Departamento de Engenharia Elétrica e de Computação
EESC-USP

SEL-415 Introdução à Organização de Computadores

Aula Exercícios
2ª. Prova

Profa. Luiza Maria Romeiro Codá

Autores: Prof. Dr. Marcelo Andrade da Costa Vieira
Profa. Maria Stela Veludo de Paiva
Profa. Luiza Maria Romeiro Codá

Reset da CPU

O reset da CPU é feito através do pino RST mantendo-se um nível alto por, no mínimo, 24 períodos de oscilador. A CPU responde executando um reset interno O reset da CPU coloca os seguintes valores nos registradores:

REGISTRADOR	CONTEÚDO
PC	0000 H
ACC	00 H
B	00 H
PSW	00 H
SP	07 H
DPTR	0000 H
P0 - P3	0FF H
IP(8051)	XXX00000 B
IP(8052)	XX000000 B
IE(8051)	0XX00000 B
IE(8052)	0X000000 B
TMOD	00 H
SCON	00 H

REGISTRADOR	CONTEÚDO
TCON	00 H
TH0	00 H
TL0	00 H
TH1	00 H
TL1	00 H
SBUF	INDETERMINADO
PCON (CMOS)	0XXX0000 B
PCON (NMOS)	0XXXXXXX B
TH2(8052)	00 H
TL2(8052)	00 H
RCAP2H(8052)	00 H
RCAP2L(8052)	00 H

Lista nº 11

15. Dado o segmento de programa da Figura 1 escrito em *assembly* para o microcontrolador Intel 8051. Responda:

15.1 Determine os valores de **End1, End2,End3 e dado1** para que: o endereço da instrução **MOV A, #dado1** seja **100CH**, que o endereço acessado na memória **RAM** seja 2009H, que o endereço acessado na memória **EPROM** seja **2010H** e que a rotina chamada inicie no endereço **1100H.**

```
        ORG   End1
        MOV   DPTR, #End2
        MOV A,#dado1
        MOVC  A,@A + DPTR
        MOV R0,A
        MOVX  A,@DPTR
        LCALL   ROT
        SJMP    $
        ORG   End3
ROT:    SUBB A, R0
        MOV   20H,A
        RET
        END
```

Figura 1

End da EEPROM	Conteúdo da EEPROM	significado	Número de bytes
End1	75H	Opcode de MOV DPTR, #End2	3
	LSB End2		
	MSB End2		
100CH	74H	Opcode de MOV A,#dado1	2
	Dado1	dado1 de 8 bits	
	93H	Opcode MOVC A,@A+DPTR	1
	F8H	Opcode de MOV R0,A	1
	F0H	MOVX A,@DPTR	1
	12H	Opcode LCALL ROT	3
	LSB ROT		
	MSB ROT		
	80H	Opcode da SJMP $	2

Lista nº 11

15.1 MOV A, #dado1 esta no **100CH**, endereço acessadomemória **RAM** = 2009H; o endereço acessado na memória **EPROM** =**2010H** e ROT=**1100H.**

```
        ORG  End1
        MOV  DPTR, #End2
        MOV A,#dado1
        MOVC A,@A + DPTR
        MOV R0,A
        MOVX A,@DPTR
        LCALL  ROT
        SJMP   $
        ORG  End3
ROT:    SUBB A, R0
        MOV  20H,A
        RET
        END
```

Figura 1

End da EEPROM	Conteúdo da EEPROM	significado	Numero De Bytes
ROT	98H	Opcode de SUBB A,R0	1
	E5H	Opcode de MOV 20H,A	2
	20H	20H= endereço acessado da RAM interna cujo conteúdo é copiada em A	
	22H	Opcode de RET	1

Lista11. 15.1 Resposta: **MOV A, #dado1** esta no **100CH**, endereço acessadomemória **RAM** = 2009H; o endereço acessado na memória **EPROM** =**2010H** e ROT=**1100H.**

```
        ORG  End1
        MOV  DPTR, #End2
        MOV A,#dado1
        MOVC A,@A + DPTR
        MOV R0,A
        MOVX A,@DPTR
        LCALL  ROT
        SJMP  $
        ORG  End3
ROT:    SUBB A, R0
        MOV  20H,A
        RET
        END
```

Figura 1

End da EEPROM	Conteúdo da EEPROM	significado	Número de bytes
End1	75H	Opcode de MOV DPTR, #End2	3
100AH	LSB End2		
100BH	MSB End2		
100CH	74H	Opcode de MOV A,#dado1	3
100EH	dado1	dado1 de 8 bits que é movido para o Ac.	
100FH	93H	Opcode MOVC A,@A+DPTR	1
1010H	F8H	Opcode de MOV R0,A	1
1011H	F0H	Opcode de MOVX A,@DPTR	1
1012H	12H	Opcode de LCALL ROT	3
1013H	LSB ROT	LSB do Endereço da subrotina	
1014H	MSB ROT	MSB do Endereço da subrotina	
1015H	80H	Opcode da SJMP $	2

Resp: End1 = 1009H; End2= 2009H,que é o valor do DPTR qdo a instrução MOVX A,@DPTR que acessa a RAM externa é executada;
Endereço acessado na EEPROM é 2100H através da MOVC A,@A+DPTR então A+ DPTR = 2010H, dado1 que foi armazenado em A = 2010H – 2009H = 07H

Lista nº 11

15.1 End 3?

MOV A, #dado1 esta no **100CH**, endereço acessado na memória **RAM** = 2009H; o endereço acessado na memória **EPROM** =**2010H** e ROT=**1100H.**

```
        ORG  End1
        MOV  DPTR, #End2
        MOV A,#dado1
        MOVC A,@A + DPTR
        MOV R0,A
        MOVX A,@DPTR
        LCALL  ROT
        SJMP   $
        ORG  End3
ROT:    SUBB A, R0
        MOV  20H,A
        RET
        END
```

Figura 1

End da EEPROM	Conteúdo da EEPROM	significado	Numero De Bytes
ROT	98H	Opcode de SUBB A,R0	1
1101H	E5H	Opcode de MOV 20H,A	2
1102H	20H		
1103H	22H	Opcode de RET	1

Resp: End3 é o endereço da sub rotina que é 1100H

Lista nº 11

15.2 Dado um “reset” no microcontrolador 80C51, em função das características do registrador SP (*stack pointer* ou ponteiro de pilha) responda qual o valor do SP, da pilha e do PC (Program Counter), após a instrução (LCALL) de chamada de subrotina ter sido executada, e quais esses valores após a instrução RET da subrotina ter sido executada?

```
        ORG   End1
        MOV   DPTR, #End2
        MOV A,#dado1
        MOVC  A,@A + DPTR
        MOV R0,A
        MOVX  A,@DPTR
        LCALL   ROT
        SJMP    $
        ORG   End3
ROT:    SUBB A, R0
        MOV   20H,A
        RET
        END
```

Figura 1

	Após execução da instrução LCALL ROT	Após execução da instrução RET
Valor de SP	09H (endereço de SP após armazenar o endereço para retorno no programa princuipal)	07H (recuperado endereço inicial de SP)
Valor da Pilha (conteúdo de SP)	10H (O MSB do endereço de retorno do programma principal (Instrução SJMP $ endereço 1016Hl)	Valor da posição 07H (não conhecido neste exercicio)
Valor do PC (Endereço da próxima instrução a ser executada)	1100H (endereço da subrotina)	1016H

Lista nº 11

15.3 Considerando o valor do ponteiro de pilha do ítem 15.2, quais os bancos de registradores que estão liberados?
s valores após a instrução RET da subrotina ter sido executada?

