NOTAS DE AULA

06

MICROCONTROLADOR 8051

VOL. 06

11 APLICAÇÕES COM O 8051.

11.1 Interfaceamento de um Display de Crital Líquido (LCD) ao 8051

Nos últimos anos o LCD tem obtido relativo sucesso em vendas quando comparado aos LED's (de 07 segmentos ou LED's multisegmentais) em função do declínio de seu preço, da habilidade de exibir números, caracteres e gráficos. Incorporação do circuito de refresh no próprio LCD, pois no caso dos LED's, em geral, o refresh é realizado pela CPU. Em última análise, é um dispositivo de fácil programação tanto para gráficos como para caracteres.

Descrição dos pinos:
O LCD utilizado neste capítulo tem 14 pinos e suas funções são apresentadas na tabela abaixo.

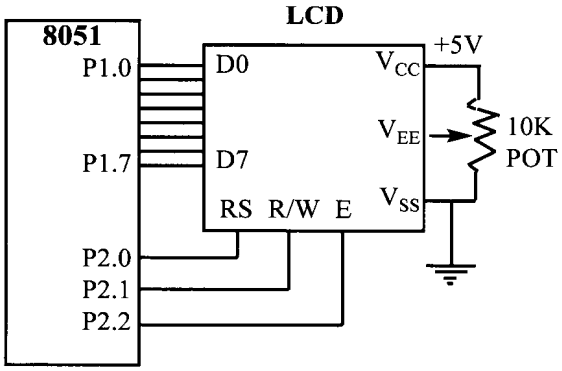
Pino	Símbolo	I/O	Descrição
1	Vss	---	Terra
2	Vcc	---	+5V
3	Vee	---	Fonte para controle do contraste
4	RS	I	Register Select: RS = 0 → para selecionar registrador de instruções de comando, RS = 1 → para selecionar registrador de dados
5	R/W	I	Leitura/Escrita: R/W = 0 para escrita e R/W = 1 para leitura
6	E	I	Habili tar
7 – 14	DB0 – DB7	I/O	O barramento de dados de 8 bits

RS: Register Select. Existem dois registradores internos no LCD e o pino RS é usado para sua seleção. Se RS=0, o registrador de código e comando das instruções é selecionado, podendo-se utiliza-lo para o usuário enviar comandos, tais como, limpar o display, etc. Se RS=1 o registrador de dados é selecionado, permitindo ao usuário enviar os dados a serem exibidos no LCD. Para exibir letras e números, nós enviamos os códigos ASCII das letras A-Z, a-z e números 0-9 para os pinos D0-D7 enquanto RS=1.

Existem também instruções específicas que podem ser enviadas ao LCD para limpa-lo ou reiniciar o cursor usando as instruções listadas na tabela abaixo.

Código (Hex.)	Comando para o registrador de instruções do LCD
1	Limpar o display
2	Retornar o cursor para o início
4	Decrementar o cursor (deslocar o cursor para a esquerda)
6	Incrementar o cursor (deslocar o cursor para a direita)
5	Deslocar a exibição para a direita
7	Deslocar a exibição para a esquerda
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor on
F	Display on, cursor piscando
10 14	Deslocar a posição do cursor para a esquerda Deslocar a posição do cursor para a direita
18	Deslocar o display todo para a esquerda
1C	Deslocar o display todo para a direita
C0	Forçar o cursor para iniciar na segunda linha
38	2 linhas e matrix 5x7

Para enviar qualquer comando da Tabela anterior ao LCD, é necessário que RS = 0. Para dados, fazemos RS = 1. Então enviar um pulso alto para baixo para o pino E para habilitar o latch interno do LCD. Isto pode ser implementado através do código a seguir:




```
; chamar uma rotina de atraso antes de enviar o próximo comando/dado
; P1.0 – P1.7 → são conectados aos pinos de dados D0-D7
; P2.0 é conectado ao pino RS do LCD
; P2.1 é conectado ao pino R/W do LCD
; P2.2 é conectado ao pino E do LCD
ORG
        MOV  A, #38H        ; comando 38H → LCD: 2 linhas, matriz 5x7
        ACALL     COMESCR
        ACALL     DELAY
        MOV  A, #0EH        ; display on, cursor on
        ACALL     COMESCR
        ACALL     DELAY
        MOV  A, #01         ; limpa LCD
        ACALL     COMESCR
        ACALL     DELAY
        MOV  A, #06H        ; desloca cursor para a direita0
        ACALL     COMESCR
        ACALL     DELAY
        MOV  A, #84H        ; cursor linha 1, posição 4
        ACALL     COMESCR
        ACALL     DELAY
        MOV  A, #'G'        ; exibe a letra G
        ACALL     DADOESCR
        ACALL     DELAY
        MOV  A, #'R'        ; exibe a letra R
        ACALL     DADOESCR
NOVO:   SJMP  NOVO
COMESCR:                    ; sub-rotina para enviar comandos ao LCD
        MOV  P1, A          ; copia regs. A para a P1
        CLR  P2.0           ; RS = 0 para comando
        CLR  P2.1           ; R/W = 0 para escrever
```

```
            SERTB     P2.2        ; E = 1 para pulso alto
            CLR       P2.2        ; E = 0 para pulso alto para baixo
RET
DADOESCR:                         ; sub-rotina para escrever dados no LCD
            MOV  P1, A
            SETB P2.0             ; RS = 1 para dados
            CLR  P2.1
            SETB P2.2
            CLR  P2.2
RET
DELAY:      MOV  R3, #50          ; sub-rotina de atraso
AQUI2:      MOV  R4, #255
AQUI:       JNZ  R4, AQUI
            DJNZ R3, AQUI2
RET
END


; outra solução;

; monitorar um flag antes de enviar um comando ou dado ao LCD

; P1.0 – P1.7 → são conectados aos pinos de dados D0-D7
; P2.0 é conectado ao pino RS do LCD
; P2.1 é conectado ao pino R/W do LCD
; P2.2 é conectado ao pino E do LCD
ORG
            MOV  A, #38H          ; comando 38H → LCD: 2 linhas, matriz 5x7
            ACALL     COMANDO
            MOV  A, #0EH          ; display on, cursor on
            ACALL     COMANDO
            MOV  A, #01           ; limpa LCD
            ACALL     COMANDO
            MOV  A, #06H          ; desloca cursor para a direita0
            ACALL     COMANDO
            MOV  A, #86H          ; cursor linha 1, posição 6
            ACALL     COMANDO
            MOV  A, #'G'          ; exibe a letra G
            ACALL     DADOESCR
            MOV  A, #'R'          ; exibe a letra R
            ACALL     DADOESCR
NOVO:       SJMP NOVO
COMANDO:                          ; sub-rotina para enviar comandos ao LCD
            ACALL     PRONTO      ; o LCD está pronto
            MOV  P1, A            ; copia regs. A para a P1
            CLR  P2.0             ; RS = 0 para comando
            CLR  P2.1             ; R/W = 0 para escrever
            SERTB     P2.2        ; E = 1 para pulso alto
            CLR       P2.2        ; E = 0 para pulso alto para baixo
```

```
RET
DADOESCR:                          ; sub-rotina para escrever dados no LCD
            ACALL PRONTO
            MOV   P1, A
            SETB  P2.0             ; RS = 1 para dados
            CLR   P2.1
            SETB  P2.2
            CLR   P2.2
RET
PRONTO:
            SETB  P1.7             ; faz P1.7 porta de entrada
            CLR   P2.0             ; RS = 0
            SETB  P2.1             ; R/W = 1
VOLTA:
            CLR   P2.2
            SETB  P2.2
            JB    P1.7, VOLTA
            RET
END
```

; Não esqueça de verificar o modelo do LCD usado → consultor os databooks

11.2 Interfaceamento de um conversor analógico para digital (A/D ou ADC) AO 8051

Revisão básica:

Conversores A/D são usados em diversas aplicações principalmente em sistemas de aquisição de dados. Computadores digitais utilizam valores digitais (discretos), mas o mundo físico é analógico (contínuo), como por exemplo, temperatura, pressão, umidade, velocidade e etc. Um parâmetro físico é convertido para sinais elétricos (tensão, corrente) usando um dispositivo chamado transdutor (em geral são denominados também de sensores, mas cuidado com esta definição). Portanto, necessitamos utilizar um conversor A/D para transformar sinais analógicos para digitais. Os A/D's ou séries mais facilmente encontrados no mercado são: ADC 0804, MAX 158, MAX 186 e MAX 111 (existem diversos fabricantes que fornecem amostras grátis de A/D's e D/A's aconselho tentarem obtê-las para brincar um pouco – entrar nas páginas da National Semiconductores, Harris, Analog Devices, Maxim, Linear Devices, etc).

Apenas para exemplificar usarei o A/D 0804 (da família ADC0800 da National). Este A/D trabalha com tensão de 5V e tem uma resolução de 8 bits. Além disso, outro fator importante no uso do A/D é seu tempo de conversão, ou seja, o tempo que leva para converter uma entrada analógica para uma saída digital. No ADC 0804, o tempo de conversão varia dependendo do sinal de clock aplicado aos pinos CLK R e CLK IN, mas não passa de 110μs.

Descrição básica dos pinos:
CS (Chip Select): seleção do integrado. Sinal de entrada ativo baixo que permite ativar o 0804.

