Microcontroladores

Projeto de HW

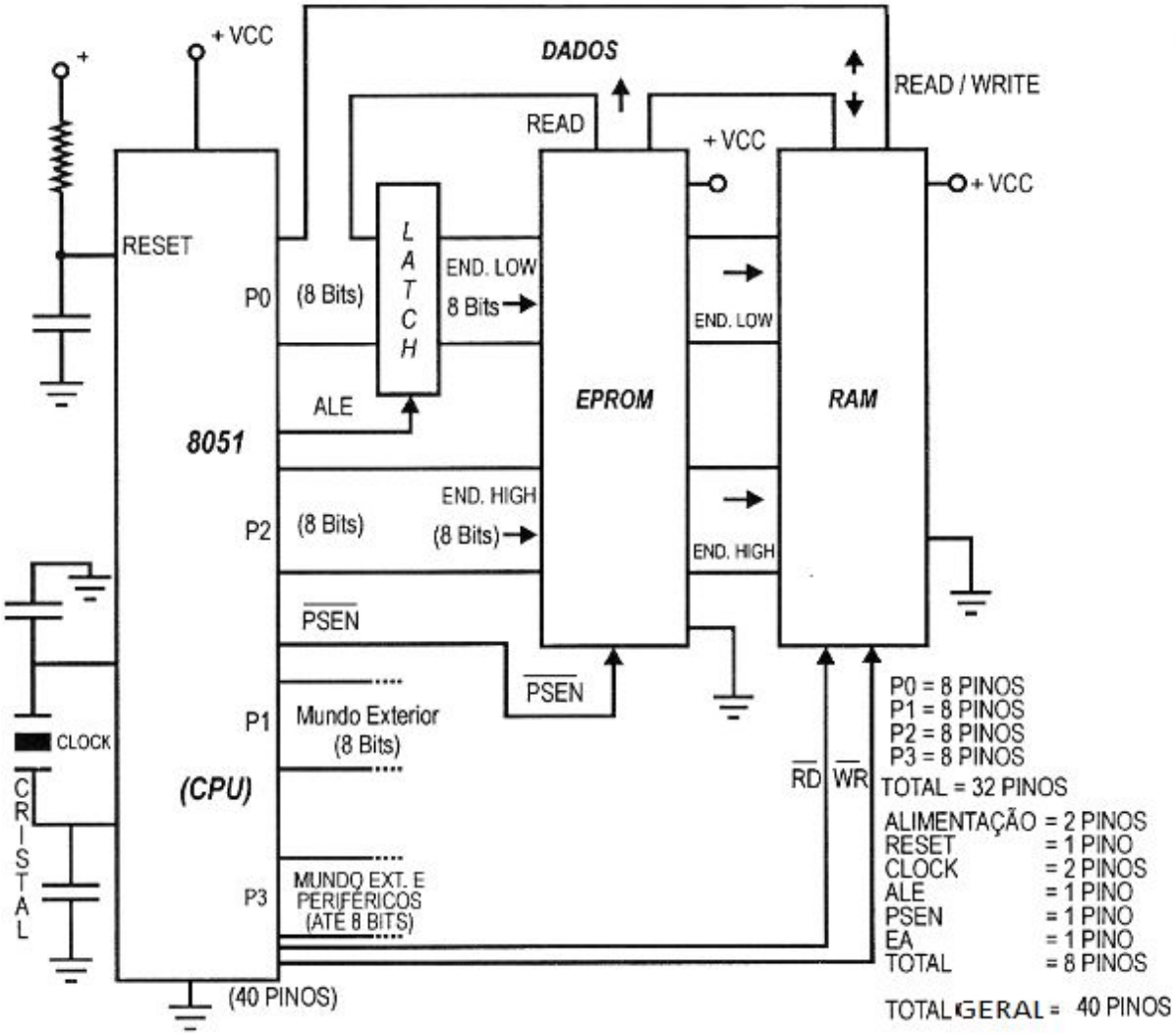
Prof. Guilherme Peron
Prof. Ronnier Rohrich
Prof. Rubão

Observação

- Foi disponibilizado no moodle para quem ainda não conseguiu um arquivo de testes da placa.
  - Este arquivo está pronto para ser testado no Kit P51. Os procedimentos para teste estão em "Arquivos Gerais / Aula Geral - Placa".
  - Obs.:"Lembrar de gerar o arquivo .HEX para ser gravado no 8051".

