Departamento de Engenharia Elétrica e de Computação
EESC-USP

SEL-415 Introdução à Organização de Computadores

Princípio de Operação de memórias
Expansão de Memórias
Lógica de Seleção

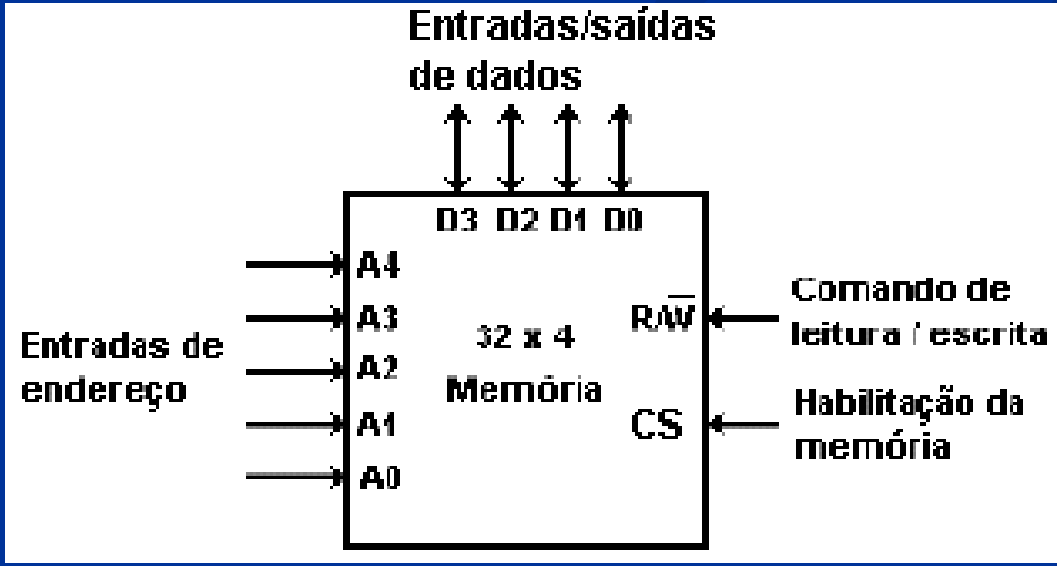
Aula 5

Profa. Luiza Maria Romeiro Codá

Memórias Semicondutoras

PRINCÍPIOS DE OPERAÇÃO DAS MEMÓRIAS

- Selecionar o endereço a ser acessado (leitura ou escrita);
- Se a operação for escrita, fornecer os dados de entrada;
- Se a operação for leitura, os dados estarão disponíveis na saída;
- Habilitar a memória (*CS*) para que as portas de I/O sejam liberadas para a operação desejada;
- Selecionar o tipo de operação: leitura ou escrita ($R/\overline{W}$);

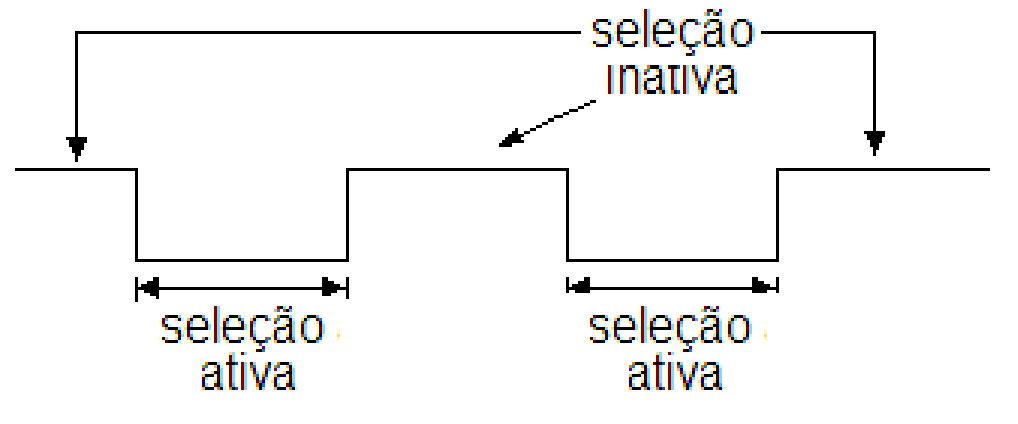


Sinais nos pinos de controle

Sinal de Habilitação:

$\overline{ME}$ – *Memory Enable*
$\overline{E}$ – *Enable*
$\overline{CS}$ – *Chip Select*

É um sinal de seleção, ativo em "0" → seleciona o dispositivo.
Se colocado em nível "1" → desabilita o dispositvo , geralmente colocando em estado de alta-impedância (tristate).

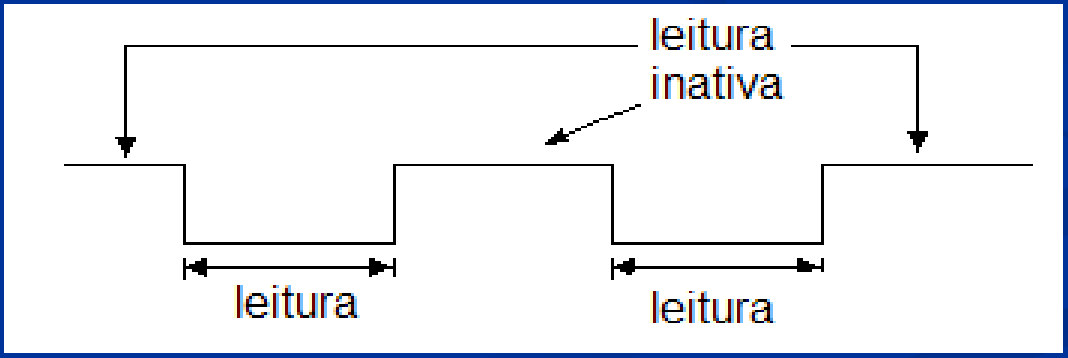


Sinais nos pinos de controle

Sinal de Leitura:

$\overline{RD}$ – *Read*

É um sinal de leitura, ativo em nível lógico "0" → Coloca o dado armazenado na memória, na posição definida no duto de endereços, no duto de dados.

