QUESTÕES (EXP.Nº 6)

Responda sucintamente:

1) Como se faz a contagem de pinos de um CI TTL ?

2) Que pinos correspondem normalmente ao V_{CC} e ao GND? Cite 2 exceções.

3) Que é FF sensível ao nível ?

4) Que é FF sensível à borda ?

5) O que é circuito "debouncer" para chaves mecânicas ?

6) Escreva em bases Octal e Hexadecimal os números binários:

   (a) 1 0 0 1 0 1 1 1 1

   (b) 1 1 0 0 1 0 0 1 0

7) O que é barramento? Exemplifique com um circuito simples.

8) O que é porta com saída de três estados ?

9) Por quê eles são usados no sistema de transferência de conteúdos entre registros no circuito usado nesta experiência (Circuito de Teste Nº 1) ?

10) Observe que no sistema dado (Circuito de Teste Nº 1) é usado elemento "tri-state" apenas quando a saída de alguma fonte é conectada ao barramento. Quando se conecta alguma entrada ao barramento o mesmo não é usado. Por quê ?

CAPÍTULO-8

MEMÓRIA ROM (PARTE-A)
MÚSICA ELETRÔNICA (PARTE-B)

EXP. Nº 7

8.1 - PARTE TEÓRICA (A)

8.1.1 - INTRODUÇÃO

Neste capítulo apresentamos o conceito de memórias. Dentro de um processador digital, a função da memória é armazenar informações na forma de palavras digitais. Tais palavras podem ser números de operandos, instruções de máquina, códigos de entrada e saída em um computador ou então são apenas dados em vários tipos de equipamentos.

Numa primeira diferenciação, as memórias podem ser classificadas quanto ao modo de acesso de seu conteúdo em |16|:

a) Memórias de acesso sequencial;
b) Memórias de acesso aleatório (RAM - "Random-Access Memory");
c) Memórias somente para leitura (ROM - "Read-Only Memory").

Nesta experiência nos deteremos mais sobre a memória ROM.

Nas memórias de acesso sequencial as palavras são escritas ou lidas em sequência. Como exemplo, temos as fitas de papel, os cartões perfurados, os cartões magnéticos e as fitas magnéticas. Assim, se a $k_{-ésima}$ palavra está sendo lida, a (k+n)-ésima palavra só estará disponível para leitura após "n" passos. O disco magnético usado em computadores para armazenar programas de "software" é um tipo de memória em que o acesso não é estritamente sequencial mas é sequencial por partes. Do que foi exposto, nota-se que a limitação do tipo de memória sequencial é o tempo de acesso a uma determinada posição de memória. A

vantagem em relação aos outros tipos de memória é o custo, tendo larga aplicação nos casos em que a leitura é feita na mesma ordem de escrita como por exemplo no armazenamento de programas em cartões.

Nas memórias RAM, o tempo de acesso para escrita e o tempo de acesso para leitura são quase os mesmos para todas as posições. Através de um código de endereçamento podemos ter acesso a qualquer posição de memória sem passar pelas demais, donde o nome. Como exemplo, temos a memória principal de um computador que pode ser constituída de núcleos magnéticos. Memórias RAM são disponíveis também na forma de CI's fabricados com semicondutores.

As memórias ROM diferem das RAM pelo fato de não permitirem escrita quando em funcionamento em tempo real, isto é, no circuito a que foi destinada, donde o nome dado. O conteúdo da memória ROM construído com semicondutores é pré-programado pelo fabricante, muitas vezes, de acordo com o pedido específico do usuário e geralmente não pode ser modificado pelo mesmo. As principais características da memória ROM são:

a) Para cada endereço, existe apenas um dado ou palavra, naturalmente em código binário;
b) Uma vez programada a memória, o dado é fixo, imutável e pode apenas ser lido;
c) O conteúdo ou dado é "não volátil", isto é, ele não será perdido quando se faz a leitura ou mesmo quando a sua fonte de alimentação é desligada.

Um tipo muito usado de ROM é a PROM ("Programmable Read-Only Memory") em que o conteúdo da memória ROM é programado pelo usuário. Essa programação comumente denominada "queima" da PROM pode ser bastante trabalhosa, pelo fato de exigir um circuito especial para rompimento dos "fusíveis" (diodos) a fim de gravar o conteúdo desejado. Além disso o circuito queimador da PROM pode diferir de fabricante para fabricante ainda que seja para CI's de mesmo nome.

Outro tipo de ROM é a EPROM ("Erasable Programmable Read-Only Memory") que permite a reprogramação pelo usuário. Para cada nova programação o usuário deve apagar o conteúdo anterior deixando a EPROM em condições de receber o novo dado.

As EPROM's podem ainda ser classificadas quanto à maneira de apagar o seu conteúdo em:

a) UVEPROM ("Ultraviolet EPROM")
b) E^2 PROM ("Electrically EPROM")

No caso da UVEPROM a programação do conteúdo é feita eletricamente, enquanto que o apagamento é feito pela exposição à luz ultravioleta. Deve-se frisar que o apagamento é integral não se podendo escolher uma posição de memória determinada para se apagar. Além disso, o tempo necessário para apagamento pode ser relativamente longo, por exemplo para o 2716 que é uma UVEPROM de 16 k (2K x 8) esse tempo é de 15 a 20 minutos usando-se uma lâmpada ultravioleta de 12.000 $\mu W/cm^2$, com uma dosagem de 15 $W.seg/cm^2$, sendo o comprimento de onda de 2537Å.