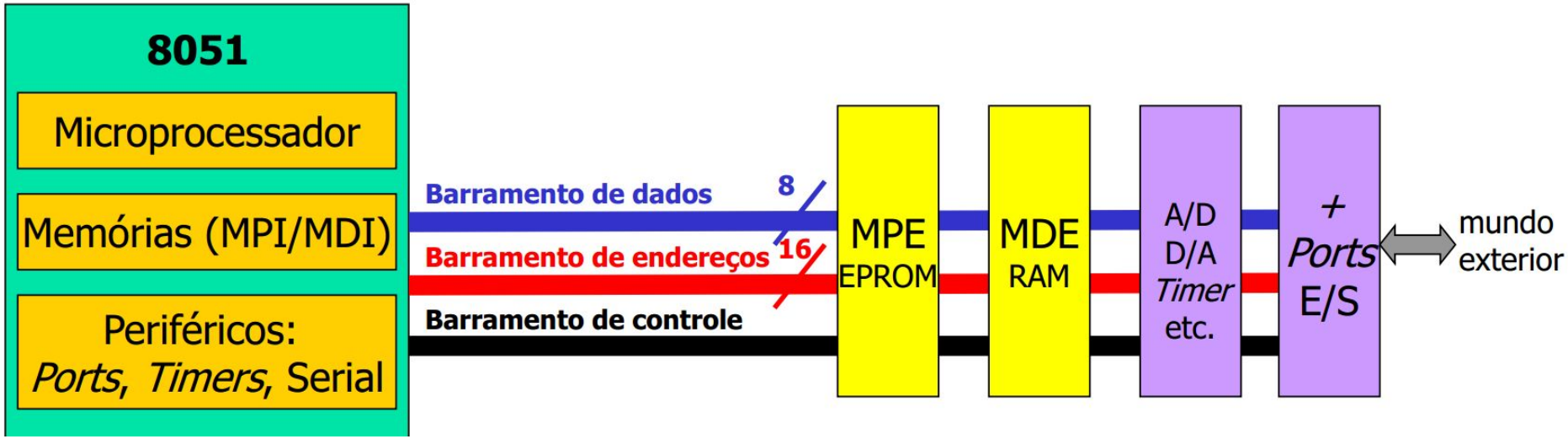
Introdução

Introdução


+ VCC
DADOS
READ / WRITE
READ
+ VCC
+ VCC
RESET
P0
(8 Bits)
LATCH
END. LOW
8 Bits
END. LOW
EPROM
RAM
ALE
8051
END. HIGH
P2
(8 Bits)
(8 Bits)
END. HIGH
PSEN
PSEN
P1
Mundo Exterior
(8 Bits)
CLOCK
(CPU)
RD
WR
P0 = 8 PINOS
P1 = 8 PINOS
P2 = 8 PINOS
P3 = 8 PINOS
TOTAL = 32 PINOS
CRISTAL
P3
MUNDO EXT. E
PERIFÉRICOS
(ATÉ 8 BITS)
ALIMENTAÇÃO = 2 PINOS
RESET = 1 PINO
CLOCK = 2 PINOS
ALE = 1 PINO
PSEN = 1 PINO
EA = 1 PINO
TOTAL = 8 PINOS
(40 PINOS)
TOTAL GERAL = 40 PINOS