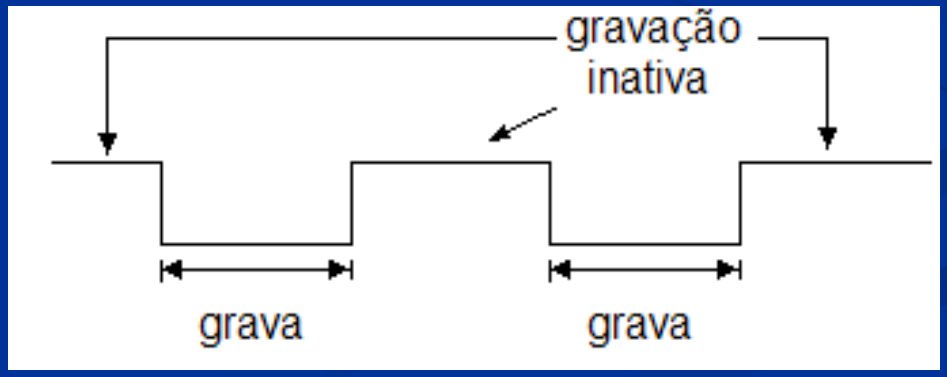


Sinais nos pinos de controle

Sinal de Escrita:

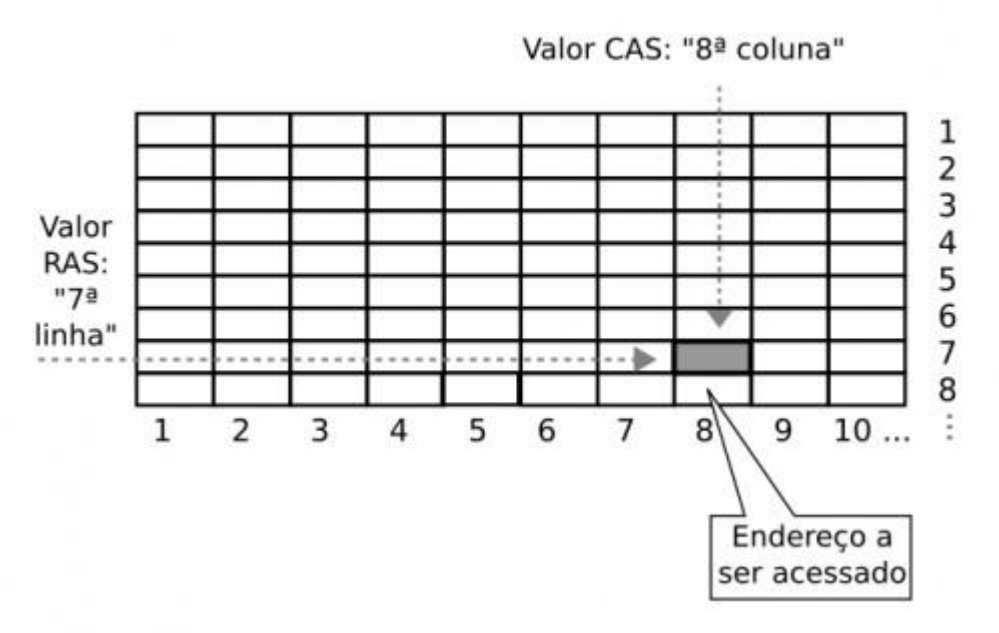
$\overline{W}$ – *Write*

É um sinal de escrita (gravação), ativo em nível lógico "0" → Armazena o dado presente no duto de dados na posição de memória definida no duto de endereços.



Acesso à Memória

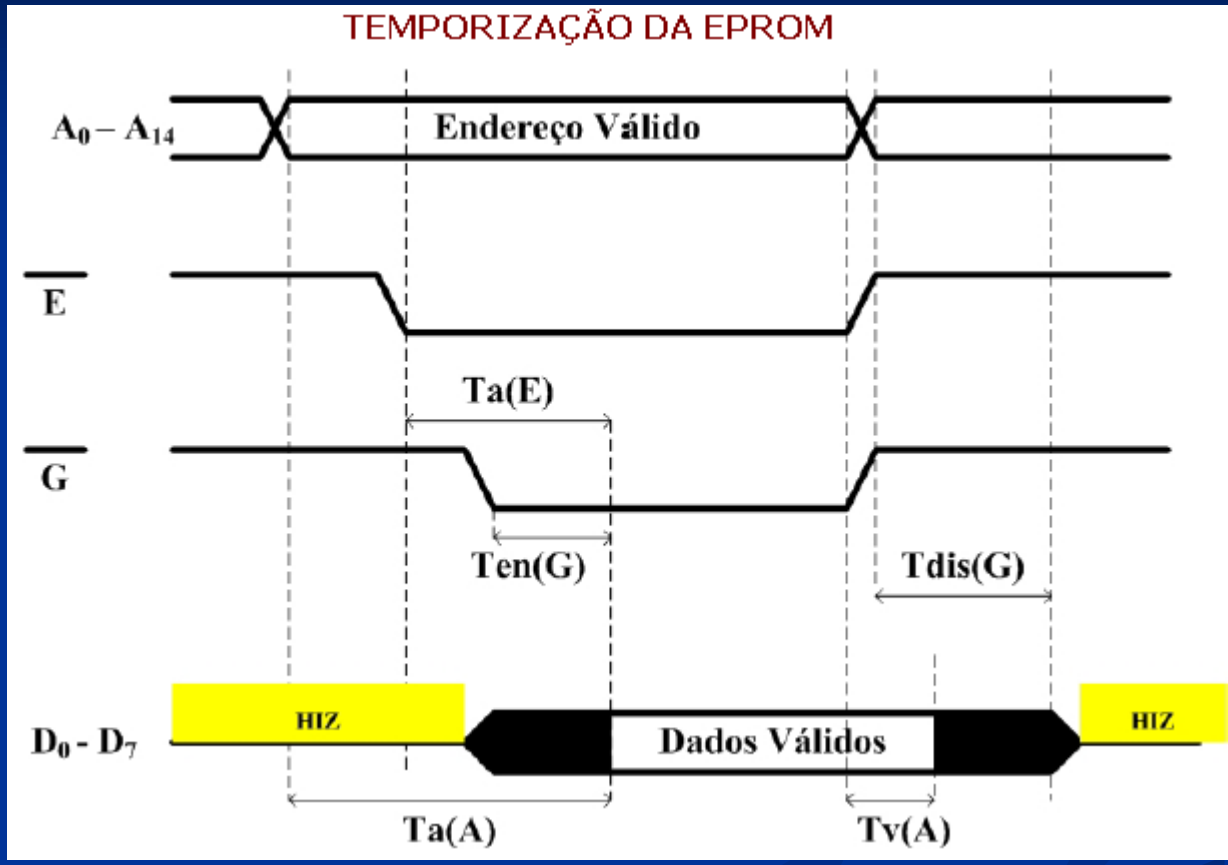
- as células de memória são organizadas em uma espécie de matriz, ou seja, são orientadas em um esquema que lembra linhas(*wordline*) e colunas(*bitline*).
- controlador de memória: acessa a memória gerando primeiro o valor **RAS** (*Row Address Strobe*) (nºda linha de qual o endereço faz parte), e depois do valor **CAS** (*Column Address Strobe*) da coluna .



http://www.hardware.com.br/tutoriais/memoria-ram/

Tempos de Chaveamento das Memórias Semicondutoras

Ta(A): Tempo de acesso após endereço válido.
Ta(E): Tempo de acesso após habilitação do chip.
Ten(G): Tempo de acesso após habilitação da saída (tristate).
Tv(A): Tempo em que os dados estão válidos após a mudança de endereço, de E ou de G.
Tdis(G): Tempo para desabilitar a saída após a mudança de endereço.