No caso da E^2PROM tanto a programação como o apagamento do conteúdo são feitas eletricamente. Além disso o apagamento não precisa ser integral. Por exemplo, para o 2816 que é uma E^2PROM de 16 K (2Kx8) é possível escolher e apagar ou escrever qualquer "byte" (palavra binária de 8 bits) em 10 ms sem afetar o restante do conteúdo da memória. Todo o conteúdo da memória também pode ser apagado em 10 ms. Logo quanto à utilização a E^2PROM é mais versátil que a UVEPROM.

8.1.2. - MEMÓRIAS SEMICONDUTORAS

Os três tipos de memória citadas podem ser implementadas usando-se materiais semicondutores. Porém, deve-se frisar inicialmente que não é propósito deste livro fornecer pormenores de construção dessas memórias. Daremos apenas noções de aspecto geral, enfatizando mais aquelas referentes as ROM's.

8.1.2.1. - MEMÓRIAS DE ACESSO SEQUENCIAL E RAM

As memórias de Acesso Sequencial usando CI's podem ser implementadas com registradores de deslocamento. Por outro lado, as RAM's contêm em sua configuração Flip-Flops, além de outras circuitarias. Ambos os tipos podem fazer uso da tecnologia MOS. Além disso, neste caso, essas memórias podem ser classificadas em memórias estáticas e dinâmicas. As estáticas podem trabalhar com relógio desde DC (frequência zero) até uma frequência máxima. As dinâmicas não podem operar em DC mas sim a partir de uma frequência de relógio mínima até uma máxima. Isso resulta do fato de que os bits são guardados na forma de carga armazenada em um capacitor parasita existente na porta (G) do transistor MOS. Esse capacitor é da ordem de 0,5 pF e não pode segurar um valor aproveitável de carga além de cerca de 1 ms se o elemento a ser alimentado é também um MOS. Naturalmente, se for alimentado um transistor bipolar esse tempo irá diminuir bastante. Logo, a informação deve ser usada dentro desse intervalo (1 ms). Caso contrário, a informação para não se perder deve ser recuperada periodicamente. Essa recuperação é denominada "refresh" da memória (refrescamento).

Deve-se também ressaltar que pela própria natureza dessas memórias elas perdem a informação, quando a fonte de alimentação é desligada. Logo são "memórias voláteis".

8.1.2.2. - MEMÓRIAS ROM

Já foi dito que as memórias ROM são "não voláteis". Na verdade, isso acontece porque a ROM é um circuito combinacional como veremos a seguir.

A ROM é um codificador. No entanto, o CI comercial que contém a ROM já traz junto um decodificador como mostra a Fig.8.1.

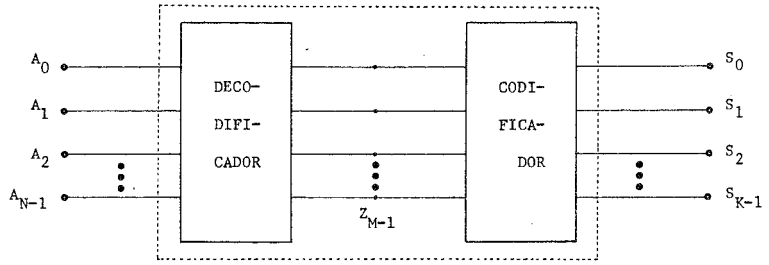


Fig.8.1 - ROM com decodificador.

O decodificador possui N entradas ($A_0,\ldots, A_{N-1}$) e $M=2^N$ saídas ($Z_0,\ldots, Z_{M-1}$). Ex: N=5 M=32 saídas. Para cada condição da entrada, apenas uma saída é "1" sendo que todas as demais são "0". É claro que pode ser o contrário. A Tab.8.1.(a) mostra a tabela verdade do decodificador de N entradas. A Tab.8.1.(b) mostra um exemplo para N = 2.

$A_{N-1}\cdots A_1\ A_0$	$Z_{M-1}\cdots Z_3\ Z_2\ Z_1\ Z_0$
0 ... 0 0	0 ... 0 0 0 1
0 ... 0 1	0 ... 0 0 1 0
0 ... 1 0	0 ... 0 1 0 0
0 ... 1 1	0 ... 1 0 0 0
⋮ ⋮ ⋮	⋮ ⋮ ⋮
1 ... 1 1	1 ... 0 0 0 0

(a)

$A_1\ A_0$	$Z_3\ Z_2\ Z_1\ Z_0$
0 0	0 0 0 1
0 1	0 0 1 0
1 0	0 1 0 0
1 1	1 0 0 0

Ex: N = 2
M = 4

(b)

Tab.8.1 - Tabela Verdade do Decodificador contido na ROM.

O codificador possui então M entradas ($Z_0,\ldots, Z_{M-1}$) e K saídas ($S_0,\ldots, S_{K-1}$). Não existe uma relação obrigatória entre M e K, ou seja K pode ser qualquer valor inteiro. As saídas $S_0,\ldots, S_{K-1}$ são funções das palavras que se desejam armazenar. As entradas ($A_0,\ldots, A_{N-1}$) do decodificador são chamadas bits de endereço. A cada endereço ($A_{N-1}\ldots A_0$) corresponde uma palavra de saída da ROM ($S_{K-1},\ldots, S_0$) que é aquela que foi armazenada.