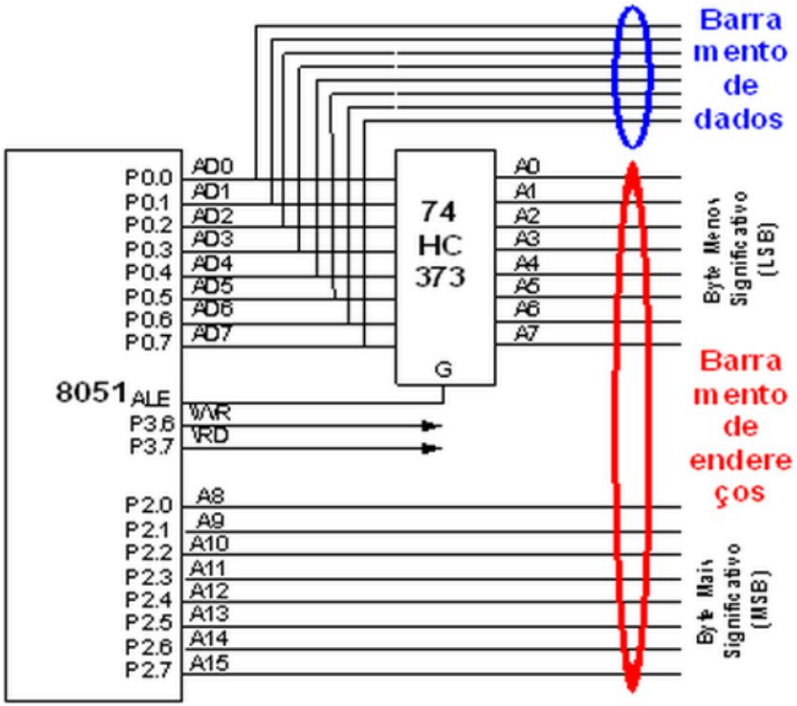
Expansão de capacidade

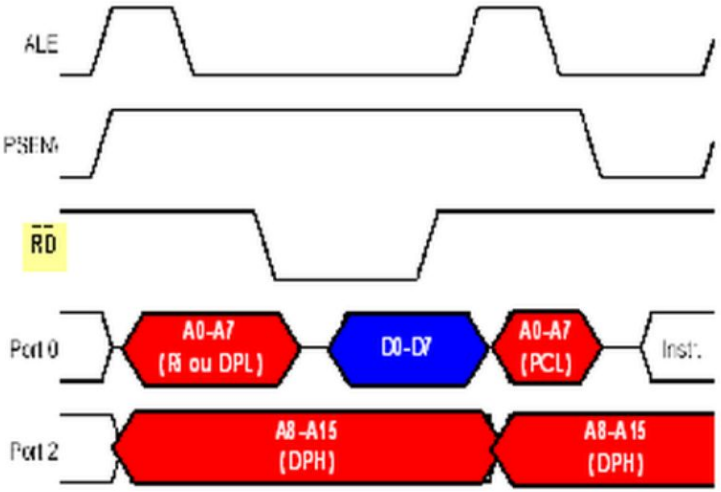
- Embora o 8051 *default* tenha memórias (MPI/MDI) e periféricos (*ports E/S*, 2 *timers* e interface serial) internos, dependendo da aplicação, pode ser necessário expandir a capacidade das memórias ou periféricos.



Expansão de capacidade

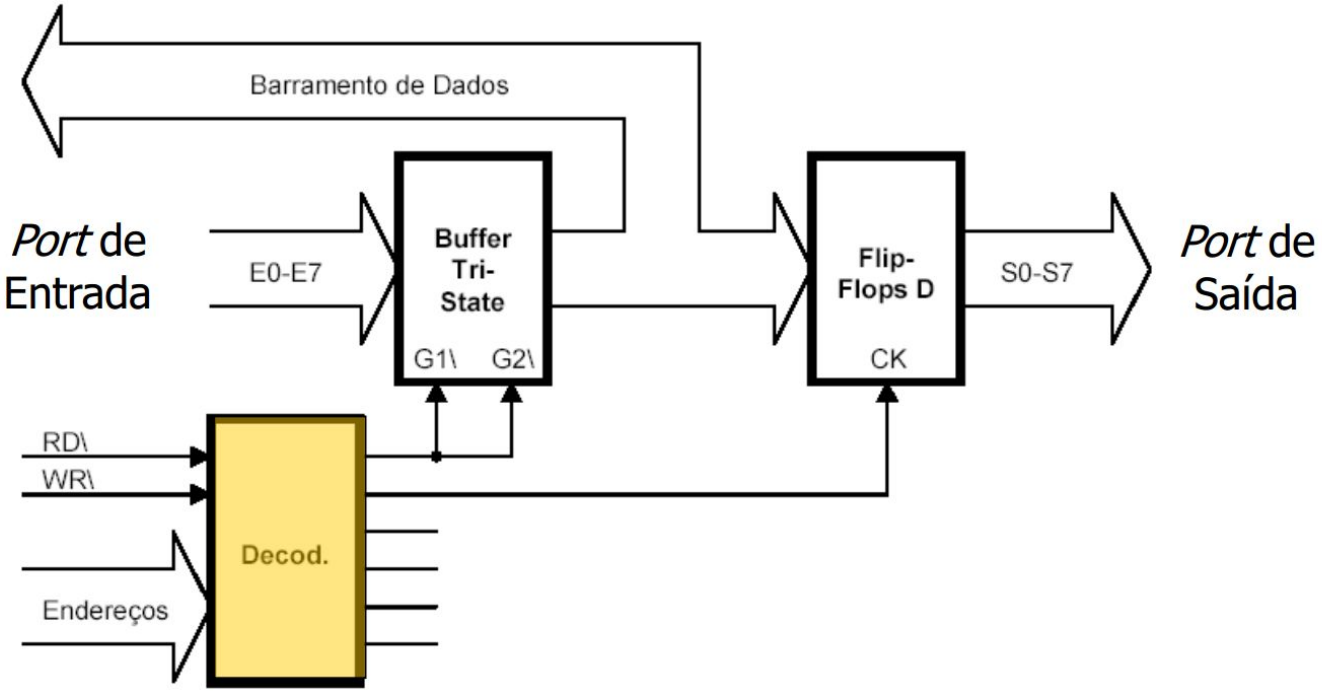
- Para utilizar MDE/MPE:
  - os ports P0 e P2 não podem ser utilizados como E/S
  - requer os sinais \RD (P3.7) e \WR (P3.6)





