Engº Luiz Benedito Cypriano
Engº Paulo Roberto Cardinali

MICROPROCESSADOR Z80

HARDWARE VOL.1

LIVROS ÉRICA EDITORA LTDA.

MICROPROCESSADOR

Z80

HARDWARE

VOL.1

CIP-Brasil. Catalogação-na-Publicação
Câmara Brasileira do Livro, SP

Cipriano, Luís Benedito, 1957-
C523m Microprocessadores Z-80 / Luiz Benedito Cypriano, Paulo Roberto Cardinali. -- São Paulo : Érica, 1983.

Bibliografia.
Conteúdo: v.1. Hardware.

1. Microprocessadores I. Cardinali, Paulo Roberto, 1959- II. Título.

17. CDD-621.381958
83-1682 18. -621.38195835

Índices para catálogo sistemático:

1. Microprocessadores Z-80 : Engenharia eletrônica
621.381958 (17.) 621.38195835 (18.)

Engº Luiz Benedito Cypriano
Engº Paulo Roberto Cardinali

MICROPROCESSADOR Z80

HARDWARE

VOL.1

1983

LIVROS ÉRICA EDITORA LTDA.

Desenhos:
Linda Maria da Silva Sanches

Composição:
Roseli Barbaresco de Oliveira

Revisão:
Ayako Tabata
Rosamaria G. F. Mello

LIVROS ÉRICA EDITORA LTDA.

Rua Jarinu, 594 - Tatuapé - São Paulo
Fone: 294-8686 - C.G.C. 50.268.838/0001-39
Caixa Postal 15.617

À

nossos Pais.

SUMARIO

PREFÁCIO

É visível o crescente desenvolvimento das atividades relacionadas com microcomputação nos dias atuais. Frente a isto, deparamo-nos com um número considerável de problemas que, por vezes, dificultam o acesso a este campo.

Enquanto estudantes, vê-se a escassa bibliografia existente na língua portuguêsa, a maioria das fontes apresentam-se em Inglês ou Castelhano. Isto faz com que a aprendizagem assuma um carácter árduo sendo que na realidade, é a falta de material didático o que dificulta o processo ensino-aprendizagem.

Aqueles que se iniciam neste ramo, sem no entanto ter uma formação condizente para tal, outras dificuldades lhes são impostas. O material a que têm acesso é, na maioria das vezes, caracterizado por uma linguagem altamente técnica. Para suprir este tipo de necessidade, faz-se mister a existência de fontes que além de apresentar-se com uma linguagem fácil e acessível, também encerrem um conteúdo eficiente, o bastante para capacitar o leitor a exercer funções ativas.

Foi pensando nestes tipos de problemas que comumente são enfrentados e, no crescente avanço tecnológico da micro-computação que surgiu-nos o interesse de redigir este livro.

Esperamos que com o presente livro, consigamos fornecer dados claros e eficientes, para propiciar condições àqueles que desejam, por um motivo ou por outro, se realizar neste campo tão complexo e ao mesmo tempo envolvente da micro-computação.

OS AUTORES

CAPÍTULO 1 – INTRODUÇÃO

Na época em que os primeiros computadores surgiram eram incômodos devido ao seu grande porte e grande quantidade de memória que exigiam. Apresentavam baixa velocidade de processamento.

Com o avanço da tecnologia nos países industrializados, tais inconvenientes foram gradativamente eliminados, graças à compactação de milhares de componentes como resistências, diodos e transistores em pastilhas de não mais de "1" centimetro quadrado.

Com este avanço pudemos chegar a sistemas de computação como o da figura (1.1), em um espaço milhares de vezes menor do que o primeiro sistema que surgiu.

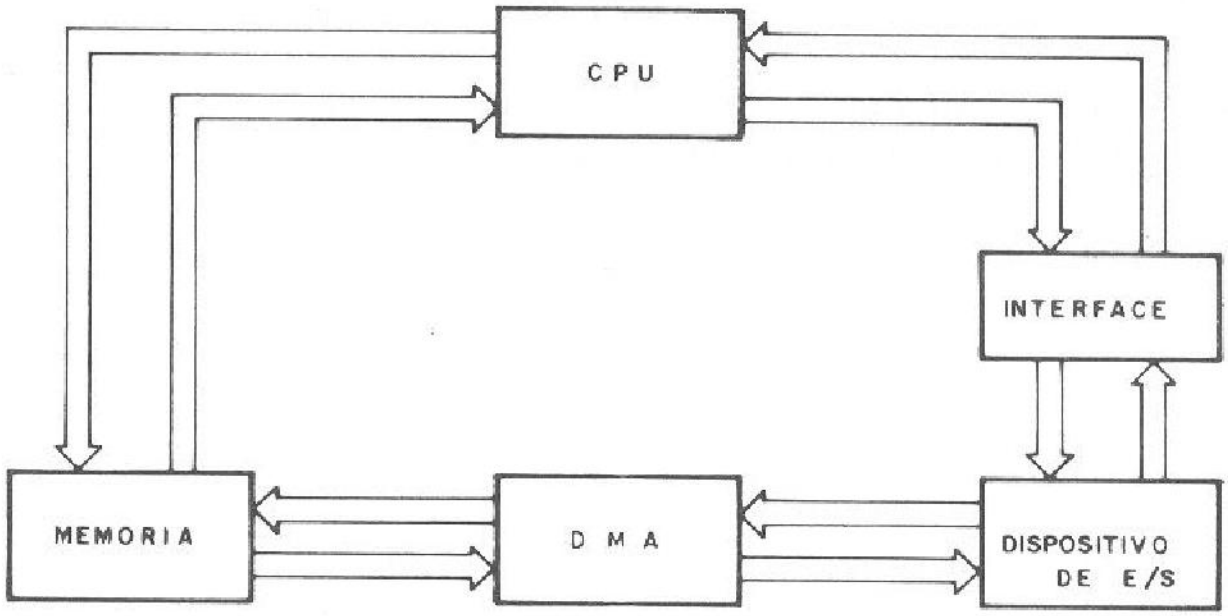


Fig. 1.1 - Sistema Básico de Computação.

Apenas alguns anos atrás, a INTEL CORPORATION, lançou no mercado o Microprocessador "8080", altamente compacto, apresentando grande flexibilidade em sistemas de computação, com um poderoso "set" de instruções. Tal microprocessador foi absorvido por todos os ramos de atividades.

Despertou-se então, principalmente no setor industrial, um grande interesse por estes pequenos "computadores".

Um grupo de engenheiros da INTEL CORPORATION, resolveu então, se desligar e partir para o projeto de um microprocessador mais avançado.

Assim surgiu o Z-80, com instruções mais poderosas do que as do microprocessador "8080", também resolvendo um grande problema que é a necessidade de circuitos auxiliares a CPU.

1.1 – CPU

É a Unidade Central de Processamento de Sistema. É ela que controla todo o fluxo de informações além de interpretar e executar as instruções do programa e monitorar todos os dispositivos do sistema.

Todo este controle é feito através de sinais gerados por ela a partir da decodificação das instruções do programa contidas na memória.

Embora a Unidade Central de Processamento seja o " cerebro"do sistema ela é incapaz de executar qualquer função se a ela não estiver associada uma memória.

1.2 – MEMÓRIA

É uma dispositivo capaz de armazenar informações que podem ser apenas dados ou programas. Sem este dispositivo para armazenamento de programas, a CPU fica inoperante.

1.3 – DISPOSITIVO E/S (I/0)

São dispositivos de entrada e saída também chamados de periféricos. Estes dispositivos podem ser memórias de grande capacidade, impressoras, controladores industriais, etc., que auxiliam a " CPU " no processamento do sistema.

1.4 – INTERFACE

A interface é um dispositivo que auxilia na transferência de informações entre a Unidade Central de Processamento e

dispositivos periféricos. Ela é usada principalmente em sistemas que possuem somente uma CPU e vários periféricos.

1.5 – DMA

" DMA " é o processo de comunicação direta entre dispositivos periféricos e memória de grande capacidade.

Este processo se deve ao fato da necessidade de se aumentar a velocidade de transferência de dados.

CAPÍTULO 2 – ALGORITMO

2.1 – INTRODUÇÃO

O principal intuito deste capítulo é o de esclarecer o mecanismo dos algoritmos.

Os algoritmos são dificilmente encontrados em livros e os que os contêm apresentam-se em uma linguagem de difícil compreensão.

Neste capítulo apresentamos, soma e subtração em BCD, multiplicação e divisão de números positivos e negativos.

No transcorrer deste capítulo será apresentada uma série de exercícios para a fixação dos algoritmos apresentados.

2.2 – ARITMÉTICA EM COMPLEMENTO UM (C_1)

Neste item será apresentada a soma e subtração em complemento um (C_1).

2.2.1 Soma em complemento um

A soma de dois números binários, faz-se somando BIT a BIT segundo a tabela a seguir.

O "vai um", representa o estouro de uma casa, estouro esse que deve ser somado a casa seguinte.

X	Y	X + Y	VAI UM
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tabela de Soma

Na seqüência serão apresentados alguns exemplos onde poderemos acompanhar o mecanismo da soma em binário.

Exemplos:

a.) A = 0100 B = 0011 C = A + B

```
A → 0100 → (4)
B → 0011 + → (3)
C → 0111 → (7)
```

b.) A = 0101 B = 0001 C = A + B

```
  1 → vai um
A → 0101 → (5)
B → 0001 + → (1)
C → 0110 → (6)
```

c.) A = 01011 B = 01101 C = A + B

```
 1111 → vai um
A → 01011 → (11)
B → 01101 + → (13)
    11000 → (24)
```

2.2.2 Subtração em complemento um

A subtração de números inteiros pode ser obtida através da seguinte seqüência:

- Devemos efetuar o complemento "um" do número negativo. Para tanto basta inverter cada bit do número, ou seja, o que é "um" passa a ser "zero" e o que é "zero" passa a ser "um".

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

Efetuamos então uma "soma" entre o primeiro operando e o segundo em complemento "um".

Exemplo:

$$C = A - B \Rightarrow C = A + \overline{B}$$

- O resultado da operação deve ser também complementado para obtermos o módulo do número se o operando "B" for maior que o operando "A".

Todo este processo será melhor entendido através dos exemplos que serão dados e pelo fluxograma da fig. 2.1

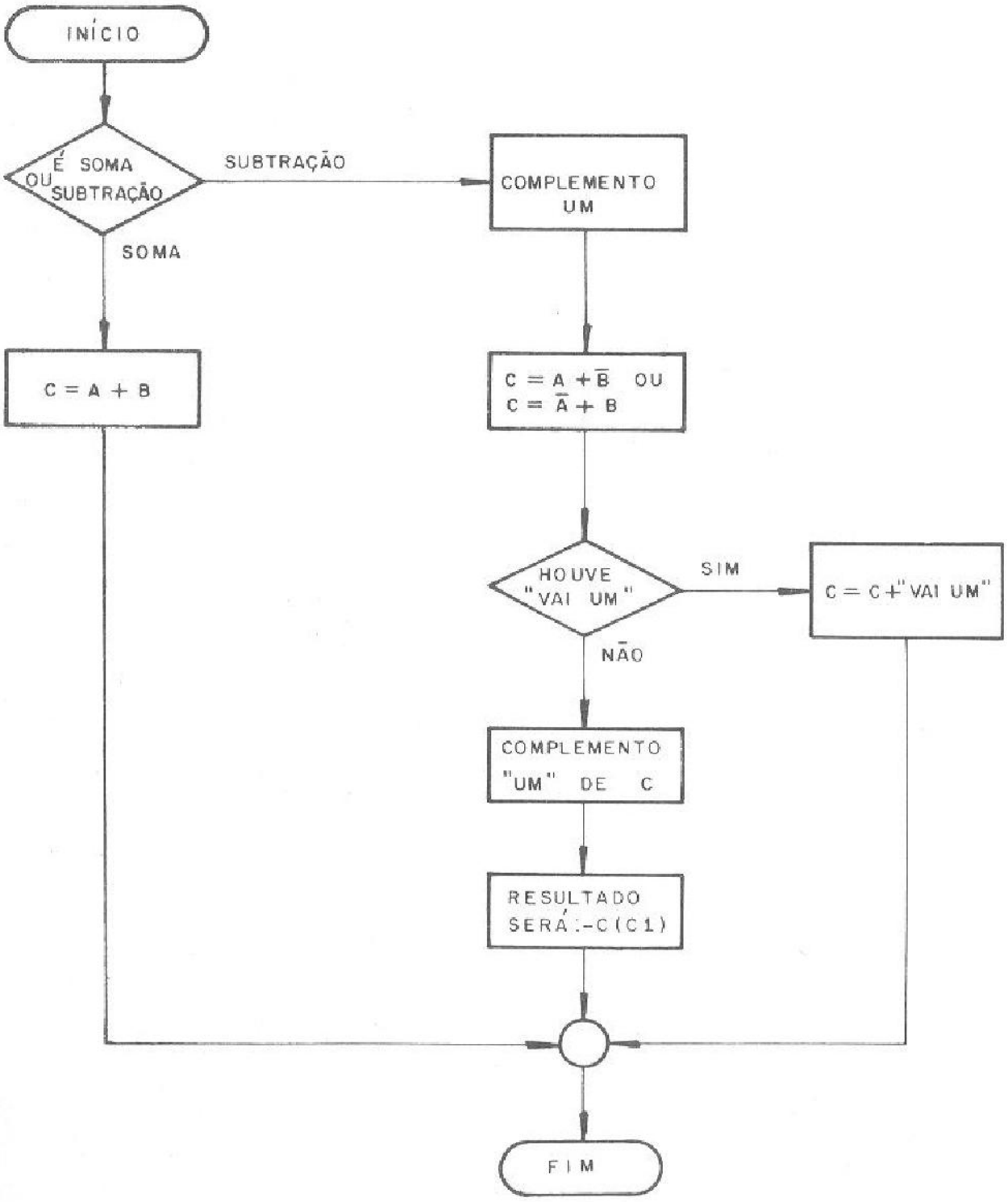


Fig. 2.1 - Fluxo das operações em complemento um.

Exemplos:

a.) A = 0101 $\quad \overline{A} = 1010$

b.) B = 1100 $\quad \overline{B} = 0011$

c.) C = 0001 $\quad \overline{C} = 1110$

d.) D = 01010 $\quad \overline{D} = 10101$

e.) A = 01010 (10_{10}) $\quad$ B = 01100 (12_{10}) $\quad$ C = A - B

B = 01100 => $\overline{B}$ = 10011

^{1} → vai um

A → 01010 → (10)

$\overline{B}$ → $\underline{10011}^{+}$ → (-12)

C → 11101 → (-2)

C = 11101 => $\overline{C}$ = 00010 → (+2) => Resultado = -C = -2

f.) A = 01011 $\quad$ B = 00011 $\quad$ C = A - B

B = 00011 => $\overline{B}$ = 11100

$_{1}$ → vai um

A → 01011 → (11)

$\overline{B}$ → $\underline{11100}^{+}$ → (-3)

00111_{+}

vai um ← (1) → 1

C → 01000 → (8)

g.) A = 01111 $\quad$ B = 00101 $\quad$ C = A - B

B = 00101 => $\overline{B}$ = 11010

^{111} → vai um

A → 01111_{+} → (15)

$\overline{B}$ → $\underline{11010}$ → (-5)

^{1} → vai um

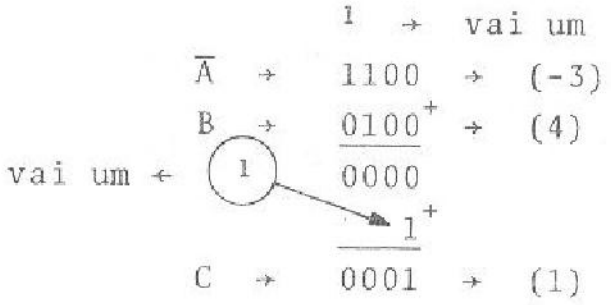
vai um ← (1) $\quad$ 01001

1^{+}

C → 01010 → (10)

h.) A = 0011 B = 0100 C = -A + B

A = 0011 => $\overline{A}$ = 1100



2.3 – ARITMÉTICA EM COMPLEMENTO DOIS (C_2)

Do processo apresentado em complemento "um" (C_1), o complemento dois (C_2) apresenta diferença apenas na operação de subtração.

O complemento Dois (C_2) de um número binário, é o complemento "um" do número acrescido de uma unidade, ou em outras palavras, inverte-se Bit a Bit e adiciona-se uma unidade.

Exemplos:

a.) B = 011 → número
$\overline{B}$ = 100 → número em complemento um
1 +
B(C_2) = 101 → número em complemento dois

b.) A = 001000 → número
111 → "vai um"
$\overline{A}$ = 110111 → número em complemento um
1 +
A(C_2) = 111000 → número em complemento dois

c.) A = 01111 → número
$\overline{A}$ = 10000 → número em complemento um
1 +
A(C_2) = 10001 → número em complemento dois

2.3.1 Subtração em complemento dois

Na subtração em complemento 2 quando houver "vai um" do algarismo mais significativo, este é abandonado.

Quando não houver "vai um", o resultado deverá ser complementado para se achar o módulo o qual deverá ser acompanhado do sinal negativo.

Na fig. 2.2, temos o fluxo das operações em complemento dois.

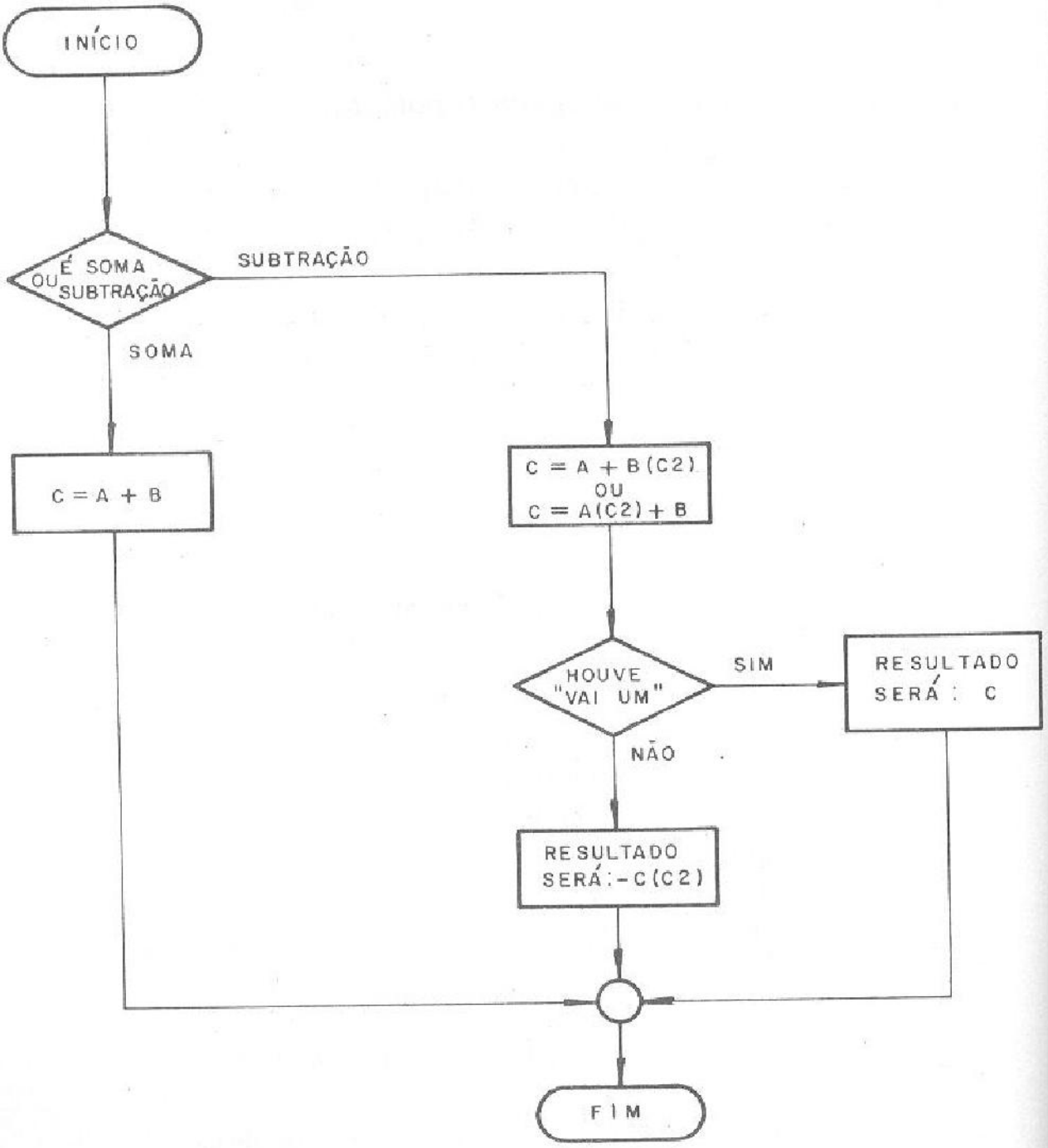


Fig. 2.2 - Fluxo de operações em complemento dois.

Exemplos:

a.) A = (+01000) B = (+00111) C = A - B

"vai um" da última casa → (1) ^{1} ← vai um

A → 01000 → (8)

$B(C_2)$ → 11001^{+} → (-7)

C → 00001 → (1)

b.) A = (+01010) B = (01110) C = A - B

^{1} ← vai um

A → 01010 → (10)

$B(C_2)$ → 10010^{+} → (-14)

C → 11100

Se não houver "vai um" da última casa, portanto o resultado será o complemento dois do obtido.

$R = - C(C_2)$

C → 11100 - resultado obtido

$\overline{C}$ → 00011 - complemento um

1^{+}

$C(C_2)$ → 00100 - complemento dois

∴ R = -(00100) = - 4 - Resultado final

2.4 – ARITMÉTICA COM SOMA E SUBTRAÇÃO EFETIVA

Neste item apresentaremos uma técnica de soma e subtração efetiva. O princípio desta técnica é o de analisar o sinal dos operandos antes de executar as operações. Deste momento em diante apenas as amplitudes importam.

ADIÇÃO EFETIVA	SUBTRAÇÃO EFETIVA
a.) A + B	e.) A - B
b.) (-A) + (-B)	f.) (-A) - (-B)
c.) A - (-B)	g.) A - (-B)
d.) (-A) - B	h.) (-A) - B

Na fig. 2.3 temos o fluxograma completo de soma e subtração efetiva.

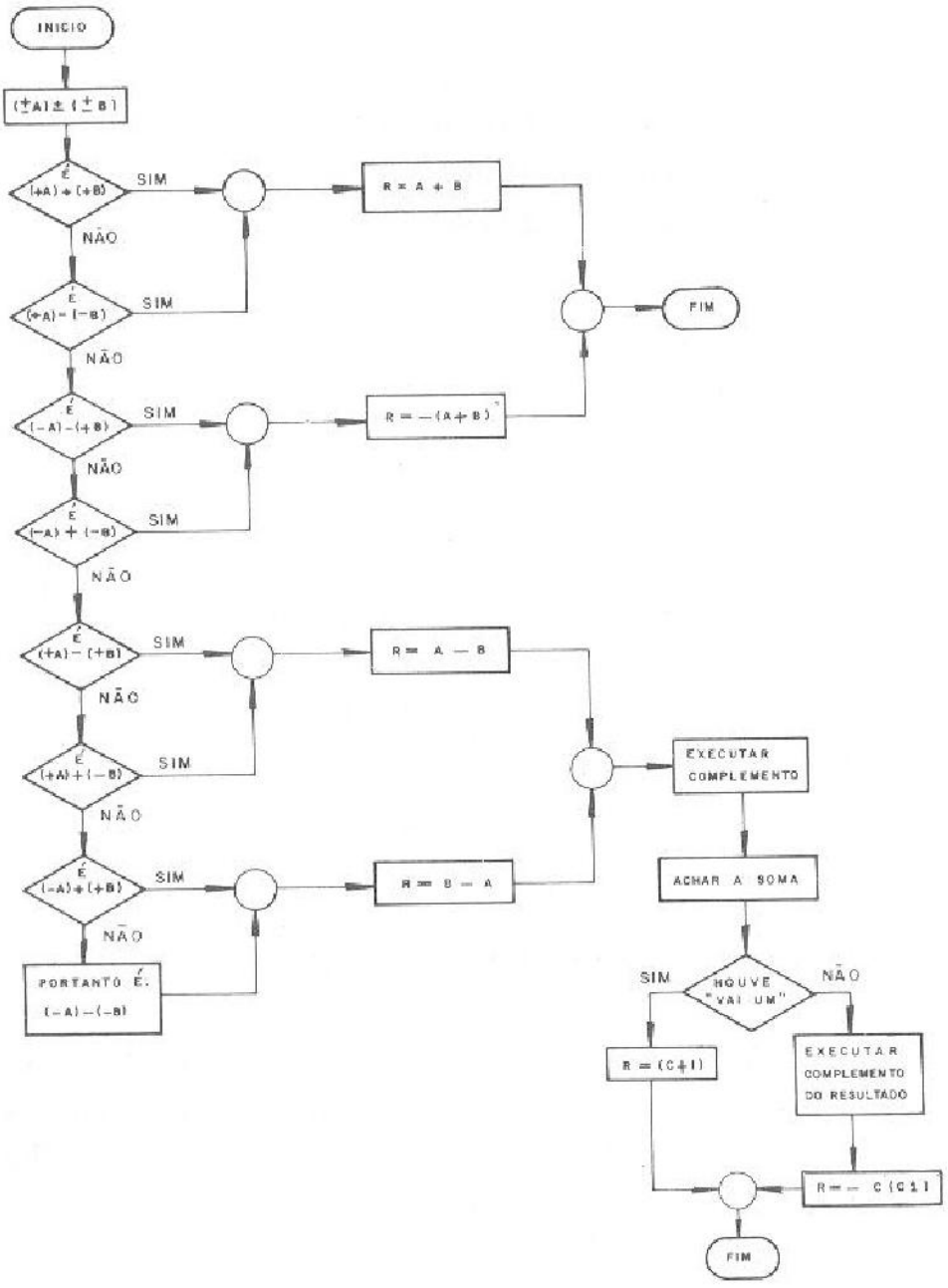


Fig. 2.3 - Fluxograma de soma e subtração efetiva.

Exemplos:

a.) A = -(0101) B = -(0110)

C = A + B => C = -A + (-B) => C = -A - B

∴ C = -(A + B)

1 ← vai um

A → 0101 → (5)

B → 0110 + → (6)

C → 1011 → (11)

∴ C = (-1011) →(-11)

b.) A = (0100) B = (-0011)

C = (A - B) => C = A - (-B)

∴ C = (A + B)

A → (0100) + → (4)

B → (0011) → (3)

∴ C → (0111) → (7)

c.) A = (0100) B = (-0001)

C = (A + B) => C = A + (-B)

C = (A - B)

1 ← vai um

A → (0100) → (4)

$\overline{B}$ → (1110) + → (-1) → complemento um

vai um ← (1) 0010

1 +

∴ C → 0011 → (3)

d.) A = (-0110) B = (-0001)

C = (A - B) => C = (-A) - (-B) => C = -A + B

∴ C = B - A

$$
\begin{array}{lcll}
 & & 1 \leftarrow \text{vai um} & \\
B & \rightarrow & 0001 & \rightarrow (1) \\
\overline{A} & \rightarrow & \underline{1001}^{+} & \rightarrow (-6) \rightarrow \text{complemento um} \\
C & \rightarrow & 1010 & \rightarrow (-5)
\end{array}
$$

$$\therefore \quad R = -\overline{C} = (-0101) \rightarrow (-5)$$

2.5 – ARITMÉTICA EM BCD

Este tipo de aritmética é usada em programação de computador onde as operações são feitas em BCD.

As operações apresentadas aqui serão efetuadas em aritmética de soma e subtração efetiva.

2.5.1 Soma em BCD

A soma em BCD é dividida em três casos.

a.) Dígito legal: não há "vai um" e o resultado está entre 0 e 9.

b.) Dígito ilegal: não há "vai um", mas o resultado está entre 10 e 15, neste caso precisa-se corrigi-lo e propagar o resultado para a casa seguinte.

c.) Dígito ilegal: há "vai um" e neste caso o resultado está entre 16 e 20, portanto também é preciso fazer a correção mas, não é necessário propagar o "vai um".

A correção acima mencionada é feita somando-se seis $(0110)_2$ a cada vez que houver necessidade.

Na fig. 2.4 temos o fluxo da soma em BCD.

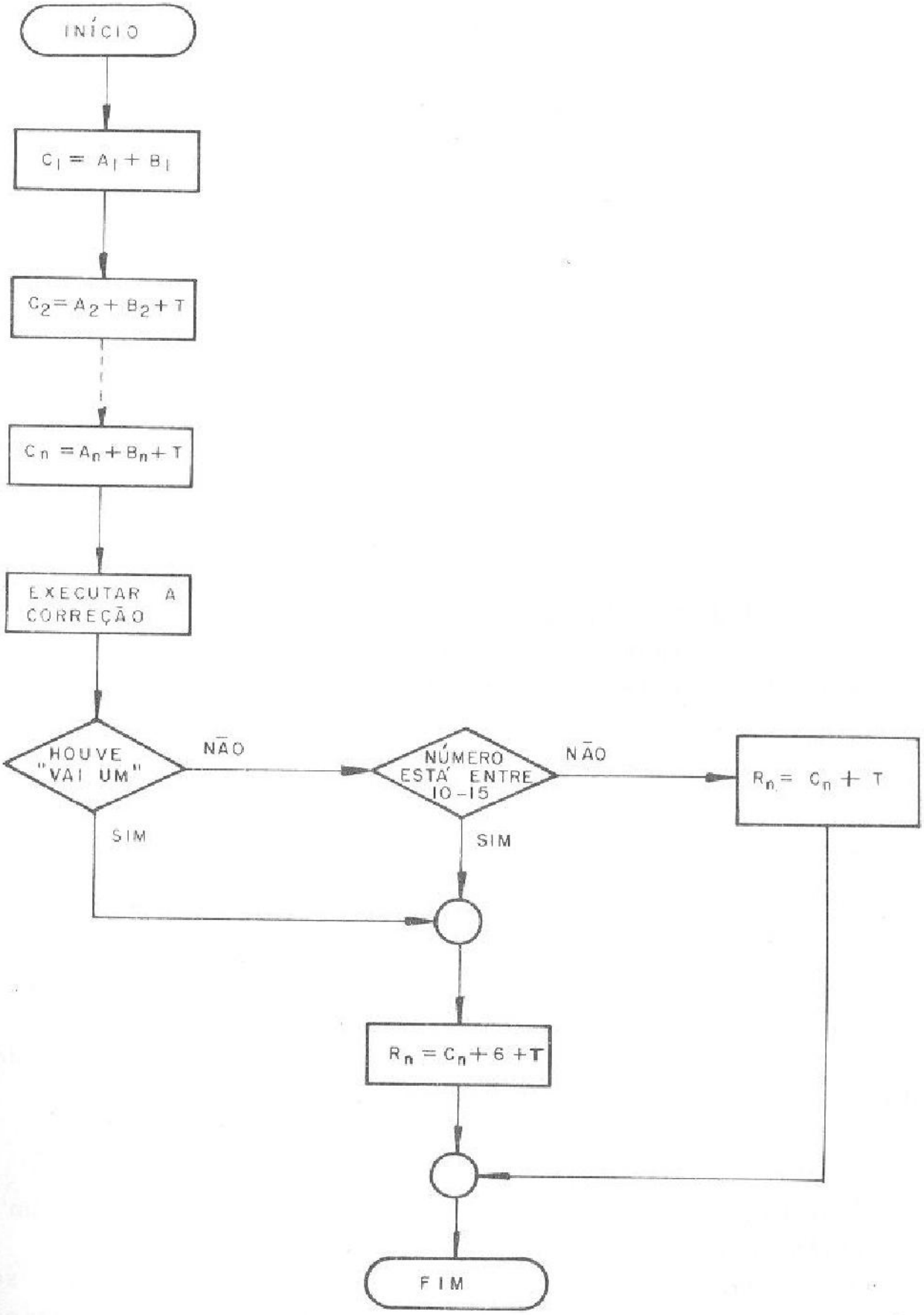


Fig. 2.4 - Fluxograma da soma de dois números em BCD.

Exemplos:

a.) A = 743 B = 974 C = A + B

	^{1} →vai um	^{111} →vai um.	^{1} →vai um	
A →	0000	0111	0100	0011 → (743)
B →	0000	1001	0111	0100^{+} → (974)
		^{1}→ vai um	^{11}→ vai um	
	0001	0000	1011	0111_{+}
	0000	0110	0110	0000 → correção
C =	0001	0111	0001	0111 → (1717)

b.) A = 958 B = 764 C = A + B

	^{1}→ vai um	^{111}→ vai um	^{1}→ vai um	
A →	0000	1001	0101	1000 → (958)
B →	0000	0111	0110	0100^{+} → (764)
		^{1}→ vai um	^{1111}→vai um	^{1} →vai um
	0001	0000	1011	1100
	0000	0110	0110	0110^{+} → correção
C →	0001	0111	0010	0010 → (1722)

2.5.2 Soma Rápida

Neste item apresentaremos a soma em BCD chamada de rápida. Este método funciona da seguinte maneira.

a.) Soma-se 6 a todos os dígitos da parcela A.
b.) A seguir somam-se as parcelas levando em conta o "vai um" normalmente.
c.) Nos dígitos em que não houver " vai um " para a casa seguinte, soma-se dez, sendo que os "vai um" desta última operação serão desprezados.

Na fig. 2.5 temos o fluxo da soma rápida em BCD.

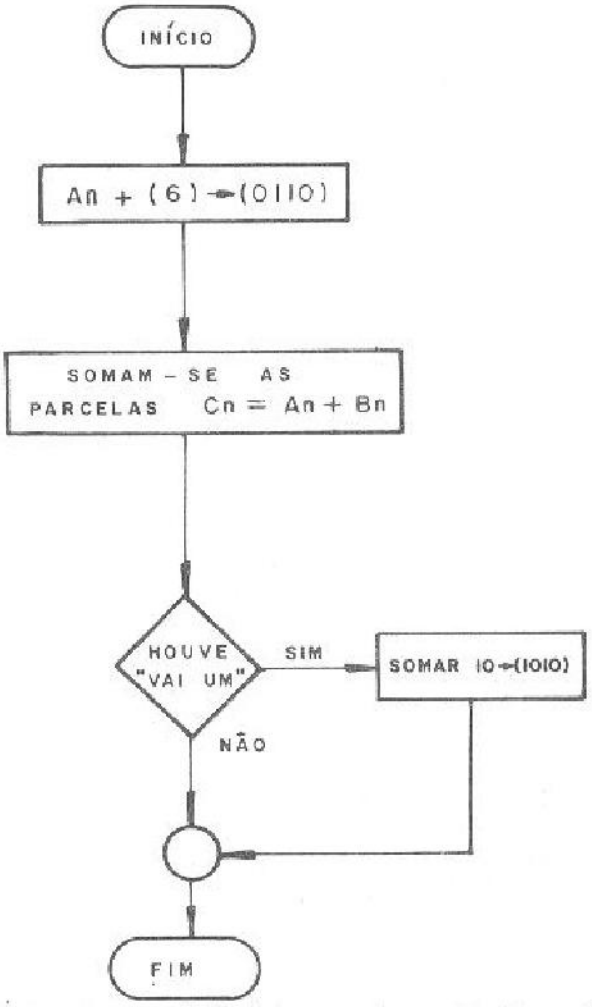


Fig. 2.5 Fluxograma da Soma Rápida entre Dois números em BCD.

Exemplos:

a.) A = 479 B = 835 C = A + B

		1 ← vai um	11 ← vai um		
A →	0000	0100	0111	1001	→ (479)
	0110	0110	0110	0110 +	→ soma 6
	1 ← vai um	1 ← vai um	1111 ← vai um	111 ← vai um	
	0110	1010	1101	1111	
B →	0000	1000	0011	0101 +	→ (835)
(1)	11 ← "vai um"				
	0111	0011	0001	0100	
	1010			+	→ soma 10
C =	0001	0011	0001	0100	→ (1314)

b.) A = 324 B = 732 C = A + B

		11← vai um	11← vai um	1 ←	vai um
A →	0000	0011	0010	0100	→ (324)
	0110	0110	0110	0110^{+}	→ soma 6
	1 ←vai um	111 ←vai um		1 ←	vai um
	0110	1001	1000	1010	
B →	0000	0111	0011	0010^{+}	→ (732)
	⊘1 11 ← vai um	⊘1 ←vai um	⊘1 ←vai um		
	0111	0000	1011	1100	
	1010	0000	1010	1010^{+}	→ soma 10
C =	0001	0000	0101	0110	→ (1056)

A vantagem deste método é que ao se corrigir os dígitos não é necessário transportar o"vai um",ele é simplesmente abandonado.

2.5.3 Subtração em BCD

A subtração é dividida em 3 partes:

a.) Complementar cada dígito do subtraendo.
b.) Após o complemento fazer a soma de mais 1. ($A + \overline{B} + 1$)
c.) Nos dígitos em que não houver "vai um" deve-se somar 10, a fim de corrigi-los.

Exemplos:

a.) A = 719 B = 494 C = A - B

B →	0100	1001	0100	→ (494)
$\overline{B}$ →	1011	0110	1011	→ complemento

	(1) 111 ← vai um	1111 ← vai um	11	← vai um
A →	0111	0001	1001	→ (719)
$\overline{B}$ →	1011	0110	1011	→ complemento
	0000	0000	0001 +	→ soma de 1
	(1) ← vai um			
	0010	1000	0101	
	0000	1010	0000 +	→ soma 10
C =	0010	0010	0101	→ (225)

b.) A = 735 B = 696 C = A - B

B →	0110	1001	0110	→ (696)
$\overline{B}$ →	1001	0110	1001	→ complemento
	(1) 111 ← vai um	11 ← vai um	1	← vai um
A →	0111	0011	0101	→ (735)
$\overline{B}$ →	1001	0110	1001	→ complemento
	0000	0000	0001 +	→ soma de 1
	(1) ← vai um	(1) ← vai um	1	← vai um
	0000	1001	1111	
		1010	1010 +	→ soma 10
C =	0000	0011	1001	→ (039)

2.6 – ARITMÉTICA COM MULTIPLICAÇÃO E DIVISÃO

Os algoritmos desenvolvidos neste item usarão o processo de soma e deslocamento. Existe também outro tipo de se efetuar multiplicação e divisão, que é o de soma sucessiva para multiplicação, e de subtração sucessiva para divisão. Este segundo método é muito lento; para se ter uma idéia, se os computadores aplicassem este método ao processar uma simples contabilidade de uma firma, demorariam horas.

2.6.1 Multiplicação

A multiplicação se divide em 5 partes:

a.) Converter R_1 e R_2 para binário.

b.) Colocar o multiplicador e o multiplicando um do lado do outro (R_1 e R_2).

c.) Abaixo dos dois operandos colocamos zeros.

d.) Analisa-se do Bit menos significativo ao mais significativo de R_2.

f.) Se o Bit analisado for "zero" apenas se desloca R_1 com o "vai um"; e se o Bit for "um" soma-se o último resultado da $1^{\underline{a}}$ coluna com R_1 e desloca.

Exemplos:

a.) $A = (0011)_2 \rightarrow (3)_{10}$ $\quad B = (0100)_2 \rightarrow (4)_{10}$ $\quad C = A \times B$

$1^{\underline{a}}$ COLUNA $\quad$ $2^{\underline{a}}$ COLUNA

R_1 →		0011	R_2 → dcba 0100	
R =		0000	0000	→ zeros iniciais
R =	0 →	0000	0000	→ desloca
R =	0 →	0000	0000	→ desloca
		0011 +	0000	→ soma : R_1+R
R =	0 →	0001	1000	→ desloca
R_F =		0000	1100	→ desloca

$R_F = (00001100) = (12)_{10}$

Bit a = 0 Desloca o resultado de uma casa.
Bit b = 0 Desloca o resultado de uma casa.
Bit c = 1 Soma (R + R_1) e desloca uma casa.
Bit d = 0 Desloca o resultado de uma casa.

b.) $A = (1001)_2 = (9)_{10}$ $B = (0111)_2 = (7)_{10}$ $C = A \times B$

1ª COLUNA 2ª COLUNA

$R_1 \rightarrow 1001$ $R_2 \rightarrow 0111$

R =		0000	0000	→ zeros iniciais
R =	0 →	1001 + → 0100	0000 → 1000	→ soma : R_1 + R → desloca
R =	0 →	1101 + → 0110	1000 → 1100	→ soma : R_1 + R → desloca
R =	0 →	1111 + → 0111	1100 → 1110	→ soma : R_1 + R → desloca
R_F =	0 →	→ 0011	→ 1111	→ desloca

$R_F = (00111111)_2 = (63)_{10}$

c.) $A = (1110)_2 = (14)_{10}$ $B = (0011)_2 = (3)_{10}$ $C = A \times B$

1ª COLUNA 2ª COLUNA

$R_1 \rightarrow 1110$ $R_2 \rightarrow 0011$

R=		0000 +	0000	→ zeros iniciais
R=	0 →	1110 → 0111 +	0000 → 0000	→ soma: R_1 + R → desloca
R=	vai um (1) →	0101 → 1010	0000 → 1000	→ soma : R_1 + R → desloca
R=	0 →	→ 0101	→ 0100	→ desloca
R=	0 →	→ 0010	→ 1010	→ desloca

$R_F = (00101010)_2 = (42)_{10}$

2.6.2 Divisão

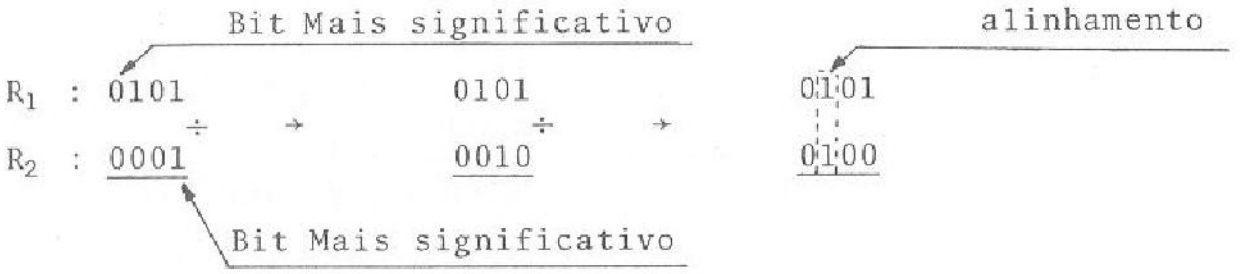
A divisão consiste basicamente de subtrações e deslocamentos sucessivos. Tais subtrações e deslocamentos estão condicionados aos resultados de cada fase da operação, ou seja, se o resultado foi negativo ou positivo.

Podemos decompor a divisão em 3 partes, e para que tudo fique bem claro, as descreveremos com o auxílio de um exemplo.

Suponhamos a divisão entre o número 5, que chamaremos de R_1, e o número 1, que chamaremos de R_2.

$$R_1 = (5)_{10} = (0101)_2 \qquad R_2 = (1)_{10} = (0001)_2$$

a.) A primeira parte será o ajuste dos operandos. Para isso colocamos um abaixo do outro, e logo após deslocamos R_2 para a esquerda, casa por casa, até que seu bit mais significativo igual a "um" fique alinhado com o bit mais significativo igual a "um" de R_1.