Resposta: Se SP = 07H então o banco 0 é que está liberado para utilização do programador, pois o naco 0 tem endereço de R0 a R7 de 00H a 07H, respecticamente

Lista nº 11

15.4 Qual (ou quais) instrução da Figura 1 apresenta(m) modo de endereçamento imediato. Explique porque?

Resposta:

Modos de Endereçamento do 8051

Endereçamento Imediato

- Opera sobre o dado localizado na própria instrução
- Identificado através do sinal #

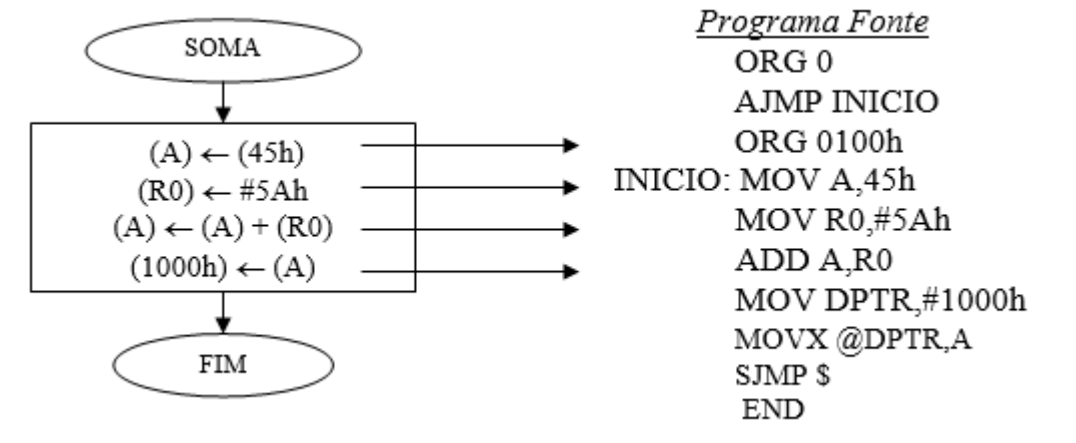
```
MOV DPTR,#End2 ; o ponteiro DPTR recebe o valor End2=2009H neste exercício

MOV A,#dado1 ; o valor de dado1 = 07H é armazenado no acumulador
```

Lista nº 11

16. Baseado na estrutura do processador 8051, onde o fluxograma e o assembly do programa que será executado, é mostrado na Figura 2, o qual executa a soma do valor 1Ah com o valor 5Ah e o resultado é armazenado na RAM externa. Responda as questões abaixo, lembrando que:

- AC: acumulador de 8 bits;
- DPTR de 16 bits;
- Ocorre um reset na CPU após a energização do microcontrolador;

16.1 Mostre quais os passos necessários para executar uma instrução localizada no endereço 0105h.



Lista 11. 16.1 Mostre quais os passos necessários para executar uma instrução localizada no endereço 0105h.

Programa Fonte

```
        ORG 0
        AJMP INICIO
        ORG 0100h
INICIO: MOV A,45h
        MOV R0,#5Ah
        ADD A,R0
        MOV DPTR,#1000h
        MOVX @DPTR,A
        SJMP $
        END
```

End da EEPROM	Conteúdo da EEPROM	significado	Nº bytes
0000H	Opcode AJMP	Salta para outro endereço numa faixa de 2Kbytes referentes à atual posição da memória de programa.(salta p endereço INICIO)	2
	2ª byte AJMP	Define o endereço de salto 0100H	
0100H	Opcode de MOV A, 45H	Copia o conteúdo da posição 45H da RAM interna no acumulador	2
0101H	45H		
0102H	Opcode de MOV R0, #5AH	Copia o valor do 2º byte da instrução(5AH) em R0	2
0103H	5AH	Valor	
0104H	Opcode de ADD A,RO	Soma o conteúdo de R0 ao conteúdo do acumulador e salva resultado no acumulador	1
0105H	Opcode do MOV DPTR, #1000H	Opcode LCALL ROT	3
0106H	00H	LSB do valor da instrução	
0107H	10H	MSB do valor da instrução	
	Opcode de MOVX @DPTR,A	Copia no endereço da RAM externa contido em DPTR o valor do acumulador(soma de r0 com A)	1
	Opcode do SJMP $	Salta p o endereço relativo ao endereço da instrução -128 a 128	2
	Endereço da relativo do opcode a esta posição		

Lista 11. 16.1 Mostre quais os passos necessários para executar uma instrução localizada no endereço 0105h.

Resp:

O PC é carregado com endereço da instrução que será executada(0105H).

No chip do microprocessador o pino /PSEN é colocado em nível baixo, sinalizando que será acessada memória de Programa(EEPROM), e o pino ALE é colocado em nível alto, indicando a presença de endereço no duto multiplexado (P0) . O microprocessador coloca então oso valores do 8 bits LSB (05H) do endereço na porta paralela P0 e dos MSBs na porta paralela P2.

Os 8 bits menos significativos do endereço são armazenado em um latch, para que esse endereço esteja prersente na entrada de endereços da memória de programa (EEPROM) durante todo o tempo de acesso à memória e possibilite seu cionteúdo ser acessado.

Então, o microprocessador coloca o pino ALE em nível baixo indicando que a porta multiplexada P0 foi transformada em duto de dados e está apta a receber os dados da memória.

O conteúdo da posição cujo endereço está presente nas entradas de endereço da memória é acessado.e apresentada nas saídas de dados da memória. O conteúdo dessa posição que corresponde ao OPCODE da instrução MOV DPTR,#1000H (cujo valor é 90H), transita agora pelo duto multiplexado da porta P0, sendo esse valor carregado no registrador de instrução(RI) na CPU. O valor desse opcode é decodificado pelo processador e este processo define se a instrução tem apenas 1 ou mais bytes. No caso desta instrução, são 3 bytes, então o PC é incrementado, a porta P0 passa a conter endereço e o processo de busca na EEPROM se repete para os dois bytes seguintes cujo valor são 00H e 10H.

A´pós finalizada a busca dos dois próximaos bytes, a execução da instrução é finalizada armazenando o valor desses dois bytes em DPL e DPH, e portnato DPTR = 1000H

Programa Fonte

```
        ORG 0
        AJMP INICIO
        ORG 0100h
INICIO: MOV A,45h
        MOV R0,#5Ah
        ADD A,R0
        MOV DPTR,#1000h
        MOVX @DPTR,A
        SJMP $
        END
```

0105H	Opcode do MOV DPTR, #1000H	Opcode LCALL ROT	3
0106H	00H	LSB do valor da instrução	
0107H	10H	MSB do valor da instrução	

Lista 11. 16.2 Qual o conteúdo da memória principal após a execução programa (memória de programa)?