Ciclo de Leitura

Obs: Existe um atraso de propagação entre a aplicação das entradas (endereços e controle e seleção) de uma ROM e a aparição das saídas de dados durante a operação de leitura.

TEMPOS DE CHAVEAMENTO DAS MEMÓRIAS SEMICONDUTORAS (Cont.)

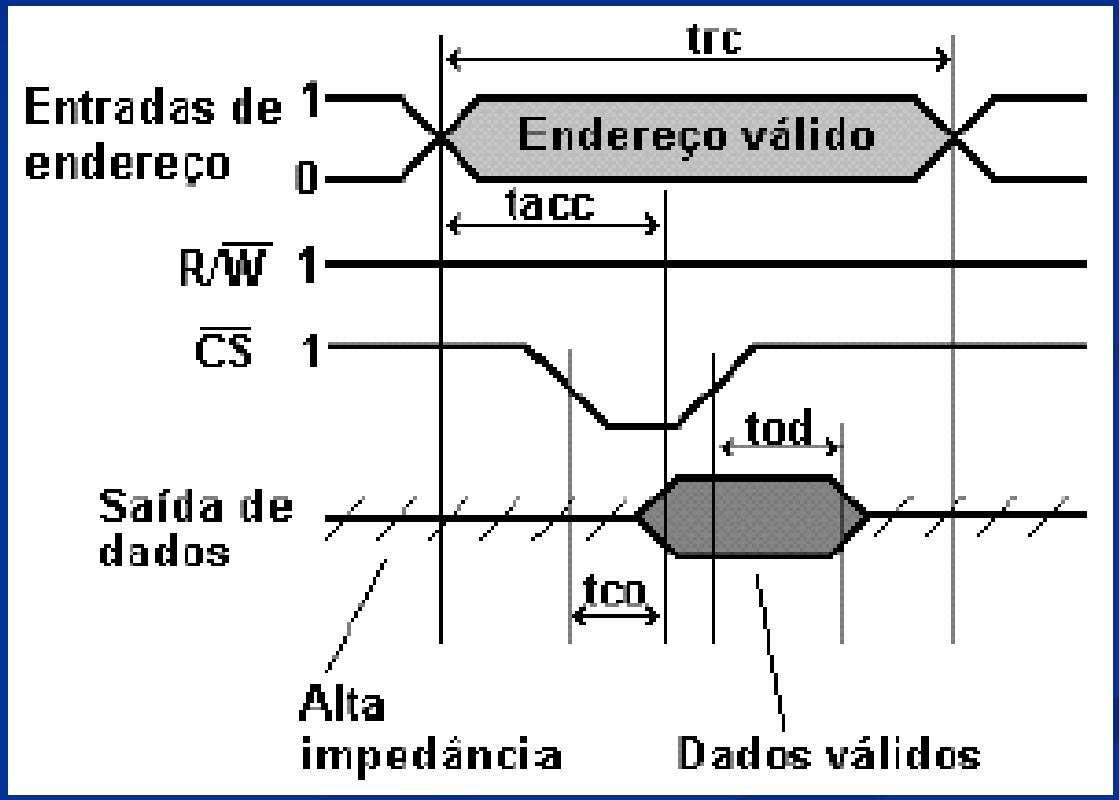
trc = intervalo de duração do ciclo de leitura;

tacc = tempo de acesso à RAM;

tco = tempo que a saída da RAM leva para sair de alta impedância e ter um dado válido;

tod = tempo decorrido entre a desabilitação da RAM e o instante que as saídas da RAM vão para alta impedância.

Temporização memória RAM

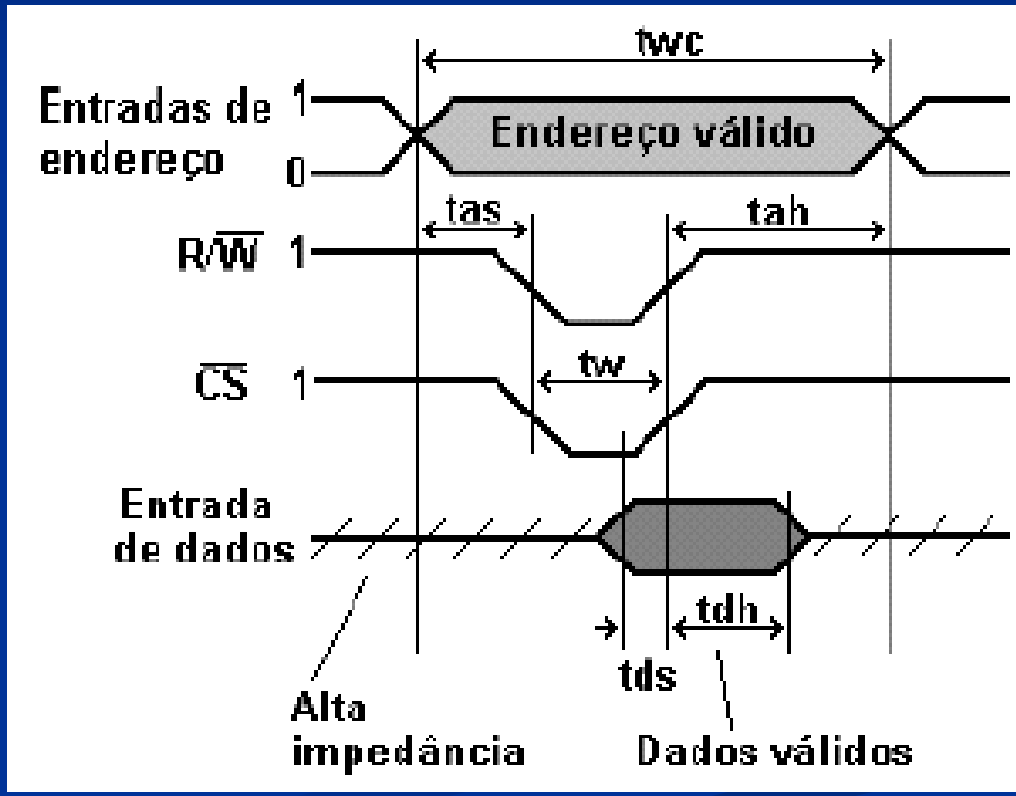


Ciclo de Leitura

TEMPOS DE CHAVEAMENTO DAS MEMÓRIAS SEMICONDUTORAS (cont.)