Exemplo 8.1: Seja N=2. Logo pode-se armazenar $M=2^2=4$ palavras. Suponha que cada palavra P_i (i = 1, 2, 3, 4) tenha 5 dígitos, isto é, $P_i = S_4\ S_3\ S_2\ S_1\ S_0$ e que sejam:

$P_1 = 0\ 0\ 0\ 1\ 0$

$P_2 = 1\ 0\ 0\ 1\ 0$

$P_3 = 0\ 1\ 0\ 0\ 1$

$P_4 = 1\ 1\ 1\ 1\ 1$

Suponha que se deseja armazená-las nos endereços $A_1\ A_0$ indicados a seguir na Tab.8.2.

$A_1\ A_0$	$S_4\ S_3\ S_2\ S_1\ S_0$	$Z_3\ Z_2\ Z_1\ Z_0$
0 0	0 0 0 1 0	0 0 0 1
0 1	1 0 0 1 0	0 0 1 0
1 0	0 1 0 0 1	0 1 0 0
1 1	1 1 1 1 1	1 0 0 0

Tab.8.2 - Tabela de associação dos endereços com as palavras.

Observe que cada saída S_i (i= 0,...,4) pode ser obtida diretamente da tabela verdade. Porém o novo propósito aqui é usar a ROM.

Uma maneira de se realizar a ROM é com diodos. Neste exemplo a configuração da ROM é aquela mostrada na Fig.8.2.

Nessa figura as portas de saída "C" são seguidores enquanto que um dos terminais dos resistores "R" está aterrado. Pode-se ver que se a linha Z_m na saída do decodificador estiver no nível "0" então os diodos "D" ligados a essa linha não conduzem. O diodo "D" só conduz se a linha à qual estiver conectada assume o nível lógico "1". Se um determinado diodo conduz, então a linha S_k ligada a esse diodo adquire o

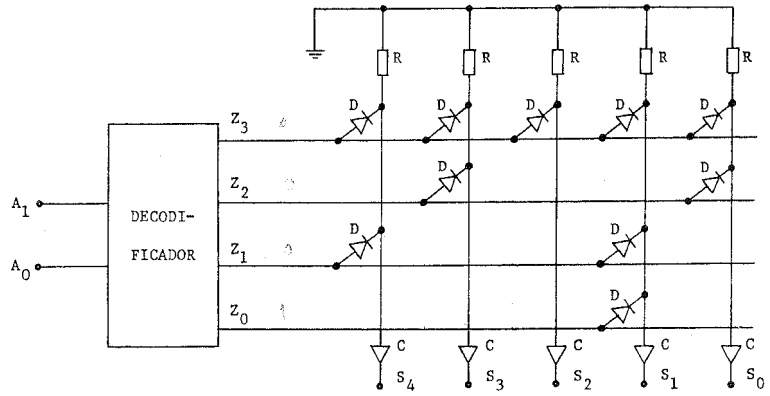


Fig.8.2 - Implementação da ROM usando diodos.

nível "1" e tem-se a saída S_k igual a "1". Dessa forma para se armazenar por exemplo a palavra P_1 = 00010 no endereço A_1A_0 = 00 coloca-se um diodo entre a linha Z_0 e a linha S_1. Assim, quando A_1A_0 00 tem-se $Z_3\ Z_2\ Z_1\ Z_0$ = 00010 e consequentemente tem-se na saída a palavra P_1 desejada. Da mesma forma se armazenam as demais palavras como se pode ver nessa Fig.8.2.

Por outro lado ao invés da ROM o usuário pode preferir a outra versão que é a PROM.

No caso da PROM tem um diodo em série com um fusível em todos os cruzamentos das linhas decodificadas $Z_{M-1},\ldots, Z_0$ com as linhas de saída $S_{k-1},\ldots, S_0$, conforme mostra a Fig.8.3.(a). Nessa figura "F" indica o fusível.

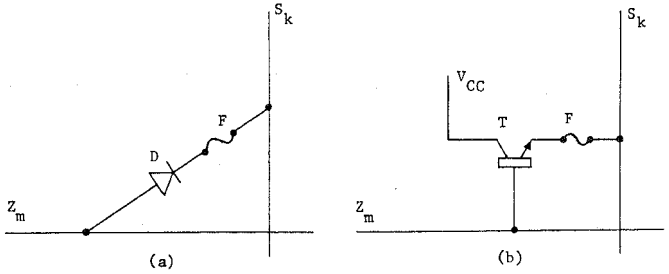
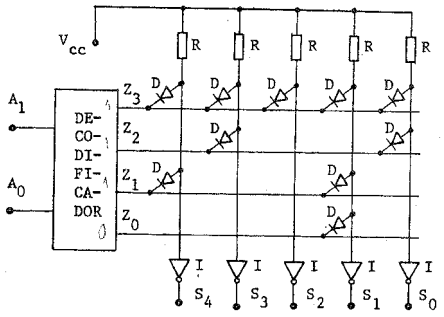


Fig.8.3 - Detalhes das ligações dos diodos e fusíveis na PROM.

O fusível "F" é rompido ou não dependendo de se desejar um nível "0" ou "1", respectivamente, no caso da Fig.8.2. Além disso, a fim de evitar que a corrente que circula pelo diodo venha direto das saídas do decodificador, na prática usa-se um transistor como mostra a Fig.8.3.(b). Nesta situação a corrente é fornecida pela fonte V_{CC}. Além disso, uma vez que os coletores dos transistores estão ligados ao V_{CC} enquanto que a base à linha comum "Z_m", é usual o aproveitamento de transistores multi-emissor.