E/S mapeada em memória

- É uma técnica para acessar periféricos de E/S como se estivessem em posições de memória
- É necessário um decodificador para selecionar fisicamente o periférico na faixa de endereços adequada

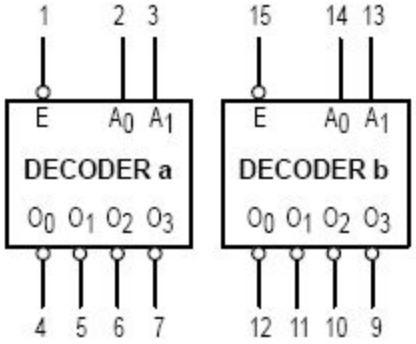


Função do decodificador

- O microcontrolador 8051 utiliza a arquitetura Harvard modificada (dados são separados de programa);
- Portanto há dois espaços de endereçamento:
  - MPE com 64 Kbytes
  - MDE com 64 Kbytes
- A função do decodificador é **segmentar** em faixas o espaço de endereçamento disponível, permitindo que mais de um dispositivo físico possa ser habilitado nesta faixa de endereçamento.

Decodificadores 74xx139 e 74xx138

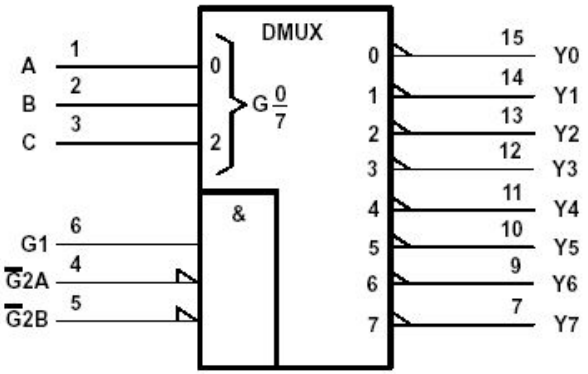
- 74xx139: duplo 2x4, 1 pino de controle:



INPUTS			OUTPUTS			
E	A_0	A_1	O_0	O_1	O_2	O_3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	H	L	H	L	H	H
L	L	H	H	H	L	H
L	H	H	H	H	H	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