- **twc** = intervalo de duração do ciclo de escrita;
- **tas** = tempo para estabilização do duto de endereços, antes de habilitar a RAM;
- **tah** = intervalo necessário para que o duto de endereços permaneça estável;
- **tw** = tempo de escrita, onde /CS e R/W ficam em "0";
- **tds** = tempo em que os dados devem ser mantidos na entrada, antes da desabilitação de CS e W / R ;
- **tdh** = tempo em que os dados devem ser mantidos na entrada depois da desabilitação de CS e R /W .

Temporização memória RAM



Ciclo de Escrita (ou Gravação)

Expansão de Memórias Semicondutoras

Expansão de Memórias

a) Aumentar o número de bits da palavra:

* Exemplo:

+ Organização desejada: **2K x 8** (EPROM ou RAM)
+ Memória disponível: **2K x 4**

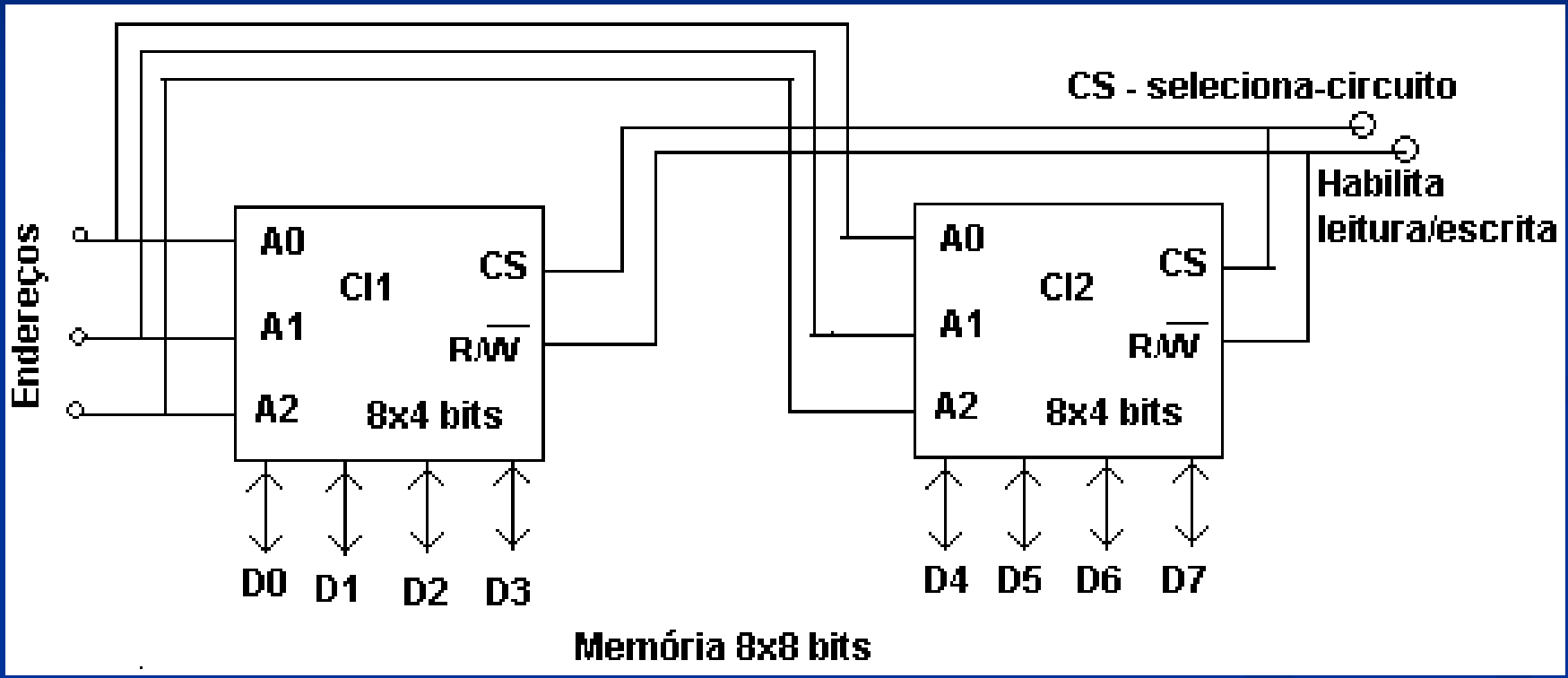
b) Aumentar o número de palavras (endereços):

* Exemplo:

+ Organização desejada: **4K x 8** (EPROM ou RAM)
+ Memória disponível: **2K x 8**

Aumentar o Tamanho (nº de bits) da Palavra

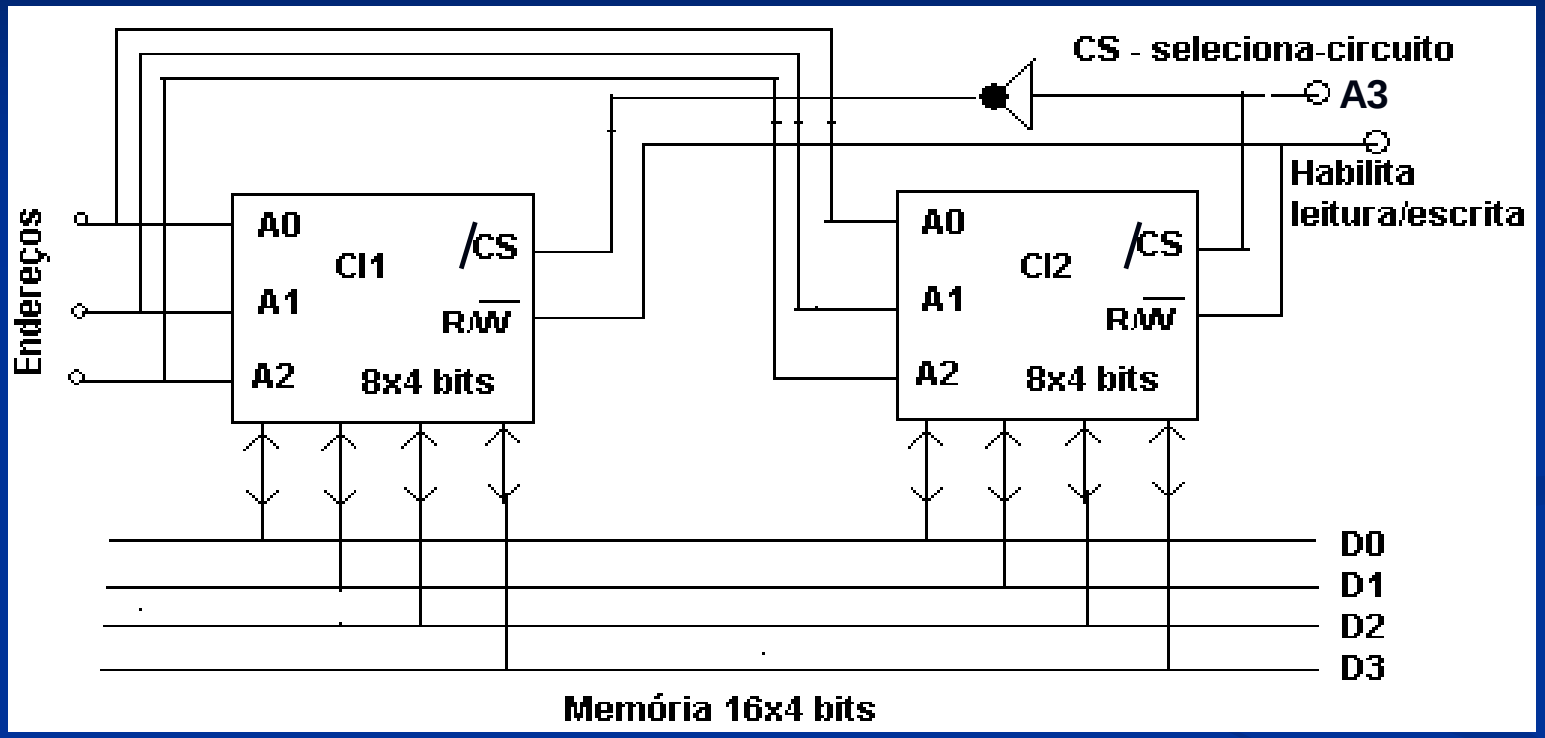
Dois CI's com 3 linhas de endereços → $2^3 = 8$ endereços. Cada endereço aponta para uma palavras de 4 bits (8x4), ligados de modo a formar uma memória de 8 palavras de 8 bits (8 x8)