Exemplo 8.2: Suponha, no exemplo anterior, que o decodificador usado seja tal que apenas uma saída fique "0" e todas as demais em "1" para cada condição de entrada. A Fig.8.4.(a) mostra o circuito neste caso e a Fig.8.4.(b) a tabela verdade. Observe que os resistores "R" estão ligados ao V_{CC} e na saída há inversores "I" e não seguidores. Além disso, os diodos "D" conduzem quando a linha Z_m a qual estão ligados está no nível "0". Se o nível é "1", não conduzem. Por outro lado, quando o diodo "D" ligado a linha S_k conduz, estabelece-se nessa linha um nível lógico "0", sendo que na saída do inversor correspondente tem-se então S_k=1.



(a)

A_1 A_0	Z_3 Z_2 Z_1 Z_0	P_i
0 0	1 1 1 0	P_1
0 1	1 1 0 1	P_2
1 0	1 0 1 1	P_3
1 1	0 1 1 1	P_4

(b)

Fig.8.4 - ROM para decodificador com lógica invertida.

Voltando agora à Tab.8.2 do Exemplo 2.1 pode-se ver que as saídas S_4, S_3, S_2, S_1 e S_0 podem ser implementadas como funções booleanas das entradas A_1 e A_0. De fato tem-se:

$$S_4 = A_0;\quad S_3 = A_1;\quad S_2 = A_1A_0;\quad S_1 = \overline{A_1}\overline{A_0};\quad S_0 = A_1 \qquad (8.1)$$

Logo, do que foi exposto pode-se inferir que a ROM com decodificador pode realizar a implementação de qualquer tabela verdade , desde que o número de entradas e saídas sejam suficientes. É por esse motivo que a ROM é usada para implementar funções complexas como multiplicação, divisão, raiz quadrada, funções trigonométricas (sin x , cos x, etc), funções logarítmicas, etc, onde devido a quantidade de portas necessárias no caso de implementação direta é preferível a ROM por ser mais econômico em tamanho, peso e custo.

Cabe ainda ressaltar que a ROM embora receba o nome de memória não é um circuito sequencial mas sim combinacional já que não contém circuitos com memória no sentido dado na experiência número 2. Por outro lado, a RAM é sequencial pois contém Flip-Flops em sua estrutura.

8.1.3. - CIRCUITO DE UM "PEQUENO PROCESSADOR E SEQUENCIADOR"

Nesta experiência teremos oportunidade de verificar o funcionamento de ROM's implementadas com diodos e decodificadores. Os decodificadores usados são CI's. Essas ROM's servem para armazenar as instruções para operação do "pequeno circuito processador sequenciador" assim chamado neste laboratório porque ele interpreta e processa a sequência de instruções contidas na ROM em forma de palavras binárias. Essa sequência de instruções será chamada de "programa" na tentativa de estabelecer um paralelo com os programas usados no computador. Na verdade, esta experiência tem como um dos objetivos introduzir conceitos introdutórios relativos ao microprocessador, através de uma idéia bastante simplificada do mesmo. Essa idéia é aqui denominado de "pequeno processador e sequenciador".

Em experiência anterior foram apresentados os aspectos de natureza geral de um sistema de computação clássica na concepção arquitetônica de "Von Neumann".

Dentro do contexto apresentado, fez-se então um estudo suscinto e funcional de cada uma das principais unidades ou secções que compõem tal sistema. Deu-se ênfase especial ao estudo pormenorizado da Unidade Lógica e Aritmética (ULA) a qual foi então objeto da parte experimental. Ainda naquela experiência foi definido e apresentado o conceito de barramento que é o meio físico responsável pela transferência de dados e informações entre as várias unidades. Dentro destas unidades citou-se também os diversos tipos de registros existentes. Ressaltou-se então a importância do ciclo de memória que consiste na busca de dados e instruções, na interpretação das instruções e na execução das mesmas. Para tanto a unidade de controle faz uso dos vários registros (acumulador, de instrução, etc) existentes na UCP (Unidade Contral de Processamento). Neste capítulo procuramos mostrar os conceitos apresentados elaborando um "pequeno sequenciador e processador de instruções", visando também apresentar aspectos práticos de "hardware" e de "software". Na linguagem de computação o termo "hardware" engloba os elementos de circuitaria e o "software" os de programação.

Já vimos que o Flip-Flop é o elemento unitário de memória e como tal serve para guardar um dígito binário de informação. A fim de se guardar mais de um dígito usa-se registradores construídos com vários Flip-Flops. Tais registros podem então armazenar uma sequência de bits ou uma palavra binária de informação. Se a palavra possui 8 bits ela é denominada de "byte". Por outro lado para se armazenar uma sequência de palavras pode-se usar memórias propriamente didas como por exemplo a RAM.

Também já foi visto que tanto o computador bem como o mini e o micro computador possuem uma secção denominada Unidade de Memória. Nesta secção distinguem-se a memória principal e as memórias secunrias. As secundárias são o disco magnético, a fita de papel, a fita magnética, a fita cassete, o cartão de papel e o cartão magnético. A ROM e a PROM também podem ser usadas como memórias secundárias. A memória principal de um computador pode ser de núcleo de ferrite magnetizado. No microcomputador tem-se usado quase que exclusivamente a RAM de semicondutores. Quanto a função da memória principal, é nela que se executa o processamento de um programa. Normalmente, um programa é constituído de duas partes que são:

a) Instruções;

b) Dados.

Tanto as instruções como os dados devem estar armazenados na memória principal sob a forma de palavras binárias, ou seja, em linguagem de máquina, obedecendo determinadas regras para a escrita da palavra. Tais regras que visam a interpretação correta pela máquina das instruções desejadas, naturalmente podem variar com o tipo, marca, modelo e fabricante do equipamento.