Programa Fonte

```
        ORG 0
        AJMP INICIO
        ORG 0100h
INICIO: MOV A,45h
        MOV R0,#5Ah
        ADD A,R0
        MOV DPTR,#1000h
        MOVX @DPTR,A
        SJMP $
        END
```

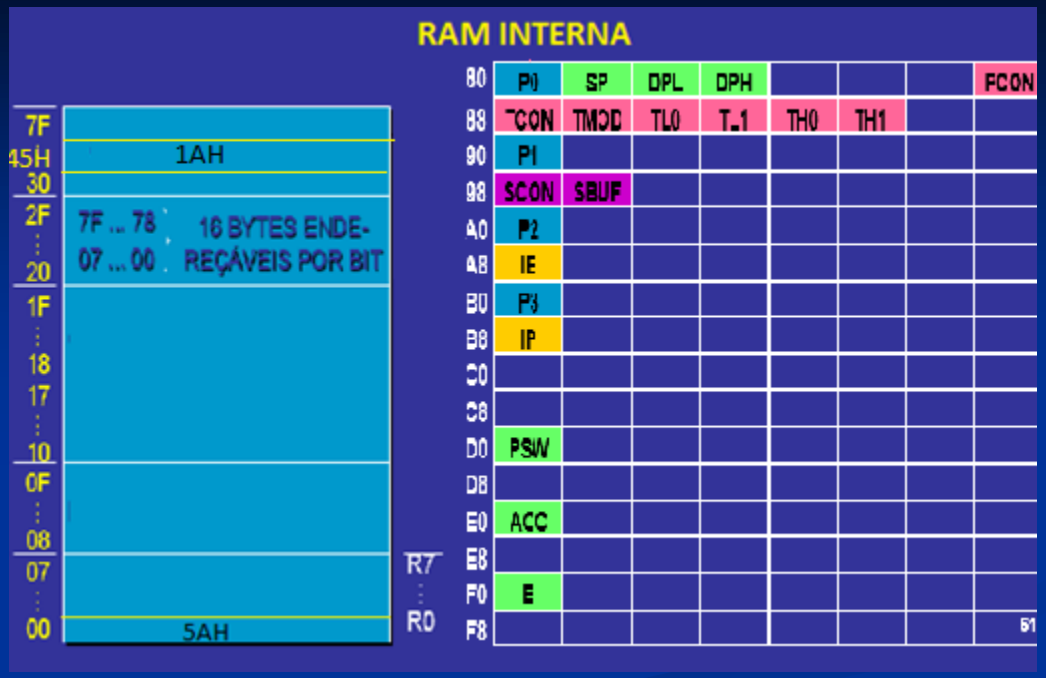
End da EEPROM	Conteúdo da EEPROM	significado	Nº bytes
0000H	Opcode AJMP (01H, 21H, 41H, 61H, 81H, A1H, C1H ou E1H)	Pula para a posição INICIO e o pcode depende do tamanho do salto	2
	2ª byte AJMP	O valor depende do endereço INICIO em relação à instrução	
0100H	E5H	Opcode de MOV A, 45H	2
0101H	45H	45H	
0102H	78H	Opcode de MOV R0, #5AH	2
0103H	5AH	Valor do 2º byte da instruçao	
0104H	28H	Opcode de ADD A,RO	1
0105H	90H	Opcode do MOV DPTR, #1000H	3
0106H	00H	00H	
0107H	10H	10H	
0108H	F0H	Opcode de MOVX @DPTR,A	1
0109H	80H	Opcode do SJMP $	2
010AH	FEH	Endereço da relativo do opcode a esta posição	

Lista 11. **16.3** Qual o conteúdo da RAM interna após execução deste programa ?

16.4 Ao final da execução do programa qual o valor da posição 1000H da RAM externa?

Programa Fonte

```
        ORG 0
        AJMP INICIO
        ORG 0100h
INICIO: MOV A,45h
        MOV R0,#5Ah
        ADD A,R0
        MOV DPTR,#1000h
        MOVX @DPTR,A
        SJMP $
        END
```



16.3 Resp: A = 74H DPL = 00H DPH= 10H R0 = 5AH conteúdo da posição 45H = 1AH
SP= 07H

16.4 Resp: o conteúdo da posição 1000H da RAM externa é 74H

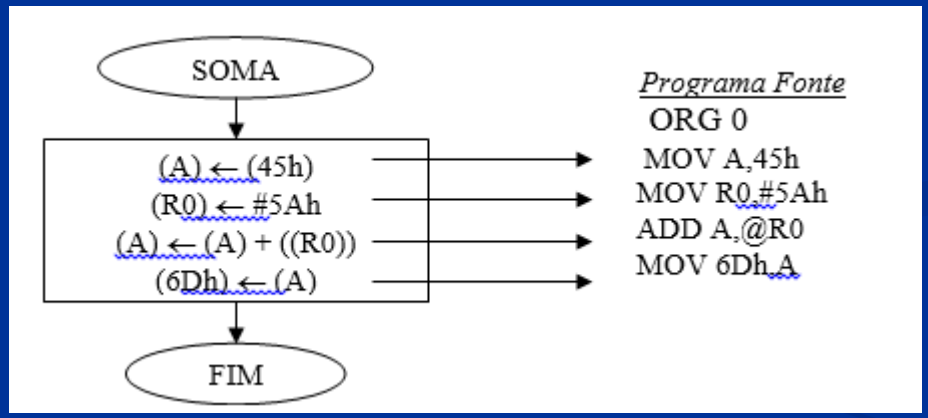
Lista 11. (proposto)17. Baseado na estrutura do processador 8051, onde o fluxograma e o assembly do programa que será executado, é mostrado na Figura 3, o qual executa a soma do valor 1Ah com o valor 7Fh e o resultado é armazenado na posição 6Dh da RAM interna. Responda as questões abaixo, lembrando que:

- AC: acumulador de 8 bits;
- DPTR de 16 bits;
- Ocorre um reset na CPU após a energização do microcontrolador;