- O duto de **endereços** e os pinos de controle dos CIs são interligados;
- O duto de **dados** fica dividido entre os CIs , de forma que cada CI contribui com uma parcela do dado
  - 4 MSB no CI2 - pino D7
  - 4 LSB no CI1 - pino D0

Aumentar o Nº de Palavras(células)

Dois CI's com 8 palavras de 4 bits cada (8x4), ligados de modo a formar uma memória de 16 palavras(células) de 4 bits (16x4)

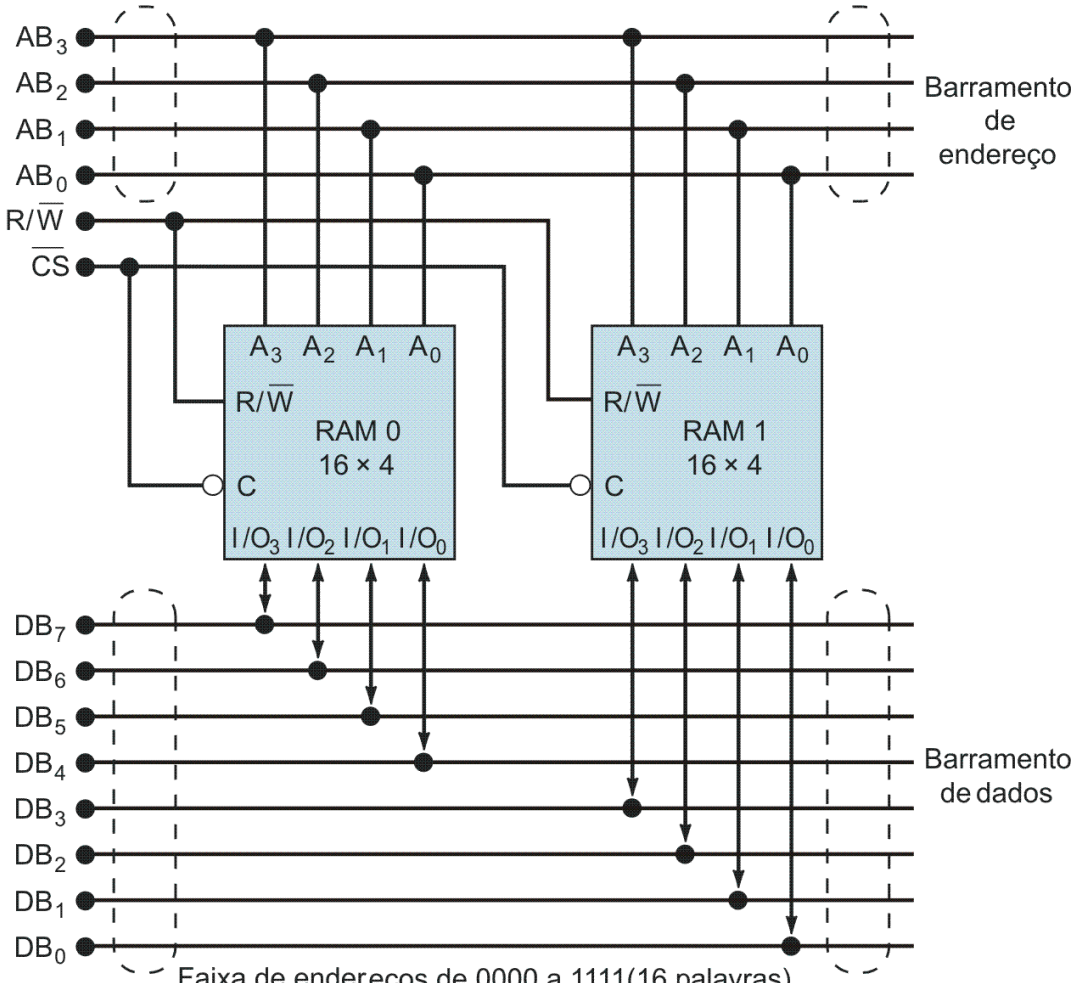


- O duto de **dados, endereços** e o pino de R/W dos CIs são interligados;
- O pino de controle (/CS) dos CIs **não** recebem o mesmo sinal; a seleção de CI1 e CI2 é feita através da linha de endereço A3, como segue:
  - Pino de endereço **A3** = 1 - Seleciona o CI1 (8 end. mais signif.)
  - Pino de endereço **A3** = 0 - Seleciona o CI2 (8 end. menos signif.)

