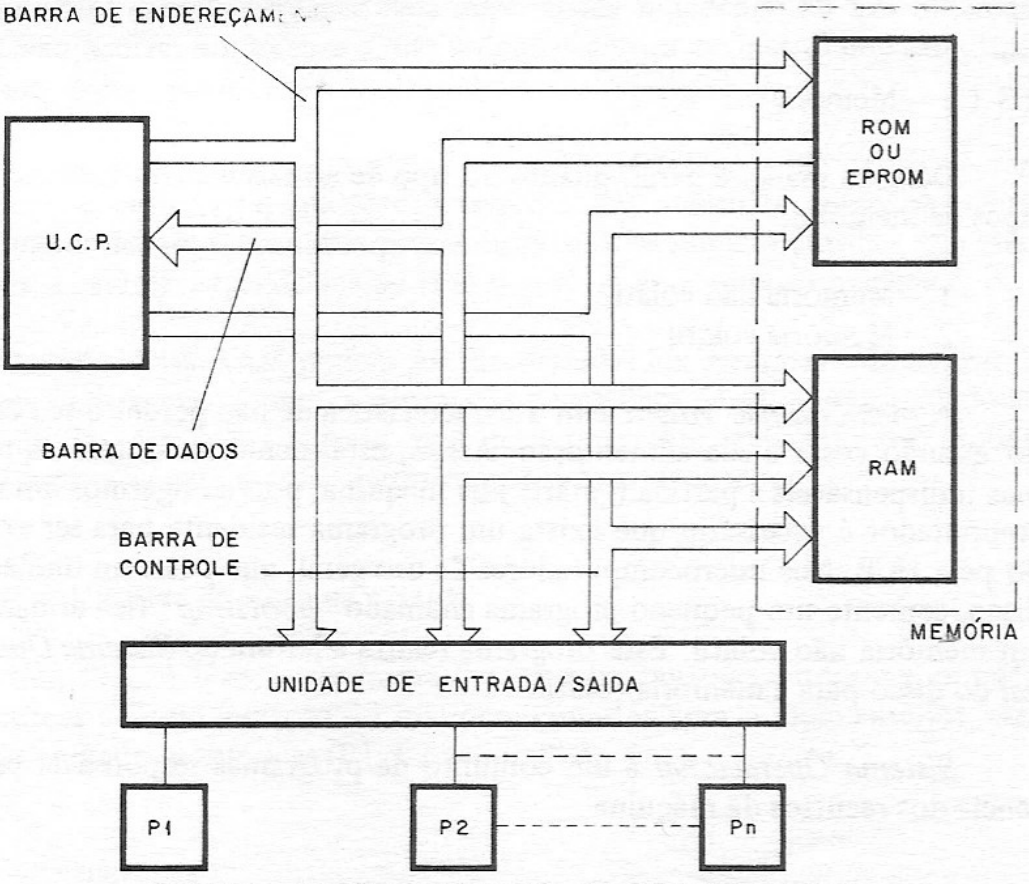


Fig. 2.2 – Arquitetura Básica de um Microcomputador

Vamos agora identificar e conhecer cada um dos componentes desta arquitetura.

2.3.1.1 – Unidade Central de Processamento (UCP)

A UPC possui além do microprocessador, circuitos lógicos adicionais que permitem a sua comunicação com a memória e com a unidade de entrada e saída, através das *barras de endereço, de dados e de controle.* Na quase totalidade dos casos, a UCP é implementada com um microprocessador comercial.

As razões da grande utilização do Z-80 devem-se às vantagens do seu hardware e do seu conjunto de instruções quando comparado com outros microprocessadores de 8 bits.

Notamos que mesmo com a existência de microprocessadores de 16 bits, os microprocessadores de 8 bits devem, por um bom tempo, continuar a dominar o mercado. Uma evidência desta tendência, é que hoje existem versões do Z-80 de 2,5 MHz, de 4,0 MHz, de 6,0 MHz e recentemente, a ZILOG lançou a versão de 8,0 MHz, designada Z-80H. Com isto, temos uma maior velocidade de processamento, sem que seja necessário investir em infra-estrutura para outros microprocessadores mais complexos.

2.3.1.2 – Memória

De uma maneira geral, quanto ao tipo de armazenamento, existem dois tipos de memória:

1 – Memória não volátil
2 – Memória volátil

A *memória não volátil* tem a característica de não perder o seu conteúdo quando cessa a sua alimentação. Assim, esta memória contém os programas indispensáveis à partida ("start") da máquina, pois ao ligarmos um microcomputador é necessário que exista um programa residente para ser executado pela UCP. Nos microcomputadores de uso geral, que possuem unidades de disco, somente um pequeno programa chamado *"bootstrap"* fica armazenado em memória não volátil. Este programa realiza a leitura do *Sistema Operacional* do disco para a memória volátil.

Sistema Operacional é um conjunto de programas responsável pela gerência dos recursos da máquina.

Atualmente as memórias não voláteis mais usadas são:

ROM – Read Only Memory
EPROM – Erasable Programable Read Only Memory

A *ROM* tem a característica de ser gravada em fábrica, sendo utilizada somente como memória de leitura. Portanto, sua utilização é economicamente viável em linhas de produção de grande escala.

A *EPROM,* que é a mais utilizada no mercado nacional, é fornecida pelo fabricante totalmente apagada (todos os bits iguais a "1"), sendo programada pelo usuário. É apagável com radiação ultravioleta, que atinge as junções através de uma janela de vidro transparente, existente no chip. Portanto, com a capacidade de ser apagável e reprogramável, torna-se uma memória de leitura bem versátil e muito utilizada.

A *memória volátil,* por possuir a característica de perder a informação quando cessa a alimentação, é usada para armazenar essencialmente *dados.* Atualmente a memória semicondutora volátil existente no mercado é chamada de *RAM* – Random Access Memory.

Uma memória é dita de *acesso randômico* quando o tempo necessário para se obter uma informação nela armazenada independe do seu endereço. O que não ocorre, por exemplo, nas unidades de fita magnética que são dispositivos de *acesso seqüencial.*

Na verdade, as memórias *ROM* e *EPROM* também são de acesso randômico. No entanto, atualmente o termo *RAM identifica memória volátil,* pois quando foram lançadas as primeiras memórias semicondutoras, que eram voláteis, a característica de acesso randômico era uma grande novidade.

As memórias RAM podem ser classificadas em *estáticas* e *dinâmicas.* A diferença entre elas é que as dinâmicas necessitam de uma lógica adicional, chamada de *circuito de refresh,* que periodicamente realiza acessos seqüenciais à memória, para manter a informação nela armazenada.

2.3.1.3 – Unidade de Entrada e Saída

Sua função básica, conforme foi visto no capítulo anterior, é oferecer os recursos de comunicação do microcomputador com o meio exterior, através das unidades periféricas (P1, P2, - - - -, PN).

2.3.2 – Aplicações

Atualmente, encontramos os microcomputadores disseminados nas mais

diversas áreas, que podem ser genericamente classificadas em:

1 – Processamento Comercial de Dados.
2 – Controle de Processos.

2.3.2.1 – Processamento Comercial de Dados

Na área de *processamento comercial de dados,* a configuração mínima dos microcomputadores é apresentada na figura 2.3.

Esta classe de microcomputadores possui, na maioria dos casos, 64K bytes de memória RAM e utiliza como *memória de massa,* unidades de disco flexível ("Floppy Disk") de 5 1/4" ou 8". Por exemplo, se fosse usado disco de 5 1/4" com gravação em densidade dupla e face simples, teríamos aproximadamente 140K bytes disponíveis por disco.

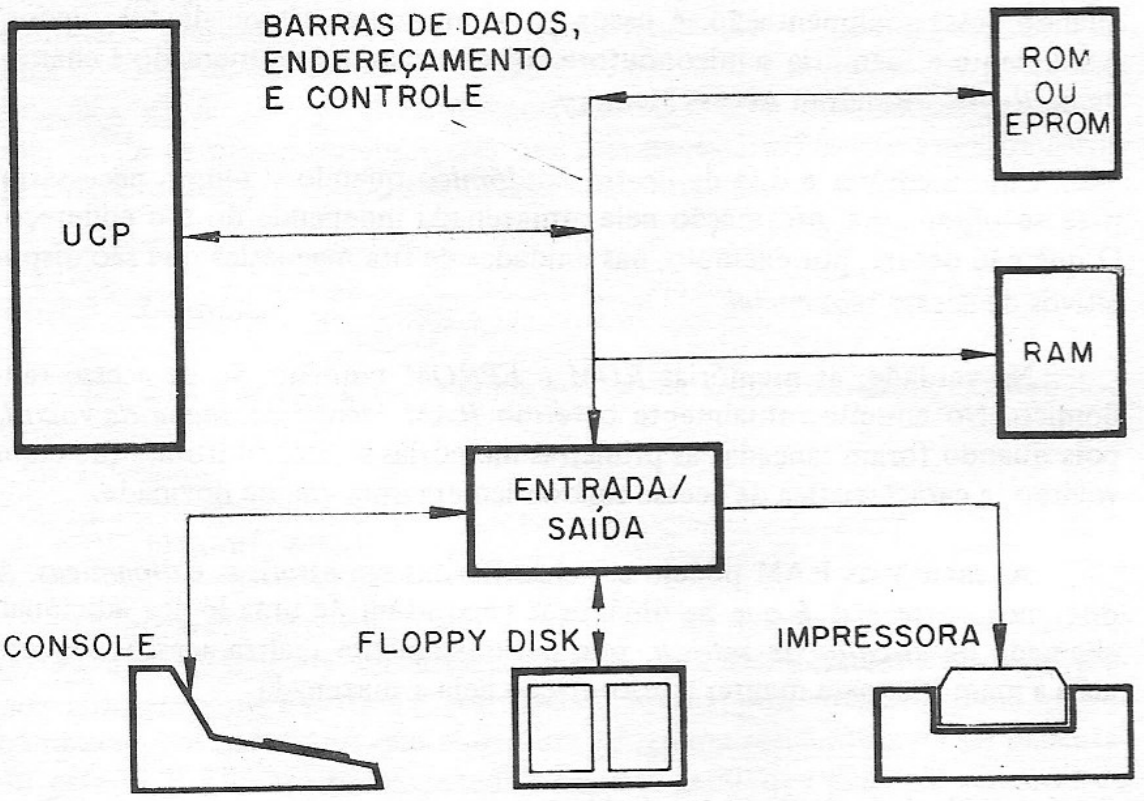


Fig. 2.3 – Processamento Comercial de Dados

O console e a impressora, de uma maneira geral, usam interface serial e código ASCII (American National Standard Code for Information Interchange).

O software básico é composto de compiladores e/ou interpretadores de linguagens de alto nível (FORTRAN, COBOL, BASIC . . .) e o sistema operacional na quase totalidade dos casos é o CP/M.

O CP/M (Control Program/Monitor) é o sistema operacional para microcomputadores mais utilizado no mercado mundial, tendo sido desenvolvido pela DIGITAL RESEARCH.

Assim, nesta área de processamento comercial de dados, os microcomputadores mesmo com seus reduzidos recursos de memória (comparando-os com computadores de grande porte), mas com linguagens de alto nível (COBOL, BASIC, PASCAL, . . .) e, principalmente, com seu reduzido custo, começam a promover uma descentralização de processamento nas grandes empresas. Isto, aliado aos recursos de telecomunicações hoje disponíveis, levam os microcomputadores a realizarem processamento local e, também, obter informações e serviços de outros computadores.

2.3.2.2 – Controle de Processos

Em *controle de processos,* encontramos microcomputadores dedicados à execução de tarefas específicas. Nesta área, para projetarmos um controlador antes do advento do microprocessador, tínhamos via de regra, duas opções:

1 – Implementá-lo através de circuitos lógicos de baixa escala de integração, tais como: flip-flops, gates, registradores, etc . . .
Esta solução gera alguns problemas: grande volume de circuitos e falta de flexibilidade; pois uma mudança nas funções do controlador acarreta alterações nos circuitos.

2 – Implementá-lo usando minicomputadores de uso geral disponíveis no mercado. Esta solução também gera alguns inconvenientes: custo elevado, instalações especiais e, na maioria dos casos, o fabricante do minicomputador não fornece todas as informações relativas ao hardware e ao software.

Os microprocessadores vieram preencher o espaço existente entre as duas soluções anteriores. Com sua flexibilidade e seu baixo custo, tornou-se possível a realização de controladores complexos, que eram inviáveis com circuitos lógicos de baixa escala de integração, mas por outro lado, não justificavam o uso de minicomputadores.

Na figura 2.4, temos a arquitetura básica de um controlador de processo, implementado por um microcomputador.

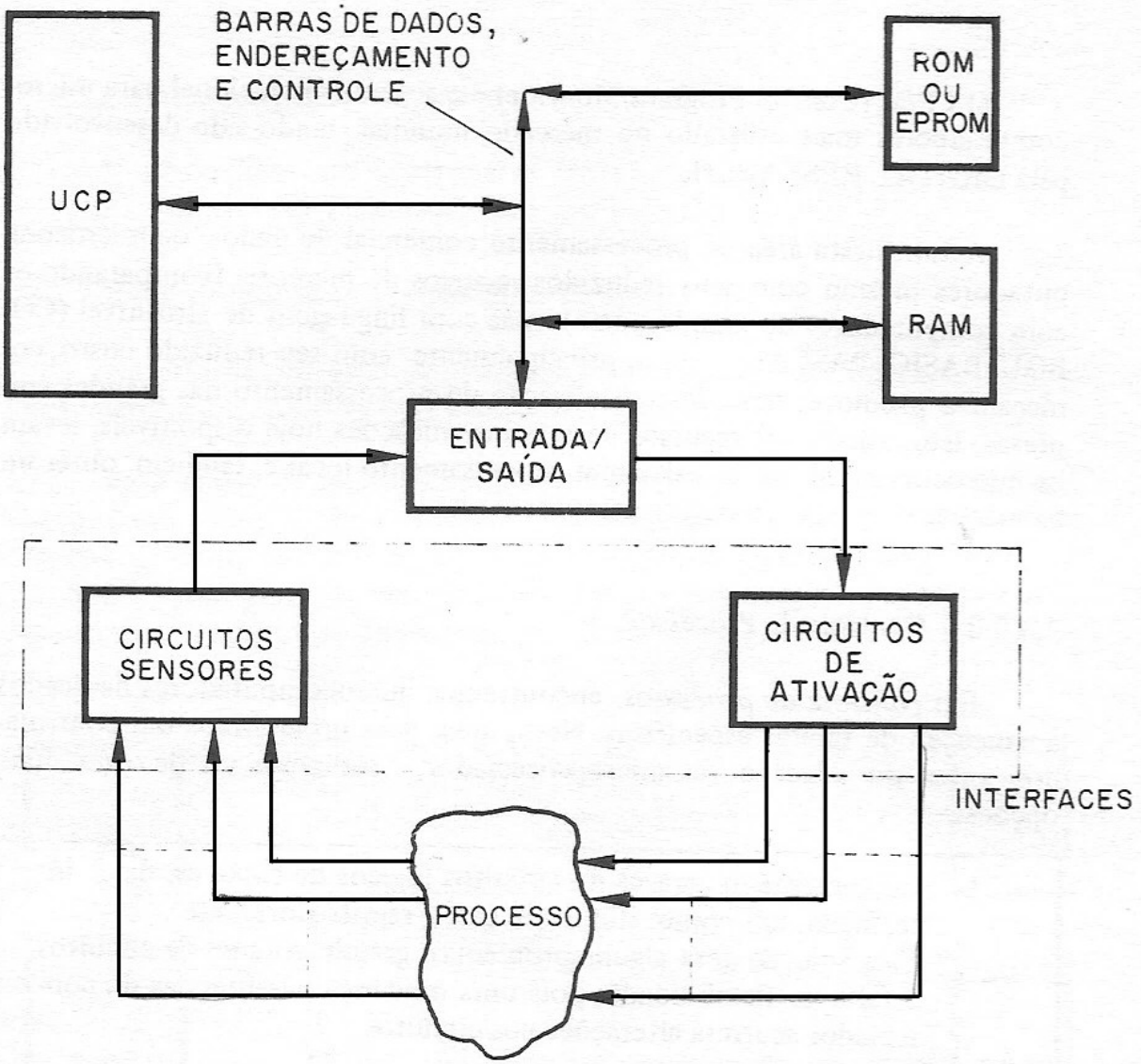


Fig. 2.4 – Controlador de Processo

Na memória não volátil (ROM ou EPROM) fica armazenado o programa de controle, que através da leitura do estado do processo, via os circuitos sensores, analisa estes dados e comanda os circuitos de ativação das saídas.

Um conceito que é importante conhecermos é a *Interrupção,* que é a capacidade do processador de interromper o programa em execução, atender a um pedido externo e retornar ao programa interrompido como se nada tivesse acontecido ao programa em execução.

A interrupção é fundamental no controle de processo, pois os eventos precisam ser tratados no instante em que ocorrem. Isto é chamado de *processamento em tempo real.*

O processador recebe um pedido de interrupção via uma entrada externa e, então, desvia a execução do programa para um endereço definido pelo hardware. Neste endereço deve estar previamente armazenada a *rotina*

de tratamento de interrupção, que após ser executada retorna o controle do processador para o programa interrompido. O ponto de retorno é a instrução que seria executada caso não tivesse ocorrido o pedido de interrupção.

Na memória volátil (RAM), normalmente ficam armazenadas as variáveis relativas ao processo controlado.

A unidade de entrada e saída comunica-se com os circuitos sensores e circuitos de ativação.

Os *circuitos sensores* são responsáveis pela conversão das diferentes entradas em sinais digitais. Dependendo da natureza destas entradas, temos conversores analógico-digitais, transdutores de temperatura, de pressão, etc.

Os *circuitos de ativação* recebem da unidade de entrada e saída informações digitais e as converte, ativando as saídas correspondentes. Dependendo da natureza das saídas, temos conversores digital-analógicos, ativadores de relés, de motores e outros.

Quando temos somente entradas a serem analisadas, denominamos estes microcomputadores de *observadores,* pois sem as saídas para interferir no processo, eles passam a ser observadores passivos.

Observando a arquitetura da figura 2.4, se desejássemos controlar outro processo, modificaríamos, basicamente, o software e a interface. Isto nos leva ao conceito de um *hardware básico* composto de UCP, Memória e Entrada e Saída, que seria utilizado em vários controladores, facilitando a manutenção e diminuindo o custo dos projetos.

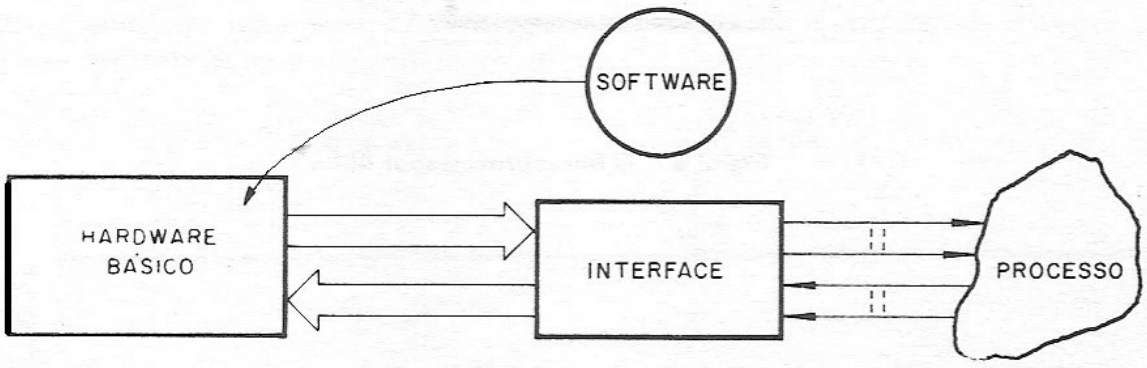


Fig. 2.5 – Hardware Básico

Temos usado este conceito de hardware básico no desenvolvimento de controladores na EMBRATEL. Assim, conseguimos minimizar o tempo de desenvolvimento, pois a cada novo controlador, nos dedicamos basicamente ao projeto da interface e do software.

2.4 – HISTÓRICO DOS MICROPROCESSADORES

Em 1971, a INTEL lançou o primeiro microprocessador no mercado, o 4004. Com repertório de 46 instruções, era adequado para aplicações de controle de processos que necessitavam de tomadas de decisões e operações aritméticas simples. O microprocessador 4004 tem uma barra de dados com extensão de 4 bits e tem a capacidade de realizar até 100.000 operações de adições por segundo, com dois operandos de quatro bits de extensão.

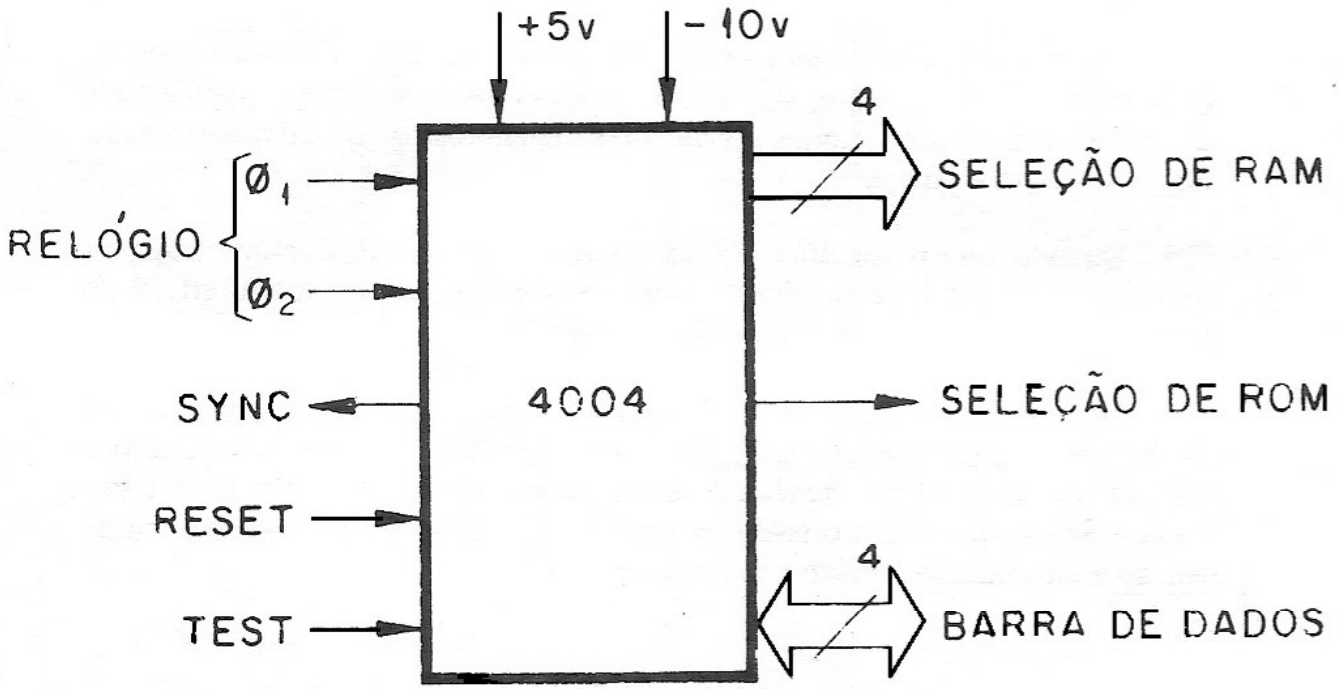


Fig. 2.6 – O Microprocessador 4004

A geração seguinte de microprocessadores caracterizou-se, principalmente, por uma barra de dados com extensão de 8 bits. Em 1972, foi lançada a pastilha 8008 pela INTEL, encabeçando os microprocessadores desta geração. Tal microprocessador apresenta um repertório de 48 instruções, podendo endereçar até 16384 células de memória, e podendo realizar aproximadamente 80.000 operações de adição por segundo, com dois operandos de oito bits de extensão.

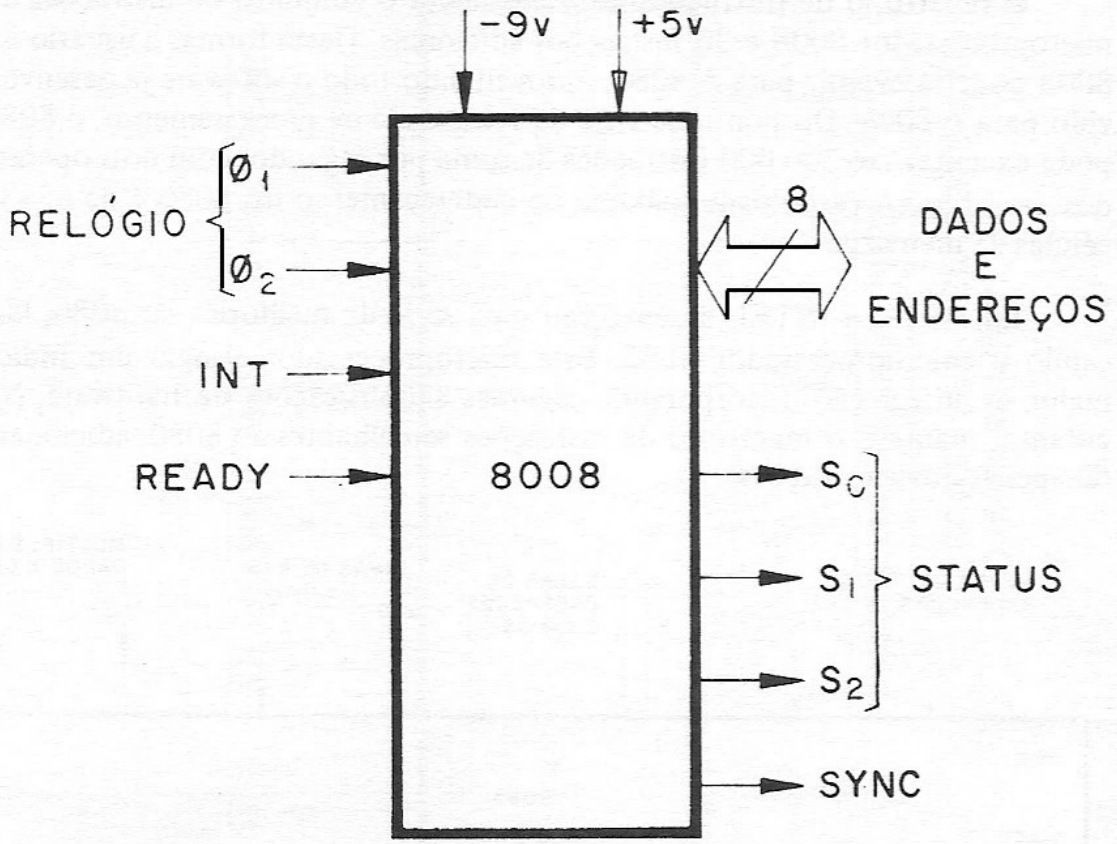


Fig. 2.7 – O Microprocessador 8008

Após o seu lançamento, o microprocessador 8008 foi largamente difundido através da indústria eletrônica num curto espaço de tempo, principalmente, porque não havia mais nada do gênero de outros fabricantes.

Ainda dentro da linha de microprocessadores de 8 bits, em 1973 a INTEL lançou o 8080, que se tornou um dos microprocessadores mais usados, assumindo uma posição privilegiada em relação aos numerosos concorrentes, posteriormente disseminados no mercado mundial.

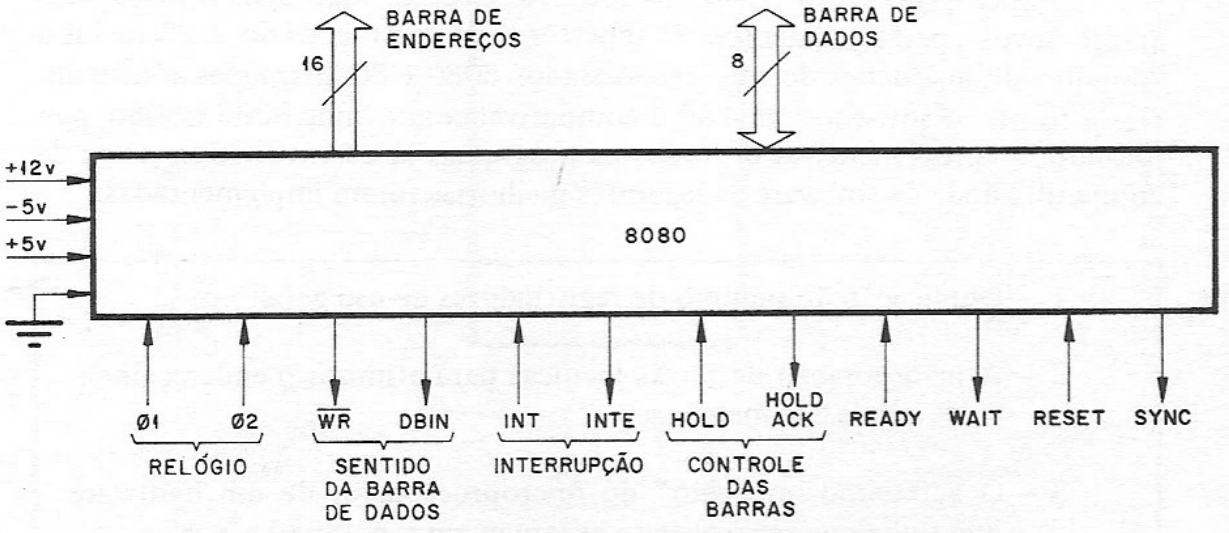


Fig. 2.8 – O Microprocessador 8080

O repertório de instruções do 8080 inclui o conjunto de instruções do microprocessador 8008 e 30 instruções adicionais. Desta forma, o usuário do 8008 poderia evoluir para o 8080, aproveitando todo o software já desenvolvido para o 8008. Do ponto de vista da velocidade de processamento, o 8080 pode executar até 500.000 instruções de soma por segundo, com dois operandos de 8 bits. A capacidade máxima de endereçamento do 8080 é de 65536 células de memória.

Em 1976, a INTEL desenvolveu uma série de melhorias no 8080, lançando o microprocessador 8085. Este microprocessador atingiu um índice maior de integração, incorporando algumas simplificações de hardware. No entanto, manteve o repertório de instruções semelhantes ao 8080, adicionando apenas duas instruções.

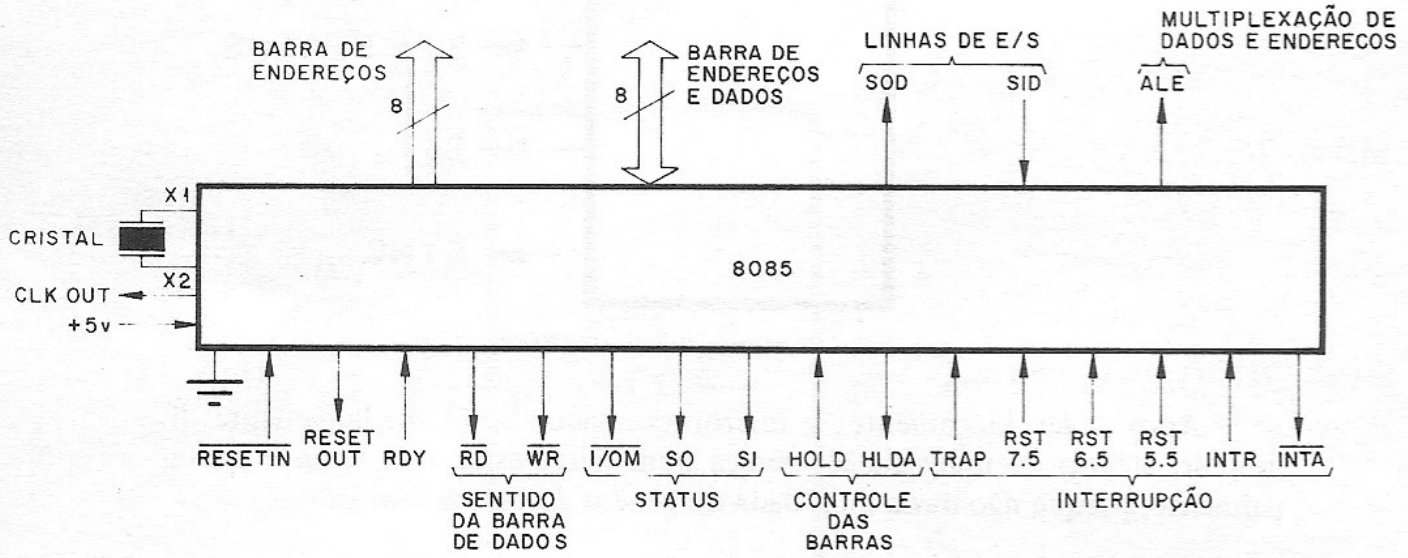


Fig. 2.9 – O Microprocessador 8085

O microprocessador Z-80 lançado pela ZILOG, logo após o 8085, veio trazer novos aperfeiçoamentos. O repertório de instruções do Z-80 inclui o conjunto de instruções do microprocessador 8080 e 80 instruções adicionais. Desta forma, o software do Z-80 é compatível com a linha 8008 e 8080, permitindo o aproveitamento de todos os programas já desenvolvidos. Além da compatibilidade de software as seguintes melhorias foram implementadas:

1 – Duplicação do número de registradores de uso geral.

2 – A incorporação de novas técnicas para otimizar o endereçamento de células de memória.

3 – O acréscimo no "chip" do microprocessador de um hardware que tinha que ser realizado externamente, no 8080 e 8085.

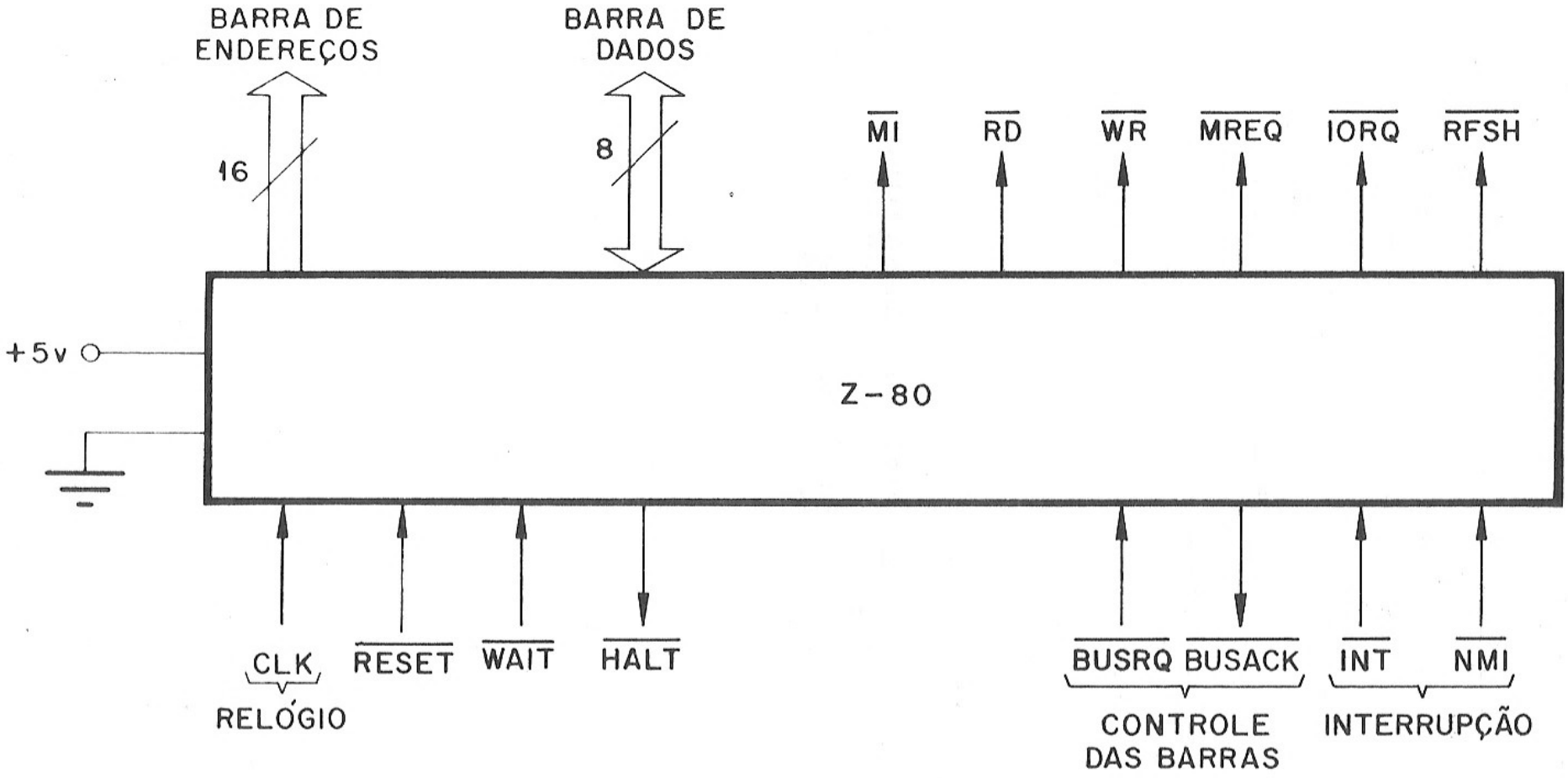


Fig. 2.10 — O Microprocessador Z-80

2.5 – EXERCÍCIOS DE FIXAÇÃO

2.5.1 – Conceituar microprocessador e microcomputador.

2.5.2 – Quais os principais parâmetros que definem a capacidade de realização de tarefas de um microprocessador?

2.5.3 – O que é memória de acesso randômico?

2.5.4 – O que é memória não volátil? Dê um exemplo.

2.5.5 – O que é sistema operacional?

2.5.6 – Quais os tipos de memória RAM?

2.5.7 – Qual o código mais utilizado entre os microcomputadores e os dispositivos de E/S?

2.5.8 – O que é CP/M?

2.5.9 – Antes dos microprocessadores quais as opções para o projeto de um controlador?

2.5.10 – Qual a função dos circuitos sensores?

2.5.11 – O que são observadores passivos?

2.5.12 – Qual a relação entre os microprocessadores 8008, 8080 e 8085?

2.5.13 – Quais as vantagens do Z-80 em relação ao 8080?

PARTE II
ESTUDO DO Z-80

3 A ARQUITETURA

3.1 – APRESENTAÇÃO

Neste capítulo iniciaremos o estudo do microprocessador Z-80, através de sua organização lógica interna e dos seus registradores.

Apresentaremos os registradores estudando-os conceitualmente e dividindo-os didaticamente em três grupos.

3.2 – ORGANIZAÇÃO LÓGICA INTERNA

A figura 3.1 mostra a organização lógica interna do microprocessador Z-80. A seguir descreveremos cada uma de suas partes:

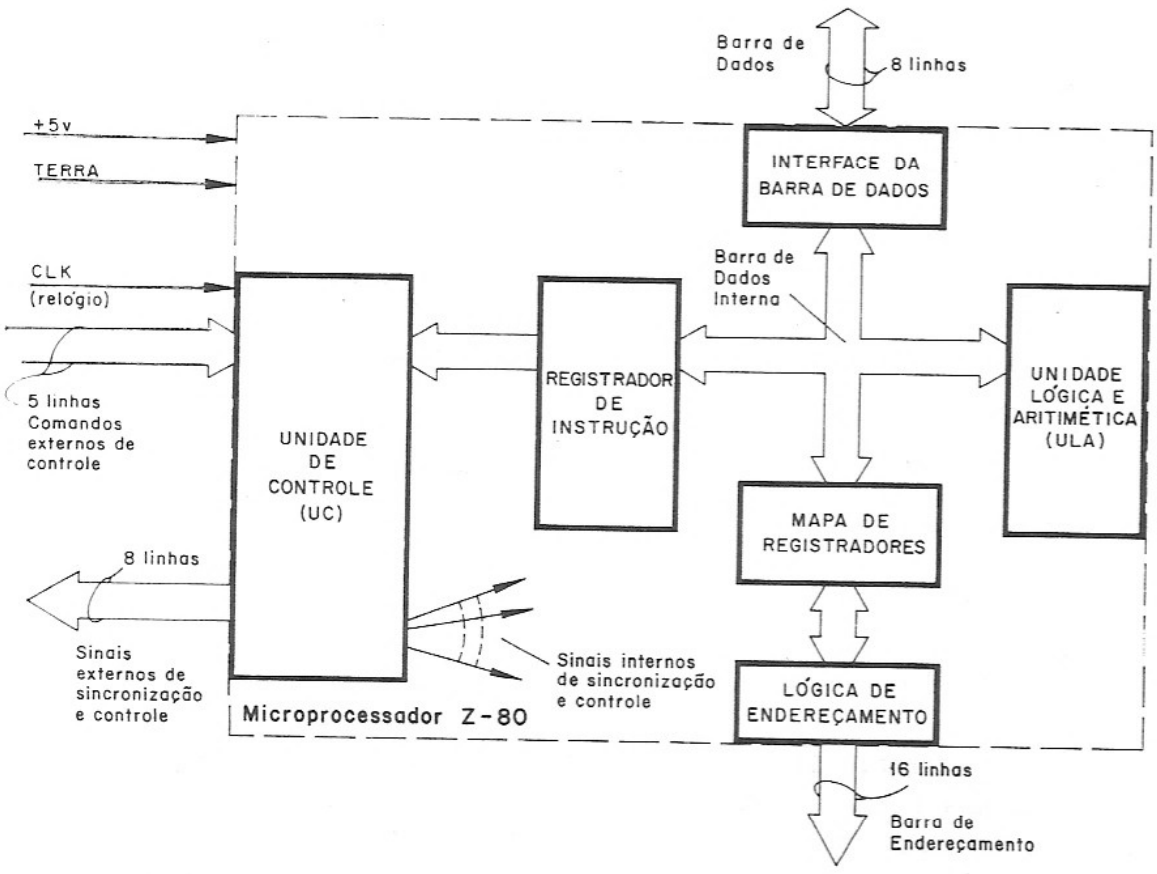


Fig. 3.1 – Organização Lógica do Z-80

A *barra de endereçamento* ("Address Bus") é composta de 16 bits, possibilitando endereçar até 2 EXP (16) = 65536 = 64K bytes de memória. Além disso, o Z-80 utiliza 8 bits desta barra para endereçar até 2 EXP(8) = 256 dispositivos de entrada ou saída.

Os dados (operandos ou instruções) destinados ao Z-80 ou dele procedentes, trafegam através da *barra de dados* ("Data Bus"), que é bidirecional e possui 8 bits. Esta barra se ramifica no interior do Z-80, através da *barra de dados interna* que estabelece a comunicação entre o *mapa de registradores*, o *registrador de instrução* e a *unidade lógica e aritmética* (ULA).

A *interface da barra de dados* acopla os setores internos e externos desta barra e tem duas funções básicas:

1 – Isolamento elétrico
2 – Captura e armazenamento

O *isolamento elétrico* ("buffer") entre circuitos elétricos das barras de dados interna e externa, faz com que as memórias e dispositivos de entrada e saída não sobrecarreguem os circuitos internos do Z-80, permitindo que um número maior destes dispositivos externos seja conectado a esta barra.

A *captura e armazenamento* ("latch") permite que estados lógicos de curta permanência na barra de dados sejam capturados e armazenados temporariamente.

A *lógica de endereçamento* determina, a cada momento, o endereço que deve ser colocado na barra de endereçamento, que também é equipada com buffers e latches.

A *ULA* concentra em si todas as operações lógicas e aritméticas.

O *mapa de registradores* reúne todos os registradores de processamento que o programador tem acesso.

O *registrador de instrução* encerra o código da instrução que está em processo de execução.

A função básica da *unidade de controle* (UC) é gerar os sinais de *sincronização e controle*, internos e externos ao Z-80, responsáveis pela consecução das atividades da máquina.

O automatismo do processamento é conseguido por meio destes sinais, que determinam quais as funções que devem ser executadas em determinados instantes de tempo, coordenando desta forma: as operações da ULA, as transferências internas ao Z-80 e as partes externas que compõem o microcompu-

tador. Tais sinais são precisamente distribuídos no tempo e no hardware, através de linhas individuais ligadas a pontos de controle dos circuitos internos ou externos ao Z-80.

A figura 3.2 ilustra o funcionamento da unidade de controle (UC).

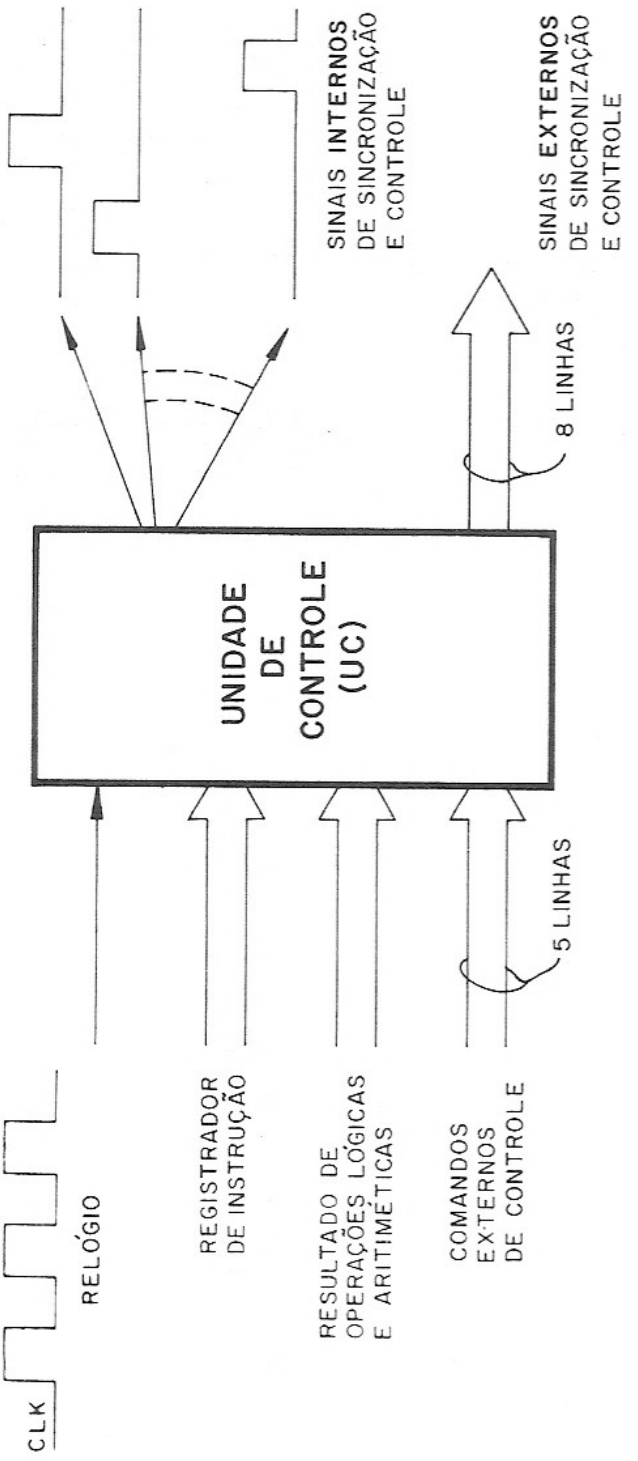


Fig. 3.2 – Funcionamento da UC

Observe que os sinais são gerados com base no conteúdo do registrador de instruções, nos resultados das operações lógicas e aritméticas e nos *comandos externos de controle.* O cadenciamento destes sinais é determinado pela entrada pulsada proveniente do relógio.

O registrador de instrução fornece informações relativas a instrução que está em fase de execução.

Através dos resultados das operações lógicas e aritméticas são escolhidos cursos alternativos de ação de programas.

Os comandos externos de controle reúnem um grupo de cinco linhas, que permitem aos módulos externos ao Z-80 interferir na seqüência natural de processamento. Estes comandos externos, quando ativados, forçam a execução de programas específicos (interrupção, reset), ou mesmo, podem paralisar temporariamente o processamento (wait, hold).

Os *sinais externos de sincronização e controle* reúnem um grupo de oito linhas que servem basicamente para controlar os acessos à memória e aos dispositivos de entrada e saída.

Finalmente, observando a figura 3.1, podemos identificar 40 pontos de comunicação do Z-80 com o meio exterior, que reúnem as barras, os sinais de controle e a alimentação.

3.3 – MAPA DE REGISTRADORES

Para o programador, um dos aspectos mais importantes a considerar no hardware do Z-80 é a arquitetura de seus registradores, ou seja, a organização lógica estrutural e funcional desses elementos.

A figura 3.3 apresenta o mapa de registradores do Z-80.

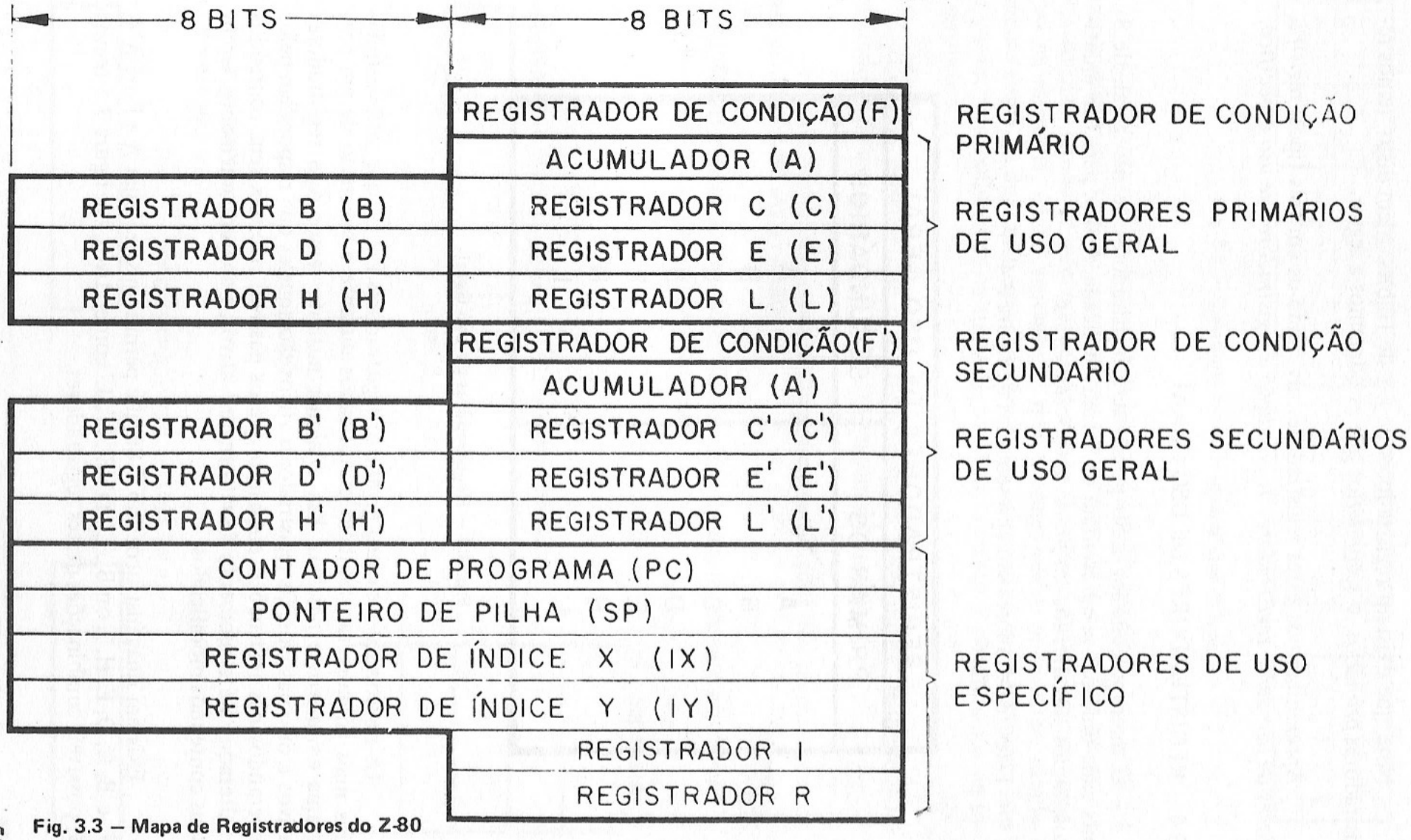


Fig. 3.3 – Mapa de Registradores do Z-80

Note que existem registradores de 8 e de 16 bits. Além disso, temos registradores *primários* e *secundários* que estudaremos a seguir.

Apresentaremos estes registradores dividindo-os em três tipos: *registradores de uso geral, registradores de condição* e *registradores de uso específico.*

3.4 – REGISTRADORES DE USO GERAL

O microprocessador Z-80 possui sete registradores de uso geral, de 8 bits cada um, que se apresentam de forma duplicada. Assim, para fins didáticos, vamos classificá-los em *primários* e *secundários.*

Estes registradores são apresentados na figura 3.4.

REGISTRADORES DE USO GERAL	
PRIMÁRIOS	SECUNDÁRIOS
A	A'
B	B'
C	C'
D	D'
E	E'
H	H'
L	L'

Fig. 3.4 – Registradores de Uso Geral

Os registradores de uso geral e os registradores F e F', que serão estudados mais adiante, são também classificados durante a execução de um programa em *ativos* e *passivos.* Em qualquer instante temos oito registradores ativos e oito passivos. Os registradores ativos são aqueles que respondem pelos operandos das instruções, enquanto que os passivos mantêm seus conteúdos inalterados até o momento de se tornarem ativos, e conseqüentemente, terem seus conteúdos modificados.

Existem duas instruções no Z-80 que permitem selecionar A e F ou A' e F' e B, C, D, E, H, L ou B', C', D', E', H', L' como ativos. Na figura 3.5 temos as possíveis combinações destes registradores.