17.1 Qual o conteúdo da memória principal para esse programa (memória de programa)?

17.2 Qual o conteúdo da RAM interna para esse programa ?

17.3 Qual o valor do PC (Counter Program) quando a instrução ADD A,@R0 está sendo executada?



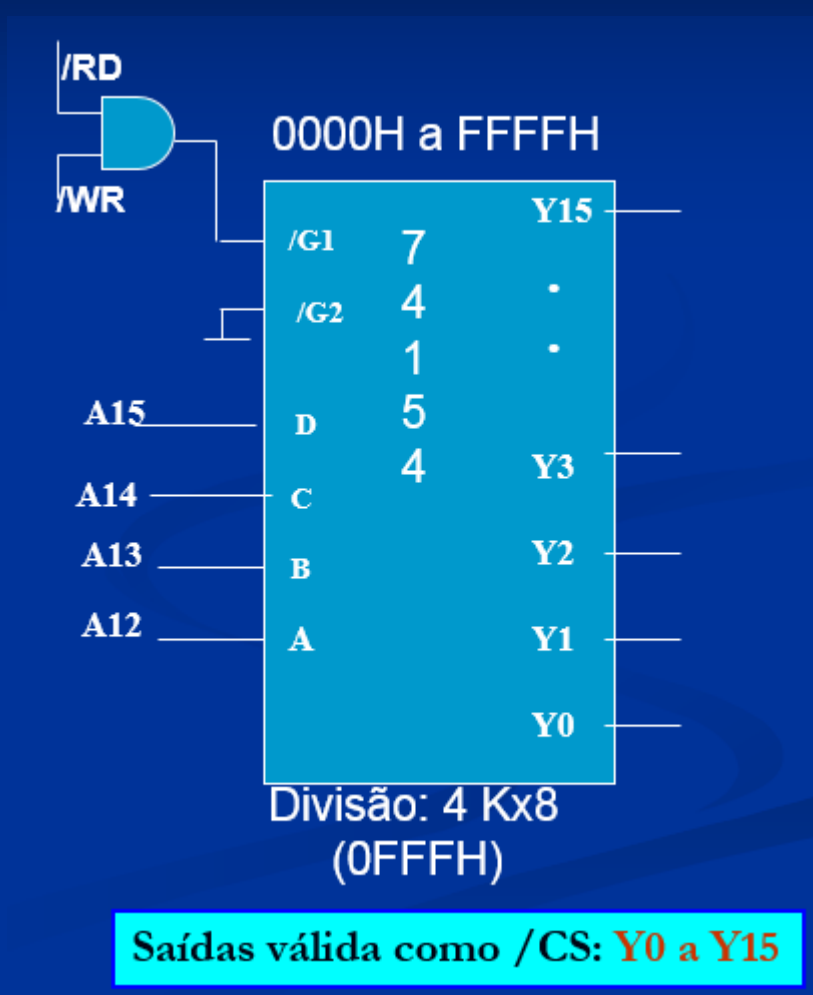
LISTA DE EXERCÍCIOS

1. Faça o projeto da lógica de seleção para dividir o espaço de endereçamento de um microprocessador de 16bits de linhas de endere;o e 8 bits de linhas de dados, em blocos de **4Kbytes,** especificando endereço inicial e final de cada bloco.

Saída do 74154	Faixa do endereços
Y0	0000H a 0FFFH
Y1	1000H a 1FFFH
Y2	2000H a 2FFFH
Y3	3000H a 3FFFH
Y4	4000H a 4FFFH
Y5	5000H a 5FFFH
Y6	6000H a63FFFH
Y7	7000H a 7FFFH
Y8	8000H a 8FFFH
Y8	9000H a 9FFFH
Y10	A000H a AFFFH
Y11	B000H a BFFFH
Y12	C000H a CFFFH
Y13	D000H a DFFFH
Y14	E000H a EFFFH
Y15	F000H a FFFFH

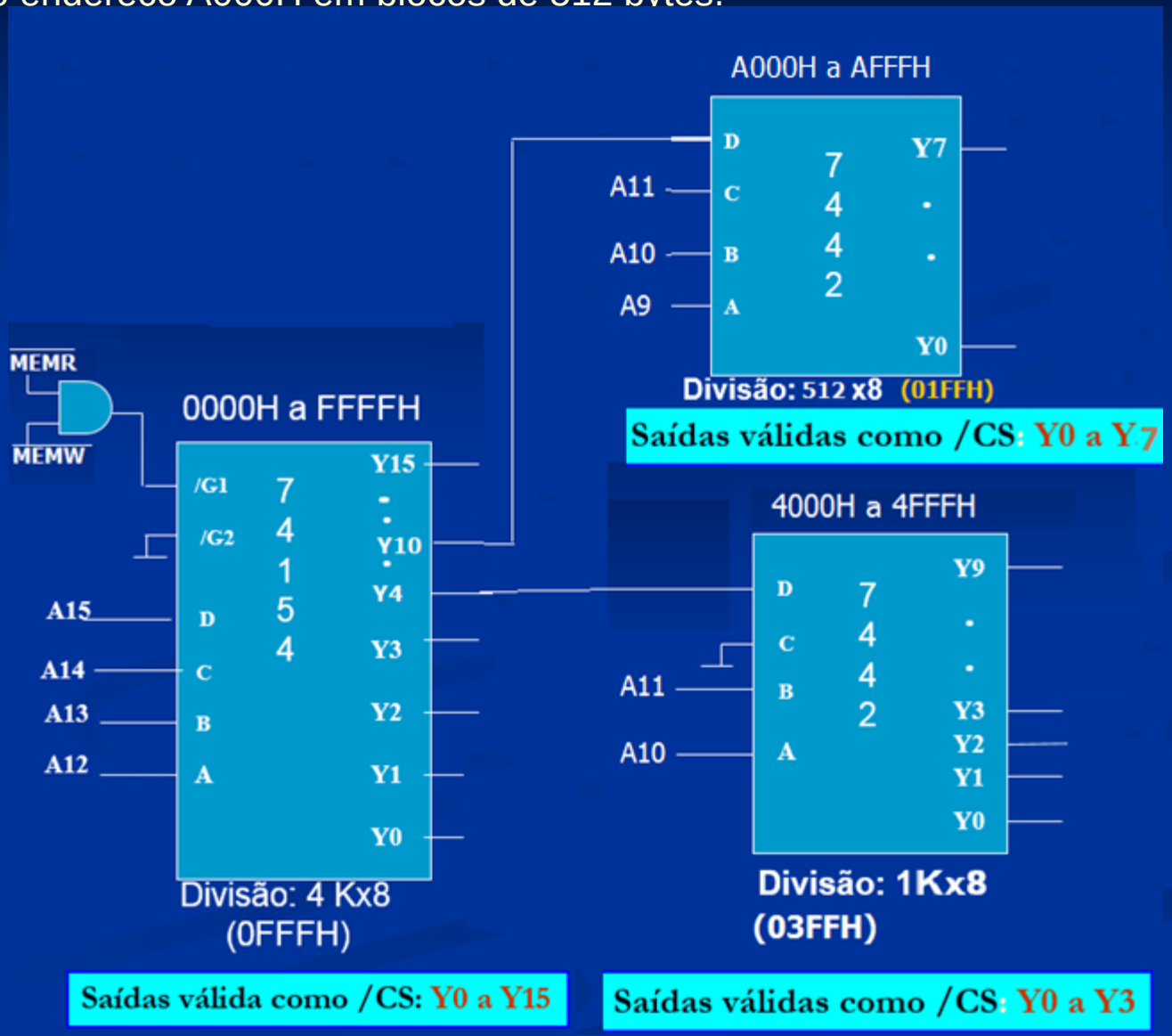
LISTA DE EXERCÍCIOS

1. Resp : Faça o projeto da lógica de seleção para dividir o espaço de endereçamento de um microprocessador de 16bits de linhas de endere;o e 8 bits de linhas de dados, em blocos de **4Kbytes,** especificando endereço inicial e final de cada bloco.