Note que foram necessários dois deslocamentos de R_2 até atingirmos o alinhamento, e que os "zeros" à esquerda do bit mais significativo igual a "um" de R_2 foram desprezados e que à direita foram introduzidos "zeros" a cada deslocada.

b.) A segunda parte consiste da subtração entre R_1 e R_2. O resultado, chamado aqui por R, será iniciado com "zero".

R_1 : 0101 R: 0000
R_2 : 0100
R_p : 0001

c.) A terceira e última parte refere-se a análise do resultado parcial -R_p e do seu sinal através da ocorrência ou não do "vai um". Na divisão, o "vai um" representa o "empréstimo" da próxima casa, quando R_1 é menor que R_2.

Se o resultado parcial for positivo, ou seja, "vai um" igual a "zero", devemos em primeiro lugar deslocar o resultado - R - de uma casa para a esquerda entrando com "um"; em segundo lugar deslocar R_2 de uma casa para a direita; e por último efetuar nova subtração entre R_2 e o último R_p.

Contudo, se o resultado parcial for negativo, ou seja, "vai um" igual a "um", devemos em primeiro lugar deslocar o resultado - R - de uma casa para a esquerda entrando com "zero"; em segundo lugar deslocar R_2 de uma casa para a direita; e por último efetuar nova subtração só que agora entre R_2 e o penúltimo R_p.

Se esta for a primeira operação, a subtração será entre R_2 deslocado e R_1 (vide exemplo A).

Em nosso exemplo, o primeiro resultado parcial foi positivo, portanto obtemos $R = (1)_{10}$:

R_1: 0101 R: 0000

R_2: 0100

R_p: 0001 → R = ←0001

Note que devemos seguir com o processo até obtermos R_2 em sua forma original, ou seja, $R_2 = (0001)_2$, quando então teremos $R = R_F$ (resultado final):

Rp: 0001 → último resultado parcial. R = 0001

0010 → R_2 deslocado.

"vai um" (1)

Rp: 1111 → segundo resultado parcial. R = ←0010

∴ 0001 → penúltimo resultado parcial. R = 0010

0001 → R_2 com última deslocada.

Rp = 0000 → último resultado parcial. R = 0101

∴ $R_F = (0101)_2 = (5)_{10}$

A seguir, daremos alguns exemplos para melhor fixação dos conceitos e do processo, mencionados.

Exemplos:

a.) $A = (1001)_2 = (9)_{10}$ $B = (0011)_2 = (3)_{10}$ $C = A \div B$

$R_1 = 1001$ → $R_1 = 1001$

$R_2 = 0011$ $R_2 = 1100$ → R_2 alinhado

$R_1 = 1001$ $R = 0000$

"vai um" (1) $\underline{R_2 = 1100}$

$Rp = 1101$ → 1º Resultado parcial (negativo)

$\therefore R = \overleftarrow{0000}$

1001 → Penúltimo resultado parcial (no caso: R_1)

0110 → R_2 deslocado

$Rp = 0011$ → 2º Resultado parcial (positivo)

$\therefore R = \overleftarrow{0001}$

0011 → último resultado parcial

0011 → R_2 deslocado

$Rp = 0000$ → 3º Resultado parcial (positivo)

$\therefore R_F = R = (0011)_2 = (3)_{10}$

b.) $A = (1010)_2 = (10)_{10}$ $B = (0010)_2 = (2)_{10}$ $C = A \div B$

$R_1 = 1010$ → $R_1 = 1010$

$R_2 = 0010$ → $R_2 = 1000$ → R_2 alinhado

$R_1 = 1010$ — $\quad R = 0000$

$R_2 = 1000$

$Rp = 0010$ → 1º Resultado parcial (positivo)

$\therefore R = \overleftarrow{0001}$

0010 → último resultado parcial

0100 → R_2 deslocado

"vai um" (1)

$Rp = 1110$ → 2º Resultado parcial (negativo)

$\therefore R = \overleftarrow{0010}$

0010 → penúltimo resultado parcial

0010 → R_2 deslocado

$Rp = 0000$ → 3º Resultado parcial (positivo)

$\therefore R = \overleftarrow{0101}$

$\therefore \quad R_F = R = (0101)_2 = (5)_{10}$

CAPÍTULO 3 – Z-80 CPU

UNIDADE CENTRAL DE PROCESSAMENTO

3.1 – INTRODUÇÃO

Nos últimos anos, o avanço da tecnologia tem propiciado o aparecimento de circuitos integrados capazes de executar inúmeras funções em tempos cada vez menores.

O maior impacto da tecnologia LSI nos últimos anos, tem sido o MOS- LSI com o qual é possível a construção de poderosos e completos computadores em uma única pastilha.

O conjunto de componentes Z-80, é um grande avanço dos microcomputadores. Estes componentes podem ser combinados com memórias a semicondutor gerando grandes sistemas com as mais variadas aplicações em todos os setores da vida cotidiana.

Por exemplo: juntando um Z-80 CPU com alguns circuitos auxiliares; memórias, que podem ter diferentes capacidades; e dispositivos de entrada e saída (I/O), teremos um microcomputador com capacidade comparável a de um computador convencional. Esta gama de características computacionais permite que um usuário tenha um módulo capaz de satisfazer a uma grande variedade de aplicações.

O domínio de componentes MOS-LSI no mercado atual se deve ao baixo custo destes componentes que vêm substituindo a lógica TTL em aplicações, como computadores pessoais, terminais inteligentes, controladores de terminais, controles industriais, etc. Em suma, em todos os ramos onde a eletrônica é utilizada, os microcomputadores têm encontrado o seu caminho.

Microcomputadores de tecnologia MOS-LSI estão estáveis no mercado e novos produtos neles baseados estão sendo desenvolvidos com muita rapidez. A conveniência do Z-80 é evidente nos seguintes fatores:

- É totalmente compatível com o Software do 8080, fartamente encontrado no mercado.
- Trabalha a uma freqüência de 4 MHz, possuindo assim uma grande velocidade de operação.

- O conjunto de componentes Z-80 é superior tanto em Hardware como em Software em relação ao 8080 e 8085.

É muito fácil se construir um microprocessador utilizando-se o Z-80. Tal sistema consiste de três partes:

- Unidade Central de Processamento (CPU)
- Memória
- Circuito de Interface para dispositivos periféricos

A CPU é a parte central do sistema, tendo por funções obter instruções da memória, gerar sinais de controle para realizar as operações desejadas, controlando o fluxo de informações. Contudo, uma CPU sem um conjunto de memórias não executa nada. Essas memórias são utilizadas para guardar instruções e muitas vezes, dados que serão processados. Tais instruções são as sequências básicas que o microcomputador deve seguir para executar certas funções e por sua vez, um conjunto de instruções devidamente ordenadas forma um programa.

Portanto, o usuário possuindo um sistema completo, tem apenas a preocupação de elaborar programas específicos para suas aplicações, baseados no conjunto de instruções do Z-80.

3.2 – ARQUITETURA GERAL

A arquitetura da CPU do Z-80, em forma de diagrama de blocos, é mostrada na figura (3.1). Este diagrama mostra todos os principais elementos da CPU, com suas respectivas designações.

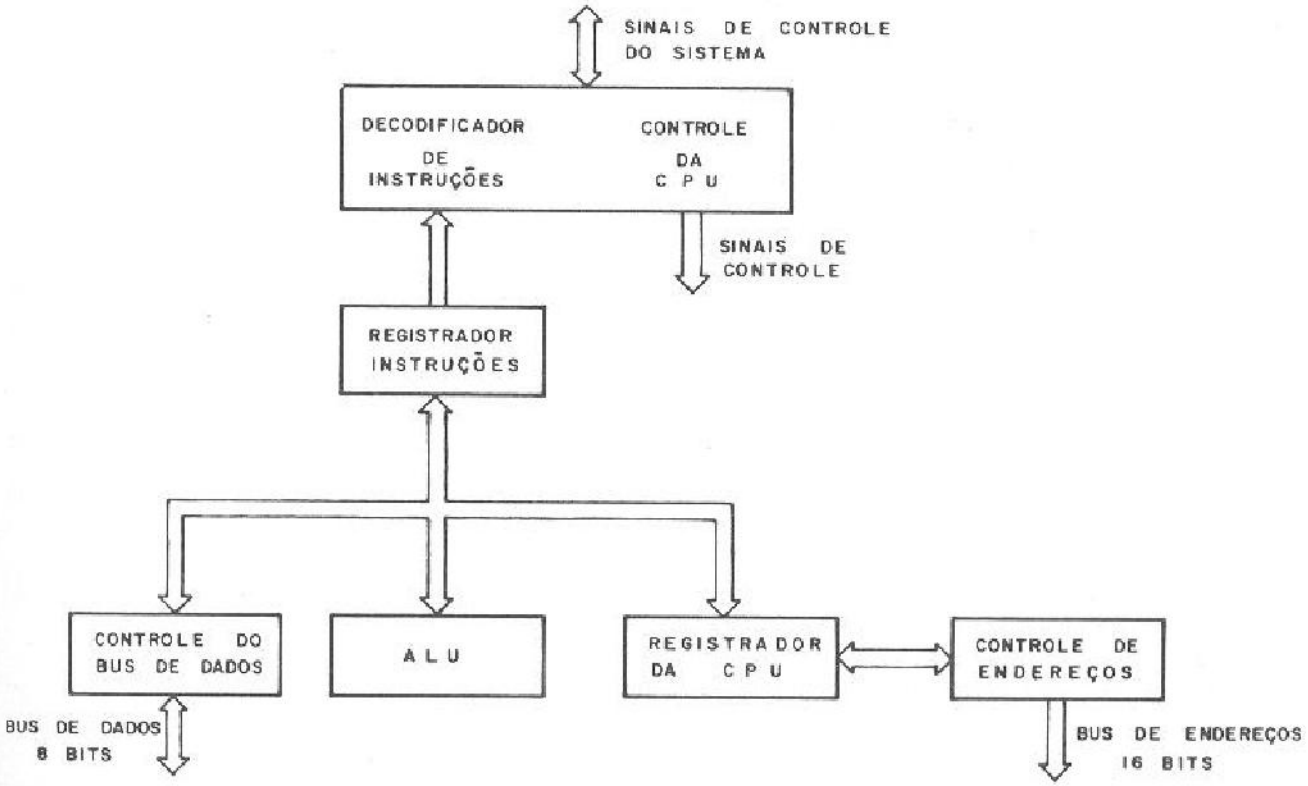


Fig. 3.1 - Diagrama em Blocos da CPU - Z-80.

3.3 – DESCRIÇÃO GERAL

3.3.1 Introdução

A Unidade Central de Processamento (CPU) tem disponíveis 208 bits de memória para escrita e leitura (R/W), que são de uso geral para o usuário. A figura (3.2) mostra como esta memória é constituída para formar 18 registradores de 8 bits e 4 de 16 bits. Todos os registradores do microprocessador Z-80 são implementados através de memórias estáticas RAM. Este grupo de registradores incluem dois grupos de seis registradores de uso geral, que podem ser utilizados individualmente como registradores de 8 bits ou em conjunto como registradores de 16 bits, também chamados de pares de registradores BC, DE e HL.

O Z-80 apresenta também dois grupos de acumuladores, registradores e flags, que são de grande utilidade para o usuário, pois se pode guardar o status dos registradores na própria CPU.

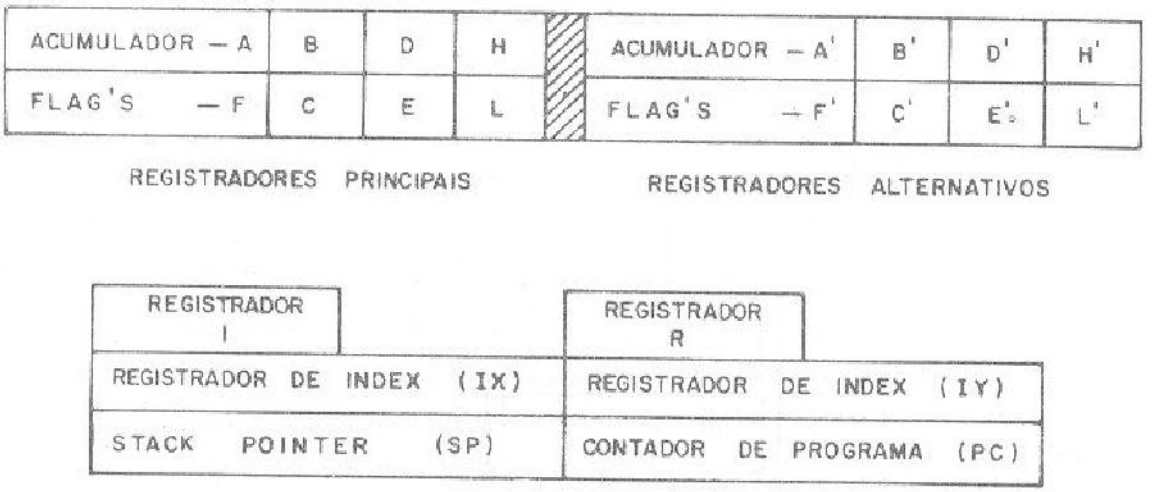


Fig. 3.2 – Estrutura dos Registradores Internos.

3.3.2 Unidade Lógica e Aritmética (ALU)

A configuração interna da Unidade Lógica Aritmética (ALU) tem acesso aos registradores e a via de dados externa, através de uma via de dados interna. As instruções aritméticas e lógicas de 8 bits da Unidade Central de Processamento (CPU), são executadas na Unidade de Lógica e Aritmética (ALU). Abaixo estão relacionadas as funções realizadas pela (ALU):

- Subtração
- Adição
- Setar Bit
- Testar Bit
- Incrementar
- Decrementar
- Comparação
- Lógica "E"
- Lógica "OU"
- Lógica "OU EXCLUSIVO"

- Deslocamentos à esquerda, à direita e rotações
- Resetar Bit

3.3.3 Registrador de Instruções e Controle da CPU

A cada ciclo de instrução; uma instrução é trazida da memória, pela via de dados, e transportada para o registrador de instruções aonde será interpretada e decodificada. A unidade de controle então, realiza sua função de fornecer os respectivos sinais de controle, necessários para ler ou escrever dados de um registrador, memórias, dispositivos de entrada e saída; controlar a (ALU) e fornecer todos os sinais externos de controle.

3.3.4 Registradores Gerais

3.3.4.1 Acumulador e Registradores de Flags

A Unidade Central de Processamento (CPU) contém dois acumuladores com 8 bits cada um e registradores associados a eles (registradores de Flags).

Todas as operações aritméticas e lógicas com 8 bits são executadas no acumulador e o resultado aí mantido, enquanto o registrador de flags indica condições específicas para operações com 8 bits ou 16 bits, exemplificando, se o resultado de uma operação for igual a zero a flag indicativa de "zero" será setado. O usuário seleciona o acumulador e o par de flags com os quais vai trabalhar, utilizando uma simples troca de instruções, o que torna fácil trabalhar com cada par.

3.3.4.2 Stack Pointer (SP)

O Stack Pointer, também chamado de ponteiro de pilha, é um registrador de 16 bits inicializado pelo usuário com o endereço inicial de um campo de memória RAM utilizado para guardar endereços em forma de pilha. A pilha, na memória externa, é organizada como um arquivo Lifo, (último dado a entrar é o primeiro dado a sair). O dado pode ser enviado para a pilha

de um par de registrador específico da CPU,ou levado da pilha para um específico registrador da CPU, através da execução das instruções (PUSH) ou (POP), que serão vistas no volume II deste livro.

O dado levado da pilha é sempre o último dado nela colocado. A pilha permite uma simples implementação de múltiplos níveis de interrupções, subrotinas e simplificação de muitos tipos de manipulação de dados.

3.3.4.3 Contador de Programa (PC)

O contador de programa (Program Counter) é formado por um registrador de 16 bits, no qual é mantido o endereço da corrente instrução que está sendo buscada na memória. O Contador de Programa PC é automaticamente incrementado de uma unidade após o seu conteúdo ter sido transferido para as linhas de endereço. Quando ocorre um Jump (salto) no programa, o novo valor é automaticamente colocado no Contador de Programa (PC).

3.3.4.4 Registradores Indexados (IX e IY)

Os dois registradores indexados são independentes e possuem um endereço base de 16 bits. São utilizados em modos de endereçamento. A sua utilidade será explicada no Volume II deste livro com maiores detalhes.

Neste modo, um registrador indexado é utilizado como base para apontar uma região específica de memória, na qual o dado deve ser armazenado ou retirado. Um byte é adicionado à instrução indexada para especificar o deslocamento nesta base.

Este tipo de registrador minimiza sensivelmente o Software de vários tipos de programas, em que o usuário tem a necessidade de percorrer tabelas.

3.3.4.5 Registrador de Interrupção de Endereço Paginado (I)

O microprocessador Z-80 possui um registrador de interrupção que é de grande ajuda quando se necessita trabalhar com vários periféricos.

O registrador (I) é utilizado para armazenar os 8 bits mais significativos do endereço indireto da instrução ou dado, enquanto que o dispositivo que provocou a interrupção fornece os 8 bits menos significativos do endereço.

Este procedimento permite que rotinas de interrupção sejam dinamicamente carregadas em qualquer lugar na memória, desde que seja em memória de leitura e escrita (RAM). A utilidade do registrador I será melhor entendida, quando tratarmos de interface paralela (capítulo 6).

3.3.4.6 Registrador de Refresh à Memória (R)

A CPU Z-80 contém um registrador utilizado como contador de refresh para a memória dinâmica, o qual simplifica a interface para essas memórias. Tais memórias guardam as informações em forma de capacitâncias que desaparecem com o tempo e por isso devem ser restauradas constantemente (Refresh).

O registrador de Refresh (R) contém 8 bits onde são usados apenas 7 bits. O dado contido no registrador de refresh (R) é enviado para a memória, através dos 7 bits menos significativos das linhas de endereço (A_0 a A_6), enquanto a CPU está decodificando e executando uma determinada instrução.

O Refresh é transparente ao usuário e não torna lenta a operação da CPU.

O usuário pode carregar o registrador (R) para finalidades de teste, mas este registrador normalmente não é utilizado. Durante o refresh, o conteúdo do registrador (I) é colocado nos 8 bits mais significativos do bus de endereço.

3.4 – DESCRIÇÃO DA PINAGEM

A CPU se apresenta na forma encapsulada de 40 pinos. Esses pinos de entrada e saída são mostrados na figura (3.3) sendo que suas funções estão descritas adiante.

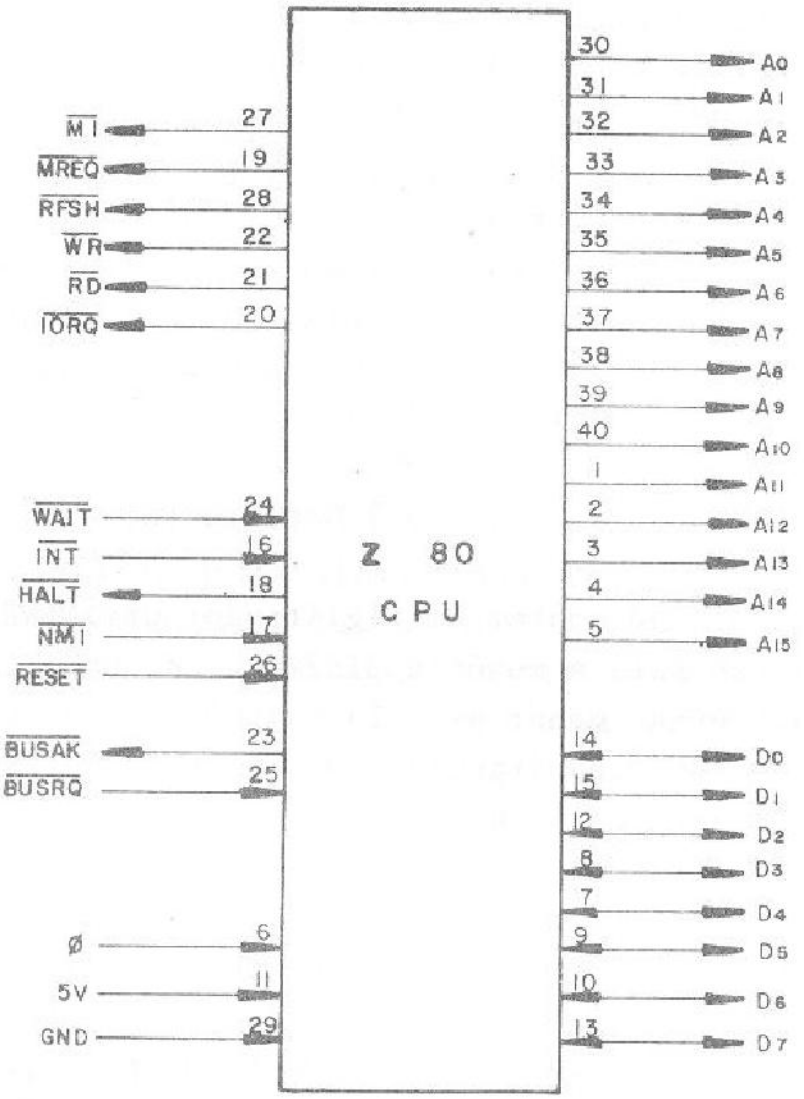


Fig. 3.3 - Pinagem do Z-80.

A_0 - A_{15} (SAÍDA)

ADDRESS BUS: Constituem 16 bits de endereço com saída em alta impedância (TRI-STATE). Esta via de endereços fornece a locação de memória assim como dispositivos de I/O. Pode-se endereçar 256 destes dispositivos, através de seus 8 bits menos significativos.

D_0 - D_7 (ENTRADA/SAÍDA)

DATA BUS: Constituem 8 bits de dados em via bidirecional com saída em alta impedância (TRI-STATE). Esta via de dados é usada para trocas de informações entre CPU e dispositivos de I/O e memória.

$\overline{M1}$ (SAÍDA)

MACHINE CYCLE ONE: Pino de saída, ativo em nível lógico baixo, indicando que a CPU está realizando um ciclo de busca. Para instruções de 2 bytes, $\overline{M1}$ é gerado para cada byte que foi buscado. Este sinal, em conjunto com $\overline{IORQ}$, também é usado para indicar o reconhecimento de uma interrupção.

$\overline{MREQ}$ (SAÍDA)

MEMORY REQUEST: Pino de saída, em estado de alta impedância (TRI-STATE), ativo em nível lógico baixo. Indica que a via de endereços possui um endereço para efetuar leitura ou gravação na memória.

$\overline{RFSH}$ (SAÍDA)

REFRESH: Pino de saída, que é ativo em nível lógico baixo, indica que os 7 bits menos significativos da via de endereços contêm a posição de memória a ser restaurada. Este sinal possibilita o uso de memórias RAM dinâmicas facilitando a elaboração do HARDWARE.

$\overline{WR}$ (SAÍDA)

MEMORY WRITE: Pino de saída em TRI-STATE, que é ativo em nível baixo, indicando que a via de dados contém um dado para ser armazenado em dispositivos de I/O ou em memória, dependendo do endereço contido na via de endereços.

$\overline{RD}$ (SAÍDA)

MEMORY READ: Pino de saída em TRI-STATE, que é ativo em nível lógico baixo, indicando que a CPU lerá dados em memória ou em dispositivos de I/0.

$\overline{IORQ}$ (SAÍDA)

INPUT/OUTPUT REQUEST: Pino de saída em TRI-STATE, que é ativo em nível lógico baixo, indicando que os 8 bits menos significativos da via de endereços possuem o endereço do periférico, ou seja, de um dispositivo de I/0, no qual será feita uma leitura ou escrita.

$\overline{WAIT}$ (ENTRADA)

WAIT: Pino de entrada, que é ativo em nível lógico baixo, indicando para a CPU que a memória ou periférico endereçado não está pronto para transferência de dados. A CPU ficará esperando enquanto o sinal estiver ativado. Este sinal possibilita o sincronismo entre CPU e memórias ajustando suas velocidades.

$\overline{INT}$ (ENTRADA)

INTERRUPT REQUEST: Pino de entrada, ativo em nível lógico baixo, gerado por periféricos, ou seja, dispositivos de I/0. Este sinal indica que um periférico está pedindo uma interrupção a qual será reconhecida no fim da instrução que está sendo executada, caso o $\overline{BUSRQ}$ não estiver ativo. Quando o pedido de interrupção é aceito pela CPU esta envia $\overline{IORQ}$ e $\overline{M1}$ como reconhecimento.

$\overline{HALT}$ (SAÍDA)

HALT STATE: Pino de saída, que é ativo em nivel lógico baixo, indicando que a CPU está executando uma parada (HALT) por instrução de Software aguardando uma interrupção. Em cada

estado HALT, a CPU nada realiza, mantendo o refresh de memória ativado.

$\overline{\text{NM I}}$ (ENTRADA)

NON MASKABLE INTERRUPT: Pino de entrada, ativo na borda de descida, Este sinal em prioridade superior ao do sinal $\overline{\text{INT}}$, independendo do Status e faz com que o Program Counter (PC) vá para a posição ØØ66 H. O PC é armazenado em uma pilha afim de poder retornar ao programa original, no ponto onde este sinal foi gerado.

$\overline{\text{RESET}}$ (ENTRADA)

RESET: Pino de entrada, ativo em nível lógico baixo. Este sinal faz com que o PC seja carregado com o endereço ØØØØ H e inicializa a CPU. Durante este sinal, todos os outros sinais ficam inativos e tanto a via de dados como a de endereços ficam em alta impedância (TRI-STATE).

$\overline{\text{BUS RQ}}$ (ENTRADA)

BUS REQUEST: Pino de entrada, ativo em nível lógico baixo. Este sinal requisita à CPU as vias de endereços, dados e controle colocando-as em TRI-STATE. Deste modo, a CPU deixa de usá-las permitindo que outros periféricos o façam. (Usado principalmente em acesso direto à memória - DMA).

$\overline{\text{BUSAK}}$ (SAÍDA)

BUS ACKNOWLEDGE: Pino de saída, que é ativo em nível lógico baixo, indicando que o sinal $\overline{\text{BUSRQ}}$ foi reconhecido e que outros dispositivos podem controlar as vias de dados, endereço e sinais de controle.

Ø (ENTRADA)

CLOCK PHASE: Única fase de nível TTL, requerendo apenas um resistor de 330 OHMS Pull-up ligado a + 5V, sendo que a oscilação é feita externamente. (vide circuito de "CLOCK").

CAPÍTULO 4 – CICLOS DE TEMPO E INTERRUPÇÃO

4.1 – CICLOS DE TEMPO DO Z-80

4.1.1 Introdução

O microprocessador Z-80 executa instruções básicamente como quase todos os microprocessadores existentes na praça, realizando os seguintes itens:

1.) Escrita e Leitura de memória (W/R)
2.) Escrita e Leitura de dispositivos, de entrada e saída (I/O).
3.) Interrupção (Acknowledge).

As instruções podem ter de 3 a 6 períodos de Clock. O período básico de Clock corresponde aos ciclos de "T" e as operações básicas a ciclos de máquina "M".

Como foi dito anteriormente uma instrução possui no mínimo três ciclos de máquina distintos (M_1, M_2 e M_3). O primeiro deles é um ciclo de Fetch que pode corresponder de 4 a 6 ciclos "T". Este ciclo de Fetch (M_1) ou ciclo de busca, é o tempo gasto para buscar a instrução a ser executada, sendo que a seguir os outros ciclos movem os dados entre CPU, memória e I/O.

A figura 4.1 mostra o diagrama básico de tempo de uma instrução.

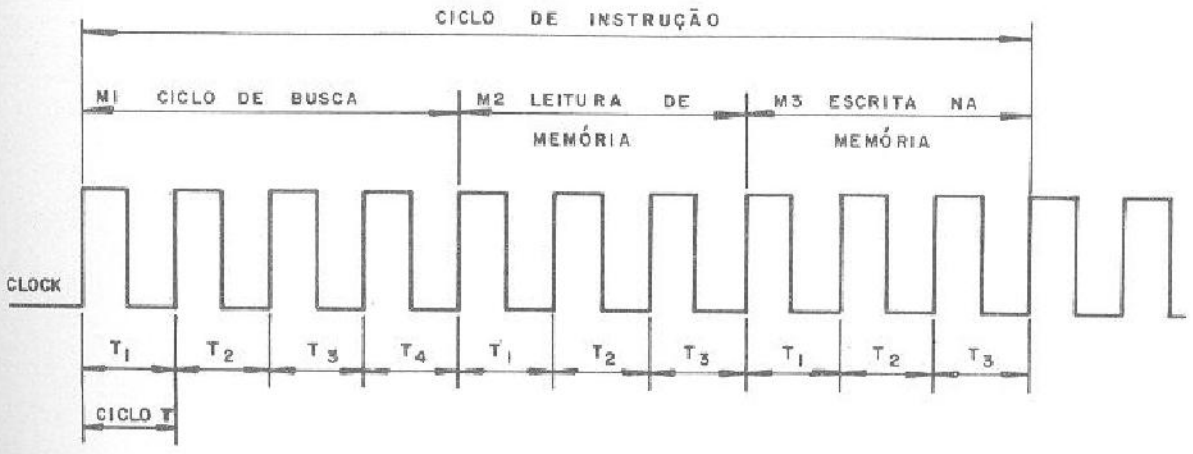


Fig. 4.1 – Diagrama Básico de Tempo.

4.1.2 Diagramas de Tempo

4.1.2.1 Ciclo de Busca (Instruction Fetch)

O diagrama, apresentado na figura 4.2, irá mostrar o primeiro ciclo de máquina (M_1). Ao analisarmos este ciclo, notemos que a primeira etapa é colocar o contador de programa (PC) na via de endereço (ADDRESS BUS).

No meio ciclo de Clock (T_1) é ativado o sinal $\overline{MREQ}$, o que nesta situação, significa leitura da memória, a segunda ativação é usada para o refresh de memórias dinâmicas.

A linha $\overline{RD}$ se mantém ativa no ciclo de leitura da memória onde neste instante o dado da memória é transferido para o registro de instrução, isto ocorrendo na subida do pulso T_3. Também na subida do pulso T_3 são desativados os sinais $\overline{MREQ}$ e $\overline{RD}$.

Os períodos de tempo T_3 e T_4 são usados para o refresh de memória dinâmica junto com o sinal $\overline{RFSH}$ ativado na subida de T_3 e desativado no fim de T_4. Este espaço aparentemente perdido é utilizado pela CPU para decodificação, interpretação e execução da instrução que veremos nos ciclos posteriores.

Obs.:

Sinal de $\overline{WAIT}$ está desativado (sinal de espera; ver descrição da pinagem 3.4).

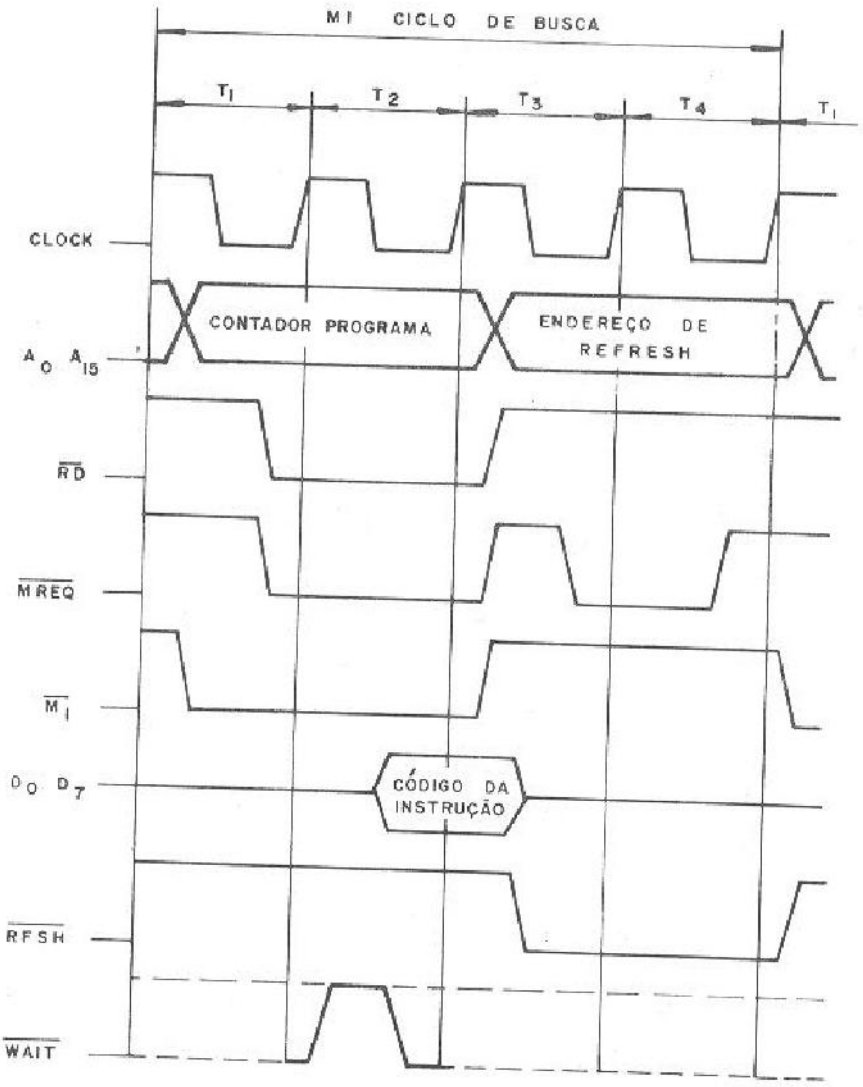


Fig. 4.2 - Ciclo de Busca de Instruções.

No diagrama de tempos da figura 4.3 o sinal de $\overline{WAIT}$ é ativado por algum dispositivo, por exemplo memória de acesso lento como são memórias Bolhas (MAGNETIC BUBBLE MEMORY), onde o tempo de acesso é de aproximadamente (1ms).

Se no ciclo T_2 o sinal de $\overline{WAIT}$ for ativado, indo a nível lógico "zero", o ciclo T_2 se repetirá subsequente a Tw, até que $\overline{WAIT}$ seja desativado indo para nível lógico "um", sendo que neste ponto se repetirá a análise anterior.

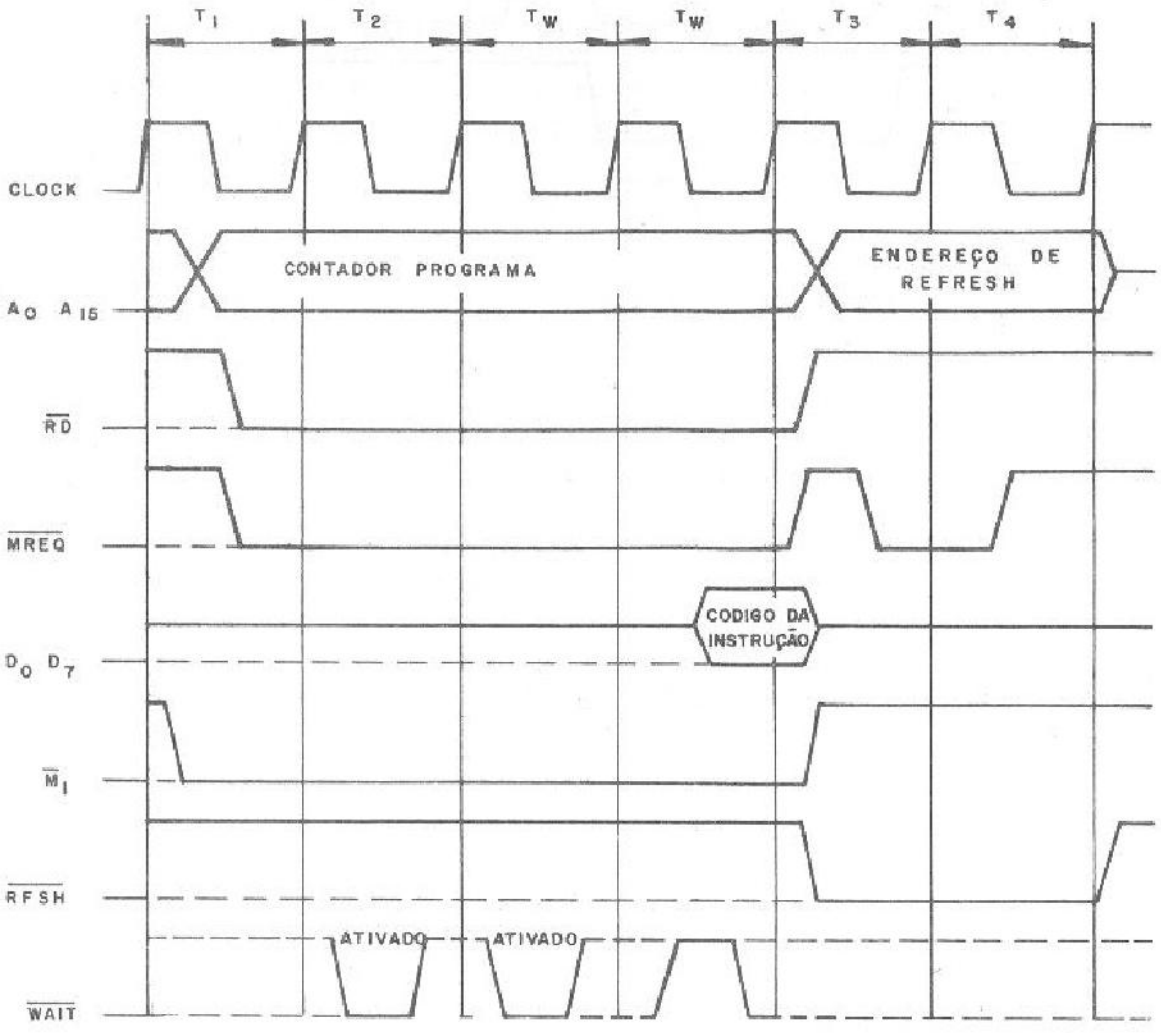


Fig. 4.3 - Ciclo de Busca com "WAIT" ativado.

4.1.2.2 Escrita e Leitura de Memórias

A figura 4.4 mostra o ciclo de leitura ($\overline{RD}$) e escrita ($\overline{WR}$).

Estes ciclos são quase sempre compostos por 3 Clocks, a menos que um sinal $\overline{WAIT}$ seja ativado como foi mostrado no ciclo de busca.

Os sinais $\overline{MREQ}$ e $\overline{RD}$ são usados do mesmo modo como no ciclo de busca.

O sinal $\overline{MREQ}$ torna-se ativo quando o endereço estiver estável na via de endereços (ADDRESS BUS), e então podemos utilizá-lo como sinal para habilitar (chip enable) as memórias dinâmicas. A linha $\overline{WR}$ é ativada quando o dado na via de dados atinge a estabilidade, portanto poderá ser usado como sinal de leitura/escrita (R/W) para qualquer memória a semicondutor.

O sinal $\overline{WR}$ é ativado na metade do ciclo T_1 e desativado na metade do ciclo T_3.

Nota importante: nunca serão ativados simultaneamente os sinais de leitura $\overline{RD}$ e de escrita $\overline{WR}$.

Obs.:

$\overline{WAIT}$ desativado.

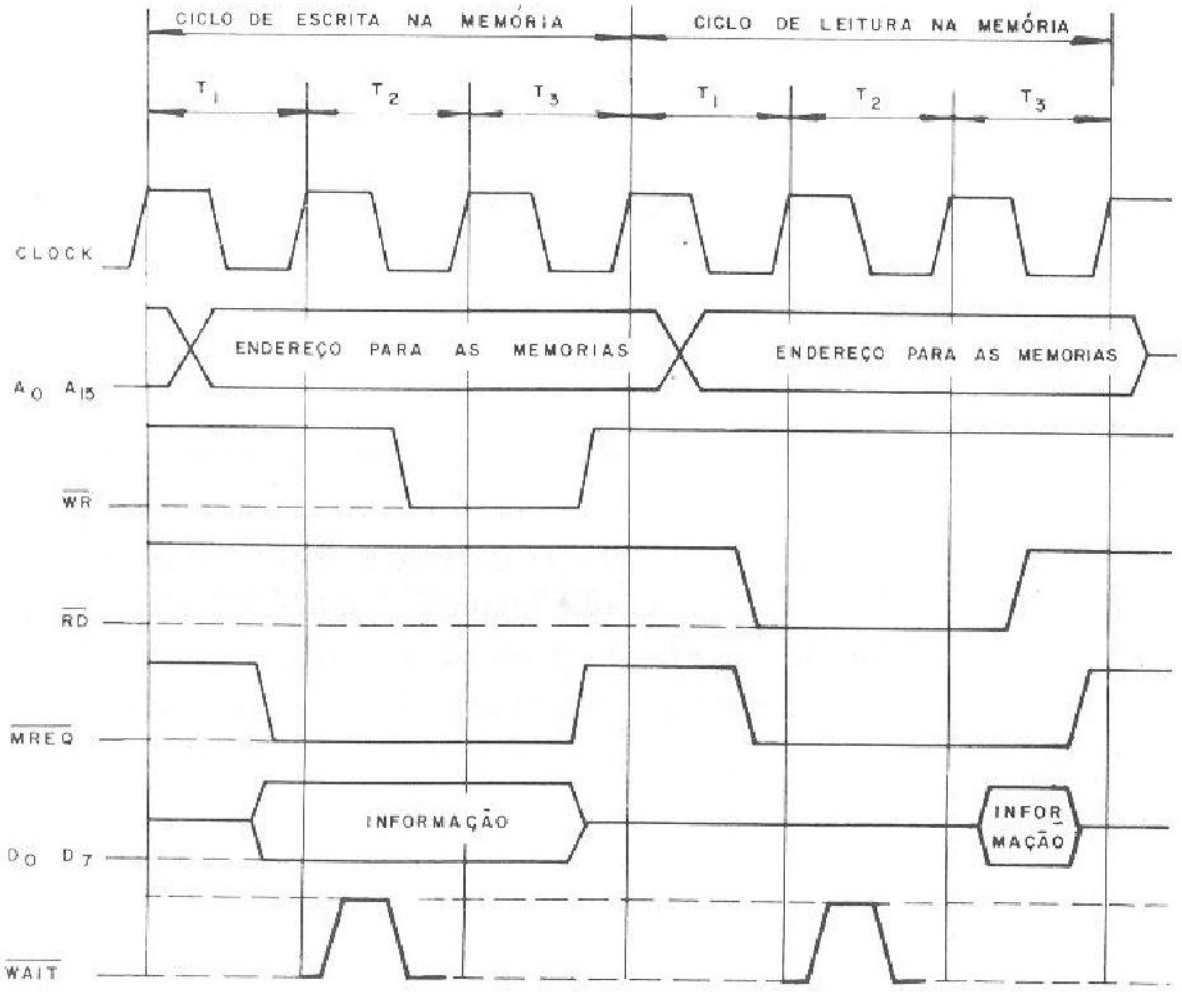


Fig. 4.4 - Ciclo de Leitura e Escrita na Memória.

A figura 4.5 mostra a mesma análise feita no ciclo de leitura $\overline{RD}$ e de escrita ($\overline{WR}$) mas com o sinal de $\overline{WAIT}$ ativado, fazendo com que todos os sinais fiquem inalterados até que o sinal $\overline{WAIT}$ seja desativado.