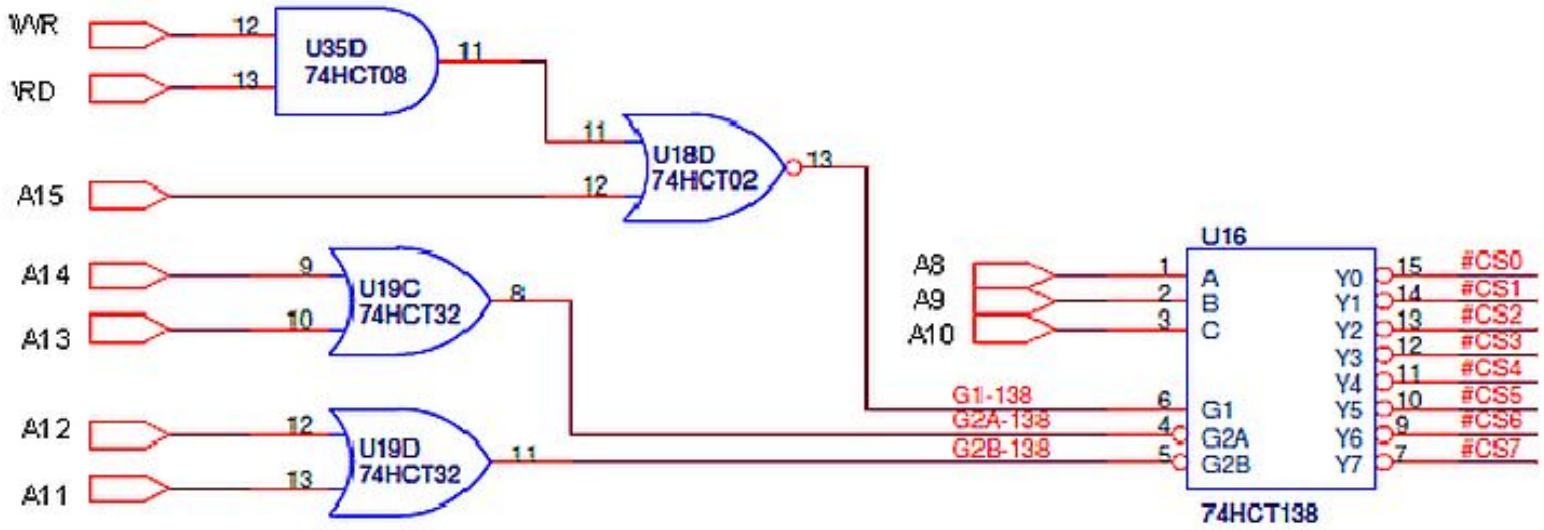
- 74xx138: 3x8, 3 pinos de controle:



INPUTS						OUTPUTS							
ENABLE			SELECT										
G1	$\overline{G2A}$	$\overline{G2B}$	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	L	H	H	H	H	H	L	H	H	H	H
H	L	L	H	L	L	H	H	H	H	L	H	H	H
H	L	L	H	L	H	H	H	H	H	H	L	H	H
H	L	L	H	H	L	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L

Exercício de decodificação

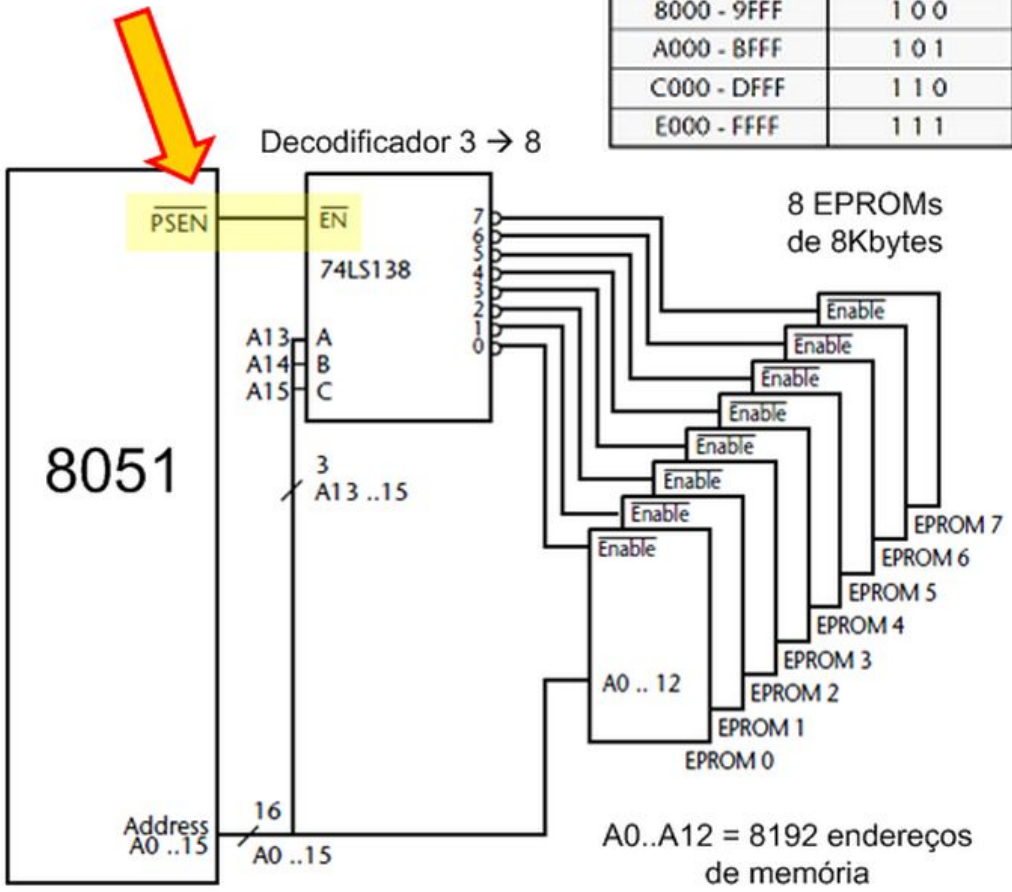
- Definir faixa de endereços de habilitação do decodificador;
- Definir as faixas de endereçamento de cada saída.