Deve-se frisar que o programa para ser executado tem que estar carregado na memória principal. A maioria dos sistemas computacionais dispõem de apenas uma memória principal, associada a uma UCP. Aqueles que possuem mais de uma UCP realizam o chamado multiprocessamento. Em um sentido mais restrito, multiprocessamento é uma configuração de sistema na qual mais de um processador central do mesmo tipo operam em paralelo.

Deve-se ainda dizer que as memórias secundárias não têm capacidade de executar programas, porém são importantes para outras funções.

Finalmente,cabe-nos constatar mais uma vez que muitos outros aspectos referentes a memórias e a computadores poderiam ser abordados, porém isso tornaria a explanação muito longa, para os propósitos deste curso, que neste capítulo visa apenas uma introdução aos "microprocessadores" e "microcomputadores". Conforme foi dito no capítulo anterior o microprocessador tem sido essencialmente a unidade central de processamento do chamado microcomputador sendo que este possui ainda unidades de memória (ROM e RAM) e interfaces de entrada e saída para diversos periféricos, tais como fitas cassetes e terminais de teletipo ou de vídeo.

8.2 - PARTE EXPERIMENTAL (A)

8.2.1 - MATERIAIS E EQUIPAMENTOS NECESSÁRIOS

(a) Circuitos de Teste, Painel e Fios.

1 Circuito de teste com a montagem da Fig.8.10;
1 Painel de montagens;
Fios de ligação e conexão.

8.2.2 - MONTAGENS NO PAINEL COM O CIRCUITO DE TESTE

a) Usaremos o circuito de teste nº 1 do "pequeno processador e sequenciador" cujo esquema está mostrado na Fig.8.10. A Fig.8.5 mostra um diagrama de blocos simplificado desse esquema.

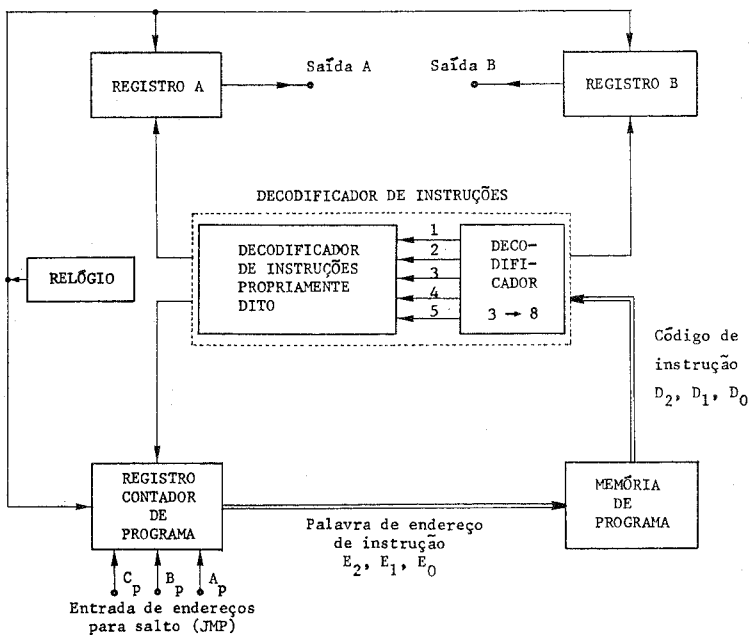


Fig.8.5 - Esquema do circuito processador e sequenciador.

Basicamente este esquema consiste de:

- 2 Flip-Flops JK-MS (7476) representados pelos registros processadores A e B;
- 1 Contador Binário Síncrono (74161) utilizado como registro contador de programa (P) ou como sequenciador de instrução;
- Portas AND, OR e inversores formando o decodificador de instruções propriamente dito. O CI decodificador 3 para 8 (7442) também foi incluído no decodificador de instruções;
- 1 Relógio sendo que usaremos o relógio do painel de montagens;
- 1 ROM de diodos representado pela memória de programa;

b) Identifique então quais elementos ou "chips" da Fig.8.10 constituem cada bloco apresentado na Fig.8.5.

c) As entradas 1, 2, 3, 4 e 5 do decodificador propriamente dito foram escolhidas de tal modo que apenas uma ou nenhuma delas esteja no nível "0",sendo que as demais ou todas no nível "1". Deve-se ressaltar que essa escolha na verdade é arbitrária existindo de fato $32 = 2^5$ possibilidades para se associar as 6 instruções selecionadas. No entanto, o circuito irá variar de acordo com a escolha feita.

A Tab.8.3 mostra essas situações.

1	2	3	4	5	clock	instruções
1	1	1	1	1	⎍⎍	NOP
0	1	1	1	1	⎍⎍	MOV A, B
1	0	1	1	1	⎍⎍	CMA
1	1	0	1	1	⎍⎍	MOV B, A
1	1	1	0	1	⎍⎍	JMP (C_p B_p A_p)
1	1	1	1	0	⎍⎍	STP

Tab.8.3 - Condições nas entradas 1, 2, 3, 4 e 5 que definem a instrução a ser executada.

Nessa tabela, as instruções estão representadas por nomes mnemônicos e serão descritas a seguir.

O circuito processador é tal que a cada pulso de relógio ele executa a instrução correspondente àquela situação estabelecida nas entradas 1 a 5 segundo a Tab.8.3.