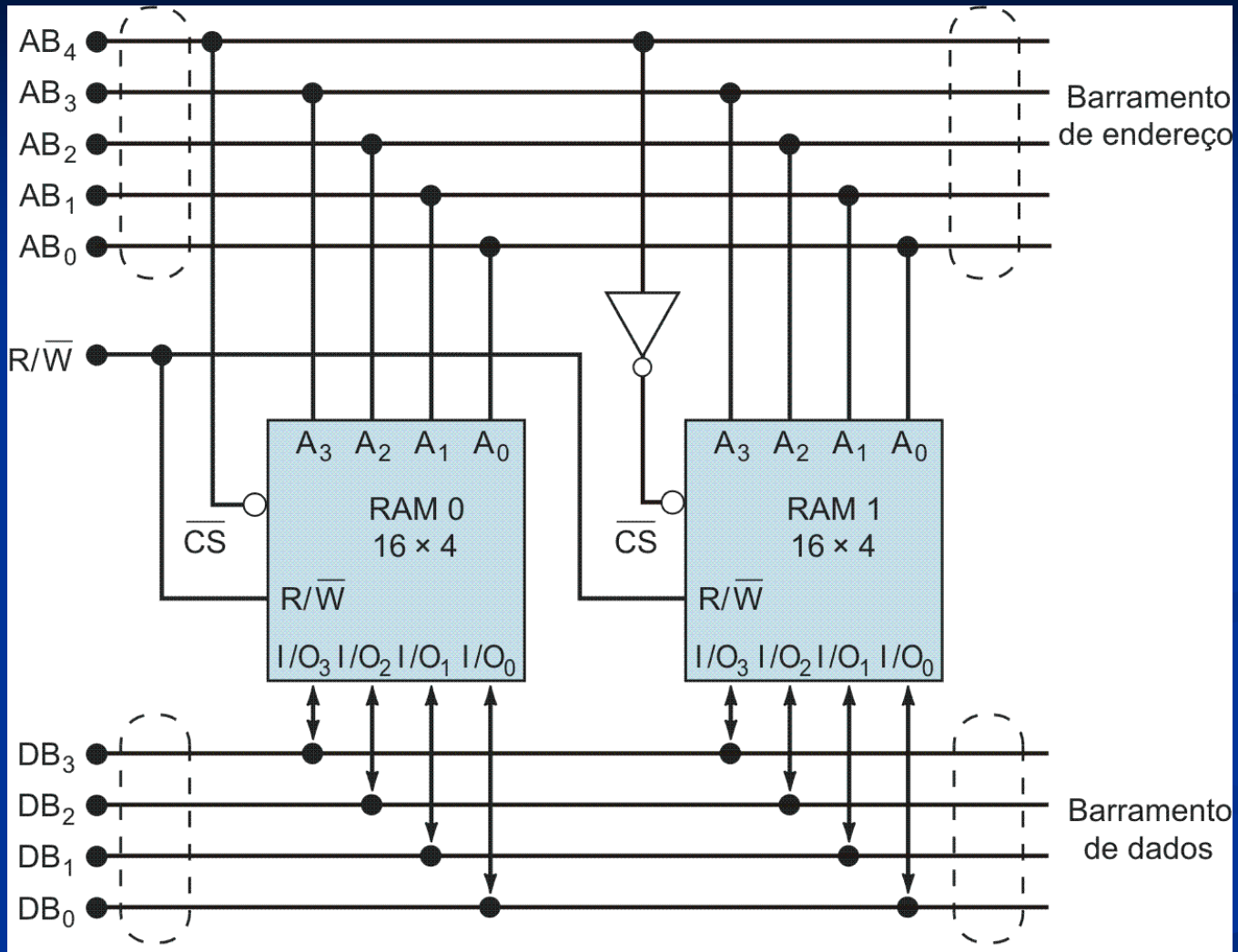
Aumentar o Nº de Palavras

- O duto de **dados, endereços** e o pino de R/W dos CIs são interligados;
- O pino de controle (/CS) dos CIs **não** recebem o mesmo sinal; ;a seleção de CI1 e CI2 é feita através da linha de endereço A3, como segue:
  - Pino de endereço **A3** = 1 - Seleciona o CI1 (8 end. mais signif.)
  - Pino de endereço **A3** = 0 - Seleciona o CI2 (8 end. menos signif.)

Duas RAMs de 16 X 4 em um módulo de 16 X 8



Duas RAMs de 16 X 4 em um módulo de 32 X 4



Faixas de endereço:	00000 a 01111 – RAM 0 10000 a 11111 –RAM 1
Total	00000 a 11111 – (32 palavras)

Tabela 2.1 Grandezas Usadas para Abreviar Valores em Computação

Nome da unidade	Valor em potência de 2	Valor em unidades
1K (1 quilo)	2^{10}	1024
1M (1 mega)	$1024K = 2^{20}$	1.048.576
1G (1 giga)	$1024M = 2^{30}$	1.073.741.824
1T (1 tera)	2^{40}	1.099.511.627.776
1P (1 peta)	2^{50}	1.125.899.906.843.624
1Ex (1 exa)	2^{60}	1.152.921.504.607.870.976
1Z (1 zeta)	2^{70}	1.180.591.620.718.458.879.424
1Y (1 yotta)	2^{80}	1.208.925.819.615.701.892.530.176

Lógica de Seleção de Memória

Mapeamento

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

1. Seleção de memórias e dispositivos de I/O

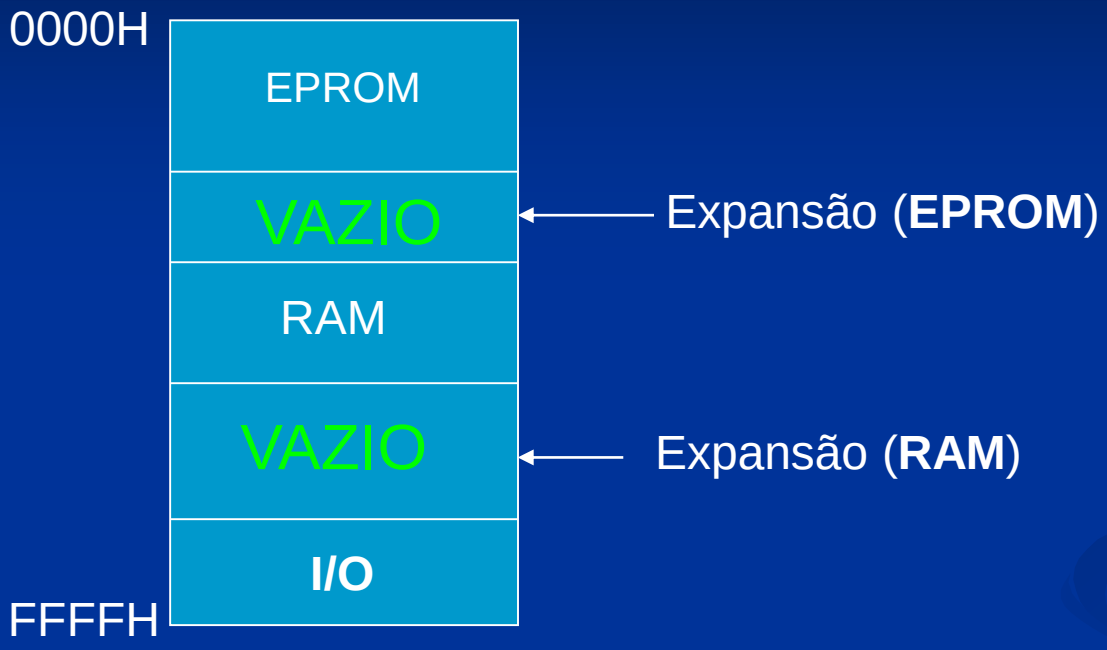
- Um microprocessador que tem duto de endereços de 16 bits e duto de dados de 8 bits, consegue endereçar $2^{16} = 65536$ (ou 64K) bytes