Interface com memórias

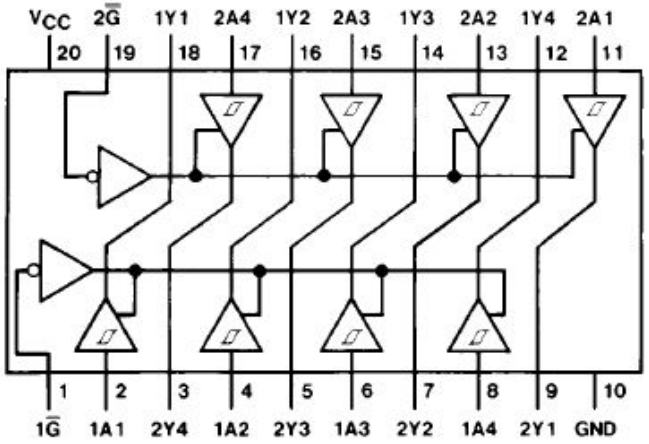
- Há dois espaços de endereçamento separados, p/ MPE e MDE;

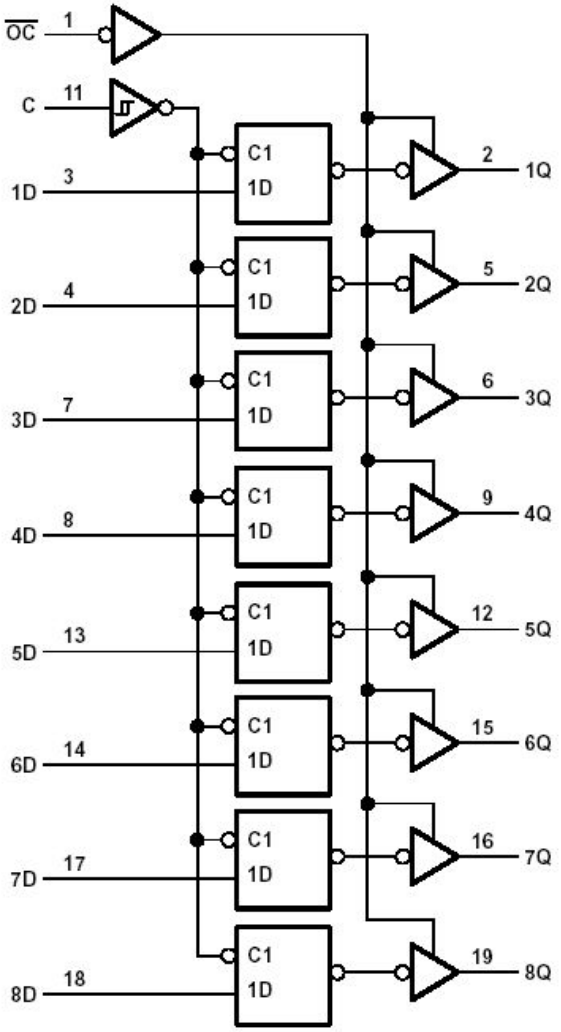
Address Range (hex)	Address bits A15 A14 A13	Decoder Ouputs 76543210	Chip Select Active for Memor y IC
0000 - 1FFF	0 0 0	11111110	EPROM 0
2000 - 3FFF	0 0 1	11111101	EPROM 1
4000 - 5FFF	0 1 0	11111011	EPROM 2
6000 - 7FFF	0 1 1	11110111	EPROM 3
8000 - 9FFF	1 0 0	11101111	EPROM 4
A000 - BFFF	1 0 1	11011111	EPROM 5
C000 - DFFF	1 1 0	10111111	EPROM 6
E000 - FFFF	1 1 1	01111111	EPROM 7