A	F
B	C
D	E
H	L

A'	F'
B'	C'
D'	E'
H'	L'

A'	F'
B	C
D	E
H	L

A	F
B'	C'
D'	E'
H'	L'

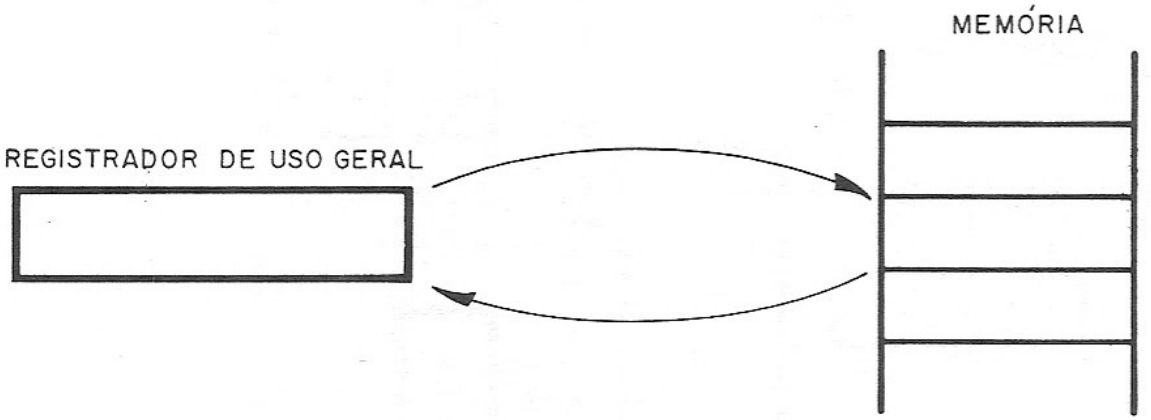
Fig. 3.5 – Combinações de Registradores Ativos

Uma vantagem de termos dois blocos de registradores de uso geral é a maior disponibilidade de registradores no interior do Z-80, o que permite o acesso mais rápido às informações do que se as mesmas estivessem armazenadas na memória. Outra vantagem é que o programador pode trocar rapidamente (uma ou duas instruções) de um bloco de registradores para outro. Isto é usado, principalmente, na troca de contexto durante o atendimento de interrupções.

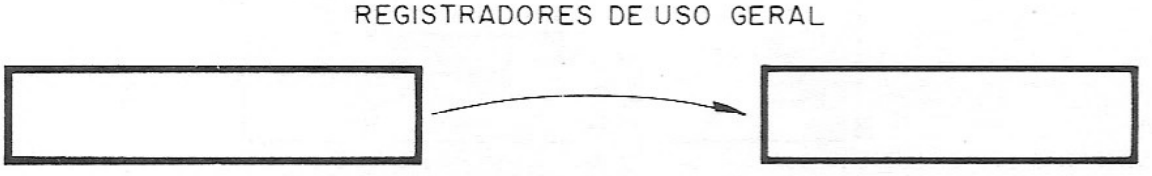
Convém ressaltar que o microprocessador 8080 possui os mesmos registradores de uso geral que o Z-80, porém sem duplicação.

A seguir temos um resumo das principais funções dos registradores de uso geral no Z-80 ou no 8080:

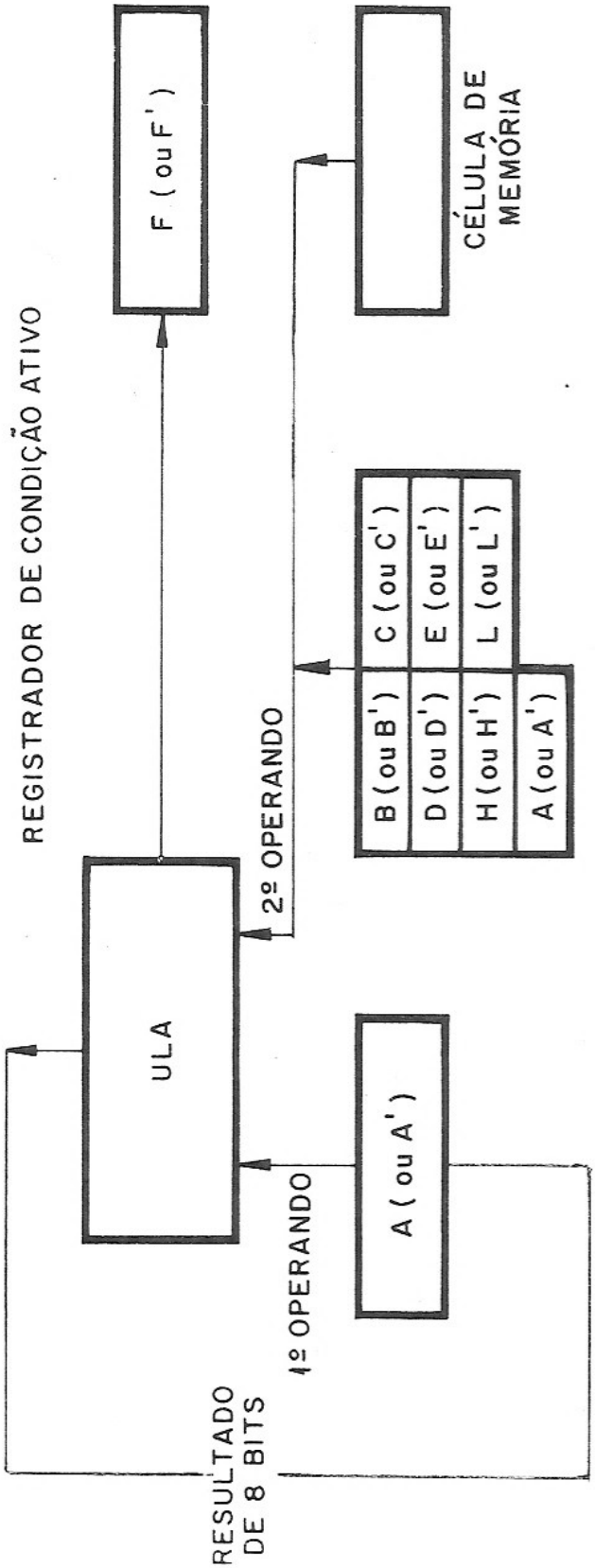
a) Transferência do conteúdo de oito bits de um registrador ativo para uma posição de memória ou vice-versa.



b) Transferência do conteúdo de um registrador de uso geral ativo para um outro.



c) Operações lógicas e aritméticas de 8 bits são realizadas usando o acumulador ativo, outro registrador ativo de uso geral ou o conteúdo de uma posição de memória. O resultado sempre é depositado no acumulador.

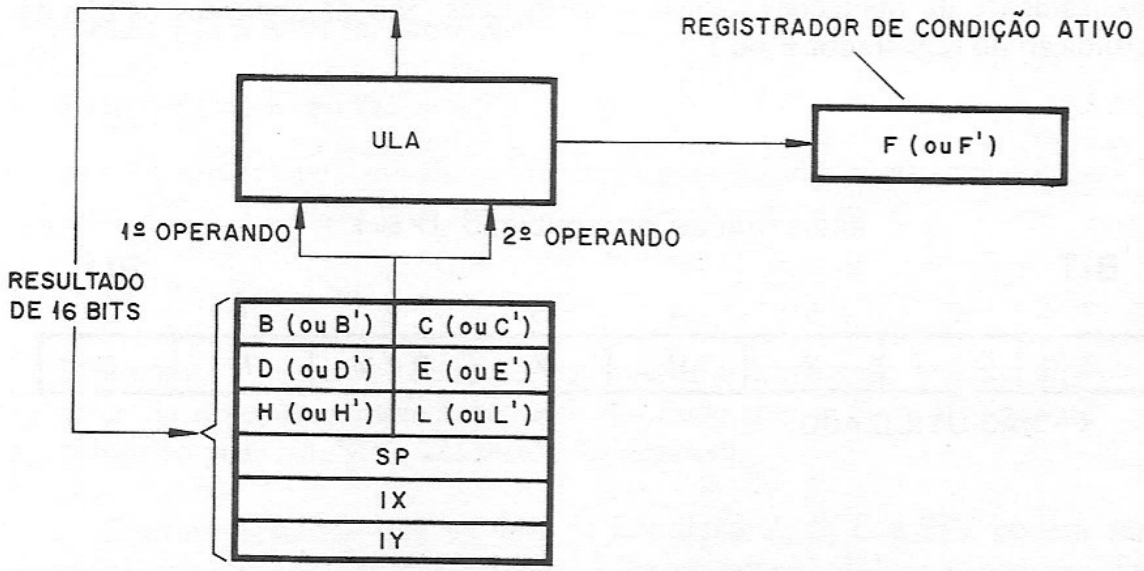


d) Os registradores de uso geral podem ser usados em *pares*, formando registradores de 16 bits. Os pares possíveis são BC, DE, HL (ou B'C', D'E' ou H'L').

B (ou B')	C (ou C')	par BC (ou B'C')
D (ou D')	E (ou E')	par DE (ou D'E')
H (ou H')	L (ou L')	par HL (ou H'L')

Em várias instruções dos microprocessadores Z-80 e 8080, os dados contidos nos três *pares de registradores* representam endereços de posições de memória. Neste caso, o conteúdo dos registradores B, D e H representam a parte mais significativa do endereço e o conteúdo dos registradores C, E e L representam a parte menos significativa. Quando o conteúdo dos pares de registradores representam endereço, eles são chamados de *ponteiros* ("pointers").

e) Os pares de registradores (e os registradores SP, IX e IY) podem também ser usados em *operações aritméticas de dupla precisão*, isto é, como operandos de 16 bits. Este tipo de operação é usado principalmente na manipulação de ponteiros.
Sem as instruções de dupla precisão seriam necessárias duas operações aritméticas de 8 bits.



Vejamos agora alguns comentários complementares sobre o uso dos registradores de uso geral.

O par de registradores HL (ou H' L') é privilegiado em relação aos outros dois, pois o seu conteúdo é usado como ponteiro em diversas modalidades de instruções. O Z-80 e o 8080 possuem instruções que transferem um byte entre qualquer registrador de uso geral e uma posição de memória endereçada pelo conteúdo do par HL (ou H' L'). Além disso, as instruções lógicas e aritméticas permitem usar o par HL (ou H'L') como ponteiro de seus operandos. Assim, o programador deve, preferencialmente, usar o par HL (ou H' L') como ponteiro no acesso à memória. Por outro lado, os registradores B, C, D e E (ou B', C', D' e E') devem ser usados para armazenamento temporário de dados ou servir de segundo operando nas operações lógicas e aritméticas. Além disso, existem algumas instruções que usam os pares BC (ou B' C') e DE (ou D' E') como ponteiros no acesso à memória. No entanto, tal grupo de instruções limita-se a transferir informações entre o acumulador e a memória.

3.5 – REGISTRADORES DE CONDIÇÃO

O Z-80 possui os registradores de condição F e F' ("Flag Register") sendo que, como já foi visto, um deles é ativo e o outro passivo.

Cada registrador de condição tem oito bits, dos quais somente seis são usados.

O registrador de condição ativo (F ou F') funciona associado ao acumulador e à ULA, e cada um de seus bits indica uma situação definida, resultante basicamente de operações lógicas e aritméticas. São os seguintes os bits de condição do registrador F ou F':

REGISTRADOR DE CONDIÇÃO (F ou F')

BIT 7	6	5	4	3	2	1	BIT 0
S	Z	X	H	X	P / V	N	C

X - NÃO UTILIZADO

Fig. 3.6 – Registrador F ou F'

Agora teremos uma descrição funcional de cada um dos bits de condição que compõem o registrador F ou F':

a) BIT C ("Carry bit")

Indica a ocorrência de "vai um" no 8º bit após a realização de uma soma, ou de empréstimo ("borrow") quando se efetua uma subtração, assumindo o valor 1 se houver tal ocorrência, ou o valor 0 se não houver.

b) BIT N ("Subtract bit")

De uma maneira geral, assume o valor 0 após as instruções de soma, e o valor 1 após as instruções de subtração.

c) BIT P/V ("Parity/Overflow bit")

Este bit pode representar de um modo geral, *paridade ou overflow,* dependendo da natureza da instrução executada.
A paridade representa a natureza par ou ímpar do número de bits '1' do resultado.
A ocorrência de overflow indica que o resultado da operação aritmética não coube no registrador a ele destinado.
Quando o bit P/V representar a paridade do resultado, P/V = 1 indica paridade par e P/V = 0 indica paridade ímpar.
No caso do bit P/V indicar overflow, se P/V = 1 após uma operação aritmética, significa a ocorrência de overflow.

d) BIT H ("Half-carry bit")

Indica a ocorrência de "vai um" ou empréstimo do quarto para o quinto bit do resultado, na realização da soma ou da subtração, assumindo o valor 1 se houver tal ocorrência.

e) BIT Z ("Zero bit")

Z = 1 indica que o resultado de uma operação lógica ou aritmética é nulo, e Z = 0 indica que o resultado é diferente de zero.

f) BIT S ("Signal bit")

Representa o sinal do resultado respeitando a convenção de sinal algébrico da *notação complemento a 2,* de modo que se S = 0 indica um resultado positivo, S = 1 um resultado negativo.

Convém ressaltar que os bits de condição Z, S, C e P/V podem ser testados pelas instruções que alteram o curso normal de um programa. As escolhas de cursos alternativos dos programas são realizadas com base nos estados dos referidos bits de condição.

Os bits H e N são usados para facilitar as operações aritméticas com números representados na *notação BCD.*

3.6 – REGISTRADORES DE USO ESPECÍFICO

Cada registrador específico tem uma função bem definida durante o processamento. Temos quatro registradores de 16 bits e dois de 8 bits:

1 – Contador de programa (PC-"Program Counter")
2 – Ponteiro da pilha (SP-"Stack Pointer")
3 – Registradores de índice (IX e IY - "Index Register")
4 – Registrador de interrupção (I - "Interrupt Register")
5 – Registrador de refresh (R - "Memory Refresh")

O microprocessador 8080 somente possui os registradores PC e SP, que realizam funções semelhantes às desempenhadas no Z-80.

A seguir apresentaremos conceitualmente estes registradores específicos.

3.6.1 – O Contador de Programa (PC)

O PC é um registrador de 16 bits que é atualizado em cada instrução, de modo a conter o endereço da próxima instrução a ser executada.

As instruções do Z-80 podem ocupar de um a quatro bytes consecutivos da memória.

Do ponto de vista do programador, o conteúdo do PC endereça sempre o início da próxima instrução a ser executada, isto é, o seu primeiro byte. Desta forma, a unidade de controle incrementa automaticamente o conteúdo do PC de um, dois, três ou quatro, dependendo do número de bytes da instrução que está em fase de execução.

Como conseqüência natural do funcionamento do PC, as instruções de um programa são organizadas na memória em posições consecutivas e em ordem crescente de endereços. A figura 3.7 ilustra o funcionameto do PC.

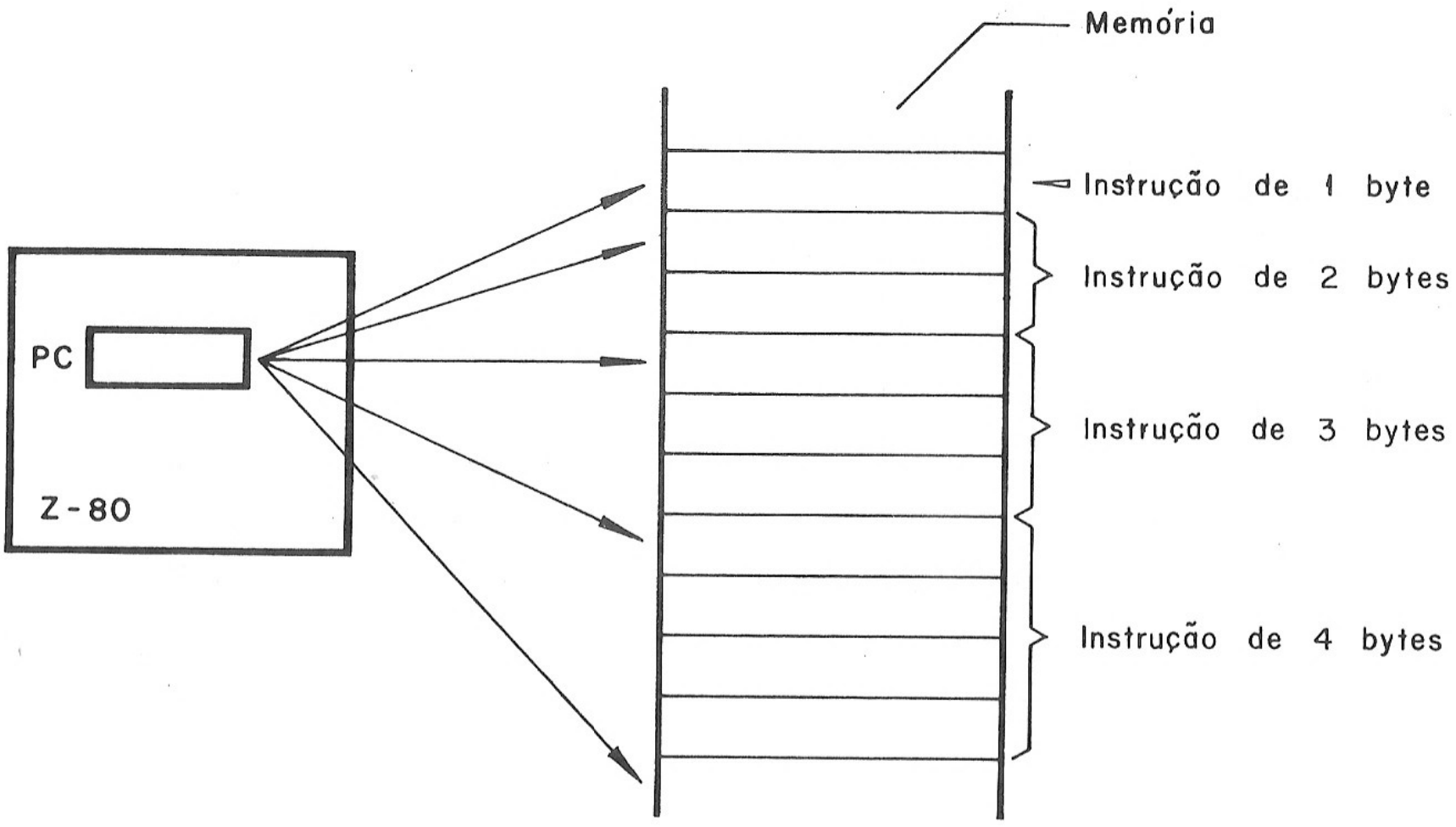


Fig. 3.7 – O Funcionamento do PC.

Esta atualização sequencial do PC é quebrada quando o Z-80 executa instruções que alteram o curso normal do programa. Estas instruções, por exemplo: JP, CALL, RET e RST, carregam o PC com um novo valor.

3.6.2 – O Ponteiro da Pilha (SP)

O SP é um registrador de 16 bits que tem a função de *ponteiro da pilha.*

A *pilha* é uma estrutura de informação implementada em RAM, que é muito utilizada em computadores de um modo geral.

Numa linguagem mais formal, a pilha é uma *lista linear* em que inserções e deleções de elementos são realizadas em um dos extremos desta lista.

Listas lineares são conjuntos de elementos X (1), X (2), ...X (n), cujas propriedades estruturais envolvem, essencialmente, posições relativas unidimensionais.

A dinâmica de funcionamento da pilha pode ser compreendida na prática, com a inserção e retirada de bolas em uma caçapa.

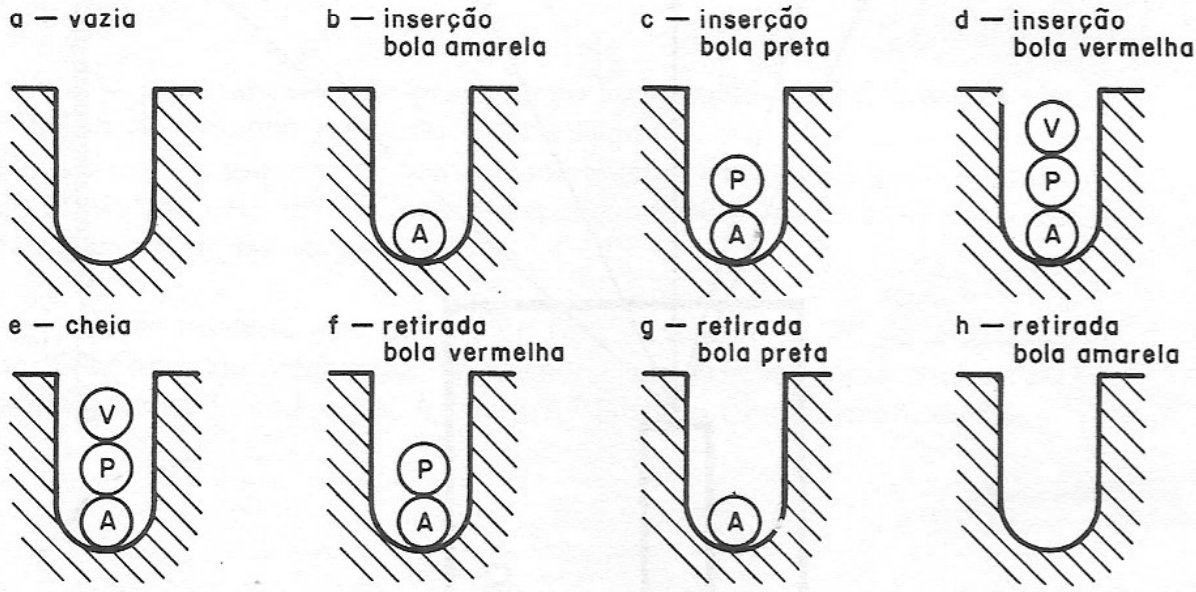


Fig. 3.8 – O Funcionamento da Pilha

Observe que nesta estrutura o *último elemento inserido é o primeiro a ser retirado.* Este é o princípio básico de funcionamento da pilha. Quando implementada na memória, o SP endereça o último dado inserido ou o seguinte ao que foi retirado, isto é, o *topo da pilha.*

Nos microcomputadores, a principal utilização da pilha é o armazenamento temporário do conteúdo de registradores de uma maneira sistemática e organizada.

As instruções do Z-80 que manipulam a pilha realizam transferências de 2 bytes. Assim, podem ser transferidos em uma única operação, o conteúdo de qualquer registrador específico de 16 bits (exceto o SP), o de qualquer par de registradores de uso geral e o do par formado pelo acumulador (A ou A') e o registrador ativo de condição (F ou F').

Outra aplicação da pilha é o armazenamento do endereço de retorno em *chamadas de sub-rotinas* e no *tratamento de pedidos de interrupção.*

3.6.3 – Os Registradores de Índice (IX e IY)

IX e IY são dois registradores de 16 bits que permitem ao Z-80 calcular endereços de operandos de instruções que utilizam a técnica de *endereçamento indexado.* Nesta técnica, o endereço efetivo do operando é a soma do campo de endereço da instrução ("displacement") com o conteúdo de um dos registradores de índice, especificado no campo do código da instrução.

As instruções com endereçamento indexado são usadas principalmente em aplicações com *listas e tabelas.*

A figura 3.9 ilustra o funcionamento do registrador de índice.

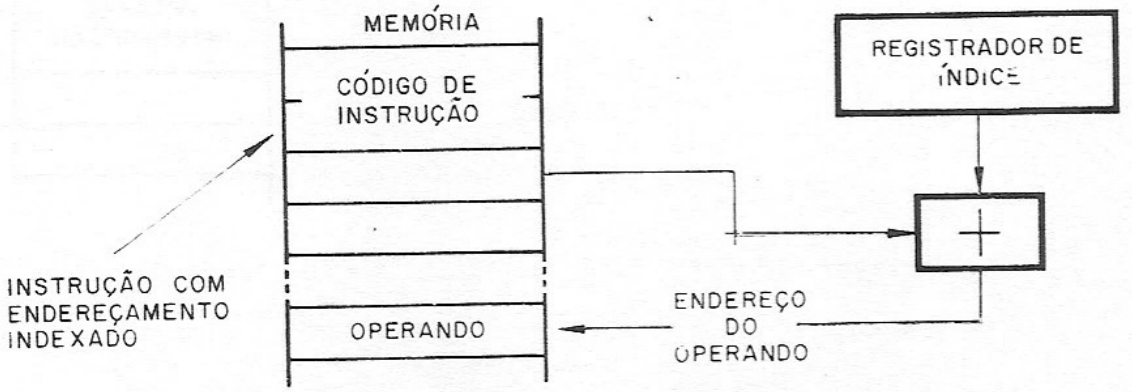


Fig. 3.9 – Endereçamento Indexado

3.6.4 – O Registrador Vetor de Interrupção (I)

O Registrador I, também chamado *Registrador Vetor de Interrupção*, tem 8 bits e é usado no atendimento de interrupções. O seu conteúdo é previamente carregado pelo programador e representa os 8 bits mais significativos do endereço do vetor de interrupção. A parte menos significativa deste endereço é gerada pelo dispositivo de entrada e saída que originou o pedido de interrupção. Estes 8 bits menos significativos são fornecidos através da barra de dados, em intervalos de tempo sincronizados pelo relógio do Z-80 e durante o processo de atendimento de interrupção.

O *vetor de interrupção* é o endereço da primeira instrução da *rotina de tratamento de interrupção*. Associado a uma instrução de desvio leva o Z-80 a executar esta rotina. Assim, cada dispositivo de entrada e saída ao solicitar uma interrupção força o Z-80 a executar a sua rotina de tratamento específica.

A figura 3.10 ilustra a formação do endereço de interrupção.

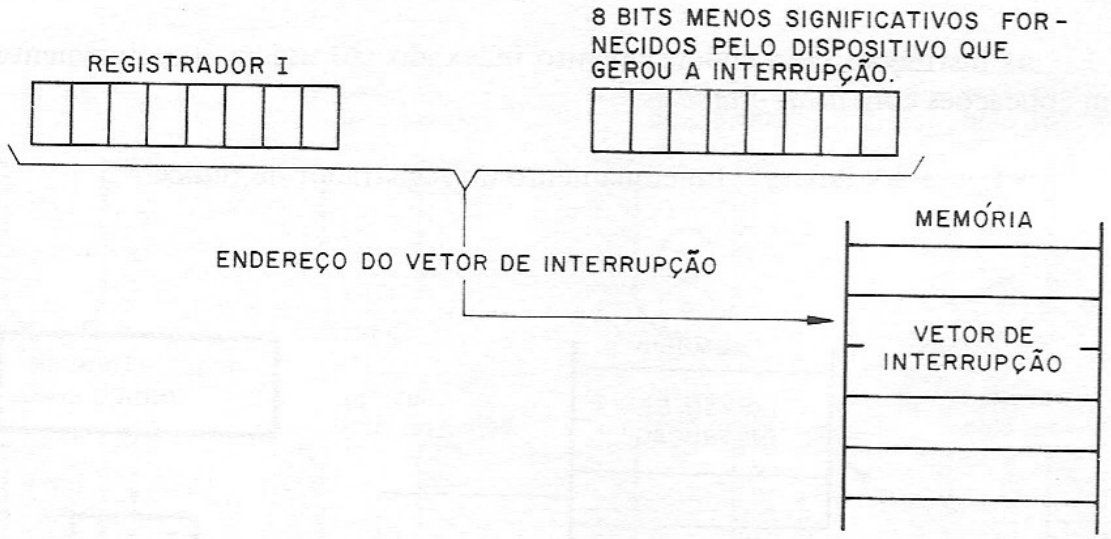


Fig. 3.10 – Formação do Endereço Vetor de Interrupção

3.6.5 – O Registrador de Refresh de Memória (R)

.strador R, chamado *registrador de refresh da memória,* possui 8 bits e sua .unção é controlar o *refresh* automático de memórias semicondutoras dinâmicas.

Normalmente, cada célula de memória dinâmica sofre um *refresh* periódico, num intervalo inferior a 2ms. O mecanismo de *refresh* é, na maioria dos casos, a realização de um ciclo de leitura, que na maioria das pastilhas processa-se de uma só vez, em grupos de até 128 células de memória. O conteúdo do registrador R designa, de uma forma cíclica, o grupo que deve sofrer o *refresh.*

Após a busca de cada instrução, o Z-80 incrementa de uma unidade o conteúdo do registrador R e o coloca nos 7 bits menos significativos da barra de endereços. Isto ocorre em paralelo com o processamento interno da instrução, justamente nos intervalos de tempo em que a execução da instrução não requer o uso das barras.

Isto é uma grande virtude do Z-80, pois além do *refresh* transcorrer de forma transparente ao programa, também dispensa o uso de circuitos externos.

3.7 – EXERCÍCIOS DE FIXAÇÃO

3.7.1 – Porque a capacidade máxima de endereçamento de memória é 65536 bytes e a de dispositivos de entrada e saída é 256?

3.7.2 – Quais as funções da interface da barra de dados?

3.7.3 – Explique o funcionamento da unidade de controle (UC).

3.7.4 – O que são registradores Ativos e Passivos?

3.7.5 – Quais as vantagens de termos os registradores de uso geral de forma duplicada?

3.7.6 – Qual o par de registradores que, preferencialmente, deve ser usado como ponteiro? Porque?

3.7.7 – Explique o significado de cada um dos bits de condição.

3.7.8 – Qual a função do registrador contador de programa (PC)?

3.7.9 – O que é memória pilha?

3.7.10 – O que é refresh de memória?

3.7.11 – Quais os registradores que permitem a utilização do endereçamento indexado?

3.7.12 – Porque as instruções são organizadas em endereços seqüenciais na memória?

3.7.13 – Quais são as principais funções dos registradores de uso geral?

3.7.14 – Explique como é determinado o endereço do operando no endereçamento indexado?

3.7.15 – O que é vetor de interrrupção?

3.7.16 – Como é determinado o endereço do vetor de interrupção?

3.7.17 – Quais são as vantagens do registrador R?

4 IDENTIFICAÇÃO DOS PINOS

4.1 – APRESENTAÇÃO

Neste capítulo estudaremos as características elétricas do Z-80 e a sua compatibilidade com a família de circuitos integrados TTL; que é a mais popularmente usada para complementar a lógica do Z-80.

Apresentaremos, ainda, os circuitos lógicos necessários ao entendimento do hardware do Z-80 e a sua ligação com módulos de memória, de entrada e saída.

Finalizando, estudaremos conceitualmente cada um dos sinais do Z-80.

4.2 – INTRODUÇÃO

Na figura 4.1 temos a identificação dos 40 pinos do Z-80. Observe que esta disposição não corresponde à ordem física dos pinos no circuito integrado, mas a uma ordem lógica que facilita o entendimento.

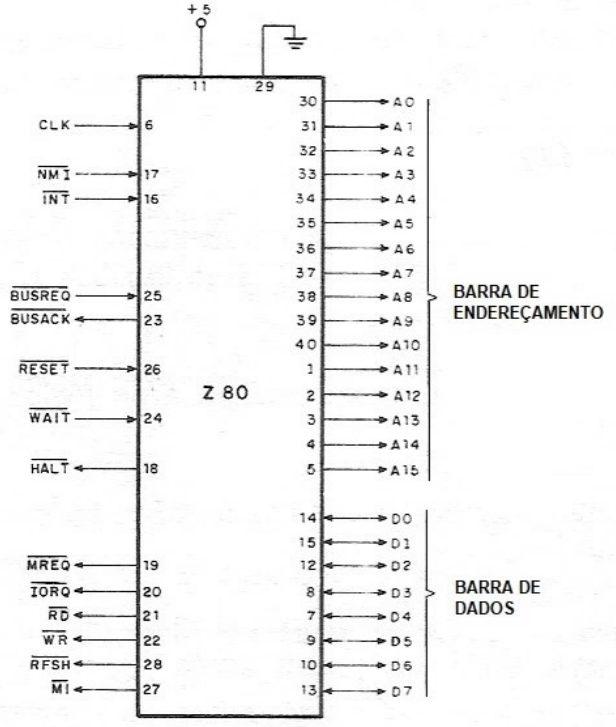


Fig. 4.1 – Os Pinos do Z-80

O microprocessador Z-80 é um circuito integrado de 40 pinos, encapsulado em cerâmica ou plástico, onde os pinos estão dispostos em duas linhas de 20 elementos cada uma.

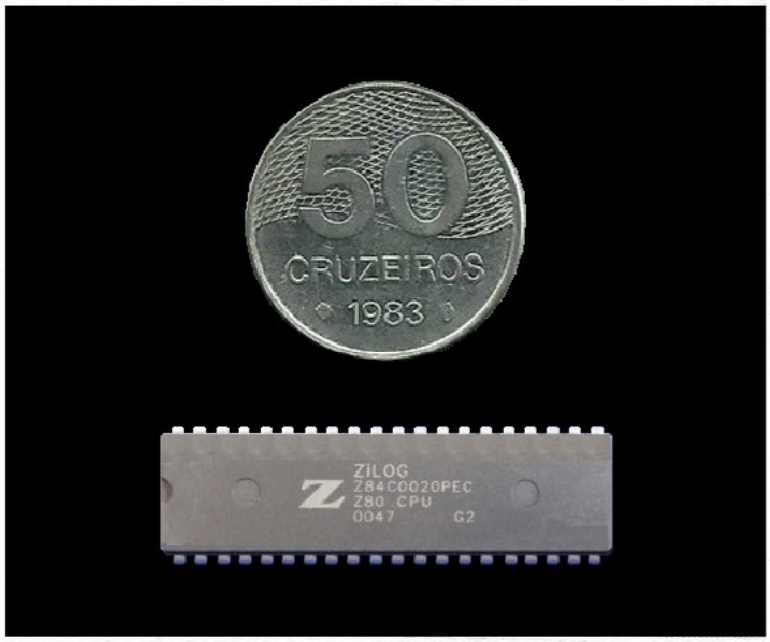


Fig. 4.2 – O Microprocessador Z-80

4.3 – COMPATIBILIDADE COM TTL

Todos os pinos do Z-80 podem assumir os estados lógicos 0 e 1. Estes estados são representados eletricamente por níveis de tensão compatíveis com a lógica TTL (Transistor Transistor Logic), como teremos oportunidade de concluir mais adiante. Para tanto, faremos um breve estudo das principais características da família lógica TTL.

4.3.1 – A Família TTL

Os circuitos integrados TTL são alimentados, unicamente, pela tensão de + 5 Volts, e seus níveis lógicos 0 e 1 estão situados dentro dos seguintes limites de tensão elétrica:

VIH MIN – Tensão mínima de entrada no nível lógico 1 = 2,0 V. (MAX = 5V)

VIL MAX – Tensão máxima de entrada no nível lógico 0 = 0,8 V. (MIN. = 0V)

VOH MIN – Tensão mínima de saída no nível lógico 1 = 2,4 V. (MAX = 5V)

VOL MAX – Tensão máxima de saída no nível lógico 0 = 0,4 V. (MIN = 0V)

Atualmente, existe disponível no mercado uma enorme variedade de circuitos integrados da família TTL, tais como: portas lógicas, memórias, somadores, decodificadores e outros.

A série 74XXX de circuitos integrados TTL, também chamada de standard, é a mais comercializada. Existem variações desta versão com características peculiares, que são úteis em aplicações específicas. A tabela 4.1 identifica estas versões, comparando o consumo típico (Icc) e o atraso médio (tp) de uma porta lógica de cada uma destas versões.

TABELA 4.1 - VERSÕES DA FAMÍLIA TTL

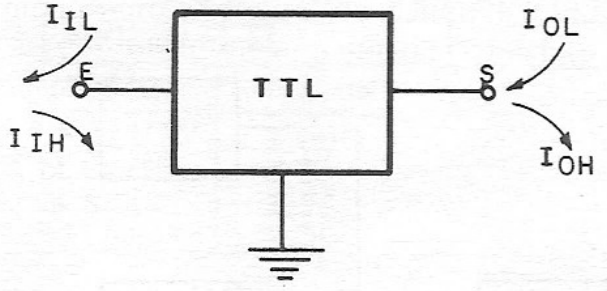
VERSÃO	IDENTIFICAÇÃO	Icc (mA)	tp (nS)
STANDARD	74 XXX	2.00	10.0
LOW POWER	74 LXXX	0.20	33.0
SCHOTTKY	74 SXXX	3.75	3.0
LOW SCHOTTKY	74 LSXXX	0.40	9.5

Tabela 4.1 — Versões da Família TTL

Atualmente, a versão Low Schottky (LS) é a mais utilizada, pois o consumo (Icc) é menor do que o da versão Standard e o atraso por porta é aproximadamente igual ao desta versão.

4.3.2 – Interligando Circuitos TTL

Uma característica importante dos circuitos TTL são as correntes elétricas provenientes da interligação destes circuitos integrados. Em um circuito TTL identificamos quatro correntes externas, conforme mostra a figura 4.3.



I_{IL} = CORRENTE DE ENTRADA COM NÍVEL LÓGICO Ø
I_{IH} = CORRENTE DE ENTRADA COM NÍVEL LÓGICO 1
I_{OL} = CORRENTE DE SAÍDA COM NÍVEL LÓGICO Ø
I_{OH} = CORRENTE DE SAÍDA COM NÍVEL LÓGICO 1

Fig. 4.3 – Correntes Elétricas Externas

Os valores destas correntes variam de acordo com a versão (74, 74L, ...) e definem o número máximo de circuitos que podem ser interligados.

As correntes máximas de entrada IIL e IIH e de saída IOL e IOH das diferentes versões são apresentadas na tabela 4.2. Os valores são válidos para temperaturas entre 0 e 70°C e com tensão máxima de alimentação (5,25 Volts).

TABELA 4.2 - CORRENTES IIL, IIH, IOL E IOH

VERSÃO	IIL MAX	IIH MAX	IOL MAX	IOH MAX
74 XXX	1,6 mA	40 µA	16 mA	400 µA
74 LXXX	180 µA	10 µA	3,6 mA	200 µA
74 SXXX	2 mA	50 µA	20 mA	1 mA
74 LSXXX	400 µA	20 µA	8 mA	400 µA

Tabela 4.2 – Correntes IIL, IIH, IOL e IOH

Os valores das correntes de saída, IOL e IOH, dependem do número de entradas conectadas à respectiva saída. É importante que as correntes IOL e IOH não ultrapassem seus limites máximos, relacionados na tabela 4.2. O fabricante só garante as tensões de saída correspondentes aos níveis lógicos, se estas correntes não ultrapassarem os valores máximos permitidos. Apresentaremos dois métodos para dimensionar o número de entradas que podem ser ligadas a uma saída: o *método direto* e o *método normalizado.*

4.3.2.1 – Método Direto

O *método direto* consiste em calcular as correntes totais IOL e IOH, decorrentes das parcelas IIL e IIH de cada entrada. Estes totais são comparados com os valores máximos de IOL e IOH da tabela 4.2. A título de exemplo, avaliaremos o circuito da figura 4.4, aplicando o método direto com o objetivo de verificar se tal circuito está corretamente dimensionado.

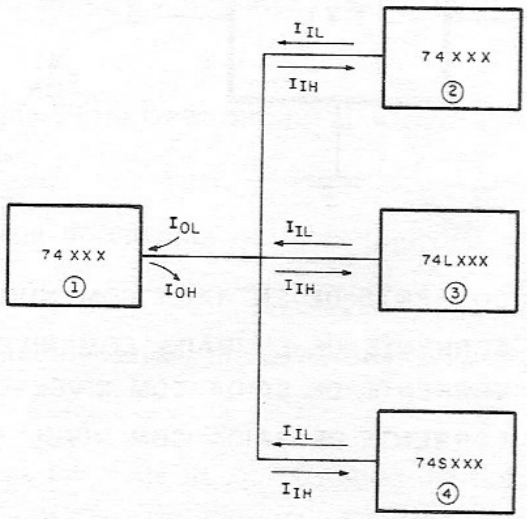


Fig. 4.4 – Método Direto

Quando o circuito 1 fornecer em sua saída um nível lógico 0 teremos:

$$IOL\ 1 = IIL\ MAX2 + IIL\ MAX\ 3 + IIL\ MAX\ 4$$

$$IOL\ 1 = 1{,}6mA + 180\ \mu A + 2mA = 3{,}78mA$$

No caso de termos na saída do circuito 1 um nível lógico 1:

$$IOH\ 1 = IIH\ MAX\ 2 + IIH\ MAX\ 3 + IIH\ MAX\ 4$$

$$IOH\ 1 = 40\ \mu A + 10\ \mu A + 50\ \mu A = 100\ \mu A$$

Portanto, como os valores de IOL e IOH são inferiores ao máximo especificado pelos fabricantes na tabela 4.2, concluímos que o circuito está bem dimensionado.

4.3.2.2 – Método Normalizado

No *método normalizado* operamos com números adimensionais, normalizados em relação aos valores da versão Standard (74XXX). Para tanto, definiremos dois conceitos importantes: *FAN-IN* e *FAN-OUT*.

O *FAN-IN* de um circuito lógico está associado às suas correntes de entrada, e é o máximo entre as seguintes razões: (IIH MAX/IIH MAX STAND) ou (IIL MAX/IIL MAX STAND).

Então, baseando-nos na tabela 4.2 e na definição de FAN-IN, chegamos na tabela 4.3 aos valores dos FAN-IN's para as diversas versões da família TTL.

TABELA 4.3 - FAN-IN DAS VERSÕES DA FAMÍLIA TTL

VERSÃO	FAN-IN
74 XXX	1.00
74 LXXX	0.25
74 SXXX	1.25
74 LSXXX	0.50

Tabela 4.3 – FAN-IN das versões TTL

O *FAN-OUT* está relacionado com as correntes de saída e representa o número máximo de entradas da série Standard que podemos conectar a uma determinada saída de um circuito TTL.

Isto é obtido através do mínimo entre as seguintes razões: (IOH MAX/IIH MAX STAND) ou (IOL MAX/IIL MAX STAND).

Então, baseando-nos na tabela 4.2 e na definição de FAN-OUT, chegamos na tabela 4.4 aos valores dos FAN-OUT's para as diversas versões da família TTL.

TABELA 4.4 - FAN-OUT DAS VERSÕES DA FAMÍLIA TTL

VERSÃO	FAN-OUT
74 XXX	10
74 LXXX	2
74 SXXX	12
74 LSXXX	5

Tabela 4.4 — FAN-OUT das versões TTL

Com o *método normalizado,* na interligação de circuitos TTL podemos garantir o bom funcionamento, no que se refere às correntes de entrada e saída, se realizarmos os seguintes passos básicos durante o projeto:

1 — Identificar os FAN-IN's de cada uma das entradas.

2 — Determinar a soma destes FAN-IN's.

3 — Se a soma dos FAN-IN's for inferior ou igual ao FAN-OUT da saída associada, então a ligação estará bem dimensionada.

O *método normalizado* é mais fácil de trabalhar pois envolve grandezas adimensionais. No entanto, chamamos a atenção do leitor pois este método, em alguns casos, pode fornecer resultados superdimensionados.

4.3.2.3 — Buffers/Drivers

Quando o número de entradas ultrapassa a capacidade da saída, temos duas opções para contornar o problema:

1 — Dispor circuitos em paralelo.

2 — Utilizar circuitos Buffers/Drivers.

Um exemplo de disposição de circuitos em *paralelo* está mostrado na figura 4.5.

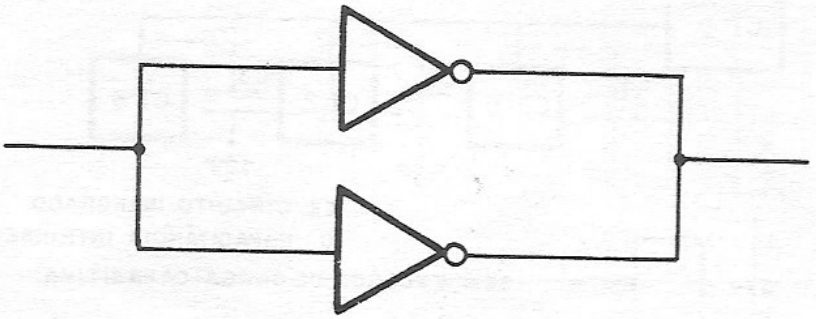


Fig. 4.5 – Inversores em Paralelo

Nesta configuração temos dois circuitos inversores em paralelo. O resultado desta ligação é a duplicação do FAN-IN e do FAN-OUT.

Os circuitos do tipo buffer/driver possuem valores de IOL MAX e IOH MAX mais elevados que os das versões TTL apresentadas. Assim podemos ligar nas suas saídas um número bem maior de entradas TTL's. Nos manuais dos principais fabricantes temos uma enorme variedade de buffers/drivers.

4.3.2.4 – Capacitância de Entrada

Uma outra preocupação no projeto do hardware é a *capacitância* associada a cada entrada de um circuito lógico. Esta capacitância é intrínseca ao processo de integração. Quando temos muitas entradas ligadas a uma determinada saída, a capacitância total é a soma de todas as capacitâncias de entrada, pois trata-se de uma associação em paralelo. Isto causa um aumento nos *tempos de subida e descida* dos sinais lógicos, podendo perturbar o funcionamento do circuito, principalmente em freqüências altas, que é uma característica dos circuitos que utilizam microprocessadores.

Também quando o sinal percorre um caminho muito longo, maior do que um metro, surgem problemas de deformação no sinal, decorrentes da capacitância intrínseca do fio.

Os circuitos do tipo buffer/driver são usados, na maioria dos casos, para resolver estes problemas.

A figura 4.6 ilustra o efeito de cargas capacitivas nos sinais lógicos.

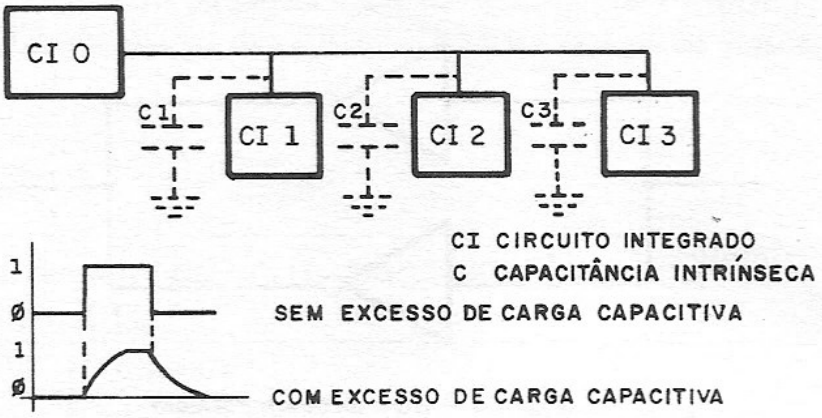


Fig. 4.6 – Efeito de Cargas Capacitivas

4.3.2.5 – Circuitos C-MOS

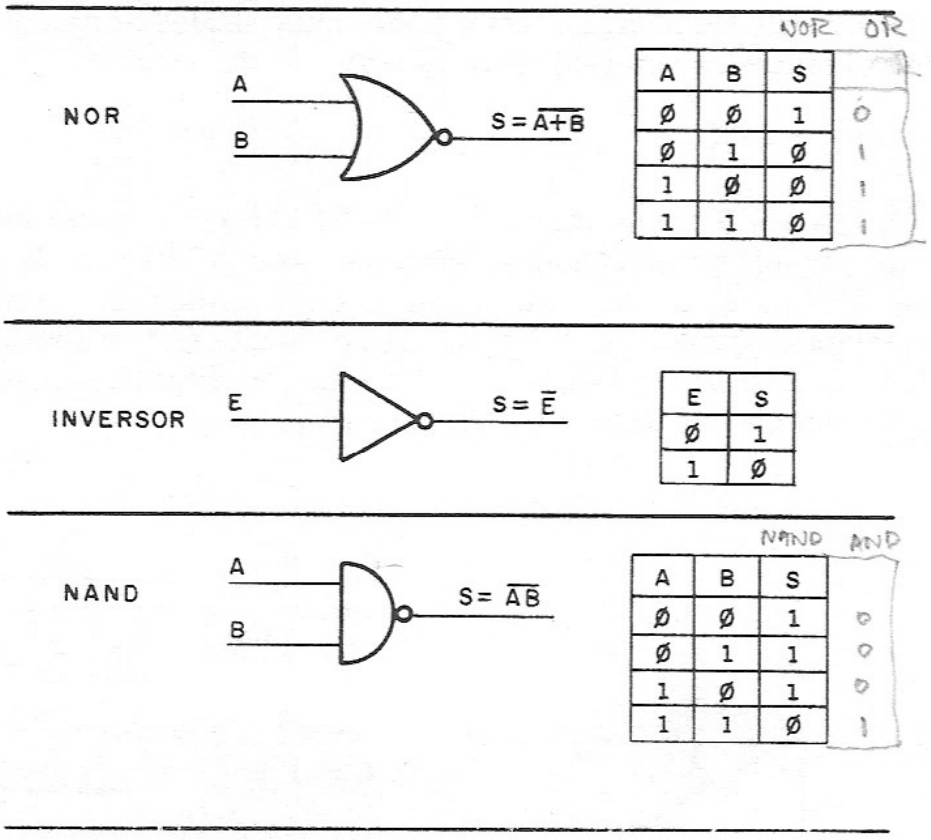
Além dos circuitos integrados TTL, existem circuitos lógicos que usam a tecnologia de integração C-MOS e são alimentados, também, pela tensão de + 5 Volts. O consumo é bem menor e as correntes IIL e IIH máximas são de 0,1 μA. Por outro lado, as correntes IOH e IOL podem assumir um valor típico de 1mA para a *série B*. Assim, devido à alta relação entre as correntes de saída e as de entrada (1mA e 0,1 μ A), uma grande quantidade de entradas C-MOS podem ser conectadas a uma mesma saída. Outra vantagem de tecnologia C-MOS é o seu baixo consumo. No entanto, cuidados devem ser tomados com relação às altas capacitâncias intrínsecas das suas entradas, que ocasionam atrasos bem superiores aos da série TTL.

4.4 – CIRCUITOS LÓGICOS

Vejamos agora alguns circuitos lógicos necessários ao estudo do hardware do Z-80.

4.4.1 – NAND, INVERSOR e NOR

Estas portas lógicas são as mais utilizadas e suas representações e respectivas *tabelas verdades* estão representadas na figura 4.7.



A	B	S
Ø	Ø	1
Ø	1	Ø
1	Ø	Ø
1	1	Ø

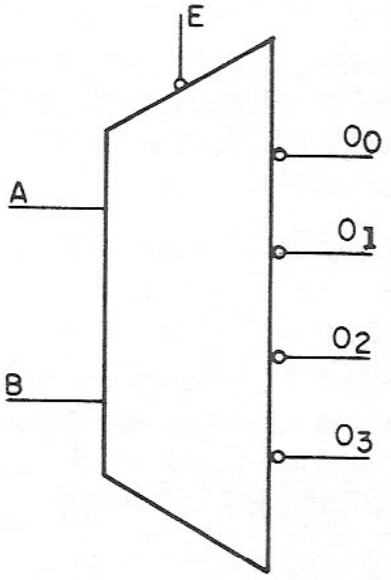
E	S
Ø	1
1	Ø

A	B	S
Ø	Ø	1
Ø	1	1
1	Ø	1
1	1	Ø

Fig. 4.7 – Principais Portas Lógicas

4.4.2 – Decodificador

Um outro circuito lógico importante e muito utilizado na ligação com memória e entrada/saída, conforme veremos mais adiante, é o *decodificador*. A figura 4.8 mostra um decodificador de 2 x 4 e sua tabela verdade.



E	A	B	O_0	O_1	O_2	O_3
1	X	X	1	1	1	1
Ø	Ø	Ø	Ø	1	1	1
Ø	Ø	1	1	Ø	1	1
Ø	1	Ø	1	1	Ø	1
Ø	1	1	1	1	1	Ø

Fig. 4.8 – Decodificador 2 x 4

Em função da aplicação, também são muito usados os decodificadores com três (3) entradas e oito (8) saídas (3 x 8).

4.4.3 – TRI-STATE

Na figura 4.9 temos um circuito TRI-STATE que é muito importante no projeto de microcomputadores. Isto porque com o TRI-STATE, podemos conectar e desconectar circuitos, atuando na sua entrada de controle "C". Neste exemplo, quando C = 1 dizemos que o circuito está no *terceiro estado* ou *estado de desconexão*. Quando C = 0, o sinal que chega na entrada é transferido para a saída sem nenhuma alteração lógica.

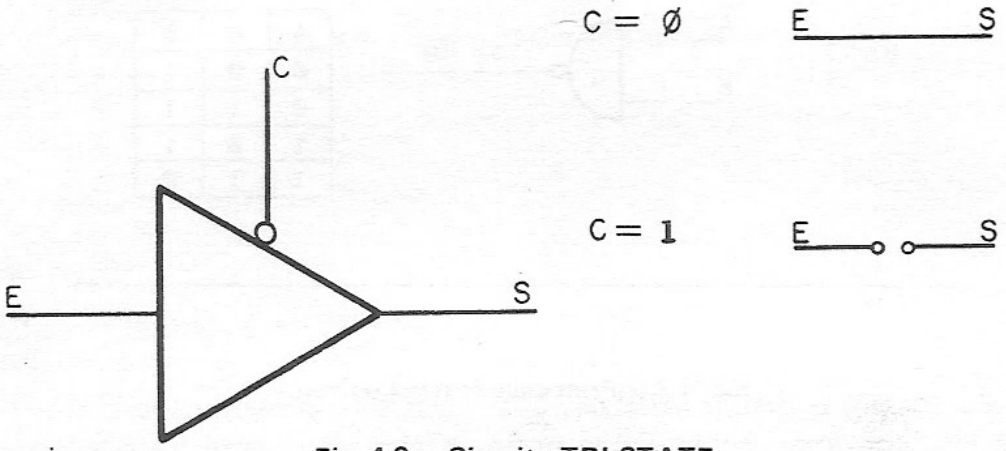


Fig. 4.9 – Circuito TRI-STATE

4.5 – ALIMENTAÇÃO

A alimentação do Z-80 é feita através de dois pinos (11 e 29). Um deles é o zero Volts (terra) e o outro é o + 5 Volts. A tolerância para a tensão de alimentação é de 5%. Isto é, pode situar-se entre 4,75 e 5,25 Volts, e o consumo máximo é de 200mA para temperaturas de 0 a 70°C.