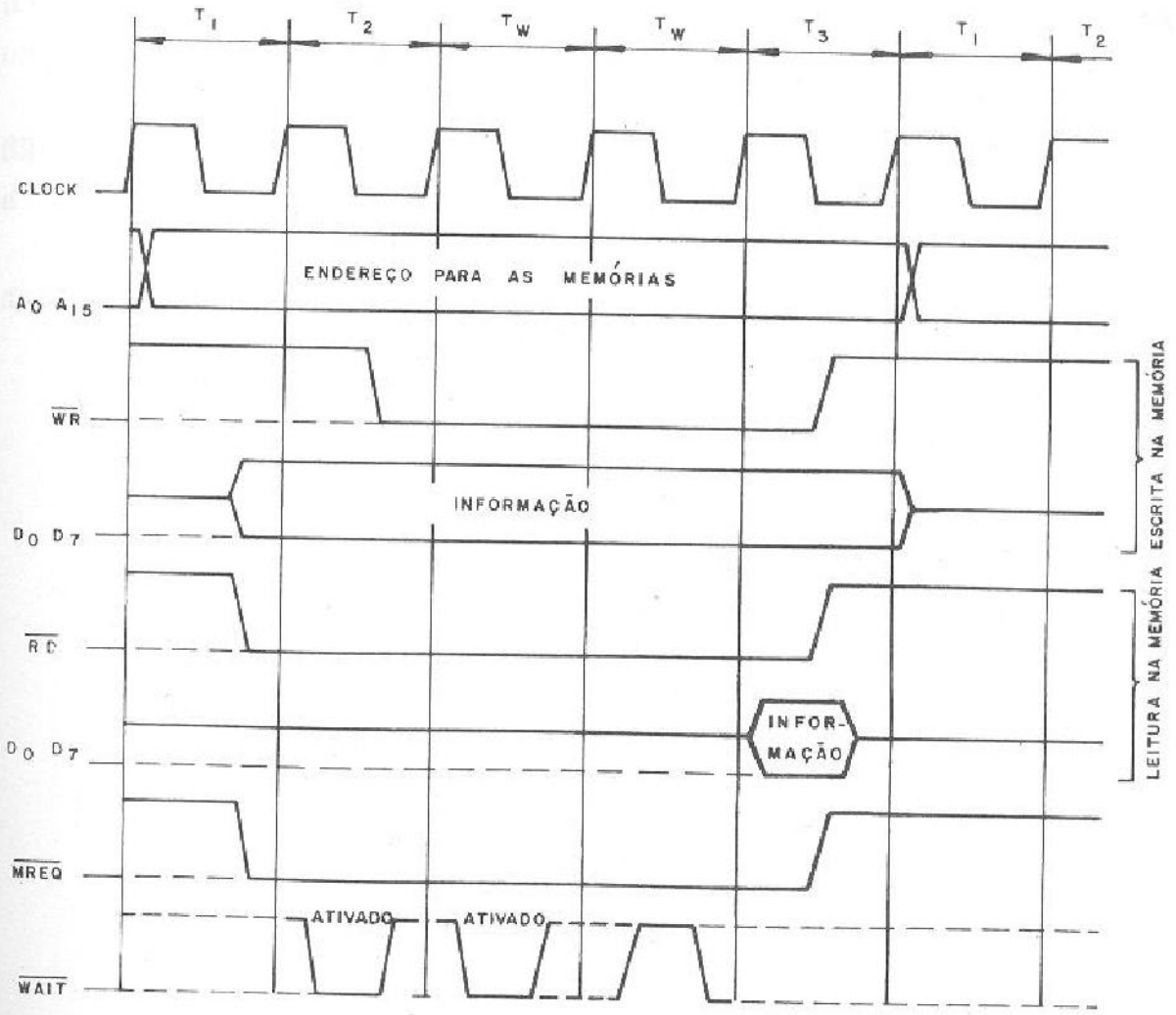


Fig. 4.5 - Ciclo de Leitura e Escrita com "WAIT" ativado.

4.1.2.3 Escrita e Leitura de Periféricos

Aqui estudaremos a leitura e escrita de periféricos (I/O) como visto na figura 4.6.

No ciclo de leitura e escrita como podemos ver, existe um ciclo em WAIT (Tω*). Este ciclo é acrescido, porque não

existe periférico que possa operar uma mesma velocidade da CPU.

Além disso, seria bastante difícil projetar circuitos externos que operassem na velocidade máxima da CPU.

Durante a operação de leitura do canal, o sinal $\overline{RD}$, habilita o endereço do dispositivo no data bus, como ocorre no caso de leitura de memória.

Para as operações de escrita nos periféricos, a linha $\overline{WR}$ é utilizada da mesma maneira que para escrita em memória.

Obs: Sinal $\overline{WAIT}$ desativado.

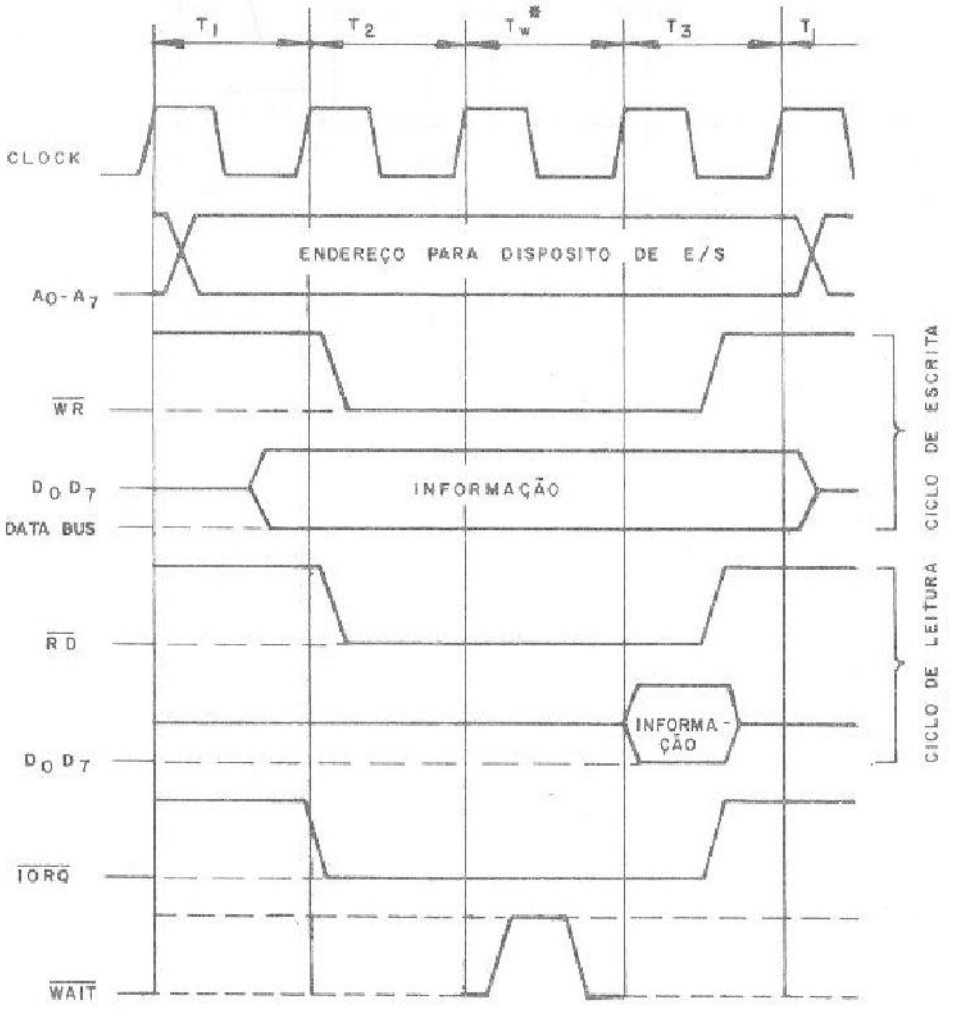


Fig. 4.6 - Ciclo de Leitura e Escrita em Periféricos.

Para a figura 4.7 teremos o mesmo estudo, mas levando em consideração o sinal $\overline{WAIT}$ ativado, com o mesmo princípio visto anteriormente.

Obs: Sinal $\overline{WAIT}$ desativado

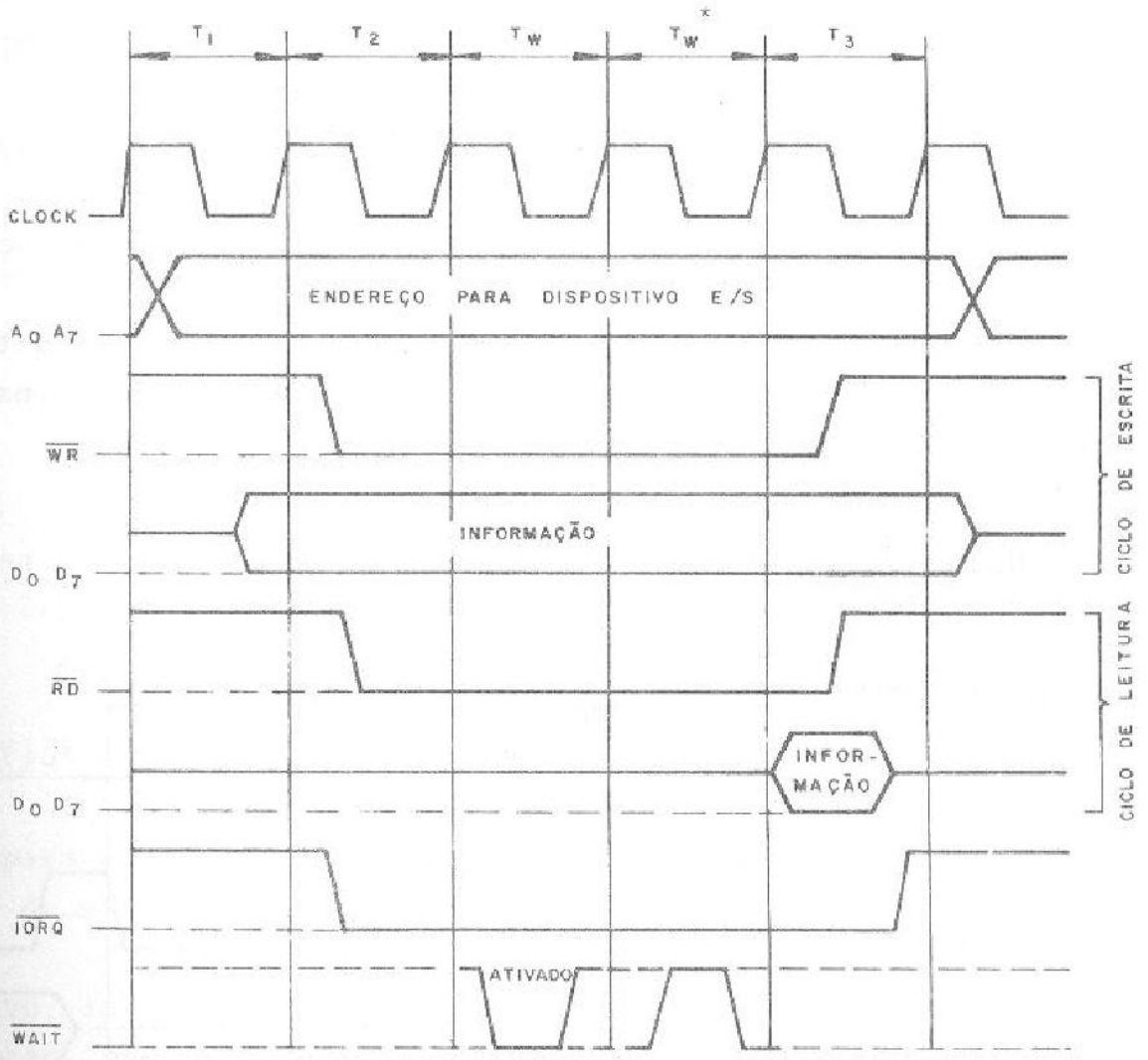


Fig. 4.7 - Ciclo de Escrita e Leitura com "WAIT" ativado.

4.1.2.4 Ciclo Request Acknowledge

Na figura 4.8 temos representado o diagrama de tempo quando existe requisição da via ou, em outras palavras, algum dispositivo externo está precisando das vias de endereço, da

dos e controle para suas operações.

Um exemplo típico desta situação é o pedido de acesso direto a memória (DMA). Isto ocorre quando no sistema existem memórias de grande capacidade (FLOP DISK), e a transferência de dados da memória de grande capacidade para a memória, onde será processado o programa (RAM), é feito diretamente sem passar pela CPU. Este artifício é usado para diminuir o tempo de transferência, veja a figura 4.9.

Isto ocorre quando o sinal $\overline{BUSRQ}$ é acionado ou seja, vai a nível lógico "zero".

O sinal $\overline{BUSAK}$ indica que a requisição da via foi aceita, em outras palavras, as vias estão em TRI-STATE.

Neste instante devemos tomar cuidado, se no circuito existirem RAMS dinâmicas. Caso existam deverão receber um sinal de refresh de um dispositivo externo e não da CPU.

Obs.:

Neste instante a CPU não aceita interrupções, nem por $\overline{NMI}$ ou por $\overline{INT}$.

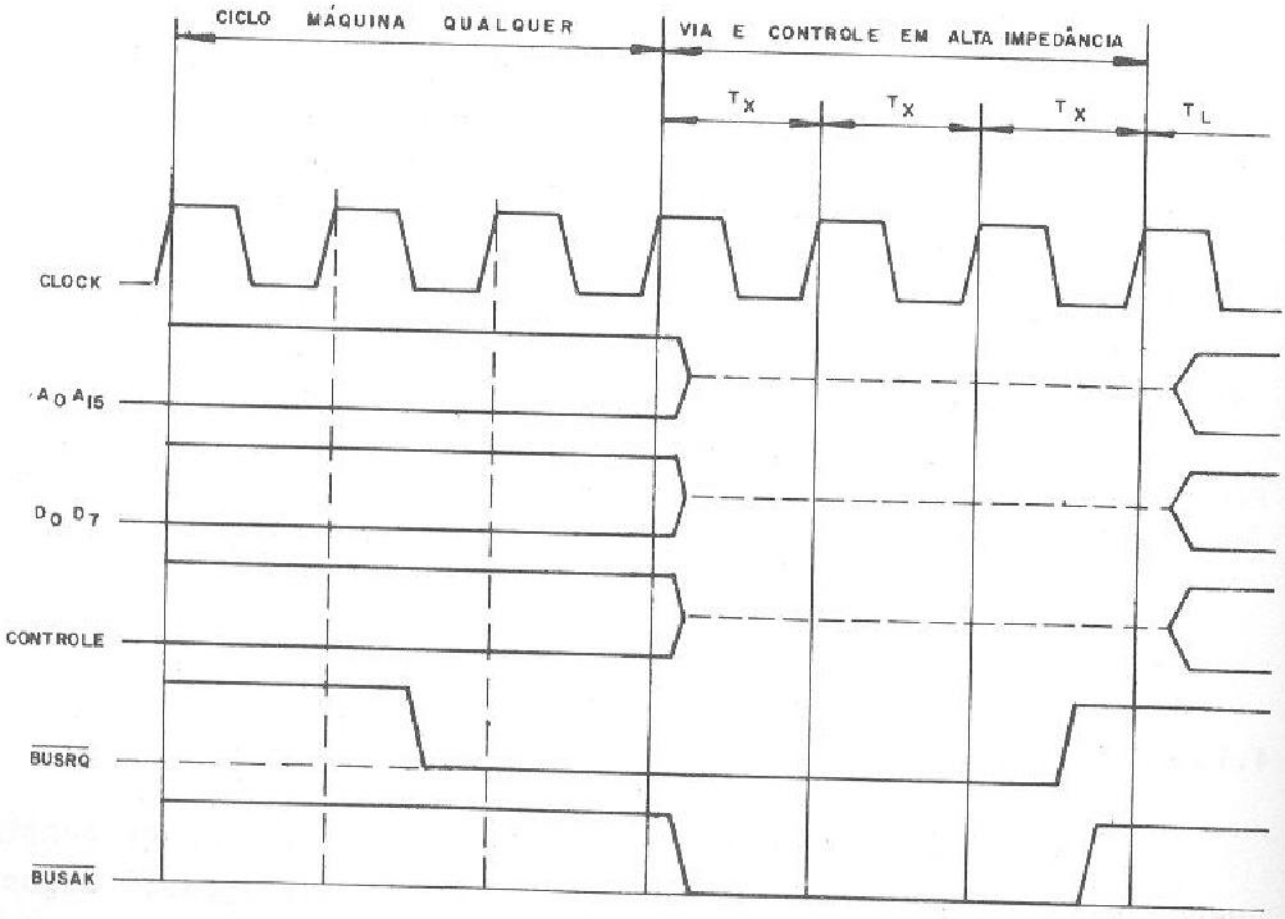


Fig. 4.8 - Ciclo Request Acknowledge.

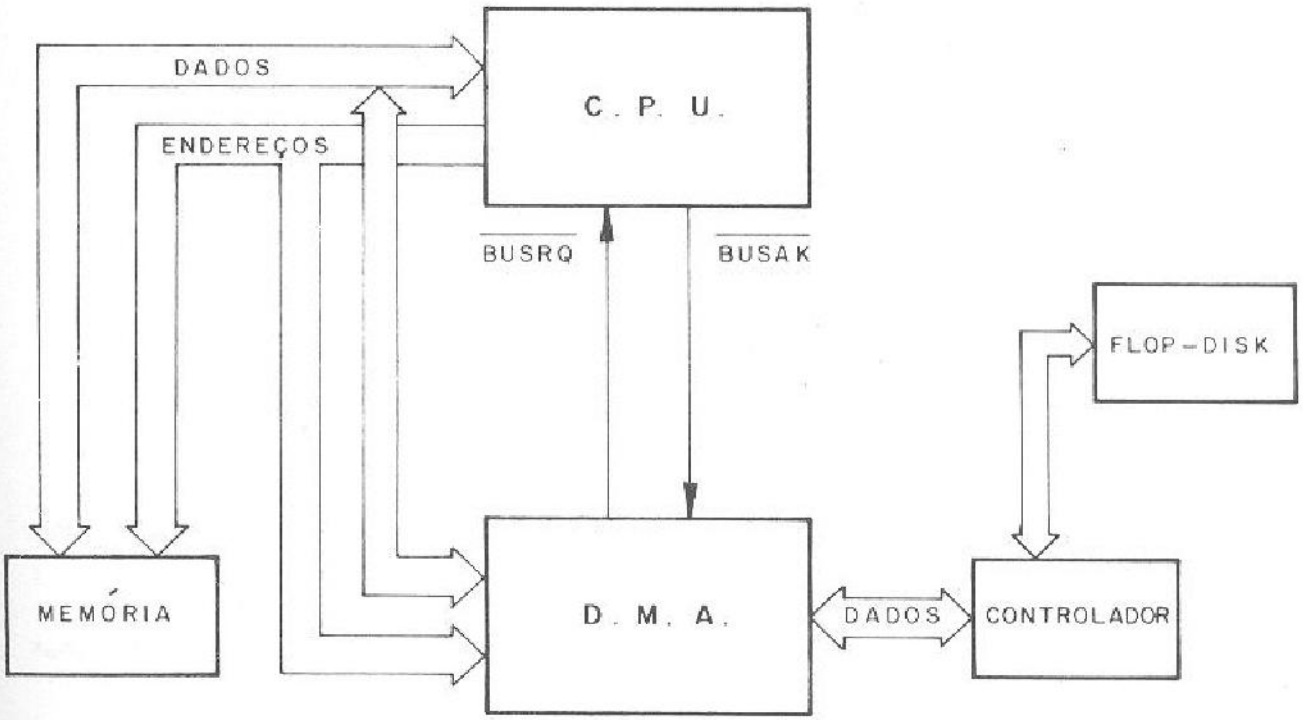


Fig. 4.9 - Esquema para DMA.

4.1.2.5 Ciclo de Interrupção

A figura 4.10 está mostrando o ciclo de interrupção ($\overline{INT}$).

Quando o sinal de interrupção é ativado, isto é, indo para nível lógico "zero", a CPU só o pesquisará no último ciclo de qualquer instrução.

Como foi dito anteriormente se o sinal $\overline{BUSRQ}$ estiver ativo, isto é, a nível lógico "zero", não será aceita interrupção, também o Flip-Flop interno deverá estar desabilitado, sendo que esta desabilitação é feita por Software.

Ao se aceitar a interrupção, é gerado um ciclo de M_1 especial e durante ele, o ciclo de $\overline{IORQ}$ torna-se ativo. Esta descrição é uma exceção a regra, porque normalmente seria ativado o $\overline{MREQ}$ com o objetivo de indicar que o dispositivo requisitante pode colocar um vetor ou uma palavra de 8 bits na via de dados para ser guardada em uma posição de memória, indicada pelo registrador de interrupção.

Dois Status de $\overline{WAIT}$ são colocados neste ciclo, isto, para tornar possível a realização de uma cadeia de interrupção com prioridade.

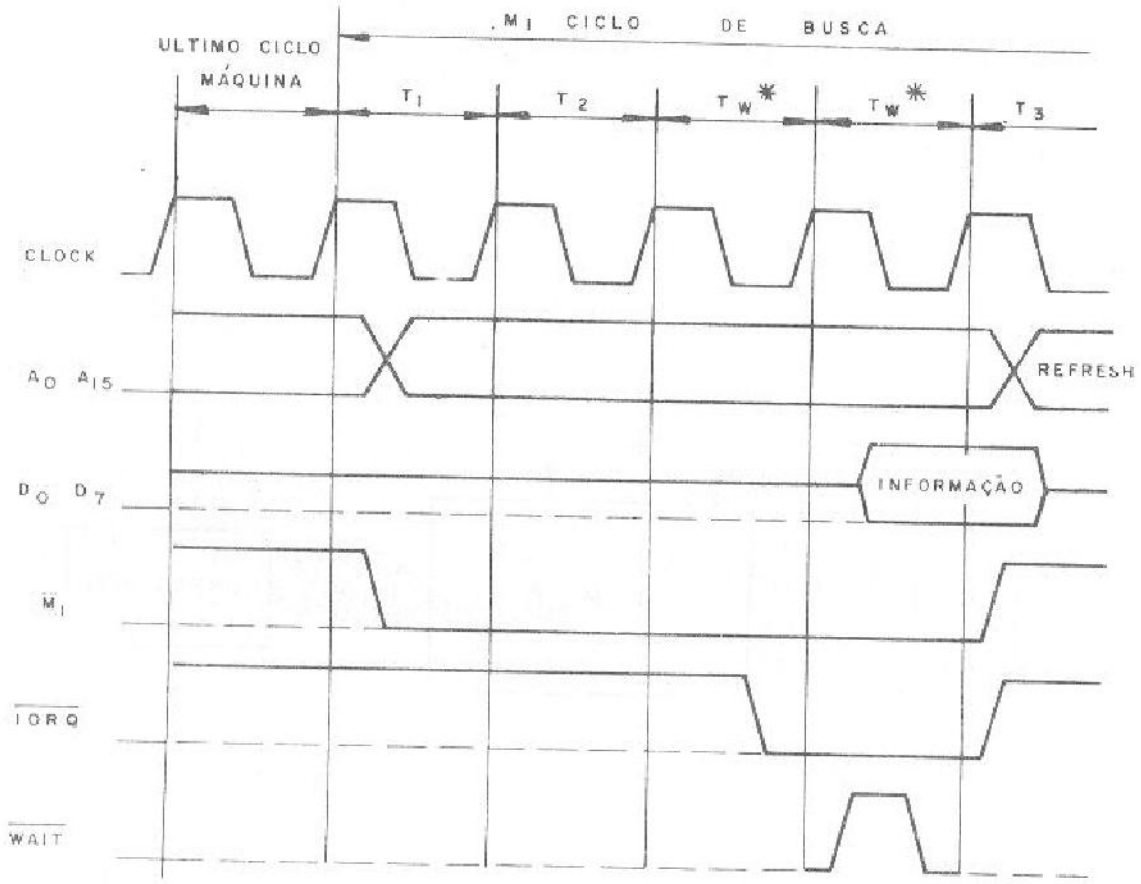


Fig. 4.10 - Ciclo de Interrupção.

Na figura 4.11 está representado o Status de $\overline{WAIT}$ e um ciclo de repouso de interrupção.

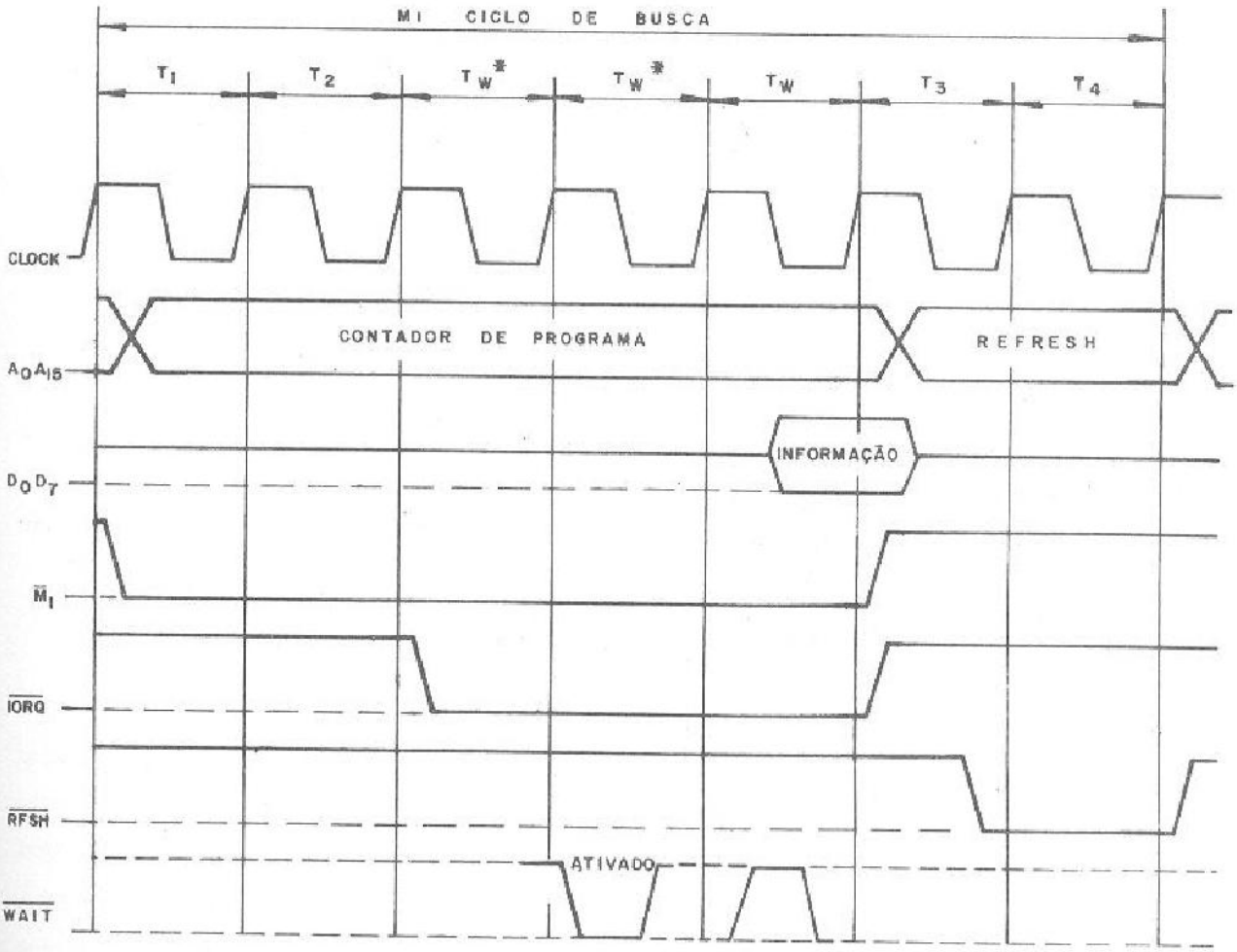


Fig. 4.11 - Ciclo de Interrupção com Status de "WAIT".

4.1.2.6 Resposta de Interrupção Não Mascarada

A interrupção não mascarada é apresentada com detalhes na figura 4.12.

O sinal $\overline{NMI}$ é verificado pela CPU, ao mesmo tempo que o sinal $\overline{INT}$, porém esta linha tem prioridade sobre a interrup

ção normal. Esta interrupção não pode ser desabilitada por Software como a $\overline{INT}$ visto anteriormente.

Este tipo de interrupção é usada apenas em operação de suma importância (ex.: Falta de Energia).

A resposta da CPU a um $\overline{NMI}$ assemelha-se a uma operação de leitura da memória. A única diferença é que o conteúdo do data bus é ignorado, enquanto que o processador armazena o PC no STACK POINTER e salta para a posição ØØ66 H.

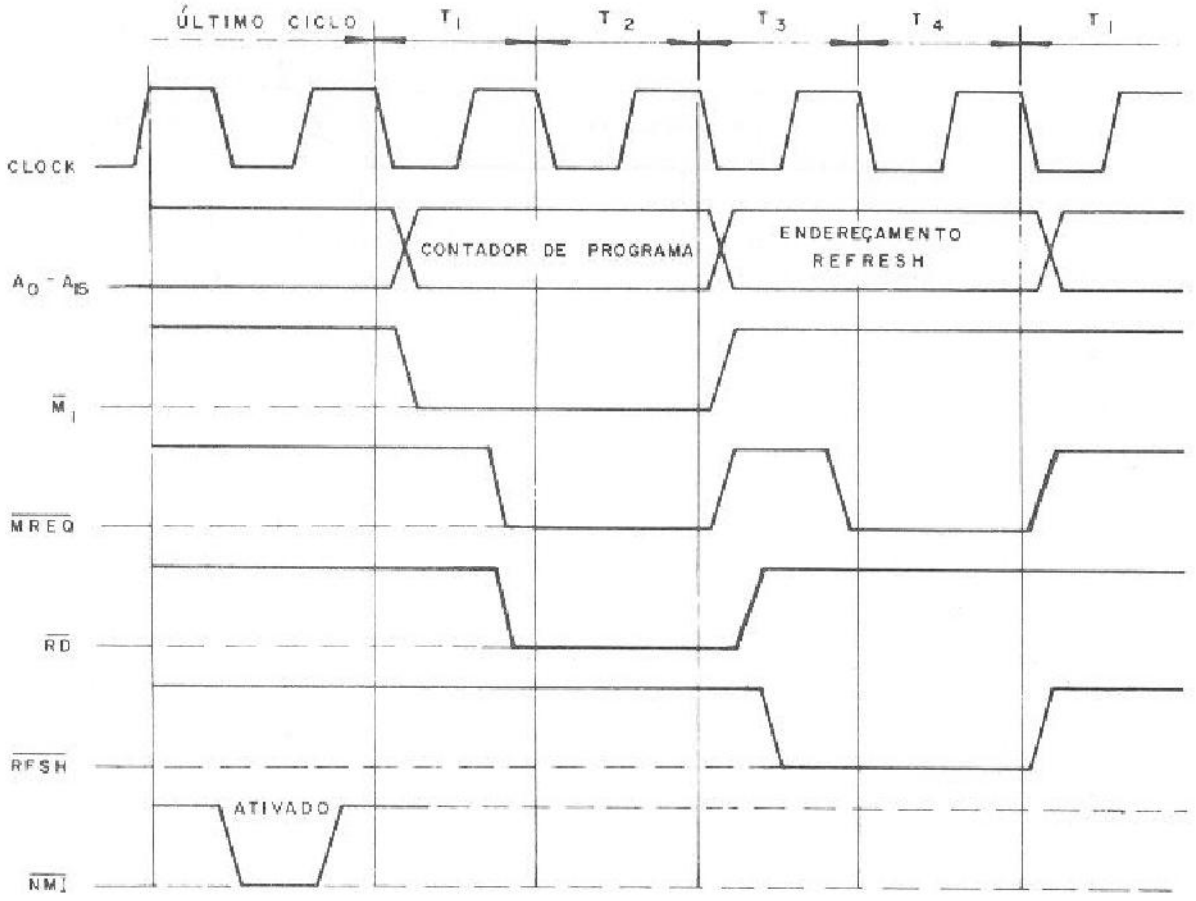


Fig. 4.12 - Interrupção não mascarável.

4.1.2.7 HALT

Sempre que no fim de algum programa de qualquer espécie existir uma instrução Halt, a CPU começa a executar instruções NOPS (NO OPERATIONS) até que uma interrupção seja requeri

da ou um reset geral seja dado.

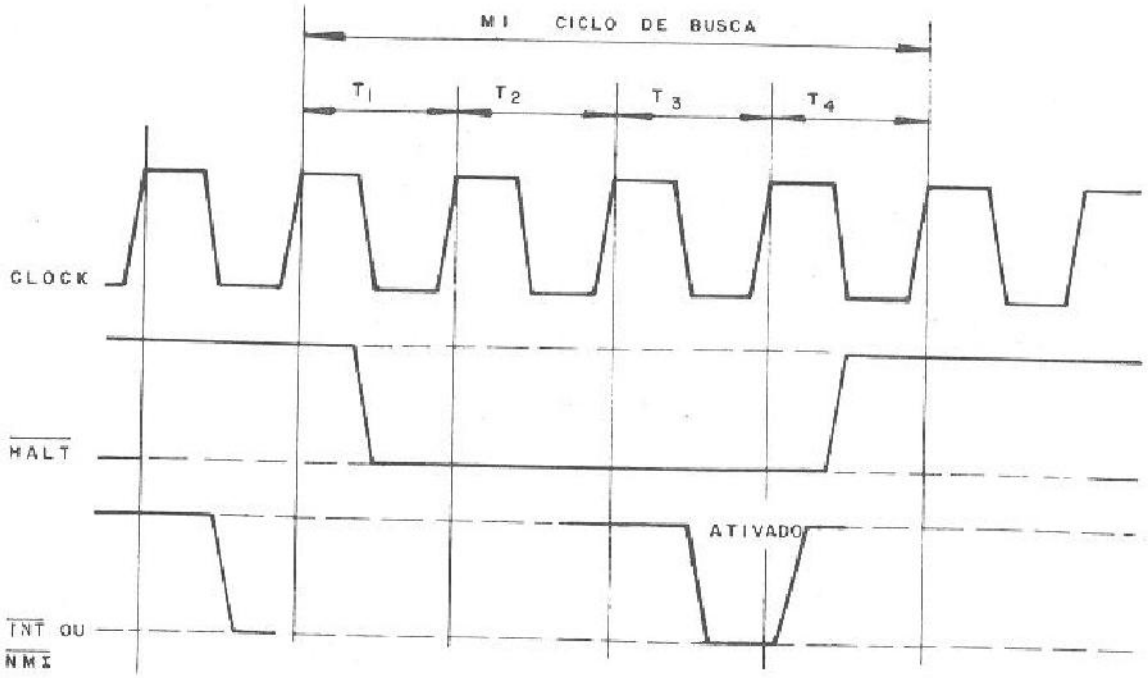


Fig. 4.13 - Ciclo de Halt.

4.2 – SISTEMAS DE INTERRUPÇÃO

4.2.1 Introdução

Quando um dispositivo periférico necessita da CPU, ele gera um pedido de interrupção, fazendo com que a CPU suspenda sua operação normal a fim de atender ao seu pedido através de uma rotina previamente programada. Ao término desta rotina, a CPU volta a executar as operações as quais executava antes da interrupção.

4.2.2 Modos de Interrupção

Entrada $\overline{NMI}$: Um sinal nesta entrada é de imediato aceito pela CPU em qualquer instante pois esta interrupção tem prioridade sobre qualquer outra, não importando as condições do programa. Neste modo de interrupção o PC pula automaticamente para a posição ØØ66 H. Tal recurso, como já dissemos, só é utilizado em casos extremos como falta de energia.

Entrada $\overline{INT}$: Um sinal nesta entrada, solicitando interrupção, pode ser aceito ou não, conforme a programação de sua máscara interna.

Esta interrupção se apresenta de 3 modos:

MODO Ø: Neste modo, é semelhante ao interrupt do microprocessador 8080 sendo sua máscara programada pela instrução IMØ. A interrupção causa um salto no programa cujo endereço de destino é dado conforme o código colocado no Data Bus durante o tempo em que $\overline{IORQ}$ e $\overline{M_1}$ estão ativados.

A correlação entre os códigos e endereços é mostrada na figura 4.14.

CÓDIGO	ENDEREÇOS
C7	00 H
CF	08 H
D7	10 H
DF	18 H
E7	20 H
EF	28 H
F7	30 H
FF	38 H

Fig. 4.14 - Relação entre Código e Endereço de Destino.

MODO 1: Um sinal na linha $\overline{INT}$ faz com o programa salte para a posição 38 H, sendo programada pela instrução IM 1.

MODO 2: Neste modo, o endereço de salto é composto. Sua parte mais significativa é composta pelos 8 bits do registro de interrupção I e a parte menos significativa pelos 7 bits do vetor interrupção do dispositivo periférico (para maiores detalhes vide item 5.3.1) sendo que o bit restante (1º bit menos significativo) deverá ser sempre Ø. Este modo é programado pela instrução IM 2, permitindo acesso a qualquer lugar de memória.

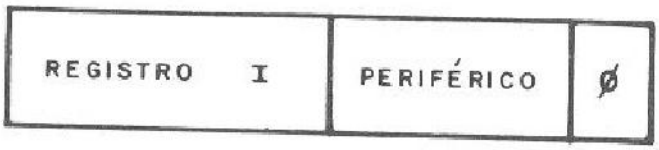


Fig. 4.15 - Composição de Endereço Destino para modo 2.

CAPÍTULO 5 – CIRCUITO CONTADOR/MARCADOR DE TEMPO – CTC

5.1 – INTRODUÇÃO

O circuito contador/marcador de tempo, ou simplesmente CTC, é um elemento que para operar corretamente deve ser programado pela CPU, Unidade Central de Processamento, ou seja, deve receber uma palavra de comando que determina como o CTC vai operar. Ele possui quatro canais que são independentes e têm por função contagem ou marcação de tempo (Timer) em sistemas que utilizam microprocessadores.

O CTC devidamente programado, pode servir de interface entre a CPU e um grande número de periféricos.

Vamos agora, ressaltar algumas características do CTC:

- Todas as entradas e saídas deste componente são totalmente compatíveis em níveis TTL.

- Para o controle das prioridades de interrupção, não é necessário nenhum tipo de circuito externo.

- Certas situações podem ser programadas para ocorrerem, quando em um determinado canal, a contagem do contador decrescente (Down Counter) atingir zero. Exemplo de aplicação: " Acionamento de Alarmes".

- Cada canal pode ser selecionado para operar no modo contador (Counter mode) ou no marcador de tempo (Timer mode).

5.2 – ARQUITETURA INTERNA

A arquitetura interna do CTC consiste basicamente de uma lógica interna de controle de interrupção, quatro contadores que correspondem ao canal 0, canal 1, canal 2, e canal 3, e ainda uma interface que interliga o bus de dados e de endereço da CPU com o CTC. Na figura 5.1, temos o diagrama em blocos do CTC.

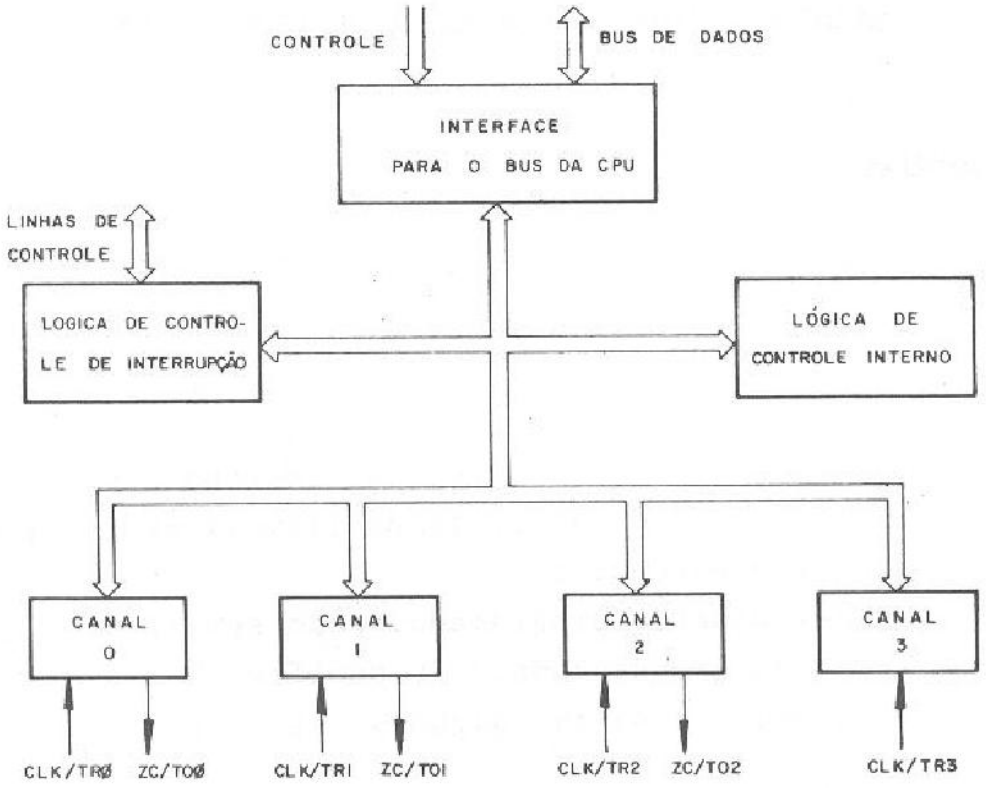


Fig. 5.1 - Diagrama em blocos do CTC.

A seleção de cada canal pela CPU, é feita através das linhas de endereço A_0 e A_1 da CPU que estão ligadas aos pinos CSØ e CS1 do CTC. O modo de seleção é visto na figura 5.2.

	CS1	CSØ	
CHØ	0	0	- maior prioridade
CH1	0	1	
CH2	1	0	
CH3	1	1	- menor prioridade

Fig. 5.2 - Seleção dos canais.

5.3 – ESTRUTURA INTERNA DO CANAL

A estrutura de cada canal é composta de uma lógica de controle, dois contadores de 8 bits cada um e de dois registradores também de 8 bits cada. Esta estrutura é vista na fig. 5.3.