Ports de I/O

- Ports de entrada necessitam de buffer tri-state pois o barramento de dados é compartilhado. P.ex. 74xx244
- Ports de saída necessitam reter a informação (latch/flip-flops).
  P.ex. 74xx74 ou 74xx374

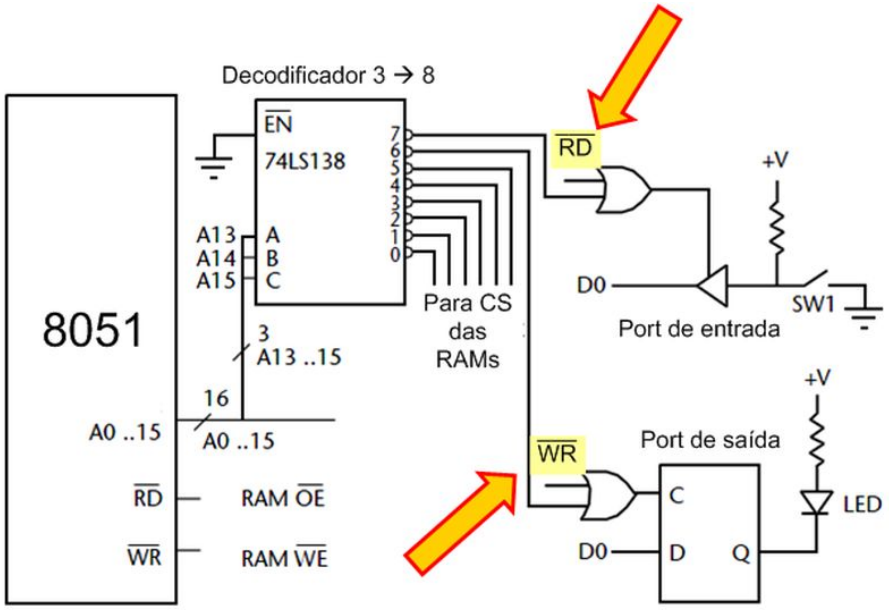




E/S mapeado em memória

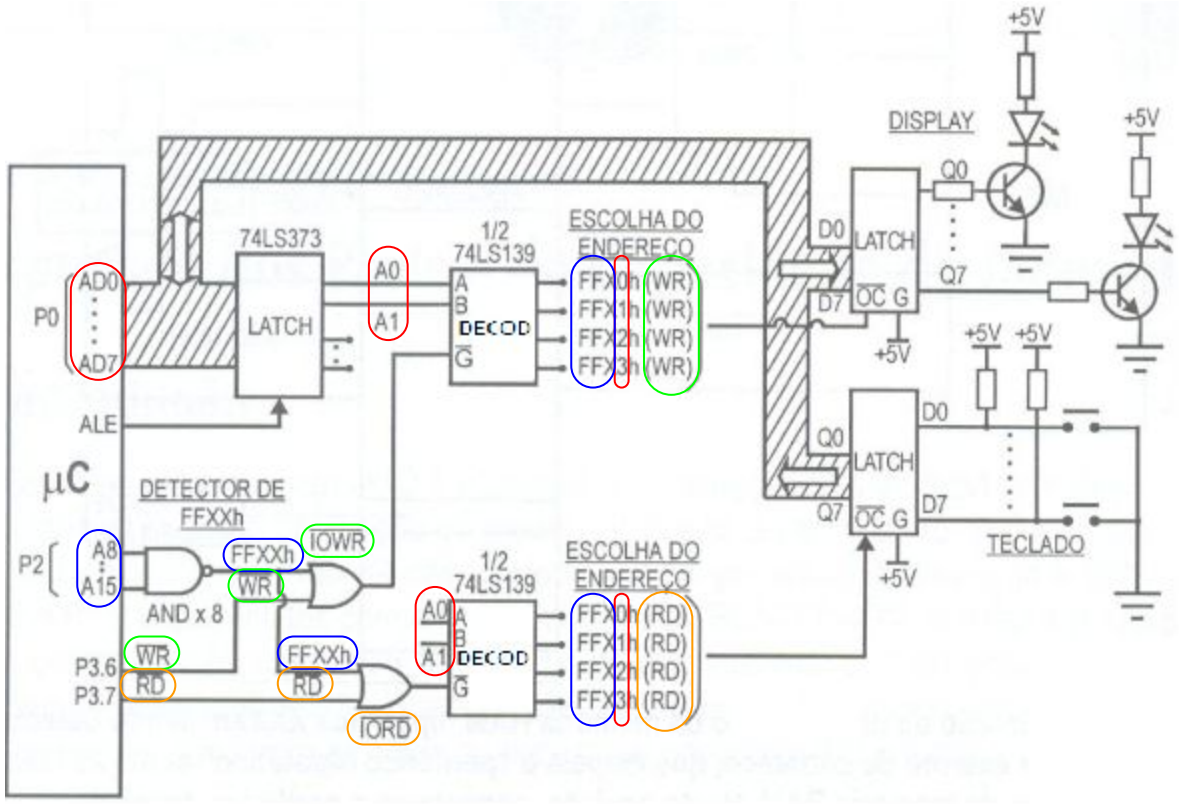
- Usa-se MOVX para acessar os ports E/S como se fossem posições de memória

Faixa de Endereçamento	Bits de seleção	Saídas do decodificador	Chip Select ativo
0000 - 1FFF	0 0 0	11111110	RAM 0
2000 - 3FFF	0 0 1	11111101	RAM 1
4000 - 5FFF	0 1 0	11111011	RAM 2
6000 - 7FFF	0 1 1	11110111	RAM 3
8000 - 9FFF	1 0 0	11101111	RAM 4
A000 - BFFF	1 0 1	11011111	RAM 5
C000 - DFFF	1 1 0	10111111	Output Port
E000 - FFFF	1 1 1	01111111	Input Port



I/O Mapeado em Memória

- **Exemplo:** Periféricos de entrada (teclado) e saída (*display* de 7 seg.)



Exemplo

Suponha que foi interconectado o endereço FFX0h nos latchs para leitura das chaves e o mesmo endereço para escrita no display.

Obs.: “Pode-se ter o mesmo endereço para os dois periféricos, pois um é de leitura e o outro de escrita, logo o \IORD e \IOWR separam um do outro”

Solução