4.6 – CARACTERÍSTICAS ELÉTRICAS

Após termos estudado os conceitos da família TTL, voltamos ao estudo do hardware do Z-80. A figura 4.10 mostra os valores limites das tensões e das correntes IIL, IIH, IOL e IOH para os pinos de entrada e saída do Z-80, em temperaturas de 0 a 70°C e tensão máxima de alimentação (5,25 Volts).

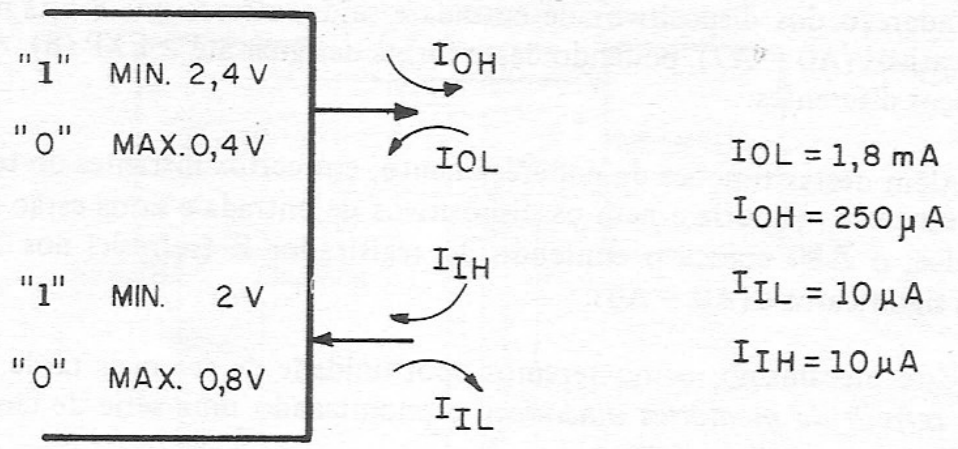


Fig. 4.10 – Correntes e Tensões Limites no Z-80

Comparando estes valores de tensão do Z-80 com os da família TTL, observamos que são compatíveis.

As correntes IIL e IIH são de 10 μA. Assim, representam cargas mínimas para as saídas TTL.

Da tabela 4.2 temos que a corrente IIL MAX de um circuito da série 74XXX é 1,6mA, que é próximo de 1,8mA; valor de IOL MAX do Z-80. Assim, na ligação com o Z-80 devemos utilizar, preferencialmente, circuitos das versões 74LXXX, 74LSXXX ou C-MOS, pois podemos conectar vários destes circuitos a uma única saída do Z-80. Por outro lado, se usarmos um circuito TTL da versão standard, só podemos ligar ao Z-80 uma única entrada.

As saídas do Z-80, segundo o manual da Zilog, podem acionar entradas com cargas capacitivas de até 50pF. Como as entradas TTL possuem capacitância de 2pF, não temos maiores problemas na ligação com circuitos TTL. As entradas do Z-80, com exceção da entrada do relógio (CLK), possuem capacitância de 5pF. Deve-se observar no projeto do relógio, a alta capacitância da entrada (CLK) que é de 35pF.

4.7 – BARRA DE ENDEREÇAMENTO (A0 – A15)

A barra de endereçamento é representada por 16 bits designados por A0, A1, ..., A15. O bit menos significativo é o A0 e o mais significativo é o A15.

Esta barra possui as saídas em tri-state; isto significa que quando não está sendo utilizada pelo Z-80 pode ficar no terceiro estado, que é o de desconexão. Através desta barra o Z-80 fornece o endereço da memória, e seus 16 bits podem identificar até 2 EXP (16) = 65536 posições. Também forne-

ce o endereço dos dispositivos de entrada e saída, através dos 8 bits menos significativos (A0 – A7), podendo desta forma designar até 2 EXP (8) = 256 endereços diferentes.

Além destas funções de endereçamento, em certos instantes do tempo, em que nem a memória e nem os dispositivos de entrada e saída estão sendo acessados, o Z-80 coloca o conteúdo do registrador R (refresh) nos 7 bits menos significativos (A0 – A6).

Este mecanismo, como teremos oportunidade de ver mais tarde, auxilia no *refresh de memórias dinâmicas* economizando uma série de circuitos adicionais.

4.8 – BARRA DE DADOS (D0 – D7)

A barra de dados tem 8 linhas representadas por 8 bits designados por D0, D1, ..., D7. O bit menos significativo é o D0 e o mais significativo é o D7. A barra de dados é *bidirecional* e, também, pode assumir o terceiro estado lógico. Sob controle do Z-80, permite que dados sejam transferidos dos registradores internos para a memória ou dispositivos de entrada e saída e vice-versa.

4.9 – SINAIS DE CONTROLE DAS BARRAS ($\overline{\text{BUSREQ}}$ e $\overline{\text{BUSACK}}$)

Existem duas linhas que atuam junto às barras de endereçamento e de dados; uma é a entrada $\overline{\text{BUSREQ}}$ – bus request/pino 25 – e a outra a saída $\overline{\text{BUSACK}}$ – bus acknowledge/pino 23.

Quando um dispositivo externo necessita controlar as barras para realizar transferências entre memória ou dispositivos de entrada e saída, ele fornece um nível 0 na entrada $\overline{\text{BUSREQ}}$. Logo após, o Z-80 coloca as barras em terceiro estado e sinaliza através da saída $\overline{\text{BUSACK}}$, colocando-a no nível 0. Quando isto acontece, o Z-80 *paralisa o seu processamento,* permitindo que algum dispositivo externo assuma o controle das barras. Por exemplo, uma unidade controladora de disco pode realizar operações de acesso direto à memória (DMA) ou, mesmo, outro microprocessador pode dispor dos recursos das barras.

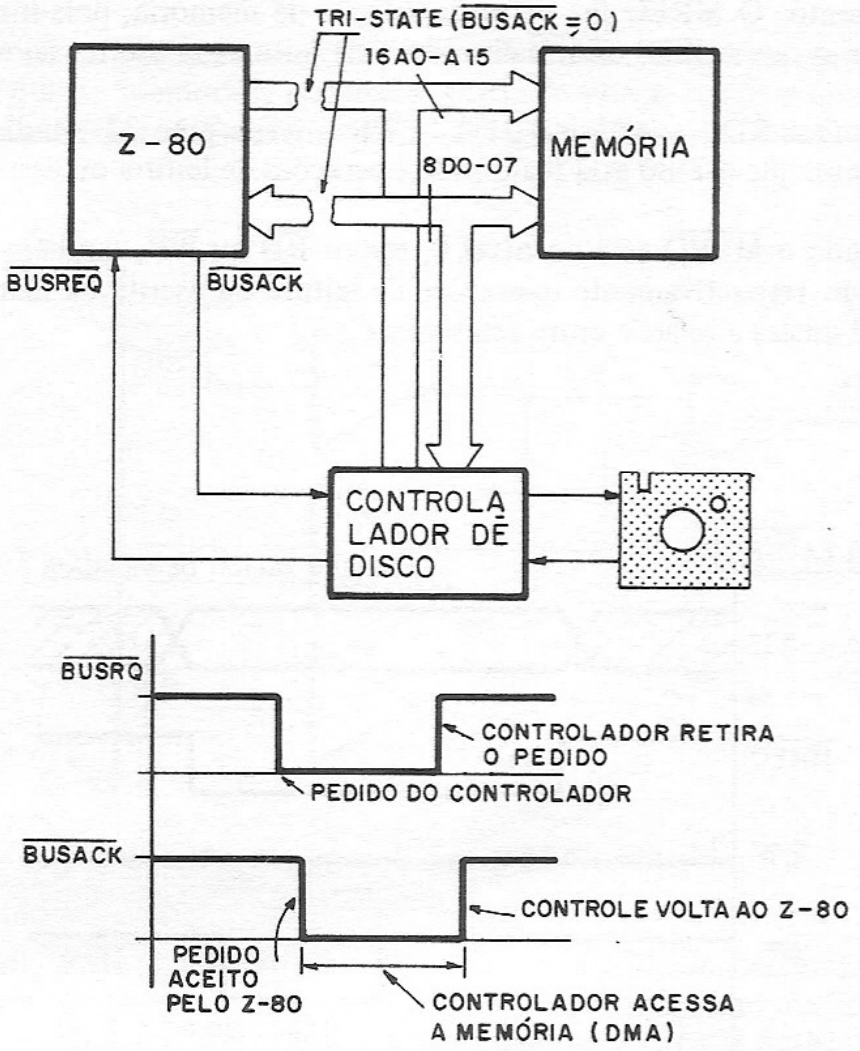


Fig. 4.11 – Acesso Direto à Memória (DMA)

Na figura 4.11, temos um diagrama de blocos de um microcomputador equipado com um controlador de disco. Destacamos as linhas $\overline{\text{BUSREQ}}$ e $\overline{\text{BUSACK}}$, que controlam as barras de dados e de endereçamento.

A razão do uso de DMA com controlador de disco deve-se ao fato do Z-80 não conseguir, em alguns casos, realizar transferências controladas por software. Isto porque, para alguns controladores, a taxa de transferência de dados é superior à velocidade de processamento do Z-80. Assim, o controlador bloqueia o Z-80, que coloca suas barras em terceiro estado, e realiza a troca de informações com a memória sem o auxílio do microprocessador.

4.10 – SINAIS DE CONTROLE DA MEMÓRIA ($\overline{\text{MREQ}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$ e $\overline{\text{RFSH}}$)

O $\overline{\text{MREQ}}$ – Memory request/pino 19 – é uma saída do Z-80 que estando no nível 0 indica que existe um endereço de memória válido na barra de

endereçamento. O $\overline{MREQ}$ faz parte da seleção da memória, pois informa ao meio externo que o Z-80 está realizando uma leitura ou escrita na memória.

As saídas $\overline{RD}$ – read/pino 21 – e $\overline{WR}$ – write/pino 22 – indicam, respectivamente, que o Z-80 está realizando operações de leitura ou escrita.

Quando o $\overline{MREQ}$ está no nível 0, temos $\overline{RD}$ ou $\overline{WR}$, também no nível 0, indicando respectivamente operações de leitura ou escrita na memória. A figura 4.12 ilustra a relação entre estes sinais.

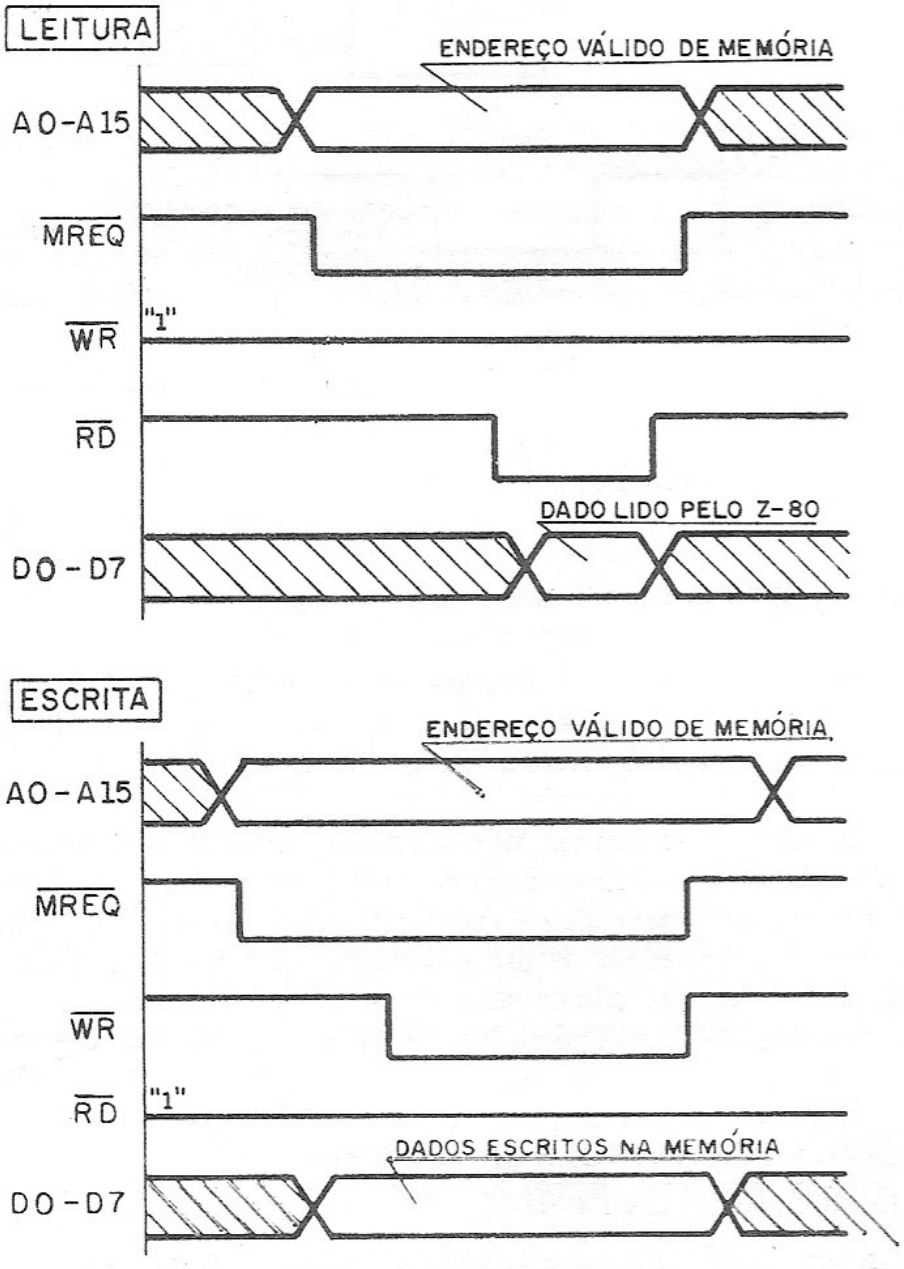


Fig. 4.12 – Relação dos sinais $\overline{MREQ}$, $\overline{RD}$ e $\overline{WR}$

Na figura 4.13 temos a geração dos sinais $\overline{MEMR}$ (leitura de memória) e $\overline{MEMW}$ (escrita na memória), obtidos a partir de portas nands e inversoras. Estes sinais são os que efetivamente comandam os módulos de memória.

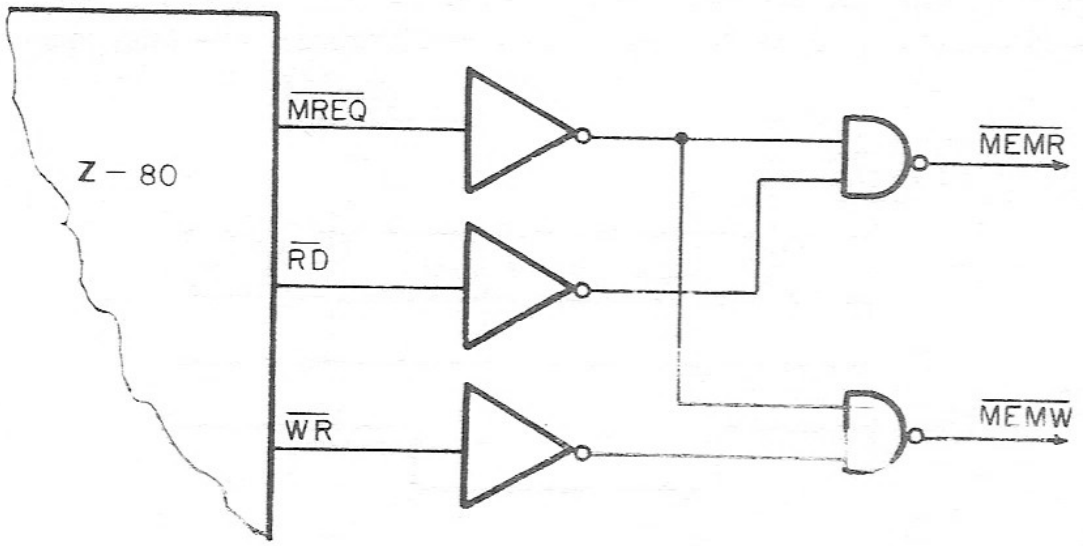


Fig. 4.13 – Geração dos sinais $\overline{MEMR}$ e $\overline{MEMW}$

O sinal $\overline{RFSH}$ – refresh/pino 28 – não está relacionado diretamente com a leitura e escrita de dados na memória. Este sinal é usado somente quando trabalhamos com memórias *RAM's dinâmicas.* Vimos na lição anterior, que este tipo de memória necessita de um *refresh* periódico, de modo a não perder a informação nela armazenada.

Basicamente, o refresh é uma operação de leitura em determinados endereços, sem que haja efetivamente transferência de informação. Para a maioria das memórias dinâmicas, o refresh de cada célula tem que transcorrer em intervalos de tempo inferiores a 2ms.

Normalmente, para implementar o refresh usamos um controlador adicional composto de contadores, portas e outros circuitos lógicos. Existem também no mercado, circuitos LSI (Large Scale Integration) que podem tornar mais econômica tal implementação. No entanto, o Z-80 através da barra de endereçamento, do registrador R e do sinal $\overline{RFSH}$ implementa as funções de refresh, sem a necessidade de controladores externos. Assim, a utilização de memórias dinâmicas fica ainda mais econômica.

Quando o $\overline{RFSH}$ = 0 e o $\overline{MREQ}$ = 0, o conteúdo do registrador R é colocado nos sete bits menos significativos (A0 – A6) da barra de endereçamento. A cada busca de instrução, o conteúdo do registrador R é incrementado de uma unidade. Voltaremos ao refresh quando estudarmos os *ciclos do Z-80.*

4.11 – SINAIS DE ENTRADA E SAÍDA ($\overline{I/ORQ}$, $\overline{RD}$ e $\overline{WR}$)

Quando $\overline{I/ORQ}$ = 0 (pino 20) significa que o Z-80 está realizando uma operação de entrada ou saída (E/S). O endereço da porta de E/S é fornecido pelos 8 bits menos significativos (A0 – A7) da barra de endereçamento. Os sinais $\overline{RD}$ e $\overline{WR}$, quando estão no nível 0 indicam, respectivamente, operações de entrada ou saída. Na figura 4.14 temos a relação entre estes sinais.

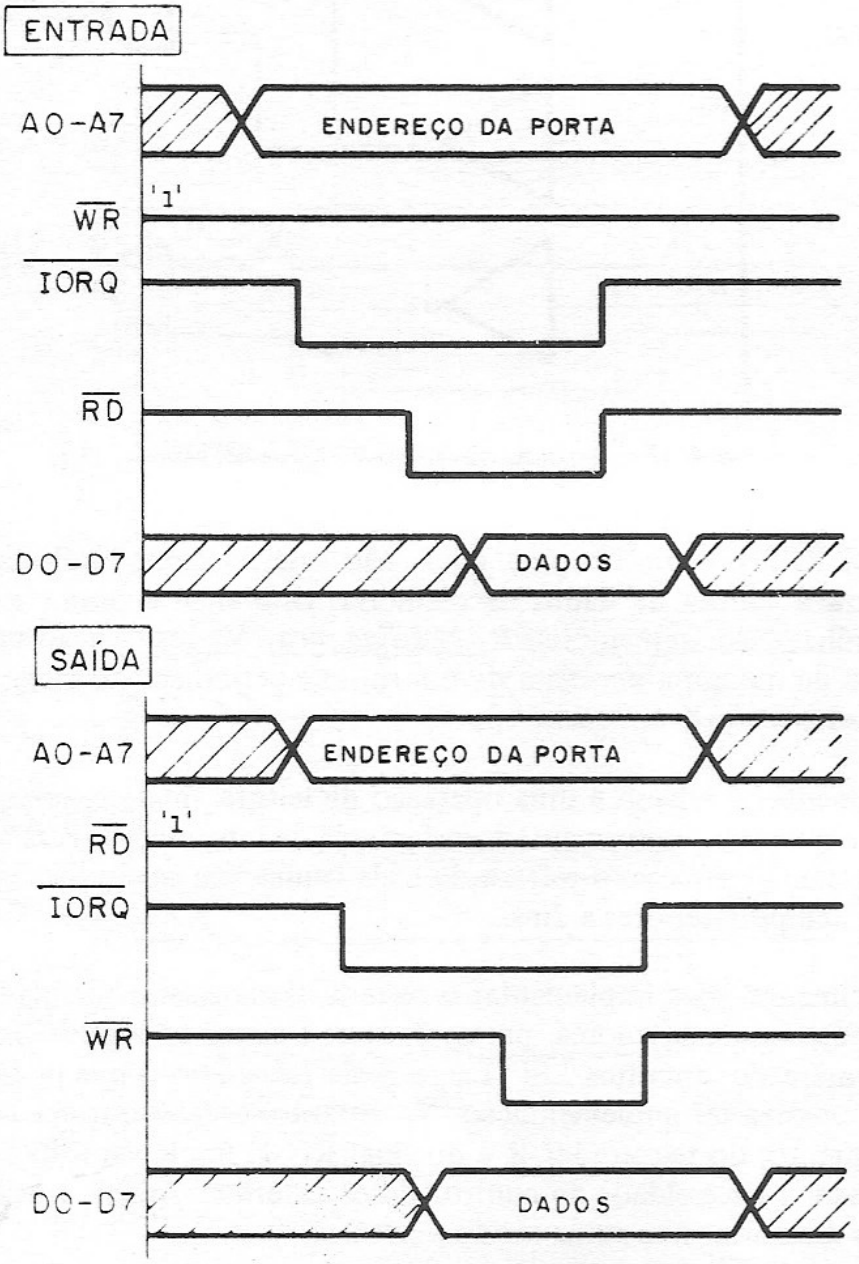


Fig. 4.14 – Relação dos sinais $\overline{I/ORQ}$, $\overline{RD}$ e $\overline{WR}$

De maneira semelhante aos sinais $\overline{MEMR}$ e $\overline{MEMW}$, temos na figura 4.15 a geração dos sinais $\overline{I/OR}$ (leitura do periférico) e $\overline{I/OW}$ (escrita no periférico). Estes sinais são os que efetivamente comandam os dispositivos de E/S.

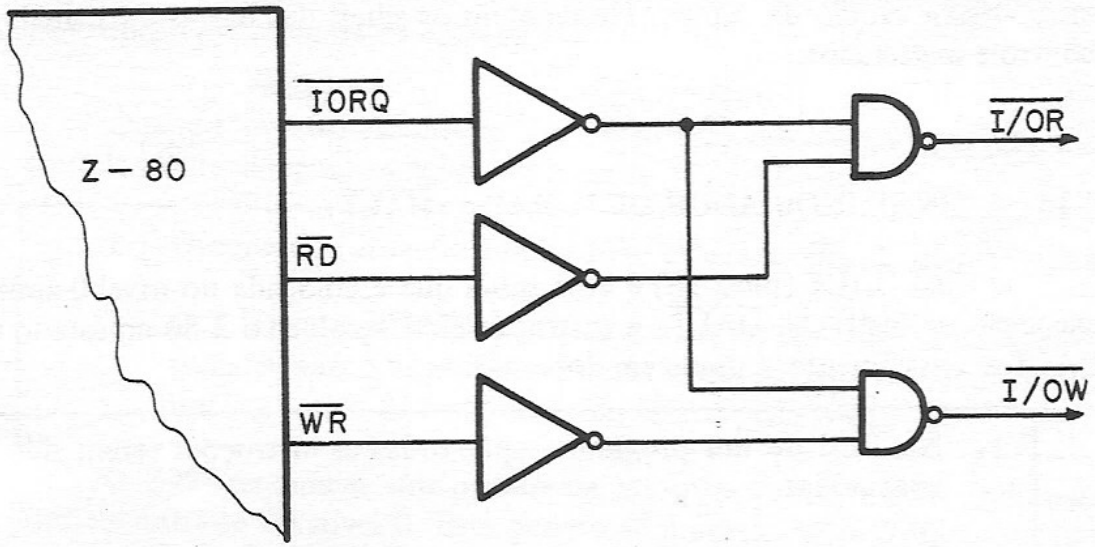


Fig. 4.15 – Geração dos sinais $\overline{I/OR}$ e $\overline{I/OW}$

4.12 – SINAL DE RESET ($\overline{RESET}$)

O sinal $\overline{RESET}$ (pino 26) é uma entrada usada para inicializar o Z-80. Deve ser ativado imediatamente após o acionamento da alimentação, o *"power-up"*, ou a qualquer instante que se deseje reinicializar o Z-80. Quando o $\overline{RESET}$ vai para o nível 0 ocorrem os seguintes eventos:

1 – As barras de endereçamento e de dados vão para o terceiro-estado.

2 – Todos os sinais de controle ficam no estado inativo (nível lógico I).

3 – Os registradores R e I ficam com o valor zero.

4 – O modo de interrupção é colocado em zero (IMO).

5 – As interrupções provenientes da entrada $\overline{INT}$ são inibidas ("disabled").

6 – O contador do programa (PC) fica com o valor 0000H (hexadecimal).

4.13 – SINAL DE PEDIDO DE ESPERA ($\overline{WAIT}$)

O sinal $\overline{WAIT}$ (pino 24) tem o objetivo de sincronizar com o Z-80, memórias e dispositivos de E/S lentos. O Z-80 testa periodicamente esta entrada, e enquanto ela for mantida no nível 0, ele fica "parado" aguardando que a memória ou o dispositivo de E/S responda a sua solicitação de leitura ou escrita. Neste estado de espera, ele mantém os sinais das barras e os sinais de controle inalterados.

4.14 – SINAL INDICADOR DE PARADA ($\overline{HALT}$)

O sinal $\overline{HALT}$ (pino 18) é uma saída que é colocada no nível 0 após a execução da instrução $\overline{HALT}$. A instrução HALT coloca o Z-80 no estado de HALT e, basicamente, é usada em dois casos:

1 – No final de um programa, após todas as instruções terem sido executadas.

2 – Quando é necessário permanecer com o Z-80 "parado", aguardando um pedido de interrupção.

O estado de HALT caracteriza-se pela paralisação do processamento. A saída deste estado só é possível através de uma interrupção ou ativando-se a linha $\overline{RESET}$.

4.15 – SINAL $\overline{M1}$

O $\overline{M1}$ (pino 27) é uma saída do Z-80 que quando vai para o nível 0, indica que está sendo executado um *ciclo de busca de instrução*. Toda instrução ao ser executada exige que o Z-80, primeiramente, realize a busca do seu código de operação que está armazenado na memória. Em seguida, o Z-80 deposita este código no registrador de instrução para então interpretá-lo.

Voltaremos a estudar o sinal M1, quando conheceremos a relação do relógio (CLK) com as linhas e barramentos do Z-80.

4.16 – SINAIS DE INTERRUPÇÃO ($\overline{NMI}$ e $\overline{INT}$)

O Z-80 possui duas entradas que são identificadas por $\overline{NMI}$ e $\overline{INT}$, através das quais podem ser feitos pedidos de interrupção.

A *entrada* $\overline{NM1}$ (pino 17), *"Non-Maskable Interrupt"*, reconhece o pedido de interrupção quando ocorre uma transição negativa, isto é, transição do nível 1 para o nível 0. Quando a entrada $\overline{NM1}$ vai para o nível 0, o Z-80 reconhece, compulsoriamente, o pedido de interrupção no final da instrução em execução. Com exceção das entradas $\overline{BUSREQ}$ e $\overline{WAIT}$, não existe outro mecanismo de hardware, nem de software, capaz de impedir o atendimento à interrupção proveniente da entrada $\overline{NM1}$.

Quando o Z-80 reconhece um pedido de interrupção proveniente da entrada $\overline{NM1}$, as seguintes ações ocorrem:

1 – O conteúdo atual do contador de programa (PC) é salvo na pilha.

2 – O Z-80 carrega 0066H (hexadecimal) no contador de programa, transferindo o controle para este endereço, onde, efetivamente, inicia a rotina de tratamento da interrupção $\overline{NM1}$.

O Z-80 reconhece um pedido de interrupção na entrada INT quando esta encontra-se no nível 0. Este pedido só é aceito, se o Z-80 estiver habilitado a reconhecer interrupções (MODO EI – "Enable Interrupt") e se a entrada $\overline{BUSREQ}$ estiver desativada ($\overline{BUSREQ}$ = 1).

Este tipo de interrupção tem três modos de operação: modos 0,1 e 2. Para os modos 0 e 2 é necessário que o circuito externo forneça um código que define o endereço inicial da rotina de tratamento. Para que isto seja possível, o Z-80 ativa a linha $\overline{I/ORQ}$ (= 0) durante o ciclo de busca de instrução ($\overline{M1}$ = 0) seguinte à aceitação do pedido. Assim, a combinação destes dois sinais, conforme mostra a figura 4.16, fornece o sinal de reconhecimento de interrupção $\overline{INTA}$.

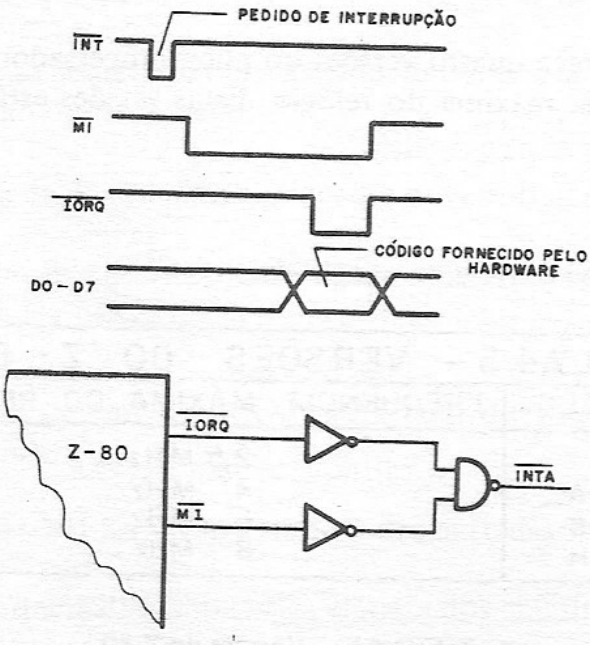


Fig. 4.16 – Sinal $\overline{INT}$ e Geração do Sinal $\overline{INTA}$

4.17 – RELÓGIO DO Z-80

O Z-80 possui uma entrada – CLK/pino 6 – para o relógio de sincronização. Este relógio na maioria das vezes é implementado com um oscilador a cristal. A capacitância da entrada CLK é de 35 pF. É portanto recomendável a colocação de um resistor de *pull up* de 330Ω, para a obtenção de valores menores nos tempos de subida e descida do sinal pulsado, conforme mostra a figura 4.17

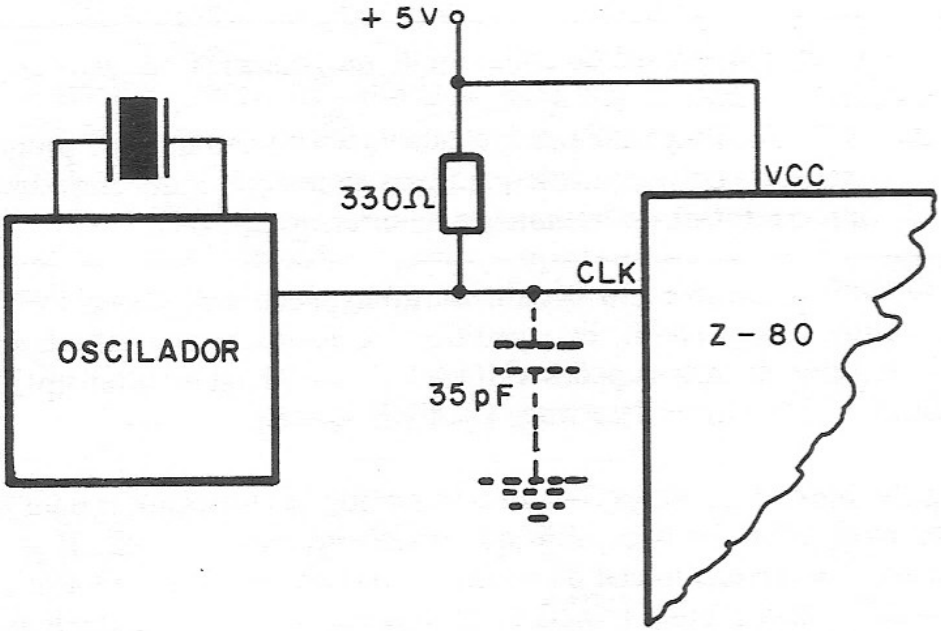


Fig. 4.17 – A Entrada CLK

A Zilog oferece quatro versões do microprocessador Z-80, caracterizadas pela freqüência máxima do relógio. Estas versões estão identificadas na tabela 4.5.

TABELA 4.5 - VERSÕES DO Z-80

MODELO	FREQUÊNCIA MÁXIMA DO RELÓGIO
Z-80	2,5 MHz
Z-80A	4 MHz
Z-80B	6 MHz
Z-80H	8 MHz

Tabela 4.5 – Versões do Z-80

4.18 – EXERCÍCIOS DE FIXAÇÃO

4.18.1 – Porque a versão LOW SCHOTTKY da família TTL é a mais utilizada?

4.18.2 – Quais os valores máximos das correntes IIH, IIL, IOH e IOL da versão standard?

4.18.3 – Explique o método direto de dimensionamento de ligações entre circuitos TTL.

4.18.4 – Utilizando o método direto, determine se os circuitos abaixo estão bem dimensionados:

a) Uma saída de um 74XXX ligada com onze entradas 74XXX.

b) Uma saída de um 74XXX ligada com cinco entradas 74XXX e cinco entradas 74SXXX.

c) Uma saída de um 74LXXX ligada com duas entradas 74XXX e uma entrada 74SXXX.

d) Uma saída de um 74LSXXX ligada a cinco entradas 74XXX e duas 74LSXXX.

e) Uma saída de um 74XXX e outra de um 74LSXXX ligadas com dez entradas 74SXXX.

4.18.5 – Conceitue FAN-OUT e FAN-IN e explique o método normalizado.

4.18.6 – Utilizando o método normalizado, determine se os circuitos abaixo estão bem dimensionados:

a) Uma saída de um 74XXX ligada com nove entradas 74SXXX.

b) Uma saída de um 74LSXXX ligada com duas entradas 74SXXX e duas entradas 74XXX.

c) Uma saída de um 74LXXX ligada com duas entradas 74SXXX e duas 74XXX.

d) Duas saídas 74LSXXX ligadas com dez entradas 74XXX.

4.18.7 – Qual a principal característica elétrica dos circuitos buffers/drivers?

4.18.8 – Qual a influência da capacitância de entrada no projeto elétrico?

4.18.9 – Determine a tabela verdade de um decodificador 3 X 8.

4.18.10 – O que podemos concluir em relação ao número máximo de entradas ligadas ao Z-80, quando comparamos as características elétricas das versões TTL com as características do Z-80?

4.18.11 – Mostre uma aplicação de utilização dos sinais $\overline{\text{BUSREQ}}$ e $\overline{\text{BUSACK}}$, que não seja a da interface com disco.

4.18.12 – Explique o procedimento de "refresh" do Z-80.

4.18.13 – Quais os sinais que indicam ao meio externo que o Z-80 está realizando uma operação de entrada de dados?

4.18.14 – Quais os eventos que ocorrem quando é acionada a entrada $\overline{\text{RESET}}$?

4.18.15 – A entrada $\overline{\text{WAIT}}$ é usada para sincronizar memórias lentas. O que determina o tempo de ativação desta entrada?

4.18.16 – Explique o procedimento de atendimento da interrupção NMI.

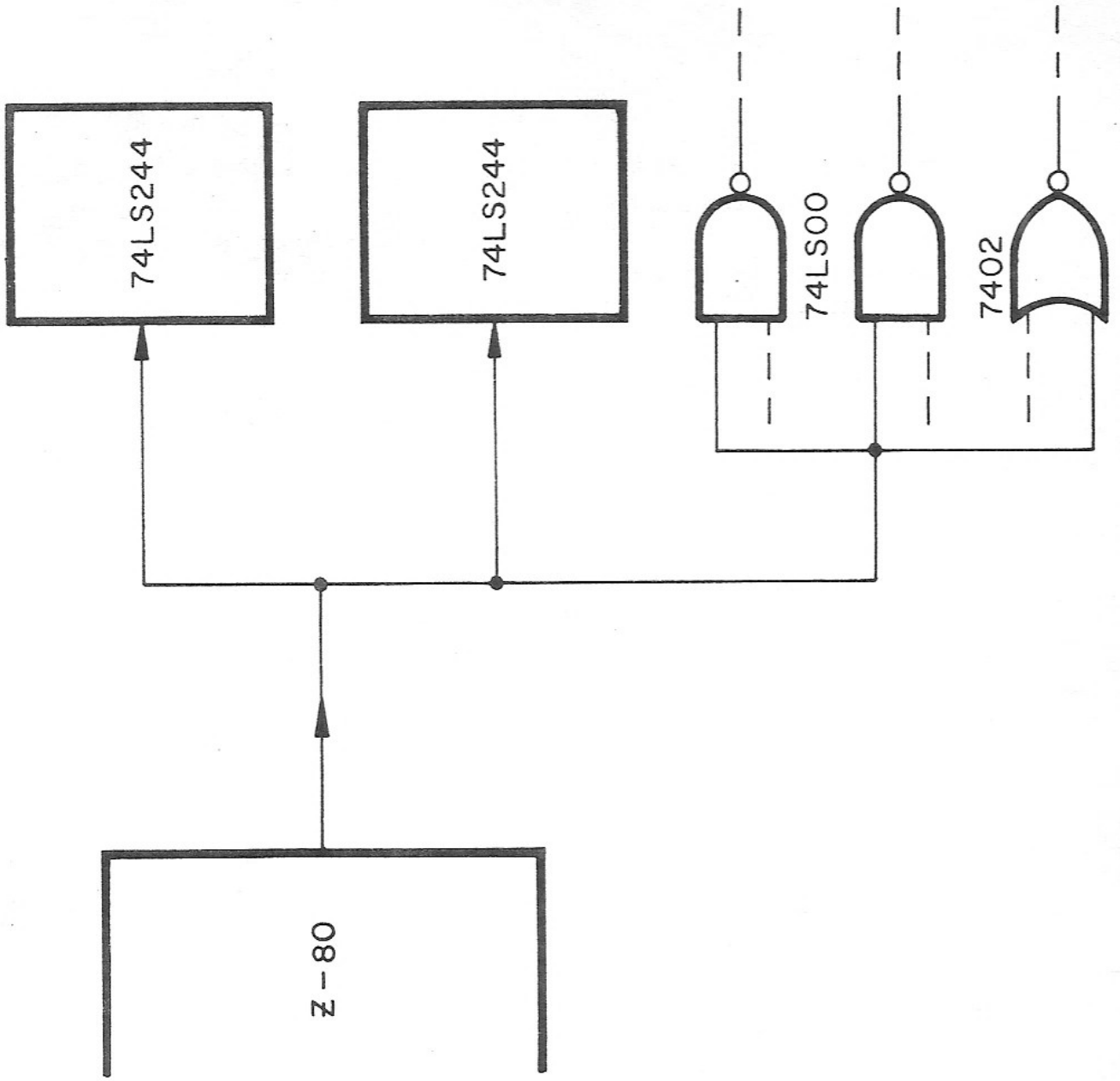
4.18.17 – Qual a função do sinal $\overline{\text{INTA}}$?

4.18.18 – O que acontece com o "refresh" de memória quando é realizado um DMA ou um RESET?

4.18.19 – Classifique o circuito integrado 74LS244, sabendo-se que tem as seguintes características:

IIL MAX = 200 μ A	IOL MAX = 24mA
IIH MAX = 20 μ A	IOH MAX = 15mA

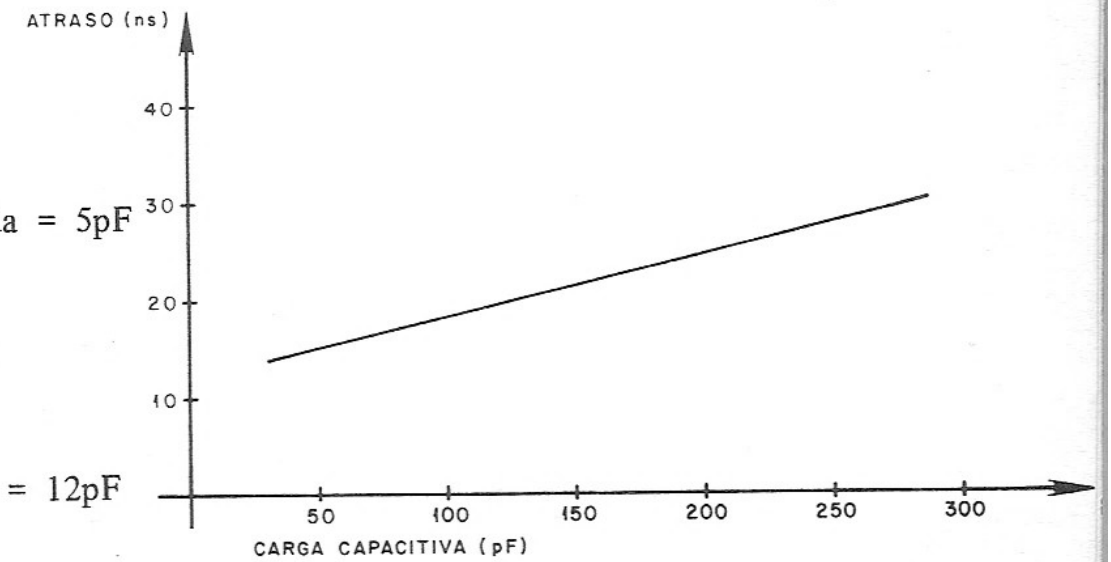
4.18.20 – Determine se a conexão a seguir está bem dimensionada. Para o 74LS244 siga as características do exercício anterior e, para os demais circuitos interligados veja a tabela 4.2.



4.18.21 – Dada a curva de atraso X carga capacitativa do buffer 8212, determine quantos circuitos 74LS244 podemos ligar em suas saídas, admitindo-se um atraso máximo do sinal através do 8212 de 25 ns.

74LS244
Capacitância de entrada = 5pF
IIH MAX = 20 μA
IIL MAX = 200 μA

8212
Capacitância de saída = 12pF
IOL MAX = 15mA
IOH MAX = 1mA



4.19.22 – Projetar um circuito que forneça na sua saída o nível "0", somente quando receber como entrada A15 = 0; $\overline{IORQ}$ = 0 e $\overline{RD}$ = 0.

4.19.23 – Porque é recomendável a colocação de um resistor de pull up na entrada CLK do Z-80?

5 O FUNCIONAMENTO

5.1 – APRESENTAÇÃO

No capítulo anterior estudamos as características elétricas e a compatibilidade do Z-80 com as versões da família TTL. Em adição, apresentamos conceitualmente cada um dos sinais do Z-80, dando ênfase à utilização e aplicação de cada um.

Agora, estudaremos os estados e os diversos ciclos de máquina do Z-80, apresentando o diagrama de tempo de cada um destes ciclos.

As medidas dos tempos envolvidos em cada um dos ciclos encontram-se no APÊNDICE A.

5.2 – CONCEITOS BÁSICOS

O primeiro conceito importante é o *estado (T)*, que é a unidade básica de tempo, sendo igual a um período do relógio. Na versão Z-80-A, o estado possui o valor mínimo de 250 ns, que é o período correspondente à freqüência de 4MHz. Um estado também é chamado por muitos autores de *T-ciclo*. A figura 5.1 mostra dois estados.

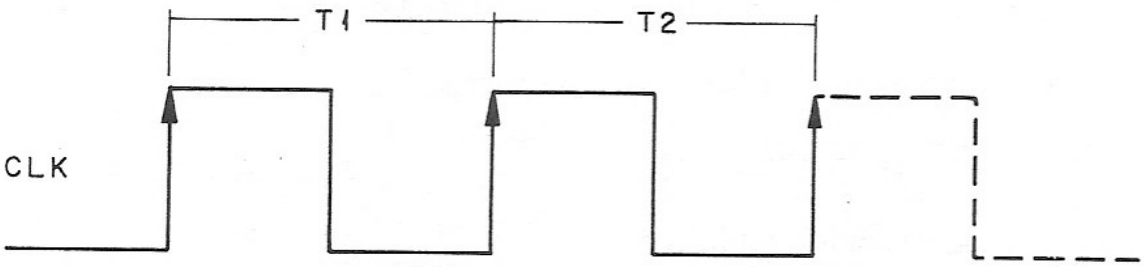


Fig. 5.1 – Os Estados T1 e T2

O *ciclo de instrução* é o tempo necessário para a busca e execução de uma instrução. Todas as instruções executadas pelo Z-80 têm sempre a duração de um número inteiro de estados e, dependendo da instrução, possuem de 4 a 23 estados, que na versão de 4MHz corresponde à faixa de 1 a 6,75 μs.

Um ciclo de instrução possui de 1 a 6 *ciclos de máquina* ou, como são chamados por alguns autores, *M-ciclos*. Basicamente, toda vez que o Z-80 necessita executar um acesso à memória ou a um dispositivo de E/S é necessário um ciclo de máquina. Cada ciclo de máquina possui de 3 a 6 estados. Na figura 5.2 temos um resumo destes conceitos.

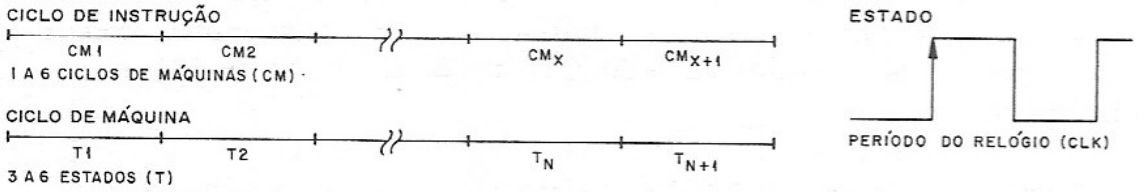


Fig. 5.2 – Estados e Ciclos

Adiantamos ao leitor, que o número de ciclos de máquina não é fixo para as instruções de pesquisa e movimentação de blocos. E, existem casos excepcionais em que ciclos de máquina são usados na consecução de operações internas ao Z-80, por exemplo, a instrução ADD HL, BC.

Existem 7 tipos de ciclos de máquina básicos que podem ocorrer durante a operação do Z-80:

1 – Ciclo de busca de instruções (M1).

2 – Ciclo de leitura e escrita de dados na memória.

3 – Ciclo de leitura e escrita em dispositivos de E/S.

4 – Ciclo de pedido de controle das barras.

5 – Ciclo de interrupção INT.

6 – Ciclo de interrupção NM1.

7 – Ciclo de escape da instrução HALT.

5.3 – CICLO DE BUSCA DE INSTRUÇÃO (M1)

Na execução de qualquer instrução existe um *ciclo de busca do código de operação* que se encontra armazenado na memória. Algumas instruções do

Z-80 possuem dois bytes para o código de operação, e nestes casos temos dois ciclos de busca do código de operação.

O ciclo de busca de instrução, por ser o primeiro ciclo de máquina de um ciclo de instrução, é conhecido como *ciclo M1*.

Um ciclo M1 realiza, basicamente, três tarefas:

1 – Realiza a leitura na memória do código de operação, colocando-o no registrador de instrução do Z-80.

2 – Decodifica o código de operação, identificando a instrução a ser executada.

3 – Executa a instrução, se esta não necessitar de outros ciclos de máquina.

Na figura 5.3 temos o diagrama correspondente à execução da instrução INC A, que é uma instrução que possui somente o ciclo M1, que é composto de 4 estados (T).

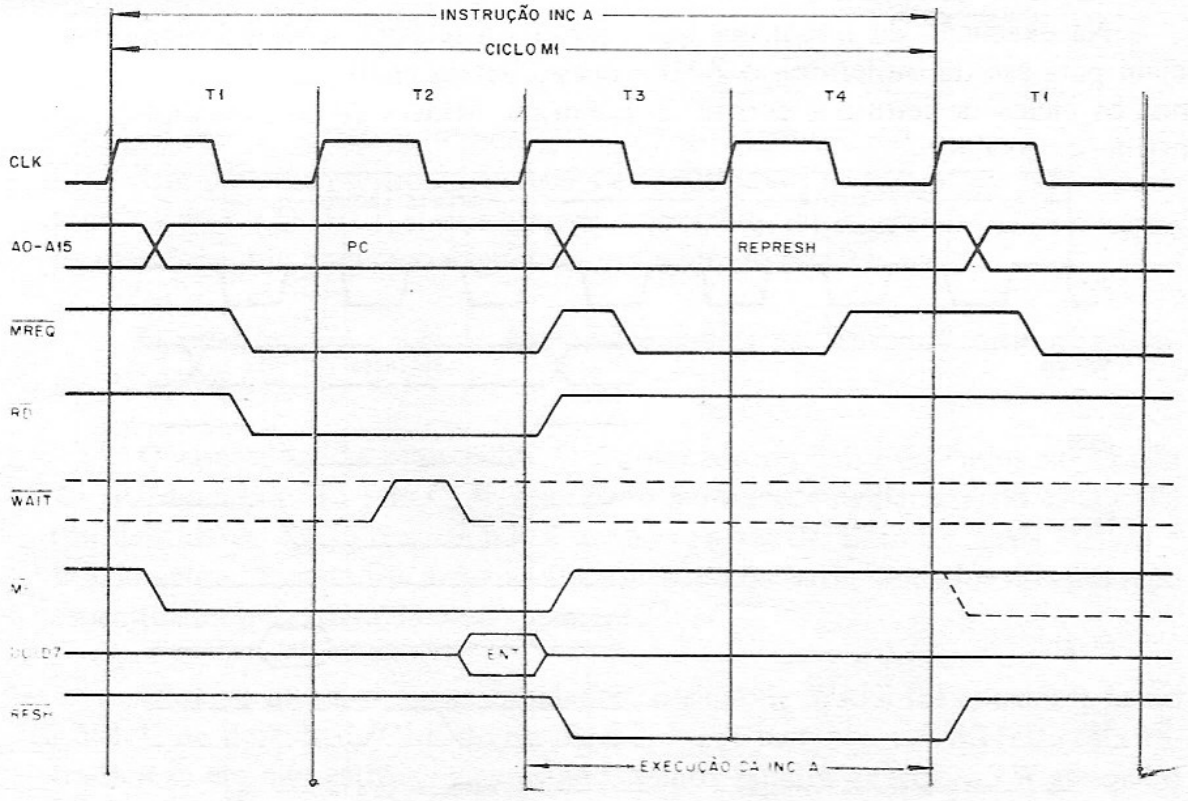


Fig. 5.3 – CICLO M1

No início do estado T1, o sinal $\overline{M1}$ vai para o nível 0, indicando ao meio externo que está começando um ciclo M1. O conteúdo do PC (Program Counter) é colocado na barra de endereçamento, e contém o endereço do código de operação da instrução a ser apanhada. No meio de T1, isto é, na descida (↘) de CLK, os sinais MREQ e RD vão para o nível 0, indicando que o Z-80 já colocou o endereço na barra e vai realizar uma leitura de memória.

A memória precisa fornecer o conteúdo (código de operação) da posição endereçada pelo Z-80, no máximo até o início do estado T3, que é caracterizado pela subida (↗) do sinal CLK. Neste instante (início de T3), o conteúdo da barra de dados, que deve ser igual ao código de operação da instrução, é colocado no registrador de instrução do Z-80. Logo após este evento, os sinais $\overline{M1}$, $\overline{RD}$ e $\overline{MREQ}$ voltam para o nível lógico 1, que é o estado inativo destes sinais.

Os estados T3 e T4 são usados pelo Z-80 para atuar no *refresh* de memórias dinâmicas, decodificar e executar a instrução INC A. Através do $\overline{MREQ}$, que é novamente ativado (nível 0), e do $\overline{RFSH}$, o Z-80 informa que o conteúdo do registrador R encontra-se na barra de endereçamento. Observe que nestes estados (T3 e T4), o Z-80, simultaneamente, realiza operações internas (decodificação e execução da instrução) e utiliza a barra de endereçamento para realizar o *refresh* de memórias dinâmicas.

5.4 – CICLOS DE LEITURA E ESCRITA NA MEMÓRIA

Na execução de instruções que necessitam realizar acessos à memória, sejam para escrita ou leitura, o Z-80 processa outros ciclos de máquina. Vejamos os ciclos de leitura e escrita na memória, através de dois exemplos de instruções do Z-80.