RD (read): sinal de entrada ativo baixo. O A/D converte a entrada analógica para seu equivalente e segura em seu registrador interno. RD é usado para conseguir o dado convertido na saída do 0804. Quando CS = 0, se um pulso alto para baixo é aplicado no pino RD, as saídas de 8 bits são mostradas nos pinos de dados D0-D7. Em alguns A/D o pino RD é denominado de Output Enable (Habilita Saída).

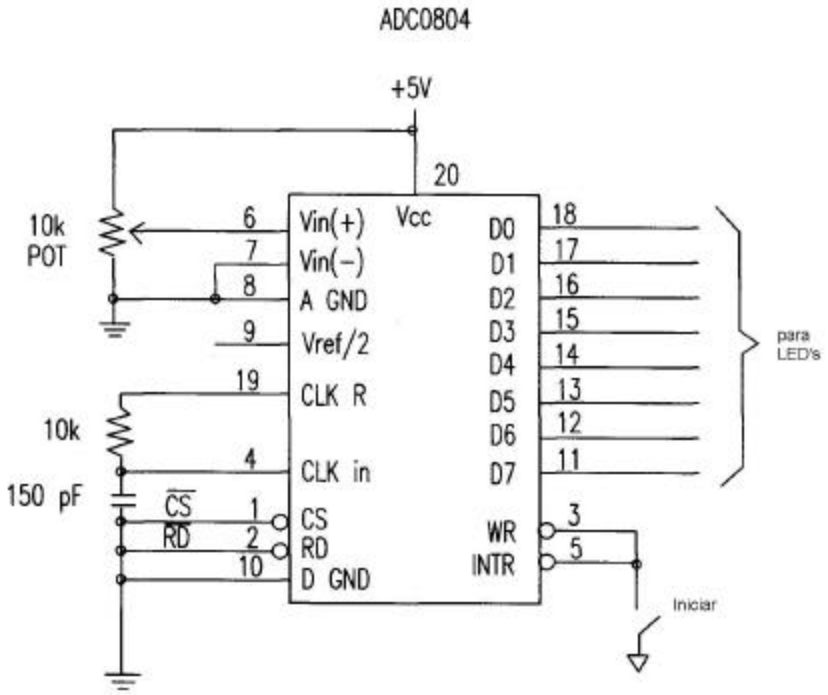
WR (write – o melhor nome para este pino seria start conversio, ou seja, iniciar a conversão): esta é uma entrada ativa baixo usada para informar ao A/D 0804 para iniciar a conversão da entrada analógica Vin para um número de 8 bits. Não esqueça que a velocidade desta conversão depende dos valores CLK IN e CLK R. Quando a conversão do dado foi completada, o pino INTR é forçado para baixo pelo A/D 0804.

CLK IN e CLK R: CLK IN é um pino de entrada conectado a uma fonte de clock externa quando deseja-se usar uma fonte de clock externa. Contudo, o 0804 tem um gerador de clock interno. Para usar o clock interno (denominado de self-clocking) do 0804, os pinos CLK IN e CLK R são conectados a um capacitor e a um resistor, conforme mostrado na Figura abaixo. Nesta configuração a freqüência do clock é determinada pela seguinte equação:

$$f = \frac{1}{1{,}1xRxC}$$

Típicos valores são: R = 10KΩ e C = 150pF. O que fornece uma freqüência de f = 606KHz o que nos fornece um tempo de conversão na ordem de 110μs.

Circuito teste do A/D:



Esta configuração é denominada de running test mode e é recomendada pelo fabricante para testar o A/D (aconselho sempre a executarem os testes padrões definidos pelos fabricantes, pois assim garantem que o dispositivo está perfeito e que entenderam o funcionamento do dispositivo). Neste modo é utilizado um potenciômetro para aplicar um tensão analógica de 0 a 5V na entrada Vin (+) do A/D 0804. A saída binária pode ser monitorada utilizando-se LED's em algum circuito treinador (que pode facilmente ser desenvolvido).

INTR (interrupt → o melhor nome seria "end of conversion" = fim da conversão): este é um pino de saída e é ativo baixo. É normalmente alto e quando a conversão é terminada vai para baixo.

Vin (+) e Vin (-): são as entradas analógicas diferenciais onde Vin = Vin (+) – Vin (-). Freqüentemente o pino Vin (-) é conectado ao terra e o pino Vin (+) é usado como a entrada analógica que será convertida em digital.

Vcc: é o pino de alimentação e, neste caso, +5V. Pode ser também usado como tensão de referência quando a entrada Vref/2 (pino 9) não é utilizado (está aberto).

Vref/2 (pino 9): é uma tensão de entrada usada como tensão de referência. Se este pino não é utilizado, a tensão de entrada analógica é limitada de 0 a 5V (a mesma do pino Vcc). Contudo, existem diversas aplicações onde a entrada analógica necessita outra faixa de limite e, neste caso, Vref/2 é utilizado. Por exemplo, se a faixa de entrada analógica de interesse precisa ser de 0 a 4V, Vref/2 é conectado a 2V. A Tabela abaixo mostra as faixas de Vin para várias Vref/2.

Vref/2 (V)	Vin (V)	Passo (mV) (= resolução)
Não usado (aberto)	0 a 5	5/256 = 19,53
2	0 a 4	4/255 = 15,62
1,5 1,28	0 a 3 0 a 2,56	3/256 = 11,71 2,56/256 = 10
1	0 a 2	2/256 = 7,81
0,5	0 a 1	1/256 = 3,90

Observe que Vcc = 5V. Resolução (passo em mV) é a menor mudança que pode ser detectada pelo A/D.

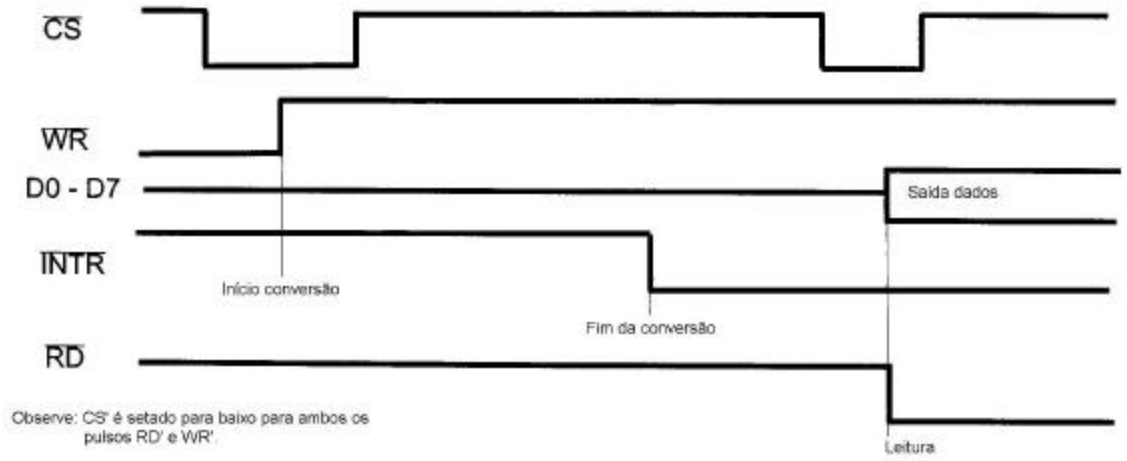
D0-D7 (D7 é o mais significativo): são os pinos de saídas digitais. São saídas tri-state e o dado convertido é acessado somente quando CS = 0 e RD = é forçado para baixo. Para calcular a tensão de saída, use a seguinte fórmula:

$$D_{out} = \frac{V_{in}}{\text{Resolução}} \text{ [em decimal]}$$

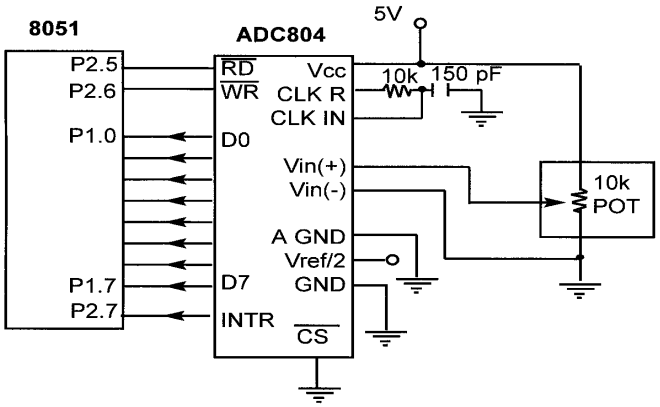
onde D_{out} é a saída do dado digital (em decimal), V_{in} é a tensão de entrada analógica e Resolução é a menor mudança, que é (2xVref/2)/256 para um conversor D/A de 8 bits.

Terra analógico e digital (A GND e D GND): ambos são pinos que fornecem o terra para ambos os sinais analógicos e digitais. A GND é conectado ao terra do Vin analógico quando o D GND é conectado ao terra do pino Vcc. Isto serve para isolar o sinal analógico Vin de transientes (esta configuração contribui para a precisão da conversão). Para facilitar nossa discussão, ambos estão conectados ao mesmo terra. Resumindo todas as discussões anteriores o seguintes passos são necessário para a conversão de dados no A/D 0804:

(a) CS = 0 e enviar para o pino WR' um pulso baixo para alto para iniciar a conversão;
(b) Monitorar o pino INTR. Se INTR é baixo, a conversão foi terminada;
(c) Após INTR ficar baixo, CS = 0 e enviamos um pulso alto para baixo no pino RD' para para podermos pegar o dado na saída de dados. O diagrama abaixo exemplifica este processo.