LISTA DE EXERCÍCIOS

1. (cont.) Divida o bloco que inicia no endereço 4000H, em blocos de 1Kbytes e o bloco que inicia no endereco A000H em blocos de 512 bytes.



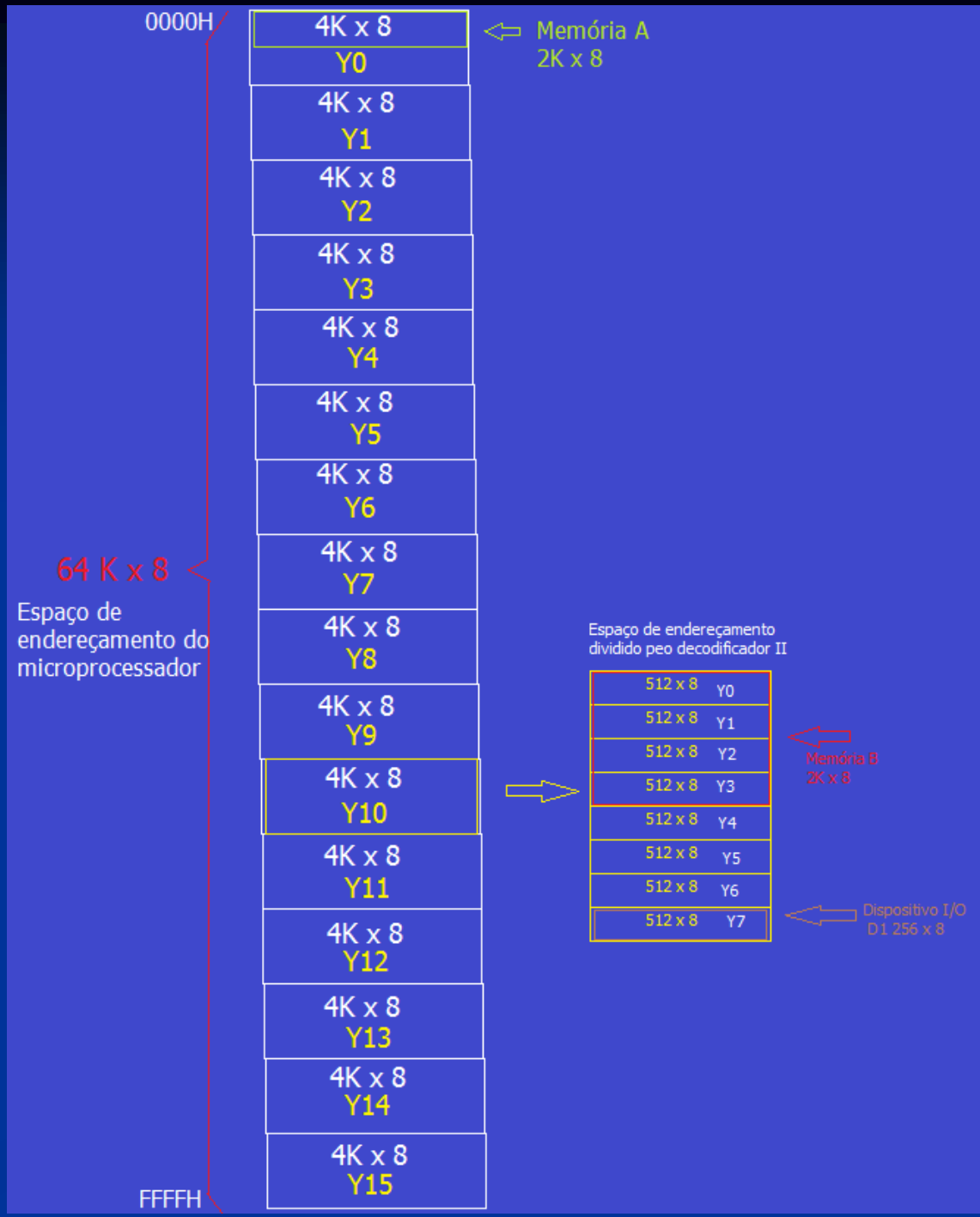
1. (cont.) Usando **lógica de seleção absoluta e mapeamento em memória**, ligue uma memória de 2kbytes, a partir do endereço 0000H, outra a partir do endereço A000H, e um dispositivo de IO de 256x8 a partir do endereço AE00H. Determine o endereço final de cada memória e dispositivo ligado.

Resp: Na lógica de seleção absoluta não podem existir espaços fantasmas(espelhos), o tamanho do espaço selecionado pelos decodificadore, através do endereço enviado pelo muicroprocessador, deve ser do mesmo tamanho do dispositivo ou memória acessados.

E, mapeamento em memória o microprocessador endereça no mesmo espaço de endereçamento memórias e dispositivos de I/O, no caso do microprocessador deste exercício o espaço é de 64K x 8

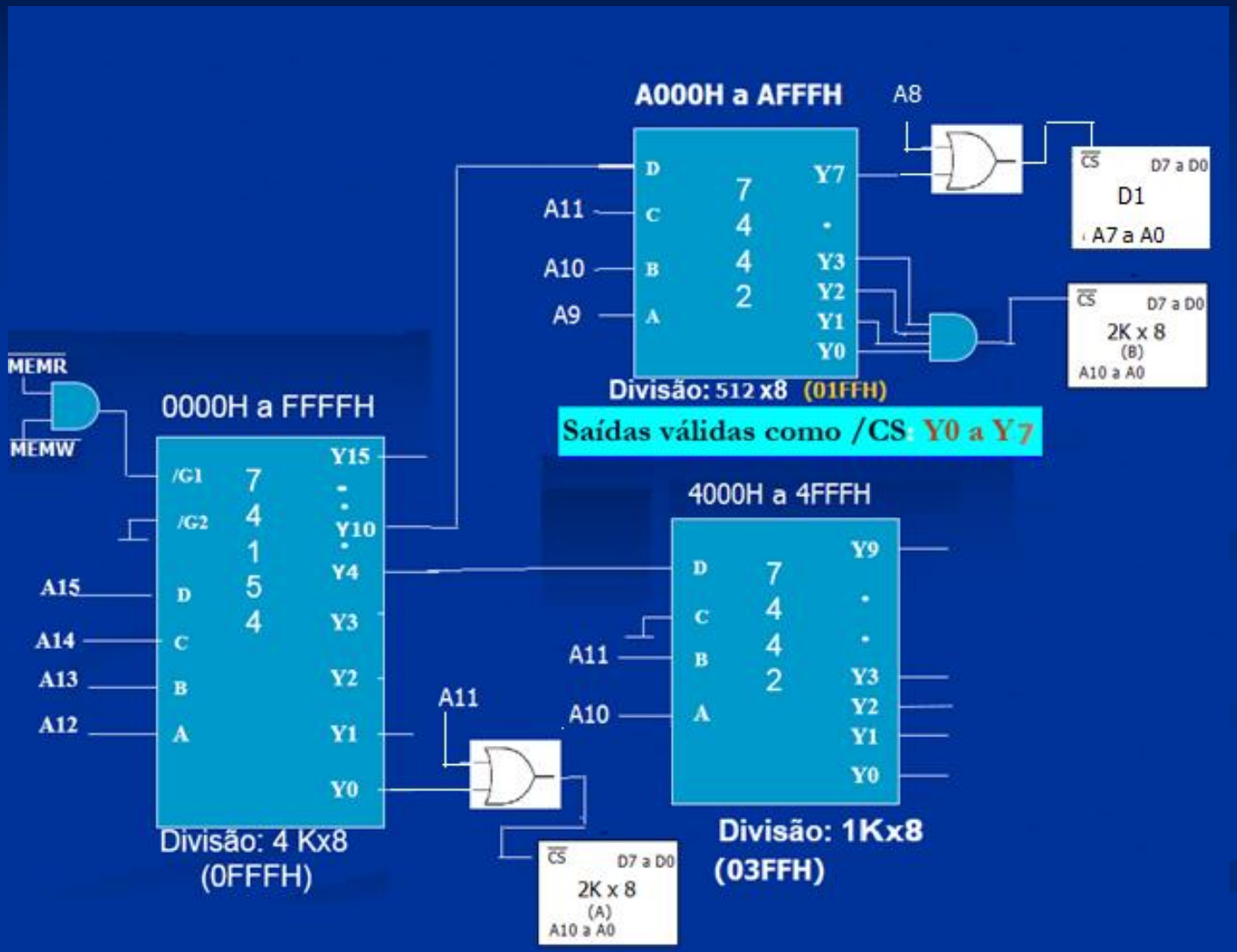
Resp(cont.): Divisão do espaço de endereçamento e faixas de endereços das memórias e dispositivos