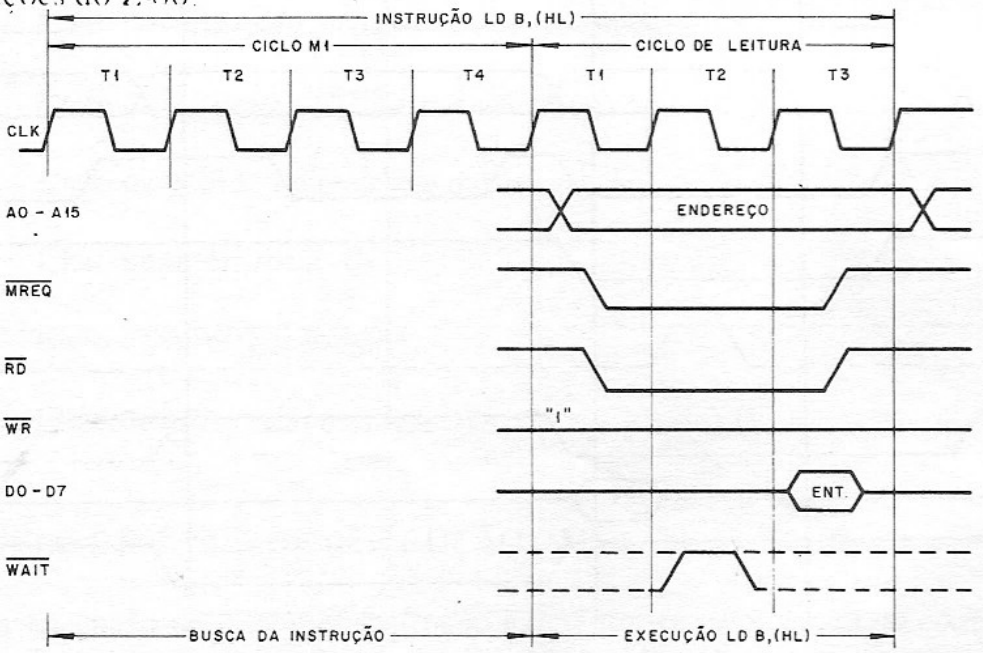


Fig. 5.4 – Ciclo de Leitura

Na figura 5.4, temos o ciclo da instrução LD B, (HL) que carrega o registrador B com o conteúdo da posição de memória cujo endereço é igual ao conteúdo do par HL. O ciclo M1 é idêntico ao anteriormente estudado, e no final deste ciclo o Z-80 decodifica o código de operação e inicia a leitura do operando de 8 bits que se encontra na memória. Novamente, como no ciclo M1, a barra de endereçamento e os sinais $\overline{MREQ}$ e $\overline{RD}$ são ativados no estado T1. Assim, na barra de endereçamento temos o conteúdo do par HL e a memória deve fornecer, via barra de dados, o operando a ser carregado no registrador B. Isto é feito no meio do estado T3, isto é, na descida (↘) do sinal CLK.

Observe que este ciclo de instrução possui um total de sete estados, sendo quatro estados no ciclo M1 e três no ciclo de leitura de memória.

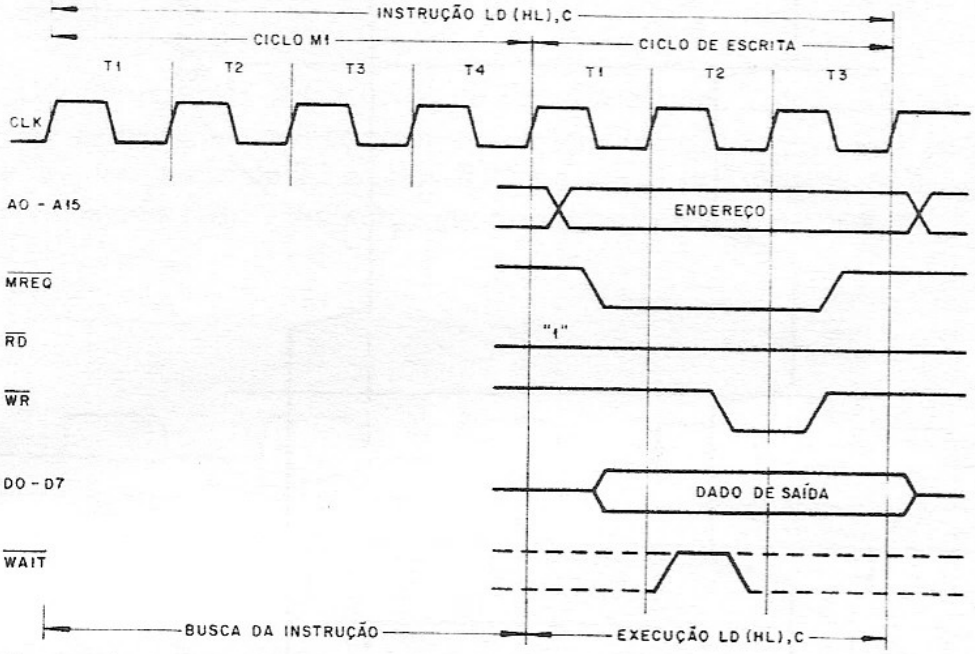


Fig. 5.5 — Ciclo de Escrita

Um ciclo de escrita na memória é apresentado na figura 5.5. Neste caso, a instrução é LD (HL), C que carrega o conteúdo do registrador C na posição de memória cujo endereço é igual ao conteúdo do par HL.

O sinal $\overline{MREQ}$ e a barra de endereçamento são ativados como no exemplo anterior.

O conteúdo do registrador C é colocado na barra de dados no estado T1, na descida (↘) do CLK. Este dado permanece na barra e, no estado T2 (na descida de CLK) o sinal $\overline{WR}$ é ativado (nível 0). Com os sinais $\overline{MREQ}$ e $\overline{WR}$ ativados, a memória armazena o conteúdo da barra de dados no endereço especificado pela barra de endereçamento.

Observe que nos ciclos estudados, o sinal de $\overline{WAIT}$ foi considerado no nível 1, na descida de CLK do estado T2. Neste instante, o Z-80 testa esta entrada e se ela não estiver neste estado, temos estados de espera TW até que o sinal de $\overline{WAIT}$ volte para o nível 1. Isto pode ser visto na figura 5.6.

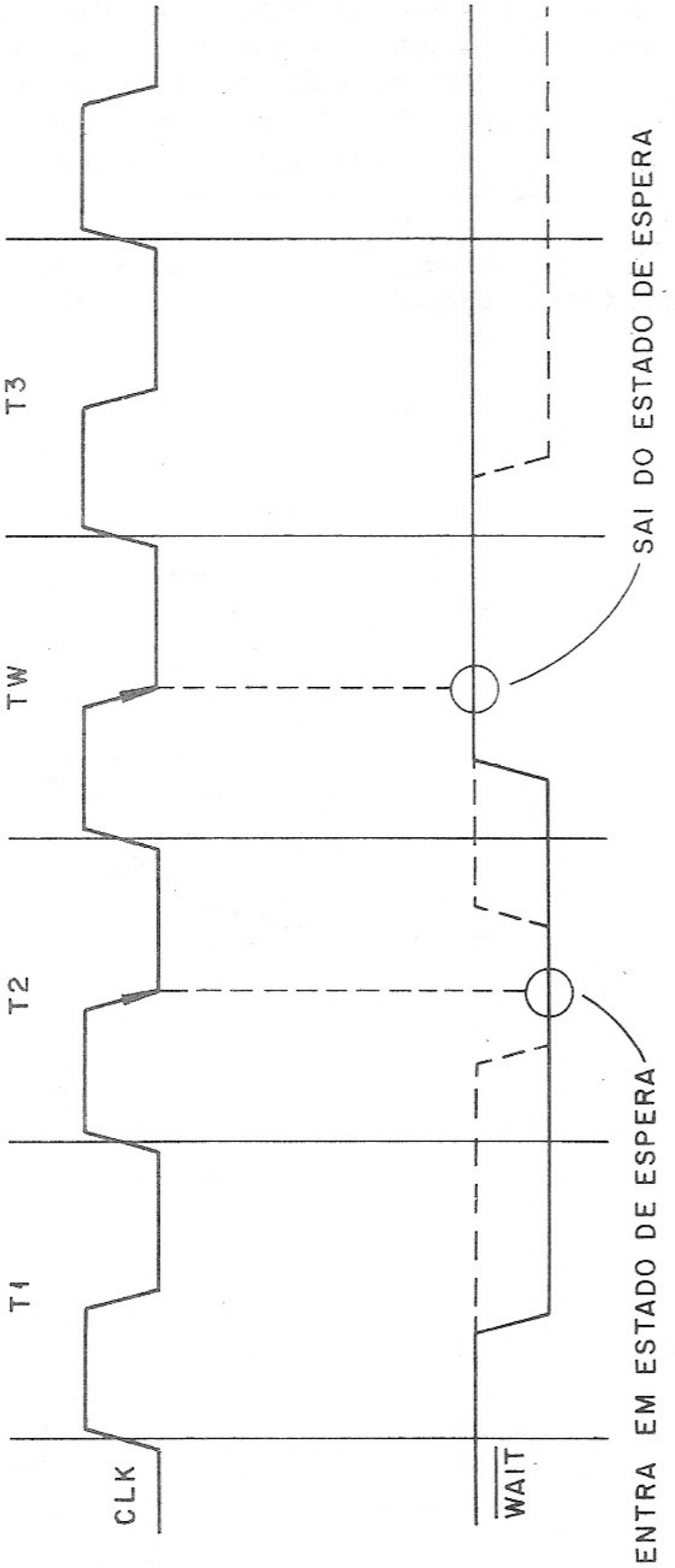


Fig. 5.6 – O Estado TW

5.5 – CICLOS DE LEITURA E ESCRITA EM DISPOSITIVOS DE E/S

Um ciclo de entrada de dados ocorre quando o Z-80 executa uma instrução de leitura de dados de um dispositivo de entrada. Um ciclo de saída de dados ocorre quando o Z-80 executa uma instrução de escrita de dados em um dispositivo de saída. As instruções de entrada e saída de dados, normalmente, possuem de três a quatro ciclos de máquina, com um total de 10 a 20 estados (T). As instruções de entrada e saída de blocos de dados (INIR, INDR, OTIR, OTDR), que transferem até 256 bytes, executam repetidos ciclos de máquina até que todos os bytes sejam transferidos. Assim, o tempo total de execução destas instruções é função do número de bytes a serem transferidos e, também, da velocidade dos periféricos de entrada/saída.

Nas figuras 5.7 e 5.8 temos, respectivamente, os ciclos de entrada e saída de dados.

O endereço dos dispositivos de entrada e saída é colocado nas linhas A0 – A7 da barra de endereçamento no início do ciclo de máquina, isto é, no estado T1. No início de T2 o sinal IORQ é ativado ($\overline{IORQ} = 0$), indicando que teremos uma leitura ou escrita em dispositivo de entrada/saída.

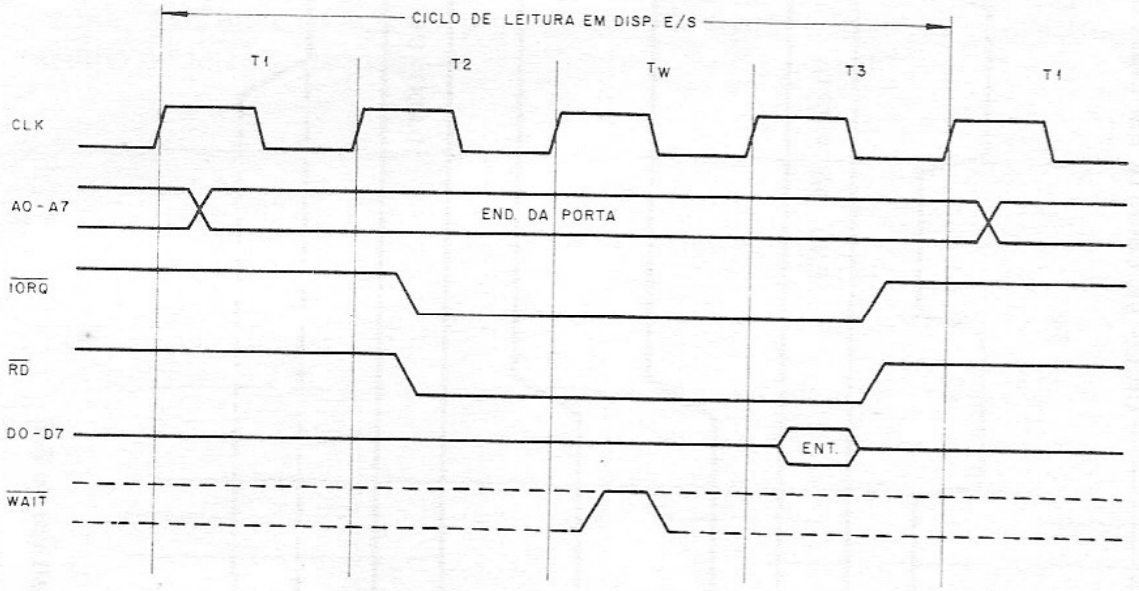


Fig. 5.7 – Ciclo de Leitura em Dispositivos de E/S

Se tivermos uma leitura de periférico, figura 5.7, o sinal $\overline{RD}$ será ativado ao mesmo tempo que o sinal $\overline{IORQ}$. O dispositivo externo reconhece o pedido de leitura através dos sinais $\overline{IORQ}$ e $\overline{RD}$, quando estes estão simultaneamente no nível 0 e, em seguida, coloca a informação na barra de dados. No estado T3, na descida de CLK, o Z-80 realiza a leitura desta informação.

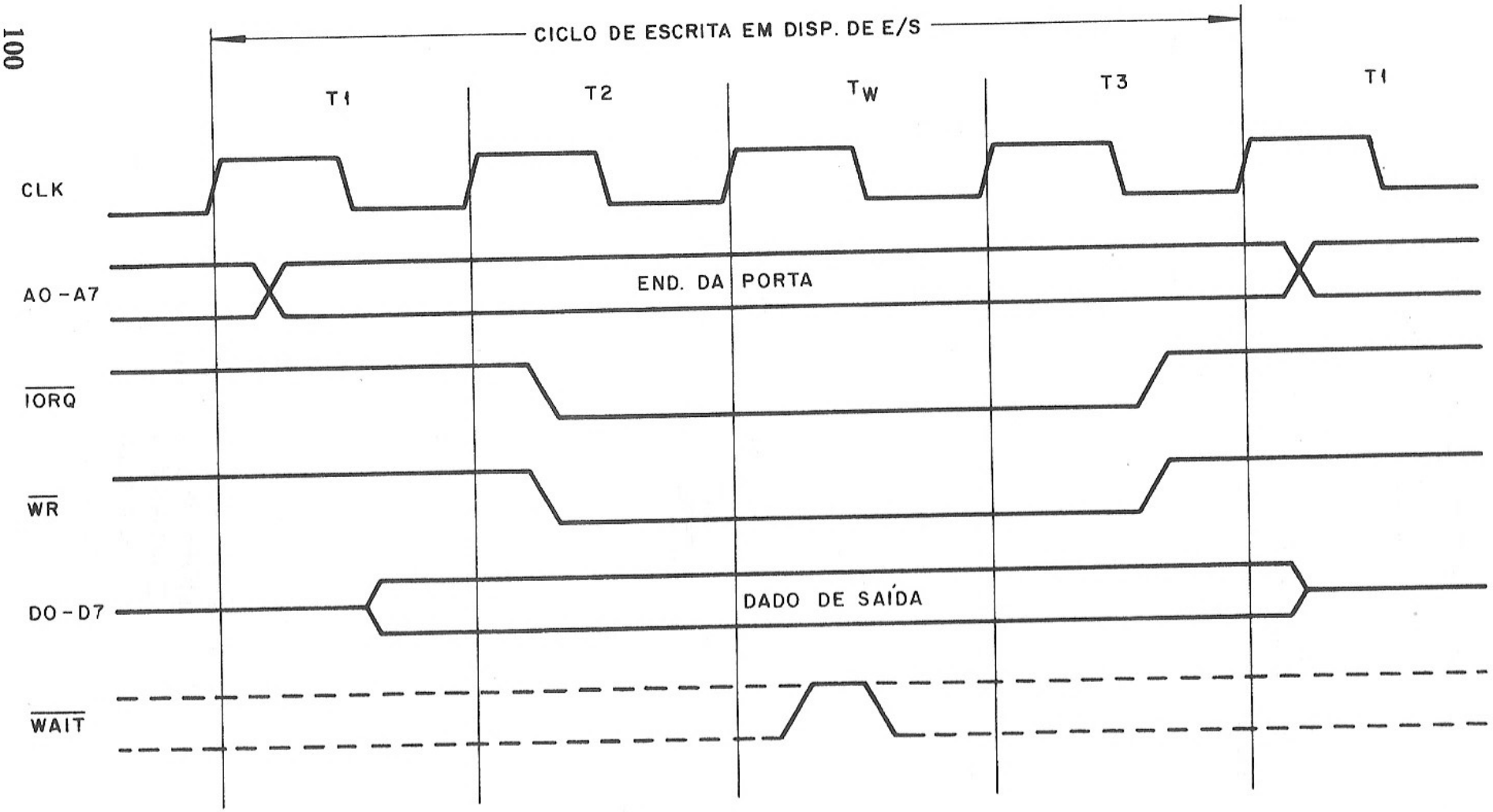


Fig. 5.8 — Ciclo de Escrita em Dispositivos de E/S

Quando o ciclo for de saída de dados, figura 5.8, o sinal $\overline{WR}$ é ativado ($\overline{WR}$ = 0) ao mesmo tempo que o sinal $\overline{IORQ}$. Observe que já no estado T1, o Z-80 coloca na barra de dados o conteúdo a ser escrito no dispositivo de saída. Este dado está disponível até o fim do estado T3, e deve ser capturado pelo dispositivo de saída.

Em ambos os ciclos de entrada e saída de dados, temos um estado TW adicional se a entrada $\overline{WAIT}$ estiver no nível 1. Se tivermos um periférico lento, podemos manter a entrada $\overline{WAIT}$ no nível 0, por um tempo suficiente que permita a resposta deste periférico. Isto é possível, pois enquanto a entrada $\overline{WAIT}$ estiver no nível 0, o Z-80 continua no estado de espera.

5.6 – CICLO DE PEDIDO DE CONTROLE DE BARRAS

Em qualquer instante, um dispositivo externo pode solicitar o controle da barra de endereçamento A0 – A15, da barra de dados D0 – D7 e dos sinais $\overline{MREQ}$, $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$ e $\overline{RFSH}$; ativando a entrada $\overline{BUSREQ}$. Isto é necessário quando desejamos transferir dados entre a memória e dispositivos rápidos de entrada e saída, ou entre posições diferentes dentro da própria memória, sem interferência do processador. Na figura 5.9 temos o diagrama de tempo deste ciclo de máquina.

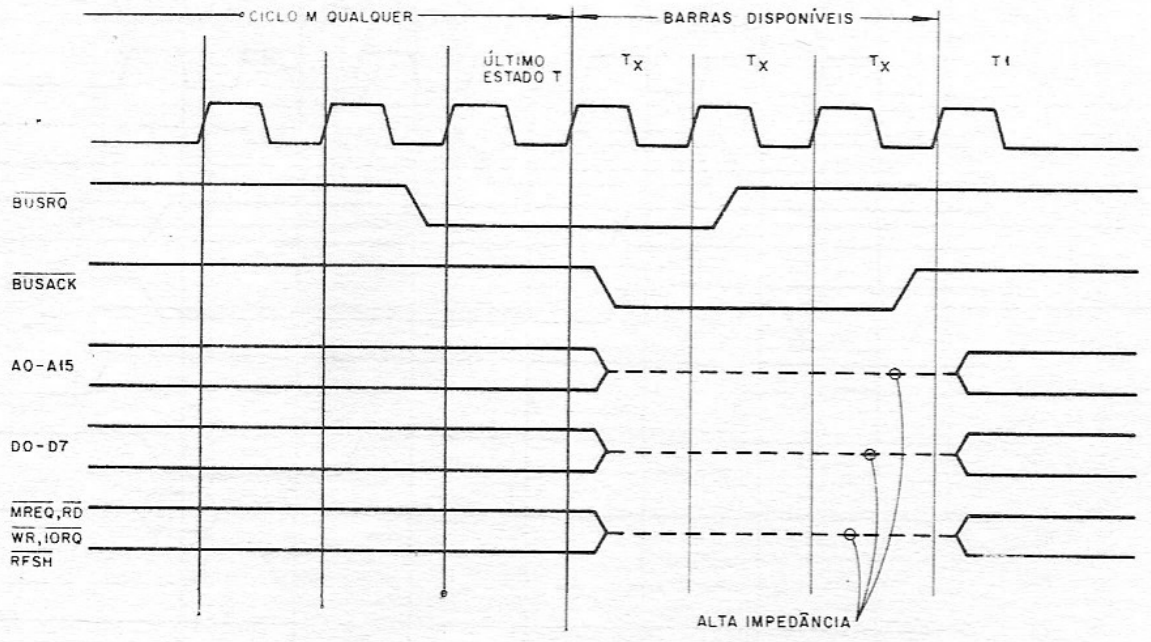


Fig. 5.9 – Ciclo de Pedidos de Controle das Barras

O Z-80 testa a linha $\overline{BUSREQ}$ no último estado de cada *ciclo de máquina*. Se $\overline{BUSREQ}$ for igual a zero, ele ativa no próximo estado a saída $\overline{BUSACK}$ ($\overline{BUSACK}$ = 0), informando ao meio externo que reconheceu o pedido de controle dos barramentos. Ainda neste instante, o Z-80 coloca em terceiro estado (alta impedância) as barras de dados e de endereçamento, e os sinais de controle. Assim, o dispositivo externo pode atuar nestas barras e linhas de controle para realizar a transferência de dados com a memória sem a interferência do Z-80. O estado da linha $\overline{BUSREQ}$ é testado, continuamente, nas subidas do sinal de clock. Quando o dispositivo externo terminar a transferência, ele desativa a linha $\overline{BUSREQ}$ ($\overline{BUSREQ}$ = 1). Esta condição será detectada pelo Z-80 na subida seguinte do sinal de clock, e com isso a linha $\overline{BUSACK}$ é desativada na próxima subida do sinal de clock. Assim, temos o início de um novo ciclo de máquina, e as barras e linhas de controle voltam a ser comandadas pelo Z-80. O processamento continua como se não tivesse ocorrido esta pausa causada pelo dispositivo externo, através da linha $\overline{BUSREQ}$.

5.7 – CICLO DE INTERRUPÇÃO INT

Um pedido de interrupção, através da entrada $\overline{INT}$, é atendido pelo Z-80 se a entrada $\overline{BUSREQ}$ estiver desativada e se o microprocessador estiver no modo ENABLE INTERRUPT, que é ativado por software através da *instrução EI*. Durante a subida do sinal de clock, do último estado do último ciclo de máquina de cada ciclo de instrução, o Z-80 testa a entrada $\overline{INT}$. Se ela estiver ativada ($\overline{INT}$ = 0), temos o início do ciclo de interrupção apresentado na figura 5.10.

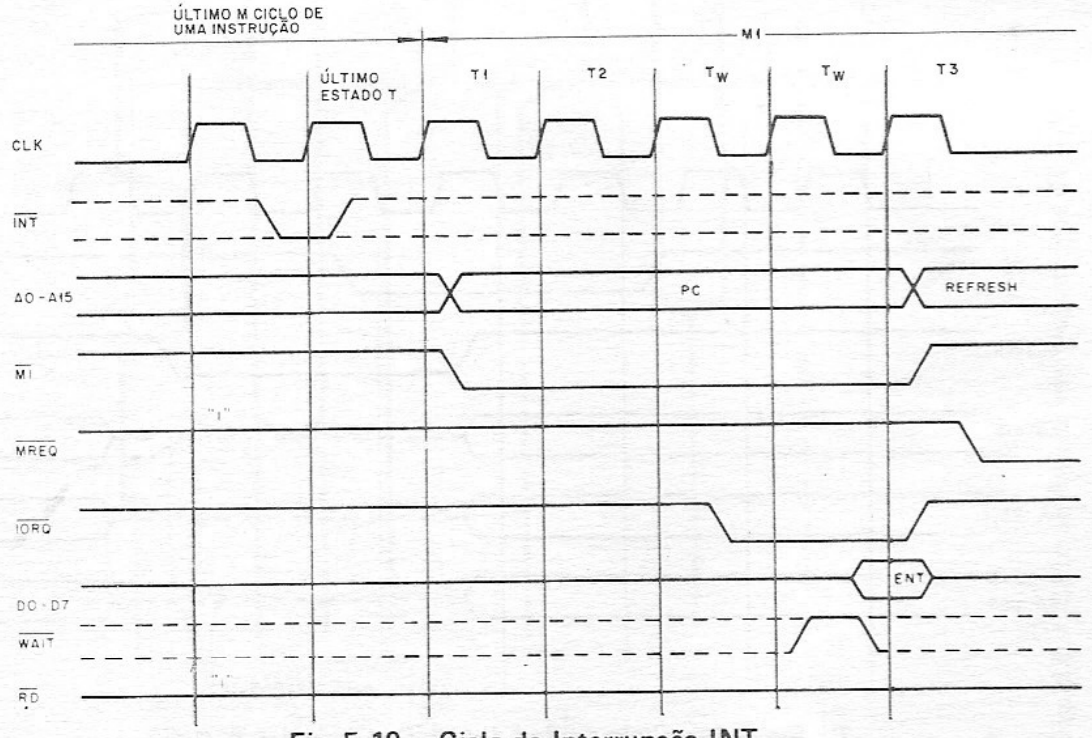


Fig. 5.10 – Ciclo de Interrupção INT

No estado T1 do ciclo de interrupção, a saída $\overline{M1}$ é ativada ($\overline{M1}$ = 0). Temos além do T2, dois estados de espera (TW) gerados internamente, de modo a permitir a utilização de dispositivos lentos.

A lógica externa de interrupção identifica o reconhecimento da interrupção quando temos os sinais $\overline{M1}$ e $\overline{IORQ}$ ativados ($\overline{M1}$ = $\overline{IORQ}$ = 0). Após identificar este reconhecimento, o circuito externo coloca o dado apropriado na barra de dados, que será lido pelo Z-80 no início do estado T3. Durante T3, os sinais $\overline{M1}$ e $\overline{IORQ}$ são desativados e o Z-80 inicia o procedimento de *refresh.*

5.8 – CICLO DE INTERRUPÇÃO NMI

O comportamento do Z-80 neste ciclo de máquina é apresentado na figura 5.11.

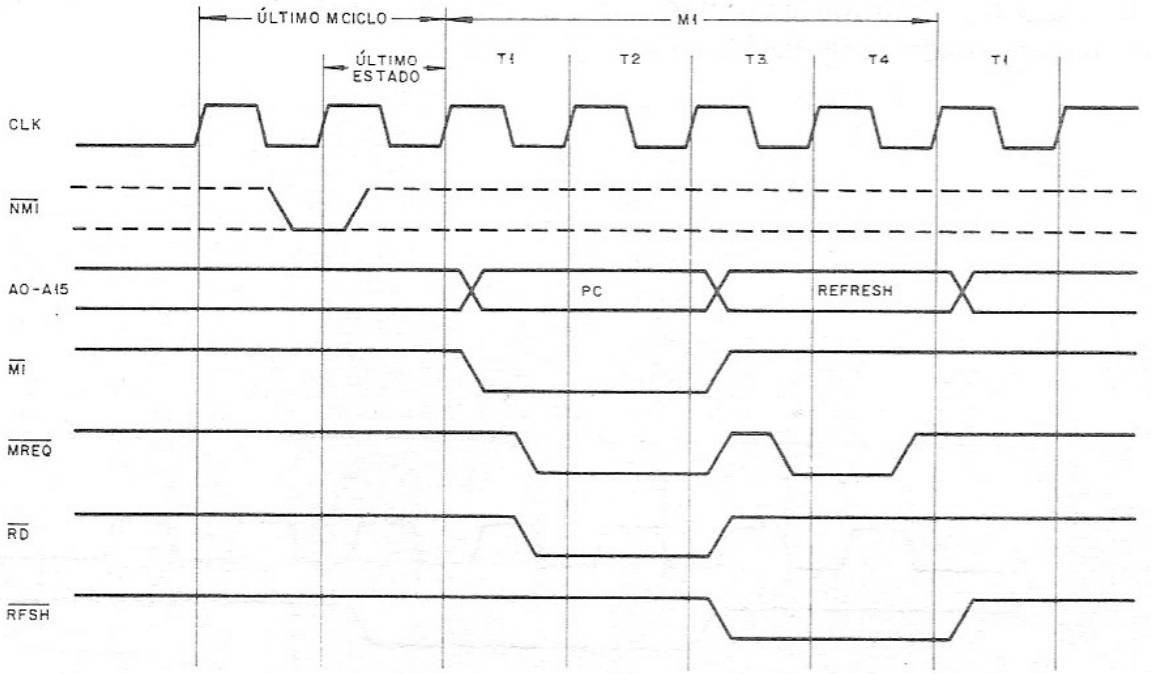


Fig. 5.11 – Ciclo de Interrupção NMI

Um pedido de interrupção, através da entrada $\overline{NMI}$, não pode ser inibido através de instruções. Além disso, é prioritário em relação aos pedidos provenientes da entrada $\overline{INT}$. Ele é reconhecido no último estado do último ciclo de máquina da instrução em andamento, conforme foi visto no ciclo de atendimento apresentado anteriormente.

O sinal $\overline{IORQ}$ não é ativado, pois não é necessário informar ao meio externo que foi aceita a interrupção. O primeiro ciclo de máquina é similar a uma operação de leitura de memória, exceto que o Z-80 não realiza esta leitura, pois o procedimento de entrada da rotina de atendimento é interno ao Z-80.

No capítulo 6 estudaremos com detalhes os modos de interrupção do Z-80.

5.9 – CICLO DE ESCAPE DA INSTRUÇÃO HALT

A instrução de HALT, quando executada, faz com que a saída $\overline{HALT}$ seja ativada ($\overline{HALT} = 0$). O Z-80 fica gerando continuamente ciclos M1, e não avança o contador de programa. Isto é importante, pois mesmo "parado" temos ciclos M1 que, por sua vez, realizam o *refresh* de memórias dinâmicas.

O estado de HALT somente pode ser interrompido pelo $\overline{RESET}$ ou por pedidos de interrupção. Quando isto ocorre, a saída $\overline{HALT}$ (pino 18) é desativada e o Z-80 inicia um novo ciclo de máquina. A saída do estado de HALT é apresentada na figura 5.12.

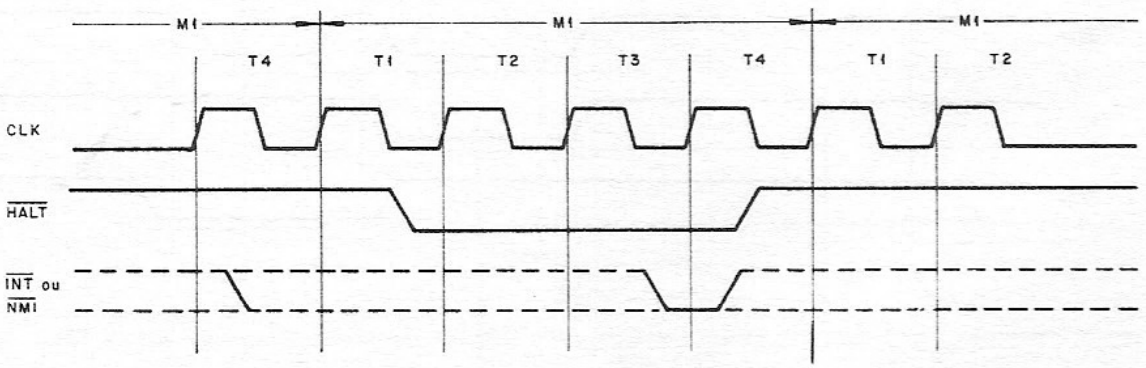


Fig. 5.12 – Ciclo de Escape da Instrução HALT

5.10 – CICLO DE RESET

A entrada $\overline{RESET}$, para ser reconhecida pelo Z-80, deve permanecer ativada ($\overline{RESET}$ = 0) durante pelo menos três ciclos do relógio. Enquanto o sinal de $\overline{RESET}$ permanecer ativado, as barras de dados e endereços ficam no terceiro estado (alta impedância) e as saídas de controle ficam inativas (nível 1). Após a entrada $\overline{RESET}$ ser desativada (nível 1), o Z-80 ainda precisa de dois estados (ciclos do relógio) antes de reiniciar o processamento em 0000H (PC = 0000H).

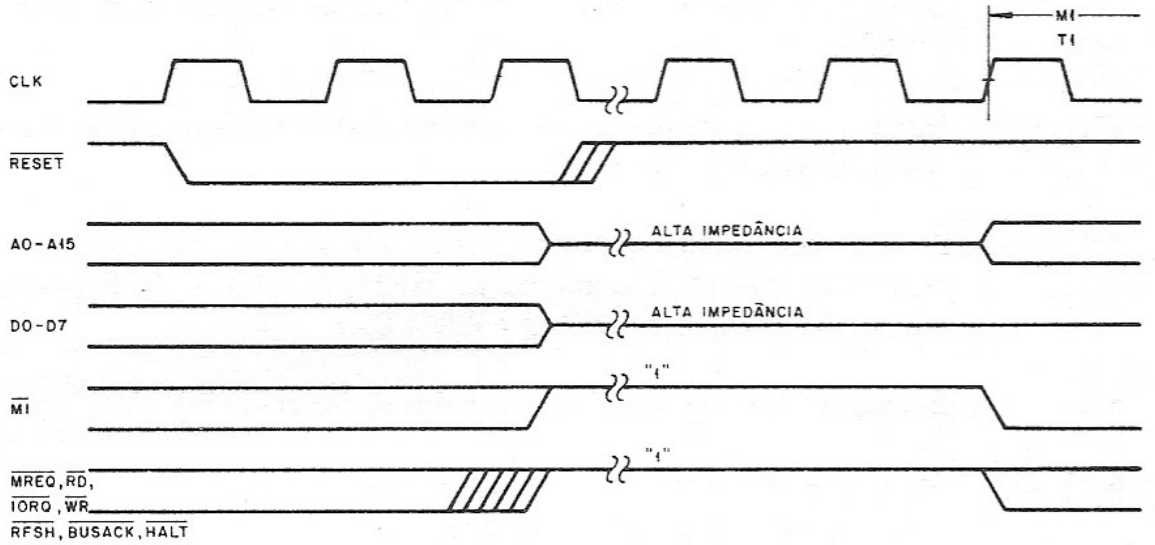


Fig. 5.13 – Ciclo de Reset

5.11 – EXERCÍCIOS DE FIXAÇÃO

5.11.1 – Determine o valor do estado (T) para as versões do Z-80 de 2,5MHz, de 6,0MHz e de 8,0MHz.

5.11.2 – O que é ciclo de instrução?

5.11.3 – Qual a relação entre estado, ciclo de máquina e ciclo de instrução?

5.11.4 – Quais as tarefas básicas realizadas pelo Z-80 durante o ciclo M1?

5.11.5 – Como é realizado o refresh de memórias dinâmicas?

5.11.6 – Quais os sinais envolvidos no projeto de um módulo de memória EPROM?

5.11.7 – Quais os sinais envolvidos no projeto de um módulo de memória RAM? E se for RAM dinâmica?

5.11.8 – Projetar um circuito lógico cuja saída seja ativada (= 0) quando tivermos na entrada a combinação $\overline{MREQ}$ = $\overline{RD}$ = 0. Faça um outro para as entradas $\overline{MREQ}$ = $\overline{WR}$ = 0.

5.11.9 – Explique como são gerados os estados de espera (TW).

5.11.10 – Quais os sinais envolvidos no projeto de uma porta de entrada de dados? E no caso de porta de saída?

5.11.11 – Explique o ciclo de pedido de controle das barras.

5.11.12 – Projetar um circuito lógico cuja saída seja ativada (= 0) quando o Z-80 executar um ciclo de reconhecimento de interrupção proveniente da entrada $\overline{INT}$.

5.11.13 – Em que estado o Z-80 realiza a leitura da entrada $\overline{INT}$?

5.11.14 – Em que condições o Z-80 atende uma interrupção proveniente da entrada $\overline{INT}$?

5.11.15 – O que acontece com o refresh de memórias dinâmicas, se o Z-80 executar uma instrução de parada (HALT)?

5.11.16 – O que determina o momento de teste das entradas $\overline{INT}$ e $\overline{WAIT}$ do Z-80?

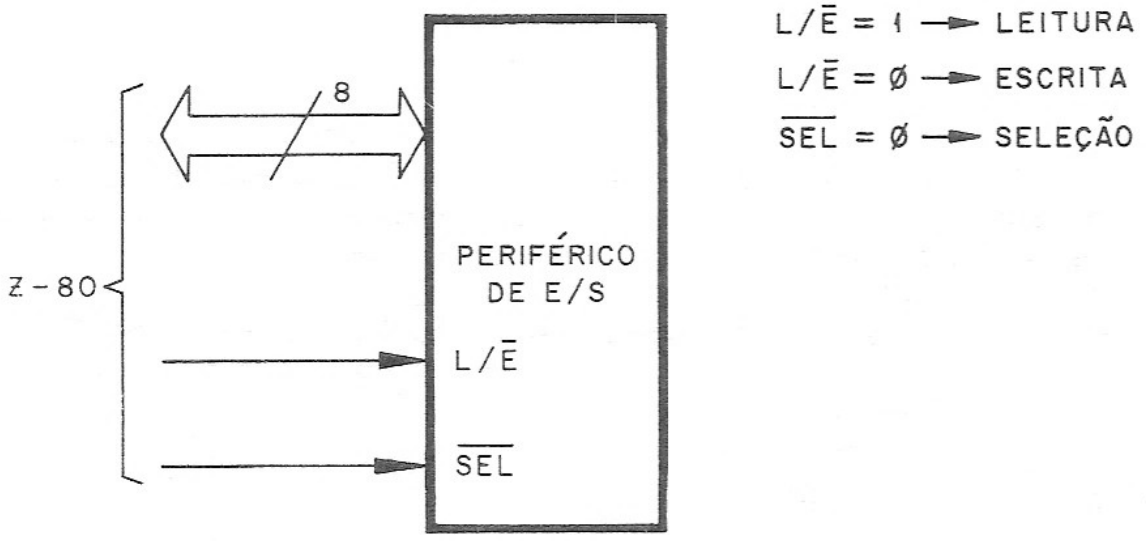
5.11.17 – Suponha que um microcomputador controla um periférico lento

que para realizar a transferência com o Z-80 leva um tempo 2T. Neste mesmo microcomputador existe um controlador de DMA, que uma vez realizado o pedido de DMA, a transferência tem que iniciar num tempo T. É possível realizar um sistema com esta especificação usando-se a entrada $\overline{BUSREQ}$ do Z-80? Como deve ser implementado um sistema com tal especificação?

5.11.18 – Suponha que para testar o funcionamento de um microcomputador baseado no Z-80, resolveu-se realizar um circuito que paralisa o microprocessador em cada estado T2. Implemente tal circuito.

5.11.19 – Com relação ao exercício anterior, realize um circuito para paralisar o Z-80 somente a cada ciclo de busca de instrução.

5.11.20 – Implemente a ligação do Z-80 com o periférico de entrada e saída mostrado abaixo. Este periférico deve ser tratado como uma posição de memória com endereço 8000H.



6 A INTERRUPÇÃO

6.1 – APRESENTAÇÃO

Após termos estudado no capítulo anterior os ciclos de máquina do Z-80, apresentaremos o conceito e os modos de interrupção existentes neste microprocessador.

O conceito de interrupção é muito importante, pois a maioria dos microcomputadores de uso geral e de controle de processo utilizam tal conceito na consecução de suas tarefas.

6.2 – CONTROLE DE E/S

Os dispositivos de entrada e saída (E/S) dos microcomputadores, tais como: teclados, displays e sensores, precisam ser controlados pelo processador de maneira eficiente, de modo a não prejudicar o desempenho dos objetivos principais do sistema. Se as técnicas de controle de E/S não forem bem escolhidas, podem se tornar um fator de degradação do sistema.

Existem duas técnicas, que são as mais usadas no controle de E/S:

1 – Varredura

2 – Interrupção

6.2.1 – Técnica de Varredura

A *técnica de varredura*, também denominada *polling* ou *scanning*, é ilustrada na figura 6.1.

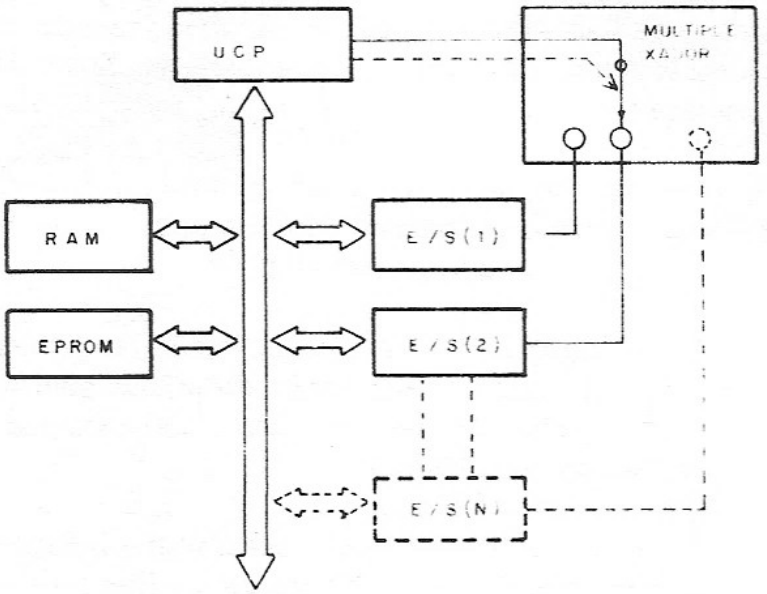


Fig. 6.1 – Técnica de Varredura

Nesta técnica o processador testa por software, e em seqüência, cada dispositivo de E/S. Este teste consiste em verificar se o dispositivo de E/S necessita de alguma tarefa do processador. Observe que grande parte do tempo é perdido nesta observação contínua destes dispositivos. Assim, existindo um número elevado de dispositivos de E/S, teremos uma degradação no desempenho do sistema, que torna-se ainda mais crítica em sistemas de tempo real. No entanto, esta técnica é usada em pequenos sistemas com um nível de processamento.

6.2.2 – Técnica de Interrupção

Na *técnica de interrupção* o processador executa o programa principal e, quando existe uma tarefa a ser executada em dispositivo de E/S, recebe um *pedido de interrupção* que "suspende", momentaneamente, a execução do programa principal para que seja realizada a tarefa. Para tanto, existe uma entrada assíncrona no processador que permite ao dispositivo de E/S interromper o processamento. Quando ocorre uma interrupção, o processador termina a execução da instrução em andamento e inicia uma nova rotina, que faz as tarefas pedidas pelo dispositivo de E/S.

Uma vez encerrada esta rotina, o processador prossegue a execução do programa interrompido, a partir do ponto onde houve a paralisação.

Se o número de dispositivos de E/S exceder o número de entradas do processador, torna-se necessário um hardware ou software adicionais para auxiliar na identificação da origem da interrupção. Uma solução simples para este problema é a pesquisa por software dos dispositivos de E/S, aplicando a técnica da varredura. Uma solução alternativa, denominada *técnica vetorada,* é

adicionar um hardware que forneça uma informação para o processador, denominada *vetor,* que identifica o dispositivo que gerou o pedido de interrupção. Além disso, este hardware tem que prever uma maneira de diferenciar interrupções geradas simultaneamente. Para tanto, deve dispor de uma lógica que escalone *prioridades.*

Estas duas alternativas podem ser, ainda, misturadas. O vetor pode identificar um grupo de entradas de interrupções que, por sua vez, são identificadas individualmente pela técnica de varredura.

A técnica de interrupção, evidentemente, melhora o desempenho dos sistemas, pois permite que mais tarefas sejam assumidas pelo microcomputador. Em contrapartida, necessita de hardware e software mais complexos que os utilizados na técnica de varredura.

A maioria dos sistemas de interrupção permite inibir (mascarar) ou habilitar interrupções por operações de software, dando liberdade ao programador de evitar interrupções durante certos tipos de processamento, tais como: loops de tempo e trechos de sistemas operacionais.

Entradas de interrupções que não podem ser mascaradas são denominadas de *non-maskable interrupt,* e servem para prevenir quedas de alimentação ou qualquer outra atividade que prevaleça no sistema.

Um exemplo de aplicação de interrupções que permite a realização de operações de entrada e saída, concomitantemente com o processamento, é o calendário da figura 6.2.

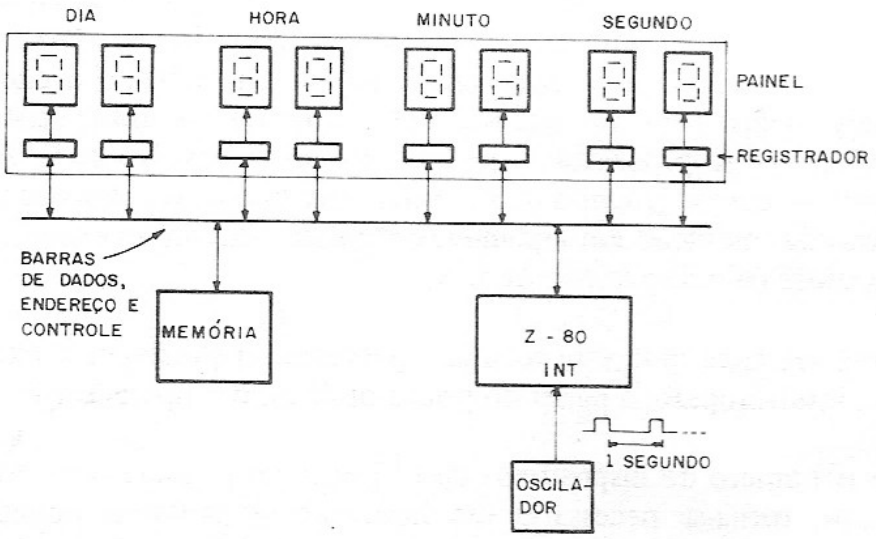


Fig. 6.2 – Calendário Controlado pelo Z-80

Neste exemplo, o calendário é mostrado no painel do microcomputador, sendo composto de um registrador para cada display, o qual armazena um dígito. Cada registrador é uma porta de entrada e saída onde o processador armazena um dígito e ainda pode ler a data/hora durante o processamento.

Um relógio gera uma interrupção a cada 1 seg, forçando o processamento de uma rotina que realiza a atualização da data/hora. Tal rotina não leva mais do que 5 ms para cada atualização, o que deixa 995 ms para outras tarefas.

6.3 – INTERRUPÇÕES NO Z-80

Como vimos no capítulo 4, o Z-80 tem duas entradas de interrupção: a $\overline{NMI}$ – non-maskable interrupt – pino 17 e a $\overline{INT}$ – a interrupção externa normal – pino 16.

A interrupção NMI tem mais alta prioridade e é sempre reconhecida pelo Z-80. A interrupção INT só é atendida se as interrupções estiverem habilitadas (*"enabled"*). Esta condição é indicada pelo flip-flop de interrupção IFF1, quando for setado pela instrução EI – enable interrupt. Quando IFF1 for resetado pela instrução DI – disable interrupt, as interrupções ficam bloqueadas. O *status* do sistema de interrupção é indicado pelos flip-flops internos do Z-80, IFF1 e IFF2, cujos detalhes estudaremos mais adiante.

O Z-80 tem uma única forma de atender interrupções provenientes da entrada NMI. No entanto, a interrupção mascarável INT tem três modos de operação: *Modo 0, Modo 1* e *Modo 2*. Estes modos são programados, respectivamente, pelas instruções IM0, IM1 e IM2.

O *Modo 0* é compatível com o sistema 8080, o *Modo 1* é muito semelhante à forma de atendimento da interrupção NMI e, o Modo 2 utiliza a técnica vetorada e é compatível com a família de circuitos integrados LSI periféricos do Z-80.

As linhas $\overline{NMI}$ e $\overline{INT}$ são amostradas na transição positiva do último período do relógio em cada instrução. O processo subseqüente de atendimento depende do tipo de interrupção que foi detectada e do modo de operação da interrupção $\overline{INT}$.

6.3.1 – A Interrupção NMI

Quando a entrada NMI ficar ativa (NMI = 0), a interrupção é reconhecida no final da instrução que estiver em processo de execução. O Z-80, efetivamente, executa uma instrução de RESTART (RST) para a posição de me-

mória 0066H (o final "H" significa que o número está representado na base hexadecimal). A instrução RST armazena o conteúdo do contador de programa na pilha e transfere o controle para uma das oito posições de memória 00000, 0008H, ..., 0038H. A interrupção NMI atua da mesma maneira, mas transfere o controle, unicamente, para a posição de memória 0066H.

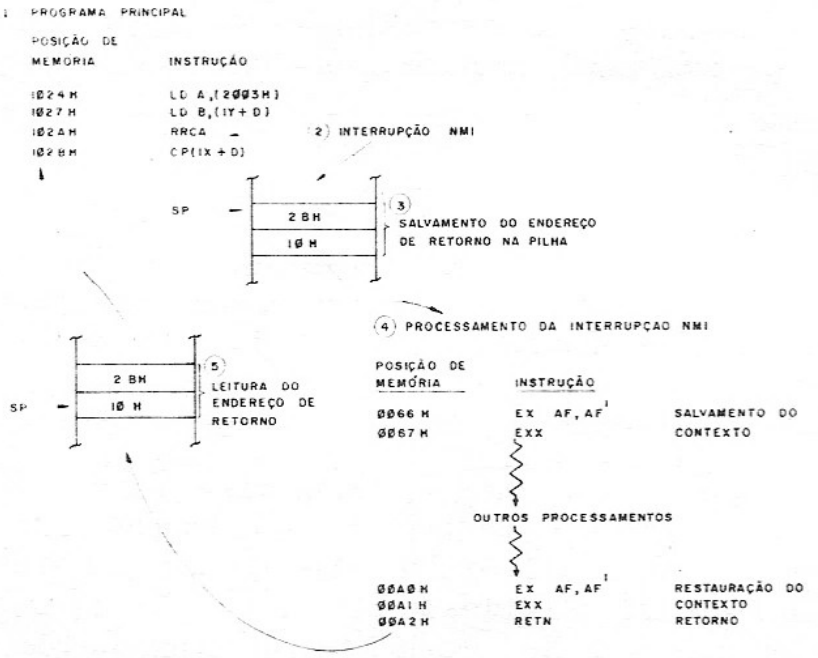


Fig. 6.3 – Exemplo do Processamento da Interrupção NMI

A figura 6.3 ilustra a ação da interrupção NMI. Neste exemplo, a interrupção NMI ocorre justamente durante a execução da instrução RRCA. No final desta instrução, o conteúdo do PC (102BH) é armazenado na pilha, o SP é decrementado de duas unidades, e a próxima instrução é apanhada da posição de memória 0066H, iniciando o programa de tratamento da interrupção NMI.

As instruções EX AF, AF' e EXX realizam o salvamento do contexto pela troca dos conteúdos dos registradores passivos e ativos. Em seguida, inicia-se, efetivamente, o processamento do pedido de interrupção. A complexidade deste processamento depende das funções atribuídas à interrupção NMI. Ao final, o contexto é restaurado pelas mesmas instruções que realizaram o salvamento.

A última instrução é a RETN, que é especial para o retorno deste tipo de interrupção. Esta instrução transfere o conteúdo do topo da pilha para o PC, fazendo com que o Z-80 dê continuidade ao programa principal na posição de memória 102BH, com todos os registradores e bits de condição no mesmo estado que se encontravam anteriormente à ocorrência da interrupção NMI. Além disso, a instrução RETN afeta os flip-flops IFF1 e IFF2, conforme explicado a seguir.

O flip-flop IFF1, conforme já vimos, assinala a condição de habilitação (EI) ou bloqueio (DI) da interrupção mascarável INT. O flip-flop IFF2 é usado para armazenar, temporariamente, o estado de IFF1 durante o processamento da interrupção NMI.

A interrupção NMI, além de armazenar o estado de IFF1 em IFF2, reseta IFF1, bloqueando a ocorrência da interrupção INT.

A instrução RETN transfere o estado de IFF2 para IFF1, restaurando assim o estado original de IFF1 antes da ocorrência da interrupção NMI.

Se o programa de tratamento da interrupção NMI permitir o atendimento de interrupções INT, durante o seu transcurso, basta executar uma instrução EI após o salvamento do contexto.

6.3.2 – O Modo 0

Após a ativação do sinal de RESET, o Z-80 entra no modo 0 ("default"), até que seja definido por software outro modo de interrupção.

O Modo 0 é semelhante ao processamento de interrupções no 8080, e também pode ser estabelecido através da instrução IMO.

Se o Z-80 estiver operando no Modo 0, o flip-flop IFF1 estiver setado e ocorrer uma interrupção INT, tomam lugar os seguintes eventos:

1 – Ocorrência da interrupção ($\overline{INT} = 0$).

2 – No final da instrução que se encontra em fase de execução a interrupção é reconhecida.

3 – O Z-80 sinaliza o reconhecimento da interrupçao ativando as linhas $\overline{IORQ}$ e $\overline{M1}$ ($\overline{IORQ} = 0$ e $\overline{M1} = 0$).

4 – O dispositivo externo, ao identificar $\overline{IORQ}$ e $\overline{M1}$ ativos, deposita o código de uma instrução na barra de dados, normalmente um RESTART (RST). Durante a busca desta instrução o Z-80 insere, automaticamente, dois estados de WAIT.

5 – A instrução RST que define a localização da rotina de tratamento de interrupção é executada.

6 – A rotina de tratamento da interrupção assume o controle da máquina e a sua última instrução executada é RETURN FROM INTERRUPT (RETI).

Ilustraremos o funcionamento do Modo 0 através do exemplo de um controlador de teclado que opera à base de interrupção. A figura 6.4 mostra as ações tomadas no atendimento da interrupção.