- As 64K posições que o microprocessador consegue endereçar podem ser representadas graficamente por um retângulo dividido em 64K posições, que é denominado **espaço de endereços do microprocessador**

Dentro do **espaço de endereços** de 64K bytes que o microprocessador consegue endereçar, são mapeadas as memórias e os dispositivos de I/O

A "Lógica de Seleção", construída pelo projetista, define as faixas de endereços do µP que irão selecionar cada uma das memórias e dispositivos

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O



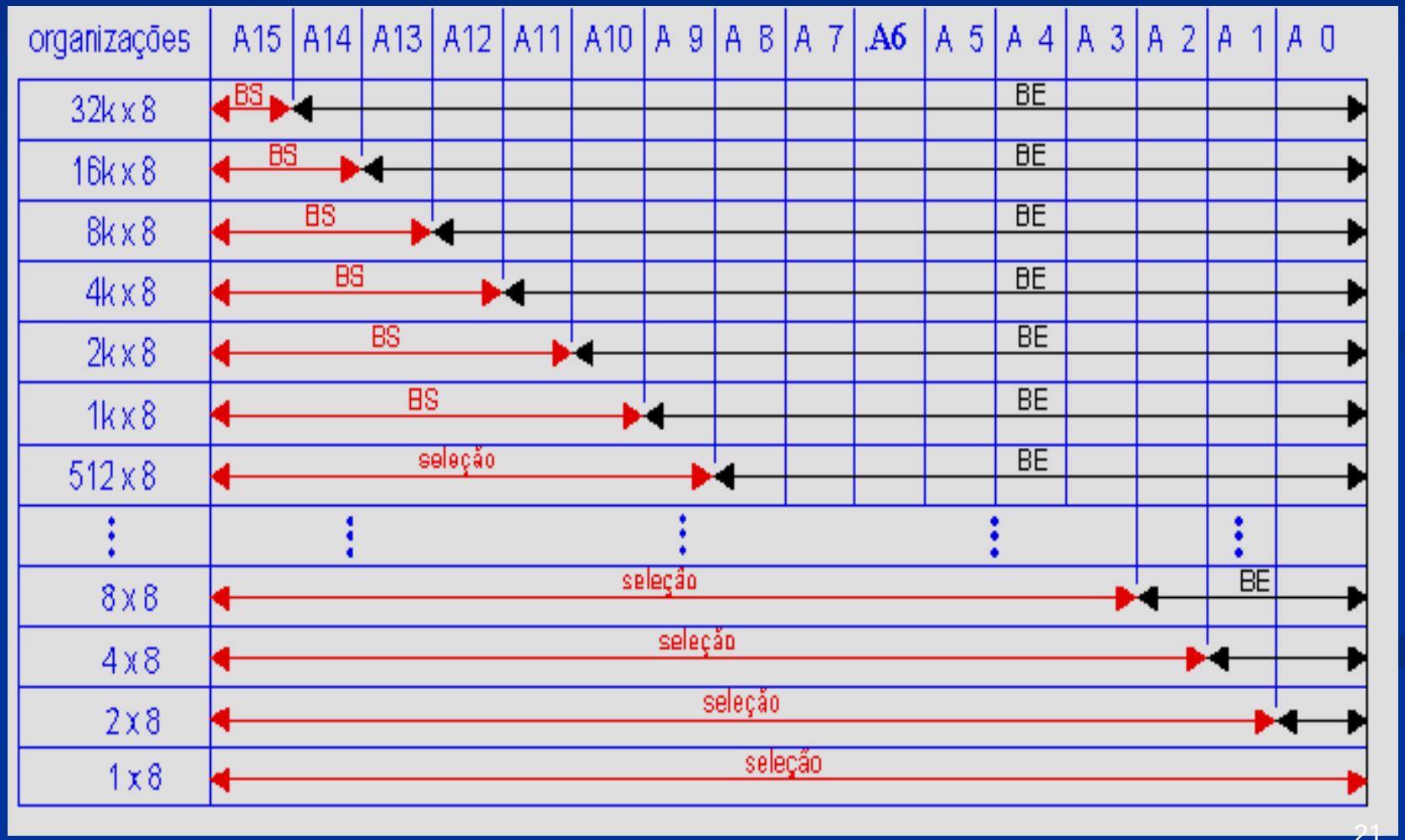
O endereço de **16 bits**, gerado pelo **uP** pode ser visto como sendo constituído por duas partes (tabela 1) :