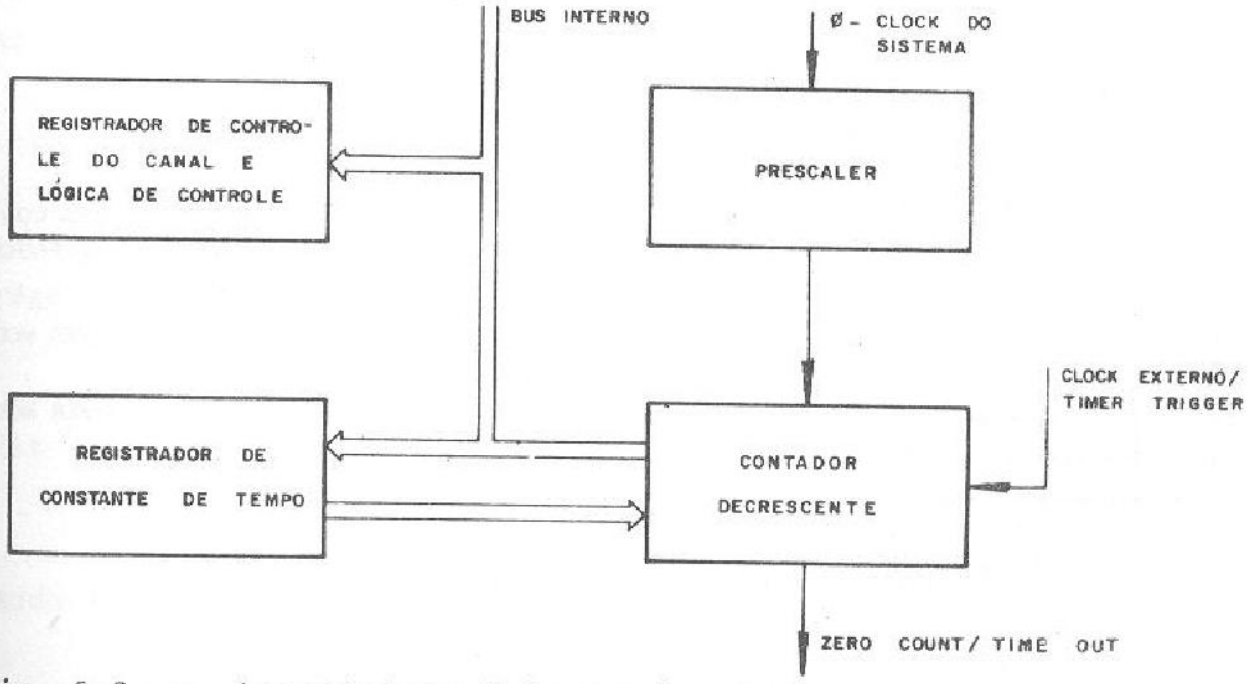


Fig. 5.3 – Arquitetura Interna do canal.

O CTC antes de começar a operar como contador ou timer, deve ser programado pela CPU através da palavra de controle do canal e da palavra que dá a constante de tempo, escritas no registrador de controle do canal (CHANNEL CONTROL REGISTER) e no registrador da constante de tempo (TIME CONSTANT REGISTER), respectivamente, do canal selecionado.

O carregamento dessas palavras é feito pela CPU num processo normal de escrita em periféricos, utilizando-se de suas linhas de endereço A_0 e A_1 e dos pinos CSØ e CS1 do CTC para selecionar o canal.

5.3.1 Channel Control Register

O registrador de controle de canal contém a palavra de controle do canal, enviada pela CPU através do data bus,

que seleciona o modo como o canal irá operar. Nesta palavra de controle, o bit DØ está sempre setado, ou seja, está sempre em nível lógico "um" e os 7 restantes selecionam o modo de operação. Esta palavra é mostrada na figura 5.4.

O Bit DØ deverá estar a nível lógico "zero" para indicar a palavra de interrupção sendo os 7 restantes Bits; neste caso conterá a palavra de interrupção.

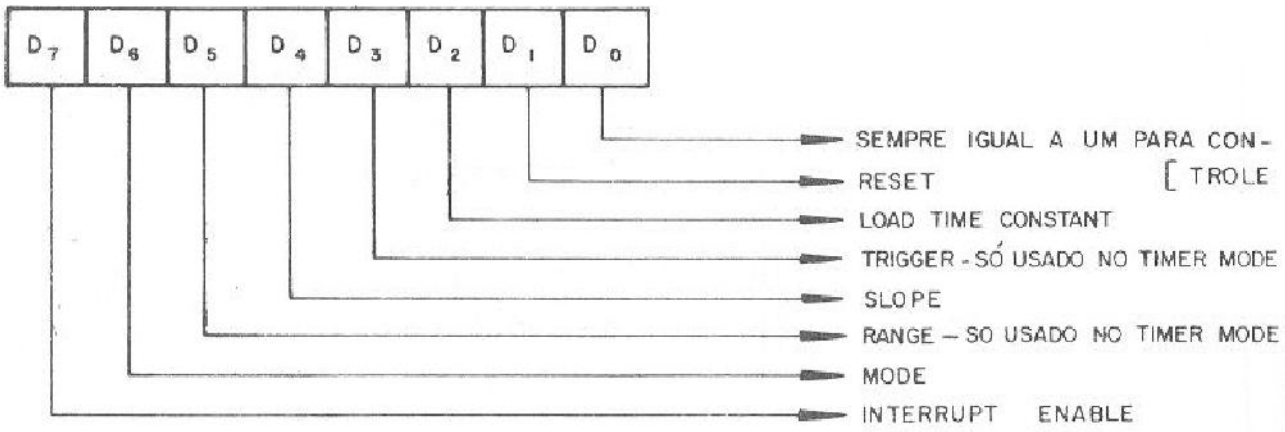


Fig. 5.4 - Palavra de Controle do C.T.C.

Bit 1:

Se for igual a "zero", o canal continuará a executar sua operação atual.

Se for igual a "um", fará com que o canal interrompa sua operação atual quer seja contar ou timer. Nesta condição o conteúdo do registrador de controle não será alterado.

Bit 2:

Se for igual a "zero", indica que a palavra que fornece a constante de tempo não será recebida logo após a palavra de controle. Assim, esta se refere ao estado atual do canal uma vez que ele não poderá atuar se a palavra de controle não for corretamente programada.

Se for igual a "um", indica que a próxima palavra a ser escrita no canal é a que fornece a constante de tempo do Time Constant Register. Toda vez que esta palavra for escrita em um canal ainda em operação, o contador decrescente (Down Counter)

continuará a decrementar até atingir "zero"e só então a nova constante de tempo será carregada.

Bit 3:

Se for igual a"zero",indica que após o carregamento da constante de tempo no canal, a operação Timer terá início no 2º período do ciclo de máquina corrente.

Se for igual a"um",indica que o sinal de Trigger externo (External Trigger) será usado para disparar a operação de Timer. Contudo, este sinal só será válido para isso, depois do 2º período de ciclo de máquina que ocorre após o carregamento da constante de tempo no canal.

Este bit só é usado no modo Timer.

Bit 4:

Este bit tem função distinta no modo contador e no modo Timer.

Se for igual a"zero", e estivermos operando no modo contador, indica que o Down Counter será decrementado na borda de descida do sinal Trigger. Caso estivermos operando no modo Timer, indica que a borda de descida do sinal Trigger dará início à operação Timer.

Se for igual a"um",e estivermos operando no modo contador, indica que o Down Counter será decrementado na borda de subida do sinal Trigger. Caso estivermos operando no modo Timer, indica que a borda de subida do sinal Trigger dará início à operação Timer.

Bit 5:

Este bit define qual será o fator do Prescaler que será usado, sendo definido só para o modo Timer. Para maiores detalhes vide item 5.3.3.

Se for igual a"zero",indica que o clock do sistema será dividido por 16, ou seja, o fator de divisão do Prescaler é 16 e que este sinal será usado como clock do Down Counter.

Se for igual a "um", indica que o clock do sistema será dividido por 256 sendo este o fator de divisão do Prescaler.

Bit 6:

Este bit seleciona o modo de operação do canal.

Se for igual a "zero", indica que o modo selecionado foi o modo Timer. Neste caso, o Prescaler terá como clock o próprio clock externo do sistema e o sinal que sua saída fornece será usado como clock para o Down Counter. Este por sua vez fornecerá em sua saída Zero Count/Time Out (ZC/TO) um trem de pulsos uniformes.

Se for igual a "um", indica que o modo selecionado foi o modo contador. Agora o Prescaler não atua mais e o Down Counter tem como clock o próprio clock externo do sistema que se encontra na entrada Clock External/Timer Trigger (CLK/TRG).

Bit 7:

Se for igual a "zero", indica que o sistema de interrupção do canal foi desativado, ou seja, desabilitado. Neste caso, qualquer possível interrupção pelo canal será anulada.

Se for igual a "um", indica que a interrupção está ativada, ou seja, está habilitada. Contudo, um canal só poderá requisitar uma interrupção à CPU a partir do instante que o Down Counter atinja o valor zero quando teremos um pulso na saída Zero Count.

Uma vez o canal em operação com este bit setado, a seleção de interrupção habilitada não poderá ser revertida mesmo que outra palavra de controle seja escrita. Antes de se ter início a operação do canal com este bit em um, é preciso que o vetor de interrupção tenha sido carregado no CTC.

Aqui cabem algumas considerações sobre o vetor de interrupção. Ele é usado para compor o ponteiro que indica a posição de memória que contém o endereço da rotina de interrupção. Os 8 bits mais significativos do ponteiro são dados pelo registrador I e os 8 bits menos significativos pelo vetor de in

terrupção. O endereço assim apontado contém o byte menos significativo e o próximo superior, o byte mais significativo do endereço onde se encontra a primeira instrução da rotina de interrupção.

5.3.2 Down Counter

O contador decrescente é um registrador usado tanto no modo contador como no modo Timer. Nele é carregado a constante de tempo presente no registrador de constante de tempo (Time Constant Register), a qual é decrementada até atingir zero, quando teremos um pulso na saída Zero Count/Time Out (ZC/TO).

Dependendo do modo em operação, o decremento do conteúdo do Down Counter é controlado por sinais diferentes, ou seja: no modo contador o decremento ocorre na borda de cada pulso do sinal clock externo, e no modo Timer isto ocorre a cada pulso do sinal fornecido pelo Prescaler, que é apenas uma variação do clock do sistema, como veremos adiante.

A CPU pode obter um número que está sendo decrementado, ascessando o conteúdo deste registrador, bastando para isso selecionar o canal através da operação de leitura em periféricos (I/O).

Se em alguma aplicação se fizer necessário o uso da saída Zero Count, devemos utilizar um dos três primeiros canais (0, 1, 2) pois o quarto canal não apresenta esta saída.

5.3.3 Prescaler

O Prescaler é um contador de 8 bits, usado somente no modo Timer, sendo programado pela CPU através da palavra de controle armazenada no registrador de controle do canal (Channel Control Register).

Tem por função dividir sua entrada, clock do sistema (ϕ), por um fator 16 ou 256 conforme a programação. Sua saída é então utilizada como clock do contador decrescente (Down Counter) para a operação de decremento da constante de tempo.

A cada instante em que o decremento atinge zero, te

remos um pulso de nível alto na saída ZC/TO do Down Counter.

5.3.4 Time Constant Register

O registrador de constante de tempo, é um dispositivo de 8 bits que é usado tanto no modo contador como no modo Timer.

Tem por função como o próprio nome sugere, guardar a palavra de constante de tempo programada pela C P U que será usada em ambos os modos de operação. Esta constante é escrita no registrador através de um ciclo de escrita em periférico (I/O) para o canal selecionado, ciclo este vindo logo após ao ciclo de escrita da palavra de controle do canal.

O fornecimento ao canal desta constante é imperativo, pois sabemos que o canal não poderá entrar em operação sem ela. Tal constante pode assumir valores entre 1 e 256 inclusive, sendo que quando todos os bits desta palavra forem zero será interpretado como 256.

Uma nova constante de tempo será carregada do Time Constant Register para o Down Counter quando este atingir zero em sua contagem.

A forma da palavra de constante de tempo é vista na fig. 5.5, onde TCØ - TC7 dão a constante entre 1 e 256.

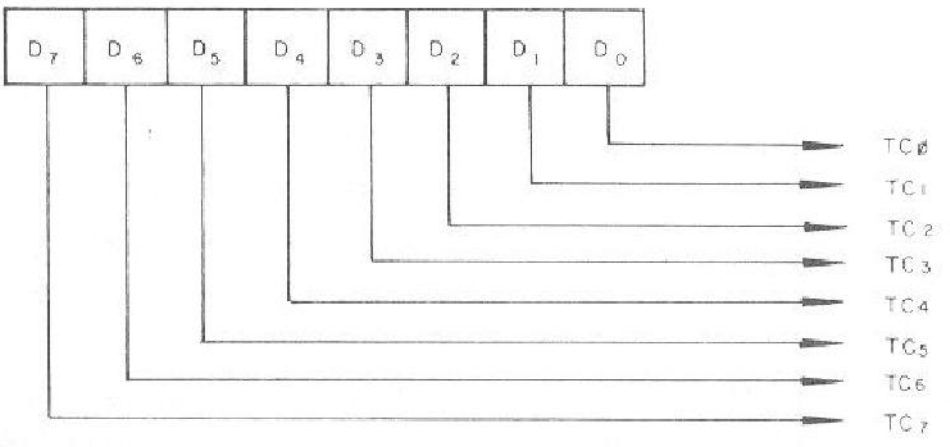


Fig. 5.5 - Palavra da constante de tempo.

5.4 – MODOS DE OPERAÇÃO

Ao acionarmos o CTC, este se apresenta inoperante e com seus estados internos indefinidos.

Para que entre em operação, a CPU deve escrever nos dispositivos correspondentes do CTC as palavras de comando, de constante de tempo e o vetor de interrupção, se algum canal tiver sua lógica de interrupção ativada.

Em seguida, temos a descrição dos dois modos de operação.

5.4.1 Timer mode

No modo marcador de tempo, o CTC tem por função gerar sinais cujo período de tempo é um número inteiro do período de clock do sistema.

A seleção deste modo de operação é realizada fazendo-se o bit 6 (mode) da palavra de controle do canal, igual a zero, podendo então o canal atuar como medidor de intervalos de tempo, baseando-se no período do clock do sistema.

Este clock é fornecido diretamente para o contador Prescaler e indiretamente ao Down Counter, isto se deve ao fato de que o Prescaler divide o clock do sistema por um fator que pode ser de 16 ou 256, dependendo de como se encontra o bit 5 (Range) da palavra de controle; e sua saída será utilizada como sinal de clock para a operação de decrementar o conteúdo do Down Counter, conteúdo este que pode ser programado entre 1 e 256.

O modo como a contagem terá início é dado pela programação do bit 3 (Trigger) na palavra de controle. A contagem poderá ter início no ciclo seguinte ao de escrita em periféricos que carregou a palavra de constante de tempo no canal, ou na segunda borda positiva do clock do sistema seguinte à borda de disparo da entrada Timer Trigger.

Se a interrupção pelo canal estiver habilitada, ou seja, o bit 7 (Interrupt Enable) da palavra de controle é igual a um, a condição do Down Counter atingir zero dará início a uma seqüência de requisição de interrupção.

O trem de pulsos presente na saída Time Out, quando o Down Counter atinge zero, tem o período dado pela expressão:

$$T = P \times tc \times TC$$

onde P é o fator 16 ou 256 do Prescaler, tc é o período do clock do sistema e TC é a constante de tempo programada pela CPU.

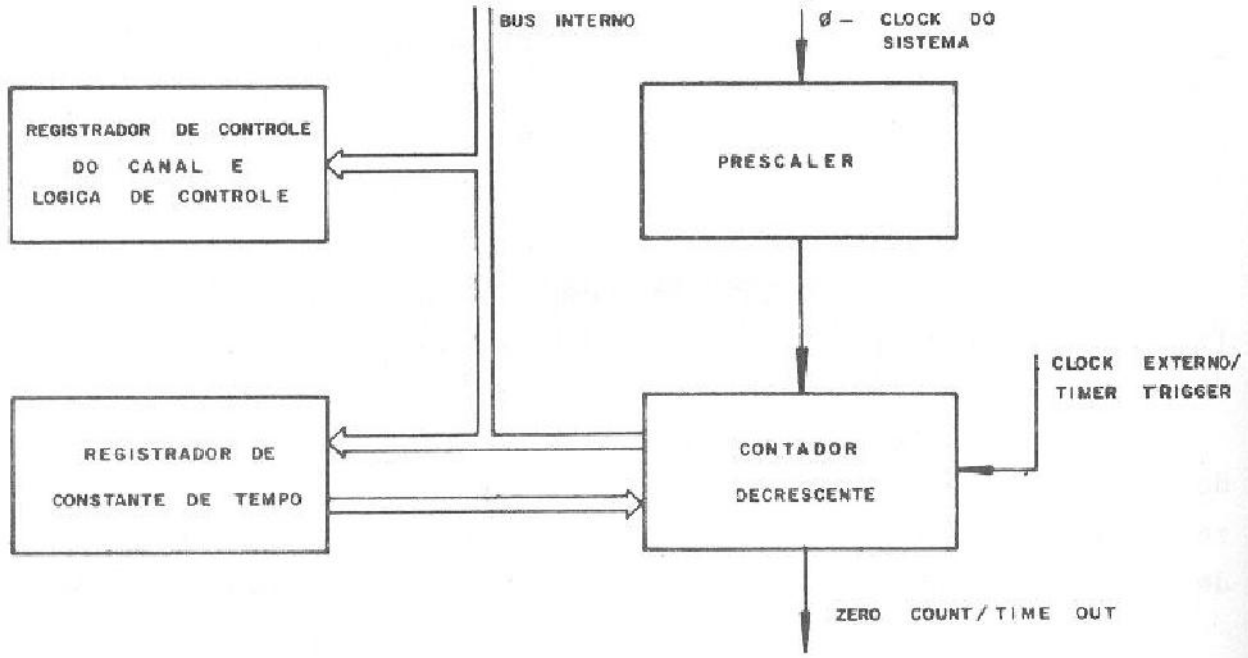


Fig. 5.6 – Arquitetura Interna do canal do modo Timer.

5.4.2 Counter Mode

O CTC, no modo contador tem por função básica contar as bordas de sua entrada CLK/TRG (External Clock/Timer Trigger).

A seleção deste modo de operação é realizada fazendo-se o bit 6 (Mode) igual a um na palavra de controle do canal.

O bit 4 (Slope) determina se o sinal clock externo irá disparar a operação de decrementação na borda de subida ou na borda de descida.

Aqui também, quando o Down Counter atinge o valor zero em sua decrementação, teremos um pulso de nível alto na saída Zero Count/Time Out (ZC/TO) do canal correspondente. E se

a interrupção estiver habilitada através do bit 7 (Interrupt Enable), a condição de se ter atingido zero causará uma requisição de interrupção.

Vale ressaltar aqui, o que foi visto no item 5.3.4. Uma nova constante de tempo só será carregada no Down Counter, quando estiver terminado sua operação.

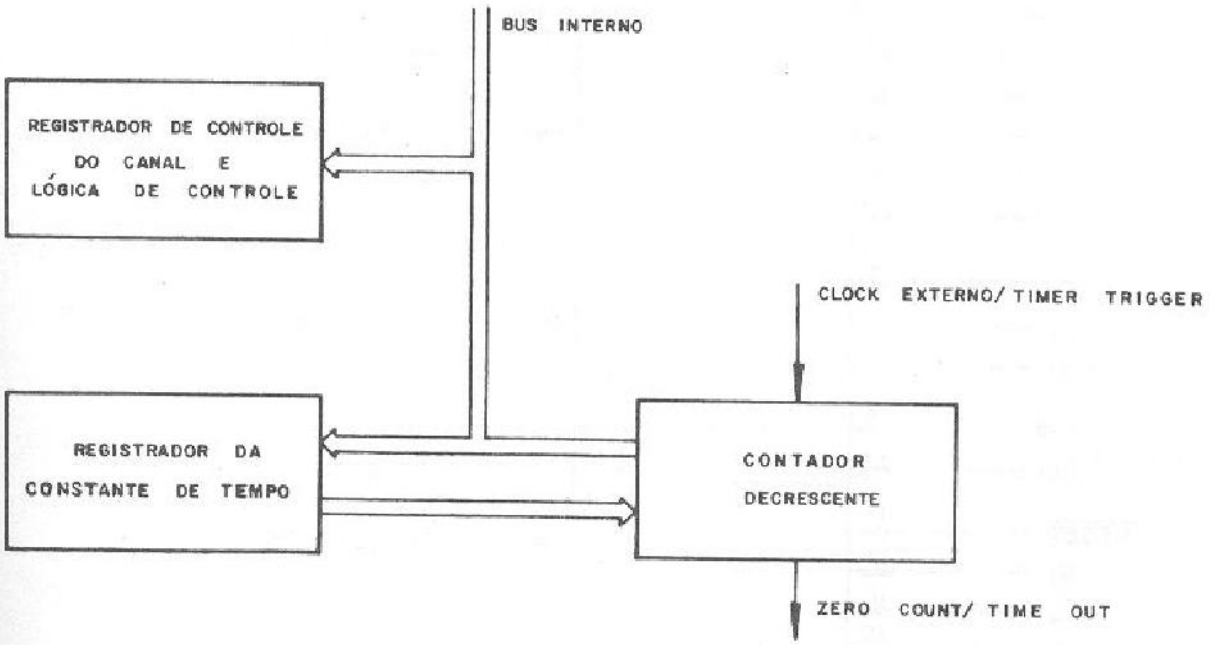


Fig. 5.7 - Arquitetura interna do canal no modo contador.

5.5 – DESCRIÇÃO DA PINAGEM

A pinagem do CTC é vista na fig. 5.8.

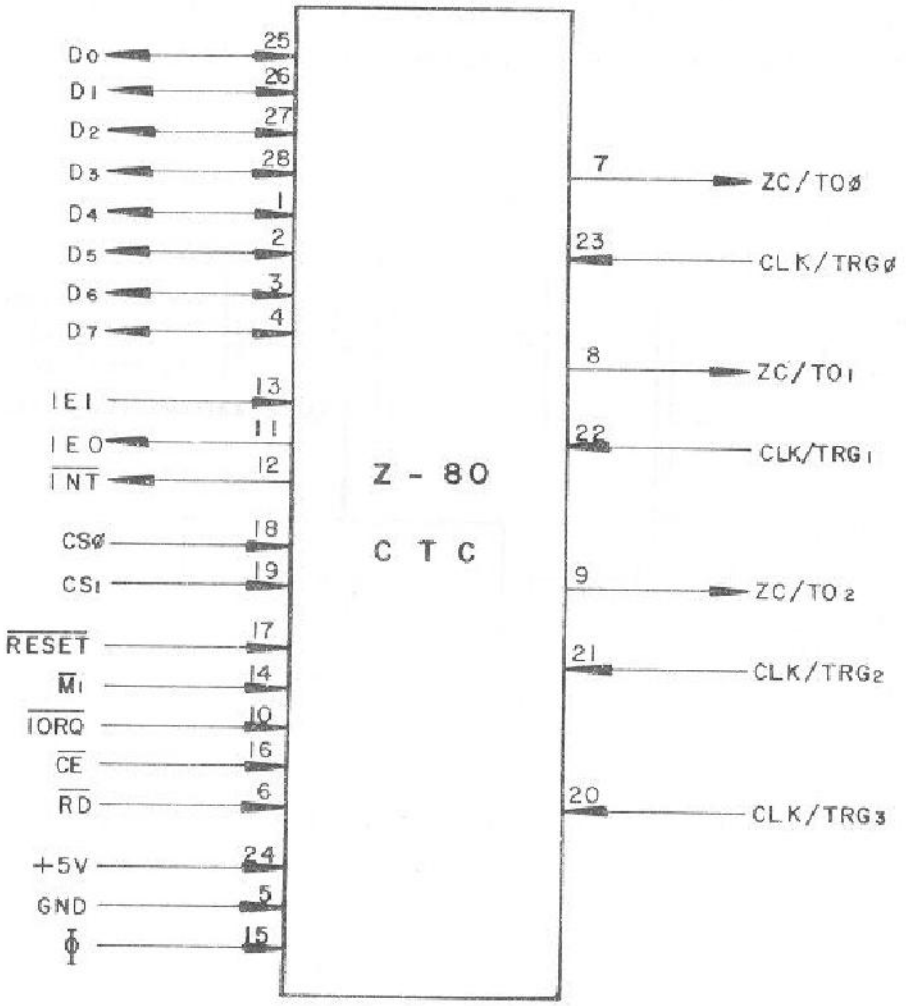


Fig. 5.8 – Pinagem do Z-80 CTC.

$D_Ø$ - D_7 (Entrada/saída)

Data Bus: Linha de dados usada entre a CPU e o CTC para comunicação de dados e palavras de comando.

IEI (Entrada)

Interrupt Enable In: Esta entrada é usada para compor o sistema de prioridades de interrupções. Quando esta entrada estiver em nível lógico "um", indica que não existe outro dispositivo de maior prioridade sendo atendido pela CPU.

IEO (Saída)

Interrupt Enable Out: Esta saída é usada em conjunto com o sinal IEI para gerar as prioridades de interrupções do sistema. Quando a CPU não está atendendo a interrupção de nenhum canal do CTC e a entrada IEI estiver em nível lógico "um", teremos a saída IEO em nível lógico "um".

$\overline{INT}$ (Saída)

Interrupt: Quando qualquer canal do CTC atingir zero, e tiver sido programado para pedir interrupção, isto fará com que esta saída vá a zero.

CSØ - CS1 (Entrada)

Chip Select: Seleciona o canal a ser lido ou programado, segundo tabela mencionada anteriormente, através das linhas de endereço $A_Ø$ e A_1.

$\overline{RESET}$ (Entrada)

Reset: Um sinal "zero" nesta entrada faz com que todos os contadores do CTC sejam resetados.

$\overline{M_1}$ (Entrada)

Machine Cycle One: Sinal vindo da CPU que junto com $\overline{IORQ}$, ambos ativados, dão a indicação que a CPU reconheceu uma interrupção, informando que a CTC requerente deve colocar no Data Bus o vetor interrupção.

$\overline{IORQ}$ (Entrada)

Input/Output Request: Sinal vindo da CPU, usado em conjunto com $\overline{CE}$, $\overline{RD}$ para transfeir dados e palavras de controle entre a CPU e o CTC. Durante o ciclo de escrita no CTC, $\overline{IORQ}$, $\overline{CE}$ = 0 e $\overline{RD}$ = 1, o CTC não recebe um sinal específico de escrita, contudo internamente ele inverte o sinal $\overline{RD}$, gerando um sinal WR. Durante um ciclo de leitura temos $\overline{IORQ}$, $\overline{CE}$ e $\overline{RD}$ = 0 para colocar o conteúdo do Down Counter no data bus.

$\overline{CE}$ (Entrada)

Chip Enable: Sinal que habilita o CTC. Nestas condições o CTC aceita dados, constantes de tempo, palavras de controle e vetor de interrupção vindos da CPU durante $\overline{IOWR}$ (Operação de Escrita em Periféricos) ou envia o conteúdo dos contadores para a CPU durante $\overline{IORD}$ (Operação de Leitura em Periféricos).

$\overline{RD}$ (Entrada)

Memory Read: Este sinal em conjunto com $\overline{IORQ}$ e $\overline{CE}$ transfere dados e palavras de controle do canal entre CPU e CTC. Durante o ciclo de escrita no CTC, $\overline{IORQ}$ e $\overline{CE}$ devem estar em zero e $\overline{RD}$ em nível lógico "um". O CTC não recebe um sinal específico de escrita entretanto este sinal é gerado internamente com o inverso de $\overline{RD}$. No ciclo de leitura $\overline{IORQ}$, $\overline{CE}$ e $\overline{RD}$ devem estar ativos em zero para colocar o conteúdo do contador no Data Bus.

Clock (Entrada)

Clock: Sinal de clock de fase única, utilizado pelo CTC para sincronizar internamente alguns sinais.

CLK/TRG Ø - CLK/TRG 3 (Entrada)

External Clock/Timer Trigger: No modo contador decrementa o Down Counter a cada subida ou descida do clock, conforme a programação feita, e no modo Timer, essa transição inicia a contagem (Timer).

ZC/TO Ø - ZC/TO 2 (Saída)

Zero Count/Time Out: Correspondem às saídas dos canais Ø, 1 e 2 do CTC tanto no modo contador como no modo Timer, o contador decrescente é decrementado até atingir a contagem zero, aparecendo então um pulso positivo nesta saída.

CAPÍTULO 6 – INTERFACE PARALELA DOS DISPOSITIVOS I/O – PIO

6.1 – INTRODUÇÃO

A PIO é uma interface paralela programável pela CPU utilizada para interfacear um grande número de periféricos como impressoras, leitoras de cartão e outros sem ajuda de circuitos externos. A PIO possue dois dispositivos internos chamados de portos, através dos quais a tarefa de interfacear a CPU com dispositivos periféricos é realizada. Este dispositivo como o nome diz, funciona no sistema paralelo, ou seja, todos os bits de dados são transmitidos ou recebidos ao mesmo tempo através do Bus de Dados.

A PIO apresenta algumas características importantes, como veremos a seguir, sendo que dentre elas, uma se destaca; é a lógica de Controle de Interrupção. Esta lógica permite um completo controle de interrupção, mesmo durante as operações de transferência de dados entre a CPU e dispositivos periféricos, sem se fazer necessário circuitos ou lógica externos. Além disso, dependendo de uma pré-programação, a PIO poderá interromper a CPU se condições especiais pré-fixadas ocorrerem no dispositivo periférico.

Damos a seguir, algumas características da PIO:

- Todas as entradas e saídas da PIO são compatíveis com TTL.

- Sua alimentação é de apenas 5 volts sendo usado uma única fase de Clock, comum ao sistema, para sincronismo interno.

- Possui dois portos totalmente independentes, com 8 linhas bidirecionais cada um, para a interface entre CPU e periféricos.

- Possui um controle de transferência de dados com linhas "HANDSHAKE", ou seja, controlando fluxo de dados em ambos os sentidos simultaneamente.

- Possui também, uma lógica de interrupção em série interligada (DAYSY CHAIN), onde os dispositivos são colocados em série de acordo com sua prioridade.

- A PIO pode ser programada para operar em 4 modos diferentes, que são:

MODO Ø - BYTE DE SAÍDA
MODO 1 - BYTE DE ENTRADA
MODO 2 - BYTE BIDIRECIONAL
MODO 3 - BIT DE CONTROLE

6.2 – ARQUITETURA GERAL DA PIO

A arquitetura da Interface Paralela para I/O é composta basicamente por: dois portos (A e B) com suas linhas "HANDSHAKE" e Bus de Dados e Controle, uma lógica de controle interno que regula o fluxo de informação vindo da CPU e dos portos que farão interface com os dispositivos periféricos, uma lógica de Controle de Interrupção que permite uma perfeita estrutura para interrupções e finalmente, uma interface para o bus de dados da CPU que permite a transferência direta de dados para esta.

O diagrama em blocos da arquitetura geral pode ser visto na figura 6.1.

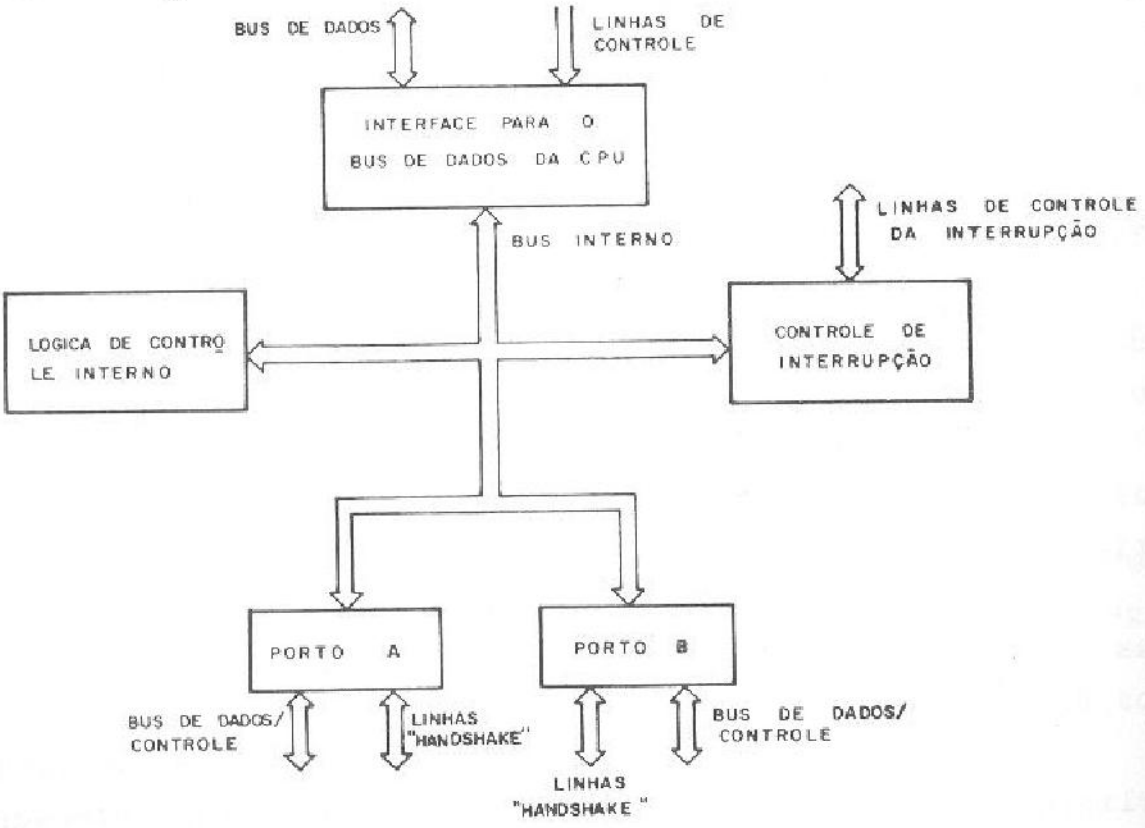


Fig. 6.1 - Arquitetura Interna da PIO.

6.3 – ARQUITETURA INTERNA DO PORTO

O porto é o dispositivo principal dentro da PIO, pois é através dele que é feita a interface direta com os dispositivos periféricos. Ele é composto por um registrador de 2 bits para controle da máscara, um resitrador máscara de 8 bits, um registrador para controle do modo de 2 bits, um registrador de seleção entrada/saída de 8 bits, um registrador de entrada de dados de 8 bits e um registrador de saída de dados de 8 bits. Possui também uma lógica de controle "HANDSHAKE".

O diagrama em blocos da arquitetura interna é visto na figura 6.2.

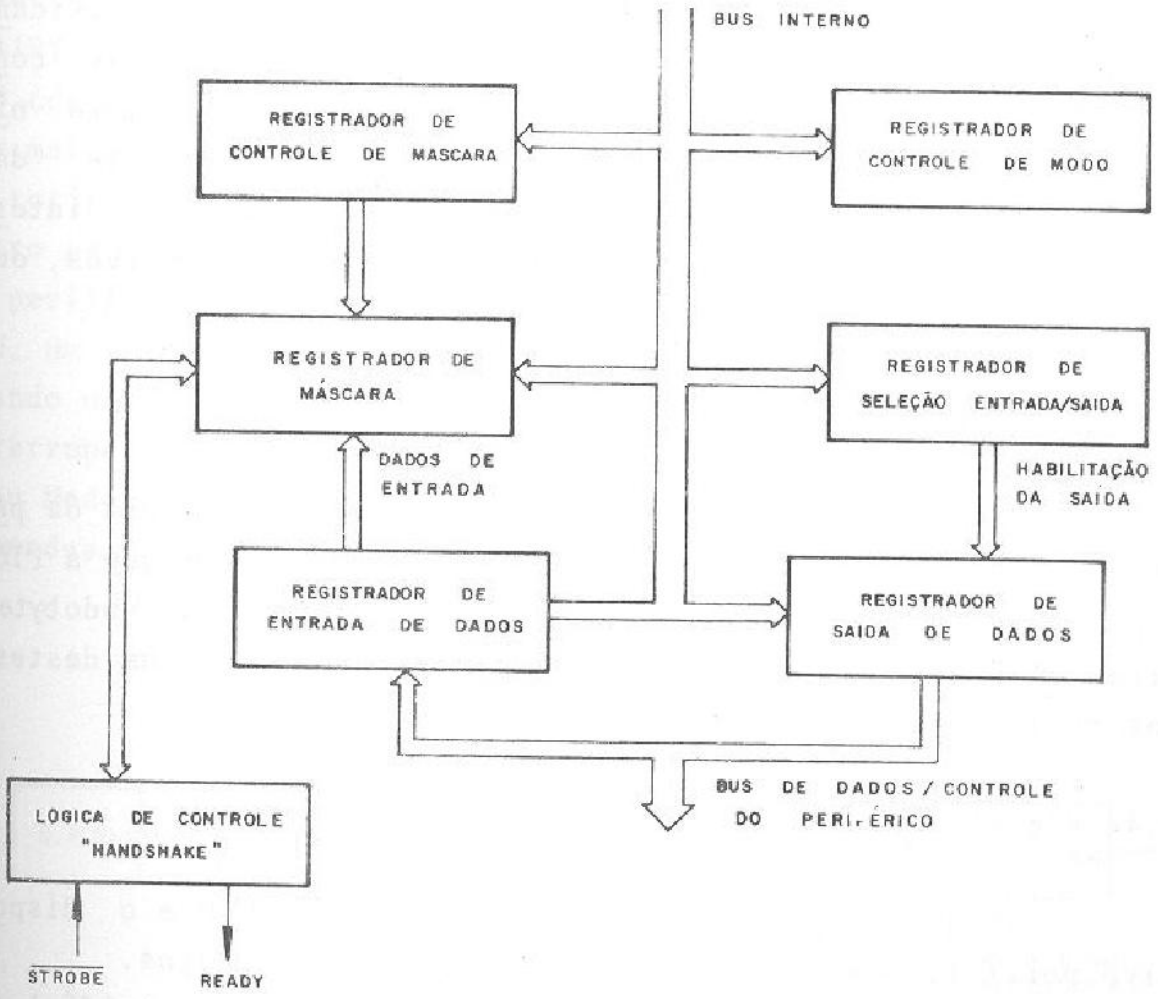


Fig. 6.2 – Arquitetura Interna do Porto.

6.3.1 Registrador de Seleção Entrada/Saída

Este registrador só é usado no Modo 3 ou no Modo Bit de controle. É através dele que cada um dos 8 bits do bus de dados/ controle são programados para atuarem como entrada ou saída.

6.3.2 Registrador Máscara/Registrador Controle de Máscara

O registrador máscara também só é usado no Modo 3 juntamente com a interrupção que tem sua ocorrência relacionada com condições especiais do periférico. Se associarmos a cada bit do bus de dados uma condição de ocorrência especial no periférico (como por exemplo, queda de força), podemos gerar interrupções analisando apenas uma condição, ou seja, verificando-se um bit não mascarado, ou se todos os bits (todas as condições) atingiram um nível especificado (zero ou um). Este nível "zero" ou "um" é determinado pelo registrador de controle da máscara que também indica em qual modo será gerada a interrupção, se no modo "AND" quando todos os bits forem ativos, ou se no modo "OR" quando um bit estiver ativo.

6.3.3 Registrador de Controle do Modo

Este registrador é programado pela CPU através da palavra de controle, de maneira a selecionar o modo em que a PIO irá operar, Modo byte de entrada, Modo byte de saída, Modobyte bidirecional e Modo Bit de controle, sendo que cada um destes modos serão melhor estudados mais adiante.

6.3.4 Registrador de Entrada/Registrador de Saída

Toda a transferência de dados entre a CPU e o dispositivo periférico é feita através destes registradores.

Quando a CPU quiser enviar um dado para o periférico, ela o escreverá no registrador de saída e quando o periférico enviar um dado para a CPU, ela o lerá no registrador de entrada. Toda essa transferência é fiscalizada apenas pela linha

"HANDSHAKE".

6.3.5 Lógica de Controle de Interrupção

Esta lógica determina toda a estrutura de interrupção no sistema CPU - periférico.

Quando uma interrupção é requisitada por um dispositivo e aceita, este deve fornecer o vetor de interrupção para que o endereço indireto da rotina de interrupção seja formado. O vetor nos fornece os 8 bits menos significativos desse endereço sendo que a parte mais significativa cabe ao registrador I da CPU.

Cada porto possui seu próprio vetor de interrupção, o qual tem seu bit menos significativo sempre em zero.

A prioridade de interrupção é determinada pelo posicionamento físico do dispositivo na configuração série interligada (DAISY CHAIN), onde todos eles estão ligados em série junto a CPU como mostra a figura 6.3. O dispositivo mais próximo a CPU tem maior prioridade. Se ele estiver sendo atendido pela CPU numa rotina de interrupção, não poderá ser interrompido por um outro pedido de um dispositivo de menor prioridade. Contudo, se um periférico de menor prioridade estiver sendo atendido pela CPU, um outro de maior prioridade poderá interromper o serviço. Quando um periférico requisita uma transferência de dados, uma interrupção será gerada se o PIO estiver operando em Modo Ø, Modo 1 ou Modo 2. Caso contrário, só será gerada quando condições programadas forem atingidas.

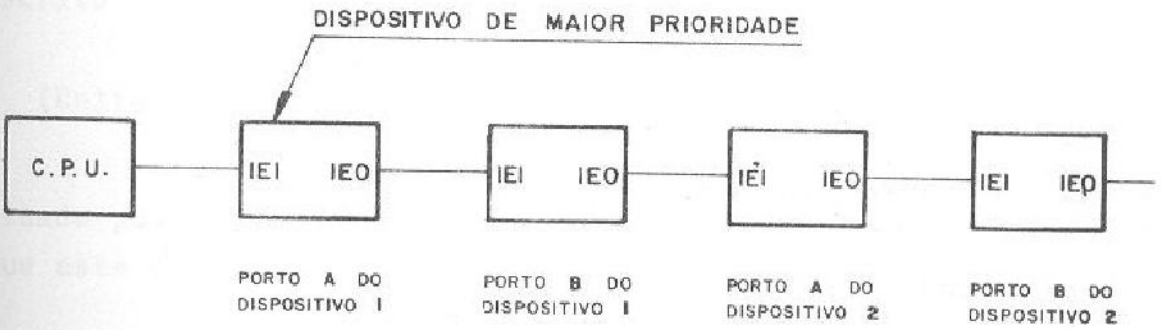


Fig. 6.3 - Configuração DAISY CHAIN.

6.4 – DESCRIÇÃO DA PINAGEM

A PIO se apresenta num chip de 40 pinos como vistos na figura 6.4, sendo que a seguir temos a descrição de cada pino.