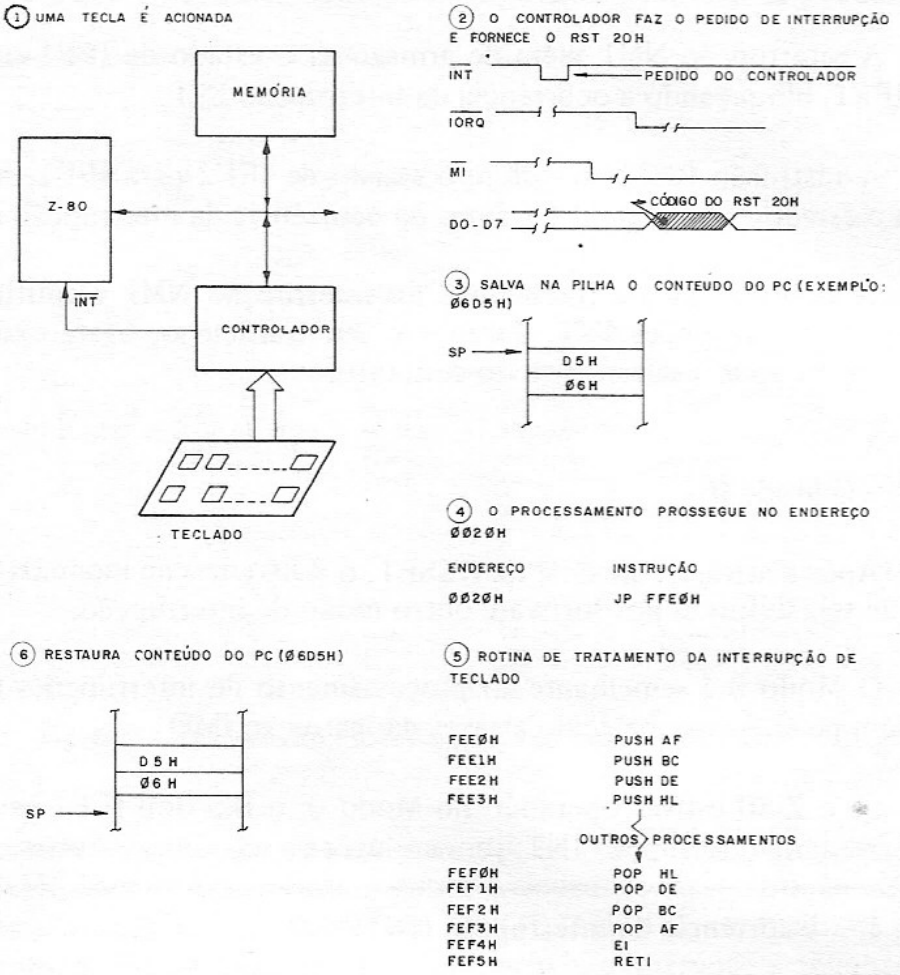


Fig. 6.4 — O Processamento do Modo 0

Quando um caractere for digitado pelo operador, o controlador de teclado força a linha $\overline{INT}$ para o nível lógico baixo ($\overline{INT}$ = 0), gerando uma interrupção. Quando o Z-80 atender a esta interrupção, as linhas $\overline{IORQ}$ e $\overline{M1}$ são colocadas no nível lógico baixo ($\overline{IORQ}$ = $\overline{M1}$ = 0). Este evento é interpretado pelo controlador de teclado como o reconhecimento do pedido de interrupção, o que o faz colocar a instrução RST 20H na barra de dados. O Z-80 executa o RST 20H, armazenando o conteúdo do PC na pilha (06D5H) e transferindo o controle para o endereço 0020H da memória. Na posição 0020H tem uma instrução JP FEE0H (JUMP). Esta instrução transfere o controle para a rotina de tratamento de interrupção de teclado, que inicia-se na posição de memória FEE0H.

Quando o Z-80 reconhece uma interrupçao através da entrada INT, o flip-flop IFF1 é automaticamente resetado, inibindo qualquer outra interrupção posterior, até que seja executada uma instrução EI.

A rotina de tratamento de interrupção de teclado armazena o conteúdo dos registradores internos do Z-80 na pilha, através de operações de PUSH. As instruções de PUSH guardam na pilha os registradores de 8 bits aos pares. Na primeira operação são armazenados o acumulador e os flags, em seguida os pares BC, DE e HL, respectivamente. Uma forma alternativa de salvar o contexto seria aplicando as operações de troca entre registradores passivos e ativos (EX AF, AF' e EXX).

Após o salvamento dos registradores, o Z-80 executa instruções que permitem ler o caractere digitado pelo operador e processá-lo. No final desta rotina, o contexto dos registradores do Z–80 é restaurado por operações de *POP*. As instruções de *POP* transferem da pilha para o Z-80 os pares de registradores na ordem inversa de armazenamento. Em seguida, a instrução EI habilita as interrupções e a instrução RETURN FROM INTERRUPT (RETI) devolve o controle do programa interrompido na posição 06D5H.

Em alguns sistemas torna-se necessário termos diversos dispositivos de E/S controlados à base de interrupção. No caso do microprocessador Z-80, só é possível usar a técnica vetorada, para identificar o dispositivo que pediu a interrupção, nos Modos 0 e 2.

Quando existe mais de um dispositivo capaz de gerar interrupção é necessário escalonar prioridades em casos de simultaneidade de pedidos. Por isso, existem circuitos integrados LSI controladores de interrupção, que resolvem pedidos simultâneos, geram o pedido de interrupção para o processador e, também, geram o *vetor* apropriado de acordo com o dispositivo de maior prioridade. A figura 6.5 ilustra o diagrama de blocos de um dispositivo controlador de interrupções. Os circuitos mais populares são: 8214 e 8259 da Intel.

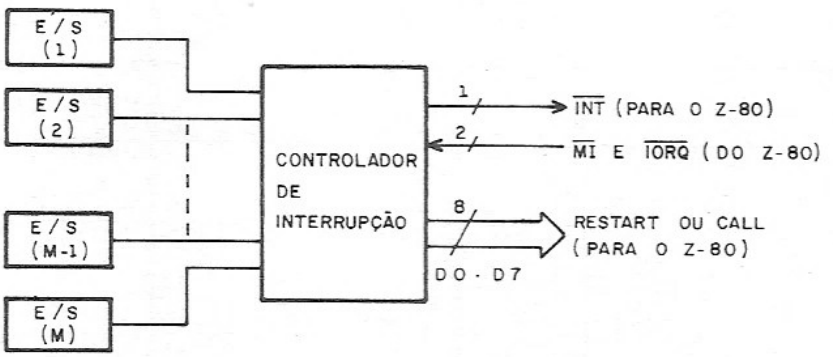


Fig. 6.5 – O Controlador de Interrupções

6.3.3 O modo 1

O *Modo 1* não é compatível com o microprocessador 8080. As ações tomadas no transcorrer do atendimento do Modo 1 são idênticas a interrupção NMI, com exceção do endereço de RESTART que é 0038H.

O PC é armazenado no topo da pilha e a rotina de atendimento de interrupção inicia-se na posição 0038H.

A vantagem do Modo 1 é que não é necessária nenhuma lógica externa adiconal para fornecer a instrução de RESTART na barra de dados no momento apropriado. A lógica externa de interrupção precisa apenas ativar a linha $\overline{INT}$ e reconhecer o início do atendimento. No entanto, só é possível um nível. Para operarmos no Modo 1 com vários dispositivos de E/S, temos que usar a técnica de varredura para identificar o dispositivo que gerou a interrupção.

6.3.4 – O Modo 2

Este modo de operação permite até 128 interrupções vetoradas para posições de memória pré-definidas pelo programador.

A família de periféricos oferecida pela Zilog (8420-PIO, 8430-CTC e outros) está especialmente equipada para interagir com o Z-80 operando no Modo 2. Tais componentes, durante o processo de atendimento da interrupção, geram parte do endereço do vetor. Além disso, podem ser conectados em cascata (*"daisy chain"*), resolvendo completamente o problema de ocorrência de interrupções simultâneas. Existem níveis de prioridades independentes para cada componente. Neste esquema de ligação, a prioridade é dada pela posição do controlador na seqüência física. A conexão em cascata está ilustrada na figura 6.6 utilizando a PIO – Parallel Input/Output Controller.

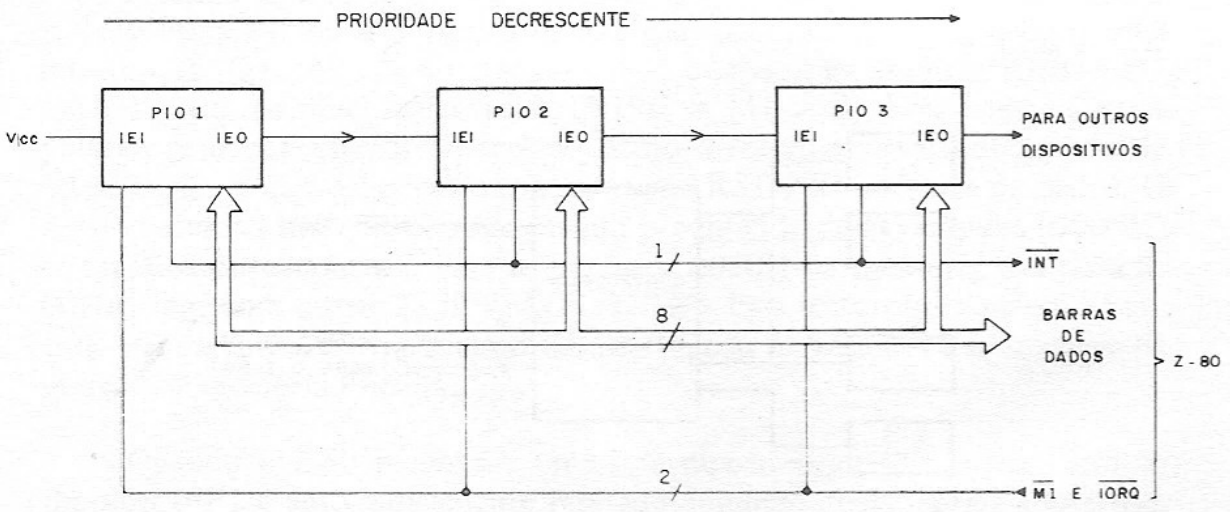


Fig. 6.6 – Conexão em Cascata com a PIO

Cada PIO está ligada à entrada $\overline{INT}$ (pino 16) do Z-80 na configuração wired-OR. Se a entrada IEI-Interrupt Enable In do dispositivo mais prioritário estiver no nível lógico alto antes da ativação do sinal $\overline{INT}$, o sinal IEO é colocado no nível lógico baixo. Isto significa que as PIO's menos prioritárias não podem pedir interrupções através do sinal $\overline{INT}$. Quando o Z-80 reconhecer a interrupção, ativando os sinais $\overline{M1}$ e $\overline{IORQ}$, a PIO automaticamente coloca na barra de dados parte do endereço do vetor. No final da rotina de atendimento, a PIO detecta a busca do código da instrução RETI. Isto significa para a PIO que o processamento da interrupção terminou e, em seguida, ela coloca no nível lógico alto a sua saída IEO, habilitando interrupções de dispositivos menos prioritários.

O esquema de prioridades apresentado acima, não só resolve problemas de simultaneidade de interrupções, como permite interrupções em vários níveis.

O Modo 2 tem como base uma *tabela de vetores de interrupção* alocada em alguma região na memória. Esta tabela tem (2 x N) bytes, onde N é número de interrupções do sistema. O seu início é dado pelo endereço IIIIIIII00000000B onde IIIIIIII é o conteúdo do registrador I – vetor de interrupção do Z-80. Para cada interrupção, o registrador I fornece os 8 bits mais significativos do endereço de entrada na tabela, e o dispositivo de E/S fornece os 8 bits menos significativos. Cada entrada na tabela de vetores fornece 2 bytes que encerram o endereço da primeira instrução da *rotina de tratamento de interrupção*. O primeiro byte armazena a parte menos significativa do endereço e o segundo byte a mais significativa. Esta tabela pode ter até 128 entradas, num total de 256 bytes, conforme ilustra a figura 6.7.

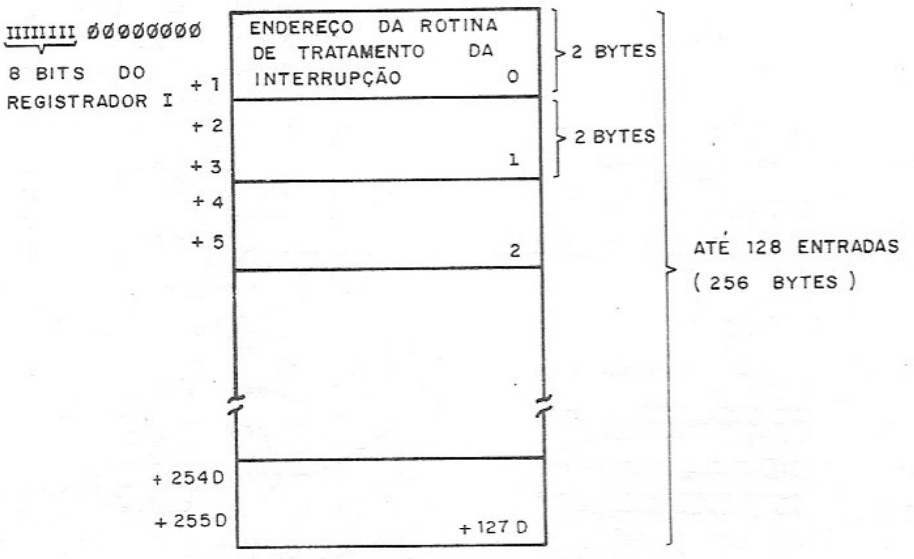


Fig. 6.7 – Tabela de Vetores

A seqüência geral de eventos no atendimento de uma interrupção no Modo 2 é a seguinte:

1 – Se IFF1 = 1 e $\overline{INT}$ = 0, o Z-80 reconhece a interrupção no próximo ciclo M1.

2 – O dispositivo de E/S responde ao reconhecimento da interrupção, depositando na barra de dados os 8 bits menos significativos do endereço de entrada na tabela de vetores.

3 – Os 8 bits fornecidos pelo dispositivo de E/S são concatenados com o conteúdo do registrador I, formando o endereço de entrada na tabela de vetores.

4 – O Z-80 transfere o conteúdo do PC para o topo da pilha.

5 – A tabela de vetores é acessada utilizando o endereço montado no passo 3.

6 – O PC é carregado com o conteúdo da tabela de vetores obtido no passo 5, para efetivamente desviar o curso do programa para a rotina de atendimento da interrupção.

Na figura 6.8 temos um exemplo do Modo 2, implementado com três PIO's.

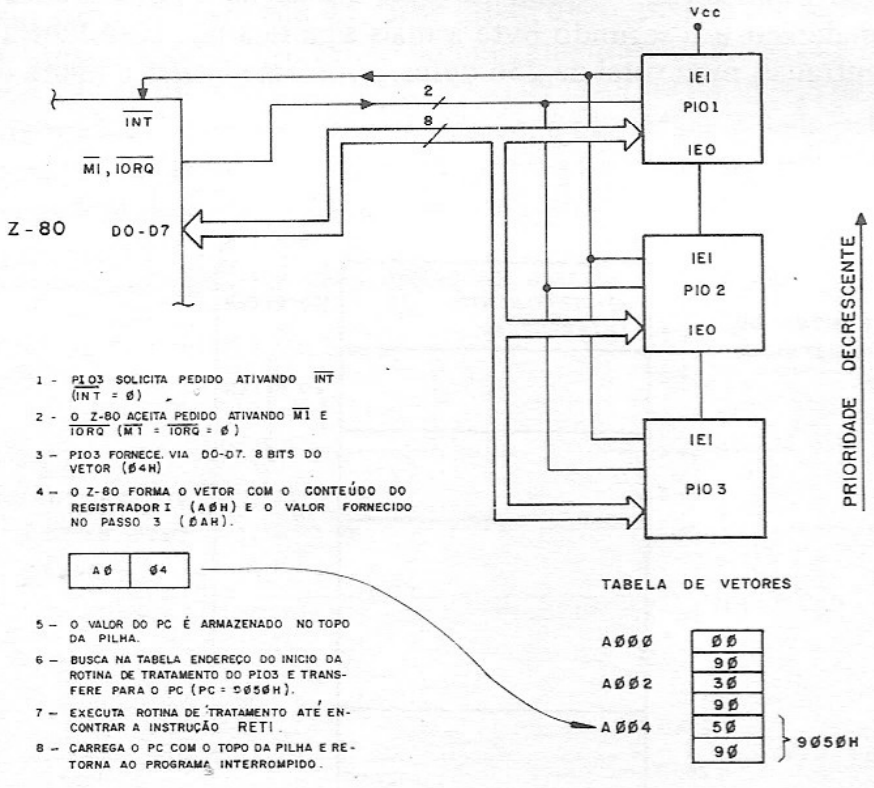


Fig. 6.8 – Exemplo do Modo 2

A tabela de vetores de interrupção inicia-se na posição de memória A000H e possui 3 entradas de 2 bytes, que definem endereços de rotinas de interrupção para cada um dos 3 dispositivos de E/S.

No início, através do "bootstrap", o registrador I é carregado com o valor A0H. Se o dispositivo de E/S número 3 ativar a linha $\overline{INT}$, o Z-80 sinaliza o reconhecimento deste pedido de interrupção, ativando os sinais $\overline{M1}$ e $\overline{IORQ}$. O dispositivo de E/S, então, coloca na barra de dados os 8 bits menos significativos do endereço de entrada na tabela de vetores. Em seguida, o Z-80 monta o endereço, concatenando o conteúdo do registrador I (A0H) e os 8 bits fornecidos pelo dispositivo de E/S (04H), formando o endereço A004H. O Z-80 armazena o conteúdo do PC no topo da pilha e busca o conteúdo das posições de memória A004H e A005H, que determinam o endereço inicial (9050H) da rotina de atendimento da interrupção. O controle é então transferido para esta posição de memória, a partir da qual a rotina de atendimento é processada até a execução da instrução RETI, quando então se dá o retorno ao programa interrompido.

Uma vez que cada entrada na tabela de vetores é composta de dois bytes, o bit menos significativo do byte fornecido pelo dispositivo de E/S é sempre zero. Logo, o dispositivo que utiliza a entrada (n) da tabela, tem que fornecer o valor (2 x n) através da barra de dados.

O esquema de prioridade do Modo 2 não só resolve problemas de simultaneidade de pedidos de interrupção, mas também controla níveis diferentes de interrupção. Para entendermos melhor o funcionamento em vários níveis com interrupções aninhadas (*"nested interrupts"*), estudaremos a configuração apresentada na figura 6.9.

Fig. 6.9 — Níveis de Interrupção

Durante a execução do programa principal, na posição de memória 0A82H, o dispositivo de E/S 2 interrompe o processamento. A rotina de atendimento inicia o seu processamento na posição 2000H, após o salvamento do contexto. Com a interrupção novamente habilitada, o dispositivo de E/S 1 interrompe a rotina de atendimento do dispositivo de E/S 2, desviando o processamento para a posição 3000H. Finalmente, o dispositivo de E/S 0 interrompe o processamento do dispositivo de E/S 1, desviando o curso do programa para 4000H. Esta última rotina é executada até o seu final, quando é encontrada a instrução RETI. Após a execução da instrução RETI, a rotina do dispositivo de E/S 1 reassume o controle na posição 3000H. Após a execução da RETI desta rotina, o dispositivo de E/S 2 volta a ser controlado a partir da posição 2000H. Esta rotina, de nível menos prioritário, encerra o seu processamento com a instrução RETI, devolvendo o controle para o programa principal na posição 0A82H.

Neste exemplo, durante o atendimento do dispositivo 0, o mais prioritário, tivemos três *interrupções aninhadas.*

6.4 – EXERCÍCIOS DE FIXAÇÃO

6.4.1 – Cite uma vantagem e uma desvantagem da técnica de varredura.

6.4.2 – Conceituar a técnica de interrupção.

6.4.3 – O que é vetor de interrupção? Quando é necessária a sua utilização?

6.4.4 – Cite uma aplicação da entrada não mascarada ($\overline{NMI}$).

6.4.5 – Em que instante o Z-80 testa as entradas $\overline{NMI}$ e $\overline{INT}$?

6.4.6 – Qual o endereço inicial da rotina de atendimento da interrupção NMI?

6.4.7 – Como é preservado o estado do flip-flop IFF1 no atendimento da interrupção NMI?

6.4.8 – Qual dos modos de interrupção é compatível com o 8080?

6.4.9 – O que deve fazer o hardware externo no atendimento do Modo 0?

6.4.10 – Qual a função do controlador de interrupções?

6.4.11 – Projete um circuito de controle para utilizar o Modo 1 com quatro dispositivos de E/S.

6.4.12 – No Modo 2, qual a função da tabela de vetores de interrupção?

6.4.13 – Porque a tabela de vetores de interrupção permite somente 128 entradas?

6.4.14 – No exemplo da Fig. 6.8, explique o que acontece com a pilha.

PARTE III
APLICAÇÕES DO Z-80

7 INTERFACEANDO O Z-80

7.1 – APRESENTAÇÃO

Após termos visto nos capítulos anteriores o hardware do Z-80, estudaremos agora como projetar módulos de memória e E/S.

Apresentaremos três projetos de módulos de memória, sendo um de EPROM e dois de RAM. Também projetaremos uma porta de entrada e outra de saída, ambas de 8 bits.

Neste capítulo daremos ênfase à parte conceitual destes projetos, deixando os aspectos práticos para o próximo capítulo.

7.2 – MÓDULO DE 6K BYTES DE EPROM

Seja o circuito integrado 2716, que é uma memória EPROM de 2K bytes. Na figura 7.1 temos o diagrama lógico e o procedimento de leitura desta memória.

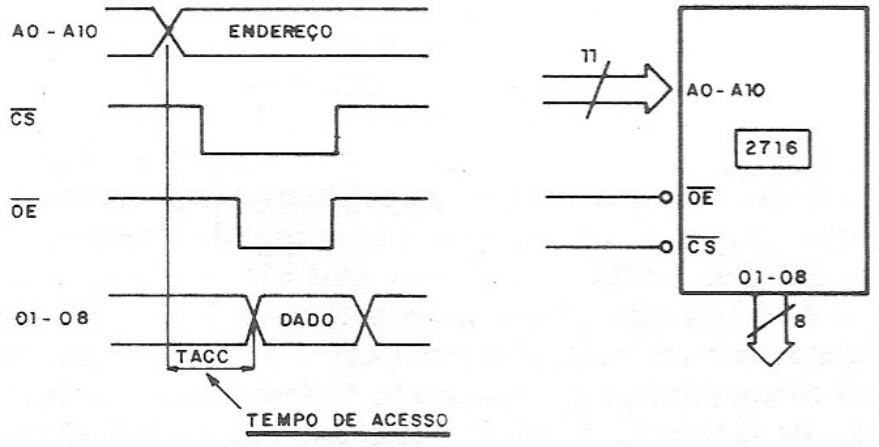


Fig. 7.1 – A EPROM 2716

Na prática, além do *tempo de acesso,* que será considerado no máximo igual a 350 ns, temos outros tempos que são levados em consideração. O *tempo de acesso* é o intervalo de tempo entre a colocação do endereço na barra e a resposta da memória. Aqui neste projeto didático, nos preocuparemos com a lógica de controle necessária ao funcionamento do Z-80 com a 2716.

Como a 2716 é uma memória de 2K bytes, ela necessita de onze linhas de endereçamento. A entrada $\overline{CS}$ ("chip select") é a responsável pela ativação da 2716, quando vai para o nível 0. A entrada $\overline{OE}$ ("output enable") libera os circuitos internos da 2716 que são buffers *tri-state.*

Na figura 7.2 temos um diagrama elétrico de um módulo de 6K bytes de EPROM usando a 2716.

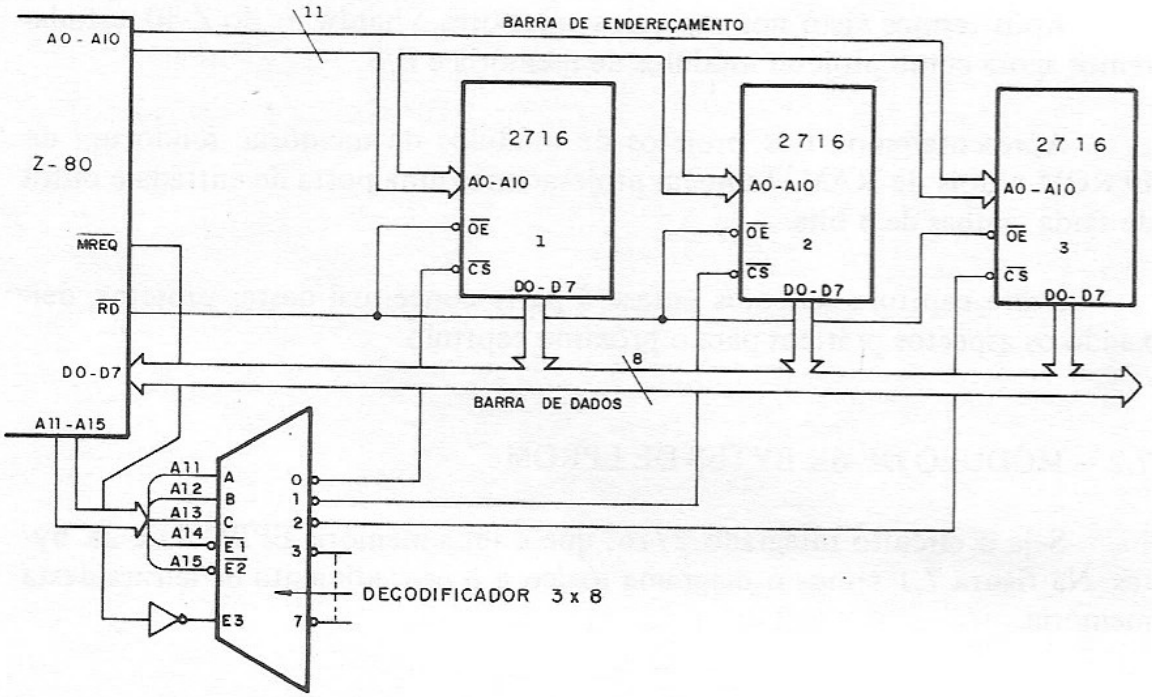


Fig. 7.2 – Projeto de um Módulo de 6K bytes de EPROM

Observe que as onze linhas menos significativas (A0 – A10) da barra de endereçamento chegam nos três circuitos integrados de memória. O controle da saída de dados da EPROM é feito pela linha $\overline{RD}$, pois quando ela vai para o nível 0, o Z-80 está com a barra de dados (D0 – D7) no modo entrada. Os bits restantes da barra de endereçamento (A11 – A15) são usados para a seleção de cada circuito integrado de memória. O decodificador utilizado possui três entradas de enable ($\overline{E1}$, $\overline{E2}$ e E3). Nas duas primeiras, que são ativadas no nível 0, temos as linhas A14 e A15. Na entrada E3, temos um inversor que recebe como entrada a linha $\overline{MREQ}$.

Assim, quando tivermos a combinação A14 = A15 = 0 e $\overline{MREQ}$ = 0, o decodificador estará habilitado. Isto ocorrerá quando o Z-80 realizar um acesso à memória ($\overline{MREQ}$ = 0) em um endereço que tenha os bits A14 e A15 iguais a zero. Os bits A11, A12 e A13, que chegam nas entradas A, B e C, respectivamente, serão os responsáveis pela ativação das saídas do decodificador. Vejamos as combinações que ativam as saídas 0, 1 e 2:

A11 = A12 = A13 = A14 = A15 = 0, $\overline{MREQ}$ = 0 SAÍDA 0
A11 = 1, A12 = A13 = A14 = A15 = 0, $\overline{MREQ}$ = 0 SAÍDA 1
A12 = 1, A11 = A13 = A14 = A15 = 0, $\overline{MREQ}$ = 0 SAÍDA 2

Na entrada $\overline{OE}$ ("output enable") ligamos o sinal $\overline{RD}$, pois quando é realizada uma leitura de memória ($\overline{MREQ}$ = 0), esta saída é ativada pelo Z-80 ($\overline{RD}$ = 0). Conseqüentemente, somente após a ativação do sinal $\overline{RD}$, é que a memória libera a informação para a barra de dados.

Na figura 7.3 temos as faixas de endereços de cada um dos circuitos integrados de memória.

A15	A14	A13	A12	A11	A1Ø - - - - - - - AØ	
0	0				X - - - - - - - X	
		Ø	Ø	Ø	– EPROM Nº 1	ØØØØ - Ø7FF
		Ø	Ø	1	– EPROM Nº 2	Ø8ØØ - ØFFF
		Ø	1	Ø	– EPROM Nº 3	1ØØØ - 17FF

Fig. 7.3 – Endereços do Módulo de EPROM

Qualquer endereço que não esteja dentro da faixa de funcionamento do módulo (0000H – 17FFH) não selecionará o decodificador e, assim, nenhum circuito integrado de memória será ativado.

Na prática, além das considerações lógicas, devemos analisar as compatibilidades elétricas e de tempo entre as memórias e o Z-80.

7.3 – MÓDULO DE 3K BYTES DE RAM ESTÁTICA

Vejamos agora um projeto de 3K bytes de memória RAM, isto é, uma memória de leitura e escrita de dados. Usaremos o circuito integrado 2114 que é uma RAM estática de 1024 posições de 4 bits cada uma. Na figura 7.4 temos o diagrama lógico e os procedimentos de leitura e escrita desta memória.

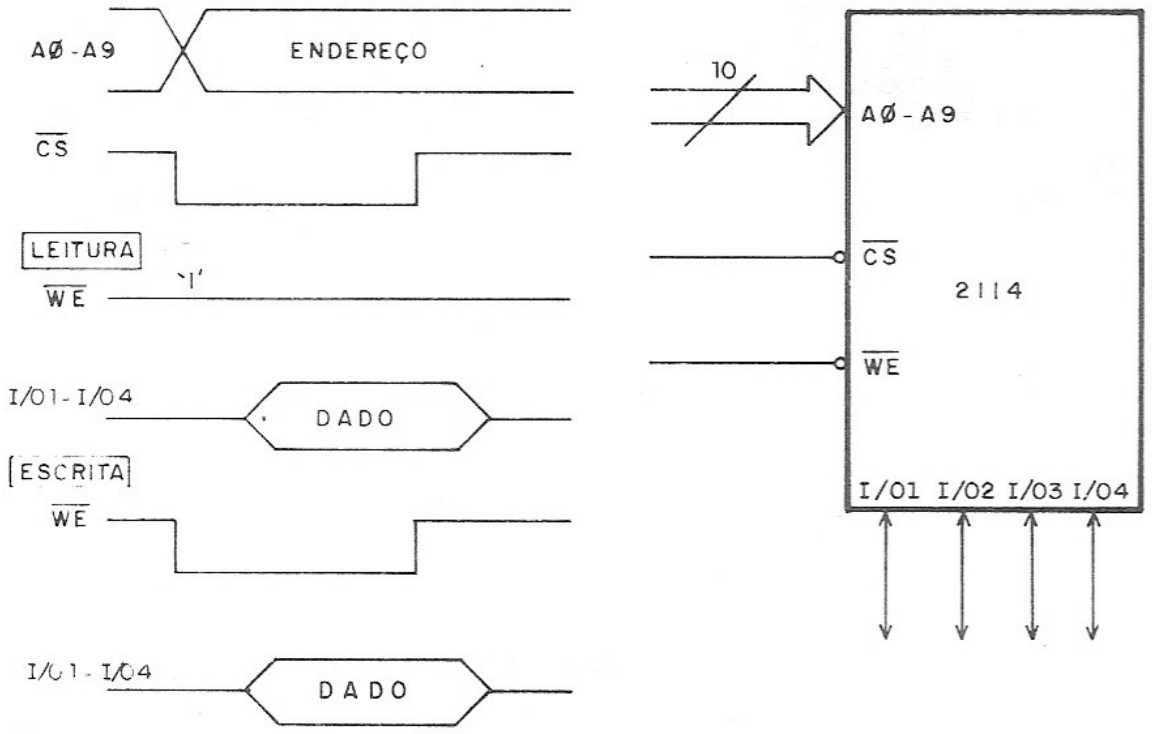


Fig. 7.4 – RAM 2114

Novamente não entraremos em todos os detalhes do diagrama de tempo ("timing") da 2114, pois daremos ênfase a lógica necessária à ligação com o Z-80. Observe que a entrada $\overline{CS}$ ("chip select") é a responsável pela seleção da 2114, e seu estado ativo é o nível 0. A comunicação com a barra de dados é feita através de quatro linhas bidirecionais, controladas pela entrada $\overline{WE}$. Quando $\overline{WE}$ ("write enable") e $\overline{CS}$ forem iguais a zero, o estado das linhas I/01 – I/04 é escrito no endereço especificado pelos bits A0 – A9. Por outro lado, quando $\overline{CS}$ = 0 e $\overline{WE}$ = 1 temos uma leitura de dados no endereço especificado. Na figura 7.5 temos o projeto do módulo de 3K bytes de RAM.

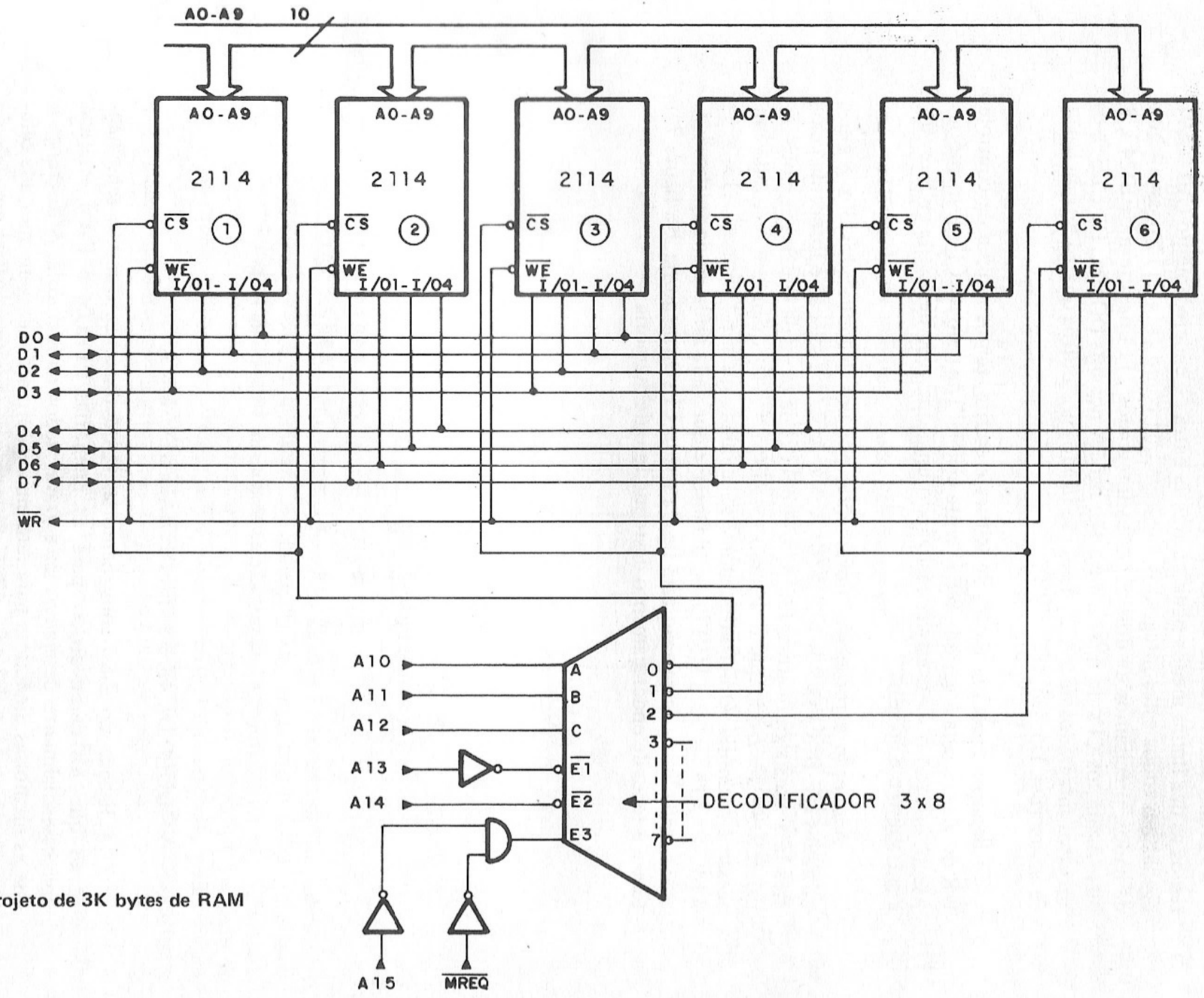


Fig. 7.5 — Projeto de 3K bytes de RAM

Como cada circuito integrado (CI) 2114 é organizado em 1024 x 4 bits, temos as dez linhas menos significativas da barra de endereçamento (A0 – A9) ligadas em todos estes integrados. Observe que como cada 2114 possui 4 bits de dados, precisamos de dois circuitos integrados 2114 para termos 1024 x 8 bits. Assim, os CI's de nºs 1, 3 e 5 são ligados nos bits DO, D1, D2 e D3; e os circuitos integrados de nºs 2, 4 e 6 são ligados nos outros quatro bits da barra de dados (D4, D5, D6, D7).

Sendo a memória RAM, de leitura e escrita, temos a linha de controle $\overline{WR}$ do Z-80 ligada à entrada $\overline{WE}$ ("write enable") de todos os CI's de memória. Assim, quando o Z-80 estiver realizando uma operação de escrita ($\overline{WR}$ = 0), a entrada $\overline{WE}$ é ativada.

A seleção é fornecida, novamente, por um decodificador 3 x 8 cujas entradas (A, B, C) são os bits A10, A11 e A12, e a seleção é fornecida pelos três bits mais significativos (A13, A14 e A15) da barra de endereços e pelo sinal $\overline{MREQ}$ do Z-80, que é ativado ($\overline{MREQ}$ = 0) quando é realizado um acesso à memória. O decodificador é selecionado quanto tivermos: A13 = 1, A14 = A15 = 0 e $\overline{MREQ}$ = 0. Observe que cada saída do decodificador, isto é, cada linha de seleção está ligada a dois CI's de memória. Na figura 7.6 temos a faixa de endereçamento deste módulo de memória.

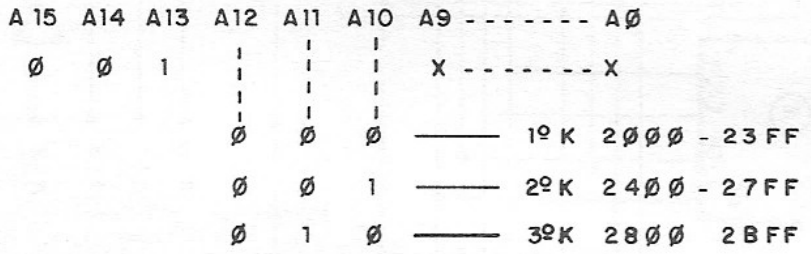


Fig. 7.6 – Endereços do Módulo de 3K bytes de RAM

7.4 – MÓDULO DE 4K BYTES DE RAM DINÂMICA

A ligação do Z-80 com memórias RAM dinâmica é muito semelhante ao que foi desenvolvido para as memórias EPROM e RAM estática, pois essencialmente desejamos ler e gravar dados na memória. A diferença é que as memórias dinâmicas para reterem a informação, exigem que cada célula sofra um *refresh periódico*. Tipicamente, isto significa realizar um ciclo de leitura em cada célula a cada 2 ms. Logo, é necessário um *hardware adicional* que além de permitir a consecução de acessos normais de leitura e escrita de dados, realize também a função do refresh periódico.

O Z-80 comparado com outros microprocessadores de 8 bits, oferece uma grande vantagem na ligação com memórias dinâmicas. Isto porque, através do registrador R e da linha $\overline{RFSH}$ (pino 28), ele oferece um refresh automático. Nos ciclos de busca de instrução (M1), quando o Z-80 está realizando operações internas (Ex: decodificando a instrução), o conteúdo do registrador R é colocado nos 7 bits menos significativos (A0 – A6) da barra de endereçamento. Este evento é informado ao meio externo, através da ativação dos sinais $\overline{RFSH}$ e $\overline{MREQ}$.

Assim, como são usados 7 bits do registrador R e a cada ciclo M1 ele é incrementado de uma unidade, temos uma repetição do conteúdo do registrador R, no máximo a cada 128 ciclos M1.

Vejamos o exemplo de um projeto de um módulo de 4 Kbytes de memória RAM dinâmica. Utilizaremos um circuito integrado que possui 4K x 1 bits. Logo, para termos 4 Kbytes utilizaremos 8 destes integrados. Na figura 7.7 temos o diagrama lógico desta memória de 4K x 1.

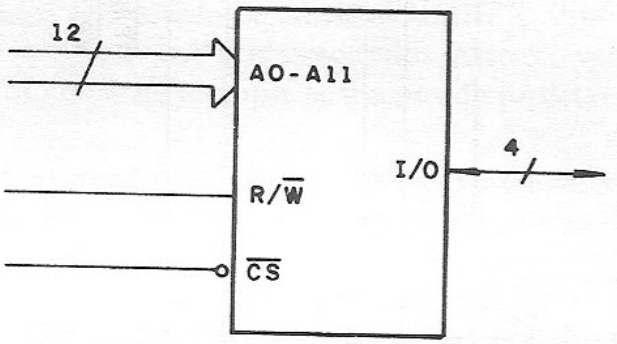


Fig. 7.7 – RAM Dinâmica de 4K x 1

Os 12 bits de endereçamento estão ligados internamente às linhas e as colunas da matriz da memória. As linhas são endereçadas por 6 bits, totalizando 64 linhas. O mesmo ocorre com as colunas, que totalizam 64. Para realizarmos o refresh, isto é, para que a informação permaneça armazenada na memória, precisamos acessar as 64 linhas a cada 2 ms. Isto não pode ser garantido por acessos randômicos realizados durante a execução de um programa. Para tanto, precisamos implementar essa leitura periódica.

A entrada $R/\overline{W}$ indica leitura ($R/\overline{W} = 1$) ou escrita ($R/\overline{W} = 0$) na memória.

Quando a entrada $\overline{CS}$ ("chip select") é ativada ($\overline{CS} = 0$), a memória fica selecionada.

A entrada e saída de dados é feita através da linha I/O, que é controlada pela linha $R/\overline{W}$.

Na figura 7.8 temos o diagrama elétrico de um módulo de 4 Kbytes, cuja faixa de endereçamento é 2000H – 2FFFH.

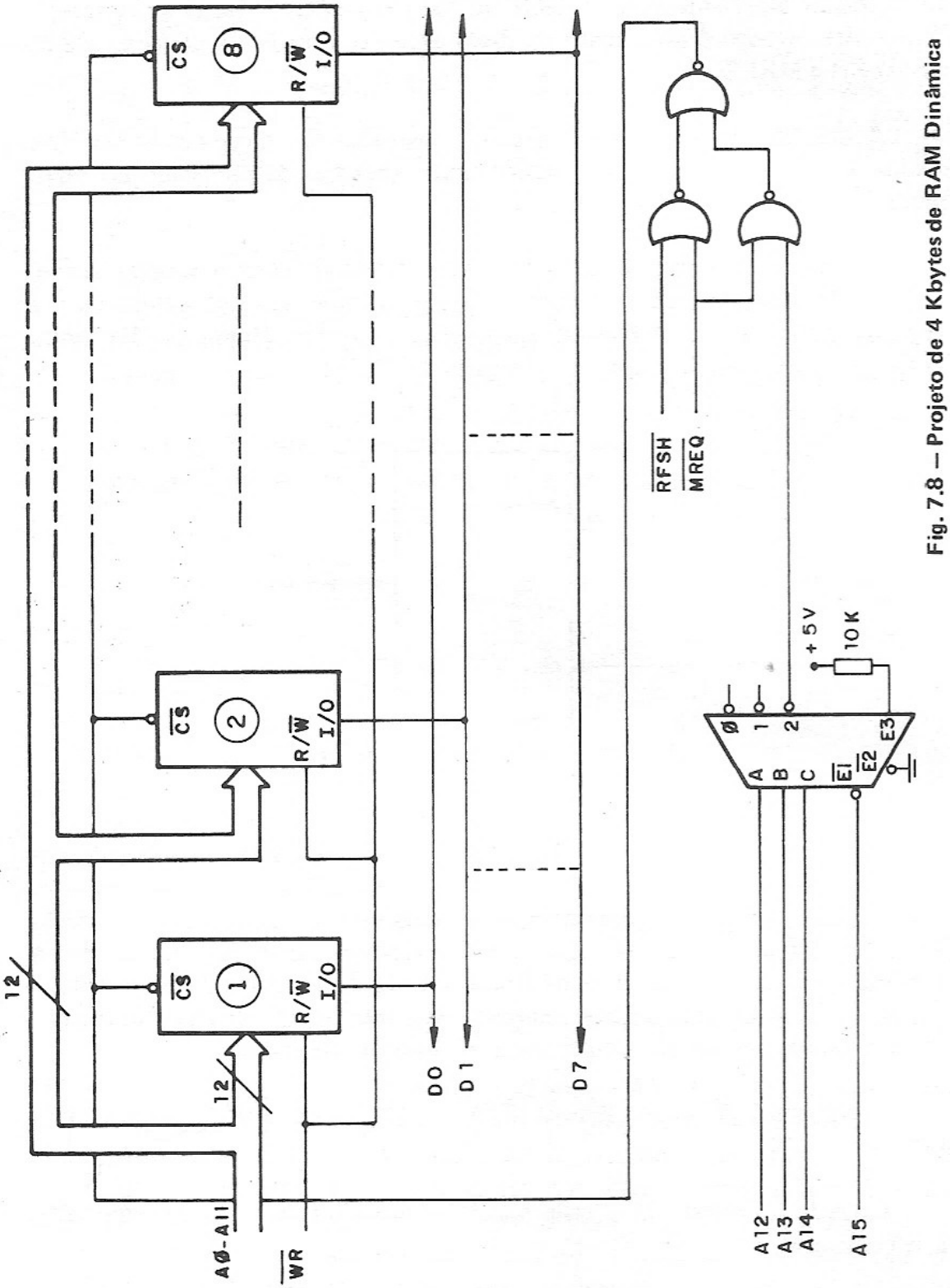


Fig. 7.8 – Projeto de 4 Kbytes de RAM Dinâmica

Observe que cada circuito integrado de memória, por ser organizado em 4K x 1, fornece um bit do total de 8 bits da barra de dados.

Na entrada R/$\overline{W}$ foi ligado o sinal $\overline{WR}$, que na escrita assume o nível lógico baixo.

A seleção do módulo de 4 Kbytes é realizada por uma lógica que fornece um nível 0 na entrada $\overline{CS}$, nas seguintes condições:

1 – Quando $\overline{MREQ}$ = 0, A15 = A14 = A12 = 0 e A13 = 1, temos a seleção para a leitura ($\overline{WR}$ = 1) ou escrita ($\overline{WR}$ = 0) de dados.

2 – Quando $\overline{RFSH}$ = $\overline{MREQ}$ = 0, temos o refresh, pois na barra de endereçamento (A0 – A6) está o conteúdo do registrador R que vai selecionar as linhas da matriz interna (A0 – A5).

Considerando um tempo médio de execução de uma instrução igual a 2,5 microsegundos (10 estados e clock de 4MHZ), temos para 64 ciclos M1, um total de 64 x 2,5 = 160 microsegundos. Este é o valor médio do ciclo de refresh do módulo de 4 Kbytes que acabamos de projetar.

Na prática, as memórias RAM's dinâmicas possuem entradas de endereçamento multiplexadas no tempo. Isto é, o endereço tem que ser fornecido em duas vezes, e o circuito integrado possui duas seleções para capturar, nos pinos de entrada de endereçamento, o endereço da linha e o da coluna. Estas seleções são o RAS (Row adress select) e o CAS (Column adress select). Apesar de complicar o hardware de interface entre o Z-80 e a memória, a multiplexação permite uma economia de pinos do CI. Isto é importante, principalmente porque cada vez mais o processo de integração em larga escala vem sendo aprimorado e, com isso, permite que mais transistores sejam colocados em um único chip.

No entanto, o projeto de módulos de memória RAM dinâmica com estes CI's que possuem entradas de endereçamento multiplexadas, não diferem muito do projeto anteriormente realizado por nós.

7.5 – INTERFACE DE ENTRADA DE DADOS

Vejamos o projeto de uma interface que permite a leitura do estado de 8 chaves. Vamos supor que o endereço desta porta de entrada é FEH (HEX). Na figura 7.9 temos o diagrama elétrico desta interface de entrada.

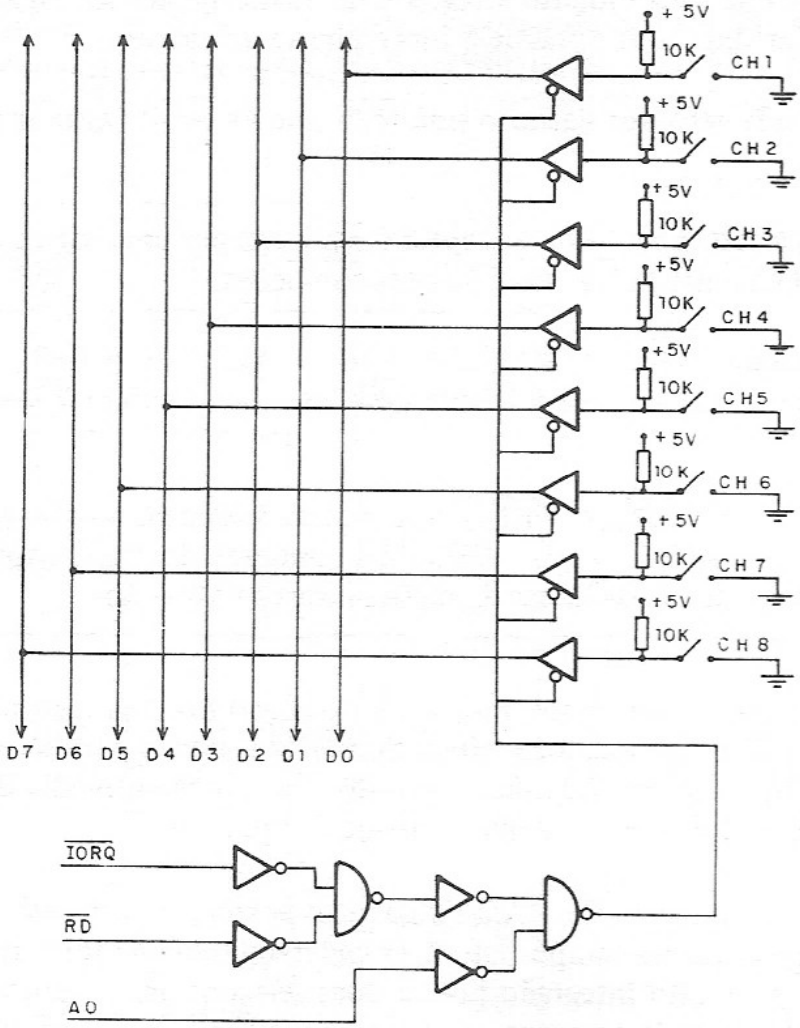


Fig. 7.9 – Interface de Entrada

Observe que as chaves (CH1 – CH8) estão ligadas a circuitos do tipo *tri-state* que, por sua vez, estão conectados à barra de dados (D0 – D7).

A lógica de seleção dos circuitos do tipo tri-state foi realizada usando-se somente NANDS e INVERSORES. Poderia ser simplificada se fossem utilizados circuitos do tipo NOR. Esta seleção é função do estado das linhas $\overline{IORQ}$, $\overline{RD}$ e A0.

Para melhor entendermos esta interface de entrada, vejamos a instrução IN A, (FE). Esta instrução carrega o acumulador (A) com o conteúdo da porta de entrada cujo endereço é FEH (hexadecimal). Assim, quando ela é executada temos os sinais $\overline{IORQ}$ e $\overline{RD}$ no nível 0, e o valor FEH nos oito bits menos significativos da barra de endereçamento. Note que no valor FEH temos somente o bit A0 no nível 0. Por isso, usamos na lógica de seleção o bit A0 e os sinais de controle $\overline{IORQ}$ e $\overline{RD}$.

Com isso, quando tivermos A0 = $\overline{IORQ}$ = $\overline{RD}$ = 0, a seleção do tri-state vai para o estado zero e o conteúdo das chaves é colocado na barra de dados, sendo em seguida transferido para o acumulador.

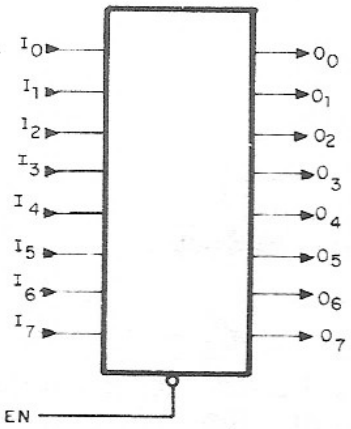
Este tipo de seleção que não utiliza decodificador é muito usado em pequenos sistemas, pois podemos usar até 8 portas com endereços:

FE (A0 = 0), FD (A1 = 0), FB (A2 = 0), F7 (A3 = 0), EF (A4 = 0), DF (A5 = 0), BF (A6 = 0) e 7F (A7 = 0).

Esta técnica de seleção é chamada de *endereçamento linear.*

7.6 – INTERFACE DE SAÍDA DE DADOS

Apresentaremos o projeto de uma porta de saída, na qual estão ligados oito (8) diodos emissores de luz (LED's). Antes, porém, é necessário apresentarmos um circuito do tipo LATCH, conforme mostra a figura 7.10.



EN	I0-I7	O0 – O7
ø	ø	ø
ø	1	
1	x	1

X – DON'T CARE

Fig. 7.10 – LATCH de 8 bits

Quando a entrada $\overline{EN}$ for igual a zero, os valores das entradas I0 – I7 são transferidos para as saídas O0 – O7. Se a entrada $\overline{EN}$ for igual a 1, independente dos valores das entradas I0 – I7, os valores das saídas O0 – O7 permanecerão inalterados.

Na figura 7.11 temos uma interface com endereço FEH, que permite atuar em 8 LED's.