```
INICIO: MOV   DPRT,#0FF00h ;carrego DPRT com
 endereço de I/O mapeado em FF00h
RETORNO: MOVX A,@DPTR ;"leio" o que está escrito no
 endereço de I/O mapeado das teclas
MOV 20h,A ;Salva o conteúdo de A, que é a imagem do
 estado das teclas, no registrador 20h. Este
 registrador é bit endereçável, com endereço de
 cada bit de 00h (tecla 0) até 07h (tecla 7)
jb 00h, TECLA1 ;testo cada bit das teclas. Se uma
 tecla foi apertada (bit=0), o software carrega no
 display o dado que desenha o número correspondente
 à tecla apertada. Se a tecla não foi apertada
 (bit=1), posso testar o próximo bit, isto é, a
 próxima tecla.
```

Solução

```
MOV A,#3Fh ;desenho do número 0 em display de 7
 segmentos
MOVX @DPTR,A ;carrego no endereço de memória FF00h
 o conteúdo de A, que no caso, temos neste endereço
 o display
TECLA1: JB 01h,TECLA2 ;testo bit da tecla1, se não
 for apertada, vou para tecla2
TECLA2: - mesmo código até a penúltima tecla
TECLA7: jb 07h, RETORNO ;teste última tecla, se não
 foi apertada volta para ler todas as teclas de
 novo
MOV A,#07h ;desenho do número 7
MOVX @DPTR,A ;carrego no display o número 7
JMP INICIO ;volto para o início.
```