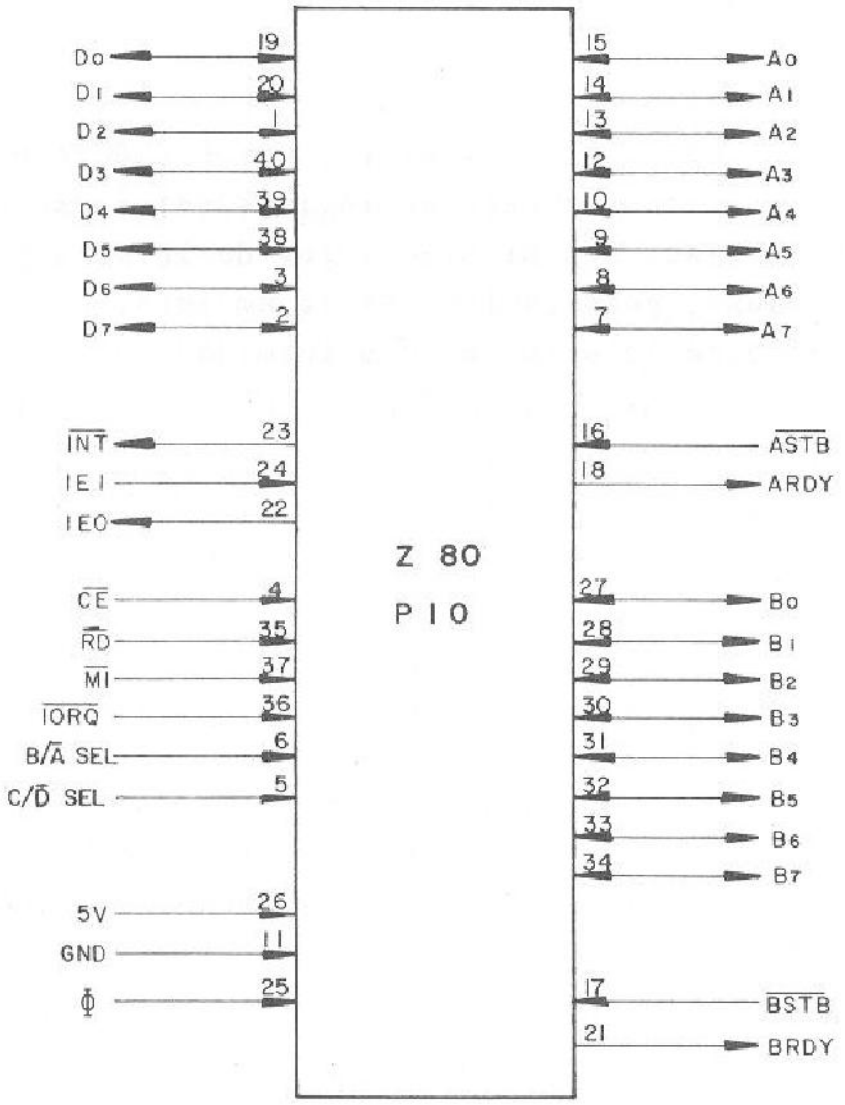


Fig. 6.4 – Pinagem da Z-80 PIO.

D_7 - D_0 (Entrada/Saída)

DATA BUS: O bus de dados é formado por 8 linhas bidirecionais com características de TRI-STATE ou seja, podem ser colocadas no estado de alta impedância. Estas linhas são usa

das na transferência de dados e comandos entre a CPU e a PIO.

$\overline{INT}$ (Saída)

INTERRUPT REQUEST: O pedido de interrupção da CPU pela PIO é feito através desta linha. Isto ocorre quando temos esta saída ativada.

IEI (Entrada)

INTERRUPT ENABLE IN: Quando em um sistema existe mais de um dispositivo de interrupção, este sinal é usado na estruturação do esquema de prioridade de interrupção em série interligada (DAISY CHAIN). Ele indica, quando ativado, que na série a qual está ligado não existe outro dispositivo utilizando a rotina de interrupção da CPU.

IEO (Saída)

INTERRUPT ENABLE OUT: Este sinal é usado juntamente com o anterior para a estruturação da prioridade. É ativo quando a PIO a qual pertence não está interrompendo a CPU, servindo como obstrução a dispositivos de menor prioridade no instante em que um de maior prioridade está sendo atendido.

$\overline{CE}$ (Entrada)

CHIP ENABLE: Este sinal é utilizado para habilitar a pastilha, de modo a aceitar dados ou comandos. Quando habilitada, a CPU pode colocar dados na PIO através de um ciclo de escrita ou pode retirar dados através de um ciclo de leitura.

$\emptyset$ (Entrada)

SISTEM CLOCK: A PIO usa o mesmo clock do sistema utilizado pela CPU para sincronizar seus sinais internos, sendo que este clock é de uma única fase.

$\overline{RD}$ (Entrada)

READ CYCLE STATUS: Este sinal indica se a operação que está sendo executada é uma "Leitura da Memória" ou uma

"Leitura em Dispositivo I/O", e em conjunto com $\overline{CE}$, $\overline{IORQ}$, C/$\overline{D}$ SELECT E B/$\overline{A}$ SELECT, é usado para transferir dados da PIO para CPU.

$\overline{M_1}$ (Entrada)

MACHINE CYCLE ON: Este sinal é fornecido pela CPU sendo usado como um pulso de sincronismo para vários controles internos da PIO, especialmente a lógica de controle da interrupção.

Quando $\overline{M_1}$ ocorre e $\overline{RD}$ ou $\overline{IORQ}$ estão inativos a PIO é colocada no estado inicial, ou seja, é dado um RESET.

A busca de uma nova instrução na memória pela CPU é feita quando temos ativos os sinais $\overline{M_1}$ e $\overline{RD}$.

O reconhecimento de uma interrupção também é feito através do sinal $\overline{M_1}$ só que em conjunto com o sinal $\overline{IORQ}$.

$\overline{IORQ}$ (Entrada)

INPUT/OUTPUT REQUEST: Para que ocorra a transferência de dados ou comandos entre a CPU e a PIO, é preciso que este sinal seja combinado com o $\overline{CE}$, $\overline{RD}$, B/$\overline{A}$ SELECT e C/$\overline{D}$ SELECT.

O porto selecionado pelo sinal B/$\overline{A}$ SELECT, enviará dados para a CPU somente quando os sinais $\overline{CE}$, $\overline{RD}$ e $\overline{IORQ}$ estiverem ativos. Do mesmo modo, a CPU irá enviar dados ou comandos (conforme seleção feita pelo sinal C/$\overline{D}$ SELECT) para o porto selecionado se os sinais $\overline{CE}$ e $\overline{IORQ}$ estiverem ativos e $\overline{RD}$ não.

B/$\overline{A}$ SEL (Entrada)

PORT A/B SELECT: Este sinal determina qual porto será acessado durante a transferência de dados entre a CPU e a PIO. Um nível baixo seleciona o porto A, enquanto um nível alto seleciona o porto B.

C/$\overline{D}$ SEL (Entrada)

DATA/CONTROL SELECT: Este sinal define como será a transferência entre a CPU e a PIO, se será de dados ou controle . Um nível baixo neste pino indica que o bus de dados será usado para transferir dados, enquanto um nível alto indica que

o bus de dados será usado para transferir comandos.

A_0 - A_7 (Entrada)

PORT A BUS: Este conjunto de 8 linhas bidirecionais, é usado para transferir dados e controle entre o porto A e o dispositivo periférico.

$\overline{ASTB}$ (Entrada)

PORT A STROBE PULSE FROM PERIPHERAL DEVICE: Este sinal é usado para identificar as operações a serem realizadas, sendo que dependem do modo selecionado. No modo Ø (saída), este sinal é usado para reconhecimento de que os dados foram recebidos pela CPU. No Modo 1 (entrada), é usado pelo periférico como indicação de que seus dados devem ser carregados no registrador de entrada do porto A.

No Modo 2 (bidirecional), é usado para colocar os dados contidos no registrador de saída do porto A no bus de dados bidirecionais. A borda de subida do STROBE reconhece a recepção de dados.

No Modo 3 (Bit de Controle), o sinal é inibido internamente.

ARDY (Saída)

REGISTER A READY: Este sinal depende do modo de operação programado.

No Modo Ø (saída), este sinal indica quando ativo, que o registrador de saída do porto A está carregado e que o bus de dados está pronto para transferir dados.

No Modo 1 (entrada), indica que o registrador de entrada do porto A está desocupado e pronto a aceitar dados.

No Modo 2 (bidirecional), indica que dados estão disponíveis no registrador de saída do porto A para serem transferidos para o periférico, sendo que isto ocorre somente quando o sinal $\overline{ASTB}$, estiver ativo.

No Modo 3 (Bit de Controle), este sinal é forçado a zero.

$B_{\emptyset}$ - B_7 (Entrada/Saída)

PORT B BUS: Este conjunto de 8 linhas bidirecionais é usado para transferir dados e/ou controles, entre o porto B e o dispositivo periférico.

$\overline{BSTB}$ (Entrada)

PORT B STROBE PULSE PERIPHERAL DEVICE: Este sinal é semelhante ao $\overline{ASTB}$ sendo que aqui se trata do porto B.

BRDY (Saída)

REGISTER B READY: Este sinal é semelhante ao ARDY sendo que aqui se trata do porto B.

Embora a PIO não tenha um pino específico, ela é resetada toda a vez que a ligamos. Com isto teremos os registradores de saída resetados assim como os registradores de máscara e dispositivos para habilitação de interrupções dos portos. Além disso teremos todo o bus de dados em TRI-STATE (alta impedância) sendo automaticamente selecionado o Modo 1 de operação.

A PIO sairá do estado de RESET tão logo receba a palavra de controle da CPU.

6.5 – MODOS DE OPERAÇÃO

A PIO pode operar em 4 Modos distintos: Modo Ø (saída), Modo 1 (entrada), Modo 2 (Bidirecional) e Modo 3 (controle). Cada modo de operação é determinado pela palavra de controle escrita na PIO pela CPU, a qual tem a forma mostrada na figura 6.5.

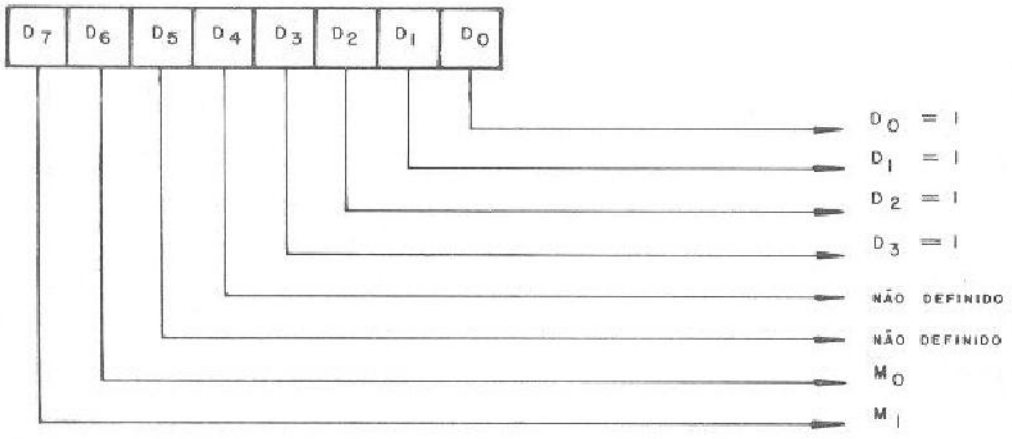


Fig. 6.5 - Palavra de Controle.

Os modos são fornecidos pela combinação dos bits D_7 e D_6 da seguinte forma:

MODO	D_7	D_6
Ø - Saída	Ø	Ø
1 - Entrada	Ø	1
2 - Bidirecional	1	Ø
3 - Controle	1	1

Sendo que os bits D_5 e D_4 não são definidos e os bits D_3 a $D_Ø$ devem ser todos iguais a "UM" para indicar que a palavra se refere ao modo de operação.

A porto A opera em qualquer modo, e o porto B em todos menos o Modo 2.

MODO Ø:

Neste modo, dados podem ser escritos no registrador de saída do porto pela CPU, através da habilitação do bus de dados do respectivo porto. O conteúdo deste registrador pode ser lido ou até trocado pela CPU a qualquer instante.

Após os dados serem escritos, o sinal READY é acionado a fim de avisar ao periférico que existem dados disponíveis. Este sinal será desativado quando o periférico ler os dados enviando um sinal STROBE.

MODO 1:

Neste Modo, colocamos o porto no modo de entrada; assim, o porto ativa o sinal READY indicando ao periférico que o registrador de entrada está vazio e que os dados devem nele ser carregados. O periférico ao carregar tais dados envia um sinal de STROBE que desativa automaticamente o sinal READY.

Após esta operação, a CPU obtém os dados do periférico efetuando uma simples operação de leitura no registrador de entrada do porto.

MODO 2:

Somente o porto A opera em Modo 2. Isto porque neste modo temos transferência de dados entre a PIO e periféricos em ambos os sentidos, para isso usando as 4 linhas "HANDSHAKE". Os sinais do porto B são usados para controlar a entrada de dados, enquanto os sinais do porto A controlam a saída de dados, sendo que a única diferença entre esta parte da operação e o Modo Ø é que o conteúdo do registrador de saída do porto A só é colocado no bus de dados quando o sinal $\overline{ASTB}$ for ativado.

MODO 3:

Este modo só é usado em atividades de controle onde cada bit do bus de dados é programado para atuar como entrada ou saída. Isto é feito por uma palavra de controle como a da figura 6.6 enviada logo após a palavra de controle que selecionou o Modo 3 de operação.

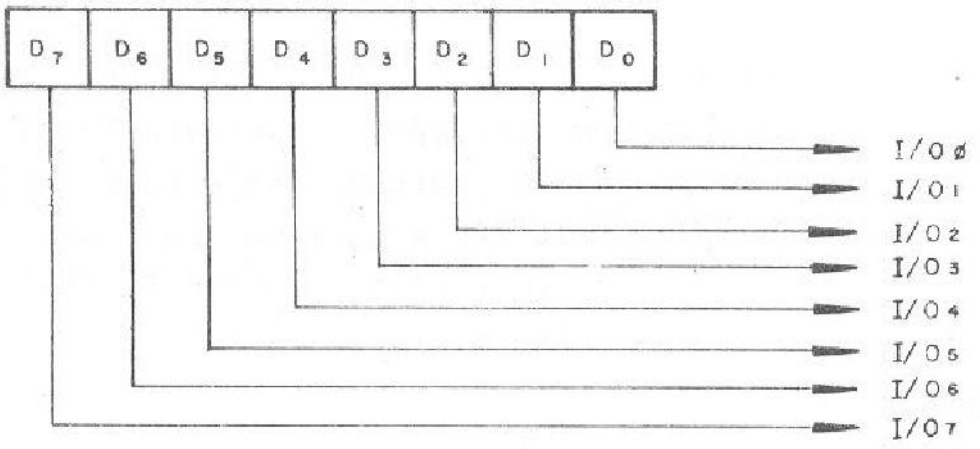


Fig. 6.6. – Palavra de Controle que define o Bus de Dados como Entrada ou Saída.

Se um bit for colocado em "um" a correspondente linha do bus funcionará como entrada, e se for colocado em "zero" funcionará como saída.

6.6 – PALAVRAS DE COMANDO

6.6.1 Palavra de Controle

Como vimos no ítem anterior, a palavra controle defi

ne o modo de operação, e para se distinguir das demais possui os 4 bits menos significativos todos em "um", como mostra a figura 6.7.

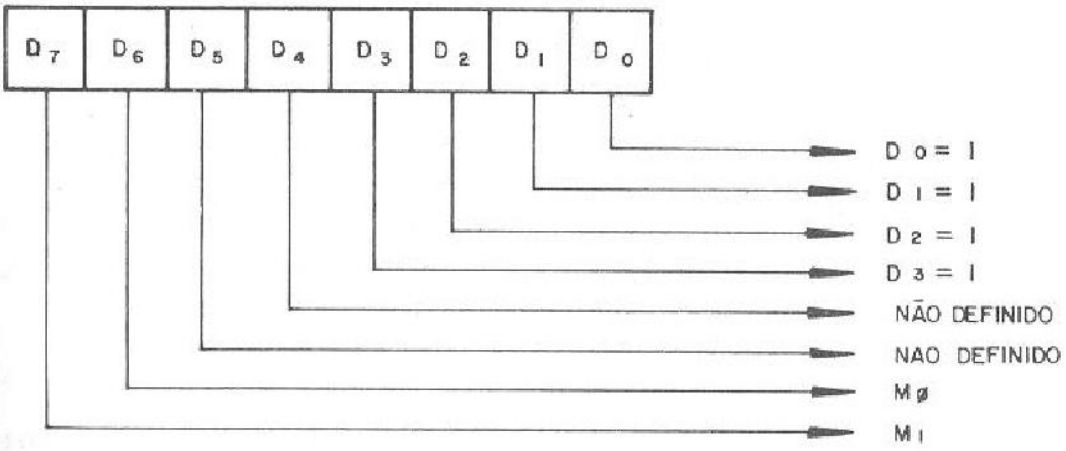


Fig. 6.7 - Palavra de Controle.

O modo é definido pela combinação dos bits D_7 e D_6.

6.6.2 Vetor de Interrupção

Na operação entre CPU e PIO, esta pode ter sido programada para atuar em Modo 2, e o sistema de interrupção do porto precisa receber do dispositivo periférico que solicitou uma interrupção, um vetor de interrupção para formar o ponteiro que indicará o endereço da rotina de interrupção.

O dispositivo de maior prioridade que requisitar uma interrupção deverá colocar o vetor interrupção no bus de dados durante o ciclo em que a PIO reconhece o pedido.

O vetor é dado pela palavra de controle, que, então, tem a forma vista na figura 6.8, onde V_1 a V_7 são os bits menos significativos do endereço.

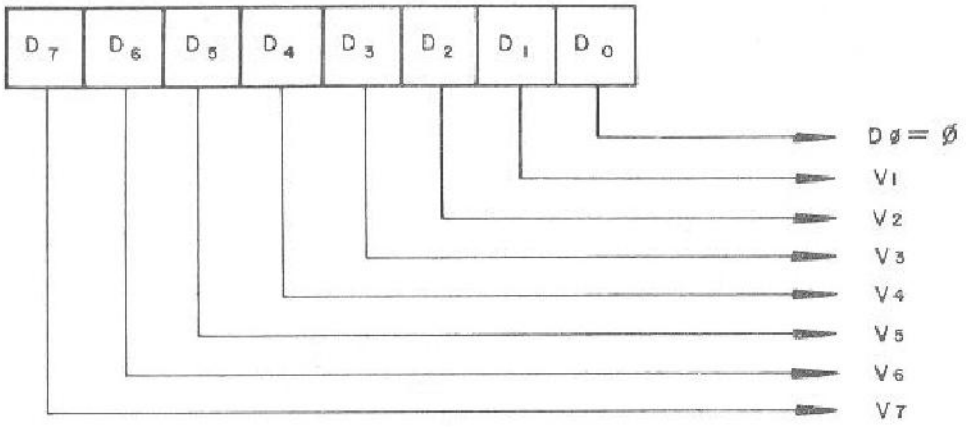


Fig. 6.8 - Vetor de Interrupção.

O bit D_0 indica que a palavra de controle é um vetor de interrupção quando colocado em "zero".

6.6.3 Palavra de Controle de Interrupção

A palavra de controle de interrupção para qualquer um dos portos tem a forma vista na figura 6.9.

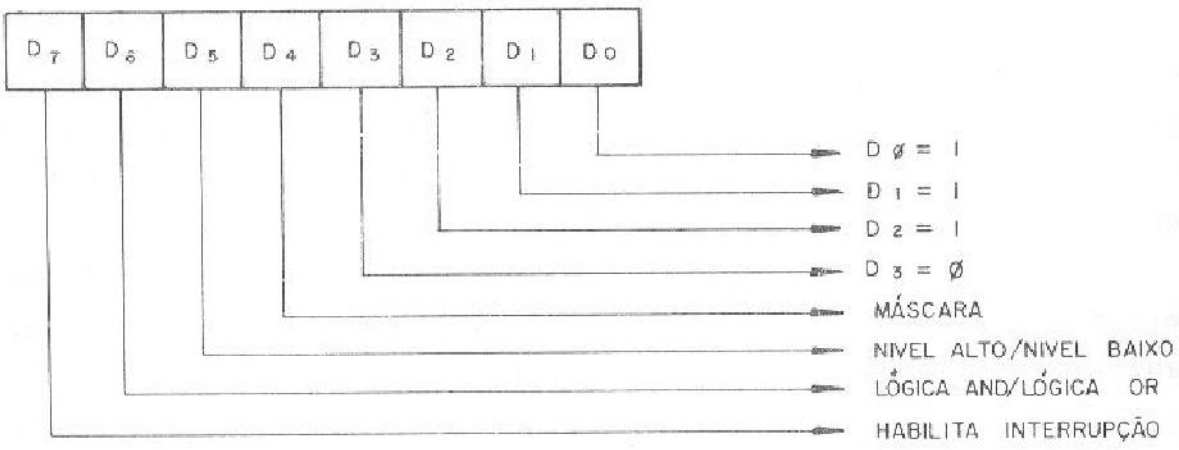


Fig. 6.9 - Palavra de Controle de Interrupção.

A configuração vista dos 4 bits menos significativos serve para indicar que a palavra de controle se refere a uma palavra de controle de interrupção.

D_7

Se for igual a "Zero", indica que o dispositivo utilizado internamente pela PIO para habilitar interrupção é colocado a zero, sendo que interrupções não poderão ser geradas. Qualquer pedido de interrupção neste caso, ficará guardado na PIO esperando habilitação.

Se for igual a "um", o dispositivo interno que habilita interrupções da PIO será colocado em "um" e interrupções poderão ser geradas pelo porto.

D_6

Se for igual a "zero", indica que durante uma interrupção relacionada com a ocorrência de condições especiais no periférico, será usada a Lógica "OR". Neste caso, se um bit for "um", será gerada uma interrupção.

Se for igual a "um", indica que será usada a lógica "AND" e neste caso, todos os bits deverão ser "um", antes da interrupção ser gerada.

D_5

Se for igual a "zero", indica que está sendo utilizada lógica negativa, e"zero"será reconhecido como nível de ativo.

Se for igual a "um", indica o uso de lógica positiva, neste caso o "um" é considerado como nível de ativo.

D_4

Este bit define que a próxima palavra que a PIO receberá fornecerá a máscara. Esta máscara nos dará quais os bits que serão controlados para gerar uma interrupção.

Esta máscara pode ser vista na figura 6.10.

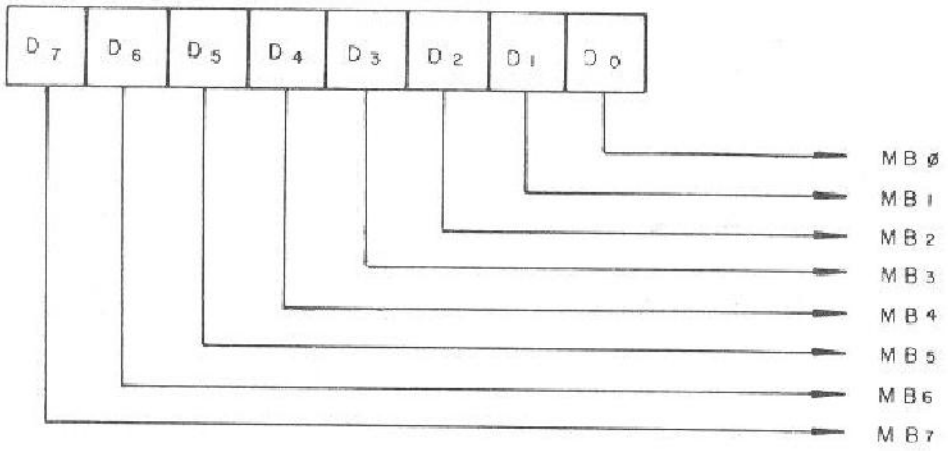


Fig. 6.10 - Máscara para controle dos bits que causarão interrupções.

CAPÍTULO 7 – MEMÓRIAS

7.1 – INTRODUÇÃO

A memória, em sistemas de computação a microcomputador, tem um papel muito importante, pois nela estão contidos dados e programas que serão usados pela CPU.

Em computadores de grande porte existe uma grande preocupação com os tipos de memória a serem utilizadas.

Existem memórias de grande capacidade chamadas de "armazenamento secundário", como fita magnética, FLOP DISK e outras, sendo que neste tipo de memória são guardados vários arquivos contendo programas e dados que podem ser chamados pelo usuário a qualquer momento.

Este tipo de memória é muito lenta em relação a CPU isto porque existe um compromisso sério entre tempo de acesso e capacidade, e nesta circunstância a CPU necessita de uma memória de acesso mais rápido, chamada de "principal", para trabalhar.

Deste modo, quando o usuário requer um arquivo, este é carregado do "armazenamento secundário" na memória "principal" de acesso rápido, aonde será executado o seu programa.

Uma nota importante é que em sistemas de grande capacidade, a transferência de dados entre "armazenamento secundário" e memória "principal" é feita através de grupo de dados em forma de "pacotes".

Outra vantagem das memórias de grande capacidade é que este tipo de memórias são não voláteis, isto é, se faltar energia elétrica ou simplesmente se desligar o sistema, as informações não se apagarão.

Desde os primórdios da computação já existiram vários tipos de memória como por exemplo: de núcleo de ferrite, memórias monolíticas e outras.

Veremos a seguir, alguns tipos de memórias existentes no mercado.

7.2 – MEMÓRIAS EPROM

As EPROMS são memórias não voláteis que podem ser programadas de acordo com a necessidade do usuário, e poderão ser apagadas e programadas apenas por um processo especial.

Para se apagar uma EPROM necessita-se de luz ultra violeta, sob a qual coloca-se a memória durante um período de 20 a 40 minutos.

Para se programar este tipo de dispositivo necessita-se conectar ao pino VPP uma tensão média de 25V sendo que o endereço da locação a ser gravada é aplicado ao bus de endereço; o dado a ser gravado é aplicado ao bus de dados do dispositivo (D_0 - D_7) e por último é aplicado um pulso negativo em torno de 50 ms ao pino $\overline{CE}$ se for o caso da EPROM 2732, ou ao pino $\overline{PGM}$ se for o caso da EPROM 2764. Na fig. 7.1 pode-se acompanhar o procedimento de gravação.

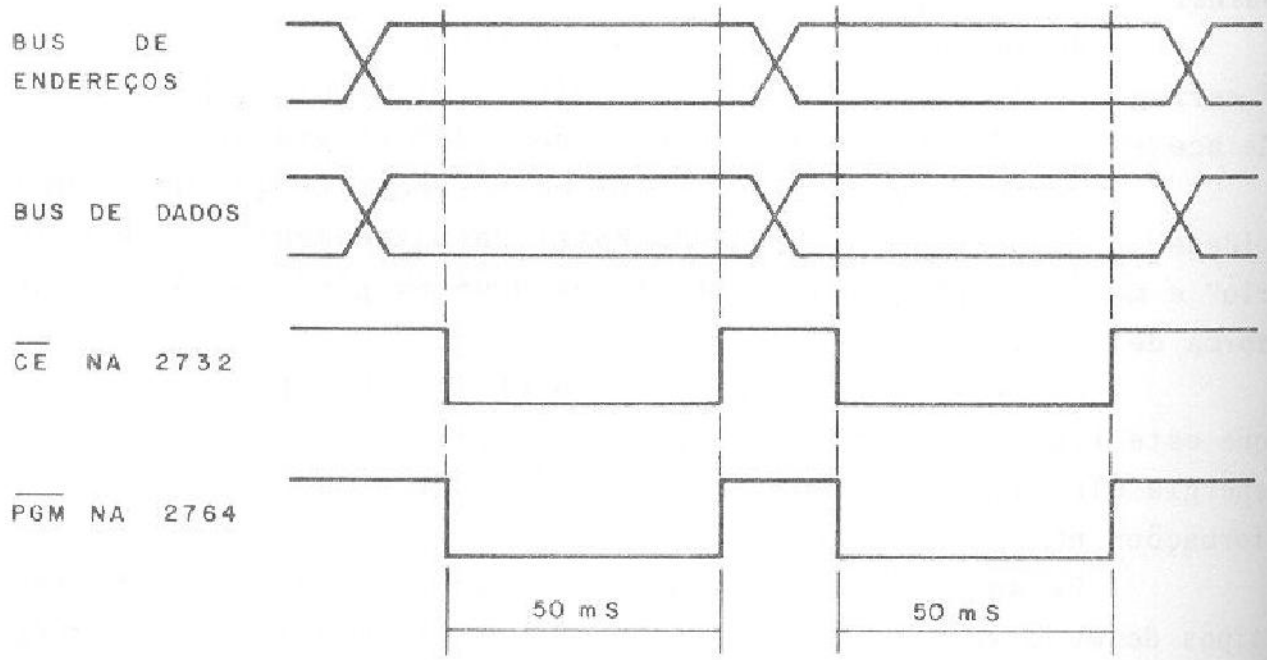


Fig. 7.1 - Ciclo de tempo de gravação de EPROM.

7.3 – ESTRUTURA DA EPROM 2732

7.3.1 Características

Capacidade - 4 K x 8 Bits
Alimentação - 5 Volts
Tempo de Acesso - 450 n^S máximo

7.3.2 Pinagem da 2732

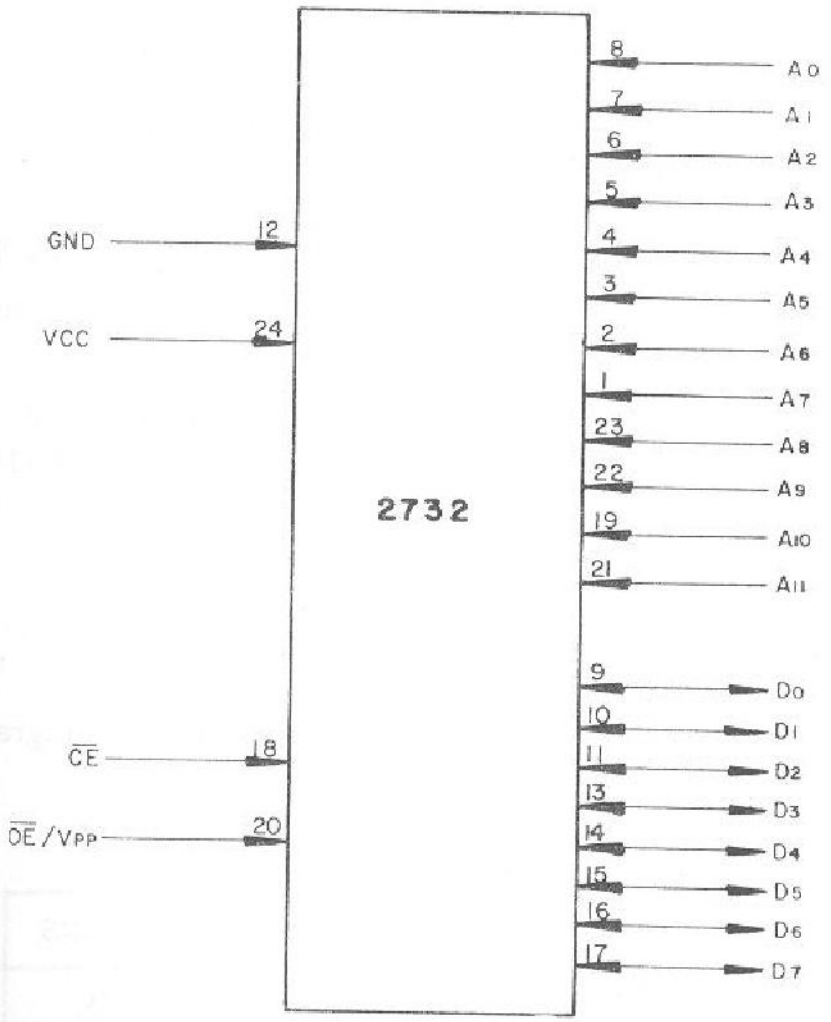


Fig. 7.2 - EPROM 2732.

V_{cc}, GND (Alimentação)

V_{cc} = 5 Volts ± 5%
GND = Referência "0" Volts

D_0 - D_7 (Entrada-Saída)

Bus de dados - Este Bus também é utilizado como entrada durante o processo de gravação.

A_{11} - A_0 (Entrada)

Bus de Endereço.

$\overline{CE}$ (Entrada)

Este sinal habilita a comunicação entre a CPU e a memória propriamente dita.

$\overline{OE}/V_{pp}$ (Entrada)

Este sinal habilita apenas a via de dados. Se $\overline{CE}$ estiver habilitado então a via de dados será habilitada como saída quando este sinal for ativado.

Este terminal também tem a função de durante a programação da 2732, servir de entrada de um nível de 25 Volts.

7.3.3 Resumo

A_0 - A_{11} - Entrada de endereço
$\overline{CE}$ - Habilita Chip
$\overline{OE}/Vpp$ - Habilita saída de dados/Tensão de programação
D_0 - D_7 - Entrada e saída de dados

MODO DE SELEÇÃO

	$\overline{CE}$	$\overline{OE}/V_{pp}$	LINHAS DE DADOS
Leitura	GND	GND	Saída - dados
Desabilita	V_{cc}	X	alta impedância
Programação	GND	V_{pp} = 25V	Entrada - dados
Verifica Programação	GND	GND	Saída - Dados
Desabilita Programação	V_{cc}	V_{pp} = 25V	Alta impedância

X = Irrelevante (V_{cc} ou GND)

7.4 – ESTRUTURA DA EPROM 2764

7.4.1 Características

A memória EPROM 2764 é totalmente compatível com a 2732 em relação a pinagem onde no pino 26 da 2764 nada consta e ao qual se conecta a V_{CC}, para uso simultâneo da 2732 e 2764 no mesmo circuito.

Capacidade	-	8 K x 8 Bits
Alimentação	-	5 Volts
Tempo de acesso	-	450 n^S máximo

7.4.2 Pinagem da 2764

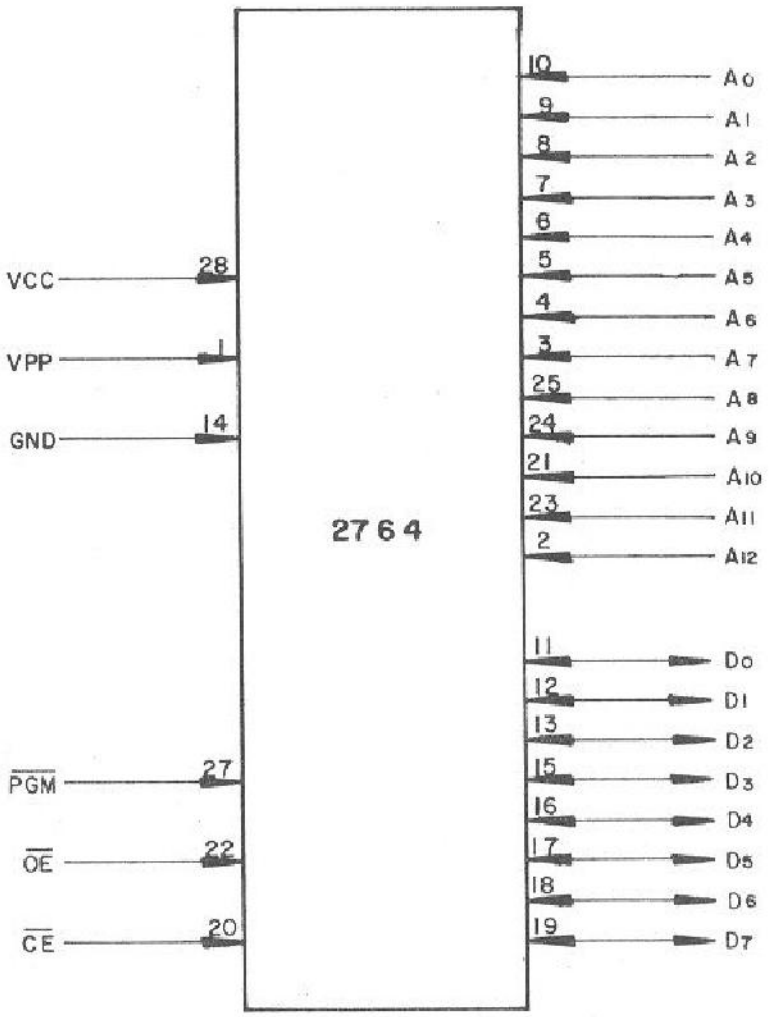


Fig. 7.3 - EPROM 2764.

V_{cc}, GND (Alimentação)

V_{cc} = 5 Volts $\pm$ 5%
GND = Referência "0" Volts

$A_{12} - A_0$ (Entrada)

Bus de endereço.

$D_7 - D_0$ (Saída - Entrada)

Bus de Dados - Também utilizado como entrada no modo de gravação.

$\overline{CE}$ (Entrada)

Este sinal habilita a comunicação entre a CPU e a memória propriamente dita.

$\overline{OE}$ (Entrada)

Este sinal habilita apenas a via de dados.
Se $\overline{CE}$ estiver habilitado então a via de dados também será habilitada como saída quando este sinal ocorrer.
Na 2764 o sinal $\overline{OE}$ apresenta apenas esta função.

$\overline{PGM}$ (Entrada)

Este sinal é usado apenas na programação, pois é através dele que será habilitada a programação.

V_{pp} (Entrada)

Tensão de programação: Durante a programação da 2764, deverá ser aplicado um nível de tensão igual a 25V neste pino e, na sua utilização normal uma tensão igual a 5V.

7.4.3 Resumo

$A_0 - A_{12}$ - Entrada de endereço
$\overline{CE}$ - Habilita Chip
$\overline{OE}$ - Habilita saída de dados
$D_0 - D_7$ - Entrada e saída de dados
$\overline{PGM}$ - Programação

MODO DE SELEÇÃO

	$\overline{CE}$	$\overline{OE}$	$\overline{PGM}$	V_{pp}	LINHAS DE SAÍDA
Leitura	GND	GND	V_{cc}	V_{cc}	Saída de dados
Desabilita	V_{cc}	X	X	V_{cc}	Alta impedância
Programação	GND	X	GND	V_{pp}	Entrada de dado
Verificação de Program.	GND	GND	V_{cc}	V_{pp}	Saída de dado
Desabilitação de Progr.	V_{cc}	X	X	V_{pp}	Alta impedância

X = Irrelevante (V_{cc} ou GND)

7.5 – ESTRUTURA DA 4801 e 4802

7.5.1 Introdução

Neste item apresentaremos duas novas memórias RAM estáticas, a 4801 e 4802, produzidas pela MOSTEK CORPORATION.

A capacidade das memórias acima apresentadas são respectivamente de 1 K x 8 bits e 2 K x 8 bits. Este tipo de memória vem substituir as 2114 com grandes vantagens e, a maior delas é que uma única pastilha tem capacidade de 1 K x 8 bits. Outra grande vantagem é a facilidade na elaboração do Lay-Out do circuito. Outra característica vantajosa é que apenas dois terminais da 4801 e 4802 diferem da 2716 e 2732 respectivamente.

7.5.2 RAM 4801

7.5.2.1 Características

Capacidade de armazenamento	-	1 K x 8 Bits
Tempo de acesso	-	4801 A - 55 → 55 n seg
		4801 A - 70 → 70 n seg
		4801 A - 99 → 90 n seg
Ciclo de Leitura e Escrita	-	4801 A - 55 → 55/65 n seg
		4801 A - 70 → 70/80 n seg
		4801 A - 99 → 90/100 n seg
Alimentação	-	5 V

7.5.2.2 Pinagem da 4801

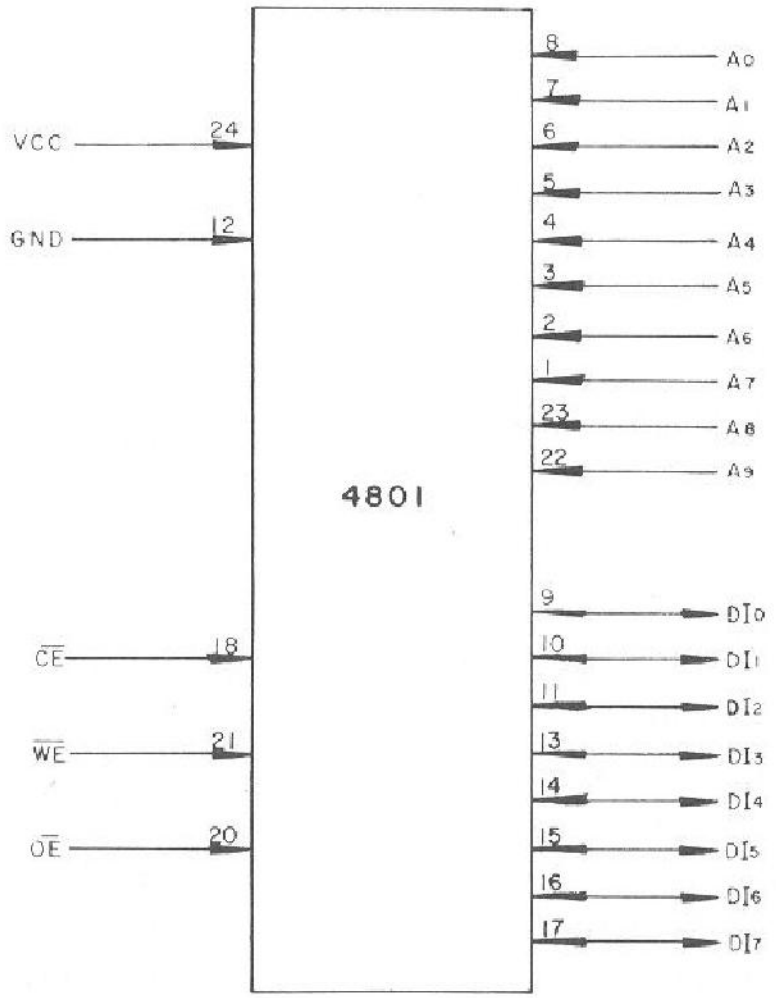


Fig. 7.4 - RAM Estática 4801.

V_{cc}, GND (Alimentação)

V_{cc} = 5 Volts $\pm$ 5%

GND = Referência "0" Volts

A_9 - A_0 (Entrada)

Bus de endereços.

DI_7 $DI_\emptyset$ (Entrada / Saída)

Bus de dados.

$\overline{CE}$ (Entrada)

Este sinal habilita a comunicação entre a CPU e a memória propriamente dita, sendo ativo em nível lógico "zero".

$\overline{OE}$ (Entrada)

Este sinal habilita apenas a via de dados. Se $\overline{CE}$ estiver habilitado, então a via de dados será habilitada como saída quando este sinal ocorrer, sendo que poderá ser usado como $\overline{RD}$, sendo ativado em nível lógico "zero".

$\overline{WE}$ (Entrada)

Este sinal é usado para que a informação possa ser escrita na memória, sendo ativado em nível lógico "zero".

Resumo:

A_0 - A_9	-	Entrada de Endereços
$\overline{CE}$	-	Habilitação de Pastilha
$\overline{WE}$	-	Habilitação de Escrita
$\overline{OE}$	-	Habilitação de Leitura
DI_0 - DI_7	-	Entrada e Saída de Dados

MODO DE SELEÇÃO

	$\overline{CE}$	$\overline{OE}$	$\overline{WE}$	LINHAS DIN/DOUT
Escrita	GND	X	GND	DIN
Leitura	GND	GND	V_{cc}	DOUT
Desabilita	V_{cc}	X	X	Alta Impedância

X = Irrelevante (V_{cc} ou GND)

7.5.3 RAM 4802

7.5.3.1 Características

Capacidade de armazenamento - 2 K x 8 Bits

Tempo de acesso - 4802 - 70 → 70 n seg
4802 - 90 → 90 n seg

Ciclo de Leitura e Escrita - 4802 - 70 → 70/80 n seg
4802 - 90 → 90/100 n seg

7.5.3.2 Pinagem da 4802

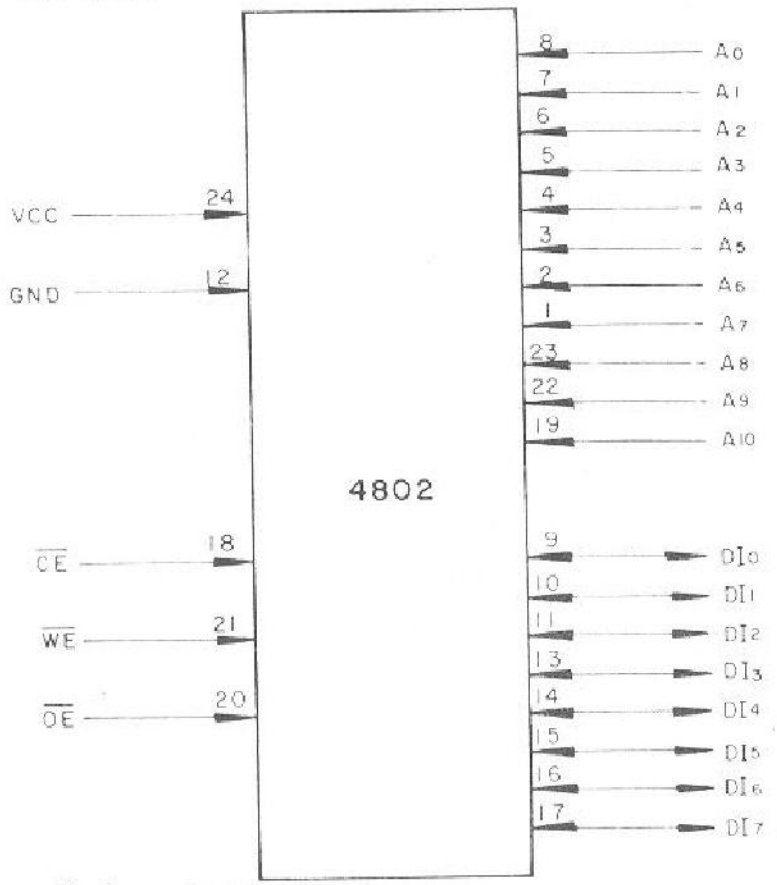


Fig. 7.5 RAM Estática 4802.