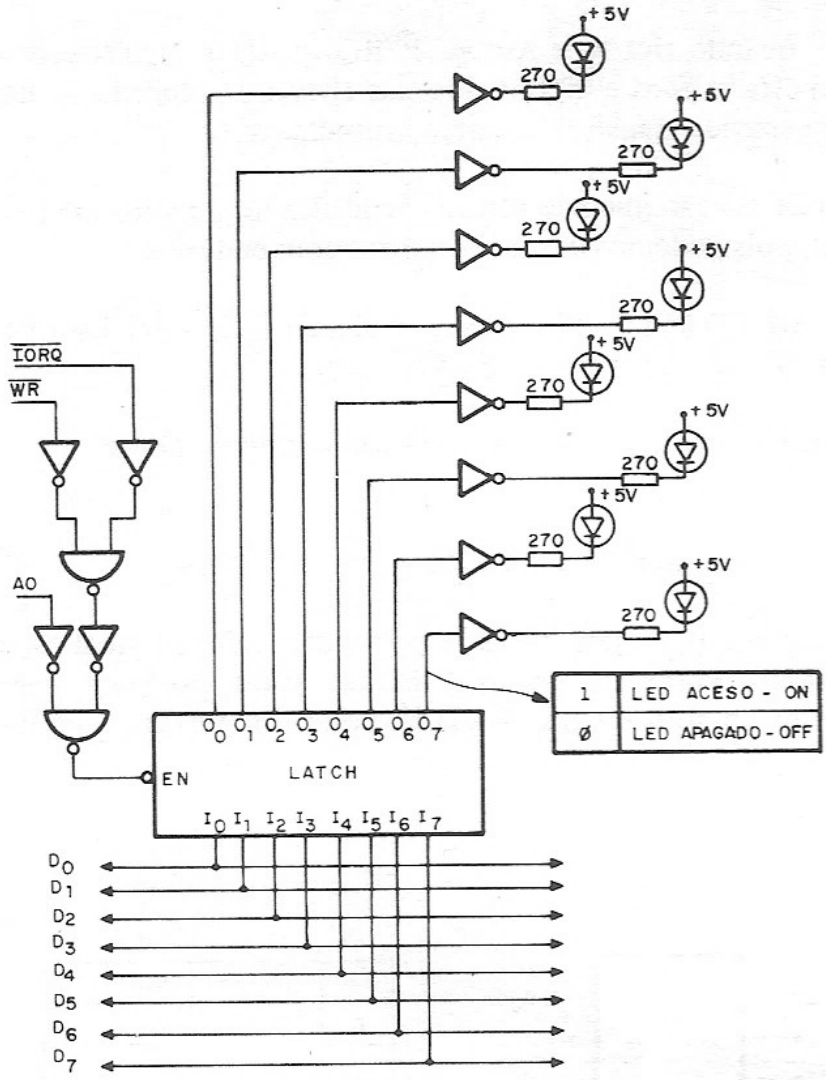


Fig. 7.11 – Interface de Saída

Quando a saída do inversor for igual a zero, teremos uma corrente de aproximadamente 12,5 mA através do LED, suficiente para acendê-lo. Isto é possível, pois a corrente IOL MAX de um TTL é 16 mA. Assim, quando tivermos um nível lógico "1" em uma das saídas O0 – O7, o LED correspondente entra no estado ON (aceso).

As entradas I0 – I7 do LATCH estão ligadas na barra de dados e irão transferir o conteúdo desta barra para as saídas O0 – O7, quando a entrada $\overline{EN}$ for igual a zero. Esta entrada ($\overline{EN}$) recebe um sinal proveniente de uma lógica envolvendo os sinais $\overline{IORQ}$, $\overline{WR}$ e AO.

Para melhor compreendermos esta interface de saída, vejamos a instrução de saída de dados OUT (FE), A. Esta instrução transfere o conteúdo do acumulador para a porta de saída cujo endereço é igual a FEH. Quando esta

instrução é executada, os sinais $\overline{IORQ}$ e $\overline{WR}$ são ativados ($\overline{IORQ}$ = $\overline{WR}$ = 0), e nos oito bits menos significativos da barra de endereçamento temos o valor FEH. Como o endereçamento é *linear* e o seu valor é FEH, também usamos o bit A0 na lógica de seleção do LATCH.

7.7 – ENTRADA/SAÍDA MAPEADA

Este modo de interface de entrada/saída é caracterizado pelo endereço do dispositivo. Este endereço ocupa uma posição de memória, pois são usados na seleção do dispositivo os sinais $\overline{MREQ}$, $\overline{RD}$, $\overline{WR}$ e a barra de endereçamento (A0 – A15).

Assim, podemos usar um número de dispositivos de E/S bem maior do que 256, pois agora utilizamos os 16 bits da barra de endereçamento.

Uma outra vantagem é que a comunicação com os dispositivos passa a ser feita através das inúmeras instruções de transferência de dados entre registradores e memórias, e não, somente, via instruções IN e OUT.

A desvantagem é que são usados endereços reservados para a memória, e se a lógica de seleção não for bem implementada, podemos perder inúmeros endereços válidos.

Na figura 7.12 temos um exemplo de E/S mapeada. É um teclado organizado em uma matriz 3 x 8, totalizando 24 teclas.

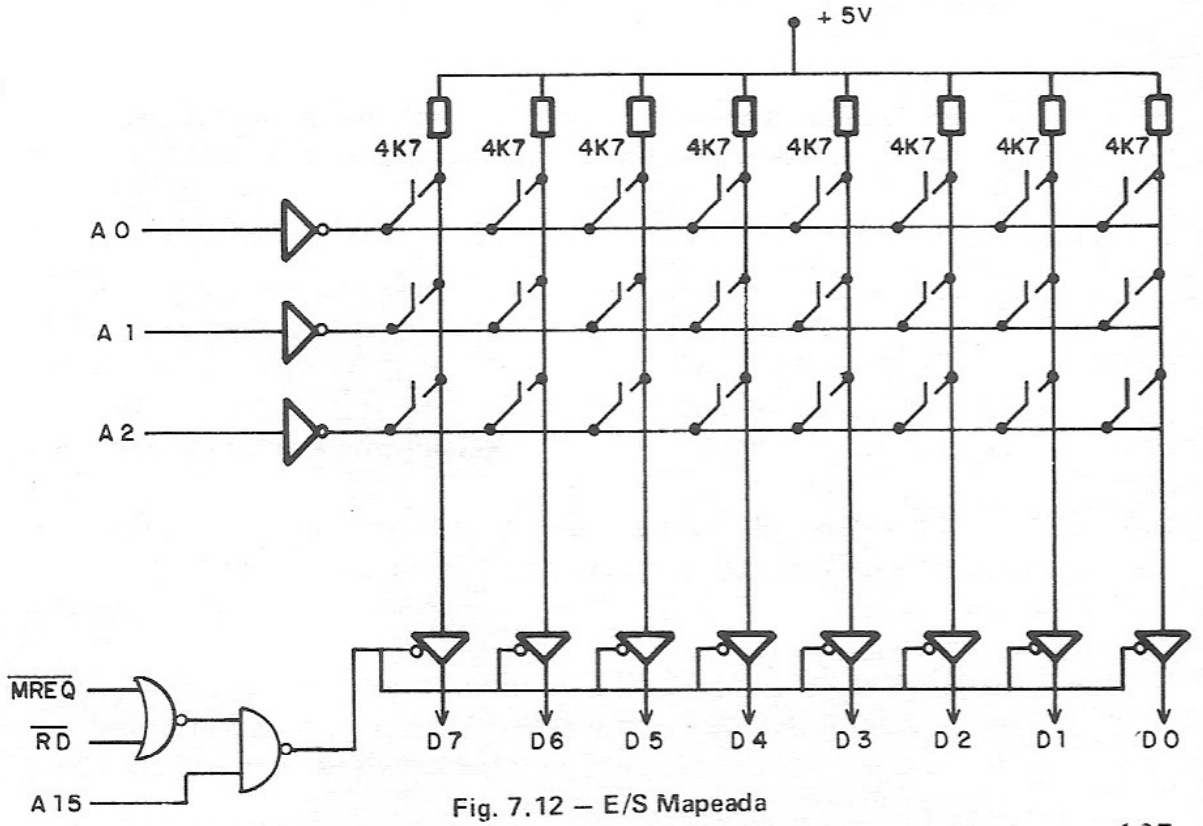


Fig. 7.12 – E/S Mapeada

Observe que a seleção do tri-state é ativada quando A15 = 1, $\overline{MREQ}$ = 0 e $\overline{RD}$ = 0. Assim, uma leitura de memória cujo endereço tenha A15 = 1, irá habilitar o tri-state que, conseqüentemente, colocará o estado das colunas da matriz na barra de dados (D0 – D7).

A seleção das linhas da matriz é feita através dos bits A0, A1 e A2 da barra de endereçamento. Quando uma linha for selecionada (Ex: A0 = 1), a saída do inversor correspondente vai para o nível 0. Então, se tivermos uma tecla acionada, a coluna correspondente ficará no nível 0.

Assim, os endereços válidos para este teclado são:

	A15			A0
LEITURA 1ª LINHA	1XXX	XXXX	XXXX	X001
LEITURA 2ª LINHA	1XXX	XXXX	XXXX	X010
LEITURA 3ª LINHA	1XXX	XXXX	XXXX	X100

Os problemas de ruído e de teclas acionadas simultaneamente são solucionados por software.

Observe que neste exemplo, qualquer endereço com A15 = 1 ativa a seleção do tri-state. Assim, só podemos usar memórias na faixa de endereços 0000H – 7FFFH, pois esta é a faixa em que A15 = 0. Uma maneira de resolver este problema é usarmos um decodificador para a seleção do tri-state.

7.8 – EXERCÍCIOS DE FIXAÇÃO

7.8.1 – Sabendo-se que o tempo de acesso da 2716 é 350 ns, com auxílio do apêndice A, determine se é necessário o uso do WAIT ao usarmos o Z-80B.

7.8.2 – Sabendo-se os seguintes parâmetros máximos da memória 2716:

IOL = 2,1 mA IOH = 400μA Capacitância de entrada = 6pF
IIL = 10μA IIH = 10μA Capacitância de saída = 12pF
Corrente no 3º estado = ± 10μA.

E, os seguintes parâmetros máximos do Z-80:

IOL = 1,8 mA, IOH = 250μA, IIL = 10μA, IIH = 10μA
Capacitância de entrada = 5pF, Capacitância de saída = 10pF

E, ainda, admitindo-se que o tempo de acesso da 2716 utilizado com o Z-80B é suficiente para não usarmos o WAIT, desde que:

a) Nas saídas de cada 2716 tenhamos uma carga capacitiva inferior a 200pF.

b) Nas saídas do Z-80 tenhamos uma carga capacitiva inferior a 50pF.

Quantas memórias 2716 podemos ligar ao Z-80B sem a necessidade de utilizarmos buffers?

7.8.3 – Faça um projeto de um módulo de 16K bytes de memória EPROM com a 2716. A faixa de endereçamento deve ser de 0000H a 3FFFH.

7.8.4 – Faça um projeto de um módulo de 16K bytes de RAM com a 2114. A faixa de endereçamento deve ser de 4000H e 7FFFH.

7.8.5 – Quando resetamos o Z-80 os ciclos de refresh são paralisados. Dê uma solução para este problema.

7.8.6 – Quando realizamos operações de DMA através do sinal $\overline{\text{BUSREQ}}$, os ciclos de refresh são paralisados. Dê uma solução para este problema.

7.8.7 – Faça um projeto de um módulo de 32K bytes de memória RAM dinâmica, utilizando a mesma memória da seção 7.4. A faixa de endereçamento deve ser de 8000H a FFFFH.

7.8.8 – Faça um projeto de uma porta de entrada que permita a leitura do estado de 16 chaves. Os endereços são FEH e FDH.

7.8.9 – Faça um projeto de uma porta de saída (LATCH) com endereço 01. (Usar decodificador).

7.8.10 – O que é Entrada/Saída mapeada?

8 PROJETO DE UM MICROCOMPUTADOR

8.1 – APRESENTAÇÃO

Neste capítulo faremos o projeto de um microcomputador de uso geral que denominaremos *Micro Z-80.*

O *Micro Z-80* utiliza a PIO (Z8420) como dispositivo de entrada e saída e as memórias *RAM 6116* e *EPROM 2716 e 2732.*

Para tanto, estudaremos estes componentes separadamente e, em seguida, realizaremos o projeto do *Micro Z-80.*

8.2 – A PIO

A PIO – Parallel Input/Output Controller é um circuito integrado de 40 pinos, compatível com o Z-80, que facilita a sua ligação com dispositivos periféricos paralelos. A figura 8.1 mostra o diagrama lógico da PIO.

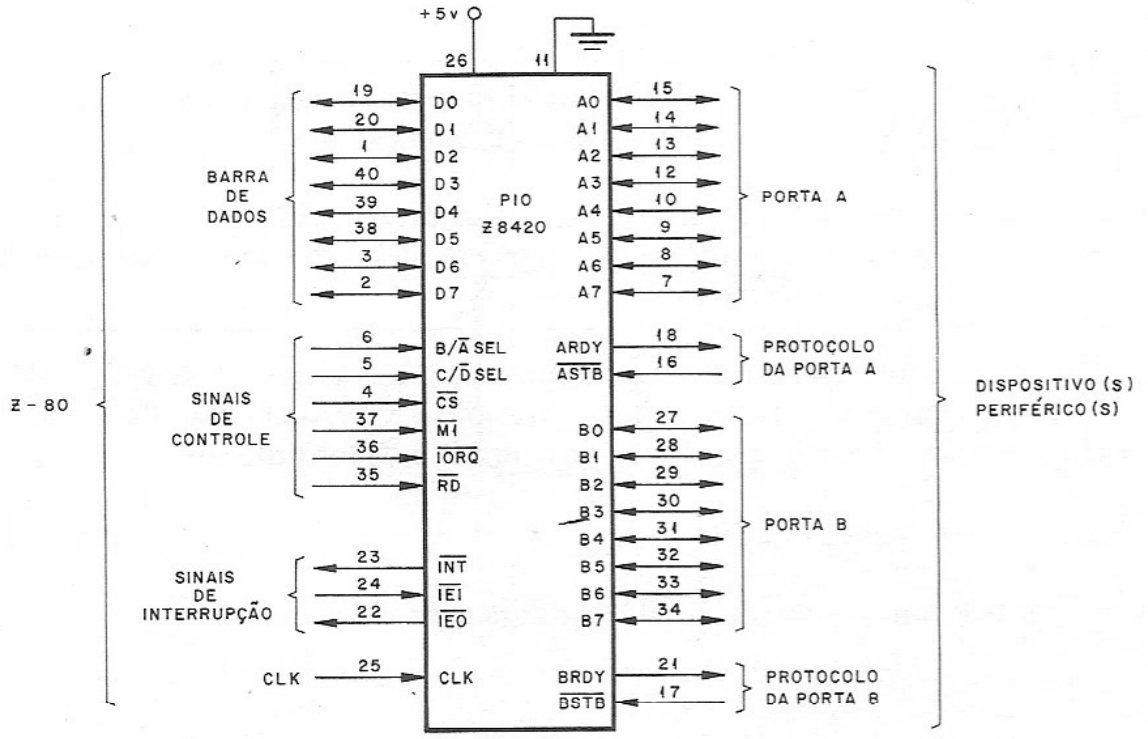


Fig. 8.1 – Diagrama Lógico da PIO

Existem dois conjuntos de 8 bits (A0 – A7 e B0 – B7), através dos quais trafegam os dados na comunicação com os periféricos. Eles podem ser programados para operar como *entrada ou saída.* O primeiro grupo de 8 bits denominados de *porta A* e o segundo de *porta B.* Associado a cada porta existem duas linhas, RDY e $\overline{STB}$, usadas para o protocolo ("handshaking") entre o periférico e a PIO.

Os dados procedentes do Z-80 ou a ele destinados, trafegam através da *barra de dados* (D0 – D7).

Seis sinais de controle procedentes do Z-80 controlam a operação da PIO, a partir do programa em execução. A linha B/$\overline{A}$ SEL seleciona a porta A (B/$\overline{A}$ SEL = 0) ou porta B (B/$\overline{A}$ SEL = 1). Uma outra entrada C/$\overline{D}$ SEL indica se a transferência é de um dado (C/$\overline{D}$ SEL = 0) ou de um controle (C/$\overline{D}$ SEL = 1). A linha $\overline{CS}$ (chip select) indica à PIO que o seu endereço foi decodificado numa operação de entrada ou saída. Através da linha $\overline{M1}$ chega o sinal $\overline{M1}$ gerado pelo Z-80 e, através da linha CLK chega o sinal do relógio de sincronização. $\overline{IORQ}$ e $\overline{RD}$ são os sinais do Z-80 relativos as operações de entrada e saída.

O controle de interrupções é feito através das linhas $\overline{INT}$, $\overline{IEI}$ e $\overline{IEO}$, conforme foi estudado no capítulo 6.

8.2.1 – Dados e Controles

A comunicação entre o Z-80 e a PIO se faz através de dois tipos de informações:

1 – Os dados que são efetivamente transmitidos (saída) ao dispositivo periférico ou que são dele lidos (entrada).

2 – Os controles que determinam as características operacionais da PIO e que denominaremos, também, de *comandos operacionais* ou *palavras de controle.*

A identificação do tipo de informação se faz através do sinal C/$\overline{D}$ SEL e cada porta recebe seus comandos operacionais separadamente. Para tanto, cada porta tem os seus registradores de controle e dados próprios.

8.2.2 – A Arquitetura da PIO

O diagrama de blocos da PIO está mostrado na figura 8.2

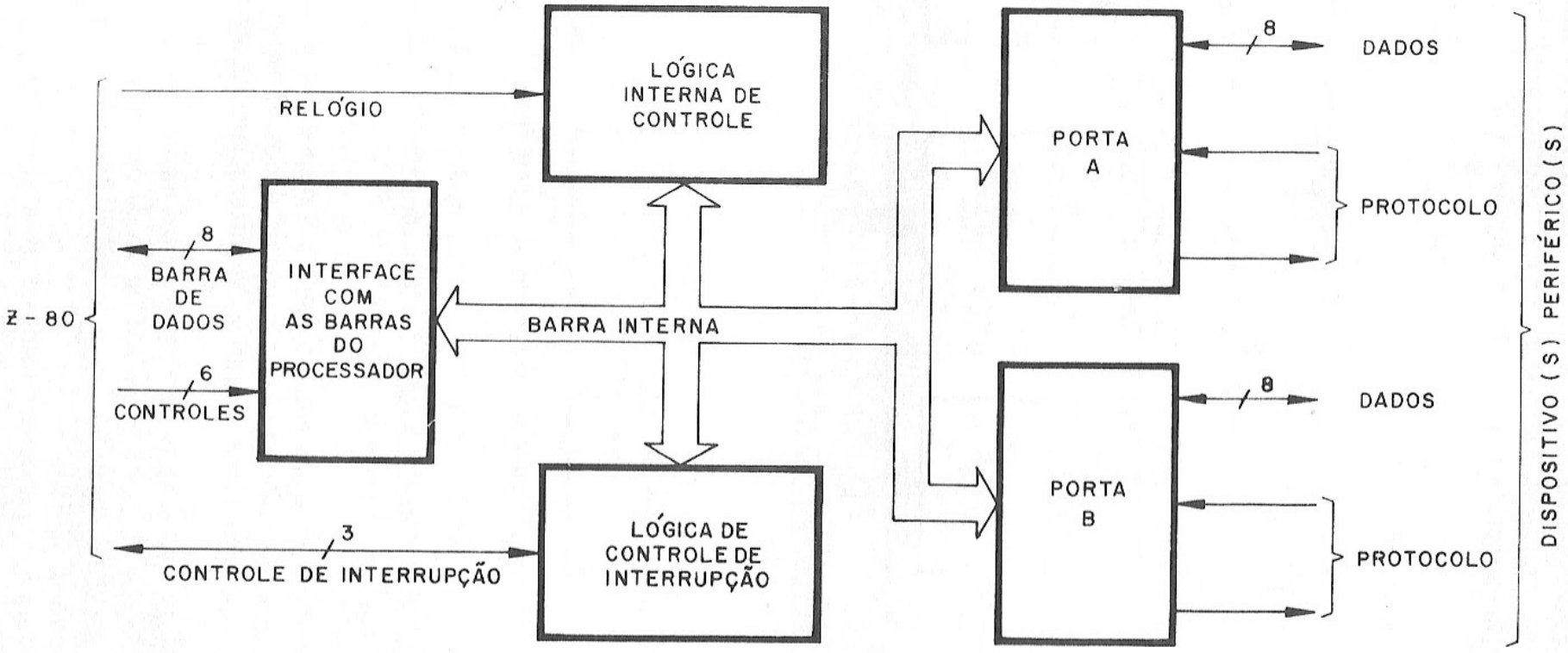


Fig. 8.2 – Diagrama de Blocos da PIO

Os dados de entrada ou saída e os comandos operacionais procedentes do microprocessador ou a ele destinados, trafegam através da barra de dados, que se ramifica no interior da PIO, estabelecendo a comunicação entre a lógica de controle operacional, de interrupção e as portas A e B.

8.2.3 – A Arquitetura das Portas

O diagrama de blocos de uma porta (A ou B) com seus controles associados está mostrado na figura 8.3.

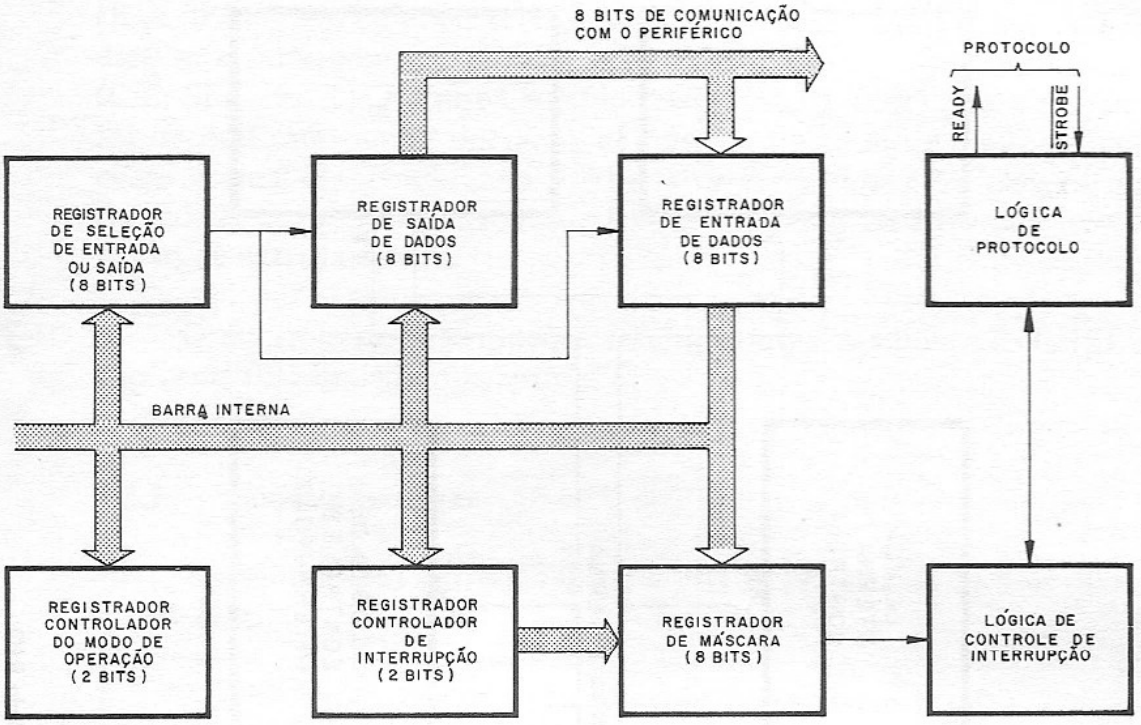


Fig. 8.3 – Diagrama de Blocos de uma Porta

Cada porta da PIO tem uma lógica e uma série de registradores próprios, que se comunicam através da barra interna.

A troca de informações entre o Z-80 e o dispositivo periférico se faz através dos *registradores de entrada e saída* de dados. O *registrador de seleção de entrada ou saída* controla a ativação destes registradores de dados.

O *registrador controlador do modo de operação* determina as características operacionais da porta ou o seu *modo de operação*. Este registrador tem dois bits que são programados quando o Z-80 endereça o registrador controlador do modo ($C/\overline{D}$ SEL = 1 e $\overline{CS}$ = 0) e envia a *palavra de controle* mostrada na figura 8.4, que denominamos de *palavra de modo*.

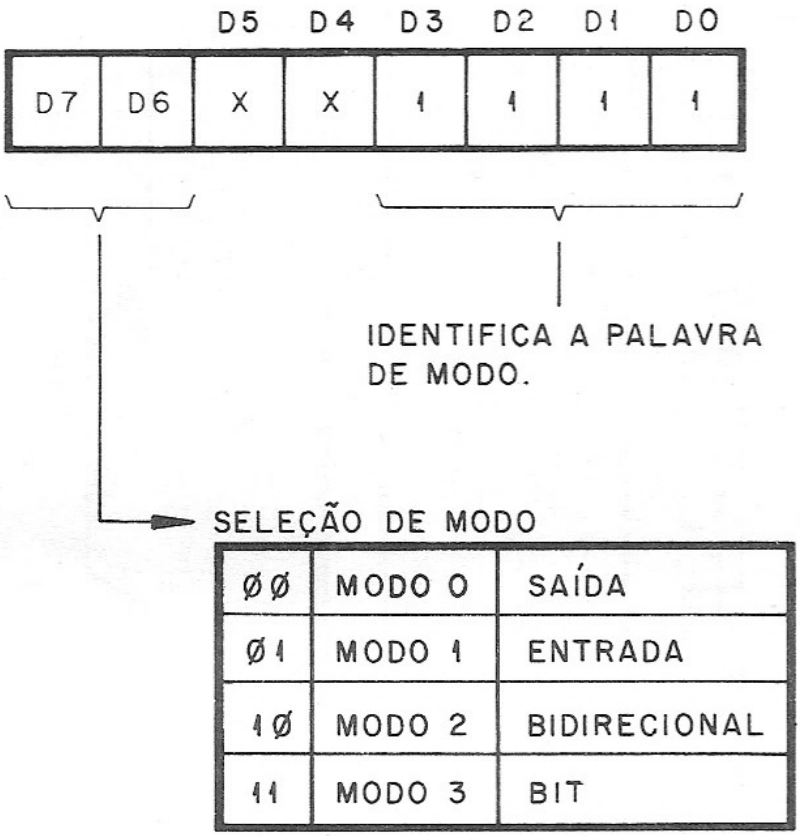


Fig. 8.4 – Palavra de Modo

Observe que cada combinação de D7 e D6 determina um dos *quatro modos de operação* da PIO. O modo de operação de cada porta é independente.A porta A pode operar nos modos 0, 1, 2 ou 3, e a porta B só pode operar nos modos 0, 1 ou 3.

8.2.4 – Modo 0

No *Modo 0,* as portas A ou B operam como saída. As operações de transferência de dados para os registradores de saída da PIO são feitas através das instruções de saída de dados (OUT). O conteúdo dos registradores de saída também podem ser lidos pelo Z-80 através das instruções de entrada (IN).

Quando o Z-80 transfere um dado para o registrador de saída de uma

porta, a linha RDY associada é ativada (RDY = 1), indicando ao periférico que um dado está disponível na porta de saída. Após a leitura do dado, o periférico responde com um pulso através da linha $\overline{STB}$, desabilitando o sinal RDY (RDY = 0) e gerando uma interrupção, se a PIO estiver programada para tal. A figura 8.5 mostra o diagrama de tempo do Modo 0.

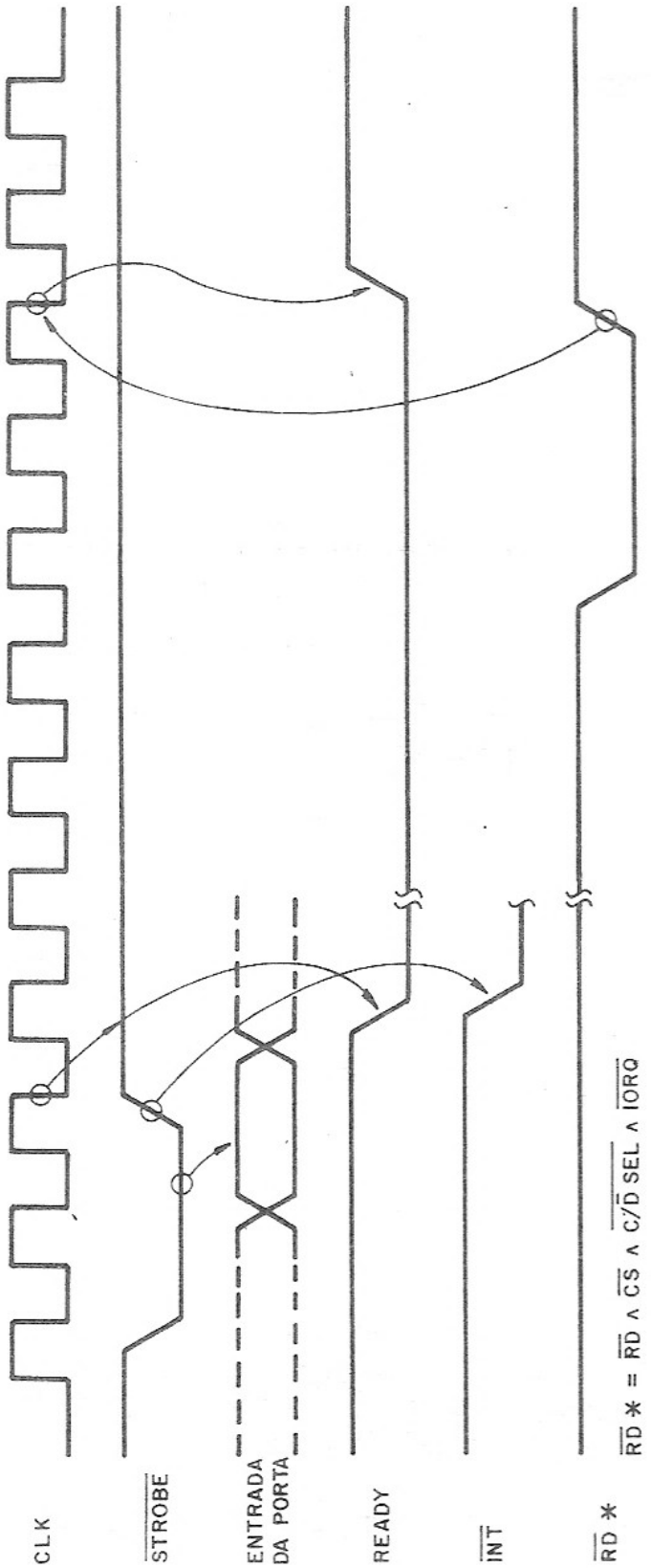


Fig. 8.5 – Diagrama de Tempo do Modo 0

8.2.5 – Modo 1

No Modo 1, as portas A ou B operam como entrada. A seqüência de operações de entrada no Modo 1 está esquematizada na figura 8.6.

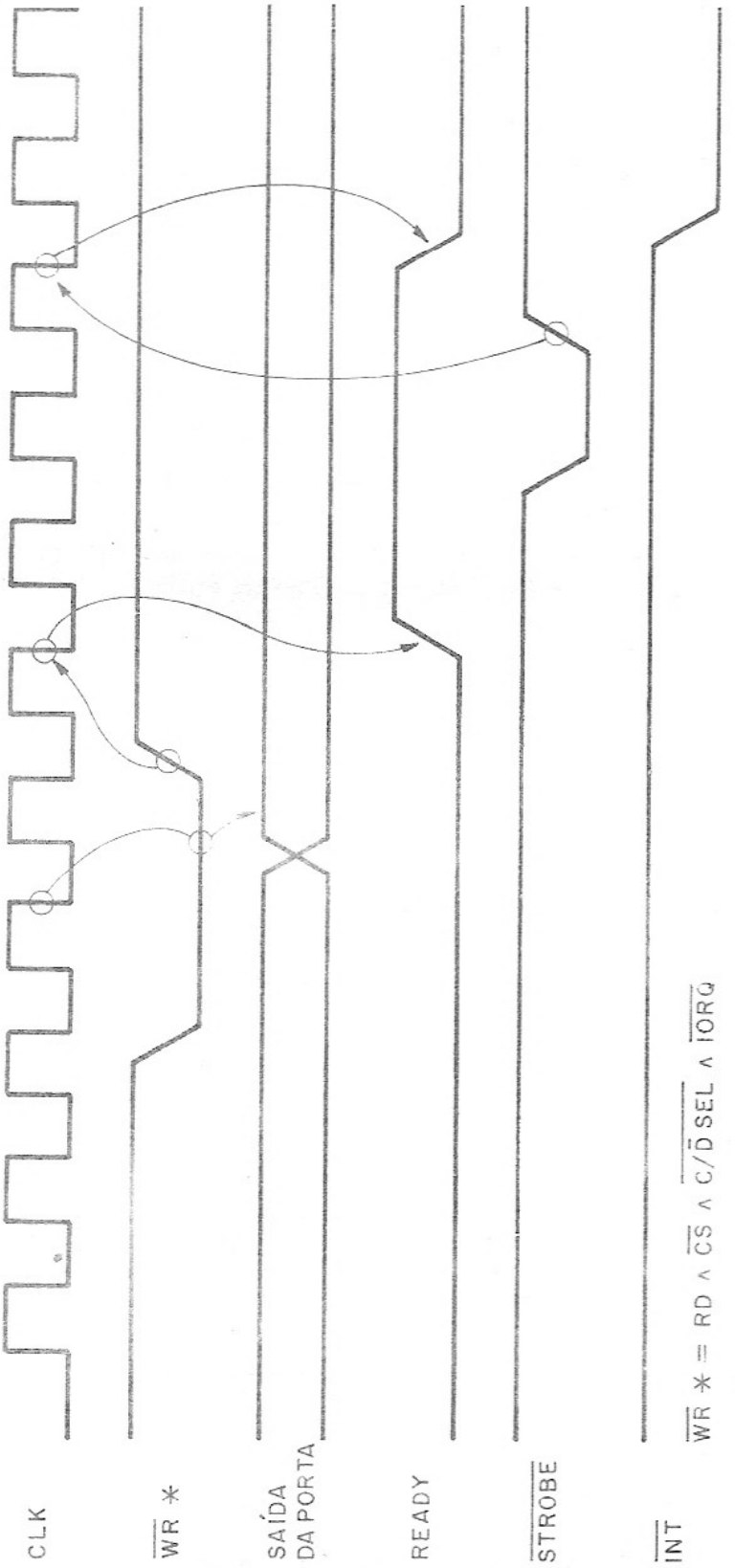


Fig. 8.6 – Diagrama de Tempo do Modo 1

O dispositivo periférico testa a linha RDY. Se ela estiver no nível lógico alto, o periférico coloca o dado nas 8 linhas paralelas e gera um pulso através da linha $\overline{STB}$. O dado é então armazenado no registrador de entrada de dados da PIO. Isto faz com que o sinal RDY vá para o nível lógico baixo, indicando ao dispositivo periférico que o registrador de entrada da PIO tem um dado para ser lido pelo Z-80. Além disso, a PIO gera uma interrupção se estiver programada para tal.

O Z-80 lê o dado da PIO através de uma instrução de IN. Ao final da instrução IN, a PIO recoloca a linha RDY no nível lógico alto, indicando para o dispositivo periférico que um novo dado pode ser transferido para o registrador de entrada de dados da PIO.

8.2.6 – Modo 2

No Modo 2 o tráfego de dados é bidirecional. Para tanto, são necessárias quatro linhas de protocolo e somente a *porta A* pode ser usada neste modo.

As linhas de protocolo da porta A são usadas nas operações de saída e as linhas de protocolo da porta B nas operações de entrada.

Quando a linha $\overline{ASTB}$ estiver baixa, um dado estará sendo transferido do registrador de dados da porta A para o dispositivo periférico e, quando a linha $\overline{BSTB}$ estiver baixa processa-se a operação inversa. A figura 8.7 mostra o diagrama de tempo do Modo 2.

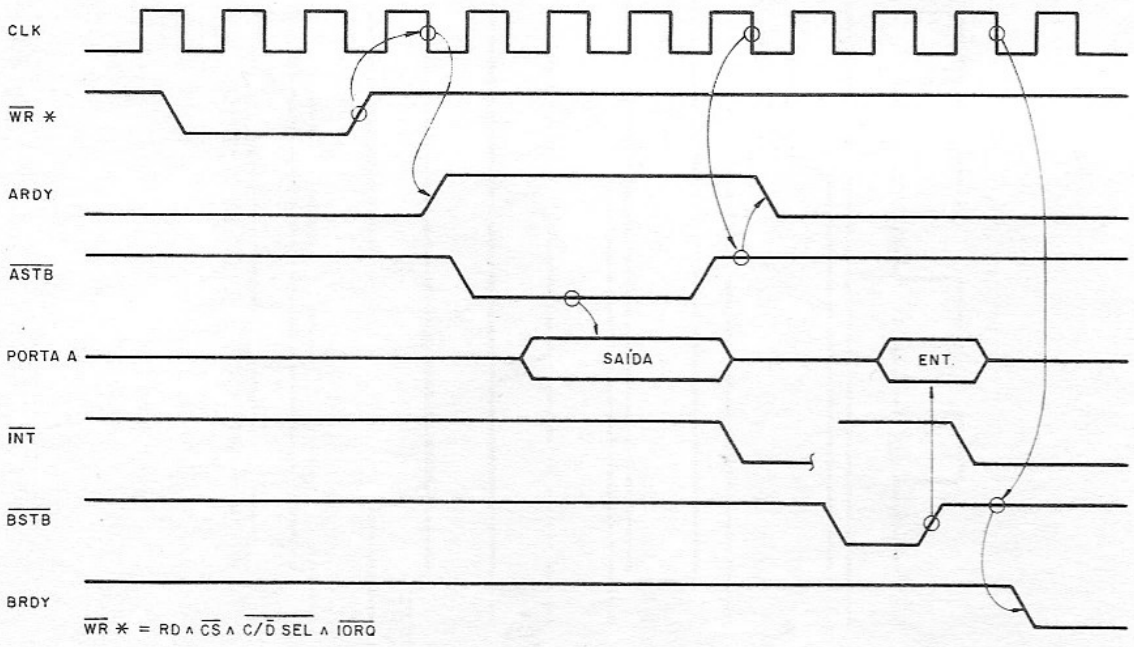


Fig. 8.7 – Diagrama de Tempo do Modo 2

8.2.7 – Modo 3

No Modo 3, cada linha pode ser programada individualmente como entrada ou saída. As linhas de protocolo, Ready e Strobe, não são usadas. Interrupções podem ser programadas para serem geradas se uma entrada mudar ou se todas mudarem.

Após a programação do registrador controlador do modo de operação para o Modo 3, um segundo byte de controle tem que ser enviado à PIO. Este byte define as linhas que vão operar como entrada ou saída, conforme mostrado na figura 8.8. Se o bit for zero significa que a linha associada vai operar como saída.

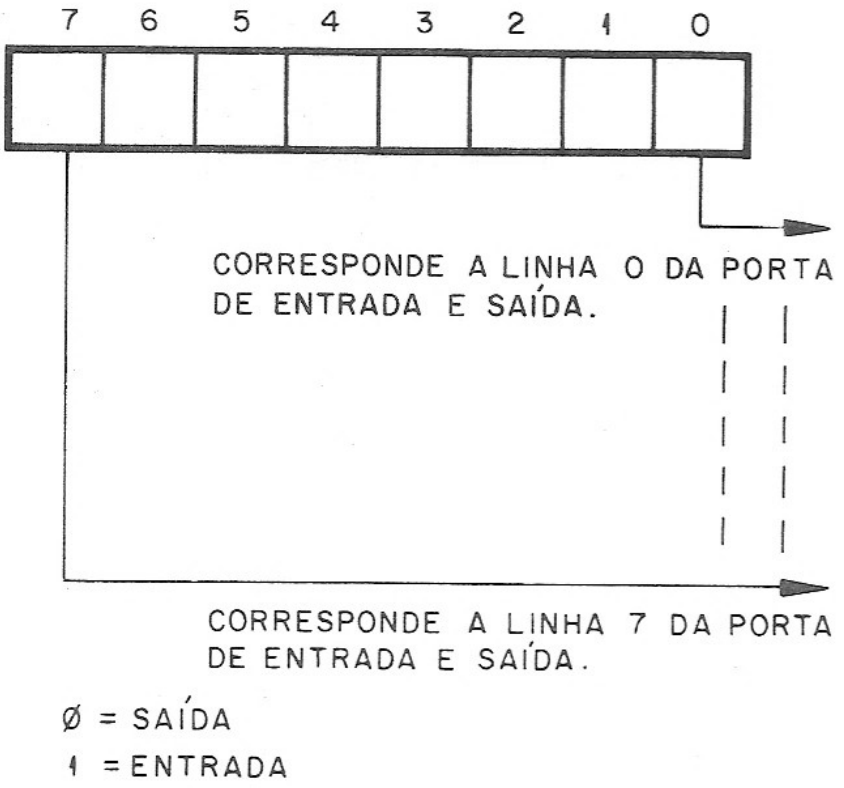


Fig. 8.8 – Programação de Entrada e Saída do Modo 3

Uma vez inicializado o Modo 3, os dados podem ser lidos ou escritos nos registradores de entrada e saída da PIO sem restrições, pois não há sinais de protocolo.

Os dados transferidos do Z-80 para a PIO afetam apenas as linhas programadas como saída, enquanto que os dados lidos pelo Z-80 fornecem informações de todas as linhas.

8.2.8 – Interrupções na PIO

8.2.8.1 – Vetor de Interrupção

Cada porta da PIO pode ser programada para provocar uma interrupção no Z-80, tanto para operações de entrada como saída. A PIO somente pode operar com o Z-80 no Modo 2 de atendimento de interrupção.

Na figura 8.9 temos o formato da palavra de controle que deve ser enviada à PIO, que denominamos de palavra de controle do vetor de interrupção.

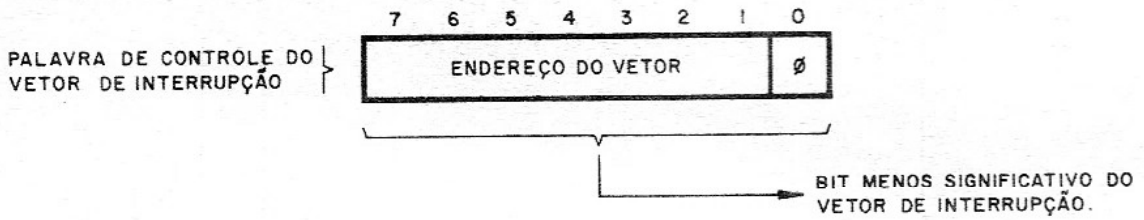


Fig. 8.9 – Palavra de Controle do Vetor de Interrupção

Nas interrupções subseqüentes ao envio da referida palavra de controle, a PIO envia este byte ao Z-80 que, em conjunto com o conteúdo do Registrador I, forma o endereço do *Vetor de Interrupção*, conforme explicado no capítulo 6.

8.2.8.2 – Registrador Controlador de Interrupção

Associado a cada porta da PIO existe um *registrador controlador de interrupção* de dois bits e um *registrador de máscara*, mostrado na figura 8.3.

O registrador controlador de interrupção armazena as informações relativas a forma de operação da interrupção, transferidas pela *palavra de controle de interrupção* mostrada na figura 8.10.

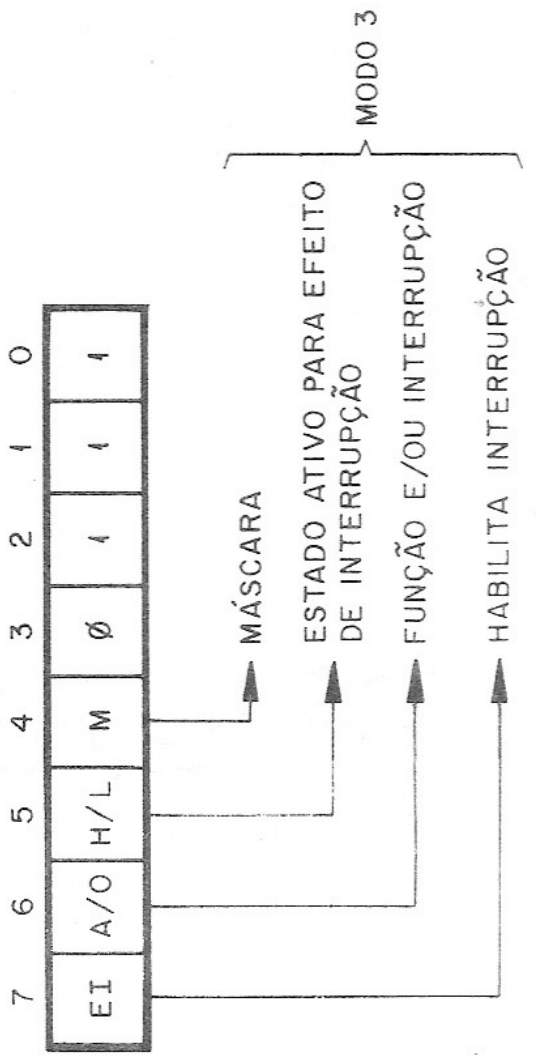


Fig. 8.10 – Palavra de Controle de Interrupção

A palavra de controle de interrupção é transferida para o registrador controlador de interrupção através de uma instrução de OUT (C/$\overline{D}$ SEL = 1, $\overline{CS}$ = 0).

O bit 7 da palavra de controle de interrupção determina se as interrupções estão habilitadas (bit 7 = 1) ou inibidas, para aquela porta. Como pode ser observado nos diagramas das figuras 8.5, 8.6 e 8.7, a interrupção é gerada para os Modos 0, 1 e 2 na transição positiva do sinal $\overline{STB}$.

Os bits 6, 5 e 4 só são válidos para o Modo 3. O bit 5 define o estado ativo das linhas da porta de entrada e saída; se bit 5 = 1 o estado ativo é o nível lógico alto. O bit 6 especifica se a função de interrupção é um "e" ou "ou" lógico. Se bit 6 = 1, todos os bits têm que ir para o estado ativo para que seja gerada uma interrupção. Se bit 6 = 0, basta que um bit vá para o estado ativo para que uma interrupção seja gerada.

As linhas da porta monitoradas pela condição definida pelo bit 6 são determinadas pela *máscara,* que fica armazenada no *registrador de máscara* mostrado na figura 8.3. Se o bit 4 da palavra de controle de interrupção for igual a 1 após o carregamento do registrador controlador da interrupção, o próximo byte enviado à PIO tem que ser a máscara de interrupção, que está mostrada na figura 8.11. Os bits da máscara que estiverem no nível lógico 0, indicam que as linhas correspondentes da porta de entrada e saída serão usadas como uma linha ativa na geração da interrupção.

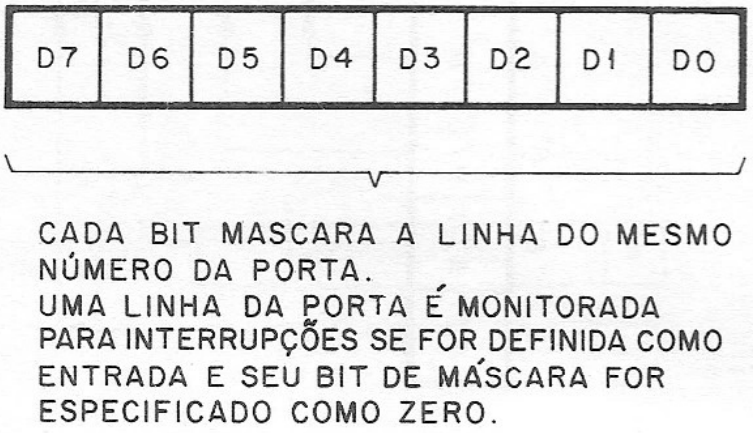


Fig. 8.11 – Máscara de Interrupção

8.3 – MEMÓRIAS

O Micro Z-80 dispõe de três posições onde podem ser equipadas memórias EPROM 2716 ou 2732 ou a memória RAM 6116.

8.3.1 – Memória RAM

O diagrama lógico da memória RAM estática 6116 está mostrada na figura 8.12.

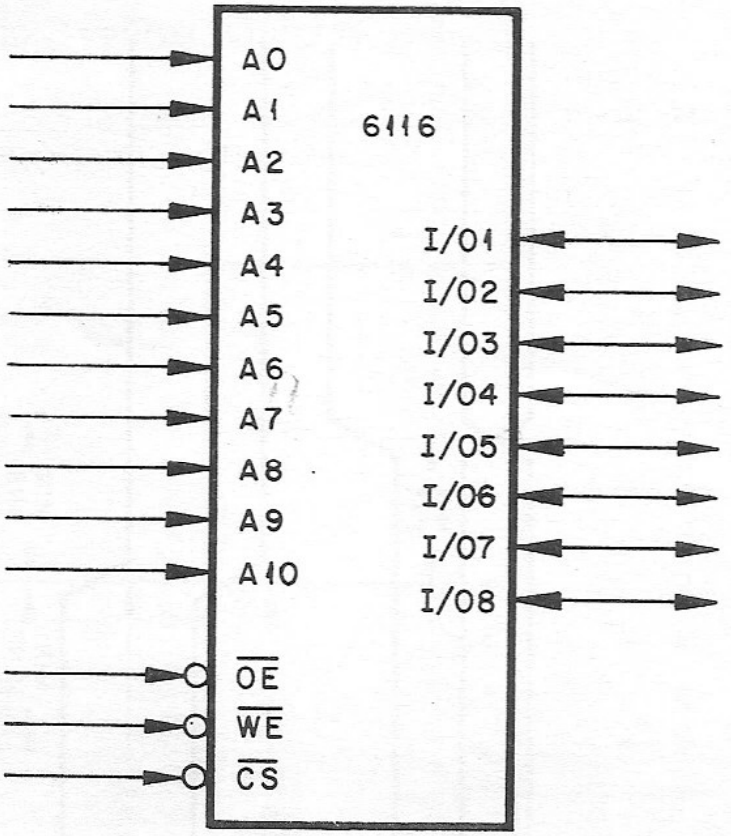


Fig. 8.12 – Diagrama Lógico da RAM 6116

Esta memória é organizada em 2K bytes (2K x 8). Portanto, temos 11 linhas de endereçamento (A0 – A10) e 8 linhas de dados (I/01 – I/08). Os sinais de controle são $\overline{CS}$, $\overline{OE}$ e $\overline{WE}$. O sinal $\overline{CS}$ seleciona o circuito integrado, $\overline{OE}$ habilita os buffers de saída e $\overline{WE}$ é o comando de escrita.

As figuras 8.13 e 8.14 mostram, respectivamente, os diagramas de tempo de leitura e escrita da HM6116P-2, fabricada pela HITACHI.

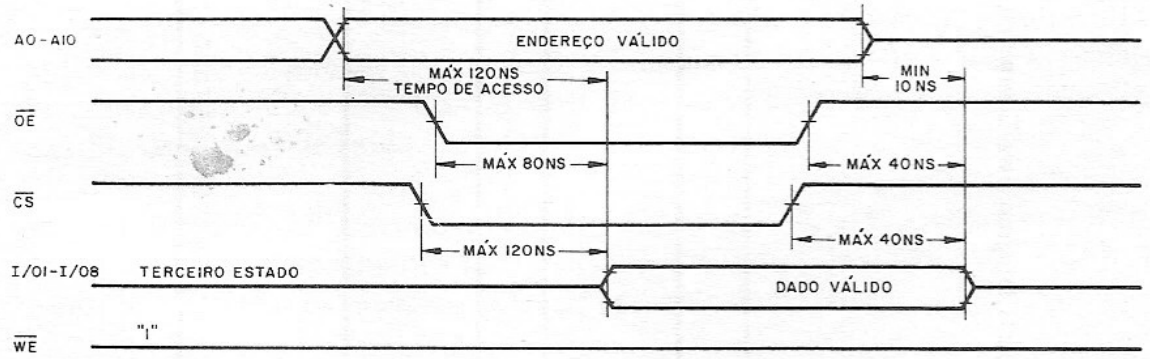
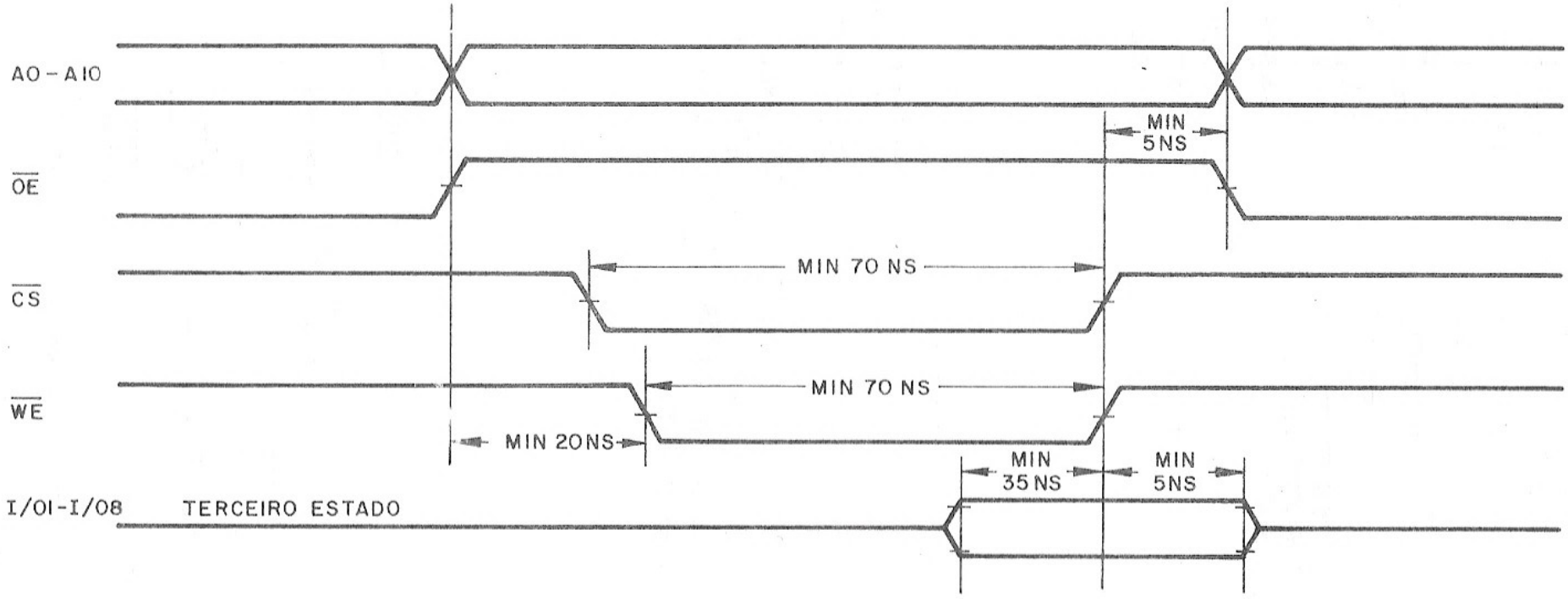


Fig. 8.13 – Diagrama de Tempo de Leitura na RAM HM6116P-2

Fig. 8.14 — Diagrama de Tempo de Escrita na RAM HM6116P-2



8.3.2 – Memórias EPROM

O diagrama lógico da memória EPROM 2716 está mostrado na figura 7.1 e o da EPROM 2732 na figura 8.15.