Exemplo_1: Examinar a conexão abaixo do 8051 com o A/D 0804. Escrever uma rotina para monitorar o pino INTR.



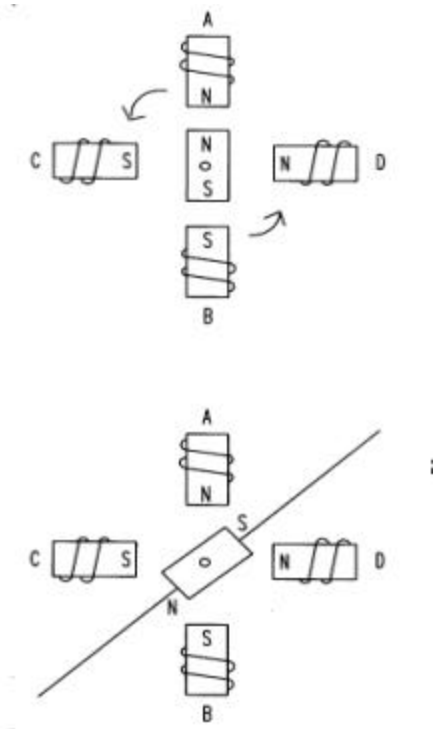
; solução

; P2.6 → WR (para iniciar a conversão é necessário pulso baixo para alto)
; P2.7 → quando baixo, fim da conversão
; P2.5 → RD (um pulso alto para baixo é iniciado a leitura do dado)
; P1.0 a P1.7 → D0-D7 do A/D

```
            MOV   P1, #0FFH              ;  torna P1 → porta de entrada
VOLTA:      CLR   P2.6                   ; WR = 0
            SETB  P2.6          ; WR = 1 → Baixo para Alto → começa conversão
AQUI:       JB    P2.7, AQUI             ; espera pelo fim da conversão
            CLR   P2.5                   ; conversão terminada → habilitar RD
            MOV   A, P1                  ; leitura do dado
            ACALL CONVERSÃO                          ; rotina hexa para ASCII
            ACALL       EXIBIÇÃO_DO_DADO             ; exibir dado
            SETB  P2.5                   ; RD = 1 para a próxima aquisição
            SJMP  VOLTA
```

11.3 Interfaceamento de um motor de passo ao 8051

Revisão básica: um motor de passo é um dispositivo largamente usado que transforma pulsos elétricos em movimentos mecânicos. Existem diversas aplicações, entre elas, disk drivers, robótica, impressoras, onde normalmente é utilizado para controlar posições. Os motores de passo tem um rotor magnético permanente (normalmente chamado de shaft) cercado por um estator (os motores mais comuns possuem 4 estatores).



Para maiores detalhes sobre o funcionamento (comportamento magnético do motor) buscar informações nas disciplinas anteriores. A Tabela abaixo mostra a seqüência normal de um motor de 4 passos.

# Passo	A	B	C	D
1	1	0	0	1
2	1	1	0	0
3	0	1	1	0
4	0	0	1	1

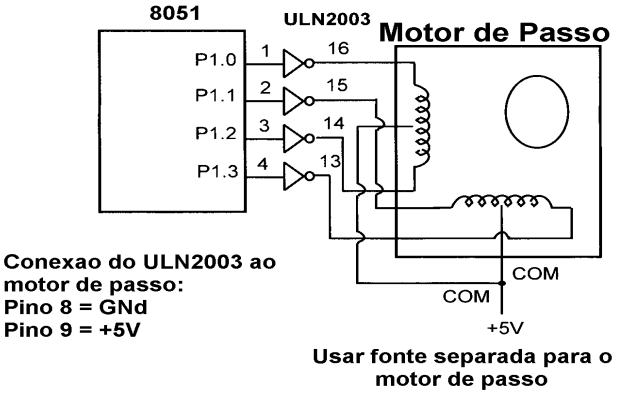
Observe: 0 = off e 1 = on.

A grande pergunta é como o movimento está associado com um simples passo? Isto depende da construção interna do motor (em particular da configuração do estator e do rotor). O ângulo do passo é o mínimo grau de rotação associado com um simples passo. Vários motores possuem diferentes ângulos de passo. Vamos considerar a tabela abaixo que mostra alguns ângulos de passos para vários motores.

Ângulo de passo	Passos por revolução
0,72	500
1,8	200
2	180
2,5	144
5	72
7,5	48
15	24

Observe que passos por revolução é o número total de passos necessários para completar uma rotação completa (360°), por exemplo, 180 passos x 2 graus = 360.

Exemplo: Escrever uma rotina para o motor de passo do esboço dado abaixo rotacionar continuamente.

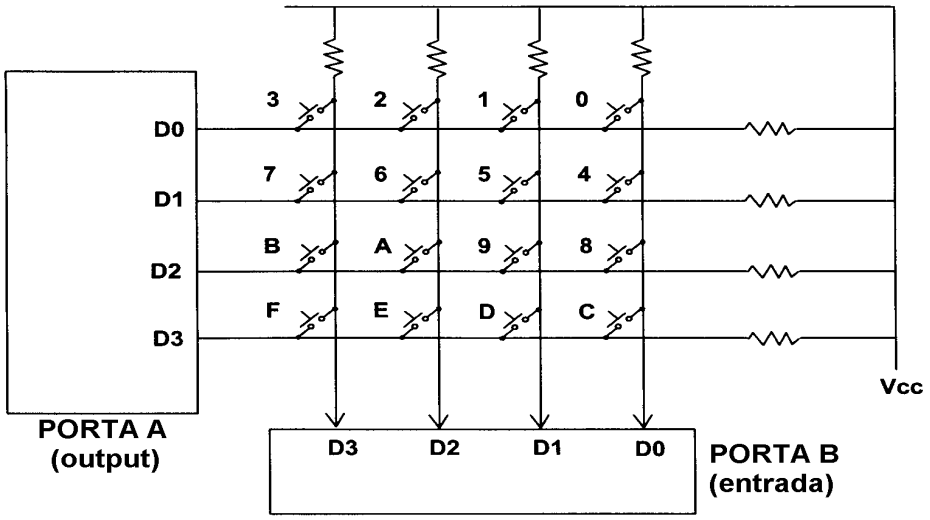



```
        MOV  A, #66H          ; carrega a seqüência de passos
VOLTA:  MOV  P1, A
        RR   A
        ACALL      DELAY
        SJMP VOLTA
        ...
DELAY:
        MOV  R2, #100
H1:     MOV  R3, #255
H2:     DJNZ R3, H2
        DJNZ R2, H1
RET
```

11.4 Interfaceamento de um teclado ao 8051

Teclados são organizados em uma matriz de linhas e colunas. O microprocessador acessa ambas as linhas e colunas através das portas; portanto, com duas portas de 8 bits cada, uma matriz 8x8 de teclas podem ser conectadas a estas portas e acessadas pelo microprocessador. Quando uma tecla é pressionada, a linha e coluna correspondente a esta tecla apresenta um estado alto (ou baixo); por outro quando uma tecla não é pressionada, a linha e coluna, correspondente a tecla não pressionada, estão em nível baixo (ou alto). Em sistemas simples baseados em microcontroladores e teclados a EPROM possui o programa cuja função é varrer o teclado continuamente e identificar qual tecla foi ativada e apresentar a tecla pressionada ao microprocessador.

Para facilitar o entendimento do parágrafo anterior, considere o teclado a seguir (matriz 4x4) conectado a duas portas. As linhas estão conectadas a uma porta de saída e as colunas estão conectadas a uma porta de entrada. Se nenhuma tecla está pressionada, a leitura da porta de entrada obtém nível lógico alto (1) para todas as colunas, pois estas estão conectadas ao Vcc. Se todas as linhas estão aterradas e uma tecla é pressionada, uma das colunas vai para nível lógico baixo, pois a tecla pressionada fornece um caminho ao terra. Cabe lembrar que é função do microprocessador varrer o teclado continuamente para detectar e identificar a tecla pressionada.



Sugestão:

a) nível lógico zero para as linhas e leitura das colunas: para detectar a tecla pressionada, o microprocessador aterra todas as linhas fornecendo 0's ao latch de saída (porta de saída), e realiza a leitura das colunas. Por exemplo, se o dado lido das colunas é D_3 – D_0 = 1111, nenhuma tecla foi pressionada e o processo continua até a detecção de uma tecla. Por outro lado, se um dos bits da coluna for zero, significa, ou seja, indica que uma tecla foi pressionada. Por exemplo, se D_3 – D_0 = 1101, significa que a tecla em D_1 da coluna foi pressionada.

b) Após uma tecla ter sido pressionada, o microprocessador deve identificar a tecla pressionada. Iniciando com a linha superior, o microprocessador aterra pelo fornecimento de um nível baixo na linha D_0 somente; então realiza a leitura das colunas. Se os dados lidos são todos 1's, nenhuma tecla está ativada e o processo é movido para a próxima linha. Aterra a próxima linha, lê a coluna e verifica para qualquer zero. Este processo continua até identificar uma linha. Após a identificação da linha cuja tecla foi pressionada, a próxima tarefa é determinar a tecla pressionada.