As instruções executadas pelo circuito são em número de 6 (seis) e designadas a seguir por um nome mnemônico. São elas:

1) NOP : Não executa nada, porém o contador é incrementado de "1";
2) MOV A, B : Coloca o conteúdo de B em A;
3) CMA : Complementa o conteúdo do registro A;
4) MOV B, A : Coloca o conteúdo de A em B;
5) JMP ($C_pB_pA_p$): O programa pula para o endereço dado pelas entradas C_p B_p A_p se o valor do registrador A for zero;
6) STP : Inibe o clock, isto é, para o programa.

d) Com o painel desligado faça as ligações:

- de alimentação do circuito de teste usando o V_{CC} e o terra do painel;
- das entradas 1, 2, 3, 4 e 5 ligando-as às chaves escolhidas do painel;
- do relógio "CLOCK". Use o relógio do painel;
- das entradas de "CLEAR" ou seja de CLRA, CLRB e CLRP que "resetam" o conteúdo dos registros A, B e P respectivamente;
- das entradas de "preset" ou seja de PRA e PRB que "setam" o conteúdo dos registros A e B, respectivamente;
- das entradas C_p, B_p e A_p que determinam os endereços para saltos (JMP = "jump");
- das saídas A e B que correspondem às saídas dos registros A e B respectivamente. Use os diodos LED's;
- das saídas E_2 E_1 E_0 do contador de programa P. Use o mostrador de 7 segmentos.

e) Teste o circuito verificando a coluna de instruções da Tab.8.3. Indique as instruções apenas pelos nomes mnemônicos "NOP", "MOV A,B", "CMA", "MOV B, A", "JMP" e "STP"

f) É claro que cada uma das condições na Tab.8.3 pode ser analisada no esquema da Fig.8.10.(a).

Como ilustração, é explicada a instrução que corresponde a:

1 2 3 4 5 = 0 1 1 1 1

Assim:

$$\text{Se } B = 0 \rightarrow \begin{cases} J_A = 0 \\ K_A = 1 \end{cases} \rightarrow A=0 \quad (\text{após o } \sqcap\!\!\sqcup) \qquad (8.1)$$

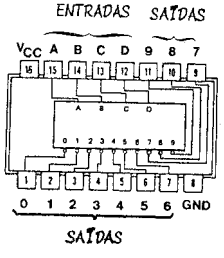
$$\text{Se } B = 1 \rightarrow \begin{cases} J_A = 1 \\ K_A = 0 \end{cases} \rightarrow A=1 \quad (\text{após o } \sqcap\!\!\sqcup) \qquad (8.2)$$

Conclusão:

$$A \leftarrow |B|$$

Ou seja, o conteúdo do registro B passa para o registro A conforme queríamos mostrar.

g) Considere-se agora o esquema da Fig.8.10.(b). Trata-se do decodificador 7442 ("BCD/Decimal Decoder") que possui 4 entradas A, B, C e D, e 10 saídas 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9. A Fig.8.6.(a) mostra o diagrama de pinagem. A tabela verdade deste decodificador é dada na Fig.8.6.(b) a seguir.



(a)

Nº	ENTRADA D	C	B	A	SAÍDA DECIMAL
0	L	L	L	L	H H H H H H H H H
1	L	L	L	H	H L H H H H H H H
2	L	L	H	L	H H L H H H H H H
3	L	L	H	H	H H H L H H H H H
4	L	H	L	L	H H H H L H H H H
5	L	H	L	H	H H H H H L H H H
6	L	H	H	L	H H H H H H L H H
7	L	H	H	H	H H H H H H H L H
8	H	L	L	L	H H H H H H H H L
9	H	L	L	H	H H H H H H H H H
INVÁLIDO	H	L	H	L	H H H H H H H H H
	H	L	H	H	H H H H H H H H H
	H	H	L	L	H H H H H H H H H
	H	H	L	H	H H H H H H H H H
	H	H	H	L	H H H H H H H H H
	H	H	H	H	H H H H H H H H H

(b)

Fig.8.6 - Decodificador 7442 (4 linhas para 10 linhas) |8|.
(a) Diagrama de pinagem.
(b) Tabela Verdade.

Pela tabela nota-se que para cada condição nas entradas A B C D correspondendo desde 0 até 9 em decimal apenas uma das saídas 0 a 9 está baixa sendo que as demais estão no nível alto. Assim essas saídas são usadas para se obter as entradas 1, 2, 3, 4 e 5 da Fig. 8.10.(a), entradas essas que definem as instruções a serem executadas.

Note-se que para selecionar uma de seis instruções bastam três variáveis de controle. Como temos 5 variáveis (1, 2, 3, 4 e 5), a idéia de se usar o decodificador é diminuir o número de variáveis de controle para o mínimo necessário. Essas novas variáveis serão denominadas D_2, D_1 e D_0.

h) Faça então as ligações indicadas a seguir na Fig.8.7. Antes porém, desfaça as ligações das entradas 1, 2, 3, 4 e 5 com as chaves do painel.

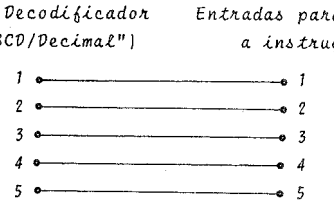


Fig.8.7 - Correspondência escolhida entre as saídas do decodificador e as entradas que definem a instrução.

i) Complete então o quadro a seguir da Tab.8.4 de acordo com a ligação feita no item anterior. Nesta tabela D_2 D_1 D_0 são as entradas do decodificador.