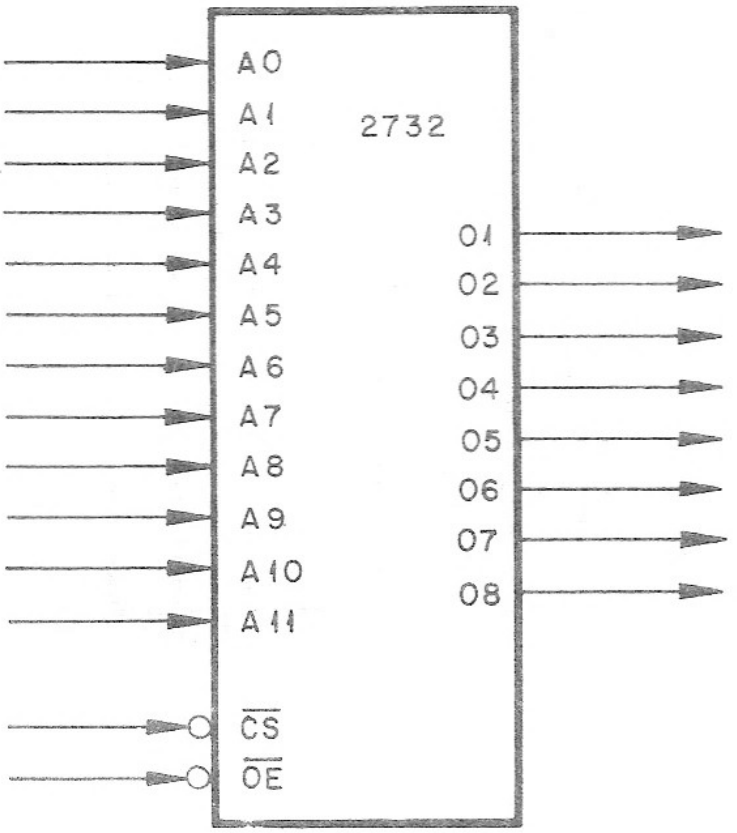


Fig. 8.15 – Diagrama Lógico da EPROM 2732

A EPROM 2716 é organizada em *2k bytes* (2k x 8), tendo, portanto, 11 linhas de endereçamento (A0 – A10) e 8 linhas de dados (01 – 08). Os sinais de controle são $\overline{CS}$ e $\overline{OE}$. $\overline{CS}$ é o sinal de seleção do circuito integrado e $\overline{OE}$ habilita os buffers de saída.

A EPROM 2732 é organizada em *4K bytes* (4K x 8), tendo, portanto, 12 linhas de endereçamento (A0 – A11) e 8 linhas de dados (01 – 08). A EPROM 2732 tem os mesmos sinais de controle da 2716.

A figura 8.16 mostra o diagrama de tempo de leitura das EPROMS 2716 e 2732, fabricadas pela INTEL.

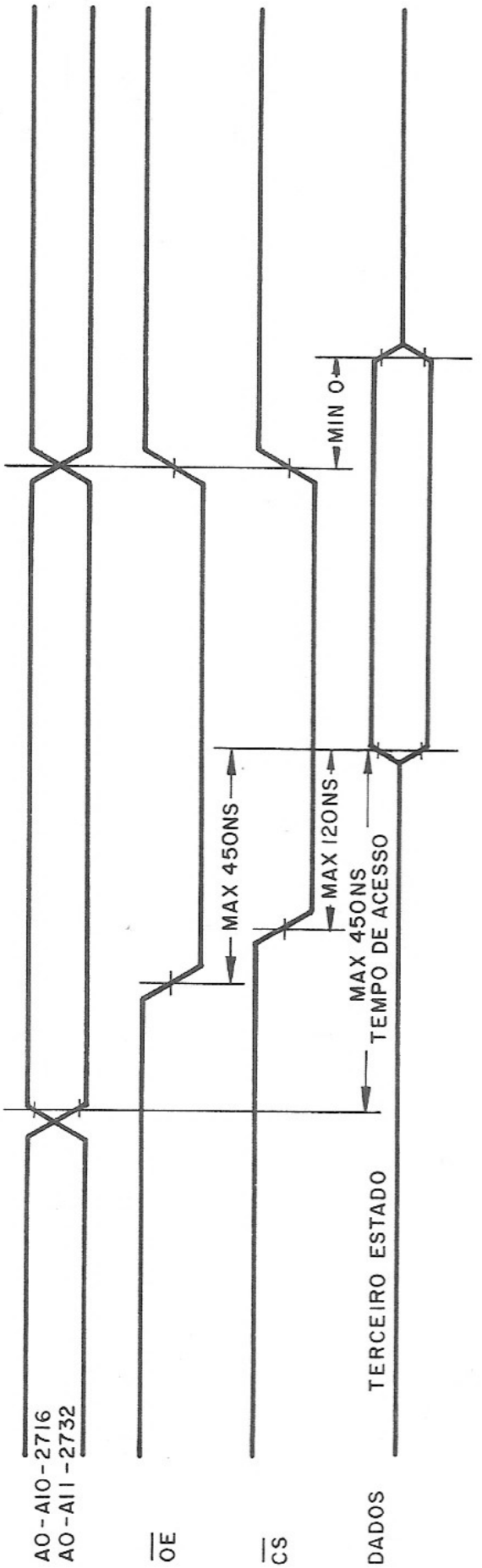


Fig. 8.16 — Diagrama de Tempo de Leitura das EPROMS 2716 e 2732

8.3.3 – Compatibilidade de Memórias

Os sinais de entrada e saída das memórias 6116, 2716 e 2732 são eletricamente compatíveis com a lógica TTL, o que facilita a ligação com o Z-80.

Agora observemos a figura 8.17 que mostra a designação dos pinos destas três memórias.

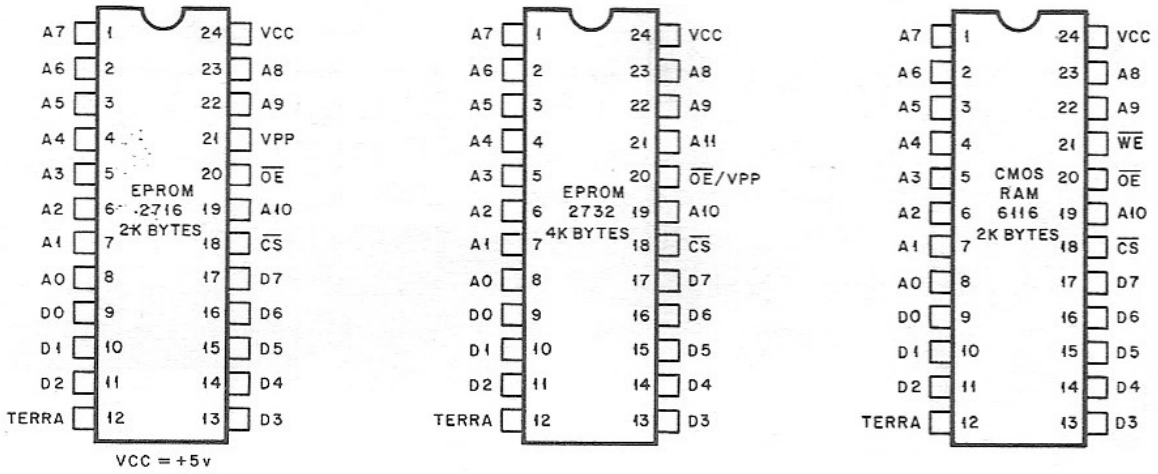


Fig. 8.17 – Pinos das Memórias 2716, 2732 e 6116

A diferença entre as memórias 2716, 2732 e 6116 está nos pinos 20 e 21. Na 2716, o pino 21 serve para reprogramar a EPROM e durante os ciclos de leitura tem que ficar no nível lógico baixo. O pino 21 na memória 2732 é o sinal de endereçamento A11. O pino 20 passa a ter dupla função; durante a leitura é o sinal $\overline{OE}$ e durante a escrita serve para reprogramar a 2732. Na RAM 6116 o pino 21 é o comando de escrita $\overline{WE}$, e o pino 20 atua, unicamente, habilitando os buffers.

No projeto do Micro Z-80 implementamos uma programação adequada para o pino 21 dos soquetes de memória. Assim, podemos utilizar a 2716, a 2732 ou a 6116, o que proporciona maior flexibilidade ao projeto.

8.4 – PROJETO DO MICRO Z-80

A figura 8.18 mostra o diagrama elétrico do Micro Z-80

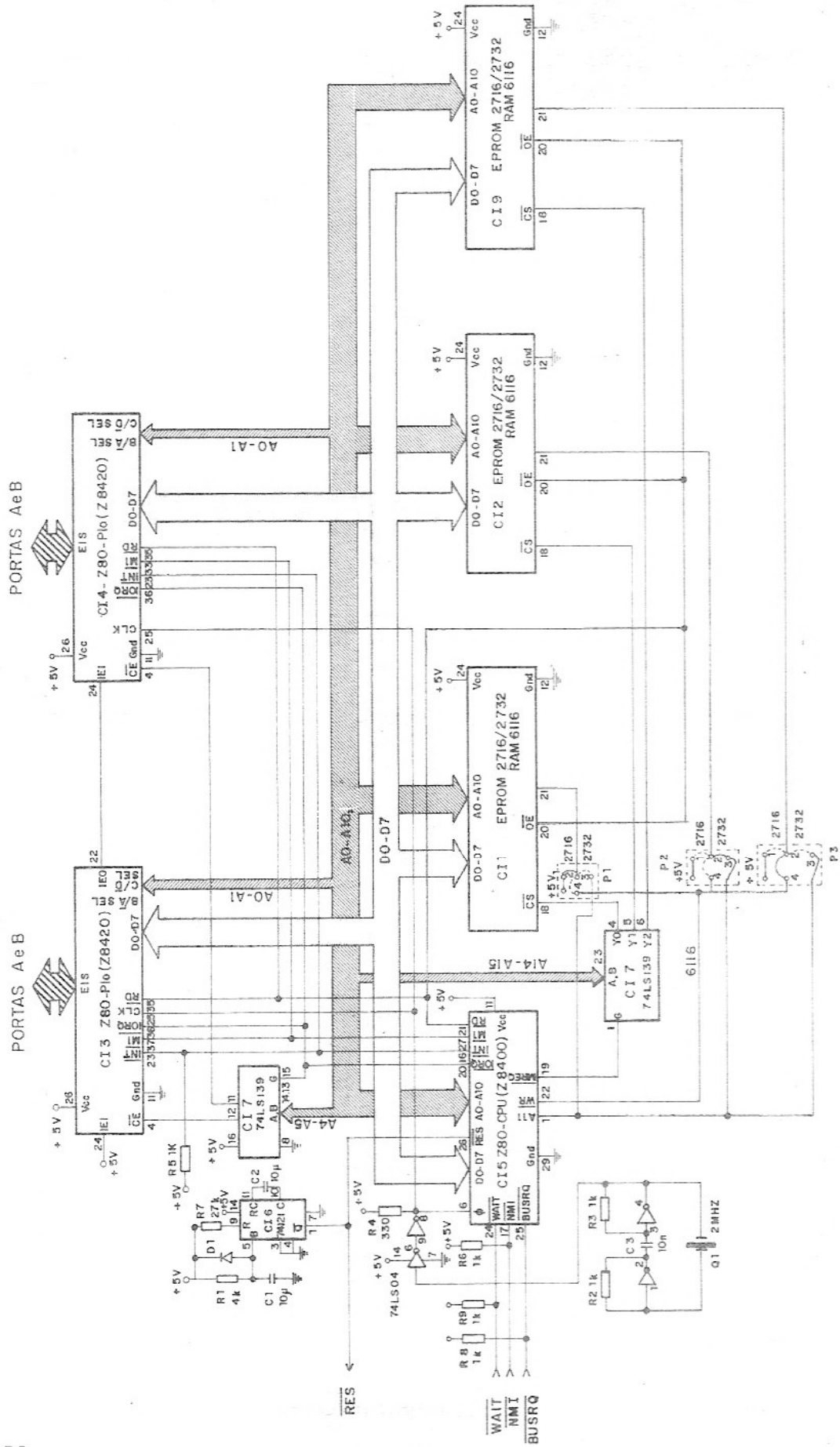


Fig. 8.18 — Diagrama Elétrico do Micro Z-80

O Micro Z-80 é composto dos seguintes elementos lógicos:

1 – Microprocessador Z-80.

2 – Relógio de sincronização.

3 – Lógica de reset.

4 – Três soquetes que podem ser equipados com as memórias 2716, 2732 ou 6116.

5 – Duas PIO's.

6 – Lógica de endereçamento.

O Micro Z-80 da forma apresentada é adequado para aplicações dedicadas. No entanto, se adicionarmos um teclado, um display para os endereços e dados e um programa monitor, o Micro Z-80 torna-se uma valiosa ferramenta para o desenvolvimento de programas.

8.4.1 – O Relógio de Sincronização

O relógio de sincronização, mostrado na figura 8.19, consiste de um oscilador controlado à cristal (2MHz), implementado com o circuito integrado 74LS04, mostrado na figura 8.20, que é composto de seis inversores. Observe que a saída do inversor do último estágio tem um resistor de "pull up" de 330Ω, para compensar a alta capacitância da entrada CLK do Z-80 (35pF) e das PIO's (10pF).

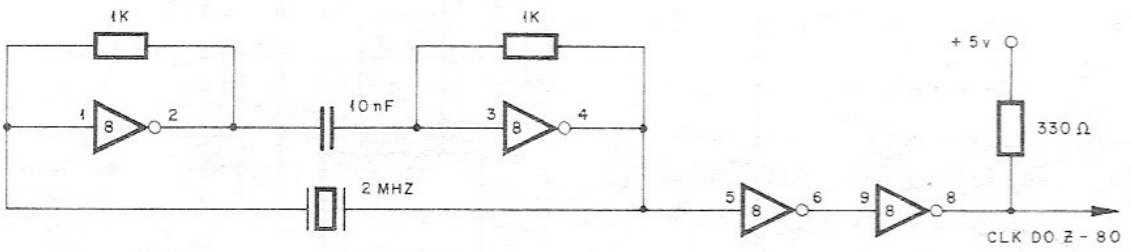


Fig. 8.19 – O Relógio de Sincronização

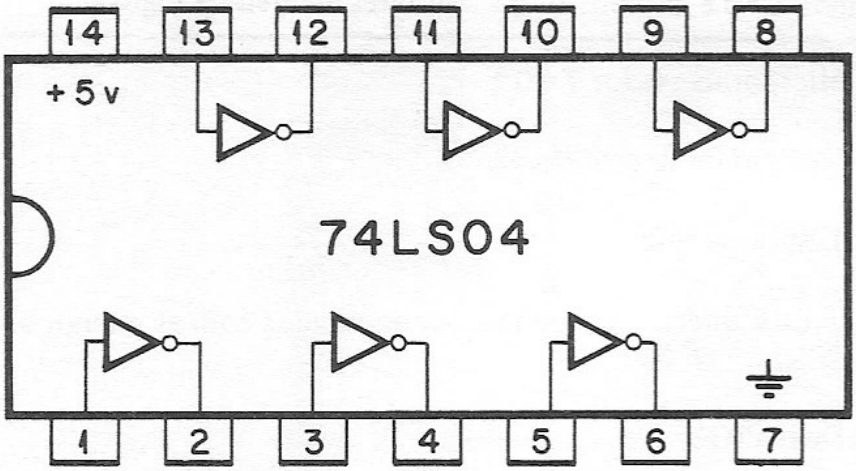


Fig. 8.20 – O Circuito Integrado 74LS04

8.4.2 – A Lógica do Reset

A lógica do reset consiste do monoestável 74121, que tem o objetivo de garantir um nível lógico baixo na entrada $\overline{RES}$ (pino 26) do Z-80, durante um tempo suficiente para resetar o microprocessador.

A figura 8.21 mostra as ligações realizadas no 74121, para a utilização com o Z-80. A entrada B é *schmitt-trigger*, o que retarda o disparo do pulso de Reset até o carregamento do capacitor C1. Note que o pulso de reset só é disparado após o acionamento da alimentação. A largura do pulso (T) gerado pelo 74121, na configuração da figura 8.21, é dado pela fórmula.

$$T = 0{,}7\ (R_{INT} + R7)\ C2$$

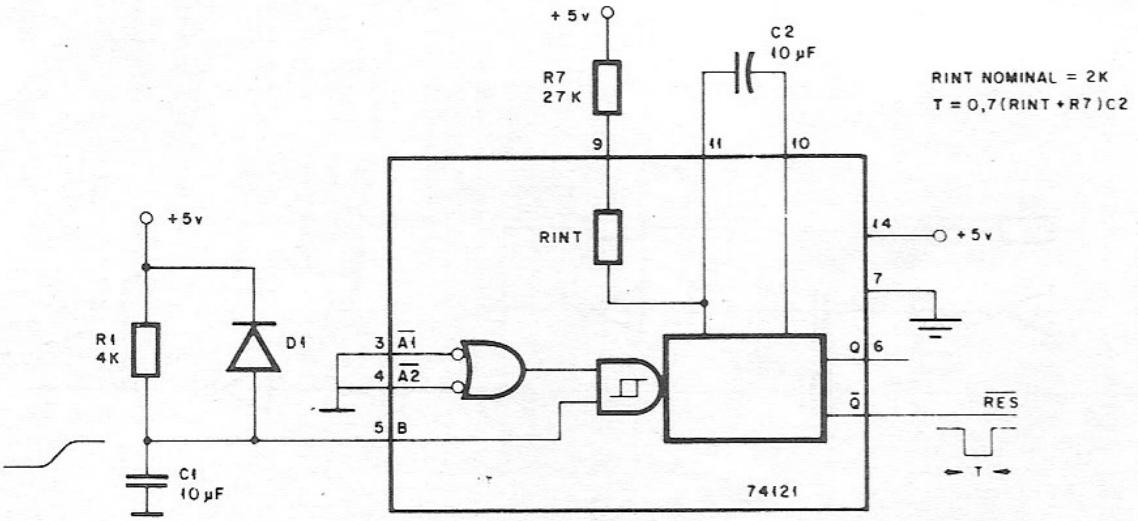


Fig. 8.21 – O Circuito de Reset

A figura 8.22 mostra o diagrama de tempo do circuito de reset.

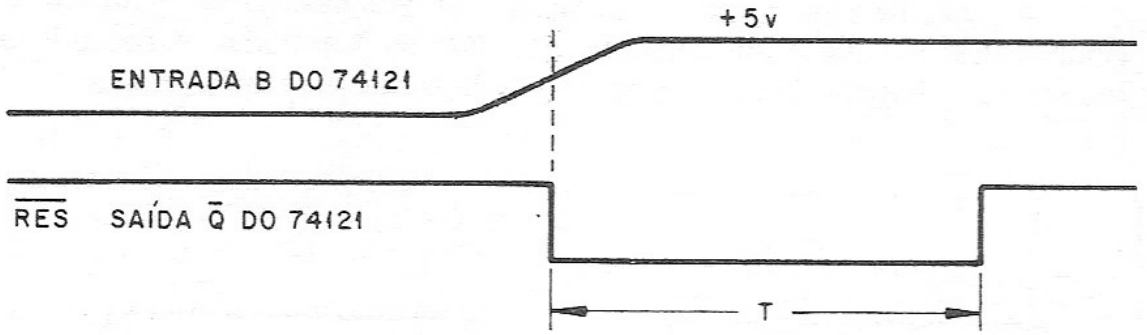


Fig. 8.22 – Diagrama de Tempo do Reset

8.4.3 – Memórias

O Micro Z-80 dispõe de 3 soquetes nos quais podem ser equipadas as memórias 2716, 2732 ou 6116.

As programações P1, P2 e P3 determinam o tipo de memória usada; a ligação 1-2 designa a EPROM 2716, 3-2 designa a EPROM 2732 e 4-2 a RAM 6116.

Os sinais $\overline{OE}$ (pino 20) das memórias são comandados pela sinal $\overline{RD}$ (pino 21) do Z-80. Os sinais $\overline{CS}$ (pino 18) das memórias são comandados pela lógica de endereçamento.

8.4.4 – PIO's

O Micro Z-80 dispõe de duas PIO's que oferecem 40 linhas para operações de entrada e saída, com controle de interrupções.

As linhas $\overline{INT}$ (pino 23) das PIO's estão conectadas em *wired-OR* com um resistor de pull-up de 1K, que por sua vez, estão ligadas a entrada $\overline{INT}$ do Z-80 (pino 16). A resolução de prioridades se faz pela técnica *daisy chain* estudada no capítulo 6.

As entradas $\overline{M1}$, $\overline{IORQ}$ e $\overline{RD}$ estão conectadas diretamente nas saídas correspondentes do Z-80. E, as entradas CLK (pino 25) estão ligadas a saída do relógio de sincronização.

Os sinais B/$\overline{A}$ SEL (pino 6) e C/$\overline{D}$ SEL (pino 5) são comandados pela lógica de endereçamento.

8.4.5. – Lógica de Endereçamento

A lógica de endereçamento do Micro Z-80 está baseada no circuito integrado 74LS139, que é um decodificador dual de 4 entradas. A figura 8.23 mostra o seu diagrama lógico e a figura 8.24 mostra a sua tabela verdade.

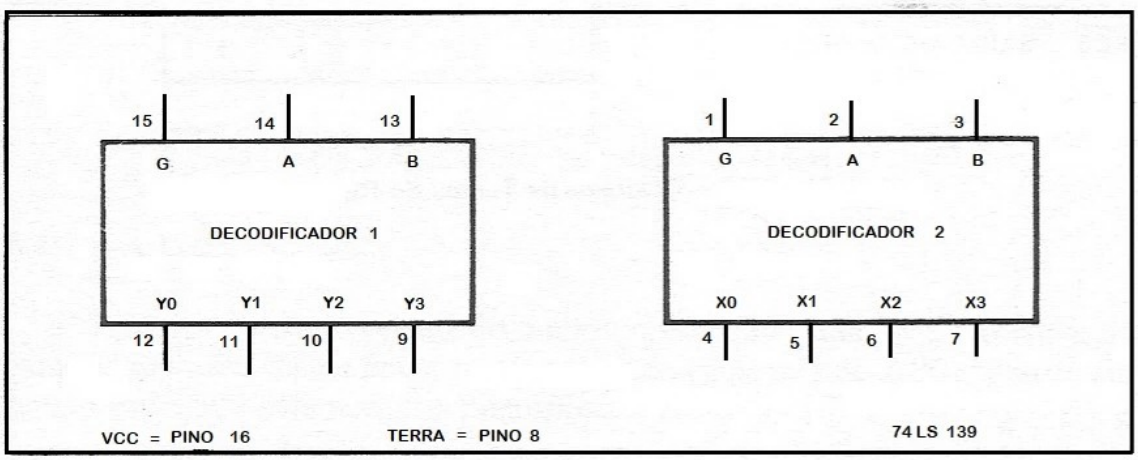


Fig. 8.23 – Diagrama Lógico do Circuito Integrado 74LS139

ENTRADAS			SAÍDAS			
G	A	B	Y0/X0	Y1/X1	Y2/ X2	Y3/X3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	H	L	H	L	H	H
L	L	H	H	H	L	H
L	H	H	H	H	H	L

H = NÍVEL LÓGICO ALTO
L = NÍVEL LÓGICO BAIXO

Fig. 8.24 – Tabela Verdade do Circuito Integrado 74LS139

O decodificador 1 do 74LS139 é usado na seleção das PIO's. A habilitação deste decodificador se faz pelo comando $\overline{IORQ}$ (pino 20) do Z-80 e o endereço é selecionado pelas linhas de endereço A4 e A5. Admitindo que as demais linhas de endereço fiquem no nível lógico baixo durante uma seleção, temos então os seguintes endereços (em hexadecimal):

PIO – CI3	PORTA A	DADOS	00H
		CONTROLE	02H
	PORTA B	DADOS	01H
		CONTROLE	03H
PIO – CI4	PORTA A	DADOS	10H
		CONTROLE	12H
	PORTA B	DADOS	11H
		CONTROLE	13H

O decodificador 2 do 74LS139 é usado na seleção das memórias. A habilitação deste decodificador se faz pelo comando $\overline{MREQ}$ (pino 19) do Z-80 e o endereço é selecionado pelas linhas de endereço A14 e A15. Admitindo que as linhas de endereço não usadas fiquem no nível lógico baixo durante uma seleção, temos então os seguintes endereços (em hexadecimal):

CI1	2716 ou 6116	0000H A 07FFH
	2732	0000H A 0FFFH
CI2	2716 ou 6116	4000H A 47FFH
	2732	4000H A 4FFFH
CI9	2716 ou 6116	C000H A C7FFH
	2732	C000H A CFFFH

8.5 – OPERAÇÕES COM A PIO

Faremos agora alguns exemplos de utilização da PIO no Micro Z-80. Primeiramente, estudaremos o caso de operações sem interrupção e, em seguida, com interrupções.

8.5.1 – Operação sem Interrupções

Para operarmos uma porta da PIO sem interrupções temos que usar o *método de polling* estudado no capítulo 6.

Os seguintes passos devem ser seguidos para operarmos a PIO-CI3 do Micro Z-80 como entrada (Porta B) e saída (Porta A), sem utilizarmos interrupções:

1 – O reset da PIO, quando é acionado a alimentação, bloqueia as interrupções.

2 – Através de duas instruções de OUT, transferir a palavra de controle de interrupção 07H (figura 8.10) aos endereços 02H e 03H. Isto bloqueia interrupções para as portas A e B.

3 – Através de uma instrução OUT, transferir a palavra de modo 0FH (Fig. 8.4) para a porta A no endereço 02H. Isto determina que a porta A vai operar como saída.

4 – Através de uma instrução OUT, transferir a palavra de modo 4FH para a porta B no endereço 03H. Isto determina que a porta B vai operar como entrada.

5 – Ler o conteúdo do registrador de entrada de dados da porta B, através de uma instrução IN no endereço 01H. Este dado lido deve ser ignorado. No entanto, esta leitura ativa a linha BRDY, informando ao dispositivo periférico que o Z-80 está pronto para operação.

6 – Agora a porta A está pronta para efetuar saídas de dados e a porta B para efetuar entradas de dados.

8.5.2 – Operações com Interrupções

Para efetuarmos operações de entrada e saída com a PIO, controladas por interrupção, temos primeiramente que transmitir à PIO a palavra de controle do vetor de interrupção (figura 8.9) e, em seguida, a palavra de controle de interrupção (figura 8.10) apropriada.

Se o endereço da rotina de tratamento de interrupção da porta de saída (A) for D500H e da porta de entrada (B) for D502H, então o registrador I do Z-80 tem que se carregado com o valor D5H, antes da ocorrência de qualquer interrupção da PIO.

O carregamento da palavra de controle do vetor de interrupção 00H deve ser feito através de uma instrução OUT para o endereço 02H. E, o da palavra de controle do vetor de interrupção 02H, também, através de uma instrução de OUT para o endereço 03H.

O carregamento da palavra de controle de interrupção 87H tem que ser enviada aos endereços 02H e 03H, também, através de instruções de OUT. Isto habilita as portas A e B. As interrupções ocorrerão na porta A sempre que o dispositivo periférico ler um dado no registrador de saída de dados e, na porta B sempre que o dispositivo periférico carregar um dado no registrador de entrada de dados.

8.6 – EXERCÍCIOS DE FIXAÇÃO

8.6.1 – Quais as linhas de comunicação da PIO com o Z-80?

8.6.2 – Quais os tipos de informações transacionadas entre a PIO e o Z-80?

8.6.3 – Qual é a condição das entradas da PIO para selecionar o registrador de dados da porta B? E para a transferência de palavras de controle?

8.6.4 – Porque cada porta da PIO tem os seus registradores de dados e controles próprios?

8.6.5 – Para que serve o registrador controlador de modo? Porque não existe um único registrador controlador de modo para as portas A e B?

8.6.6 – Seria possível as portas A e B operarem, simultaneamente, no modo 2?

8.6.7 – O que diferencia a palavra de modo das demais palavras de controle?

8.6.8 – Explique o sinal WR* da figura 8.5.

8.6.9 – Qual a função do sinal READY no modo 0 da PIO? E do sinal $\overline{\text{STROBE}}$?

8.6.10 – Para que serve a interrupção no modo 0?

8.6.11 – Como o Z-80 pode testar uma porta no modo 0 para identificar se o dado foi lido pelo dispositivo periférico, operando sem interrupções?

8.6.12 – Explique o sinal RD* na figura 8.6.

8.6.13 – Qual a função do sinal READY no modo 1 da PIO? E do sinal $\overline{\text{STROBE}}$?

8.6.14 – O que indica uma interrupção no modo 1?

8.6.15 – Cite uma aplicação para a operação no modo 2 da PIO.

8.6.16 – Explique o funcionamento da PIO no modo 3.

8.6.17 – Porque é necessário um segundo byte de controle para programar a PIO no modo 3?

8.6.18 – Qual é o modo de atendimento de interrupção do Z-80 compatível com a PIO?

8.6.19 – O que diferencia a palavra de controle vetor de interrupção das demais palavras de controle?

8.6.20 – Explique a função dos bits 4, 5, 6 e 7 da palavra de controle de interrupção. E, porque os bits 4, 5 e 6 só podem ser associados ao modo 3?

8.6.21 – O que significa o bit 3 da máscara igual a zero?

8.6.22 – Realize a interface PIO/leitora de fita de papel da figura 8.25 e especifique o modo de operação da PIO. Faça apenas o projeto lógico, sem levar em conta as características elétricas.

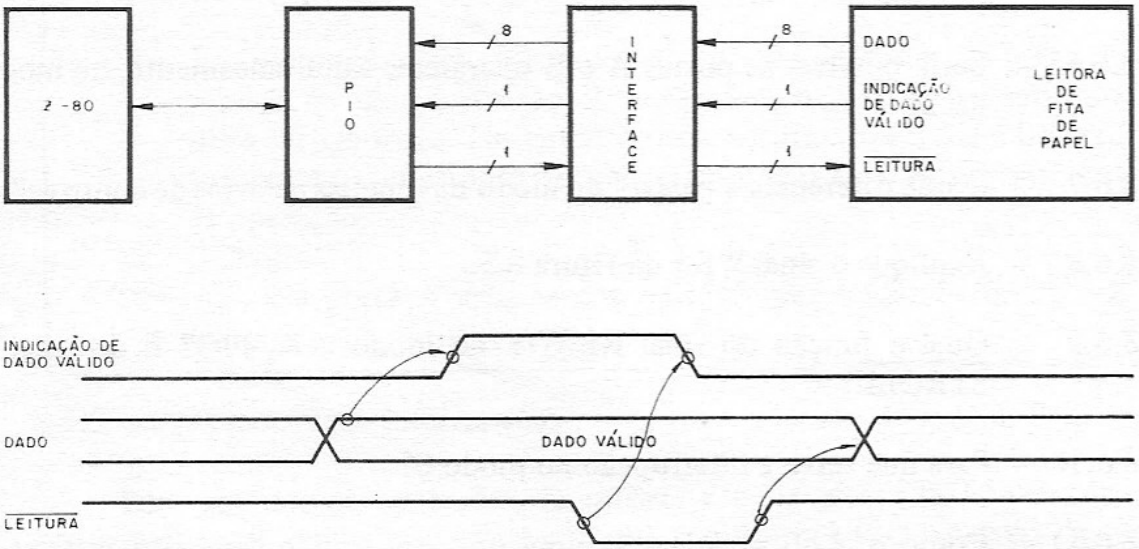


Fig. 8.25 – Interface PIO/Leitora de Fita de Papel

8.6.23 – Com relação ao exercício anterior, explique a seqüência de inicialização da PIO, operando com e sem interrupção. Admita que após ligarmos a alimentação são garantidas as seguintes condições:

1 – Interrupções bloqueadas.
2 – Registradores de saída e de máscara resetados.
3 – Modo 1 selecionado ("default").
4 – Linhas de entrada e saída no terceiro estado lógico.
5 – Sinais de protocolo inativos (nível lógico baixo).

8.6.24 – Faça um fluxograma do programa que deve controlar a leitora de fita de papel, operando com e sem interrupções.

8.6.25 – Verifique se é necessário o uso do sinal de $\overline{WAIT}$ quando o Z-80, Z-80A e Z-80B acessam a memória 6116. Usar os diagramas das figuras 8.13 e 8.14 e os dados do apêndice A.

8.6.26 – Determine quantos circuitos integrados da memória 6116 podem ser ligados ao Z-80 sem o uso de buffers. Os dados do Z-80 estão no apêndice A e os parâmetros elétricos máximos da 6116 são os seguintes:

IOH = 1mA, IOL = 2,1mA, IIH = IIL = 10μA
Corrente de fuga no 3º estado = 10μA
Capacitância de entrada = 5pF
Capacitância de saída = 10pF

8.6.27 – Determine quantos circuitos integrados da memória 2716 podem ser ligados ao Z-80 sem o uso de buffers. Os dados do Z-80 estão no apêndice A e os parâmetros elétricos máximos da 2716 são os seguintes:

IOH = 400μA, IOL = 2,1mA, IIH = IIL = 10μA
Corrente de fuga no 3º estado = 10μA
Capacitância de entrada = 6pF
Capacitância de saída = 12pF

8.6.28 – Verifique se é necessário o uso do sinal de $\overline{WAIT}$ quando o Z-80, Z-80A e Z-80B acessam a memória 2716. Usar o diagrama da figura 8.16 e os dados do apêndice A.

8.6.29 – Cite uma vantagem para a compatibilidade dos pinos dos circuitos integrados 2716, 2732 e 6116.

8.6.30 – Quais são os blocos lógicos que integram o Micro Z-80?

8.6.31 – Qual é a largura T do pulso gerado pelo monoestável 74121 da lógica de reset do Micro Z-80?

8.6.32 – Porque a entrada B do 74121 da lógica de reset do Micro Z-80 foi ligada a alimentação através do resistor R1 e do capacitor C1? A entrada B poderia ter sido ligada diretamente na saída da fonte de + 5V?

8.6.33 – Porque é necessário o reset do Micro Z-80 logo após o acionamento da alimentação?

8.6.34 – Porque foi usado o resistor de "pull up" R4 na saída do relógio de sincronização do Z-80?

8.6.35 – Verifique se os sinais que comandam as entradas $\overline{OE}$, $\overline{CS}$ e $\overline{WE}$ das memórias do Micro Z-80 estão de acordo com os diagramas de tempo das figuras 8.13, 8.14 e 8.16.

8.6.36 – Altere a lógica de endereçamento das PIO's no Micro Z-80 para que os novos endereços da PIO-CI3 sejam 20H, 21H, 22H, 23H e, da PIO-CI4 sejam 30H, 31H, 32H e 33H.

8.6.37 – Diga como programar a PIO–CI4 do Micro Z-80 para que a porta A opere como saída e a porta B como entrada, sem interrupções.

8.6.38 – Porque o registrador I do Z-80 tem que ser preparado antes de qualquer interrupção da PIO?

8.6.39 – Diga como programar a PIO-CI4 do Micro Z-80 para que a porta A opere como entrada e a porta B como saída, com interrupções. O endereço da rotina de interrupção da porta de saída é EE00H e da porta de entrada EE10H.

PARTE IV
APÊNDICES

A CARACTERÍSTICAS ELÉTRICAS DO Z-80

O conteúdo deste apêndice é um resumo das principais informações existentes no manual da ZILOG.

Na implementação de projetos com o Z-80, recomendamos a consulta adicional ao manual do fabricante.

As designações dos parâmetros e dos tempos existentes neste apêndice serão mantidas na versão original em inglês.

A.1 – CARACTERÍSTICAS DC

Símbolo	Parâmetro	Min	Max	Unidade	Condições de Teste
V_{ILC}	Clock Input Low Voltage	- 0.3	0.45	V	
V_{IHC}	Clock Input High Voltage	V_{cc} - .6	V_{cc} + .3	V	
V_{IL}	Input Low Voltage	- 0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{cc}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 1.8 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 250 μA
I_{cc}	Power Supply Current				
	Z80		150[1]	mA	
	Z80A		200[2]	mA	
	Z80B		200	mA	
I_{LI}	Input Leakage Current		10	μA	V_{IN} = 0 to V_{cc}
I_{LEAK}	3-State Output Leakage Currente in Float	- 10	10[3]	μ A	V_{OUT} = 0.4 to V_{cc}

OBS.:

1 – Na versão militar, Icc = 200 mA

2 – Média típica para o Z-80A é de 90 mA

3 – AI5–AO, D7–DO, $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ e $\overline{WR}$

A.2 – CAPACITÂNCIAS

Símbolo	Parâmetro	Min	Max	Unidade
C_{CLOCK}	Clock Capacitance		35	pF
C_{IN}	Input Capacitance		5	pF
C_{OUT}	Output Capacitance		10	pF

OBS.:

1 – TA = 25°C, f = 1MHz
2 – Pinos não utilizados, colocados na terra

A. 3 – CARACTERÍSTICAS AC

Temos a sequir os diagramas de tempo dos diversos ciclos de máquina do Z-80, nos quais encontram-se assinalados os tempos envolvidos.

A. 3.1 – Ciclo de Busca de Instrução

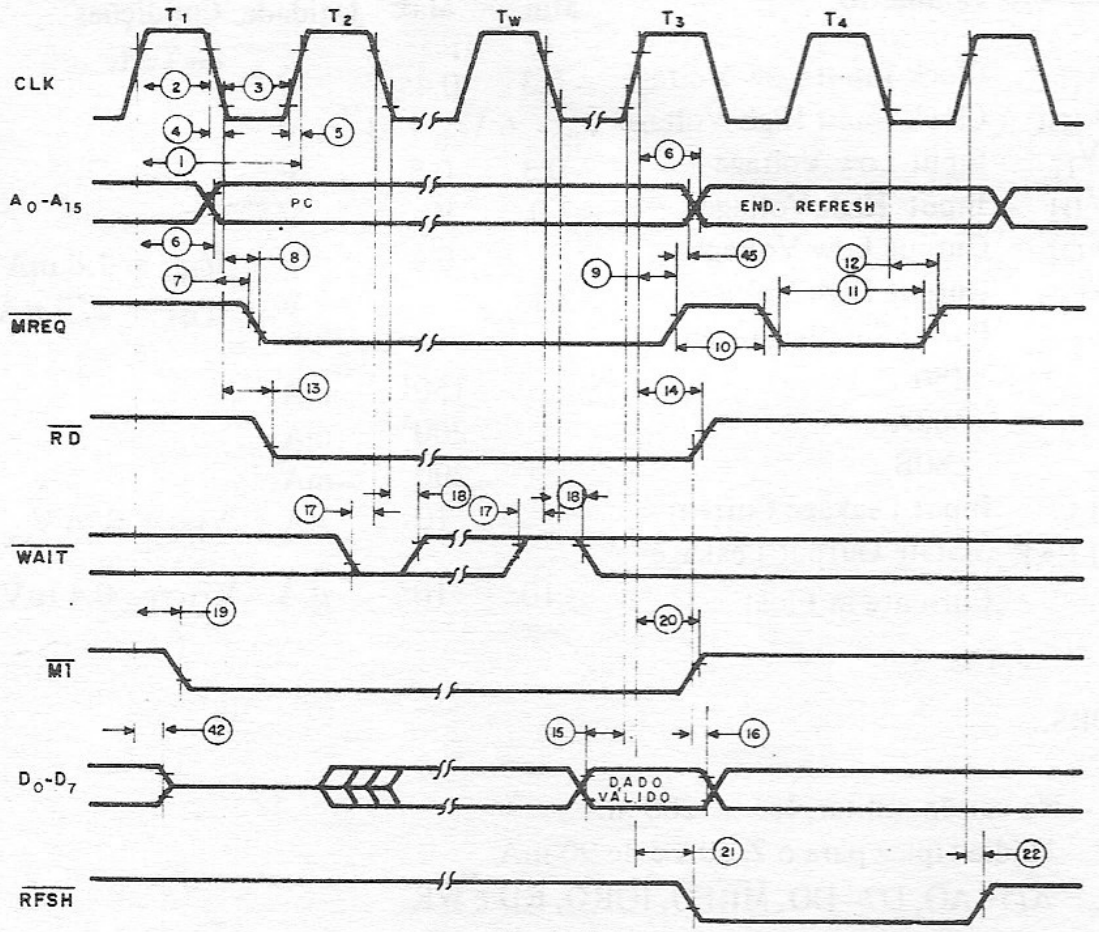


Fig. A. 3.1

A. 3.2 – Ciclos de Leitura e Escrita na Memória

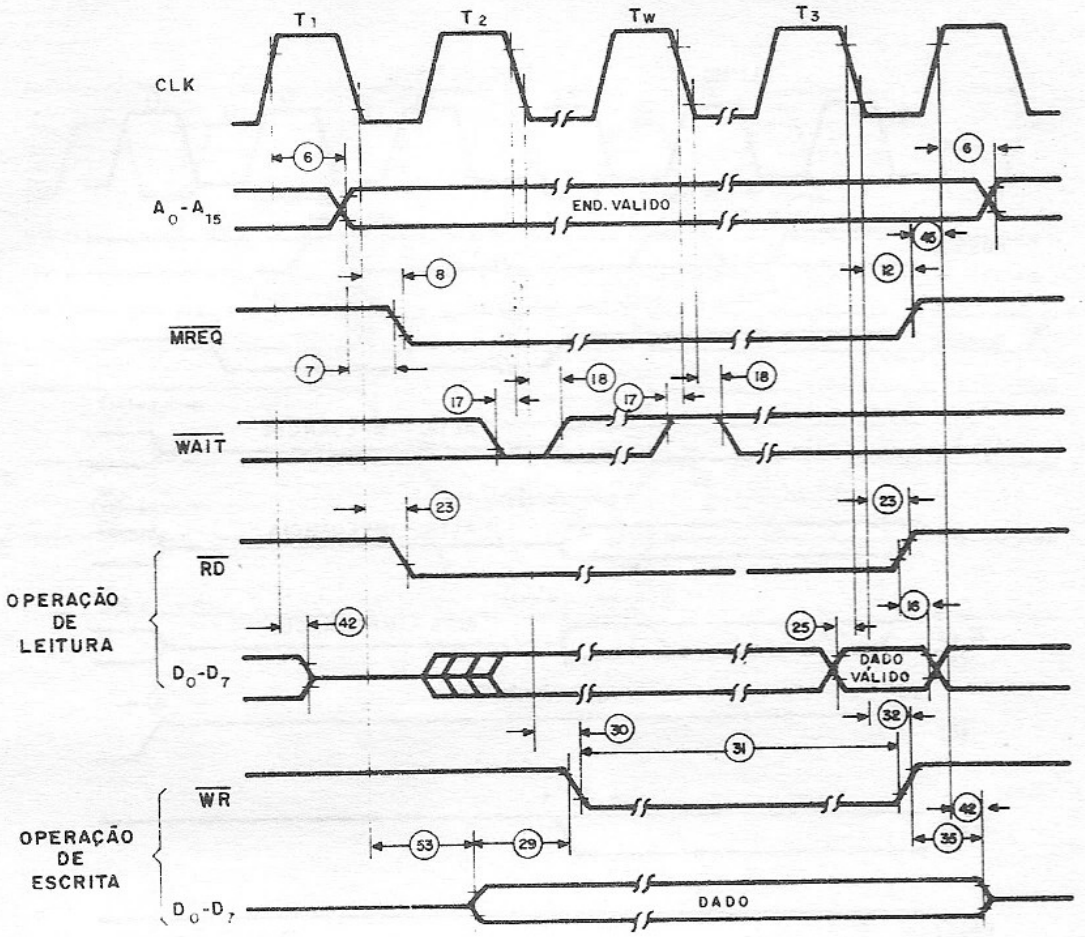


Fig. A. 3.2

A. 3.3 – Ciclos de Leitura e Escrita em Dispositivos de E/S

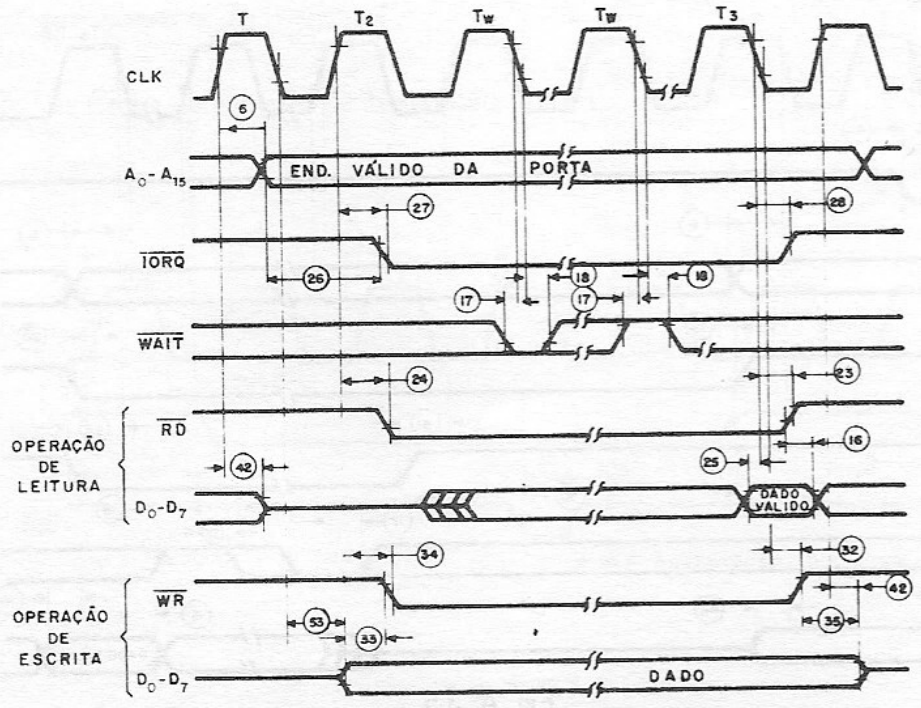


Fig. A. 3.3

A. 3.4 – Ciclos de Pedido de Controle das Barras

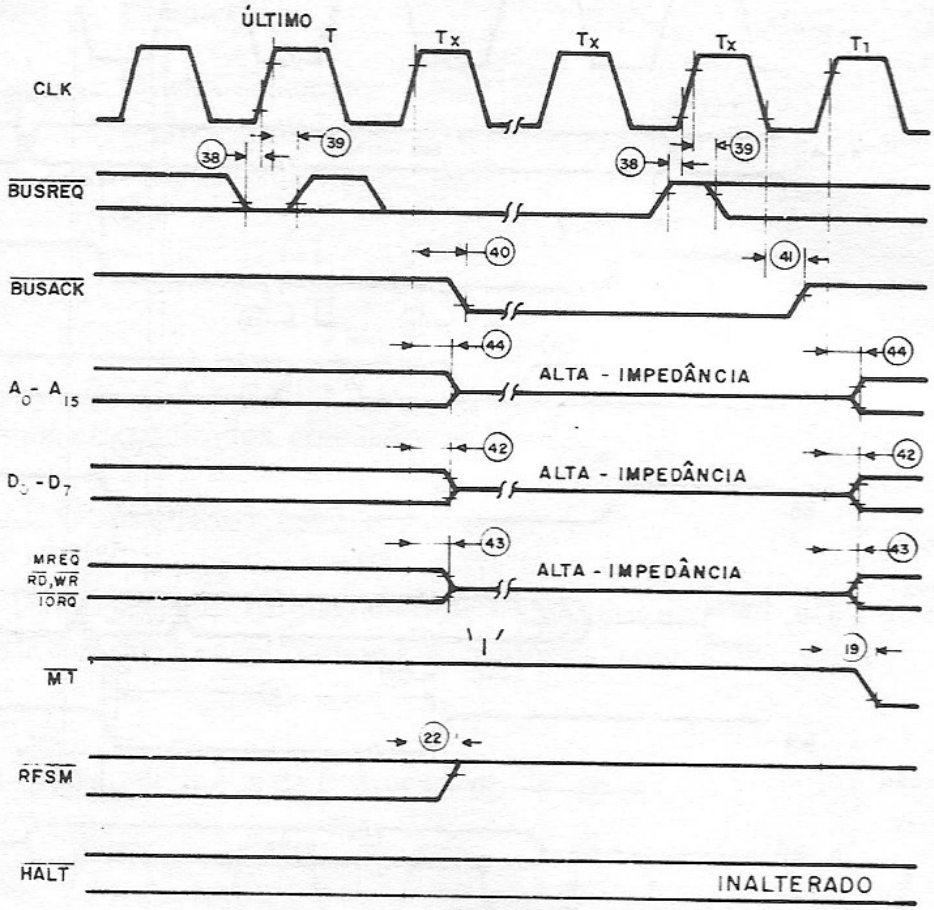


Fig. A. 3.4

A. 3.5 – Ciclo de Interrupção INT

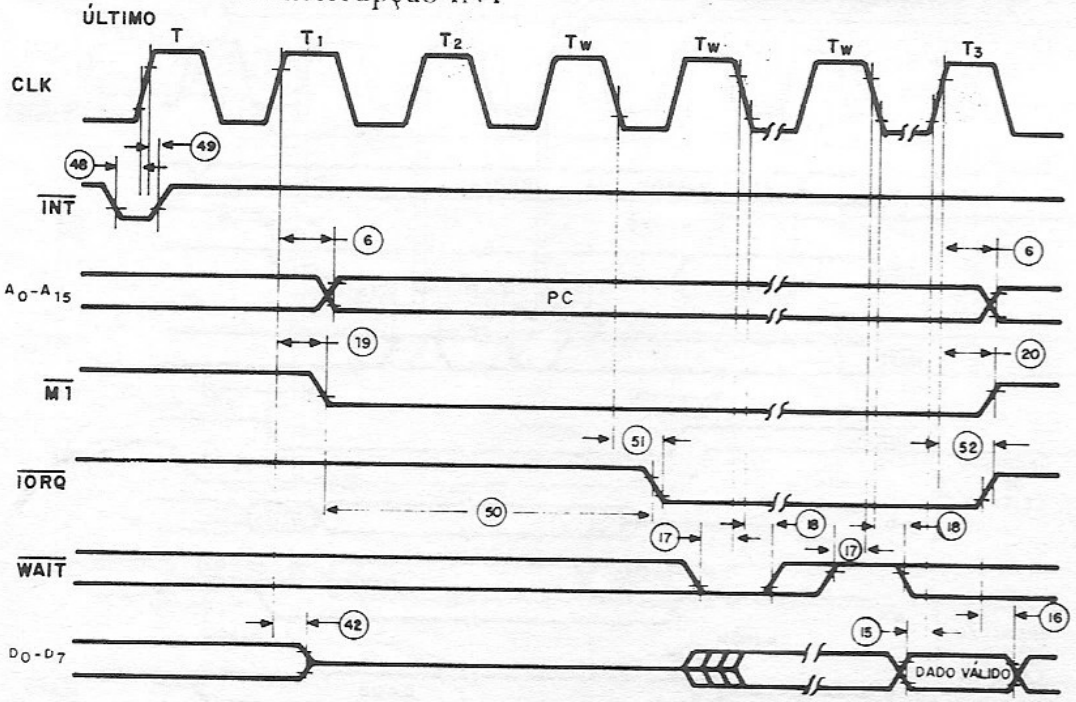


Fig. A. 3.5

A. 3.6 – Ciclo de Interrupção NMI

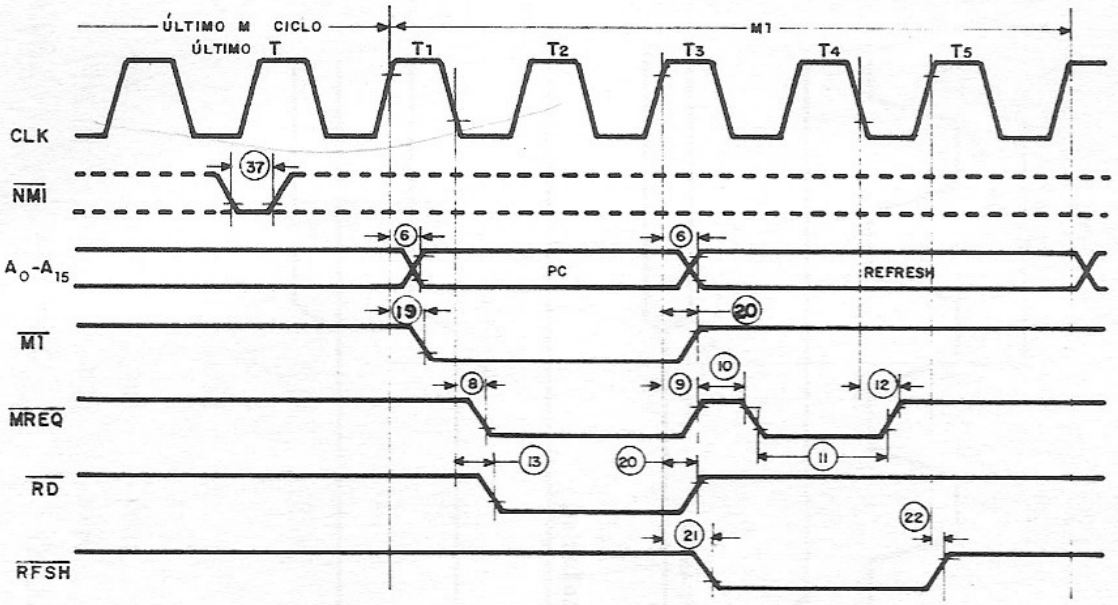


Fig. A. 3.6

OBS.:

1 – Para garantir o atendimento da interrupção NMI no próximo ciclo de máquina, a transição negativa de $\overline{NMI}$ tem que ocorrer antes da transição positiva do último T.