Exemplo: Considerando a matriz 4x4 anterior (teclado 4x4), identifique a tecla pressionada para cada uma das seguintes condições:

(a) D_3 – D_0 = 1110 → para as linhas, D_3 – D_0 = 1011 → para as colunas;

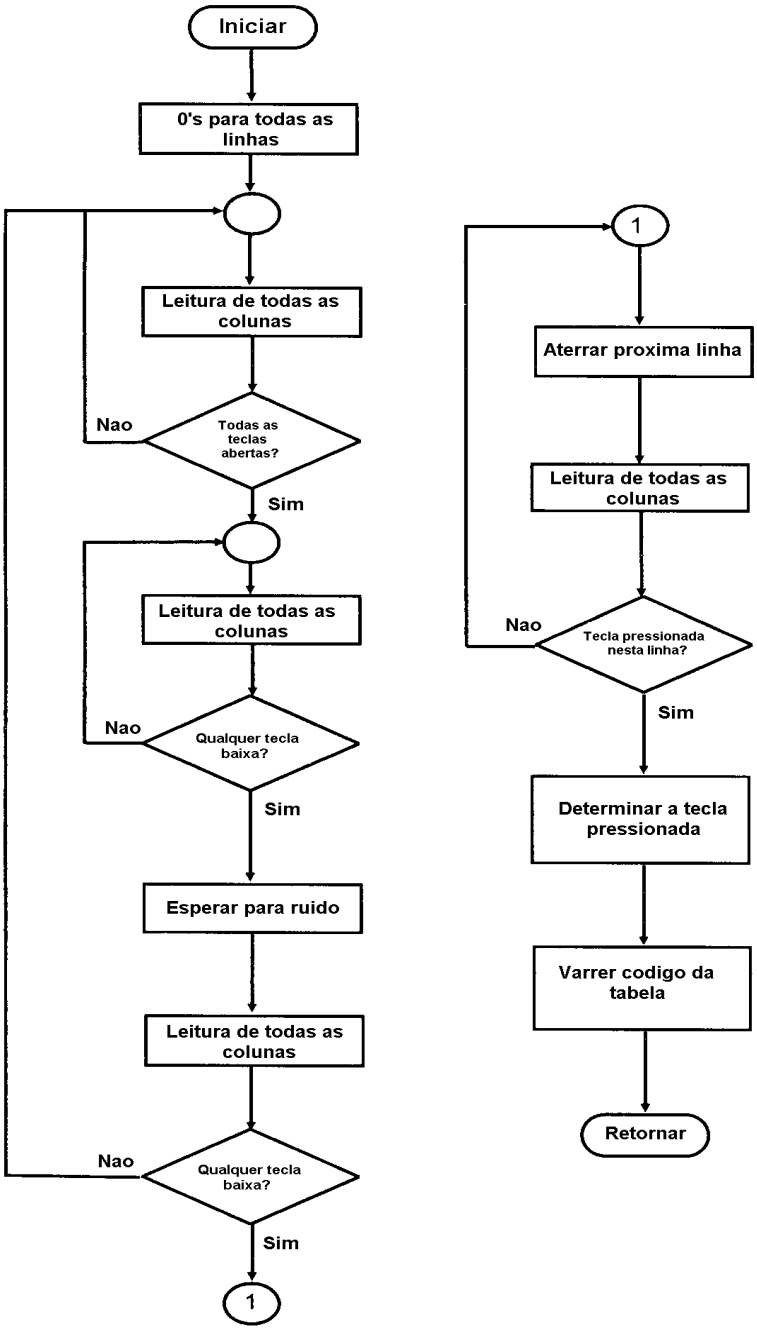
(b) D_3 – D_0 = 1101 → para as linhas, D_3 – D_0 = 0111 → para as colunas.

Solução:

(a) A tecla 2 foi pressionada;

(b) A tecla 7 foi pressionada.

O fluxograma abaixo exemplifica uma rotina de varredura para este teclado. Cabe salientar que existem no mercado diversos integrados que incorporam as funções de varredura de teclado e decodificação, como por exemplo, o integrado da National Semiconductor's MM74C923.




```
; sub-rotina para o teclado
; P1.0 – P1.3 conectado as linhas
; P2.0 – P2.3 conectado as colunas
                MOV   P2, #0FFH
T1:             MOV P1, #0
                MOV   A, P2
                ANL   A, 0000 1111B
                CJNE  A, #0000 1111B, T1
T2:             ACALL        DELAY
```

```
                MOV   A, P2
                ANL   A, #0000 1111B
                CJNE  A, #0000 1111B, T3
                SJMP  T2
T3:             ACALL DELAY
                MOV   A, P2
                ANL   A, #0000 1111B
                CJNE  A, #0000 1111B, T4
                SJMP  T2
T4:             MOV   P1, #1111 1110B
                MOV   A, P2
                ANL   A, #0000 1111B
                CJNE  A, #0000 1111B, LINHA_0
                MOV   P1, #1111 1101B
                MOV   A, P2
                ANL   A, #0000 1111B
                CJNE  A, #0000 1111B, LINHA_1
                MOV   P1, #1111 1011B
                MOV   A, P2
                ANL   A, #0000 1111B
                CJNE  A, #0000 1111B, LINHA_2
                MOV   P1, #1111 0111B
                MOV   A, P2
                ANL   A, #0000 1111B
                CJNE  A, #0000 1111B, LINHA_3
                LJMP  T2
LINHA_0:        MOV   DPTR, #TECCOD0
                SJMP  ENCONTRAR
LINHA_1:        MOV   DPTR, #TECCOD1
                SJMP  ENCONTRAR
LINHA_2:        MOV   DPTR, #TECCOD2
                SJMP  ENCONTRAR
LINHA_3:        MOV   DPTR, #TECCOD3
ENCONTRAR:      RRC   A
                JNC   MAT
                INC   DPTR
                SJMP  ENCONTRAR
MAT:            CLR   A
                MOVC        A, @A+DPTR
                MOV   P0, A
                LJMP  T1
ORG   300H
TECCOD0:        DB    '0', '1', '2', '3'
TECCOD1:        DB    '4', '5', '6', '7'
TECCOD2:        DB    '8', '9', 'A', 'B'
TECCOD3:        DB    'C', 'D', 'E', 'F'
END
```

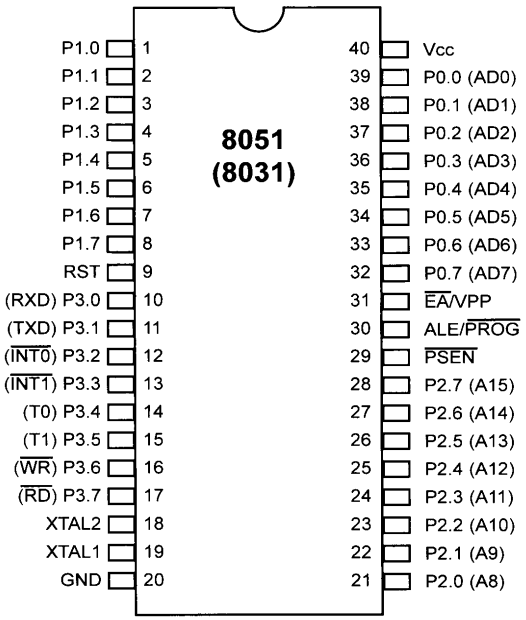
11.5 Interfaceamento do 8051 a memória externa

Aspectos teóricos sobre memórias → assunto já tratado no 80x86

Métodos de decodificação de memória → assunto já tratado no 80x86 (portas lógicas como um decodificador, o decodificador 74LS138 ou uma PAL).

11.5.1 Interfaceando o 8051 com memória ROM externa

Novamente vamos dar uma boa olhada na pinagem do 8051:



Percebe-se os seguintes pinos com funções relacionadas as memórias externas:

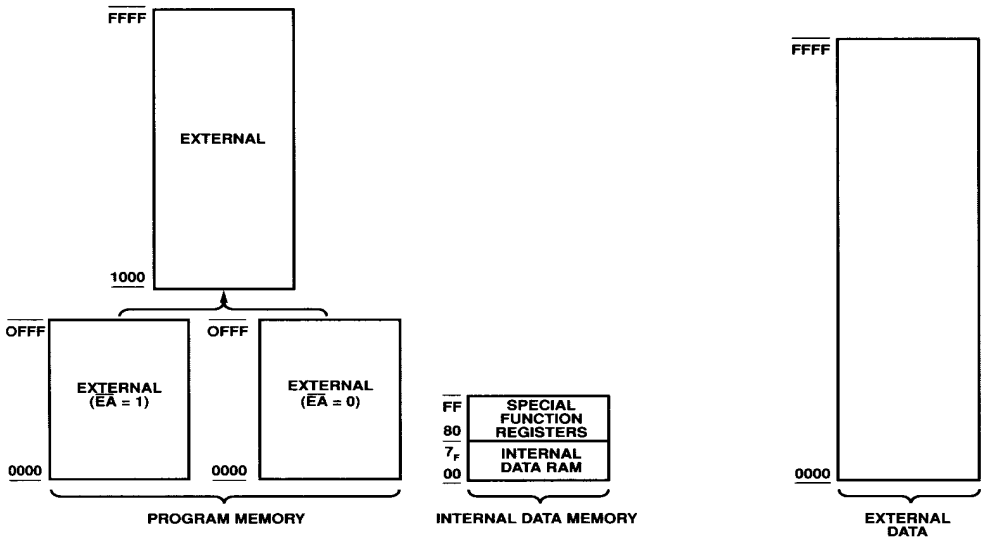
- pino EA' (external access): utilizado para determinar quando a memória ROM é interna ou não. Nos modelos da família µC 0851 que possuem ROM interna (como por exemplo, o 8751, 89C51, etc), o pino EA' deve estar concetado ao Vcc. Para os membros da família µC 0851 que não possuem ROM interna (como por exemplo, 8031, 8032, etc) e, portanto, o programa é armazenado em uma ROM externa e o ciclo de busca deve ocorrer na memória externa, o pino EA' deve estar conectado ao GND para indicar que o programa está armazenado externamente. Observe que este pino não pode estar nunca desconectado;

- pino PSEN' (Program Store Enable - utilizado principalmente nos sistemas baseados no 8031): em sistemas com memória ROM externa este pino é conectado ao pino OE da ROM;
- pino ALE (Address Latch Enable - utilizado principalmente nos sistemas baseados no 8031): é um pino de saída ativo alto. Quando o 8031 está conectado a uma memória externa, a Porta 0 serve tanto como barramento de endereços e dados (barramento multiplexado, como no 8086 e 8085). Este pino é utilizado para demultiplexar o endereço e dado pela conexão deste pino ao pino G do buffer 74LS373 (conforme visto no 8086).