Código da instrução			*Instrução que será decodificada*
D_2	D_1	D_0	*Símbolo Mnemônico*
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Tab.8.4 - Correspondência entre as entradas do decodificador e as instruções a serem executadas.

j) Considere-se o esquema da Fig.8.10.(c). Trata-se da ROM de diodos . Na entrada da ROM temos novamente o CI decodificador 7442 cujo funcionamento já foi descrito. As entradas A, B e C correspondem a E_0 E_1 e E_2 respectivamente sendo que a entrada "D" está aterrada. As linhas de saída do decodificador da ROM foram chamadas de P_0, P_1, P_2, P_3, P_4, P_5, P_6 e P_7. As saídas da ROM correspondem a D_2 D_1 e D_0. Conforme foi visto na parte teórica, coloca-se ou não o diodo nos cruzamentos das linhas P_i(i = 0,..., 7) com as linhas D_j(j = 0, 1, 2) conforme a palavra que se deseja armazenar no endereço E_2 E_1 E_0. Em cada linha do decodificador grava-se então uma palavra de instrução de 3 bits (D_2, D_1 e D_0). A Tab.8.4 apresentada mostra a correspondência entre o código e o símbolo mnemônico da instrução. A Fig.8.8 mostra um esquema simplificado da ROM de diodos.

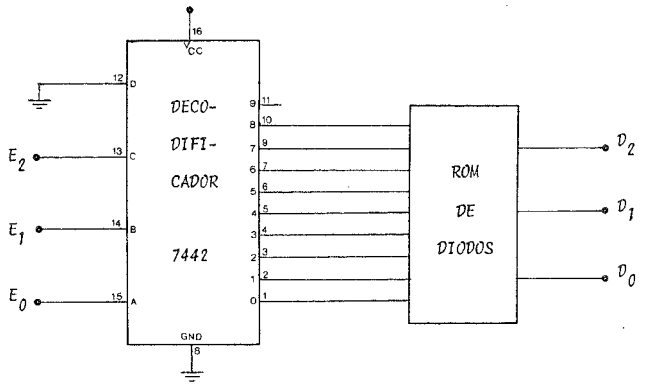


Fig.8.8 - ROM Decodificada

Chamaremos todo o conjunto montado em uma só placa simplesmente de ROM.

k) Observação: Na parte teórica, apresentou-se o conceito de "memória principal" de um sistema de computação. Descreveu-se ainda a experiência anterior que a cada posição de memória associa-se um único número que se identifica como sendo o endereço desta posição de memória. Na fase de processamento, o acesso a qualquer uma destas posições é controlado por um elemento chamado de "sequenciador de posição de memória". Na descrição dada anteriormente o sequenciador foi representado pelo contador de programa ou Registro P.

No circuito experimental desta experiência utiliza-se como contador de programa o CI 74161, que é um contador binário síncrono de 4 bits. Um dos bits não é usado, sendo aproveitado apenas as saídas Q_A, Q_B e Q_C que são suficientes para os nossos propósitos.

As Figs.8.9.(a) e (b) mostram o diagrama de pinagem e de tempo para o contador 74161.

O contador binário síncrono 74161, utilizado como contador de programa (registro P) tem a função de contar sequencialmente a cada pulso de relógio sendo que este pulso coincide com o início de execução de cada nova instrução. O resultado desta contagem "E_2 E_1 E_0" forma o endereço da posição de memória também chamado de "E_2 E_1 E_0" na ROM. Cada endereço corresponde a uma posição de memória da ROM ou seja a uma linha P_i (i = 0, ... , 7) do decodificador. Em cada posição está gravada uma instrução.

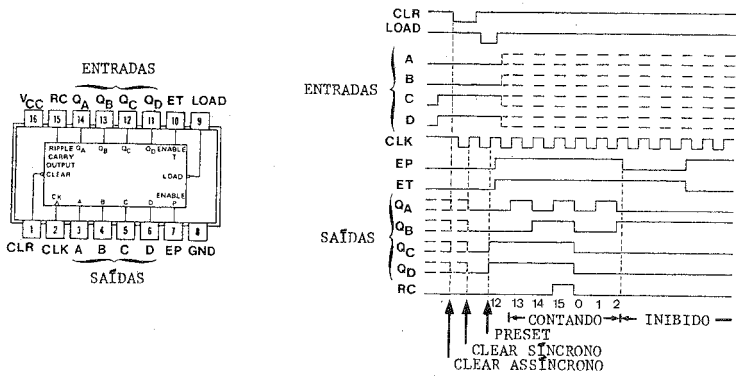


Fig.8.9 - Contador síncrono 74161 |8|.
(a) Diagrama de Pinagem.
(b) Diagrama de Tempo.

A contagem sequencial segue normalmente, a menos que a entrada do terminal "LOAD" no 74161 receba um pulso negativo, o que ocor

re para A = 0 e "4" = 0. Nesta situação "LOAD" sai da condição de inibição e passa à condição de ativação. Como sequência é carregado o endereço $C_p\ B_p\ A_p$, disponível na entrada do contador, o qual é possivelmente o endereço resultante de uma instrução de Salto ("GO TO" "JUMP", "SKIP"). Este endereço é então carregado ("LOADED"), isto é, $E_2\ E_1\ E_0 = C_p\ B_p\ A_p$ e na ROM irá apontar para a correspondente posição de memória, cujo conteúdo $D_2\ D_1\ D_0$ é lido pelo decodificador de instruções. Uma vez decodificado o código de instrução , a contagem segue normalmente, a partir do endereço $C_p\ B_p\ A_p$ que foi carregado.