V_{CC}, GND (Alimentação)

V_{CC} = 5 Volts $\pm$ 5%
GND = Referência "0" Volts

A_{10} - A_0 (Entrada)

Bus de endereço.

DI_7 , $DI_\emptyset$ (Entrada/Saída)

Bus de Dados.

$\overline{CE}$ (Entrada)

Este sinal habilita a comunicação entre a CPU e a memória propriamente dita, sendo ativado em nível lógico"zero".

$\overline{OE}$ (Entrada)

Este sinal habilita apenas a via de dados como saída se $\overline{CE}$ estiver habilitada. Ele poderá ser usado como $\overline{RD}$, sendo ativado em nível lógico"zero".

$\overline{WE}$ (Entrada)

Este sinal é usado para que a informação possa ser escrita na memória, sendo ativado em nível lógico"zero".

Resumo

A_0 - A_{10}	-	Entrada de Endereço
$\overline{CE}$	-	Habilitação da Pastilha
$\overline{WE}$	-	Habilitação de Escrita
$\overline{OE}$	-	Habilitação de Leitura
$DI_\emptyset$ - DI_7	-	Entrada e Saída de Dados

MODO DE SELEÇÃO

	$\overline{CE}$	$\overline{OE}$	$\overline{WE}$	LINHAS DIN/DOUT
Escrita	GND	X	GND	DOUT
Leitura	GND	V_{cc}	V_{cc}	DIN
Desabilita	V_{cc}	X	X	Alta Impedância

X = Irrelevante (V_{cc} ou GND)

CAPÍTULO 8 – ESQUEMA GERAL DE UM MICROCOMPUTADOR

8.1 – INTRODUÇÃO

O principal intuito deste capítulo é a aplicação dos circuitos e dispositivos, estudados e analisados nos capítulos anteriores.

O projeto desenvolvido apresentará a construção minuciosa do HARDWARE de um microcomputador, que terá os seguintes aspectos: CPU Z-80, PIO Z-80, CTC Z-80, RAM 4802 e EPROM 2732; com as seguintes funções: teclado Hexa Decimal, conjunto de Display e gravador de EPROM.

Como é de conhecimento de todos, a aplicação de microcomputadores está divergindo em todos os sentidos, e no nosso caso este Kit se propõe a controlar qualquer tipo de maquinário, terminal de vídeo, ou memórias de grande capacidade como Flop Disk, por exemplo.

Um exemplo típico seria o controle da velocidade de um motor, pois ao se aplicar uma carga em um motor a sua velocidade tende a diminuir, e com o auxílio do microcomputador conseguiremos manter a velocidade constante, fazendo leituras da velocidade, analisando e atuando na própria velocidade do motor se preciso, como poderemos ver na figura 8.1.

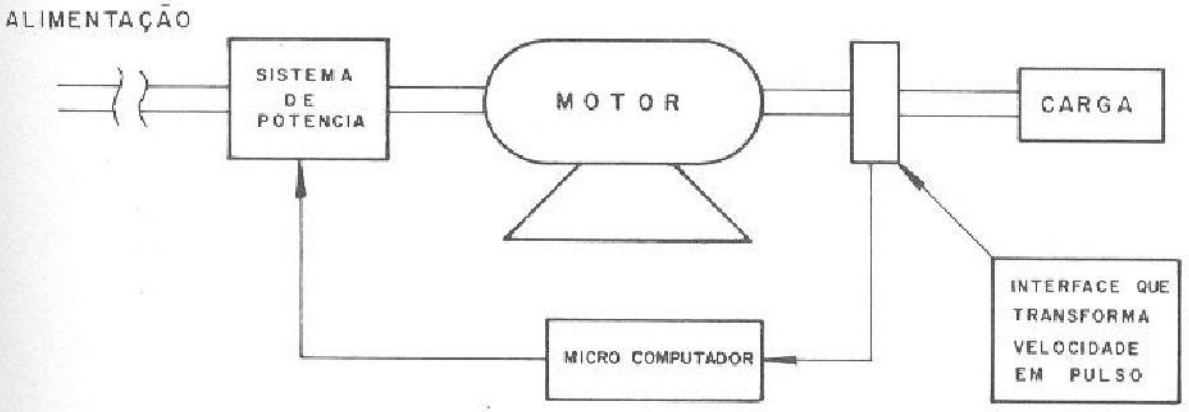


Fig. 8.1 - Aplicação de um microcomputador no controle de velocidade de um motor.

8.2 – CHAVE RESET E "POC"

A CPU ou qualquer outro sistema de pequeno porte requer um dispositivo que forneça o chamado RESET. Este sinal é simplesmente uma condição para que o sistema comece a processar seus arquivos do início.

Em um microprocessador como o Z-80, sempre que pressionada a chave de RESET., impomos um nível lógico igual a "zero" ao pino de $\overline{RST}$ da CPU, conseguindo desta forma com que o contador de programa (PC) assuma o valor (ØØØØ) H que é a locação inicial do programa monitor na memória, que no caso será a EPROM.

É de grande ajuda que, ao ligar a fonte de alimentação do microcomputador, tal operação se realize automaticamente, e isto é conseguido através do sistema POC - POWER ON CLEAR - ou seja "limpa ao ligar".

Isto ocorre por causa do capacitor de 47μ F, que se encontra descarregado, isto é, a nível lógico "zero" e que se carregará através do resistor de 10 KΩ.

Este capacitor permanecerá carregado, a menos que se pressione a chave RESET ou que se desligue a fonte. Neste último caso, o capacitor se descarregará pelo diodo, para que na próxima vez que se ligar a fonte de alimentação a carga do capacitor seja nula de modo que se possa ter o RESET automático. O circuito é visto na figura 8.2, onde os inversores (7404) são usados para aumentar o nível de corrente do sinal.

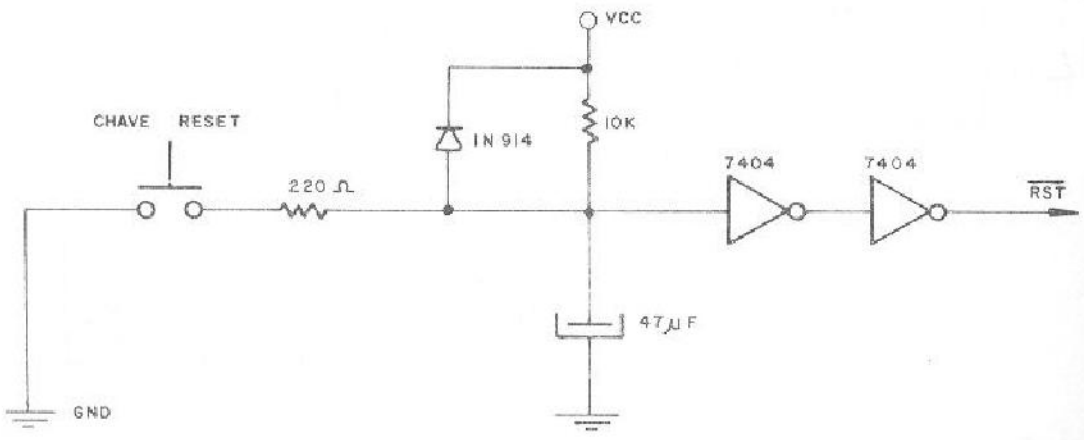


Fig. 8.2 - Chave RESET e Circuito POC.

8.3 – MAPEAMENTO DE MEMÓRIAS

O assunto de que trataremos neste capítulo, será o da implantação e associação de memórias aplicadas no projeto do microcomputador.

O primeiro passo é definir o quanto de memória iremos usar e em que área a localizaremos.

No nosso caso teremos 8K x 8 bits de memória EPROM e 8K x 8 bits de memória RAM, localizadas nos primeiros 16k do programa, ou em outras palavras, do endereço ØØØØ H à 3FFF H, e o restante das locações de memórias ficarão vagos, sendo reservados a futuras expansões.

A segunda etapa será a definição da capacidade individual de cada memória.

Neste projeto, as memórias serão divididas em duas classes: memória EPROM 2732 com 4K x 8 bits e a RAM 4802 de 2K x 8 bits.

A terceira etapa será a construção do mapa das linhas de endereço, que neste caso em especial será feito de 2K em 2K bytes. Neste mapa E_1 corresponde à primeira EPROM 2732, E_2 à segunda EPROM 2732; R_1 à primeira RAM 4802 e assim sucessivamente, como mostra a figura 8.3.

		A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	0000 H
E_1	2K	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	07FF H
	4K	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0FFF H
E_2	6K	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	17FF H
	8K	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF H
R_1	10K	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	27FF H
R_2	12K	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	2FFF H
R_3	14K	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	37FF H
R_4	16K	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF H

Fig. 8.3 - Mapa das Linhas de Endereço.

Como podemos notar, existem 8 grupos de linhas de endereço cada grupo correspondendo a uma memória. Isto quer dizer que necessitamos de um decodificador de oito posições de saída para selecionar cada grupo das linhas. Neste caso será usado o decodificador 74138, visto aqui na figura 8.4.

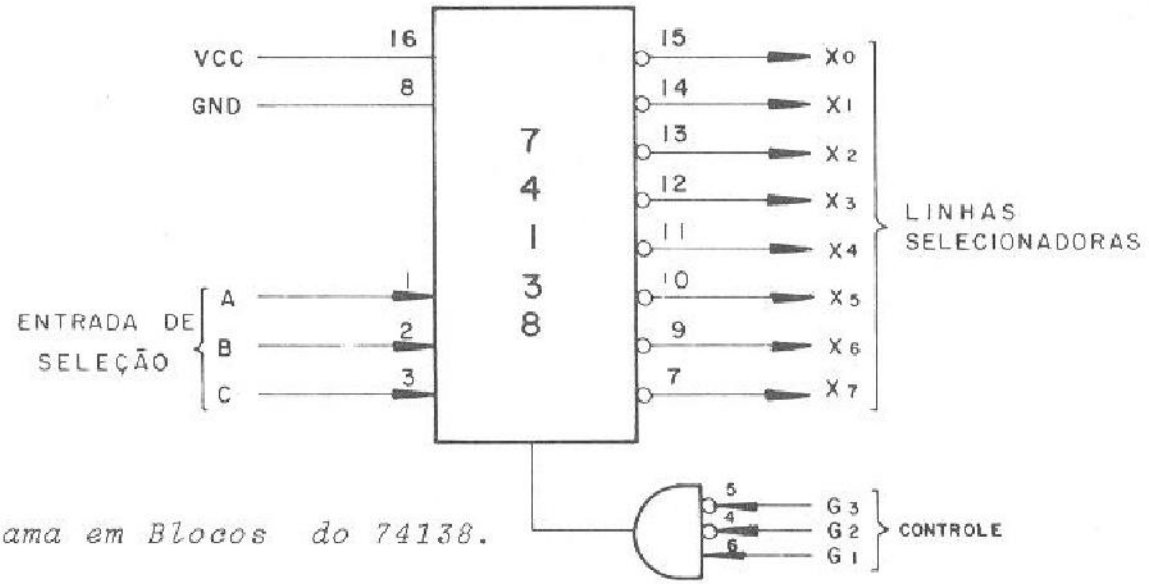


Diagrama em Blocos do 74138.

CONTROLE			ENTRADA DE SELEÇÃO			LINHAS SELECIONADORAS							
G_1	G_2	G_3	C	B	A	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7
X	1	1	X	X	X	1	1	1	1	1	1	1	1
0	X	X	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

Tabela do Decodificador 74138. X = Irrelevante

Fig. 8.4 - Esquema de Decodificação para as Memórias.

Este decodificador irá selecionar cada grupo de 2K bytes, sendo que esta seleção será controlada pelas linhas de endereço. Olhando para o mapa das linhas de endereço, notamos que as linhas A_{15} e A_{14} permaneceram sempre a nível lógico "zero", isto significa que para selecionar um dos grupos de 2K bytes, as linhas A_{15} e A_{14} deverão estar sempre a nível lógico "zero". Notemos agora que, na passagem de um grupo de 2K bytes para outro, apenas as linhas A_{13}, A_{12}, A_{11} se modificam; portanto, chegamos à conclusão de que as linhas A_{13}, A_{12}, A_{11}, serão as linhas decodificadoras. O decodificador que apresenta estes requisitos é o 74138, que no nosso caso será o 74LS 138 devido ao baixo consumo, o qual podemos ver na figura 8.5.

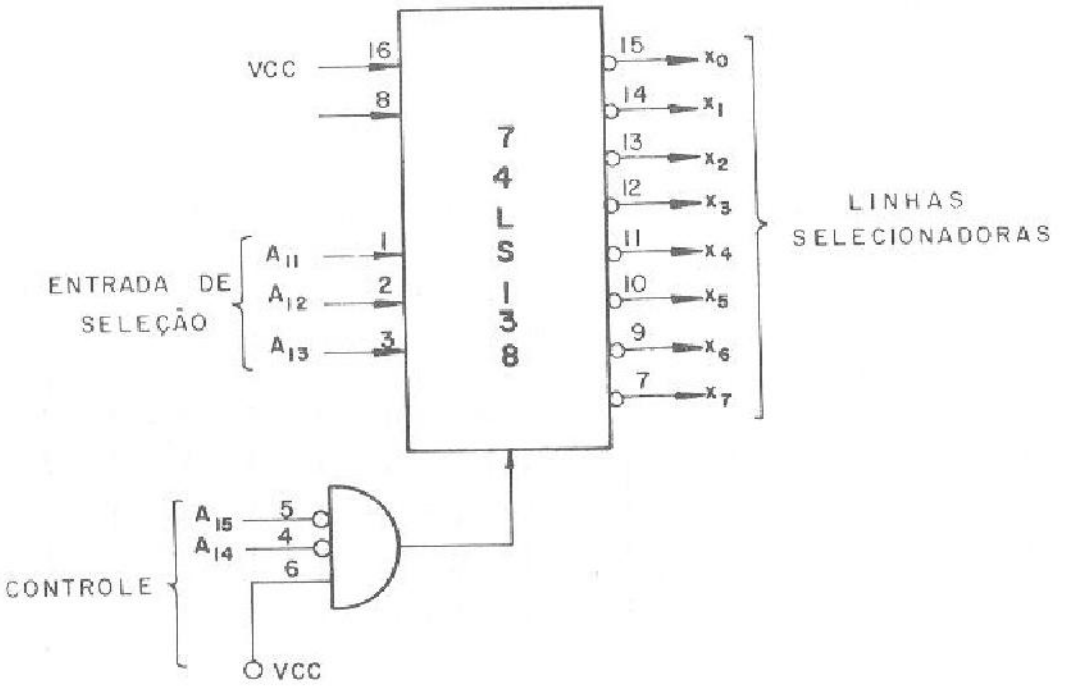


Fig. 8.5 - Esquema geral do 74LS 138 usado na seleção de memória.

Com a utilização da EPROM 2732 de 4K bytes de capacidade, encontramos uma incoerência, pois o decodificador seleciona blocos de 2K em 2K bytes.

Portanto teremos que agrupar a primeira com a segunda linha selecionadora e a terceira com a quarta. Ao se agrupar desta forma ficou evidente que as linhas selecionadoras X_0

ou X_1 e as linhas X_2 ou X_3 do decodificador irão selecionar os blocos E_1 e E_2 respectivamente. Como a própria condição está impondo, teremos que usar um bloco lógico "E" reunindo as linhas X_0 com X_1 e X_2 com X_3 respectivamente. Olhando para a figura 8.6, verificamos a implementação completa de 8 K bytes de EPROM e 8K bytes de RAM, indo do endereço 0000 H a 3 FFF H. O sinal $\overline{RD}$ é conectado às entradas $\overline{OE}$ das memórias para que apenas no ciclo de leitura as linhas de dados sejam habilitadas como saída.

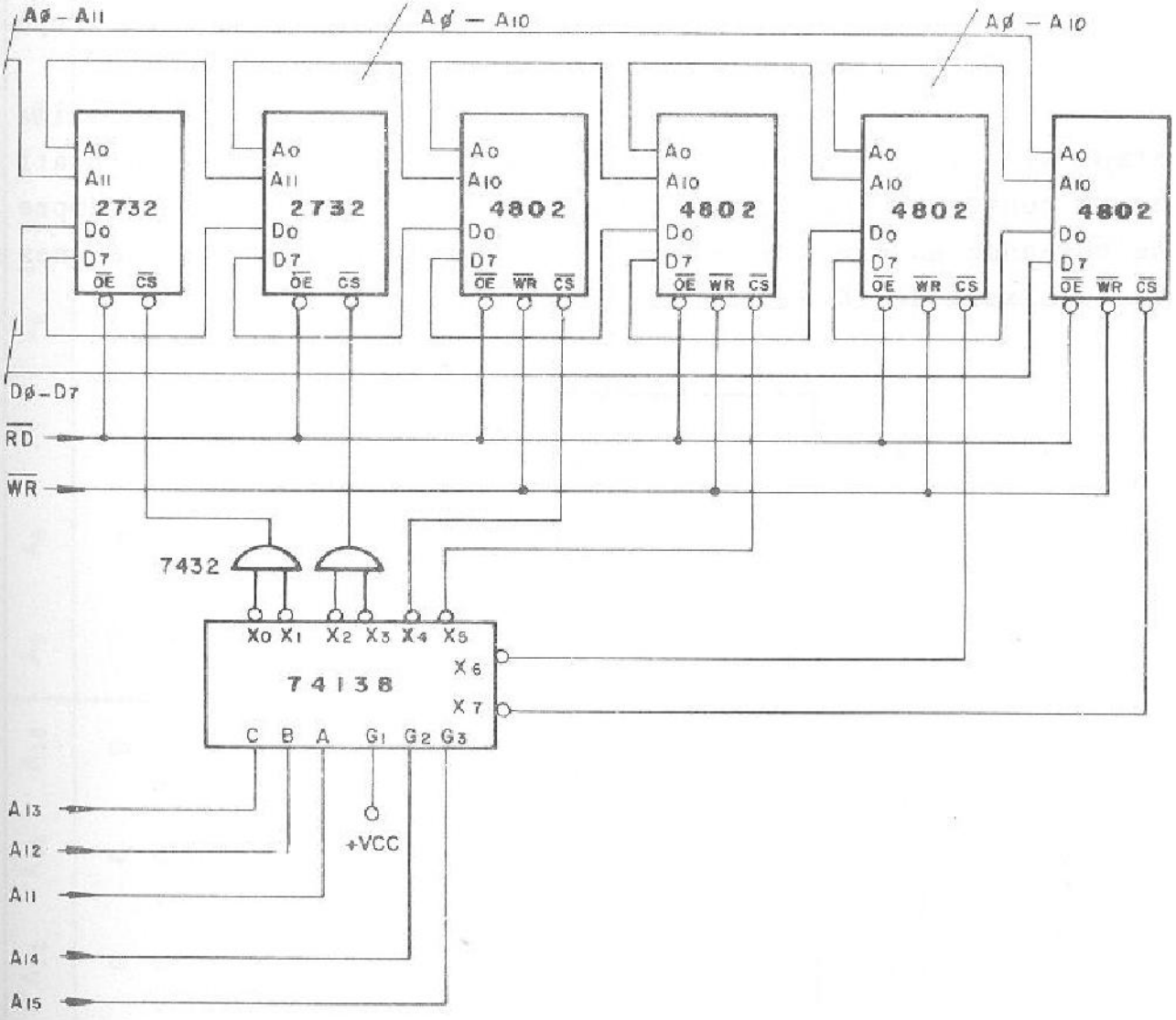


Fig. 8.6 - Implementação do Banco de Memórias.

Ainda dentro deste item, apresentaremos a seguinte implementação:

Suponhamos que encontremos no mercado um lote de 32 memórias RAMS 2114 e queiramos expandir a capacidade de memória a partir do endereço 4ØØØ H.

Este tipo de memória vista na figura 8.7, tem 1K x 4 bits de capacidade, e, como a via de dados do Z-80 tem 8 bits de dados, necessitamos de duas memórias associadas para formar 1K x 8 bits.

O lote das memórias é de 32 unidades, e necessitamos de duas memórias para formar 1K x 8 bits portanto com o grupo de 32 memórias 2114 teremos um total de 16K x 8 bits.

Definida a capacidade total das memórias, a próxima etapa será a construção do mapa de endereços. Uma regra prática na construção do mapa dos endereços é construí-los sempre se baseando na capacidade individual de cada memória, que neste caso será de 1K em 1K byte.

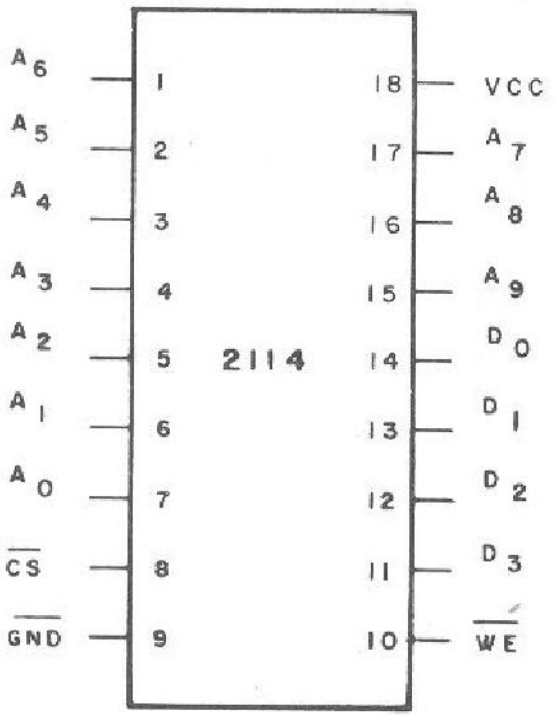


Fig. 8.7 - Memória RAM 2114.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	4000 H
0	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	43FF H
0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	47FF H
0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	4BFF H
0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	4FFF H
0	1	0	1	0	0	1	1	1	1	1	1	1	1	1	1	53FF H
0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	57FF H
0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	5BFF H
0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	5FFF H
0	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	63FF H
0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	67FF H
0	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	6BFF H
0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	6FFF H
0	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	73FF H
0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	77FF H
0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	7BFF H
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFF H

Fig. 8.8 - Mapas das Linhas de Endereço.

O mapa construído foi dividido de 1K em 1K bytes, e como podemos notar, existem 16 grupos. Nesta implementação necessitaremos de um decodificador de 16 linhas de selecionamento; portanto o decodificador usado será o 74154 esquematizado na figura 8.9.

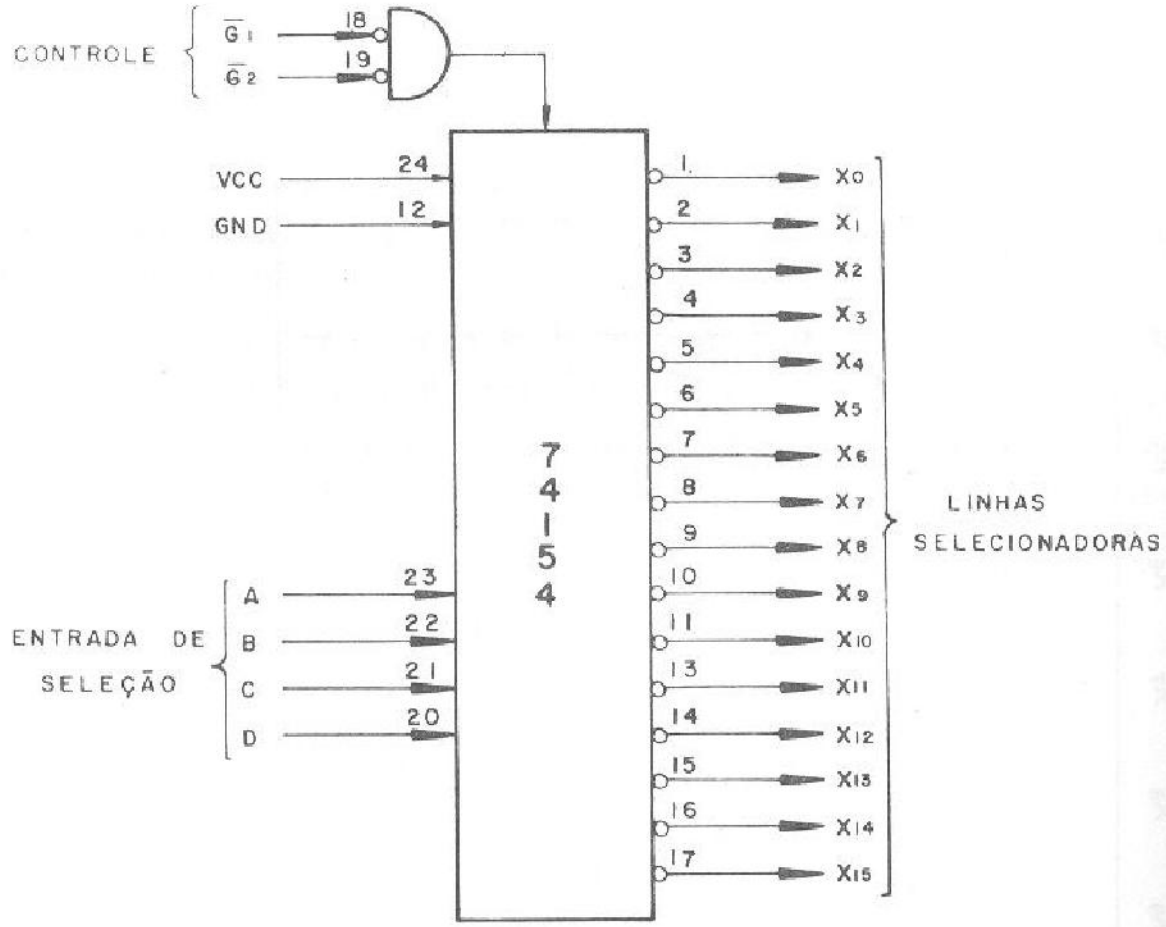


Diagrama em Blocos do 74154.

CONTROLE		ENTRADA DE SELEÇÃO				LINHAS SELECIONADORAS															
G_1	G_2	D	C	B	A	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
0	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Fig. 8.9 - Mapa de Decodificação usando o 74154.

Analizando o mapa de endereços notamos que as linhas A_{15} e A_{14} ficaram sempre em nível lógico "zero" e nível lógico "um" respectivamente, portanto estas linhas serão as linhas de controle do decodificador (G_1 e G_2). Continuando a análise, nota-se que as linhas A_{13}, A_{12}, A_{11}, A_{10} se alteram para cada grupo de 1K byte portanto chegamos a conclusão que, as linhas A_{13}, A_{12}, A_{11}, A_{10} serão as linhas de seleção do decodificador. Na figura 8.10 poderemos acompanhar o esquema global das ligações.

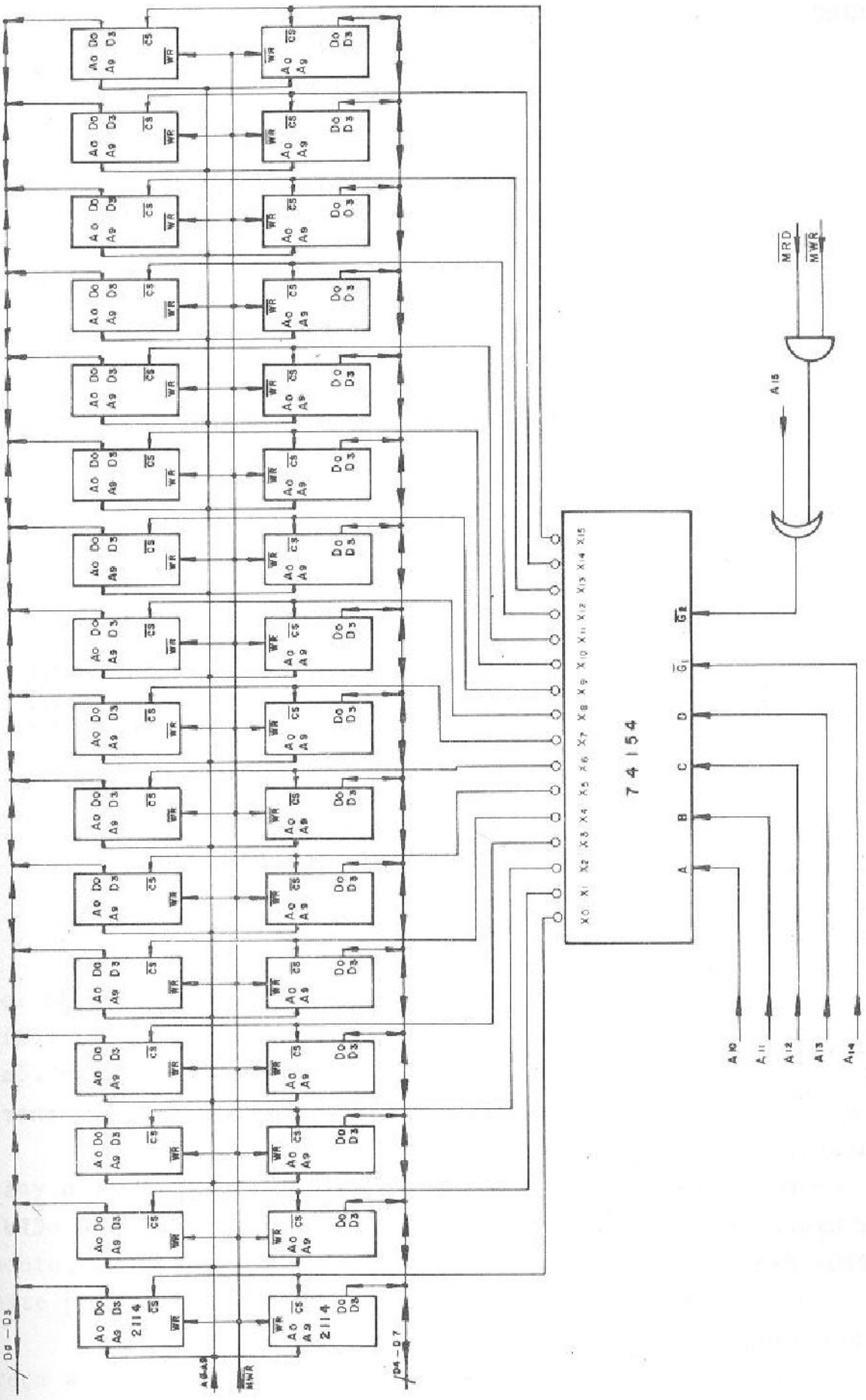


Fig. 8.10 - Expansão de Memória utilizando a 2114.

8.4 – CIRCUITO DE CLOCK

O Circuito de Clock neste projeto, foi feito de maneira a termos uma freqüência próxima a 2 MHertz. Este valor foi escolhido para que obtivéssemos uma máxima compatibilidade de velocidade com os periféricos de outras famílias.

No microprocessador Z-80, a oscilação tem que ser feita por meio de um circuito externo; já este tipo de circuito, no microprocessador 8085, não é necessário, porque a oscilação é feita internamente.

Basicamente, o sinal de clock é obtido através de um oscilador a cristal conectado ao CI 7474, como mostra a figura 8.11.

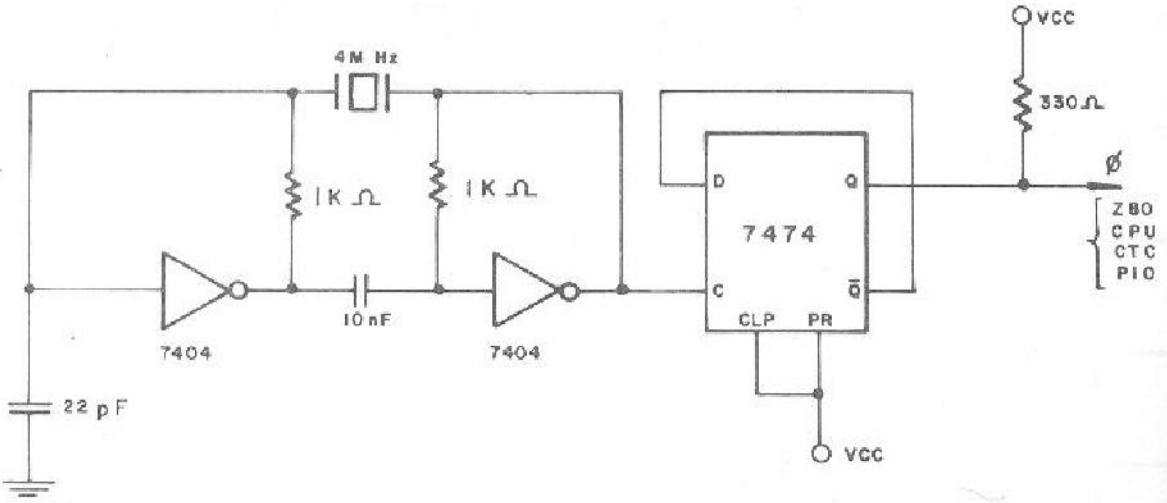


Fig. 8.11 – Circuito de Clock.

O flip-flop tipo D (7474) recebe então o sinal de oscilação, que neste caso será próximo de 4MHz, e devido a realimentação existente entre a saída $\overline{Q}$ e a entrada D, obtem-se na saída Q do flip-flop exatamente a metade da freqüência de entrada, ou seja, próximo de 2MHertz.

Desta forma, com o resistor de Pull-Up (330Ω), o sinal de clock gerado irá alimentar os seguintes CI'S: Z-80 CPU, Z-80 PIO, Z-80 CTC com um valor próximo a 2 MHz.

8.5 – DISPLAY

Com o avanço da tecnologia, já existem no mercado vários tipos de Displays além do tipo comum de 7 segmentos, sendo que um deles é o Alfanumérico.

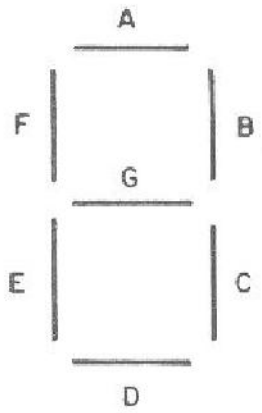
Atualmente existem dois tipos de Display Alfanuméricos, onde pode-se representar letras, números e símbolos. Um tipo é o matricial 5x7 colunas x linhas e o segundo tipo é o Alfanumérico formado por 16 segmentos e dois pontos.

Veremos a seguir alguns tipos de Display bem como algumas de suas características.

8.5.1 Características do FND 500

O tipo FND 500, é, um display de apenas 1 dígito de cátodo comum.

SEGMENTO		PINO
A	–	7
B	–	6
C	–	4
D	–	2
E	–	1
F	–	9
G	–	10
CÁTODO	–	3



8.5.2 Características do DL 5735

Este componente pertence a Siemens Company. Tal display é formado por 35 pontos ou 35 Leds, cujo acionamento é feito da seguinte forma, por exemplo: para ativar o primeiro ponto, aciona-se a primeira coluna e a primeira linha, sendo este processo usado para acionar qualquer outro ponto.

Como mostra a figura 8.12 por exemplo, para que apareça a letra "A" devemos proceder da seguinte maneira.

1º - Acionar a coluna 1 e as linhas 2, 3, 4, 5, 6, 7.
2º - Acionar a coluna 2 e as linhas 1 e 4.
3º - Acionar a coluna 3 e as linhas 1 e 4.
4º - Acionar a coluna 4 e as linhas 1 e 4.
5º - Acionar a coluna 5 e as linhas 2, 3, 4, 5, 6, 7.

Este processo acima descrito ocorre em um determinado instante de tempo, porque este tipo de display requer um circuito externo para fazer a multiplexagem da informação, afim de proporcionar uma varredura completa no campo de pontos do display.

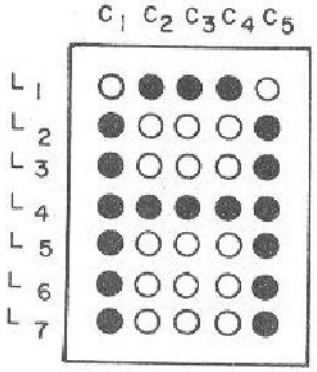


Fig. 8.12 - Exemplo de Acionamento do Display DL 5735.

8.5.2.1 Pinagem

PINO	FUNÇÃO	TIPO
1 -	Linha 1	Catodo
2 -	Linha 2	Catodo
3 -	Coluna 2	Anodo
4 -	Coluna 1	Anodo
5 -	Linha 6	Catodo
6 -	Linha 7	Catodo
7 -	Coluna 3	Anodo
8 -	Linha 5	Catodo
9 -	Coluna 4	Anodo
10 -	Linha 4	Catodo
11 -	Linha 3	Catodo
12 -	Coluna 5	Anodo

Características:

Corrente	-	20 mA
Tensão Máxima	-	2 V
Tensão Máx.reversa	-	3 V

8.5.3 Características do DL 1416

Este display DL 1416 é produzido também pela Siemens Company.

O display acima citado é Alfanumérico com 4 dígitos, memória e driver, sendo chamado de display inteligente porque não requer refresh de multiplex, decodificador e driver.

Os códigos apresentados no seu visor correspondem a 64 códigos do ASCII como se pode observar na figura 8.13, pela própria configuração do visor.

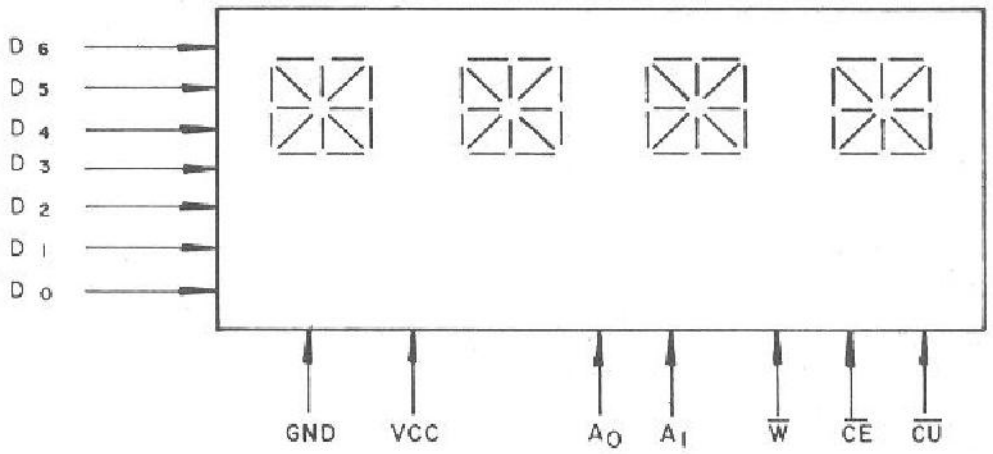


Fig. 8.13 - Configuração do DL 1416.

8.5.3.1 Pinagem

1 - $\underline{D_5}$ Dado entrada
2 - $\underline{D_4}$ Dado entrada
3 - $\underline{D_0}$ Dado entrada
4 - $\underline{D_1}$ Dado entrada
5 - $\underline{D_2}$ Dado entrada
6 - $\underline{D_3}$ Dado entrada
7 - $\overline{CE}$ Habilita o conjunto
8 - $\overline{W}$ Habilita a escrita

9 - $\overline{CU}$ Entrada do cursor
10 - A_0 Seleção dos dígitos menos significativos
11 - A_1 Seleção dos dígitos mais significativos
12 - Nada consta
13 - Nada consta
14 - Nada consta
15 - Nada consta
16 - Nada consta
17 - Nada consta
18 - V_{CC}
19 - GND
20 - D_6 dado entrada

Características:

V_{CC} = 5V
IC máximo = 100 mA
IIL = 160 A máx. V IN = 0,8 V
VIL = 0,8 V máx.
VZH = 2,7 V mín.

Existe um terminal de entrada ($\overline{CU}$), ativo em nível lógico"zero",que quando acionado e fornecido o endereço correspondente a um dígito, pelas entradas A_0 e A_1, fará com que sejam acionados todos os segmentos do correspondente dígito. Este terminal pode ser usado para se testar cada um dos segmentos de cada dígito ou indicar em qual dígito está sendo colocada a informação, sendo que nesta segunda aplicação ele se comporta como um cursor.

Uma nota importante é que ao acionar o cursor, este não destrói a informação anterior; isto significa que ao se desativar o $\overline{CU}$ aparecerá a informação que lá estava anteriormente.

Existem duas entradas (A_0 e A_1) que são usadas como linhas decodificadoras, para acionar cada um dos respectivos dígitos.

A entrada $\overline{CE}$ é usada para selecionar o conjunto inteiro, e a entrada $\overline{W}$ é usado para escrever normalmente, como

se escreve dados em uma memória do tipo RAM.

Todas as entradas são compatíveis com TTL.

Na figura 8.14 poderemos ver os símbolos em ASCII.

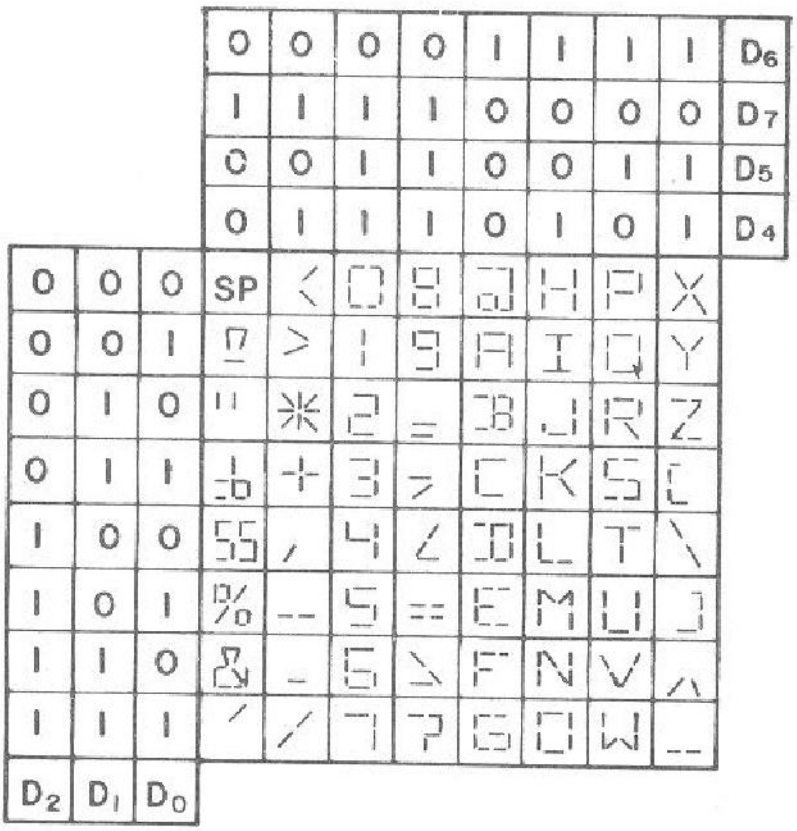


Fig. 8.14 - Seleção dos símbolos ASCII.