	Faixa do endereços
MemA	0000H a 07FFH
MemB	A000H a A3FFH
D1	AE00H a AEFFH

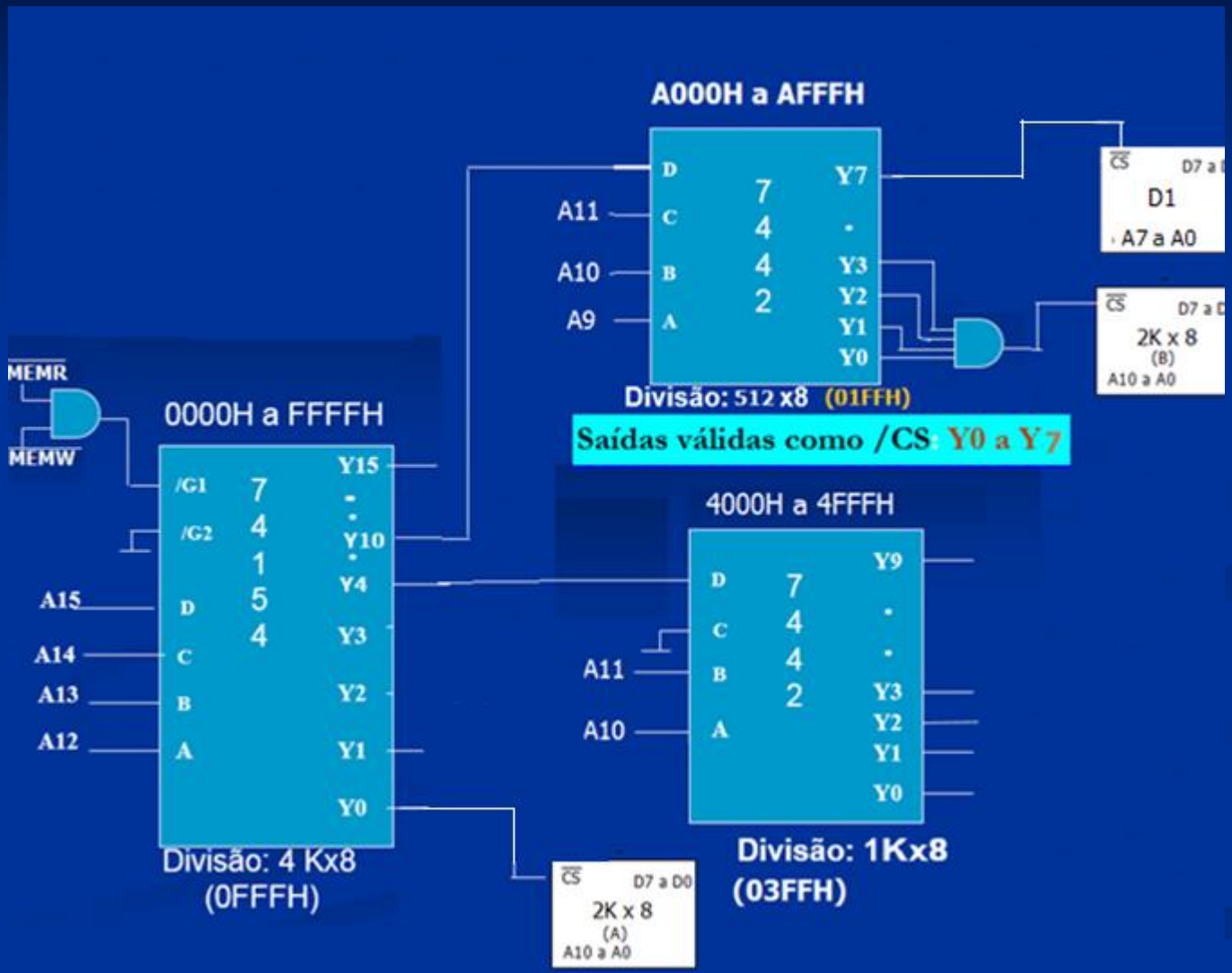


1

Resp: lógica de seleção absoluta



1. (cont.) Usando **lógica de seleção NÃO absoluta**, ligue uma memória de 2kbytes, a partir do endereço 0000H, outra a partir do endereço A000H e um dispositivo de IO de 256x8 a partir do endereço AE00H. Determine o endereço final de cada memória e dispositivo ligado e faixas fantasmas se houverem.



1. (cont.) Usando **lógica de seleção NÃO absoluta**, ligue uma memória de 2kbytes, a partir do endereço 0000H, outra a partir do endereço 4000H e outra a partir do endereço A000H, determinando o endereço final de cada memória.