Separação do espaço de memória e tamanho

O 8051 (ao contrário do 80x86) separa os espaços de endereço: um para armazenamento do programa (usualmente ROM) e outro para armazenamento de dados (RAM). Um dado endereço numérico pode estar relacionado a 2 locações de memória diferentes logicamente e fisicamente, dependendo do tipo de instrução que usa o referido endereço.

Expansão de memória (on-chip + externa)	
Memória de programa Internamente 4KB	Memória de dados internamente 128B
Expansão até 64KB.	Expansão até 64KB (externo).
Configurações possíveis: 60KB externos + 4K internos OU 64 KB externos.	Configurações possíveis: 128B internos E 64K B externos.

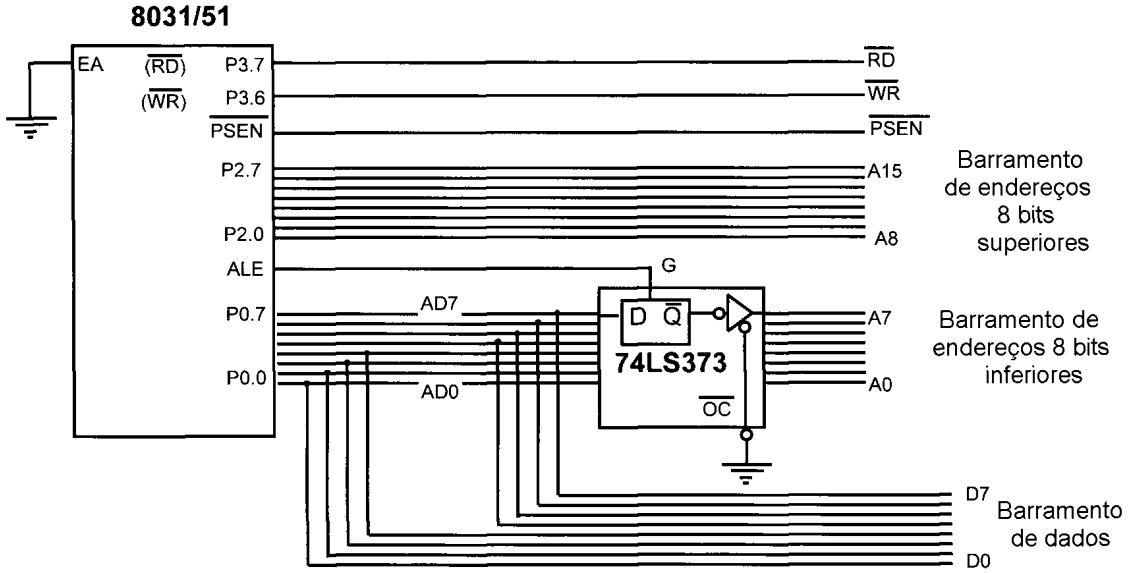


Após o RESET, a CPU inicia a busca de instruções (ciclo de fetch) iniciando no endereço 0000H. A localização física do endereço 0000H é igual no dispositivo on-board ou externo, dependendo do pino do 8051 $\overline{EA}$ (external address).

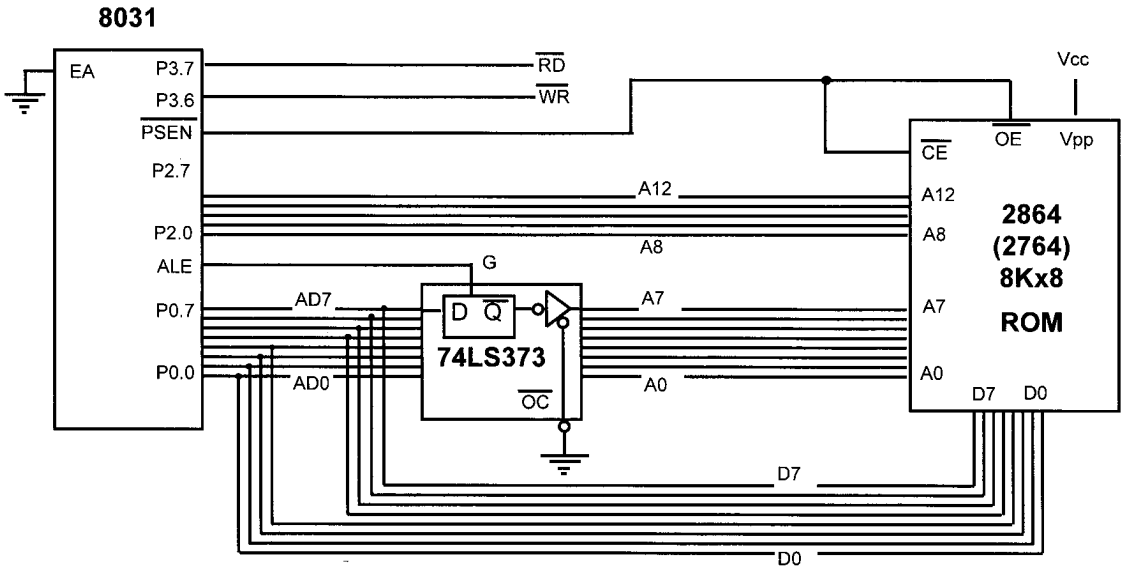
Se $\overline{EA} = 0$***, o endereço 0000H e todos os outros endereços de programas armazenados está relacionado à memória externa.***

Se $\overline{EA} = 1$***, os endereços 0000H a 0FFFH (até 1FFFFH para o 8052) está referenciado a ROM interna; endereços mais altos automaticamente estão relacionados à memória externa*** *(no 8031 →* $\overline{EA} = 0$*, pois não contém ROM interna (on-chip)). Alguns pinos de I/O são usados para endereços e dados quando usamos a memória externa.*

Barramentos de dados, endereços e controle para o 8031



Conexão do 8031 a memória externa



Resumindo temos:

ROM		
8031/8051	***8051***	***8052***
EA = GND	EA = Vcc	EA = Vcc
	Memória interna 0000H - 0FFFH	Memória interna 0000H – 1FFFFH
Memória externa 0000H a FFFFH	Memória externa 1000H – FFFFH	Memória externa 2000H - FFFFH

Espaço de memória de dados no 8051

Como já discutido, o contador de programa do 8051 tem 16 bits e portanto pode acessar até 64KB de memória. Em alguns exemplos anteriormente vistos, colocamos dados no espaço de código e usamos a instrução MOVC A, @A+DPTR para pegar o dado. A instrução MOVC (o C → indica código) indica que o dado está localizado no espaço de código do 8051 (ROM).

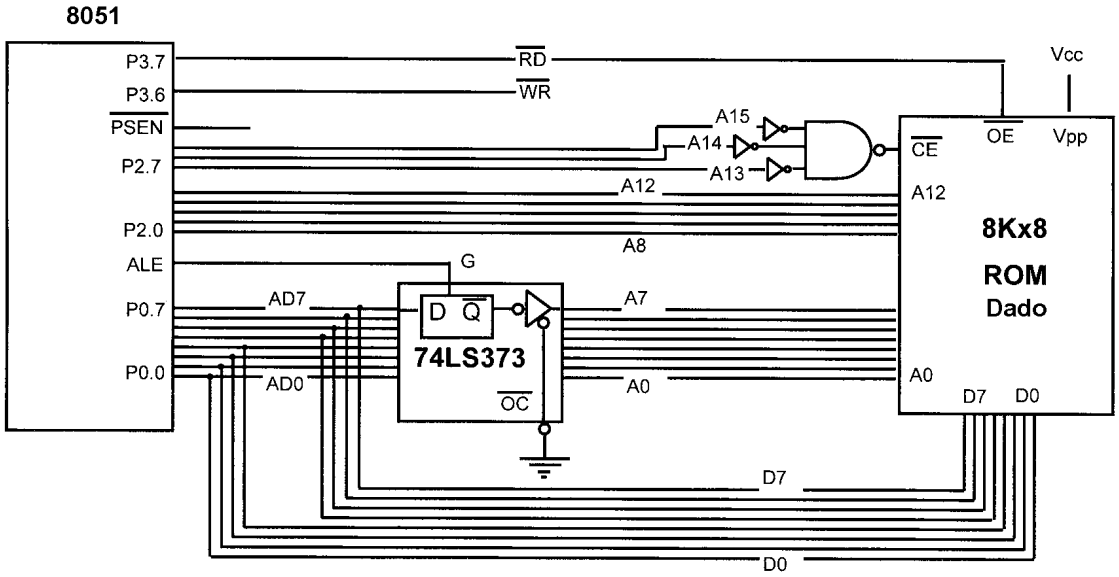
Em adição ao espaço de código, a família 8051 também possui 64KB de memória de dados.