A. 3.7 – Ciclo de Escape da Instrução HALT.

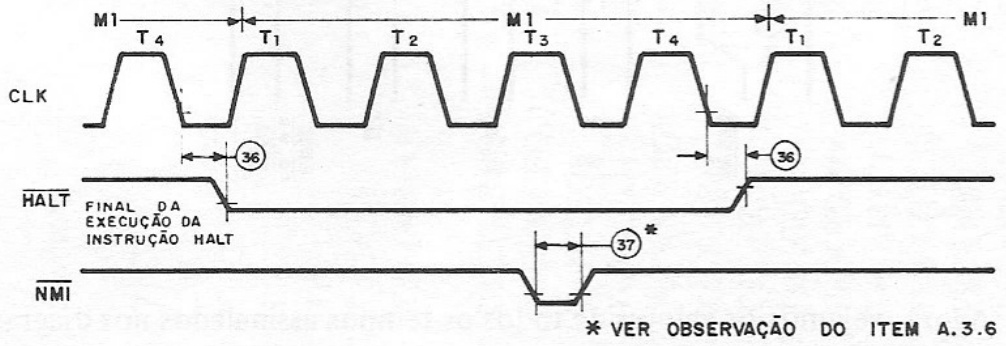


Fig. A. 3.7.

A. 3.8 – Ciclo de Reset

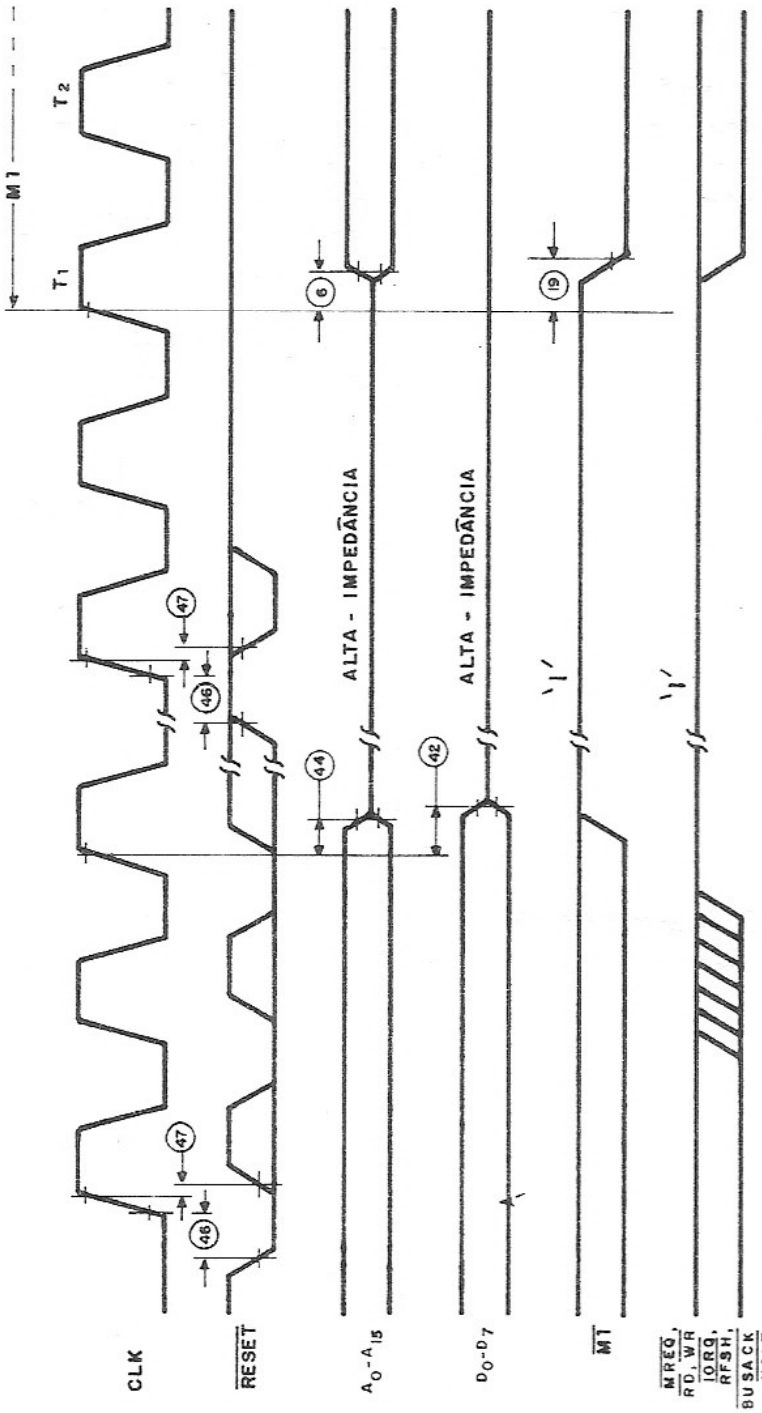


Fig. A. 3.8.

Agora, vejamos os valores de todos os tempos assinalados nos diagramas anteriores.

Número	Símbolo	Parâmetro	Z80 CPU		Z80A CPU		Z80B CPU	
			Min (ns)	Max (ns)	Min (ns)	Max (ns)	Min (ns)	Max (ns)
1	TcC	Clock Cycle Time	400*		250*		165*	
2	TwCh	Clock Pulse Width (High)	180*		110*		65*	
3	TwCl	Clock Pulse Width (Low)	180	2000	110	2000	65	2000
4	TfC	Clock Fall Time	–	30	–	30	–	20
5	TrC	Clock Rise Time		30		30		20
6	TdCr(A)	Clock ↑ to Address Valid Delay	–	145	–	110	–	90
7	TdA(MREQf)	Address Valid to $\overline{\text{MREQ}}$ ↓ Delay	125*	–	65*	–	35*	–
8	TdCf(MREQf)	Clock ↓ to $\overline{\text{MREQ}}$ ↓ Delay	–	100	–	85	–	70
9	TdCr(MREQr)	Clock ↑ to $\overline{\text{MREQ}}$ ↑ Delay	–	100	–	85	–	70
10	TwMREQh	$\overline{\text{MREQ}}$ Pulse Width (High)	170*		110*		65*	
11	TwMREQl	$\overline{\text{MREQ}}$ Pulse Width (Low)	360*	–	220*	–	135*	–
12	TdCf(MREQr)	Clock ↓ to $\overline{\text{MREQ}}$ ↑ Delay	–	100	–	85	–	70
13	TdCf(RDf)	Clock ↓ to $\overline{\text{RD}}$ ↓ Delay	–	130	–	95	–	80
14	TdCr(RDr)	Clock ↑ to $\overline{\text{RD}}$ ↑ Delay	–	100	–	85	–	70
15	TsD(Cr)	Data Setup Time to Clock ↑	50		35		30	
16	ThD(RDr)	Data Hold Time to $\overline{\text{RD}}$ ↑	–	0	–	0	–	0
17	TsWAIT(Cf)	$\overline{\text{WAIT}}$ Setup Time to Clock ↓	70	–	70	–	60	
18	ThWAIT(Cf)	$\overline{\text{WAIT}}$ Hold Time after Clock ↓	–	0	–	0	–	0
19	TdCr(Mlf)	Clock ↑ to $\overline{\text{M1}}$ ↓ Delay	–	130	–	100	–	80
20	TdCr(Mlr)	Clock ↑ to $\overline{\text{M1}}$ ↑ Delay		130		100		80
21	TdCr(RFSHf)	Clock ↑ to $\overline{\text{RFSH}}$ ↓ Delay	–	180	–	130	–	110

22	TdCr(RFSHr)	Clock ↑ to $\overline{RFSH}$ ↑ Delay	–	150	–	120	–	100
23	TdCf($\overline{R}$Dr)	Clock ↓ to $\overline{RD}$ ↑ Delay	–	110	–	85	–	70
24	TdCr(RDf)	Clock ↑ to $\overline{RD}$ ↓ Delay	–	100	–	85	–	70
25	TsD(Cf)	Data Setup to Clock ↓ during M_2, M_3, M_4 or M_5 Cycles	60		50		40	
26	TdA(IORQf)	Address Stabel prior to $\overline{IORQ}$ ↓	320*	–	180*	–	110*	–
27	TdCr(IORQf)	Clock ↑ to $\overline{IORQ}$ ↓ Delay	–	90	–	75	–	65
28	TdCf(IROQr)	Clock ↓ to $\overline{IORQ}$ ↑ Delay	–	110	–	85	–	70
29	TdD(WRf)	Data Stable prior to $\overline{WR}$ ↓	190*	–	80*	–	25*	–
30	TdCf(WRf)	Clock ↓ to $\overline{WR}$ ↓ Delay		90		80		70
31	TwWR	$\overline{WR}$ Pulse Width	360*	–	220*	–	135*	–
32	TdCf(WRr)	Clock ↓ to $\overline{WR}$ ↑ Delay	–	100	–	80	–	70
33	TdD(WRf)	Data Stable prior to $\overline{WR}$ ↓	20*	–	- 10*	–	- 55*	–
34	TdCr(WRf)	Clock ↑ to $\overline{WR}$ ↓ Delay	–	80	–	65	–	60
35	TdWRr(D)	Data Stable from $\overline{WR}$ ↑	120*		60*		30*	
36	TdCf(HALT)	Clock ↓ to $\overline{HALT}$ ↑ or ↓	–	300	–	300	–	260
37	TwNMI	$\overline{NMI}$ Pulse Width	80	–	80	–	70	–
38	TsBUSREQ(Cr)	$\overline{BUSREQ}$ Setup Time to Clock ↑	80	–	50	–	50	–
39	ThBUSREQ(Cr)	$\overline{BUSREQ}$ Hold Time after Clock ↑	0	–	0	–	0	–
40	TdCr(BUSACKf)	Clock ↑ to $\overline{BUSACK}$ ↓ Delay		120		100		90
41	TdCf(BUSACKr)	Clock ↓ to $\overline{BUSACK}$ ↑ Delay	–	100	–	100	–	90
42	TdCr(Dz)	Clock ↑ to Data Float Delay	–	90	–	90	–	80
43	TdCr(CTz)	Clock ↑ to Control Outputs Float Delay ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$)	–	110	–	80	–	70
44	TdCr(Az)	Clock ↑ to Address Float Delay	–	110	–	90	–	80

45	TdCTr(A)	Address Stable after $\overline{MREQ}$ ↑, $\overline{IORQ}$ ↑, $\overline{RD}$ ↑, and $\overline{WR}$ ↑	160*		80*		35*	
46	TsRESET(Cr)	$\overline{RESET}$ to Clock ↑ Setup Time	90	–	60	–	60	–
47	ThRESET(Cr)	$\overline{RESET}$ to Clock ↑ Hold Time	–	0	–	0	–	0
48	TsINTd(Cr)	$\overline{INT}$ to Clock ↑ Setup Time	80	–	80	–	70	–
49	ThINTr(Cr)	$\overline{INT}$ to Clock ↑ Hold Time	–	0	–	0	–	0
50	TdMlf(IROQf)	$\overline{M1}$ ↓ to $\overline{IORQ}$ ↓ Delay	920*		565*		365*	
51	TdCf(IROQf)	Clock ↓ to $\overline{IORQ}$ ↓ Delay	–	110	–	85	–	70
52	TdCf(IORQr)	Clock ↑ to $\overline{IORQ}$ ↑ Delay	–	100	–	85	–	70
53	TdCf(D)	Clock ↓ to Data Valid Delay	–	230	–	150	–	130

OBS.:

1 – (*) Para períodos de clock diferentes do valor mínimo, consulte o manual da ZILOG.

2 – Nas medidas dos parâmetros de tempo foi assumida uma carga capaciativa de 50pF. Cada carga de 50pF adicional, ocasiona um atraso de 10ns. Os valores máximos para as barras de dados e de endereçamento e controle são, respectivamente, 200pF e 100pF.

B CONJUNTO DE INSTRUÇÕES

O objetivo deste apêndice é apresentar, sob a forma de tabelas, as instruções do Z-80. Preferimos dividir este apêndice em duas partes. Na primeira apresentamos um resumo das instruções sem os códigos de máquina correspondentes, que estão apresentados na segunda parte.

B.1 Resumo das Instruções

B.1.1 Grupo de Transferências de 8 bits

Mnemônico	Operação	Bits de Condição S Z H P/V N C	Nº de Bytes	Nº de M Ciclos	Nº de Estados
LD r,r'	r ← r'	• • X • X • • •	1	1	4
LD r, n	r ← n	• • X • X • • •	2	2	7
LD r, (HL)	r ← (HL)	• • X • X • • •	1	2	7
LD r, (IX + d)	r ← (IX + d)	• • X • X • • •	3	5	19
LD r, (IY + d)	r ← (IY + d)	• • X • X • • •	3	5	19
LD (HL), r	(HL) ← r	• • X • X • • •	1	2	7
LD (IX + d), r	(IX + d) ← r	• • X • X • • •	3	5	19
LD (IY + d), r	(IY + d) ← r	• • X • X • • •	3	5	19
LD (HL), n	(HL) ← n	• • X • X • • •	2	3	10
LD (IX + d),n	(IX + d) ← n	• • X • X • • •	4	5	19
LD (IY + d),n	(IY + d) ← n	• • X • X • • •	4	5	19
LD A, (BC)	A ← (BC)	• • X • X • • •	1	2	7
LD A, (DE)	A ← (DE)	• • X • X • • •	1	2	7
LD A, (nn)	A ← (nn)	• • X • X • • •	3	4	13
LD (BC),A	(BC) ← A	• • X • X • • •	1	2	7
LD (DE),A	(DE) ← A	• • X • X • • •	1	2	7
LD (nn), A	(nn) ← A	• • X • X • • •	3	4	13
LD A, I	A ← I	↕ ↕ X 0 X IFF 0 •	2	2	9
LD A, R	A ← R	↕ ↕ X 0 X IFF 0 •	2	2	9
LD I, A	I ← A	• • X • X • • •	2	2	9
LD R, A	R ← A	• • X • X • • •	2	2	9

OBS: 1 – r,r' são os registradores A, B, C, D, E, H ou L
2 – Em relação aos bits de condição temos:
x – Não importa
● – Inalterado
↕ – Modificado
IFF – Conteúdo do Flip-Flop enable interrupt
3 – n é uma constante de 8 bits
4 – nn é uma constante de 16 bits
5 – Os parênteses indicam posições de memória.

B. 1.2 Grupo de transferências de 16 bits

Mnemônico	Operação	Bits de Condição S Z H P/V N C	Nº de Bytes	Nº de M Ciclos	Nº de Estados
LD dd,nn	dd ← nn	● ● X ● X ● ● ●	3	3	10
LD IX, nn	IX ← nn	● ● X ● X ● ● ●	4	4	14
LD IY, nn	IY ← nn	● ● X ● X ● ● ●	4	4	14
LD HL, (nn)	H ← (nn + 1) L ← (nn)	● ● X ● X ● ● ●	3	5	16
LD dd, (nn)	dd_H ← (nn + 1) dd_L ← (nn)	● ● X ● X ● ● ●	4	6	20
LD IX, (nn)	IX_H ← (nn + 1) IX_L ← (nn)	● ● X ● X ● ● ●	4	6	20
LD IY, (nn)	IY_H ← (nn + 1) IY_L ← (nn)	● ● X ● X ● ● ●	4	6	20
LD (nn), HL	(nn + 1) ← H (nn) ← L	● ● X ● X ● ● ●	3	5	16
LD (nn), dd	(nn + 1) ← dd_H (nn) ← dd_L	● ● X ● X ● ● ●	4	6	20
LD (nn), IX	(nn + 1) ← IX_H (nn) ← IX_L	● ● X ● X ● ● ●	4	6	20
LD (nn), IY	(nn + 1) ← IY_H (nn) ← IY_L	● ● X ● X ● ● ●	4	6	20
LD SP, HL	SP ← HL	● ● X ● X ● ● ●	1	1	6
LD SP, IX	SP ← IX	● ● X ● X ● ● ●	2	2	10
LD SP, IY	SP ← IY	● ● X ● X ● ● ●	2	2	10
PUSH qq	(SP – 2) ← qqL (SP – 1) ← qqH SP → SP – 2	● ● X ● X ● ● ●	1	3	11
PUSH IX	(SP – 2) ← IX_L (SP – 1) ← IX_H SP → SP – 2	● ● X ● X ● ● ●	2	4	15
PUSH IY	(SP – 2) ← IY_L (SP – 1) ← IY_H SP → SP – 2	● ● X ● X ● ● ●	2	4	15
POP qq	qqH ← (SP + 1) qqL ← (SP) SP → SP + 2	● ● X ● X ● ● ●	1	3	10
POP IX	IX_H ← (SP + 1) IX_L ← (SP) SP → SP + 2	● ● X ● X ● ● ●	2	4	14
POP IY	IY_H ← (SP + 1) IY_L ← (SP) SP → SP + 2	● ● X ● X ● ● ●	2	4	14

OBS.: 1 – **dd** é um dos pares BC, DE, HL ou o registrador SP
2 – **qq** é um dos pares AF, BC, DE ou HL
3 – os índices L e H indicam, respectivamente, os oito bits menos significativos e os oito bits mais significativos de 16 bits.

B. 1.3 Grupo de troca, transferências e pesquisa de blocos

Mnemônico	Operação	Bits de Condição S Z H P/V N C	Nº de Bytes	Nº de M Ciclos	Nº de Estados
EX DE, HL.	DE ↔ HL	• • X • X • • •	1	1	4
EX AF, AF'	AF ↔ AF'	• • X • X • • •	1	1	4
EXX	BC ↔ BC' DE ↔ DE' HL ↔ HL'	• • X • X • • •	1	1	4
EX (SP), HL	H ↔ (SP + 1) L ↔ (SP)	• • X • X • • •	1	5	19
EX (SP), IX	IX_H ↔ (SP + 1) IX_L ↔ (SP)	• • X • X • • •	2	6	23
EX (SP), IY	IY_H ↔ (SP + 1) IY_L ↔ (SP)	• • X • X • • •	2	6	23
LDI	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1	• • X 0 X ↕ (1) 0 •	2	4	16
LDIR	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1 Repetir até BC = 0	• • X 0 X 0 0 •	2 2	5 4	21 16
LDD	(DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1	• • X 0 X ↕ (1) 0 •	2	4	16
LDDR	(DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1 Repetir até BC = 0	• • X 0 X 0 0 •	2 2	5 4	21 16
CPI	A − (HL) HL ← HL + 1 BC ← BC − 1	↕ (2) ↕ X ↕ X ↕ (1) 1 •	2	4	16
CPIR	A − (HL) HL ← HL + 1 BC ← BC − 1 Repetir até A = (HL) ou BC = 0	↕ (2) ↕ X ↕ X ↕ (1) 1 •	2 2	5 4	21 16
CPD	A − (HL) HL ← HL − 1 BC ← BC − 1	↕ (2) ↕ X ↕ X ↕ (1) 1 •	2	4	16
CPDR	A − (HL) HL ← HL − 1 BC ← BC − 1 Repetir até A = (HL) ou BC = 0	↕ (2) ↕ X ↕ X ↕ (1) 1 •	2 2	5 4	21 16

OBS.: 1 - Bit P/V = 0 se BC − 1 = 0, caso contrário P/V = 1
2 Bit Z = 1 se A = (HL), caso contrário Z = 0

B. 1.4 Grupo de Operações Lógicas e Aritméticas de 8 bits

Mnemônico	Operação	Bits de Condição S Z H P/V N C	Nº de Bytes	Nº de M. Ciclos	Nº de Estados
ADD A,r	A ← A + r	↕ ↕ X ↕ X V O ↕	1	1	4
ADD A, n	A ← A + n	↕ ↕ X ↕ X V O ↕	2	2	7
ADD A, (HL)	A ← A + (HL)	↕ ↕ X ↕ X V O ↕	1	2	7
ADD A, (IX + d)	A ← A + (IX + d)	↕ ↕ X ↕ X V O ↕	3	5	19
ADD A, (IY + d)	A ← A + (IY + d)	↕ ↕ X ↕ X V O ↕	3	5	19
ADC A,s	A ← A + s + CY	↕ ↕ X ↕ X V O ↕			
SUB s	A ← A − s	↕ ↕ X ↕ X V 1 ↕			
SBC A,s	A ← A − s − CY	↕ ↕ X ↕ X V 1 ↕			
AND s	A ← A ∧ s	↕ ↕ X 1 X P O O			
ORs	A ← A ∨ s	↕ ↕ X O X P O O			
XOR s	A ← A ⊕ s	↕ ↕ X O X P O O			
CP s	A − s	↕ ↕ X ↕ X V 1 ↕			
INC r	r ← r + 1	↕ ↕ X ↕ X V O •	1	1	4
INC (HL)	(HL) ← (HL) + 1	↕ ↕ X ↕ X V O •	1	3	11
INC (IX + d)	(IX + d) ← (IX + d) + 1	↕ ↕ X ↕ X V O •	3	6	23
INC (IY + d)	(IY + d) ← (IY + d) + 1	↕ ↕ X ↕ X V O •	3	6	23
DEC m	m ← m − 1	↕ ↕ X ↕ X V 1 •			

OBS.: 1 – s pode ser r, n, (HL), (IX + d) e (IY + d)
2 – m pode ser r, (HL), (IX + d) e (IY + d)

B. 1.5 Grupo de Controle e Operações Aritméticas de Uso Geral

Mnemônico	Operação	Bits de Condição S Z H P/V N C	Nº de Bytes	Nº de M. Ciclos	Nº de Estados
DAA	Ajuste Decimal	↕ ↕ X ↕ X P • ↕	1	1	4
CPL	$A \leftarrow \overline{A}$	• • X 1 X • 1 •	1	1	4
NEG	A ← 0 − A	↕ ↕ X ↕ X V 1 ↕	2	2	8
CCF	$CY \leftarrow \overline{CY}$	• • X X X • 0 ↕	1	1	4
SCF	CY ← 1	• • X 0 X • 0 1	1	1	4
NOP		• • X • X • • •	1	1	4
HALT		• • X • X • • •	1	1	4
DI	IFF ← 0	• • X • X • • •	1	1	4
EI	IFF ← 1	• • X • X • • •	1	1	4
IM 0	Seta modo 0	• • X • X • • •	2	2	8
IM 1	Seta modo 1	• • X • X • • •	2	2	8
IM 2	Seta modo 2	• • X • X • • •	2	2	8

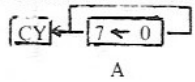
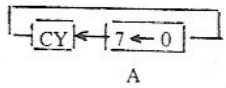
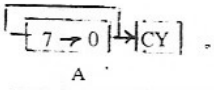
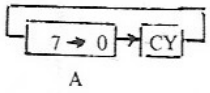
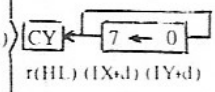
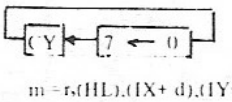
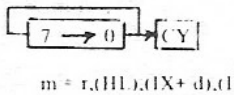
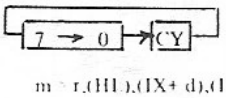
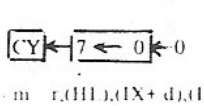
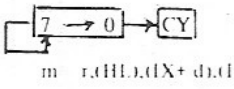
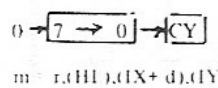
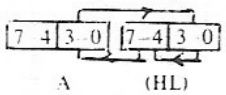
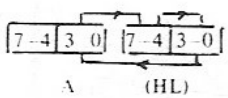
OBS.: 1 – IFF – Flip-Flop enable interrupt

B. 1.6 Grupo de Operações Aritméticas de 16 bits

Mnemônico	Operação	Bits de Condição S Z H P/V N C	Nº de Bytes	Nº de M. Ciclos	Nº de Estados
ADD HL,ss	HL←HL + ss	• • X X X • 0 ↕	1	3	11
ADC HL,ss	HL←HL + ss + CY	↕ ↕ X X X V 0 ↕	2	4	15
SBC HL, ss	HL←HL-ss-CY	↕ ↕ X X X V I ↕	2	4	15
ADD IX, pp	IX←IX + pp	• • X X X • 0 ↕	2	4	15
ADD IY, rr	IY←IY+rr	• • X X X • 0 ↕	2	4	15
INC ss	ss←ss + 1	• • X • X • • •	1	1	6
INC IX	IX←IX + 1	• • X • X • • •	2	2	10
INC IY	IY←IY + 1	• • X • X • • •	2	2	10
DEC ss	ss←ss–1	• • X • X • • •	1	1	6
DEC IX	IX←IX –1	• • X • X • • •	2	2	10
DEC IY	IY←IY–1	• • X • X • • •	2	2	10

OBS: 1 – **ss** são os registradores BC, DE, HL ou SP
2 – **pp** são os registradores BC, DE, IX ou SP
3 – **rr** são os registradores BC, DE, IY ou SP

B. 1.7 Grupo de Rotação e Deslocamento

Mnemônico	Operação	Bits de Condição S Z H P/V N C	Nº de Bytes	Nº de M. Ciclos	Nº de Estados
RLCA		• • X 0 X • 0 ↕	1	1	4
RLA		• • X 0 X • 0 ↕	1	1	4
RRCA		• • X 0 X • 0 ↕	1	1	4
RRA		• • X 0 X • 0 ↕	1	1	4
RLC r		↕ ↕ X 0 X P 0 ↕	2	2	8
RLC (HL)		↕ ↕ X 0 X P 0 ↕	2	4	15
RLC (IX + d)		↕ ↕ X 0 X P 0 ↕	4	6	23
RLC (IY+ d)		↕ ↕ X 0 X P 0 ↕	4	6	23
RL m		↕ ↕ X 0 X P 0 ↕			
RRC m		↕ ↕ X 0 X P 0 ↕			
RR m		↕ ↕ X 0 X P 0 ↕			
SLA m		↕ ↕ X 0 X P 0 ↕			
SRA m		↕ ↕ X 0 X P 0 ↕			
SRL m		↕ ↕ X 0 X P 0 ↕			
RLD		↕ ↕ X 0 X P 0 •	2	5	18
RRD		↕ ↕ X 0 X P 0 •	2	5	18

B. 1.8 Grupo de Teste, Reset e Teste Bit

Mnemônico	Operação	Bits de Condição S Z H P/V N C	Nº de Bytes	Nº de M Ciclos	Nº de Estados
BIT b,r	$Z \leftarrow \overline{r_b}$	X ↕ X 1 X X 0 •	2	2	8
BIT b, (HL)	$Z \leftarrow \overline{(HL)_b}$	X ↕ X 1 X X 0 •	2	3	12
BIT b, (IX + d)b	$Z \leftarrow \overline{(IX + d)_b}$	X ↕ X 1 X X 0 •	4	5	20
BIT b, $(IY + d)_b$	$Z \leftarrow \overline{(IY + d)_b}$	X ↕ X 1 X X 0 •	4	5	20
SET b,r	$r_b \leftarrow 1$	• • X • X • • •	2	2	8
SET b, (HL)	$(HL)_b \leftarrow 1$	• • X • X • • •	2	4	15
SET b, (IX + d)	$(IX + d)b \leftarrow 1$	• • X • X • • •	4	6	23
SET b,(IY + d)	$(IY + d)_b \leftarrow 1$	• • X • X • • •	4	6	23
RES b,m	$m_b \leftarrow 0$ m ≡ r, (HL), (IX + d), (IY + d)	• • X • X • • •			

OBS.: 1 -- b indica a posição relativa dos bits (7 – 0)

B. 1.9 Grupo de Desvios

Mnemônico	Operação	Bits de Condição S Z H P/V N C	Nº de Bytes	Nº de M Ciclos	Nº de Estados
JP nn	PC ← nn	• • X • X • • •	3	3	10
JP cc, nn	se a condição cc é verdadeira PC ← nn, caso contrário continua	• • X • X • • •	3	3	10
JR e	PC ← PC + e	• • X • X • • •	3	3	12
JR C, e	If C = 0 continua	• • X • X • • •	2	2	7
	If C = 1, PC ← PC + e		2	3	12
JR NC, e	If C = 1, continua	• • X • X • • •	2	2	7
	If C = 0, PC ← PC + e		2	3	12
JP Z,e	If Z = 0 continua	• • X • X • • •	2	2	7
	If Z = 1, PC ← PC + e		2	3	12
JR NZ, e	If Z = 1, continua	• • X • X • • •	2	2	7
	If Z = 0 PC ← PC + e		2	3	12
JP (HL)	PC ← HL	• • X • X • • •	1	1	4
JP (IX)	PC ← IX	• • X • X • • •	2	2	8
JP (IY)	PC ← IY	• • X • X • • •	2	2	8
DJNZ, e	B ← B − 1 If B = 0, continua	• • X • X • • •	2	2	8
	If B ≠ 0, PC ← PC + e		2	3	13

OBS. 1 cc indica a condição de desvio

NZ	Z = 0
Z	Z = 1
NC	CY = 0
C	CY = 1
PO	Paridade impar
PE	Paridade par
P	S = 0
M	S = 1

B. 1.10 Grupo de Chamada e Retorno de Sub-rotinas

Mnemônico	Operação	Bits de Condição S Z H P/V N C	Nº de Bytes	Nº de M Ciclos	Nº de Estados
CALL nn	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ $PC \leftarrow nn$	• • X • X • • •	3	5	17
CALL cc, nn	Se a condição é falsa continua, caso contrário, CALL nn	• • X • X • • •	3 3	3 5	10 17
RET	$PC_L \leftarrow (SP)$ $PC_H \leftarrow (SP + 1)$	• • X • X • • •	1	3	10
RET cc	Se a condição cc é falsa continua, caso contrário, RET	• • X • X • • •	1 1	1 3	5 11
RETI	Retorno de interrupção	• • X • X • • •	2	4	14
$RETN_1$	Retorno de non-maskable interrupt	• • X • X • • •	2	4	14
RST_p	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ $PC_H \leftarrow 0$ $PC_L \leftarrow P$	• • X • X • • •	1	3	11

OBS: 1 – RETN carrega IFF2 em IFF1

2 – p pode ser $\phi\phi$H, ϕ8H, 1ϕH, 18H, 20H, 28H, 3ϕH, 38 H

B. 1.11 Grupo de Entrada e Saída

Mnemônico	Operação	Bits de Condição S Z H P/V N C	Nº de Bytes	Nº de M Ciclos	Nº de Estados
IN A,(n)	A ← (n)	• • X • X • • •	2	3	11
IN r, (C)	r ← (C)	↕ ↕ X ↕ X P 0 •	2	3	12
INI	(HL) ← (C) B ← B − 1 HL ← HL + 1	X ↕ X X X X 1 •	2	4	16
INIR	(HL) ← (C) B ← B − 1 HL ← HL + 1 Repetir até B = 0	X 1 X X X X 1 •	2 2	5 (Se B ≠ 0) 4 (Se B = 0)	21 16
IND	(HL) ← (C) B ← B − 1 HL ← HL − 1	X ↕ X X X X 1 •	2	4	16
INDR	(HL) ← (C) B ← B − 1 HL ← HL − 1 Repetir até B = 0	X I X X X X I •	2 2	5 (Se B ≠ 0) 4 (Se B = 0)	21 16
OUT (n), A	(n) ← A	• • X • X • • •	2	3	11
OUT (C), r	(C) ← r	• • X • X • • •	2	3	12
OUTI	(C) ← (HL) B ← B − 1 HL ← HL + 1	X ↕ X X X X 1 •	2	4	16
OTIR	(C) ← (HL) B ← B − 1 HL ← HL + 1 Repetir até B = 0	X 1 X X X X 1 •	2 2	5 (Se B ≠ 0) 4 (Se B = 0)	21 16
OUTD	(C) ← (HL) B ← B − 1 HL ← HL − 1	X ↕ X X X X 1 •	2	4	16
OTDR	(C) ← (HL) B ← B − 1 HL ← HL − 1 Repetir até B = 0	X 1 X X X X 1 •	2 2	5 (Se B ≠ 0) 4 (Se B = 0)	21 16

OBS.: 1 – Se o resultado de B-1 for zero, o bit Z é setado, caso contrário ele é resetado.

B.2 CODIGOS DE MÁQUINA DO Z-80

INSTRUÇÕES DE TRANSFERÊNCIA DE 8 BITS

	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(nn)	n
LD A, .	7F	78	79	7A	7B	7C	7D	7E	ϕA	1A	3AXXXX	3EXX
LD B, .	47	4ϕ	41	42	43	44	45	46				ϕ6XX
LD C, .	4F	48	49	4A	4B	4C	4D	4E				ϕEXX
LD D, .	57	5ϕ	51	52	53	54	55	56				16XX
LD E, .	5F	58	59	5A	5B	5C	5D	5E				1EXX
LD H, .	67	6ϕ	61	62	63	64	65	66				26XX
LD L, .	6F	68	69	6A	6B	6C	6D	6E				2EXX
LD (HL), .	77	7ϕ	71	72	73	74	75					36XX
LD (BC), .	ϕ2											
LD (DE), .	12											
LD (nn), .	32XXXX											

	A	B	C	D	E	H	L
LD . , (IX + d)	DD7EXX	DD46XX	DD4EXX	DD56XX	DD5EXX	DD66XX	DD6EXX
LD . , (IY + d)	FD7EXX	FD46XX	FD4EXX	FD56XX	FD5EXX	FD66XX	FD6EXX
LD (IX + d), .	DD77XX	DD7ϕXX	DD71XX	DD72XX	DD73XX	DD74XX	DD75XX
LD (IY + d), .	FD77XX	FD7ϕXX	FD71XX	FD72XX	FD73XX	FD74XX	FD75XX
LD (IX + d), n		DD36XXXX		LD (IY + d), n		FD36XXXX	

		S	Z	H	P/V	N	C
LD A, I	ED57	*	*	ϕ	*	ϕ	—
LD A, R	ED5F	*	*	ϕ	*	ϕ	—
LD I, A	ED47	—	—	—	—	—	—
LD R, A	ED4F	—	—	—	—	—	—

INSTRUÇÕES DE TRANSFERÊNCIA E PERMUTA DE 16 BITS

	BC	DE	HL	SP	IX	IY
LD . . , nn	ϕ1XXXX	11XXXX	21XXXX	31XXXX	DD21XXXX	FD21XXXX
LD . . , (nn)	ED4BXXXX	ED5BXXXX	2AXXXX	ED7BXXXX	DD2AXXXX	FD2AXXXX
LD (nn), . .	ED43XXXX	ED53XXXX	22XXXX	ED73XXXX	DD22XXXX	FD22XXXX
LD SP, . .			F9		DDF9	FDF9

	BC	DE	HL	AF	IX	IY
PUSH . .	C5	D6	E5	F5	DDE5	FDE5
POP . .	C1	D1	E1	F1	DDE1	FDE1

EX (SP), HL	E3	EX DE, HL	EB	
EX (SP), IX	DDE3	EX AF, AF'	ϕ8	
EX (SP), IY	FDE3	EXX	D9	BC-BC' DE-DE' HL-HL'

INSTRUÇÕES DE TRANSFERÊNCIA DE BLOCOS E PESQUISA

		S	Z	H	P/V	N	C	
LDI	EDAϕ	—	—	ϕ	*	ϕ	—	LD (DE), (HL); INC HL; INC DE; DEC BC
LDIR	EDBϕ	—	—	ϕ	ϕ	ϕ	—	COMO LDI, REPETIR ATÉ BC = ϕ
LDD	EDA8	—	—	ϕ	*	ϕ	—	LD (DE), (HL); DEC HL; DEC DE; DEC BC
LDDR	EDB8	—	—	ϕ	ϕ	ϕ	—	COMO LDD, REPETIR ATÉ BC = ϕ
CPI	EDA1	*	*	*	*	1	—	CP (HL); INC HL; DEC BC
CPIR	EDB1	*	*	*	*	1	—	COMO CPI, REPETIR ATÉ BC = ϕ ou A = (HL)

CPD	EDA9	*	*	*	*	1	–	CP (HL); DEC HL; DEC BC
CPDR	EDB9	*	*	*	*	1	–	COMO CPD, REPETIR ATÉ BC = ϕ ou A = (HL)

"JUMP'S", "CALL'S" e "RETURN'S"

	Z	NZ	C	NC	PE	PO	M	P
JP	CAXXXX	C2XXXX	DAXXXX	D2XXXX	EAXXXX	E2XXXX	FAXXXX	F2XXXX
CALL	CCXXXX	C4XXXX	DCXXXX	D4XXXX	ECXXXX	E4XXXX	FCXXXX	F4XXXX
RET	C8	Cϕ	D8	Dϕ	E8	Eϕ	F8	Fϕ
JR	28XX	2ϕXX	38XX	3ϕXX				

	INCONDICIONAIS	(HL)	(IX)	(IY)
JP	C3XXXX	E9	DDE9	FDE9
CALL	CD XXXX			
RET	Cϕ			
JR	18XX			

RST	$\phi\phi$	ϕ8	1ϕ	18	2ϕ	28	3ϕ	38
	C7	CF	D7	DF	E7	EF	F7	FF

DJNZ	1ϕXX	DEC B; JR NZ, d
RETI	ED4D	RETORNO DE INTERRUPÇÃO
RETN	ED45	RETORNO DE INTERRUPÇÃO NÃO MASCARADA

INSTRUÇÕES DE CONTROLE

		S	Z	H	P/V	N	C	
NOP	$\phi\phi$	–	–	–	–	–	–	INSTRUÇÃO DESTITUÍDA DE OPERAÇÃO
HALT	76	–	–	–	–	–	–	
CCF	3F	–	–	?	–	ϕ	*	COMPLEMENTA O BIT CARRY
SCF	37	–	–	ϕ	–	ϕ	1	SETA O BIT CARRY
EI	FB	–	–	–	–	–	–	HABILITA INTERRUPÇÃO
DI	F3	–	–	–	–	–	–	INIBE INTERRUPÇÃO
IM ϕ	ED46	–	–	–	–	–	–	INTERRUPÇÃO MODO - 0
IM 1	ED56	–	–	–	–	–	–	INTERRUPÇÃO MODO - 1
IM 2	ED5E	–	–	–	–	–	–	INTERRUPÇÃO MODO - 2

INSTRUÇÕES DE ENTRADA E SAÍDA

	A	B	C	D	E	H	L	S	Z	H	P/V	N	C
IN . , (C)	ED78	ED4ϕ	ED48	ED5ϕ	ED58	ED6ϕ	ED68	*	*	*	*	ϕ	–
OUT (C) , .	ED79	ED41	ED49	ED51	ED59	ED61	ED69	–	–	–	–	–	–

		S	Z	H	P/V	N	C	
IN A,(n)	DBXX	–	–	–	–	–	–	
OUT(n), A	D3XX	–	–	–	–	–	–	
INI	EDA2	?	*	?	?	1	–	IN (HL), (C); INC HL, DEC B
INIR	EDB2	?	1	?	?	1	–	COMO INI, REPETIR ATÉ B = ϕ
IND	EDAA	?	*	?	?	1	–	IN (HL), (C); DEC HL, DEC B
INDR	EDBA	?	1	?	?	1	–	COMO IND, REPETIR ATÉ B = ϕ
OUTI	EDA3	?	*	?	?	1	–	OUT (C), (HL); INC HL; DEC B
OTIR	EDB3	?	1	?	?	1	–	COMO OUTI, REPETIR ATÉ B = ϕ
OUTD	EDAB	?	*	?	?	1	–	OUT (C), (HL); DEC HL; DEC B
OTDR	EDBB	?	1	?	?	1	–	COMO OUTD, REPETIR ATÉ B = ϕ

INSTRUÇÕES LÓGICAS E ARITMÉTICAS DE 8 BITS

	B	C	D	E	H	L	(HL)	A	n	(IX + d)	(IY + d)	S	Z	H	P/V	N	C
ADD .	80	81	82	83	84	85	86	87	C6XX	DD86XX	FD86XX	*	*	*	*	0	*
ADC .	88	89	8A	8B	8C	8D	8E	8F	CEXX	DD8EXX	FD8EXX	*	*	*	*	0	*
SUB .	90	91	92	93	94	95	96	97	D6XX	DD96XX	FD96XX	*	*	*	*	1	*
SBC .	98	99	9A	9B	9C	9D	9E	9F	DEXX	DD9EXX	FD9EXX	*	*	*	*	1	*
AND .	A0	A1	A2	A3	A4	A5	A6	A7	E6XX	DDA6XX	FDA6XX	*	*	1	*	0	0
XOR .	A8	A9	AA	AB	AC	AD	AE	AF	EEXX	DDAEXX	FDAEXX	*	*	0	*	0	0
OR .	B0	B1	B2	B3	B4	B5	B6	B7	F6XX	DDB6XX	FDB6XX	*	*	0	*	0	0
CP .	B8	B9	BA	BB	BC	BD	BE	BF	FEXX	DDBEXX	FDBEXX	*	*	*	*	1	*
INC .	04	0C	14	1C	24	2C	34	3C		DD34XX	FD34XX	*	*	*	*	0	—
DEC .	05	0D	15	1D	25	2D	35	3D		DD35XX	FD35XX	*	*	*	*	1	—

		S	Z	H	P/V	N	C	
DAA	27	*	*	*	*	—	*	AJUSTE DE RESULTADO BCD NO AC
CPL	2F	—	—	1	—	1	—	COMPLEMENTO A UM DO AC
NEG	ED44	*	*	*	*	1	*	COMPLEMENTO A DOIS DO AC

NSTRUÇÕES ARITMÉTICAS DE 16 BITS

		BC	DE	HL	SP	IX	IY	S	Z	H	P/V	N	C
INC		03	13	23	33	DD23	FD23	—	—	—	—	—	—
DEC		0B	1B	2B	3B	DD2B	FD2B	—	—	—	—	—	—
ADD	HL , .	09	19	29	39			—	—	?	—	0	*
ADC	HL , .	ED4A	ED5A	ED6A	ED7A			*	*	?	*	0	*
SBC	HL , .	ED42	ED52	ED62	ED72			*	*	?	*	1	*
ADD	IX , .	DD09	DD19		DD39	DD29		—	—	?	—	0	*
ADD	IY , .	FD09	FD19		FD39		FD29	—	—	?	—	0	*

INSTRUÇÕES DE DESLOCAMENTO E ROTAÇÃO

	B	C	D	E	H	L	(HL)	A	(IX + d)	(IY + d)
RR	CB18	CB19	CB1A	CB1B	CB1C	CB1D	CB1E	CB1F	DDCBXX1E	FDCBXX1E
RL	CB10	CB11	CB12	CB13	CB14	CB15	CB16	CB17	DDCBXX16	FDCBXX16
RRC	CB08	CB09	CB0A	CB0B	CB0C	CB0D	CB0E	CB0F	DDCBXX0E	FDCBXX0E
RLC	CB00	CB01	CB02	CB03	CB04	CB05	CB06	CB07	DDCBXX06	FDCBXX06
SRA	CB28	CB29	CB2A	CB2B	CB2C	CB2D	CB2E	CB2F	DDCBXX2E	FDCBXX2E
SLA	CB20	CB21	CB22	CB23	CB24	CB25	CB26	CB27	DDCBXX26	FDCBXX26
SRL	CB38	CB39	CB3A	CB3B	CB3C	CB3D	CB3E	CB3F	DDCBXX3E	FDCBXX3E

	S	Z	H	P/V	N	C	
RR/RL	*	*	0	*	0	*	DESLOC. CIRC. À DIR./ESQ. ATRAVÉS DO CARRY
RRC/RLC	*	*	0	*	0	*	DESLOC. CIRC. À DIR./ESQ.
SRA/SLA	*	*	0	*	0	*	DESLOC. ARITMÉTICO À DIR./ESQ.
SRL	*	*	0	*	0	*	DESLOC. LÓGICO À DIR./ESQ.

		S	Z	H	P/V	N	C	
RRCA	0F	—	—	0	—	0	*	DESLOC. CIRC. DO AC À DIR.
RLCA	07	—	—	0	—	0	*	DESLOC. CIRC. DO AC À ESQ.
RRA	1F	—	—	0	—	0	*	DESLOC. CIRC. DO AC À DIR. ATRAVÉS DO CARRY
RLA	17	—	—	0	—	0	*	DESLOC. CIRC. DO AC À ESQ. ATRAVÉS DO CARRY
RLD	ED6F	*	*	0	*	0	—	DESLOC. CIRC. DE DÍGITO À ESQ. ATRAVÉS DO AC E (HL)
RRD	ED67	*	*	0	*	0	—	DESLOC. CIRC. DE DÍGITO À DIR. ATRAVÉS DO AC E (HL)

INSTRUÇÕES DE MANIPULAÇÃO E TESTE DE BITS

		B	C	D	E	H	L	(HL)	A	(IX + d)	(IY + d)
BIT	ϕ	CB4ϕ	CB41	CB42	CB43	CB44	CB45	CB46	CB47	DDCBXX46	FDCBXX46
BIT	1	CB48	CB49	CB4A	CB4B	CB4C	CB4D	CB4E	CB4F	DDCBXX4E	FDCBXX4E
BIT	2	CB5ϕ	CB51	CB52	CB53	CB54	CB55	CB56	CB57	DDCBXX56	FDCBXX56
BIT	3	CB58	CB59	CB5A	CB5B	CB5C	CB5D	CB5E	CB5F	DDCBXX5E	FDCBXX5E
BIT	4	CB6ϕ	CB61	CB62	CB63	CB64	CB65	CB66	CB67	DDCBXX66	FDCBXX66
BIT	5	CB68	CB69	CB6A	CB6B	CB6C	CB6D	CB6E	CB6F	DDCBXX6E	FDCBXX6E
BIT	6	CB7ϕ	CB71	CB72	CB73	CB74	CB75	CB76	CB77	DDCBXX76	FDCBXX76
BIT	7	CB78	CB79	CB7A	CB7B	CB7C	CB7D	CB7E	CB7F	DDCBXX7E	FDCBXX7E
RES	ϕ	CB8ϕ	CB81	CB82	CB83	CB84	CB85	CB86	CB87	DDCBXX86	FDCBXX86
RES	1	CB88	CB89	CB8A	CB8B	CB8C	CB8D	CB8E	CB8F	DDCBXX8E	FDCBXX8E
RES	2	CB9ϕ	CB91	CB92	CB93	CB94	CB95	CB96	CB97	DDCBXX96	FDCBXX96
RES	3	CB98	CB99	CB9A	CB9B	CB9C	CB9D	CB9E	CB9F	DDCBXX9E	FDCBXX9E
RES	4	CBAϕ	CBA1	CBA2	CBA3	CBA4	CBA5	CBA6	CBA7	DDCBXXA6	FDCBXXA6
RES	5	CBA8	CBA9	CBAA	CBAB	CBAC	CBAD	CBAE	CBAF	DDCBXXAE	FDCBXXAE
RES	6	CBBϕ	CBB1	CBB2	CBB3	CBB4	CBB5	CBB6	CBB7	DDCBXXB6	FDCBXXB6
RES	7	CBB8	CBB9	CBBA	CBBB	CBBC	CBBD	CBBE	CBBF	DDCBXXBE	FDCBXXBE
SET	ϕ	CBCϕ	CBC1	CBC2	CBC3	CBC4	CBC5	CBC6	CBC7	DDCBXXC6	FDCBXXC6
SET	1	CBC8	CBC9	CBCA	CBCB	CBCC	CBCD	CBCE	CBCF	DDCBXXCE	FDCBXXCE
SET	2	CBDϕ	CBD1	CBD2	CBD3	CBD4	CBD5	CBD6	CBD7	DDCBXXD6	FDCBXXD6
SET	3	CBD8	CBD9	CBDA	CBDB	CBDC	CBDD	CBDE	CBDF	DDCBXXDE	FDCBXXDE
SET	4	CBEϕ	CBE1	CBE2	CBE3	CBE4	CBE5	CBE6	CBE7	DDCBXXE6	FDCBXXE6
SET	5	CBE8	CBE9	CBEA	CBEB	CBEC	CBED	CBEE	CBEF	DDCBXXEE	FDCBXXEE
SET	6	CBFϕ	CBF1	CBF2	CBF3	CBF4	CBF5	CBF6	CBF7	DDCBXXF6	FDCBXXF6
SET	7	CBF8	CBF9	CBFA	CBFB	CBFC	CBFD	CBFE	CBFF	DDCBXXFE	FDCBXXFE

FLAGS INFLUENCIADOS:

	S	Z	H	P/V	N	C
BIT	?	*	1	?	ϕ	—
SET	—	—	—	—	—	—
RES	—	—	—	—	—	—

NOTAS ADICIONAIS SOBRE O REGISTRO F:

Bit	7	6	5	4	3	2	1	ϕ
	S	Z	X	H	X	P/V	N	C

		SETADO	RESETADO	SETADO SE
C	CARRY-BIT	C	NC	"VAI UM" DO BIT 7
N	BIT SOMA/SUBTRAÇÃO			OPERAÇÃO DE SUBTRAÇÃO
P/V	BIT PARIDADE OVERFLOW	PE	PO	PARIDADE PAR/OVERFLOW
H	BIT HALF CARRY			"VAI UM" DO BIT 3
Z	BIT ZERO	Z	NZ	RESULTADO ZERO
S	BIT SINAL	M	P	RESULTADO NEGATIVO
X	NÃO USADO			

OPERAÇÕES DOS FLAGS:

1 SETADO
ϕ RESETADO
* DEPENDE DO RESULTADO DA OPERAÇÃO
— NÃO AFETADO
? INDETERMINADO

C TABELA DE ASCII

Na tabela abaixo temos todos os caracteres existentes no código ASCII (American National Standard Code for Information Interchange). Este código possui sete bits, sendo que o mais significativo é reservado para a paridade. Nesta tabela, consideramos este bit de paridade igual a zero.

DÍGITO HEXADECIMAL MAIS SIGNIFICATIVO	DÍGITO HEXADECIMAL MENOS SIGNIFICATIVO															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	—	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

BIBLIOGRAFIA

1. SERRA, C. P.: "Prática de Programação do 8080A", Edições Monitor, SP, 1981.
2. VISCONTI, A. C. J. F.: "Microprocessador 8080 e 8085", Livros Érica Editora Ltda, SP, 1982.
3. HILL, F. J., PETERSON, G. R.: "Digital Systems: Hardware Organization and Design", John Wiley & Sons, Inc., New York, 1973.
4. BARDEN, W. J.: "The Z-80 Microcomputer Handbook", Howard W. Sams & Co., Inc., Indiana, 1981.
5. BARDEN, W. J.: "Z-80 Microcomputer Design Projects", Howard W. Sams & Co., Inc., Indiana, 1982.
6. KANE, J., OSBORNE, A.: "An Introduction to Microcomputers", Volume 3, Osborne & Associates. Inc., California, 1979.
7. LEVENTHAL, A. L.: "Z-80 Assembly Language Programming", Osborne/McGraw-Hill, California, 1979.
8. KHAMBATA, A. J.: "Introduction to the Z-80 Microcomputer", John Wiley & Sons, New York, 1982.
9. TAILLIAR, A., HARBERT, D.: "Les tendances actuelles des microprocessurs 8 bits", Micro Systemes, Société Parisienne d'Edition, Paris, Nov/Dec 1982.
10. KANIS, W.: "Der Z-80-EMUF", MC Die Mikrocomputer-Zeitschrift, München, April, 1983.
11. ZILOG: "Data Book", Zilog Inc., 1981.
12. SGS: "Z-80 Microcomputer Systems", SGS-ATES Group of Companies, 1981.
13. SGS: "Databook COS/MOS B-Series Devices", SGS-ATES Group of Companies, 1981.
14. INTEL: "Component Data Catalog", Intel Corp., 1981.
15. INTEL: "MCS-80/85 Family User's Manual", Intel Corp., 1979.
16. INTEL: "Memory Design Handbook", Intel Corp., 1981.
17. SIGNETICS: "TTL Logic Data Manual", Signetics Corp., 1982.
18. HITACHI: "Data Sheet HM6116P-2/3/4", 1981.

ÍNDICE REMISSIVO

Impresso na Gráfica MEC Editora Ltda.
Rua Visconde de Santa Isabel, 420 – Grajaú – Rio
Tels.: 288-0044 – 288-8221 – 288-8375

SOBRE OS AUTORES

André Gil Rubens

- Engenheiro Eletrônico formado pela PUC/RJ em dezembro de 1975.
- Mestre em Ciência de Engenharia Elétrica pela PUC/RJ em fevereiro de 1980.
- Engenheiro da Embratel desde 1976, onde trabalha no desenvolvimento de hardware e software de microcomputadores.
- Professor Titular da Faculdade Politécnica Estácio de Sá.
- Consultor Técnico da Revista Interface.

Ney Acyr Rodrigues de Oliveira

- Engenheiro Eletrônico formado pela Escola de Engenharia da UFRJ em dezembro de 1977.
- Pós-Graduação em Sistemas Digitais na COPPE/UFRJ.
- Engenheiro da Embratel desde 1978, onde trabalha no desenvolvimento de hardware e software de microcomputadores.
- Professor Titular da Faculdade Politécnica Estácio de Sá.
- Consultor Técnico da revista Interface.

AeN

CONSULTORIA PROJETOS E PUBLICAÇÕES LTDA