	Faixa do endereços	Faixa espelho (ou fantasma)	Quantidae de faixas fantasmas
MemA	0000H a 0FFFH	0800H a 0FFFH	1 de 2K Bytes
MemB	A000H a A3FFH	A000H a A3FFH	zero
D1	AE00H a AFFFH	AF00H a AFFFH	1 de 256 bytes

LISTA DE EXERCÍCIOS

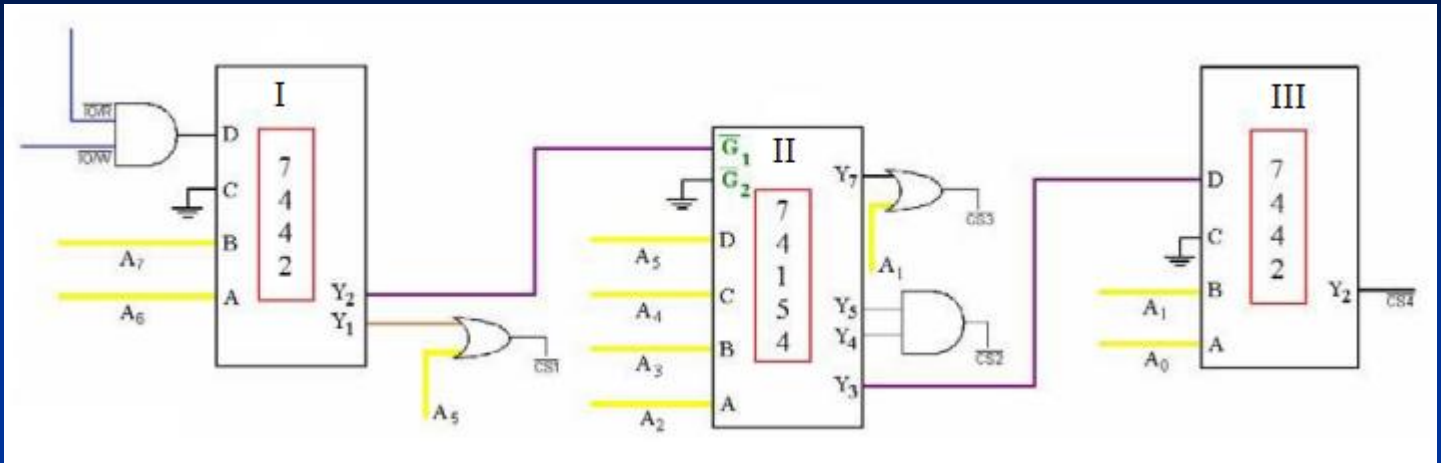
1. Faça a lógica de seleção Absoluta com mapeamento em I/O isolado considerando que um microprocessador tem a capacidade de endereçar um espaço de 256x8 para dispositivos de I/O.

- ✓ Utilizando o decodificador 7442(4X10), decodificador I, divida o espaço de endereçamento em 64 x 8 e ligue um dispositivo de tamanho 32 x8 na saída /CS1 que contém o endereço 00H;
- ✓ Utilizando o decodificador 74154(4X16), decodificador II, divida a saída que coontém o endereço 4DH em faixas de tamanho 4 x 8.
- ✓ Com as saídas do decodificador II, gere uma saída /CS2 que possibilite a ligação de dispositivos de 8 x 8 que contenha os endereçoes 92H e 95HH.
- ✓ Utilizando a saída Y7 do decodificador II gere uma saída /CS3 que possibilite ligar dispositivos de 2 x 8
- ✓ Utilizando um decodificador III, crie uma saída /CS4 que selecione dispositivos de 1x8 no endereço 8EH

	A7	A6	A5	A4	A3	A2	A1	A0	Faixa da saída	Tamanho do bloco	Faixa das saídas em hexadecimal
			1	1	1	1	1	1	3FH	64 x8	
Decoder I	0	0	0	0	0	0	0	0			00H a FFH
	1	1	1	1	1	1	1	1			
/CS0	0	0	0	0	0	0	0	0	1FH	32 x 8	00H a 1FH
	0	0	0	1	1	1	1	1			
Decoder II	1	0	0	1	0	0	1	0	92H		Saída Y4 do Decoder II
/CS2	1	0	0	1	0	1	0	1	95H		Saída Y5 do Decoder II
/CS3	1	0	0	1	1	1	0	0	9CH	2x8	9CH a 9DH
	1	0	0	1	1	1	0	1	9DH		
/CS4	1	0	0	0	1	1	1	0	8EH	1x8	Saída Y2 do decoder III

LISTA DE EXERCÍCIOS

2. Resp : Lógica de Seleção absoluta com maperamento em I/O isolado



Conjunto de instruções do 8051

Hex Code	Number of Bytes	Mnemo-nic	Operands
00	1	NOP	
01	2	AJMP	code addr
02	3	LJMP	code addr
03	1	RR	A
04	1	INC	A
05	2	INC	data addr
06	1	INC	@R0
07	1	INC	@R1
08	1	INC	R0
09	1	INC	R1
0A	1	INC	R2
0B	1	INC	R3
0C	1	INC	R4
0D	1	INC	R5
0E	1	INC	R6
0F	1	INC	R7
10	3	JBC	bit addr, code addr
11	2	ACALL	code addr
12	3	LCALL	code addr
13	1	RRC	A
14	1	DEC	A
15	2	DEC	data addr
16	1	DEC	@R0
17	1	DEC	@R1
18	1	DEC	R0
19	1	DEC	R1
1A	1	DEC	R2
1B	1	DEC	R3
1C	1	DEC	R4
1D	1	DEC	R5
1E	1	DEC	R6
1F	1	DEC	R7
20	3	JB	bit addr, code addr
21	2	AJMP	code addr
22	1	RET	
23	1	RL	A
24	2	ADD	A,#data
25	2	ADD	A,data addr
26	1	ADD	A,@R0
27	1	ADD	A,@R1
28	1	ADD	A,R0
29	1	ADD	A,R1
2A	1	ADD	A,R2
2B	1	ADD	A,R3
2C	1	ADD	A,R4
2D	1	ADD	A,R5
2E	1	ADD	A,R6
2F	1	ADD	A,R7
30	3	JNB	bit addr, code addr
31	2	ACALL	code addr
32	1	RETI	
33	1	RLC	A
34	2	ADDC	A,#data
35	2	ADDC	A,data addr
36	1	ADDC	A,@R0
37	1	ADDC	A,@R1
38	1	ADDC	A,R0
39	1	ADDC	A,R1
3A	1	ADDC	A,R2
3B	1	ADDC	A,R3
3C	1	ADDC	A,R4
3D	1	ADDC	A,R5
3E	1	ADDC	A,R6
3F	1	ADDC	A,R7
40	2	JC	code addr
41	2	AJMP	code addr
42	2	ORL	data addr,A
43	3	ORL	data addr,#data
44	2	ORL	A,#data
45	2	ORL	A,data addr
46	1	ORL	A,@R0
47	1	ORL	A,@R1
48	1	ORL	A,R0
49	1	ORL	A,R1
4A	1	ORL	A,R2
4B	1	ORL	A,R3
4C	1	ORL	A,R4
4D	1	ORL	A,R5
4E	1	ORL	A,R6
4F	1	ORL	A,R7
50	2	JNC	code addr
51	2	ACALL	code addr
52	2	ANL	data addr,A
53	3	ANL	data addr,#data
54	2	ANL	A,#data
55	2	ANL	A,data addr
56	1	ANL	A,@R0
57	1	ANL	A,@R1
58	1	ANL	A,R0
59	1	ANL	A,R1
5A	1	ANL	A,R2
5B	1	ANL	A,R3
5C	1	ANL	A,R4
5D	1	ANL	A,R5
5E	1	ANL	A,R6
5F	1	ANL	A,R7
60	2	JZ	code addr
61	2	AJMP	code addr

Conjunto de instruções do 8051(cont.)