Expansão de memória (on-chip + externa)	
Memória de programa Internamente 4KB	**Memória de dados internamente 128B**
Expansão até 64KB. Configurações possíveis: 60KB externos + 4K internos OU 64 KB externos.	**Expansão até 64KB (externo). Configurações possíveis: 128B internos E 64KB externos.**

O espaço destinado ao programa é acessado usando o Contador de Programa (PC) para localizar e buscar (ciclo de fetch) as instruções a serem decodificadas e executadas. Para o espaço de memória dedicado aos dados, este espaço é acessado através do registro DPTR e uma instrução chamada MOVX onde o X indica externo → esta instrução também indica que o espaço de dados precisa ser implementado externamente.

Diversas possibilidades de conexão com memórias:

(a) usando uma ROM externa para dados:



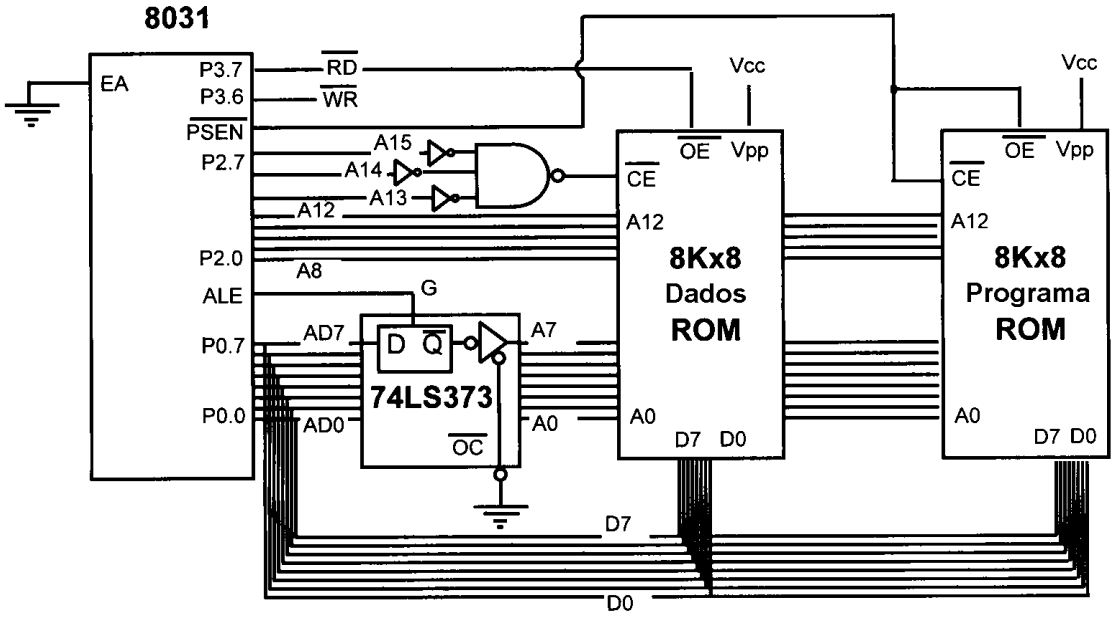
Note que o pino PSEN não está sendo usado, pois é usado para buscar o código (fetch) e o sinal RD está funcionando como o fetch dos dados. Para acessar os dados desta configuração externa, deve-se usar a instrução MOVX.

Exemplo_1: uma ROM externa é usada pelo 8051 para armazenar uma tabela de dados (iniciando no endereço 1000H). Escrever um programa para ler os 30 bytes desses dados e enviá-los pela porta P1.

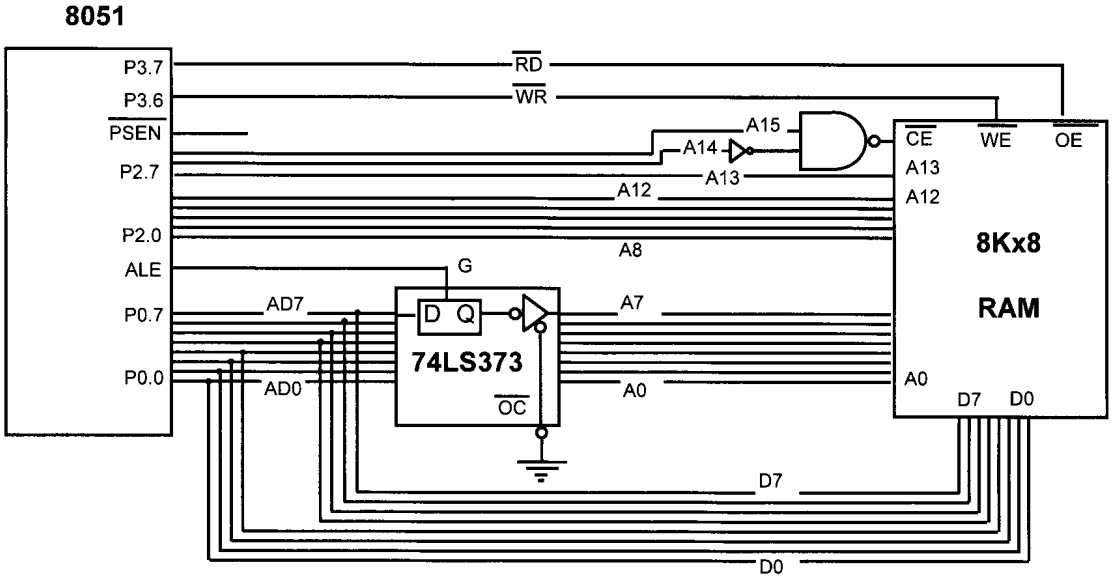
Solução:

```
MEUS_DADOS      EQU   1000H
CONTADOR        EQU   30H
...
...
                MOV   DPTR, #MEUS_DADOS     ; apontador para dados externos
                MOV   R2, #CONTADOR                 ; contador
NOVAMENTE:      MOVX A, @DPTR
                MOV   P1, A
                INC   DPTR
                DJNZ  R2, NOVAMENTE
```

Exemplo_2: apresente o esboço para interfacear um microcontrolador 8031 com uma memória ROM de 8KB para dados e uma memória ROM 8KB para código.



(b) usando uma RAM externa para dados:



Para escrever dados na RAM externa usamos a instrução MOVX @DPTR, A onde o conteúdo de A é escrito na RAM externa cujo endereço é apontado pelo registro DPTR.

Exemplo_1: Escrever uma rotina para ler 200 bytes de dados da P1 e salvar estes dados na RAM externa iniciando no endereço 5000H.

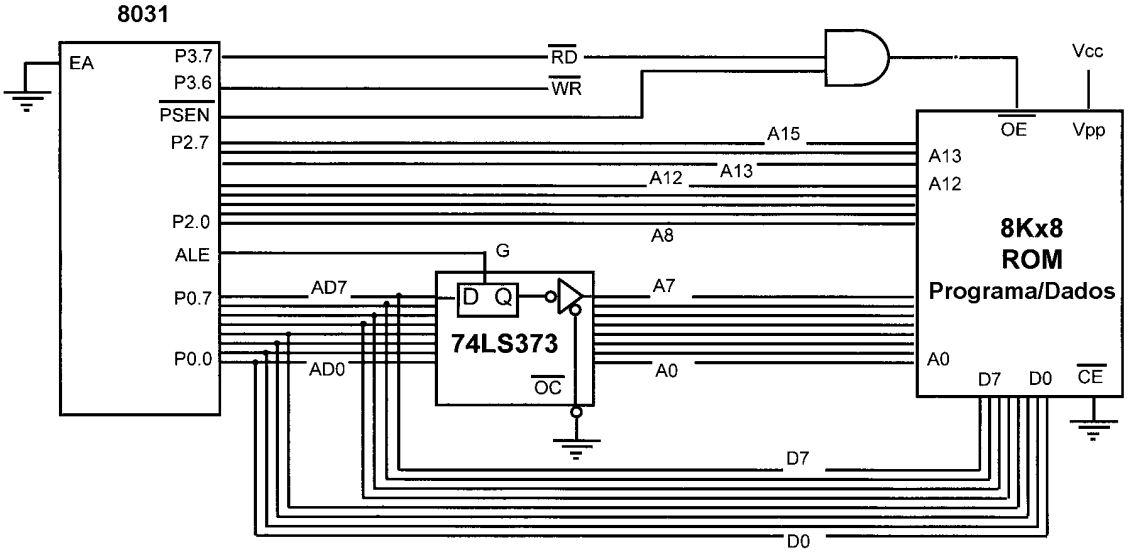
```
DADOS_DA_RAM   EQU   5000H
CONTADOR       EQU   200

               MOV   DPTR, #DADOS_DA_RAM      ; apontador
               MOV   R3, #CONTADOR            ; contador
NOVO:          MOV   A, P1                    ; lê dado da P1
               MOVX        @DPTR, A           ; salva na RAM externa
               ACALL       DELAY              ; atraso
               INC   DPTR                     ; próxima localidade
               DJNZ  R3, NOVO
AQUI:          SJMP  AQUI
```