8.5.3.2 Modos de Operação

MODO 1:

A configuração neste modo é destinada à entrada de informação. Para que isto aconteça é preciso que o terminal ($\overline{CE}$) esteja a nível lógico"zero"e que o cursor ($\overline{CU}$) esteja desabilitado, isto é, a nível lógico"um"; nas linhas de dados seja colocada a informação e nas linhas de selecionamento colocado qual dos dígitos será selecionado; e finalmente, a linha de escrita ($\overline{W}$) colocada a nível lógico"zero". Na figura 8.15 pode-se acompanhar o status.

CONTROLE			ENDEREÇO		ENTRADA DE DADOS							DIGITOS			
$\overline{CE}$	$\overline{CU}$	$\overline{W}$	A_1	A_0	D_6	D_5	D_4	D_3	D_2	D_1	D_0	T_3	T_2	T_1	T_0
1	X	X	X	X	X	X	X	X	X	X	X	T	Z	5	D
0	1	0	0	0	0	1	1	0	0	1	0	T	Z	5	2
0	1	0	0	1	1	0	0	0	0	0	1	T	Z	A	2
0	1	0	1	0	1	0	0	0	1	1	1	T	6	A	2
0	1	0	1	1	1	0	0	1	0	1	1	K	6	A	2
0	1	0	0	1	1	0	1	0	1	0	1	K	6	U	2
0	1	0	0	0	0	1	1	1	0	0	1	K	6	U	9

Fig. 8.15 - Configuração para Modo 1 de operação.

MODO 2:

O segundo e último modo é destinado ao cursor, isto é, ao acionamento de todos os segmentos de cada dígito. Para que isto aconteça e preciso que o terminal ($\overline{CE}$) esteja a nível lógico"zero";o cursor ($\overline{CU}$) deve estar habilitado isto é, a nível lógico"zero"e o sinal de escrita ($\overline{W}$) também deve estar habilitado. As linhas de selecionamento (A_0 e A_1) neste modo como também as linhas D_6, D_5, D_4 são irrelevantes. Portanto, as linhas D_3, D_2, D_1, D_0 passaram a ser as linhas de selecionamento onde D_3 corresponde ao quarto dígito e assim sucessivamente.

Neste modo a informação não é destruída, isto significa que ao posicionar o cursor em um dígito, este irá acionar todos os segmentos, e se, no instante seguinte, o cursor for desabilitado, a informação anterior, sem modificação, retornará.

Na figura 8.16 pode-se acompanhar o status.

CONTROLE			ENDEREÇO		ENTRADA DE DADOS							DIGITOS			
$\overline{CE}$	$\overline{CU}$	$\overline{W}$	A_1	A_0	D_6	D_5	D_4	D_3	D_2	D_1	D_0	T_3	T_2	T_1	T_0
1	X	X	X	X	X	X	X	X	X	X	X	T	U	3	M
0	0	0	X	X	X	X	X	0	0	0	1	T	U	3	▒
0	0	0	X	X	X	X	X	0	0	1	0	T	U	▒	M
0	0	0	X	X	X	X	X	0	0	0	0	T	U	3	M
0	0	0	X	X	X	X	X	1	0	0	0	▒	U	3	M
0	0	0	X	X	X	X	X	1	1	1	1	▒	▒	▒	▒
0	0	0	X	X	X	X	X	0	0	0	0	T	U	3	M
0	0	0	X	X	X	X	X	1	1	0	0	▒	▒	3	M

Fig. 8.16 – Configuração para Modo 2 de Operação.

8.6 – TECLADO

Em primeiro lugar, estudaremos o REED-SWITCH que é um dispositivo usado em teclado.

Este componente é formado por duas lâminas magnéticas, colocadas dentro de uma ampola de vidro hermeticamente fechada e com um gás inerte, evitando a oxidação dos contatos.

A sua velocidade e durabilidade, são superiores a reles Eletro-mecânicos, pois a velocidade de fechamento dos Eletro-mecânicos, é da ordem de 1ms devido a ação da mola das lâminas, e no REED-SWITCH, devido a sua pequena massa e o espaço entre lâminas, a sua velocidade de fechamento é da ordem de micro-segundos. Devido a sua alta sensibilidade, bastam algumas dezenas de amper-espiras para que se fechem os contatos.

Para seu funcionamento é necessário um fluxo magnético "Ø" gerado no espaço entre as lâminas, originando-se entre elas pólos magnéticos opostos, fazendo com que as mesmas se aproximem, fechando os contatos. Com o desaparecimento do campo magnético, as lâminas separam-se pela sua própria força elástica.

O método aplicado na utilização do teclado consiste em movimentar o imã que fornece campo magnético, em direção perpendicular ao eixo longitudinal do "REED-SWITCH", como se pode ver na figura 8.17.

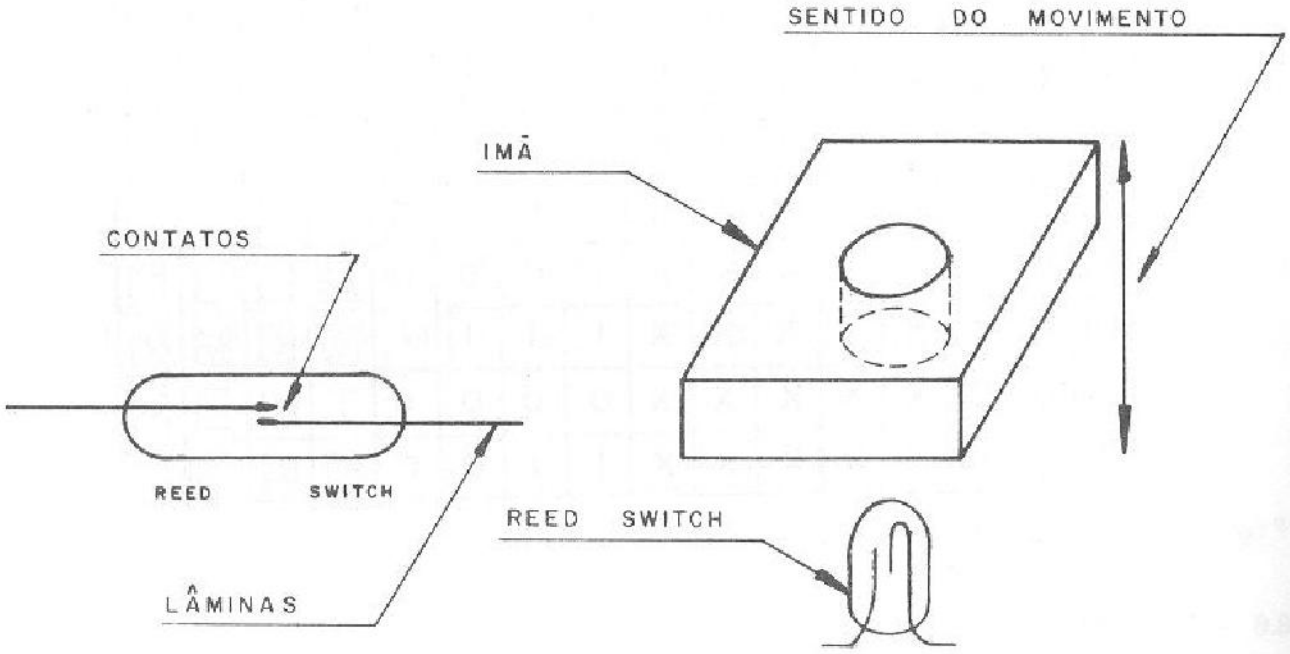


Fig. 8.17 - Detalhes de REED-SWITCH.

Ao se precionar alguma tecla pertencente ao conjunto do teclado, determinaremos um ponto da matriz, ou seja, uma linha e uma coluna.

Para saber se uma tecla foi acionada, basta colocar uma linha em nível lógico "zero" e verificar se as colunas correspondentes, que normalmente estão em nível lógico "um" devido a uma resistência ligada a V_{cc}, estão a nível lógico "zero".

Na figura 8.18 pode-se ver a estrutura acima mencionada.

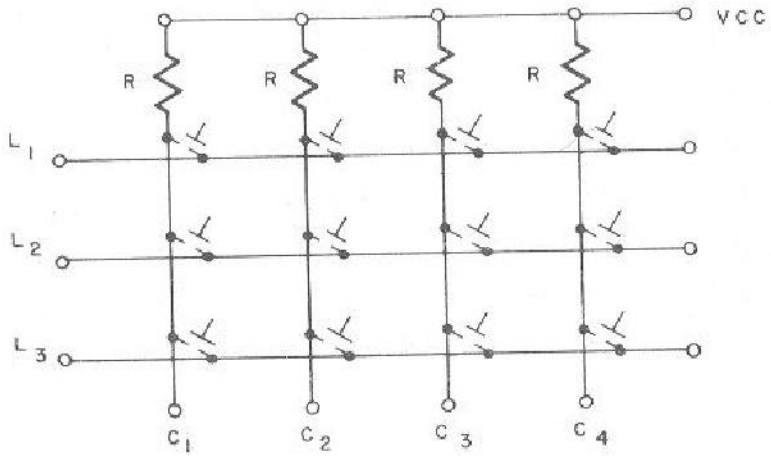
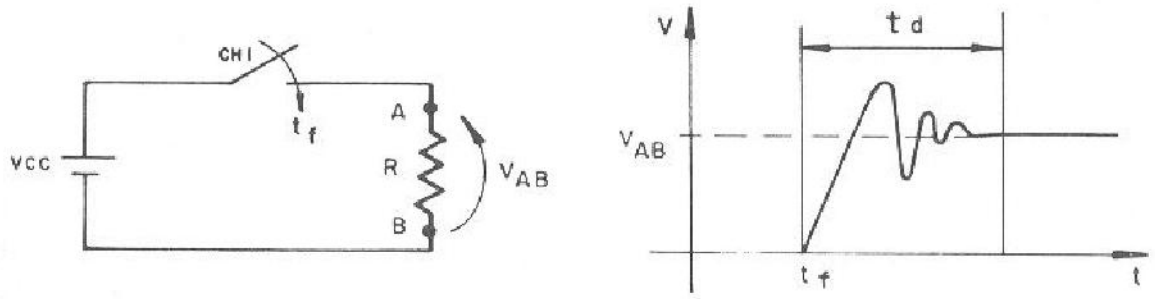


Fig. 8.18 - Matriz do Teclado.

Quando acionarmos uma tecla, ocorrerá o efeito de "DEBOUCE", que nada mais é do que um ruído da chave ocasionado quando do seu fechamento. Para que este ruído não nos atrapalhe, o sinal só será lido após um certo tempo ao se acionar a tecla, sendo que este espaço de tempo é chamado de "Delay".

Na figura 8.19 podemos acompanhar todo o processo.



tf - Instante do Fechamento
td - Tempo de "Delay"

Fig. 8.19 - Ruído de Chave (DEBOUCE).

Em geral, o tempo de Delay é de 10 a 20 ms, após o qual verificamos se ainda existe alguma tecla sendo pressionada. Se houver, a tecla é considerada válida.

Decodificando a linha e a coluna podemos gerar o código hexa ou ASCII através de uma ROM pré-programada com uma tabela de códigos, ou por meio de uma rotina de Software.

8.7 – MULTIPLEXAGEM DE DISPLAY COM TECLADO

Nos dois itens anteriores foram explanados o display e o teclado individualmente, e neste item iremos fazer o estudo englobado do display e teclado como sendo um único dispositivo.

Circuitos a microprocessador freqüentemente usam display de sete segmentos para mostrar dados, endereços e indicações em geral.

Existem dois modos de se fazer multiplexagem: por meio de Software (programação) ou por meio de Hardware (circuito). Neste projeto será feita multiplexagem por Software, ou seja, por programa e os displays serão de catodo comum sendo que para cada segmento foi usado um transistor como driver (amplifi

cador de corrente), e um outro transistor também usado para o selecionamento de cada dígito.

O tempo com o qual cada dígito será ativado com a sua respectiva informação será aproximadamente de 1 ms.

As linhas CD_1 a CD_6 tem duas finalidades. A primeira é a de selecionar qual o dígito será acionado em conjunto com as linhas Sa a Sp, e a segunda é verificar se houve teclas acionadas, em conjunto com as linhas L_1 a L_8.

Exemplificando o procedimento acima mencionado: para acionarmos o primeiro dígito, devemos ativar a linha CD_1 e, no instante seguinte colocarmos a informação de qual segmento deste dígito será acionado, através das linhas Sa a Sp. A próxima etapa é ler as linhas L_1 a L_8 e se alguma destas linhas estiver a nível lógico "um" indicará que existe tecla acionada. Por exemplo: se a tecla Ts for acionada, ao lermos as linhas L_1 a L_8, verificamos que a linha L_5 está a nível lógico "um" portanto, com a linha L_5 nessas condições e a linha CD_1 acionada saberemos que existe uma tecla ativada e que é a tecla 5.

A seguir mostraremos o diagrama em blocos do processo descrito acima.

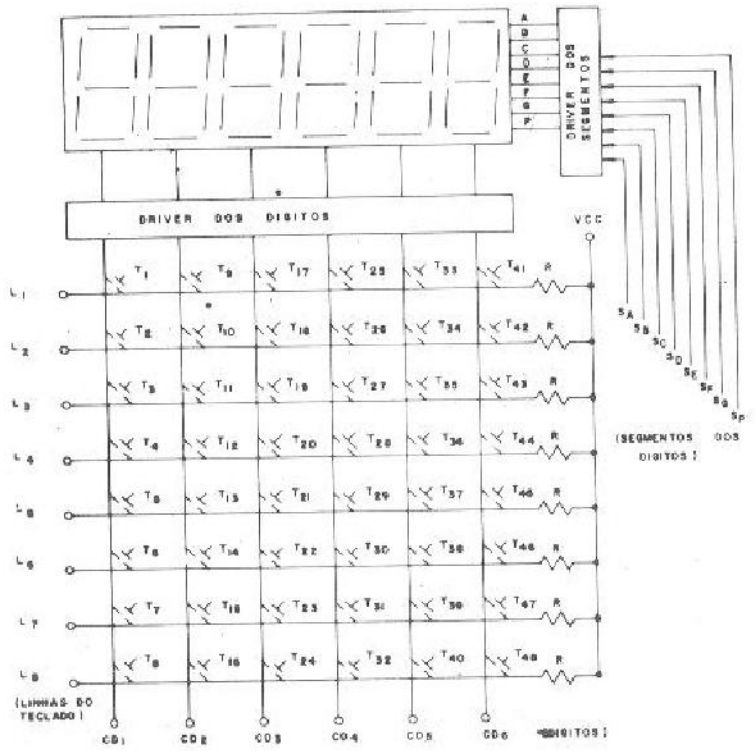


Fig. 8.20 - Diagrama de Interligação Teclado-Display.

Na seqüência, daremos o procedimento da multiplexagem envolvendo display e teclado.

Para se acionar um display é necessário que o dado seja armazenado em um BUFFER 74LS 273 que retém a informação desejada, pois este integrado contém 8 flip-flops tipo D. Tal BUFFER portanto, constitui-se numa interface para o display. Para que um dado seja armazenado neste BUFFER é preciso que a CPU execute uma instrução de OUT, fazendo com que nas linhas de endereços de A_7 a A_0 seja colocado o endereço do dispositivo display, que no caso será (A8) H, e na via de dados seja colocada a informação desejada. Este endereço é decodificado pelo 74LS 138 o qual, através de suas linhas selecionadoras, gera um clock para o CI 74LS 273 que armazena o dado presente em suas entradas, acionando também os transistores T_1 a T_8 (vide esquema 6) que têm a finalidade de amplificar o sinal que porventura venham a acionar os segmentos dos displays, eliminando-se a necessidade do decodificador BCD de 7 segmentos. Os 6 displays terão suas linhas de segmentos ligadas em paralelo.

Colocada a informação, para acionarmos os segmentos dos dígitos, necessitamos agora dizer qual dígito será acionado. Para tanto, devemos colocar esta informação em um outro BUFFER cujo endereço é (AC) H.

Resumindo, para se acionar um display com uma dada informação, são necessárias duas condições, não importando a ordem. Primeiro, colocar a informação desejada e segundo, enviar qual dígito que será acionado.

O tempo de acionamento de cada display é de aproximadamente 1m seg e após esse período, para cada display deverá se repetir o processo anterior, efetuando deste modo um REFRESH da informação.

O buffer de endereço (AC) H, tem duas funções: primeiro, de monitorar o display e segundo, o de verificar se houve alguma tecla apertada. Outro BUFFER 74LS 244, formado por 8 linhas de BUFFER TRI-STATE com drivers, controlado pelo endereço (AØ) H, é usado como interface de entrada do teclado para a via de dados, sendo que esta interface terá a função de saber qual tecla foi acionada.

8.7.1 Funções do Teclado

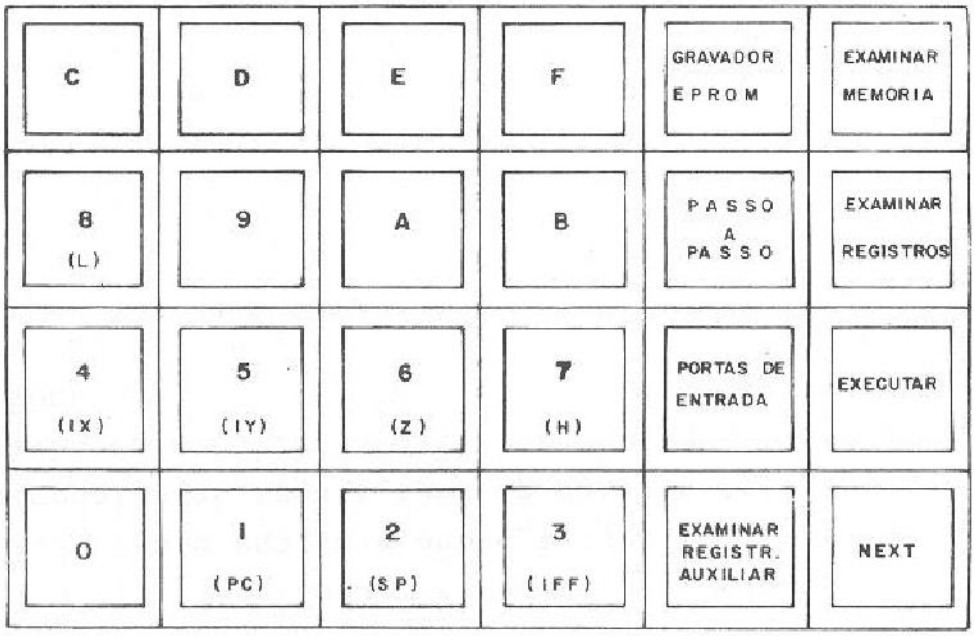


Fig. 8.21 Teclado

Este teclado é composto de 24 teclas, sendo 16 teclas hexadecimais de Ø a F e 8 funções.

8.7.1.1 Teclas Hexadecimais

Através destas 16 teclas o operador poderá colocar, examinar, e alterar dados além de programar em linguagem de máquina.

8.7.1.2 Examinar a Memória

Esta tecla pode alterar ou examinar o conteúdo de uma determinada posição de memória. Para isto, entra-se com quatro dígitos de Ø a F que representam o endereço de memória a ser verificado ou modificado e em seguida aciona-se a tecla " examinar memória" obtendo-se uma informação completa, endereço e dado. Presssionando-se a tecla Next, o endereço da memória será incrementado de uma posição e será mostrado o dado contido neste endereço.

8.7.1.3 Examinar Registrador

Os seguintes registradores podem ser examinados e/ou alterados pelo uso da tecla " Examinar Registradores", A, B, C, D, E, F, H, L, I, IFF, PC, IX, IY.

Para se saber o conteúdo de cada registrador pressiona-se a tecla "Examinar Registrador" e em seguida aciona-sea tecla do respectivo registrador.

No display será mostrado o registrador solicitado e o seu conteúdo.

8.7.1.4 Examinar Registrador Auxiliar

Mesmo processo acima descrito só que aciona-se a tecla " Examinar Registro Auxiliar", sendo os registradores utilizados A', B', C', D', E', H', L'.

8.7.1.5 Portas de Entrada

Esta tecla é utilizada para verificar ou alterar um dos 256 endereços de localização de portas permissíveis na arquitetura do Z-80. Para isto basta entrar com dois dígitos Hexadecimais; após isto, pressiona-se a tecla " Portas de Entradas" e obtém-se o conteúdo da porta selecionada.

8.7.1.6 Executar

Possibilita ao usuário comandar a execução de programas em RAM ou EPROM. Para isto entra-se com os quatro dígitos do endereço inicial e aciona-se a tecla " Executar".

8.7.1.7 Next

A tecla "Next" é utilizada em conjunto com outras funções, como exame de memória, exame de registrador, exame de portas e de programador, de EPROM. Nos três primeiros casos é utilizada para apresentar a posição de memória, conteúdo de portas ou dos registradores. Na programação de EPROMS é usada para dar início à gravação.

8.7.1.8 Passo a Passo

A tecla "Passo a Passo" permite ao usuário executar o seu programa uma instrução por vez. A cada instrução, é permitida a verificação do conteúdo de registradores, memórias e portas.

8.7.1.9 Gravador de EPROM

A tecla "Gravador de EPROM" movimenta dados da memória RAM, para EPROMS. Para isto entra-se com os quatro dígitos do endereço inicial onde se localizam os dados e aciona-se a tecla "Gravador de EPROM". Em seguida, coloca-se o endereço final de onde se localizam os dados e logo após pressionamos a tecla "Next", dando início à gravação.

8.8 – GRAVADOR DE EPROM

O intuito do estudo aqui apresentado, será o de gravar a informação desejada pelo usuário em EPROM, sendo apresentadas duas versões: gravar em EPROM 2716 e 2732.

A técnica necessária para que o microcomputador forneça os sinais para a gravação de EPROM será mostrada a seguir.

O controle será feito pelo programa monitor utilizando do seu Software que desloca as informações desejadas de um lugar específico da memória RAM para a EPROM transportanto para o endereço situado entre F000H a FFFFH.

Através do canal 2, o Z-80 CTC é programado para gerar uma onda quadrada de semi-período igual a 50 ms, calculo este que deverá ser feito a partir da freqüência do Clock utilizada (freqüência do cristal usado). Os circuitos integrados 74LS 74 do esquema quatro formam um contador sincrono controlado pela saída ZC/T02 do Z-80 CTC.

A saída "Q_6" do circuito integrado 74273 será o controle do Vpp, este controle é necessário pelo motivo que a 2732 deve variar de "0V" a aproximadamente "25V"; isto ocorre porque o pino de Vpp é o mesmo do $\overline{OE}$, sendo que este controle não é

necessário na 2716, porque o pino Vpp não é agrupado com nenhum outro pino.

O circuito integrado 7432 do esquema quatro decodifica os estados deste contador de modo a produzir a onda pulsante de 50 ms, necessária para a programação da EPROM. Esta saída é então invertida e aplicada a entrada do sinal de $\overline{\text{WAIT}}$ da Z-80 CPU, forçando a CPU a manter em seus barramentos, dados, endereço e controle durante o tempo de gravação, isto é, o tempo de 50 ms.

A seqüência é a seguinte:

a.) O contador sincrono será habilitado através do pulso PGM.

b.) Será colocado a tensão respectiva em Vpp através do PGM 1.

c.) O canal 2 do CTC é programado pelo programa monitor para gerar um atraso de 50 ms.

d.) Neste momento iniciam a escrita nos endereços decodificados ocasionando que o pino da PROM-2-SEL vá a nível lógico Ø. O endereço e o dado são colocados nas suas respectivas vias.

e.) A seleção indo a nível lógico "zero", dará um pulso no circuito integrado 74LS 74 do esquema quatro que fará com que o pino $\overline{\text{CS}}$ da memória a ser gravada vá a nível lógico "um", e o sinal PD/PGM vai a nível lógico "um", portanto o sinal de $\overline{\text{WAIT}}$ do Z-80 CPU irá a nível lógico "zero".

f.) O Z-80 CPU permanece em estado de espera até que o terminal ZC2 - Z-80 - CTC, forneça o sinal de 50 ms levando PD/PGM a nível lógico "um", e o sinal $\overline{\text{WAIT}}$ a nível lógico "um", conseqüentemente, liberando o Z-80 - CPU do estado de espera.

Neste processo acima descrito, para gravação de EPROM, não se deve esquecer que quando a CPU entra em estado de $\overline{\text{WAIT}}$

ela suspende o refresh das memórias RAM dinâmicas, e como neste sistema que descrevemos não existem RAM'S dinâmicas, não haverá nenhum problema.

8.9 – MAPEAMENTO DE DISPOSITIVOS DE ENTRADA E SAÍDA

O processo para implementação do mapeamento de dispositivos de entrada e saída é o mesmo adotado para as memórias, com apenas uma diferença, que as linhas de endereço são apenas 8 de A_0 a A_7.

Outro item importante é que tanto a Z-80 - CTC como a Z-80 - PIO, se comportam como se cada um tivesse 4 dispositivos de entrada e saída como foi visto nos respectivos capítulos.

Neste caso, cada saída do decodificador representará quatro endereços como é visto na tabela a seguir.

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	ENDEREÇO	DISPOSITIVO
1	0	1	0	0	0	1	1	A_0 - A3H	Z-80 - PIO
1	0	1	0	0	1	1	1	A_4 - A7H	Z-80 - CTC
1	0	1	0	1	0	1	1	A_8 - ABH	LATCH/SEGMENTO
1	0	1	0	1	1	1	1	AC - AFH	LATCH/DÍGITO
1	0	1	1	0	0	1	1	B0 - B3H	LATCH/TECLADO
1	0	1	1	0	1	1	1	B4 - B7H	EXPANSÃO
1	0	1	1	1	0	1	1	B8 - BBH	EXPANSÃO
1	0	1	1	1	1	1	1	BC - BFH	EXPANSÃO

Nota-se que apenas as linhas A_4, A_3, A_2, se alteram, portanto chegamos a conclusão que estas linhas serão as selecionadoras do decodificador que neste caso será o decodificador 74LS 138. Como as linhas A_7 e A_6 permanecem constantes, estas serão as linhas de controle.

Os dispositivos de entrada e saída podem ser ativadas apenas com o sinal $\overline{IORQ}$; portanto, chegamos à conclusão de que o $\overline{IORQ}$ será também uma das linhas de controle do 74LS 138. Pela figura 8.22 pode-se acompanhar as ligações de todos os sinais.

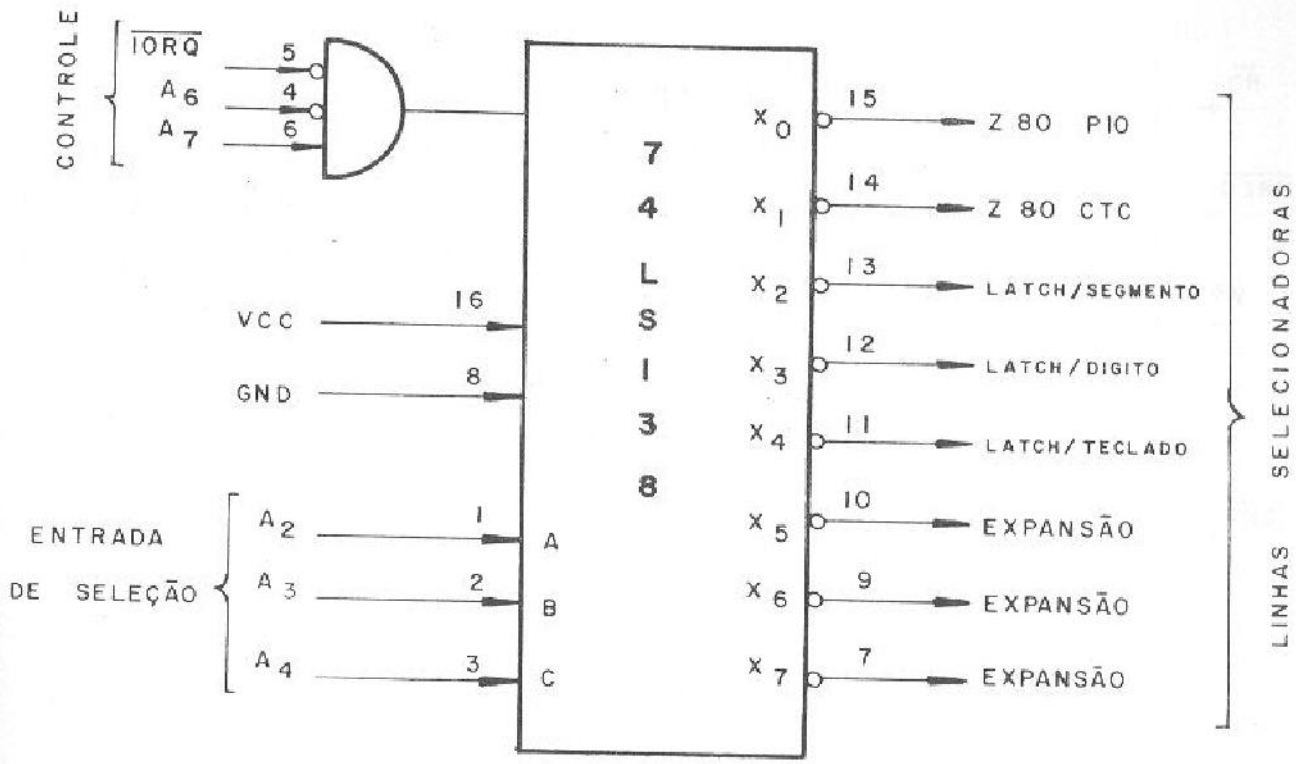
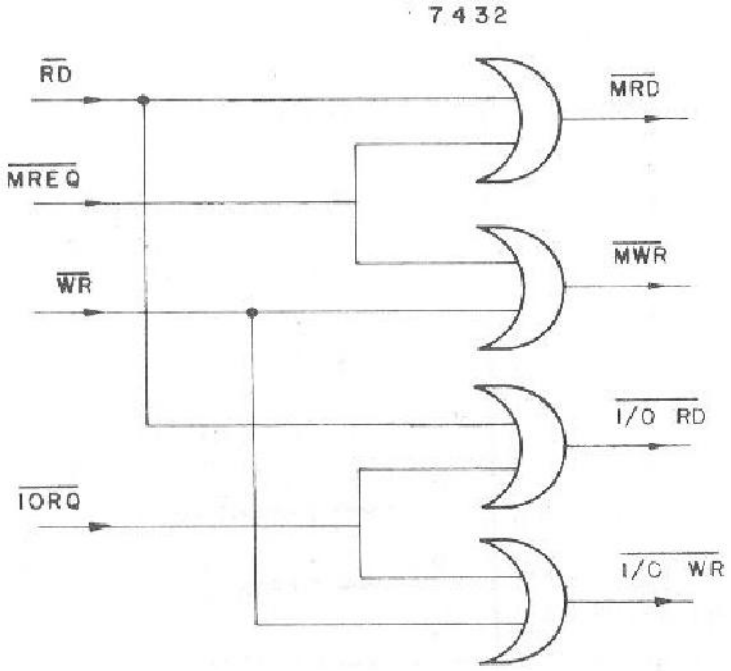


Fig. 8.22 - Configuração do decodificador para a seleção dos dispositivos I/O.

8.10 – AGRUPAMENTO DAS FUNÇÕES DE LEITURA E ESCRITA

Os sinais de leitura e escrita fornecidos pelo Z-80 CPU têm 2 (duas) funções: leitura e escrita em memórias e leitura e escrita em dispositivos de entrada e saída. Para ser ter esta separação existem os sinais $\overline{IORQ}$ e $\overline{MREQ}$ que são respectivamente operações em dispositivo de entrada e saída, e operações em memória.

Como necessitamos de apenas um sinal, onde este sinal represente leitura em memória, leitura em dispositivo de entrada e saída, escrita em memória e finalmente escrita em dispositivos de entrada e saída, este agrupamento será feito com o circuito integrado 7432 (Bloco lógico "OU") como se pode ver na figura 8.23.



$\overline{MRD}$ - Leitura em memória
$\overline{MWR}$ - Escrita em memória
$\overline{I/ORD}$- Leitura em disp. I/O
$\overline{I/OWR}$- Escrita em disp. I/O

Fig. 8.23 - Agrupamento dos Sinais $\overline{IORQ}$, $\overline{MREQ}$, $\overline{RD}$ e $\overline{WR}$ para operações de Leitura/Escrita.

8.11 – DIAGRAMA EM BLOCOS E CIRCUITOS ELÉTRICOS

8.11.1 Diagrama em Blocos Geral

Na figura 8.24 pode-se ver a arquitetura em blocos do sistema.

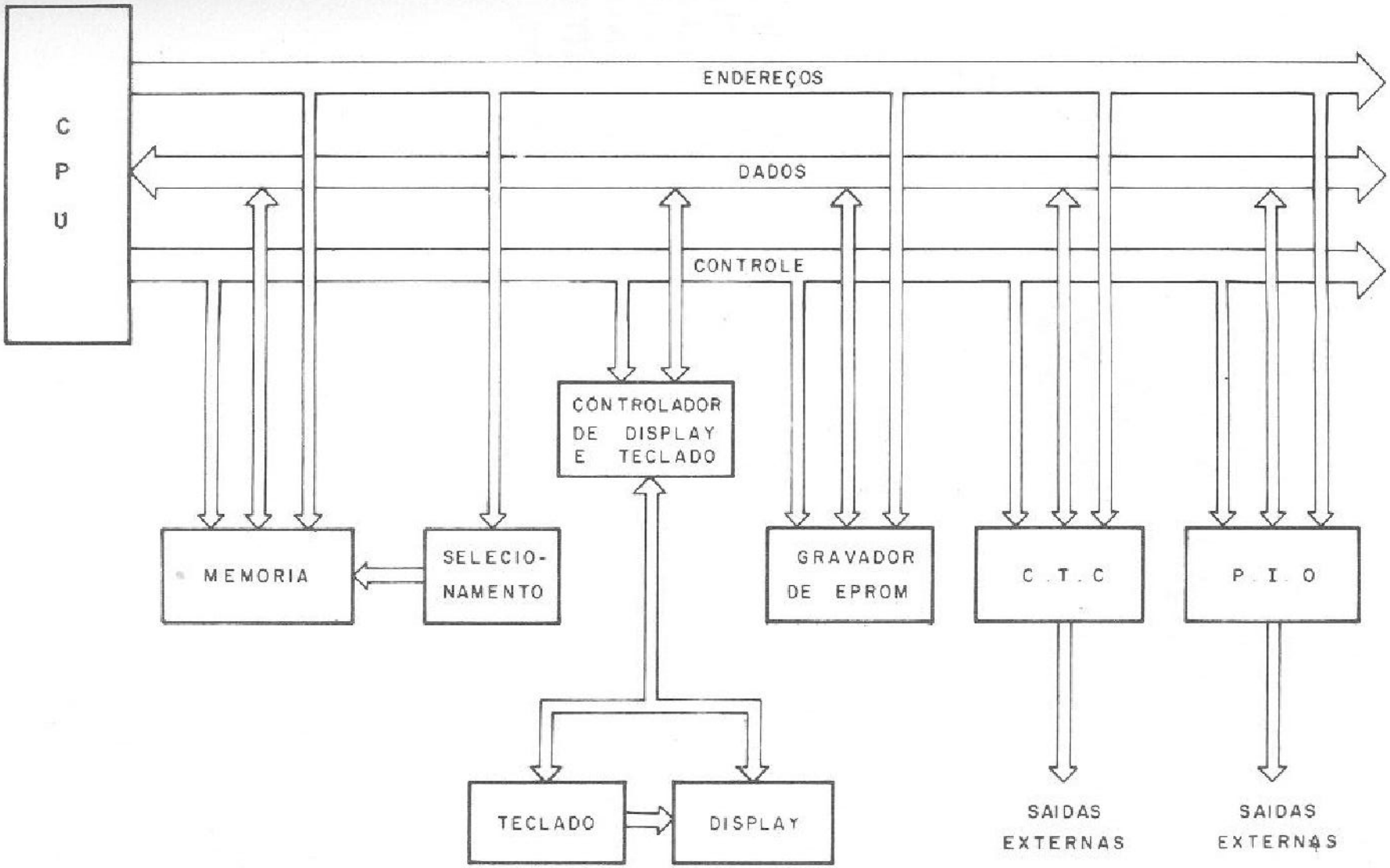


Fig. 8.24 - Arquitetura geral.

8.11.2 Circuitos Elétricos

Nas páginas seguintes serão apresentados os esquemas elétricos de um microcomputador.

Tais esquemas têm a finalidade de reunir todo o estudo realizado anteriormente.

A representação "EQ" que aparece em todos os esquemas significa, juntamente com os algarismos que o sucedem, de onde vem ou para onde vai cada sinal, dependendo do sentido das setas.

8.11.2.1 Esquema Um.

Pode-se dizer que o esquema número um representa o cérebro do microcomputador.

Ele contém o microprocessador Z-80, o circuito de Clock, o circuito de Reset e por fim o circuito que reúne a leitura e escrita em memória e a leitura e escrita em dispositivos de entrada e saída (I/O).

Pode-se notar também, que existem quatro resistências de 10 KΩ ligadas entre V_{CC} e os pinos $\overline{BUSRQ}$, $\overline{NMI}$, $\overline{INT}$ e $\overline{WAIT}$ cuja finalidade é a de evitar a penetração de ruído externo nesses pinos.

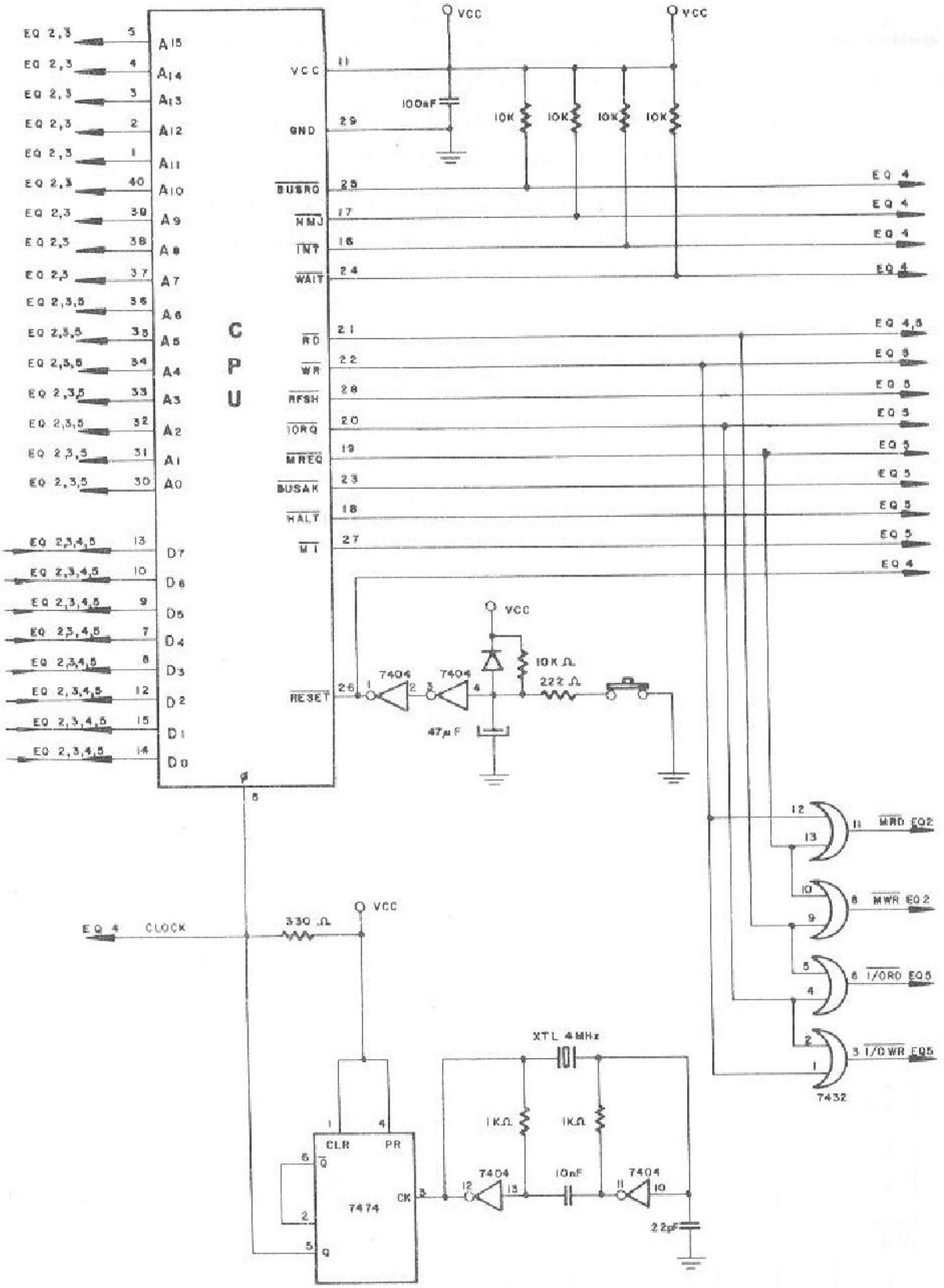


Fig. 8.25 - Esquema Um.

8.11.2.2 Esquema Dois

No esquema número dois está representando o circuito de memórias e o respectivo decodificador, através do qual é feito o mapeamento, como foi visto em itens anteriores.

É nesta parte do circuito que ocorre toda a movimentação de dados e programas que serão manipulados pela CPU.

Este circuito requer todas as linhas de dados e endereços além dos sinais $\overline{MRD}$ e $\overline{MWR}$, todos vindos do esquema um.