Deve-se ressaltar que uma vez estabelecido o endereço em $E_2\ E_1\ E_0$ o conteúdo da posição de memória apontada está disponível nas saídas $D_2\ D_1\ D_0$ da ROM e dizemos que tal conteúdo foi lido.

1) Façamos agora uma classificação das instruções do circuito processador apresentado. Complete a Tab.8.5 conforme a Tab.8.4 anterior.

TIPOS DE INSTRUÇÃO		$D_2\ D_1\ D_0$
Instruções de movimento de registros	*MOV A, B* *MOV B, A*	
Instruções lógicas	*CMA*	
Instruções de controle do programa	*JMP* ($C_p\ B_p\ A_p$) *STP*	
Não operação	*NOP*	

Tab.8.5 - Tipos de Instruções.

m) Implementação do Software do processador - Pela Tab.8.5, vemos que não dispomos nesse sequenciador, por exemplo da instrução "CMB", isto é, complementar o conteúdo do registro B. No entanto,podemos realizar essa função "CMB" a partir das instruções disponíveis fazendo um "programa" como o mostrado a seguir.

Programa:

1. Colocar [B] em A
2. Complementar [A]
3. Colocar [A] em B
4. Fim.

Observamos assim que os passos de 1 a 4 realizam a função de complementar o conteúdo do registro B.

A fim de facilitar a programação, a Tab.8.6 a seguir deve ser preenchida. Essa tabela ilustra o princípio de programação do processador estudado em "linguagem de máquina" e em "Assembly" (Montagem). Na coluna 1 desta tabela são dados os endereços (E_2,E_1, E_0) de cada instrução. Na coluna 2, o código de cada uma dessas instruções em palavras binárias. Esse código é chamado de "linguagem de máquina". Na coluna 3, as instruções são escritas na forma mnemônica que é chamada de linguagem Assembly. A sequência dessas instruções constituem o programa.

	Linguagem de Máquina	*Linguagem Assembly*
Endereço da Instrução $E_2\ E_1\ E_0$	*Código da Instrução* $D_2\ D_1\ D_0$	*Instrução*
000	*0 0 1*	*MOV A, B*
001	*0 1 0*	*CMA*
010	*0 1 1*	*MOV B, A*
011	*1 0 1*	*STP*

Tab.8.6 - Exemplo para programação.

O código de instrução $D_2\ D_1\ D_0$ deve ser gravado na linha do decodificador da ROM correspondente ao endereço $E_2\ E_1\ E_0$. Para tanto utiliza-se os diodos conforme foi explanado na parte teórica. Note-se a colocação de diodos nas intersecções que correspondem a D_i igual a zero (i = 0, 1, 2).

n) Teste o programa CMB. Observe os estados de E_2, E_1 e E_0 assim como de D_2, D_1 e D_0 nos diodos LED. Note que D_2, D_1 e D_0 são agora fornecidos pela ROM e não devem estar ligados às chaves de entrada.

o) Proposta de trabalho.
- Programar a função ZERAR A
- Programar a função ZERAR A e B

- Programar a função SETAR A
- Programar um pisca-pisca
- Programar um pulo incondicional.

p) Sugestões para os programas propostos.

ZERAR A

E_2	E_1	E_0	D_2	D_1	D_0	Instrução
0	0	0	1	0	0	JMP (010)
0	0	1	0	1	0	CMA
0	1	0	1	0	1	STP

ZERAR A e B

E_2	E_1	E_0	D_2	D_1	D_0	Instrução
0	0	0	1	0	0	JMP (010)
0	0	1	0	1	0	CMA
0	1	0	0	1	1	MOV B, A
0	1	1	1	0	1	STP

SETAR A

E_2	E_1	E_0	D_2	D_1	D_0	Instrução
0	0	0	1	0	0	JMP (010)
0	0	1	1	0	1	STP
0	1	0	0	1	0	CMA
0	1	1	1	0	1	STP

PULO INCONDICIONAL

E_2	E_1	E_0	D_2	D_1	D_0	Instrução
0	0	0	1	0	0	JMP (100)
0	0	1	0	1	0	CMA
0	1	0	1	0	0	JMP (100)
0	1	1	1	1	1	Conjunto de
	—			—		instr. puladas
1	0	0	1	0	1	STP

PISCA-PISCA

E_2	E_1	E_0	D_2	D_1	D_0	Instrução
0	0	0	0	1	1	MOV B, A
0	0	1	1	1	1	NOP
0	1	0	0	1	0	CMA
0	1	1	1	0	0	JMP (000)
1	0	0	0	1	1	MOV B, A
1	0	1	1	1	1	NOP
1	1	0	0	1	0	CMA
1	1	1	1	0	0	JMP (000)

q) Observações:

q.1) No pisca-pisca as instruções "NOP" são usadas para simetrizar os tempos em que o mesmo permanece em cada estado. Além disso, independente das condições iniciais, após alguns pulsos de relógio o pisca-pisca segue uma sequência pré-estabelecida.

q.2) A ROM de diodos serve como "memória principal" do circuito sequenciador apresentado. Porém o conteúdo da ROM é imutável em tempo real de operação. É por isso que a memória principal de um computador ou microcomputador é constituída de memória de acesso aleatório.

q.3) Dessa forma o circuito esquematizado na Fig.8.5 tem aspectos de um microcomputador rudimentar em que:

a) A memória ROM representa a unidade de memória;

b) As portas lógicas (E, OU, NE) e o contador P representam a unidade de controle;

c) Os registros A e B representam a unidade lógica e aritmética;

d) E os mostradores visuais representam os periféricos de saída.

r) Teste os programas sugeridos. Faça comentários.