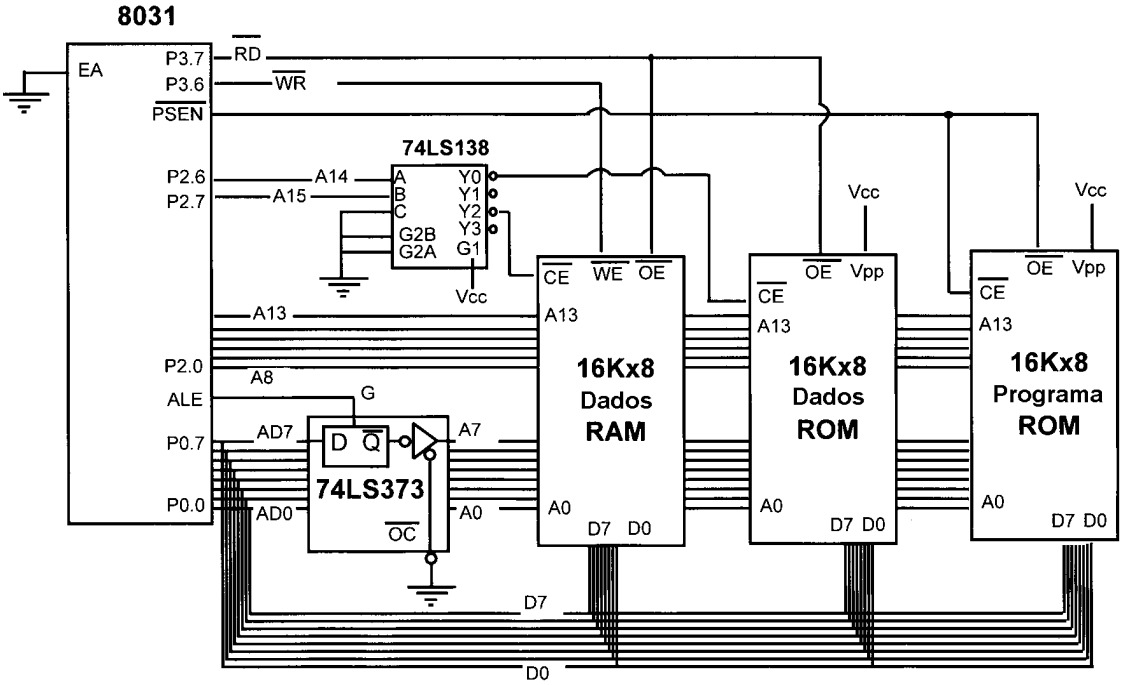
Exemplo_2: Considerando o esboço dado acima qual a faixa de endereço alocada para os dados?

A faixa de endereço é de: 8000H – BFFFH.

(c) usando uma ROM externa para dados e código:

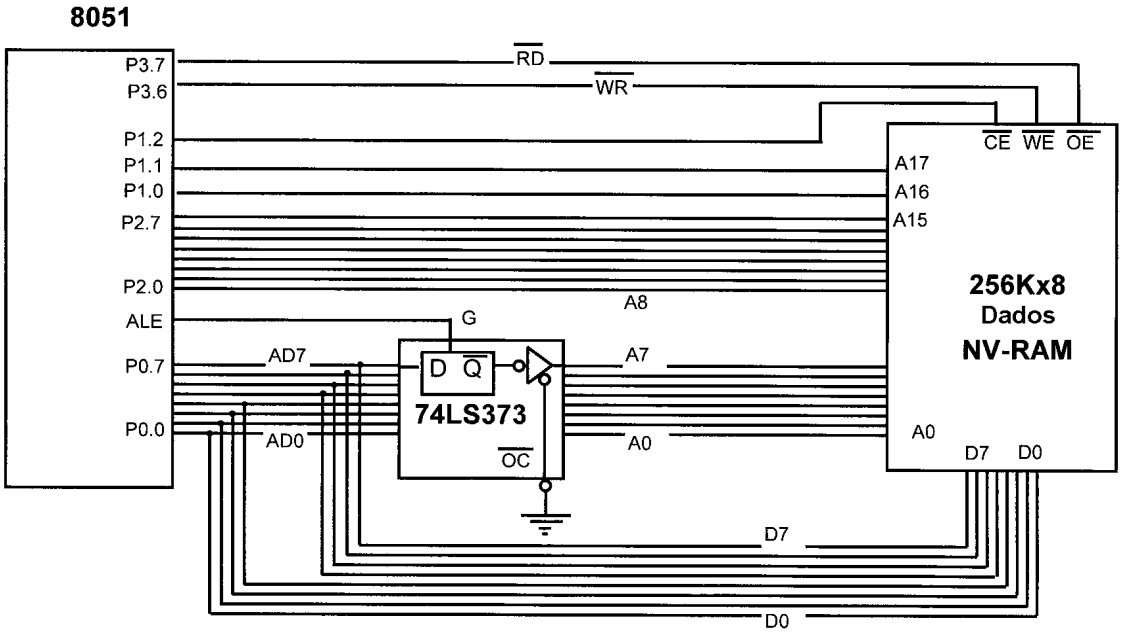


(d) usando um sistema com ROM e RAM:



(e) usando um sistema com ROM e RAM:

Em algumas aplicações é importante a utilização de memórias maiores, ou seja, com capacidade de armazenamento maior, como por exemplo, 256KB de RAM. Porém, como já visto, o 8051 suporta somente memórias de até 64KB pois o DPTR é um registro de 16 bits. Para resolver este problema, basta lançar mão da idéia elaborada pelo 80x86 para segmentar memória, porém aqui cabe salientar que não existe memória segmentada fisicamente e também não existem registradores de segmento como no 80x86 (CS, DS, ES e SS). Veja o exemplo abaixo:

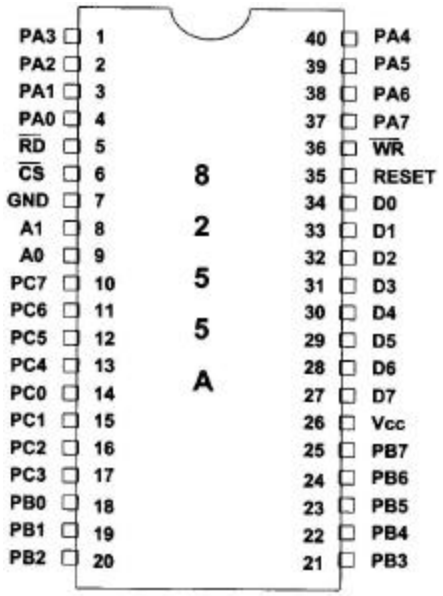


Os 256KB são divididos (segmentados) em 4 blocos:

Seleção do integrado P1.2	***A17 P1.1***	***A16 P1.0***	***Faixa de endereço do bloco***
0	0	0	0000H – 0FFFFH
0	0	1	10000H – 1FFFFH
0	1	0	20000H – 2FFFFH
0	1	1	30000H – 3FFFFH
1	X	X	RAM externa desabilitada

11.6 Interfaceamento do 8051 ao 8255 (interface periférica programável)

Como já visto no 80x86 temos: um dos integrados mais utilizados no interfaceamento de dispositivos. Pode ser utilizado para mapeamento de memória ou no uso das instruções IN e OUT (no 80x86). Possui três portas (de 8 bits cada) A, B e C separadamente e torna-se mais econômica sua utilização do que implementar portas individuais usando os buffers tri-state estudados anteriormente. Além disso, pode ser utilizado para interfacear dispositivos ao microprocessador que utilizam handshaking.



PA0-PA7 → porta de 8 bits que pode ser toda programada como entrada ou saída ou toda como bi-direcional.

PB0-PB7 → porta de 8 bits que pode ser toda programada como entrada ou como saída.

PC0-PC7 → pode ser toda programada como entrada ou saída. Pode ser dividida em 2 partes, CH (PC4-PC7) e CL (PC0-PC3).

RD' e WR' → sinais de controle de entrada. Se o 8255 é usado como interface de periféricos de I/O, IOR' e IOW' do barramento de controle do processador são conectados a esses 2 pinos do 8255.