62	2	**XRL**	data addr,A
63	3	**XRL**	data addr,#data
64	2	**XRL**	A,#data
65	2	**XRL**	A,data addr
66	1	**XRL**	A,@R0
67	1	**XRL**	A,@R1
68	1	**XRL**	A,R0
69	1	**XRL**	A,R1
6A	1	**XRL**	A,R2
6B	1	**XRL**	A,R3
6C	1	**XRL**	A,R4
6D	1	**XRL**	A,R5
6E	1	**XRL**	A,R6
6F	1	**XRL**	A,R7
70	2	**JNZ**	code addr
71	2	**ACALL**	code addr
72	2	**ORL**	C,bit addr
73	1	**JMP**	@A+DPTR
74	2	**MOV**	A,#data
75	3	**MOV**	data addr,#data
76	2	**MOV**	@R0,#data
77	2	**MOV**	@R1,#data
78	2	**MOV**	R0,#data
79	2	**MOV**	R1,#data
7A	2	**MOV**	R2,#data
7B	2	**MOV**	R3,#data
7C	2	**MOV**	R4,#data
7D	2	**MOV**	R5,#data
7E	2	**MOV**	R6,#data
7F	2	**MOV**	R7,#data
80	2	**SJMP**	code addr
81	2	**AJMP**	code addr
82	2	**ANL**	C,bit addr
83	1	**MOVC**	A,@A+PC
84	1	**DIV**	AB
85	3	**MOV**	data addr, data addr
86	2	**MOV**	data addr,@R0
87	2	**MOV**	data addr,@R1
88	2	**MOV**	data addr,R0
89	2	**MOV**	data addr,R1
8A	2	**MOV**	data addr,R2
8B	2	**MOV**	data addr,R3
8C	2	**MOV**	data addr,R4
8D	2	**MOV**	data addr,R5
8E	2	**MOV**	data addr,R6
8F	2	**MOV**	data addr,R7
90	3	**MOV**	DPTR,#data
91	2	**ACALL**	code addr
92	2	**MOV**	bit addr,C
93	1	**MOVC**	A,@A+DPTR
94	2	**SUBB**	A,#data
95	2	**SUBB**	A,data addr
96	1	**SUBB**	A,@R0
97	1	**SUBB**	A,@R1
98	1	**SUBB**	A,R0
99	1	**SUBB**	A,R1
9A	1	**SUBB**	A,R2
9B	1	**SUBB**	A,R3
9C	1	**SUBB**	A,R4
9D	1	**SUBB**	A,R5
9E	1	**SUBB**	A,R6
9F	1	**SUBB**	A,R7
A0	2	**ORL**	C,/bit addr
A1	2	**AJMP**	code addr
A2	2	**MOV**	C,bit addr
A3	1	**INC**	DPTR
A4	1	**MUL**	AB
A5		**reserved**	
A6	2	**MOV**	@R0,data addr
A7	2	**MOV**	@R1,data addr
A8	2	**MOV**	R0,data addr
A9	2	**MOV**	R1,data addr
AA	2	**MOV**	R2,data addr
AB	2	**MOV**	R3,data addr
AC	2	**MOV**	R4,data addr
AD	2	**MOV**	R5,data addr
AE	2	**MOV**	R6,data addr
AF	2	**MOV**	R7,data addr
B0	2	**ANL**	C,/bit addr
B1	2	**ACALL**	code addr
B2	2	**CPL**	bit addr
B3	1	**CPL**	C
B4	3	**CJNE**	A,#data,code addr
B5	3	**CJNE**	A,data addr,code addr
B6	3	**CJNE**	@R0,#data,code addr
B7	3	**CJNE**	@R1,#data,code addr
B8	3	**CJNE**	R0,#data,code addr
B9	3	**CJNE**	R1,#data,code addr
BA	3	**CJNE**	R2,#data,code addr
BB	3	**CJNE**	R3,#data,code addr
BC	3	**CJNE**	R4,#data,code addr
BD	3	**CJNE**	R5,#data,code addr
BE	3	**CJNE**	R6,#data,code addr
BF	3	**CJNE**	R7,#data,code addr
C0	2	**PUSH**	data addr
C1	2	**AJMP**	code addr
C2	2	**CLR**	bit addr
C3	1	**CLR**	C
C4	1	**SWAP**	A
C5	2	**XCH**	A,data addr
C6	1	**XCH**	A,@R0
C7	1	**XCH**	A,@R1

Conjunto de instruções do 8051(cont)

C8	1	**XCH**	A,R0
C9	1	**XCH**	A,R1
CA	1	**XCH**	A,R2
CB	1	**XCH**	A,R3
CC	1	**XCH**	A,R4
CD	1	**XCH**	A,R5
CE	1	**XCH**	A,R6
CF	1	**XCH**	A,R7
D0	2	**POP**	data addr
D1	2	**ACALL**	code addr
D2	2	**SETB**	bit addr
D3	1	**SETB**	C
D4	1	**DA**	A
D5	3	**DJNZ**	data addr,code addr
D6	1	**XCHD**	A,@R0
D7	1	**XCHD**	A,@R1
D8	2	**DJNZ**	R0,code addr
D9	2	**DJNZ**	R1,code addr
DA	2	**DJNZ**	R2,code addr
DB	2	**DJNZ**	R3,code addr
DC	2	**DJNZ**	R4,code addr
DD	2	**DJNZ**	R5,code addr
DE	2	**DJNZ**	R6,code addr
DF	2	**DJNZ**	R7,code addr
E0	1	**MOVX**	A,@DPTR
E1	2	**AJMP**	code addr
E2	1	**MOVX**	A,@R0
E3	1	**MOVX**	A,@R1
E4	1	**CLR**	A
E5	2	**MOV**	A,data addr
E6	1	**MOV**	A,@R0
E7	1	**MOV**	A,@R1
E8	1	**MOV**	A,R0
E9	1	**MOV**	A,R1
EA	1	**MOV**	A,R2
EB	1	**MOV**	A,R3
EC	1	**MOV**	A,R4
ED	1	**MOV**	A,R5
EE	1	**MOV**	A,R6
EF	1	**MOV**	A,R7
F0	1	**MOVX**	@DPTR,A
F1	2	**ACALL**	code addr
F2	1	**MOVX**	@R0,A
F3	1	**MOVX**	@R1,A
F4	1	**CPL**	A
F5	2	**MOV**	data addr,A
F6	1	**MOV**	@R0,A
F7	1	**MOV**	@R1,A
F8	1	**MOV**	R0,A
F9	1	**MOV**	R1,A
FA	1	**MOV**	R2,A
FB	1	**MOV**	R3,A
FC	1	**MOV**	R4,A
FD	1	**MOV**	R5,A
FE	1	**MOV**	R6,A
FF	1	**MOV**	R7,A

Instruction Opcodes in Hexadecimal Order

FIM