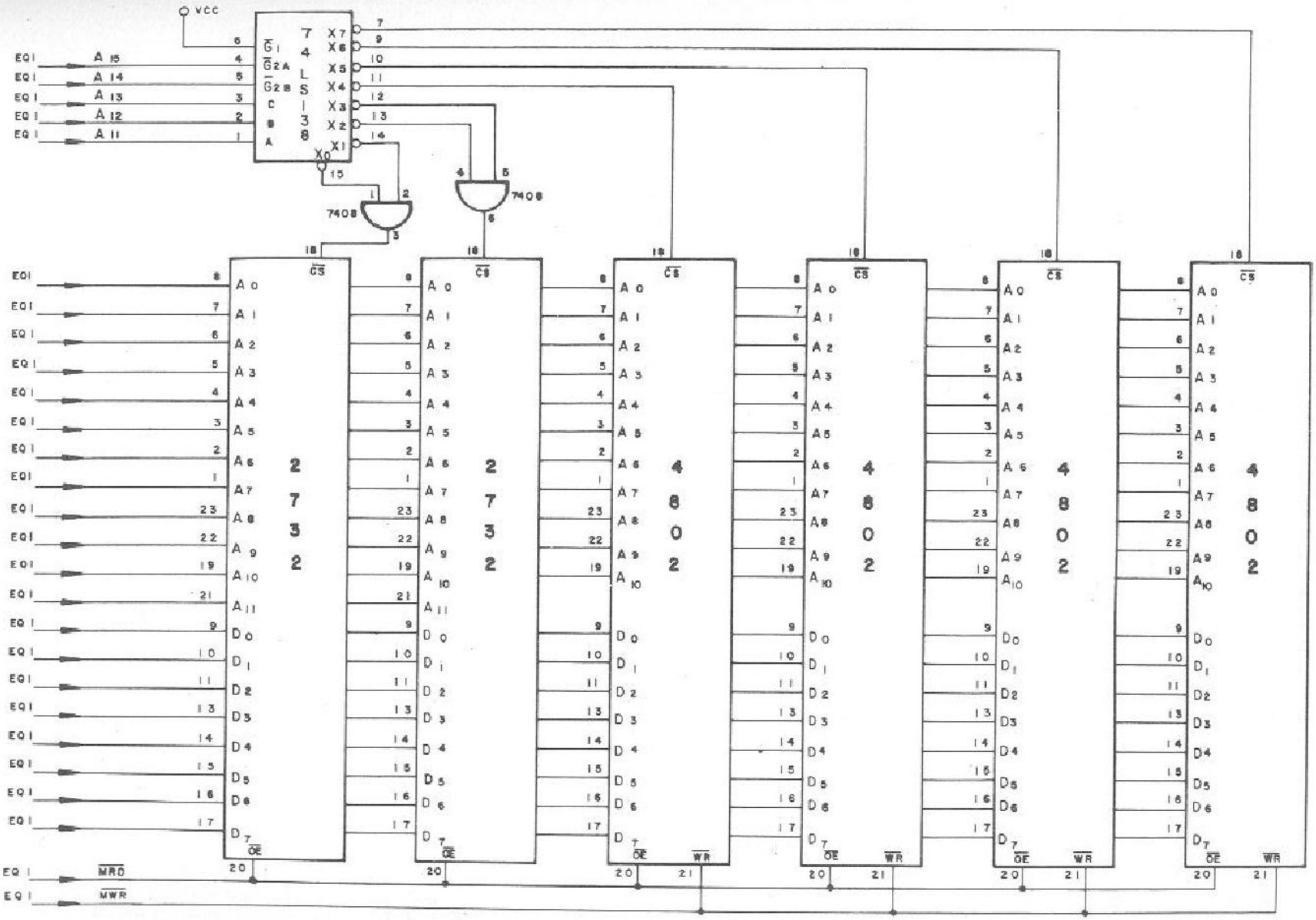


Fig. 8.26 - Esquema Dois.

8.11.2.3 Esquema Três

No esquema número três constam o gravador de EPROM e o circuito auxiliar formado pelo transistor 2N3904 cuja função é a de controlar a entrada de 25 Volts que pode variar desde "5" Volts até 25 Volts.

A porta 742Ø formada por um bloco lógico "NE" (NAND) de quatro entradas, controla o endereçamento da memória EPROM a ser gravada.

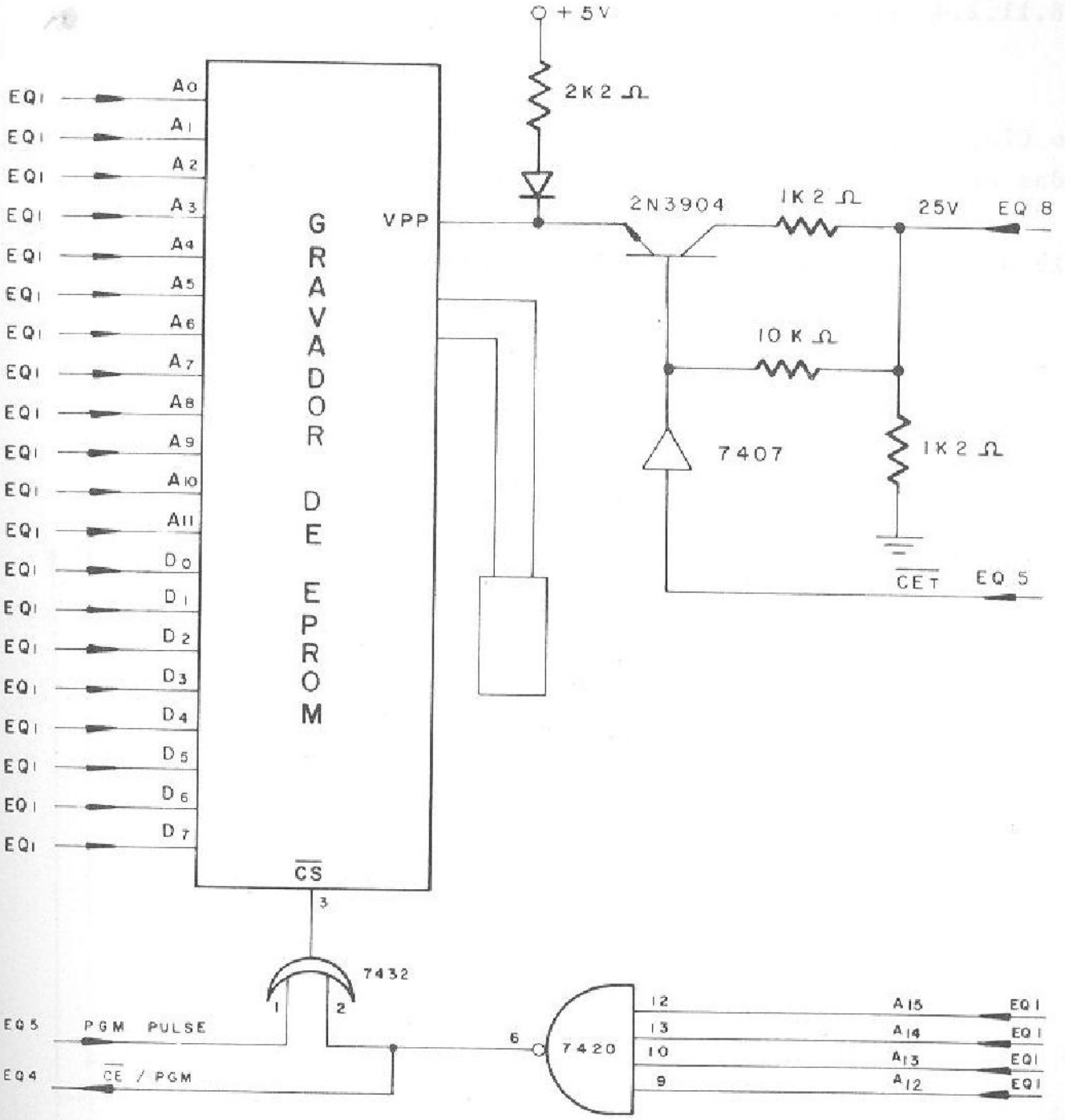


Fig. 8.27 - Esquema Três.

8.11.2.4 Esquema Quatro

No esquema número quatro estão representados a PIO e o CTC, cujo funcionamento e características já foram mencionadas em seus respectivos capítulos.

Ele contém também o circuito que irá fazer o controle do gravador de EPROM, formado pelos flip-flops tipo "D".

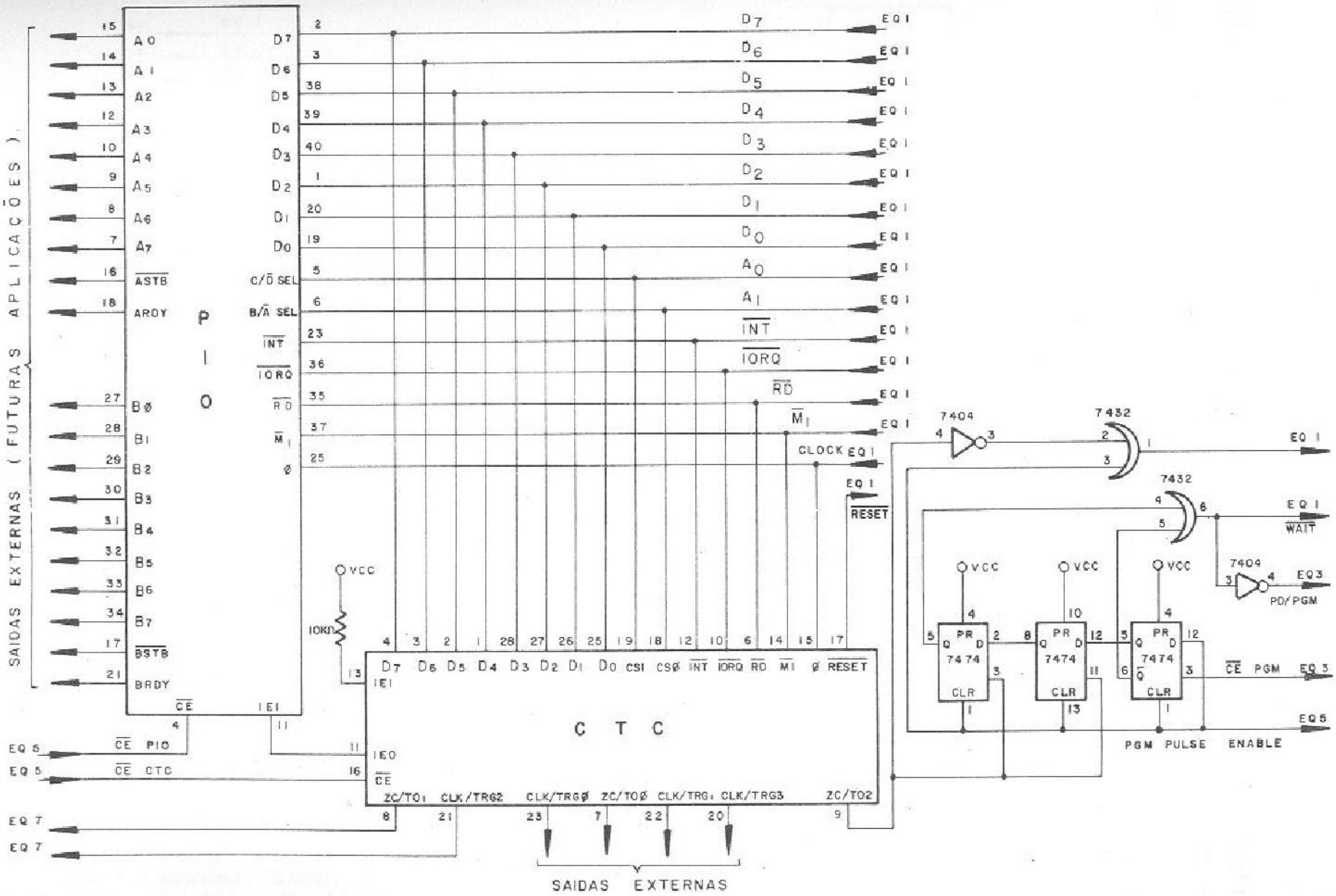


Fig. 8.28 - Esquema Quatro.

8.11.2.5 Esquema Cinco

O esquema número cinco é composto basicamente de duas partes. A primeira consta do decodificador 74LS 138 responsável pelo selecionamento dos circuitos de entrada e saída.

E a segunda é formada pelos integrados 74244 e 74273 que irão controlar o display e o teclado.

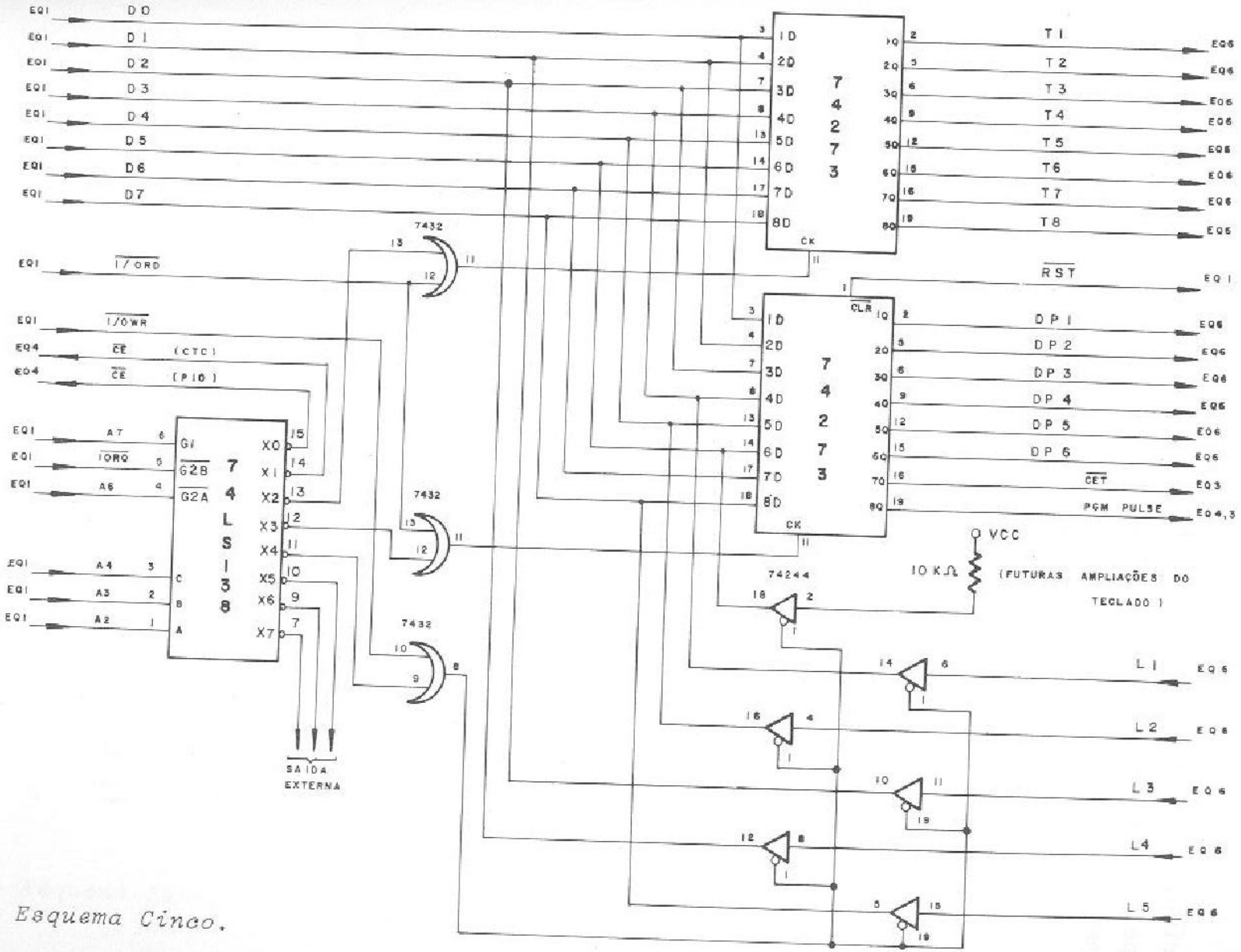


Fig. 8.29 - Esquema Cinco.

8.11.2.6 Esquema Seis

O esquema número seis está dividido em três partes. A primeira consta do driver (amplificador de corrente) composto pelos transistores de T_1 a T_{13} (2N2907). A segunda parte é composta por seis unidades de display e a terceira e última, por 28 teclas formando o conjunto do teclado.

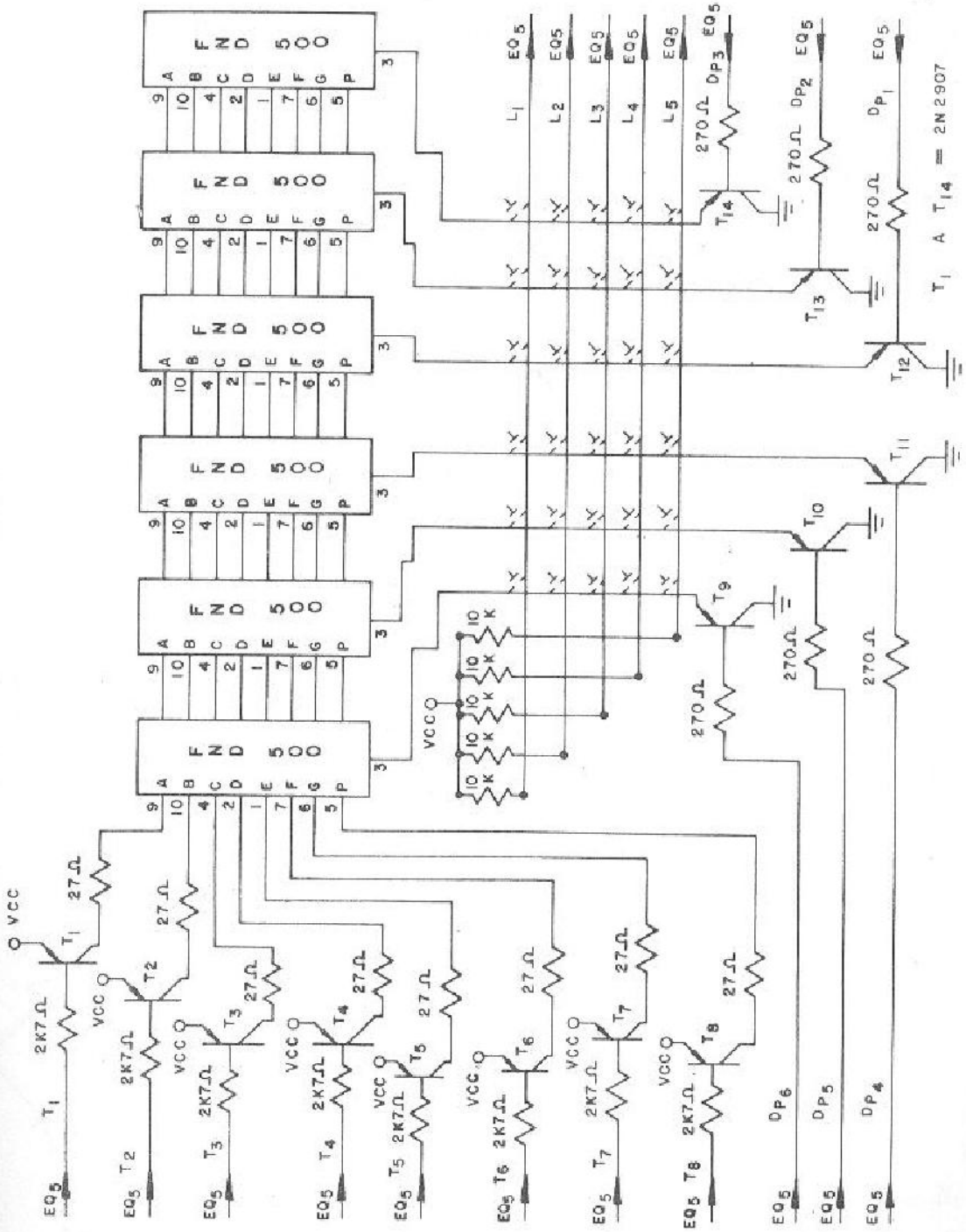


Fig. 8.30 - Esquema Seis.

8.11.2.7 Esquema Sete

O esquema número sete reúne duas fontes. Uma de 5 Volts e 3 amperes que alimenta todo o circuito, e outra de 25 Volts que alimenta o circuito do gravador de EPROM.

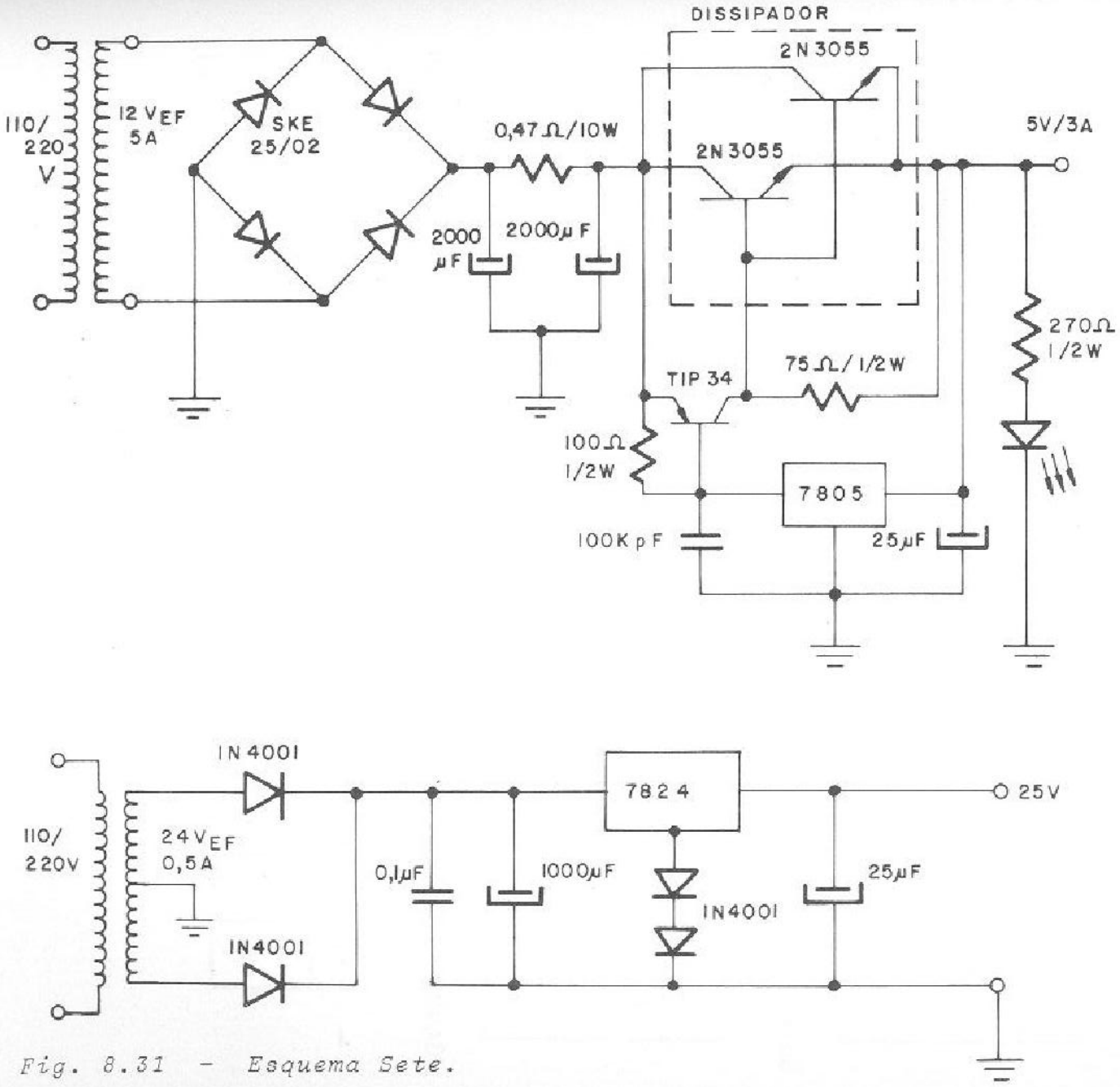


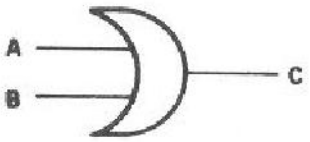
Fig. 8.31 - Esquema Sete.

APÊNDICE 1

BLOCOS LÓGICOS

Neste item serão apresentados os blocos lógicos existentes, para elaboração de funções que serão desenvolvidas nos capítulos a seguir.

BLOCO "OR" (OU)



(símbolo)

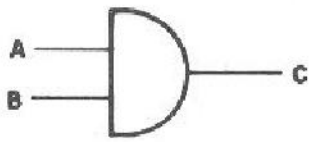
FUNÇÃO DE TRANSFERÊNCIA

$C = A + B$

TABELA DA VERDADE

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

BLOCO "AND" (E)



(símbolo)

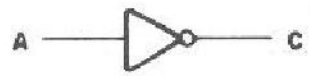
FUNÇÃO DE TRANSFERÊNCIA

$C = A \cdot B$

TABELA DA VERDADE

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

BLOCO "INVERSOR"



(símbolo)

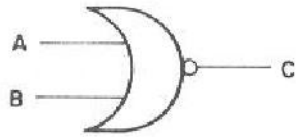
FUNÇÃO DE TRANSFERÊNCIA

$C = \overline{A}$

TABELA DA VERDADE

A	C
0	1
1	0

BLOCO "NOR" (NOU)



(símbolo)

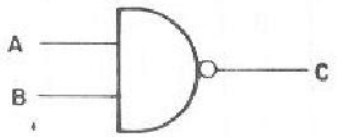
FUNÇÃO DE TRANSFERÊNCIA

$C = \overline{A + B}$

TABELA DA VERDADE

A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

BLOCO "NAND" (NE)



(símbolo)

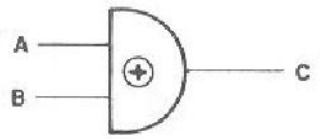
FUNÇÃO DE TRANSFERÊNCIA

$C = \overline{A \cdot B}$

TABELA DA VERDADE

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

BLOCO "OU EXCLUSIVO"



(símbolo)

FUNÇÃO DE TRANSFERÊNCIA

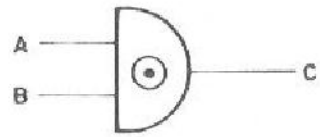
$C = A \oplus B$

$C = \overline{A}.B + A.\overline{B}$

TABELA DA VERDADE

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

BLOCO "COINCIDÊNCIA"



(símbolo)

FUNÇÃO DE TRANSFERÊNCIA

$C = A \odot B$

$C = AB + \overline{A}\,\overline{B}$

TABELA DA VERDADE

A	B	C
0	0	1
0	1	0
1	0	0
1	1	1

APÊNDICE 2

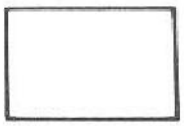
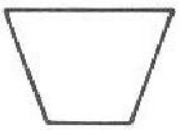
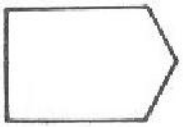
FLUXOGRAMAS

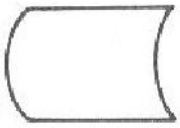
O principal intuito de um fluxograma é o de representar graficamente a ocorrência de um evento.

O fluxograma foi criado para podermos ter o delineamento de uma função, de um programa, de uma linha de montagem ou de outros eventos.

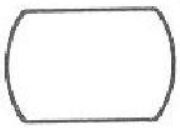
Na área de programação é de grande importância a elaboração de fluxograma, pois a visualização do programa torna-se fácil e quando necessário for corrigirmos alguma etapa do mesmo, ficará a tarefa de correção muito mais simples.

A seguir serão apresentados os símbolos usados na construção de fluxogramas.

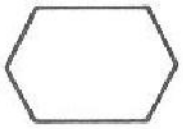
SÍMBOLOS	DESCRIÇÃO
	→ Processamento: Um grupo de instruções que executa uma função de processamento do programa.
	→ Entrada e Saída: Comunicação com dispositivos externos.
	→ Modificação de programa: Uma instrução ou grupo de instruções que modificam o programa.
	→ Documento: Documentos e relatórios de todas as variedades.
	→ Operação auxiliar: Uma operação de máquina que suplementa a função principal de processamento.



→ Acesso a dispositivo externo:
Uma operação para acessar memória do tipo disco ou tambor.

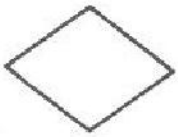


→ Operação de teclado:
Uma operação em que se utiliza um dispositivo com teclado.



→ Processamento Pré-definido:
Rotina que será executada fora do programa principal.

→ Conexão:
Uma entrada ou uma saída de, ou para outra página do diagrama.

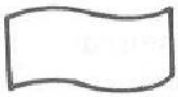


→ Decisão:
Símbolo utilizado para indicar a possibilidade de desvios para diversos pontos do programa de acordo com situações variáveis.

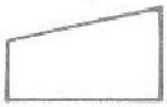
→ Fluxo:
A direção do fluxo de dados ou de processamento.

→ Cartão perfurado:
Todas as variedades de cartões perfurados.

→ Fita de Transmissão:
Fita de máquina de somar ou similar.



→ Fita perfurada:
Fita de papel ou plástico perfurada.



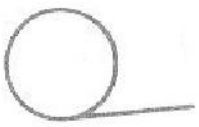
→ Teclado em linha:
Informação fornecida ou recebida pelo computador utilizando um dispositivo externo.

→ Terminal:
O ponto de início, término ou interrupção de um programa.

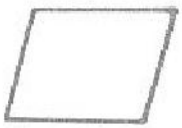
→ Conexão:
Uma entrada ou uma saída de uma para outra parte do diagrama de blocos.

→ Linha de comunicação:
Uma transmissão automática de informação, entre locais diferentes através de linhas de comunicação.

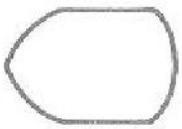
→ Memória fora de linha:
Memória fora de linha em fichas, cartões, fitas magnéticas ou perfuradas.



→ Fitas magnéticas.



→ Entrada/saída:
Qualquer função de um dispositivo de entrada/saída.



→ Exibição:
Informações exibidas por dispositivos visuais.

APÊNDICE 3

1 - SISTEMAS DE NUMERAÇÃO

O sistema de numeração conhecido por todos nós é o sistema decimal, o qual é composto por dez diferentes símbolos que são os números de "0" a "9".

No ramo da computação é utilizado um outro sistema de numeração, o binário, o qual é composto por apenas dois diferentes símbolos que são os números "0" e "1". A sua utilização está ligada à construção física de circuitos, os quais detetam a presença ou não de sinal. A existência de um dado sinal será representado pelo número "1" e, a ausência representada pelo número "0" sendo chamado então de lógica positiva e a designação ao contrário é chamado de lógica negativa.

No sistema de numeração decimal, podemos dividir um número qualquer, da direita para a esquerda, e cada uma das parcelas ocupará uma posição, a qual denominamos por unidade, dezena, centena e etc.

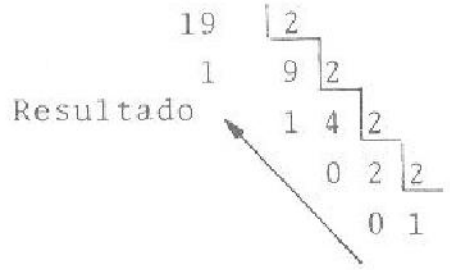
No sistema de numeração binário ocorre o mesmo, porém existem duas divisões particulares, denominadas por BIT e BYTE. A primeira representa um único número, que poderá ser "0" ou "1". A segunda representa um conjunto de oito bits.

1.1 - Conversão da Base Decimal Para Binária

O método de conversão aqui representado é o da divisão sucessiva, o qual consiste em dividirmos um número na base 10 (decimal), por 2 (binário) até que o quociente seja "1". Em seguida, tomamos o quociente mais o resto no sentido indicado nos exemplos a seguir, e o resultado será um número na base 2 (binária) equivalente a um número na base 10 (decimal).

Exemplos:

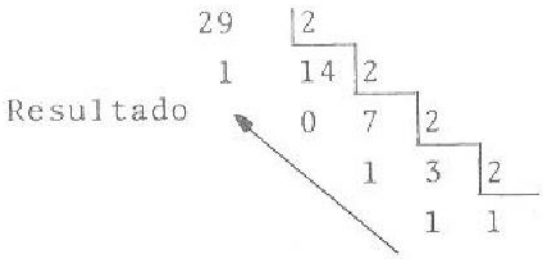
a.) $(19)_{10} = (?)_2$



Obs.: O resultado da conversão é indicado pelo sentido da seta.

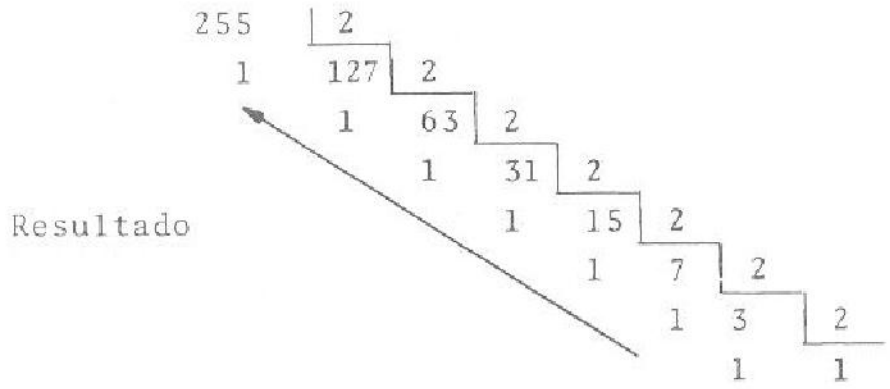
$(19)_{10} = (10011)_2$

b.) $(29)_{10} = (?)_2$



$(29)_{10} = (11101)_2$

c.) $(255)_{10} = (?)_2$



$(255)_{10} = (11111111)_2$

1.2 - Conversão da Base Binária para Decimal

Este método consiste em somar os Bits com seus respectivos pesos. Nos exemplos a seguir poderemos acompanhar o procedimento.

Exemplos:

a.) $(1101)_2 = (?)_{10}$

1	1	0	1	
↓	↓	↓	↓	
2^3	2^2	2^1	2^0	← Pesos

$$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 =$$
$$= 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 =$$
$$= 8 + 4 + 0 + 1 = 13$$
$$\therefore (1101)_2 = (13)_{10}$$

b.) $(10011)_2 = (?)_{10}$

1	0	0	1	1
↓	↓	↓	↓	↓
2^4	2^3	2^2	2^1	2^0

Pesos

$$1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 =$$
$$= 1 \times 16 + 0 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1 =$$
$$= 16 + 0 + 0 + 2 + 1 = 19$$
$$\therefore (10011)_2 = (19)_{10}$$

O método aqui apresentado poderá ser utilizado na conversão de outras bases para a decimal e vice e versa. Das

bases conhecidas, as mais importantes são a base 8, também conhecida por Octal e a base 16, conhecida também por Hexadecimal.

A base 8 é composta pelos algarismos de "0" a "7" e a 16 pelos algarismos de "0" a "9" mais as letras A, B, C, D, E e F, interando assim 16 diferentes símbolos.

1.3 - Conversão da Base Decimal para Hexadecimal

Exemplos:

a.) $(125)_{10} = (?)_{16}$

Obs.:

Na base 16 temos os caracteres A, B, C, D, E e F, os quais correspondem na base 10 aos números 10, 11, 12, 13, 14 e 15.

```
125 | 16
 13   7
```

7 → representado por 7
13 → representado por D

$\therefore (125)_{10} = (7D)_{16}$

b.) $(558)_{10} = (?)_{16}$

```
588 | 16
 78   34 | 16
 14    2    2
```

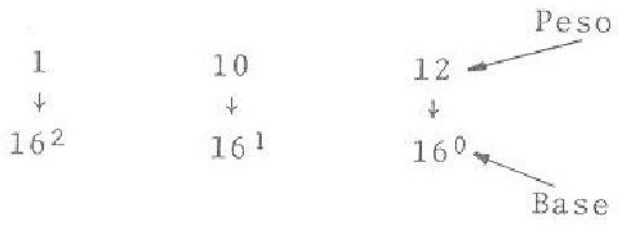
2 → representado por 2
2 → representado por 2
14 → representado por E

$\therefore (558)_{10} = (22E)_{16}$

1.4 - Conversão da Base Hexadecimal para Decimal

a.) $(1AC)_{16} = (?)_{10}$

1 = 1 na base 10
A = 10 na base 10
C = 12 na base 10



$1 \times 16^2 + 10 \times 16^1 + 12 \times 16^0 = 1 \times 256 + 10 \times 16 + 12 \times 1 = 256 + 160 + 12 = (428)$

$\therefore (1AC)_{16} = (428)_{10}$

1.5 - Conversão da Base Binária para Hexadecimal

Na conversão da base 2 para base 16 dividimos o número binário em grupos de quatro Bits, da direita para a esquerda. A seguir o conjunto de 4 Bits obtido poderá ser convertido para a base 16, bastando apenas utilizarmos a tabela 2.1.

BASE 2		BASE 16
0 0 0 0	↔	0
0 0 0 1	↔	1
0 0 1 0	↔	2
0 0 1 1	↔	3
0 1 0 0	↔	4
0 1 0 1	↔	5
0 1 1 0	↔	6
0 1 1 1	↔	7
1 0 0 0	↔	8
1 0 0 1	↔	9
1 0 1 0	↔	A
1 0 1 1	↔	B
1 1 0 0	↔	C
1 1 0 1	↔	D
1 1 1 0	↔	E
1 1 1 1	↔	F

Tabela 2.1 - Tabela de Conversão da Base 2 para a Base 16 e da Base 16 para Base 2.

Exemplos:

a.) $(1011011010)_2 = (?)_{16}$

* Completar com zero as casas faltantes para atingir "4" algarismos.

*									
0	0	1	0	1	1	0	1	1	0	1	0
2				D				A			

$\therefore (1011011010)_2 = (2DA)_{16}$

b.) $(11010001111010101)_2 = (?)_{16}$

0	0	0	1	1	0	1	0	0	0	1	1	1	1	0	1	0	1	0	1
1				A				3				D				5			

$\therefore (11010001111010101)_2 = (1A3D5)_{16}$

1.6 - Conversão de Base Hexadecimal para Binária

Para conversão da base 16 para base 2 pegamos cada número da base 16 e pela tabela (2.1) acha-se o seu correspondente número em binário.

Exemplos:

a.) $(7AD12E)_{16} = (?)_2$

7	A	D	1	2	E
0111	1010	1101	0001	0010	1110

$\therefore (7AD12E)_{16} = (011110101101000100101110)_2$

b.) $(FC71A6)_{16} = (?)_2$

F	C	7	1	A	6
1111	1100	0111	0001	1010	0110

$\therefore (FC71A6)_{16} = (111111000111000110100110)_2$

ÍNDICE REMISSIVO

ÍNDICE DE FIGURAS

BIBLIOGRAFIA

1. Livros

BARDEN JR., Wiliam - *Z80 Microcomputer Design Projects*, 1ª Edição, New York, Howrd W. Saws & CO., INC, 1982.

CIARCIA, Steve - *Buid Your Own Z80 Computer*, 1ª Edição, California, McGraw-Hill Book Company, Berkeley, 1981.

CIPELLI, Antonio M.Vicari e SANDRINI, Waldir - *Teoria e Desenvolvimento de Projetos e Circuitos Eletrônicos*, 6ª Edição, São Pauló, Livros Érica Editora Ltda, 1982

FREGNI, Edson e LANGDON JR., Glen G - *Projetos de Computadores Digitais*, 2ª Edição, São Paulo, Editora Edgard Blücher Ltda, 1979.

GLARSEN, David, TITUS, Christopher A., e TITUS, JONATHANA A.- *Microcomputer Analog Convert Software and Hardware Interfacing*, 1ª Edição, New York, Howrd W. Sams & CO., INC, 1981.

LASKOUSKI, Lester P. e TOCCI, Ronald J. - *Microprocessor and Microcomputers - Hardware and Software*, 1ª Edição, New York, McGraw-Hill Book Company, 1980.

MILLMAN - HALKIAS - *Integrated Eletronics*, 1ª Edição, New York, McGraw-Hill Book Company, 1981.

NICHOLS, Joseph C., NICHOLS, Elizabeth A. e RONY, Peter R. - *Z80 Microprocessor Programming & Interfacing*, Vol. 1 e Vol. 2, 1ª Edição, New York, Howrd W.Sams & CO., INC, 1981.

STOUT, David F. - *Microprocessor Applications*, 1ª Edição, New York, McGraw-Hill Book Company, 1982.

ZUFFO, João Antonio - *Sistemas Eletrônicos Digitais*, Vol. 1 e Vol. 2. 1ª Edição, São Paulo, Editora Edgard Blücher Ltda, 1976.

2. Manuais

Memory - Mostek, 1981.

The TTL Data Book - Texas Instruments, 1980.

Z80 Microcomputer Data Book - Mosket, 1981.

Z80 Technical Manual - Mosket, 1981.

Trabalhos gráficos:
GRAFICA EDITORA LTDA.
Rua Dr. Hinácio da Costa, 1-A - São Paulo

PUBLICAÇÕES EM ELETRÔNICA

Teoria e Desenvolvimento de Projetos de Circuitos Eletrônicos

Diodos, Transistores de Junção, FET, MOS, UJT, LDR, NTC, PTC, SCR, Transformadores, Amplificadores Operacionais e suas aplicações em Projetos de Fontes de Alimentação, Amplificadores, Osciladores, Osciladores de Relaxação e outras.
7.a Edição, 580 páginas.
Autores: Engos. Cipelli / Sandrini

Teoria e Processo de Desenvolvimento em Eletrônica

Estudo e Associação de Bipolos Passivos e Ativos, Circuitos Ressonantes, Aparelhos de Medidas, Válvulas, Semicondutores, Leis de Kirchhoff, Teoremas de Thevenin e Norton, Osciloscópio e outros. 2.a Edição, 259 páginas.
Autor: Eng.o Sidnei David

Microprocessadores 8080 e 8085 - Hardware - Vol. 1

Memórias RAM, ROM, PROM e EPROM, o 8224, 8228, 8080, 8085, 8255 e 8253, suas aplicações e montagem de um microprocessador. 3.a Edição, 138 páginas

Microprocessadores 8080 e 8085 - Software - Vol. 2

Estudo das instruções dos microprocessadores 8080 e 8085, Fluxogramas, iniciação a programação e desenvolvimento de programas com a utilização dos microprocessadores 8080 e 8085. 3.a Edição, 202 páginas.
Autor: Eng.o Antonio Carlos José Franceschini Visconti

Elementos de Eletrônica Digital

Iniciação a Eletrônica Digital, Álgebra de Boole, Minimização de Funções Booleanas, Circuitos Contadores, Decodificadores, Multiplex, Demultiplex, Display, Registradores de Deslocamento, Desenvolvimento de Circuitos Lógicos, Circuitos Somadores/Subtratores e outros. 5.a Edição, 504 páginas.
Autores: Capuano / Idoeta.

Amplificador Operacional

Ideal e Real, em componentes discretos, Realimentação, Compensação, Buffer, Somadores, Detetor de Picos, Integrador, Gerador de Sinais, Amplificadores de Audio, Modulador, Sample-Hold, etc.

Possui cálculos e projetos de circuitos e salienta cuidados especiais.

1.a Edição, 269 páginas.
Autores: Eng.o Roberto Antonio Lando / Eng.o Serg Rios Alves

Basic para Computadores Pessoais

Apresentação da Linguagem, com desenvolvimento detalhado das instruções, com uma série de exercícios resolvidos e propostos (com respostas), jogos e programas aplicativos. Linguagem simples e de grande aceitação pelos aficcionados no ramo dos microcomputadores.

1.a Edição, 267 páginas.
Autor: Arsonval Fleury Pereira