A0, A1 e CS' → seleção das portas:

CS'	A1	A0	Seleciona
0	0	0	Porta A
0	0	1	Porta B
0	1	0	Porta C
0	1	1	Registrador de controle
1	X	X	Não seleciona

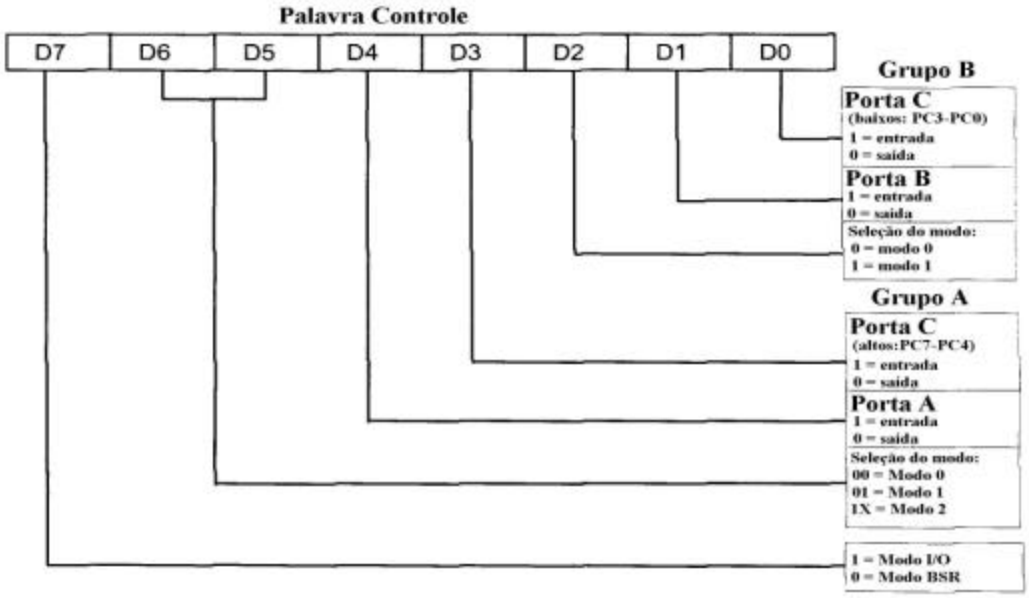
Modos de seleção do 8255A:

(a) modo 0 ou modo simples de I/O: neste modo, qualquer das portas, A, B, CL e CH podem ser programadas como entrada ou saída. Neste modo todos os bits são saídas ou entradas, ou seja, não existe controle de bits individuais;

(b) modo 1: neste modo, as portas A e B podem ser usadas como portas de entrada ou saída com capacidade handshaking. Os sinais handshaking são fornecidos pelos bits da porta C;

(c) modo 2: neste modo a porta A pode ser usada como porta bidirecional com handshaking cujos sinais são fornecidos pela porta C. A porta B pode ser usada no modo 0 ou no modo 1;

(d) modo BSR (Bit Set/Reset): neste modo, somente bits individuais da porta C podem ser programados.

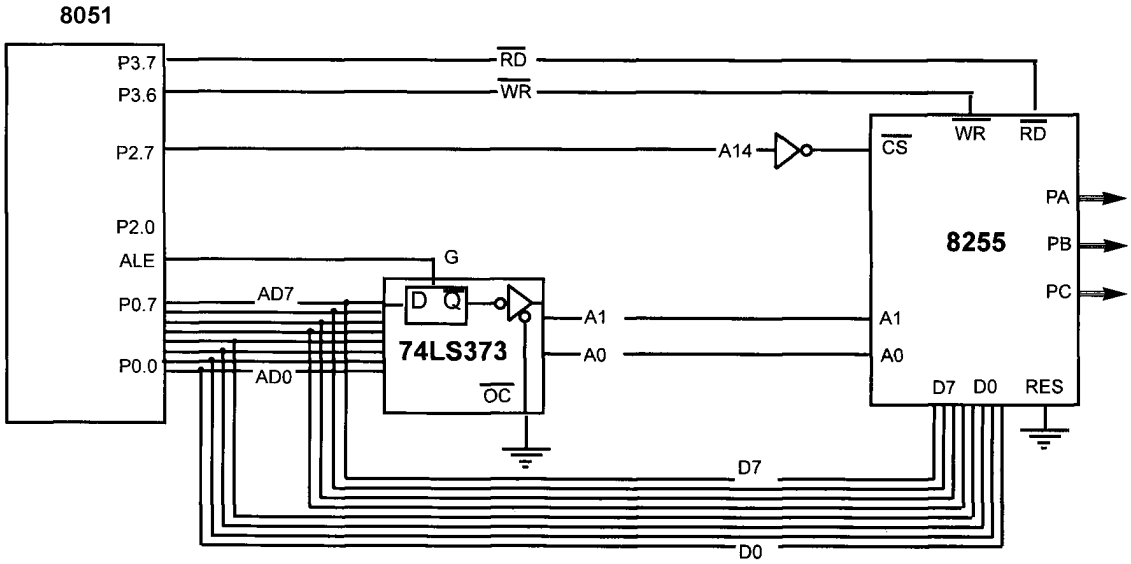


(A) Modo 0: entrada ou saída simples (básica) – simples I/O - neste modo qualquer porta pode ser configurada como entrada ou saída:

Neste estudo será apresentado o modo mais simples e mais utilizado nas diversas aplicações, porém registro a importância dos outros modos, em determinadas aplicações, como por exemplo, no uso dos sinais de hadshaking. Como esta interface programável já foi estudada apresentarei um exemplo e partimos do princípio que todos já sabem programar o 8255 para usa-lo (caso não volte ao material do 80x86 buscando o capítulo referente ao 8255).

Exemplo_1: Considerando o esboço da figura abaixo:

(a) encontrar os endereços das portas de entrada/saída (I/O) e do registrador de controle do 8255;
(b) programar o 8255 para as portas A, B e C sejam portas de saída;
(c) escrever uma rotina para enviar 55H e AAH para todas as portas continuamente.



Solução:

(a) endereços base do 8255:

A15	*A14*	*A13*	*A12*	*A11*	*A10*	*A9*	*A8*	*A7*	*A6*	*A5*	*A4*	*A3*	*A2*	*A1*	*A0*
X	1	X	X	X	X	X	X	X	X	X	X	X	X	0	0
X	1	X	X	X	X	X	X	X	X	X	X	X	X	0	1
X	1	X	X	X	X	X	X	X	X	X	X	X	X	1	0
X	1	X	X	X	X	X	X	X	X	X	X	X	X	1	1

PA → 4000H

PB → 4001H

PC → 4002H

Regs. Contr. → 40003H

(b) a palavra controle é 80H

(c) a rotina segue:

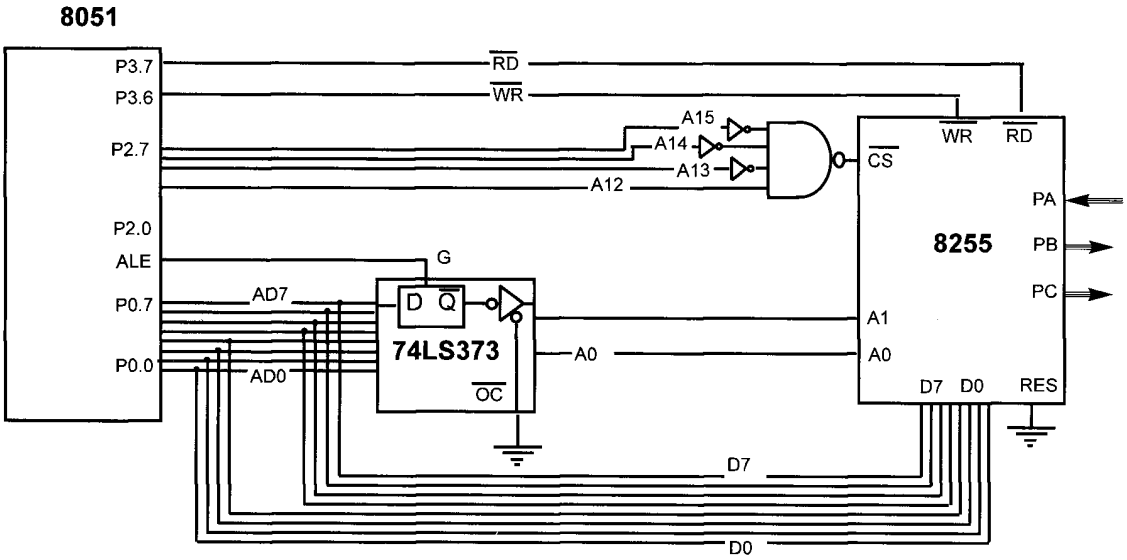
```
        MOV   A, #80H          ; palavra controle
        MOV   DPTR, #4003H     ; carrega endereço da porta registro de controle
        MOVX @DPTR, A          ;
        MOV   A, #55H          ;
NOVO:   MOV   DPTR, #4000H     ; endereço PA
        MOVX        @DPTR, A
        INC   DPTR             ; endereço PB
        MOVX        @DPTR, A
        INC   DPTR             ; endereço PC
        MOVX        @DPTR, A
        CPL   A
        ACALL       DELAY
        SJMP  NOVO
```

Exemplo_2: Considerando o esboço da figura, encontrar:

(a) os endereços das portas de I/O e do registrador de controle do 8255;

(b) encontrar a palavra controle para PA = entrada, PB = saída e PC = saída;

(c) excrever um programa para pegar os dados da PA e envia-los para as portas PB e PC.



Solução:

(a) assumindo que todos os bits são zeros, temos:

1000H → PA

10001H → PB

1002H → PC

1003H → Regs. Controle

(b) a palavra controle é 1001 0000 = 90H

(c) a rotina é:

```
MOV   A, #90H          ; PA = entrada, PB = saída e PC = saída
MOV   DPTR, #1003H     ; carrega o endereço do regs. controle
MOVX        @DPTR, A   ;
MOV   DPTR, #1000H     ; endereço da PA
MOVX        A, @DPTR   ; pega dados da PA
INC   DPTR             ; endereço da PB
MOVX        @DPTR, A   ; envia os dados para PB
INC   DPTR             ; endereço da PC
MOVX        @DPTR, A   ; envia também para PC
```