BE: Bits de endereçamento do chip

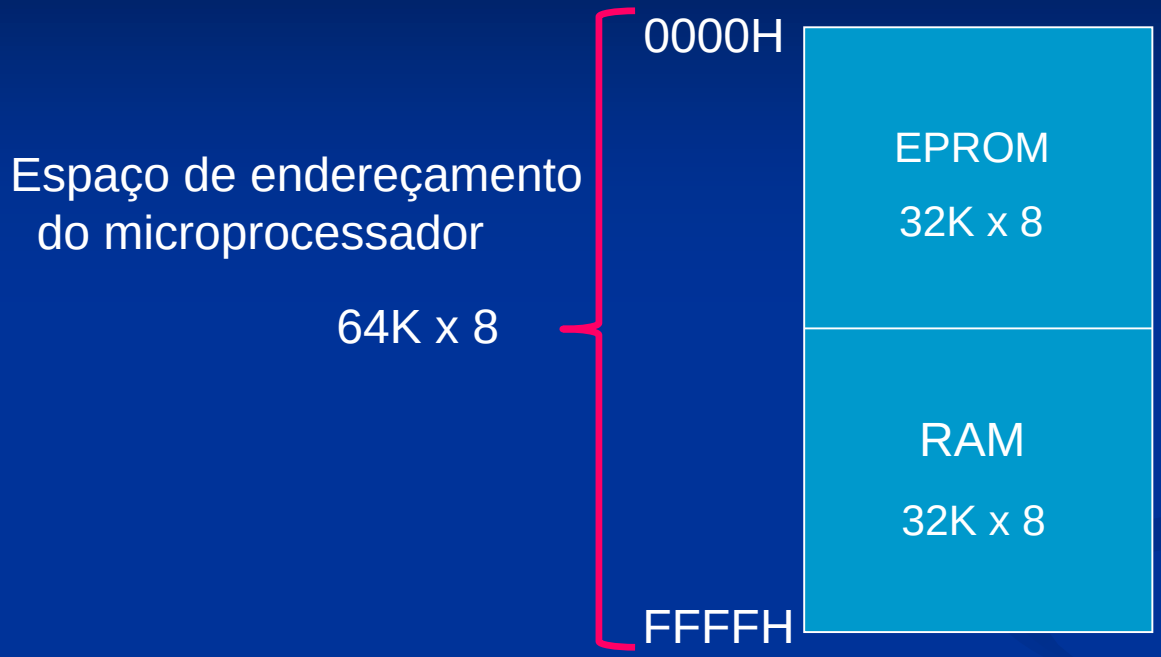
BS: Bits de seleção

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

Tabela 1 – Sinais de seleção BS para diferentes organizações



LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

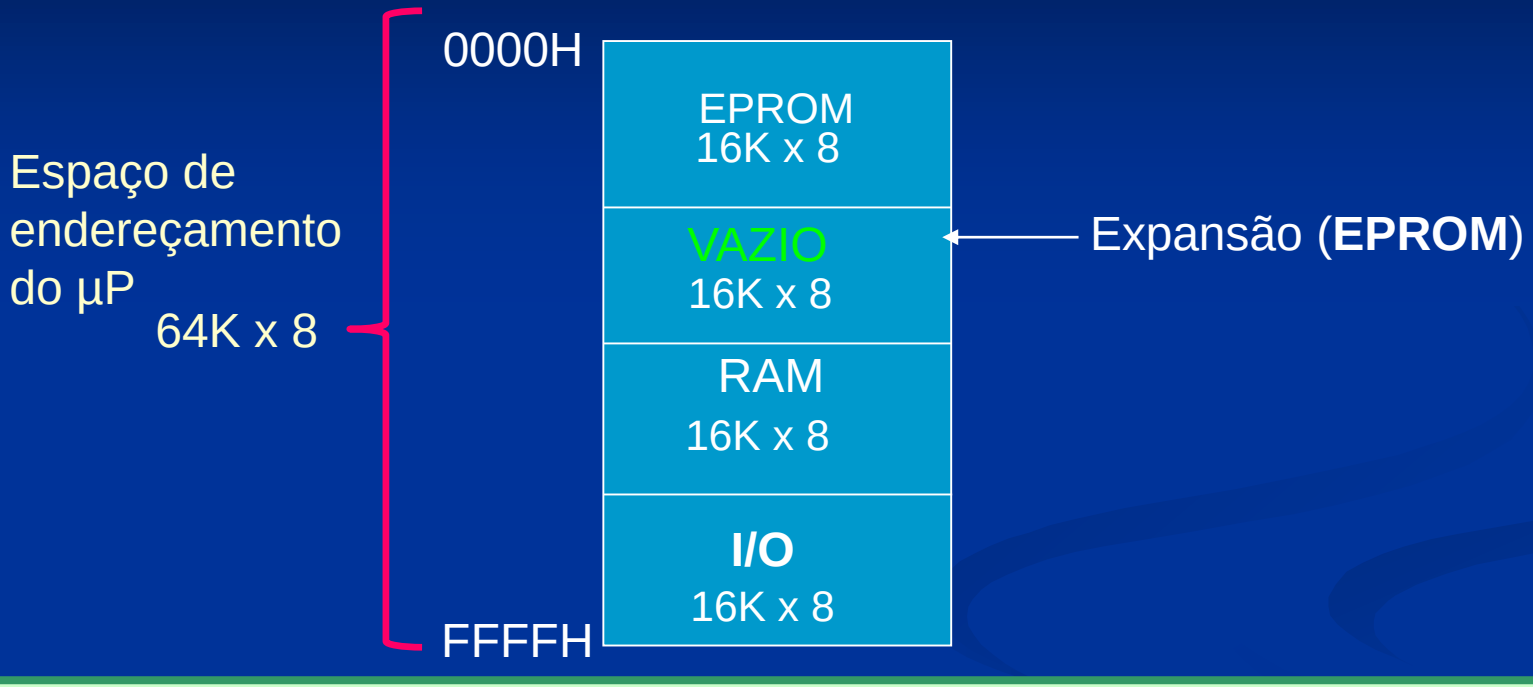


O endereço de **16 bits**, gerado pelo **uP para endereçar** cada memória de 32Kx8, pode ser visto como sendo constituído por duas partes (tabela 1)

BE: Bits de endereçamento dos chips de memória de 16Kx8 de A_{14} a A_0

BS: Bit de seleção dos chips de memória de 32Kx8 é A_{15}

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O



O endereço de **16 bits**, gerado pelo **µP para endereçar** cada memória de e dispositivos de I/O de 16Kx8, pode ser visto como sendo constituído por duas partes (tabela 1) :

BE: Bits de endereçamento dos chip de memória e dispositivos de I/O de 16Kx8 de A_{13} a A_0

BS: Bits de seleção chip de memória e dispositivos de I/O de 16Kx8 são A_{15} e A_{14}

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

2. Número de posições de memória (em hexadecimal) para cada organização.

Fazendo-se os bits de seleção (**BS**) iguais a **zero**, e bits de endereçamento (**BE**), iguais a “1”, na tabela **1**, é possível determinar o **(número de posições – 1)** ocupadas por cada organização:

32k x 8 : 7FFFH	2k x 8 : 07FFH	8 x 8 : 0007H
16k x 8 : 3FFFH	1k x 8 : 03FFH	4 x 8: 0003H
8k x 8 : 1FFFH	512 x 8 : 01FFH	2 x 8: 0001H
4k x 8 : 0FFFH	:	1 x 8 : 0000H

Esse valor é somado ao endereço inicial, no espaço de endereçamento, para se obter o endereço final da memória

		Linhas de Endereço do Microprocessador		
Blocos de Memórias	nº de Linhas de Endereço	Seleção da Memória	bits de Endereço	Tamanho do Bloco
32K X8	15	A15	A0 até A14	7FFFH
16K X8	14	A14 até A15	A0 até A13	3FFFH
8K X8	13	A13 até A15	A0 até A12	1FFFH
4K X8	12	A12 até A15	A0 até A11	0FFFH
2K X8	11	A11 até A15	A0 até A10	07FFH
1K X8	10	A10 até A15	A0 até A9	03FFH

Relação entre os blocos de memórias e as linhas de endereço de um microprocessador de 16 linhas (bits) de endereço

Blocos de Memória	n.o de linhas de endereço	Bits de seleção	Bits de endereço	Tamanho do bloco
512 X8	9	A9 até A15	A0 até A8	01FFH
256 X8	8	A8 até A15	A0 até A7	00FFH
128 X8	7	A7 até A15	A0 até A6	007FH
64 X8	6	A6 até A15	A0 até A5	003FH
32 X8	5	A5 até A15	A0 até A4	001FH
16 X8	4	A4 até A15	A0 até A3	000FH
8 X8	3	A3 até A15	A0 até A2	0007H
4 X8	2	A2 até A15	A0 até A1	0003H
2 X8	1	A1 até A15	A0	0001H
1 X8	0	A0 até A15	nenhum	0000H

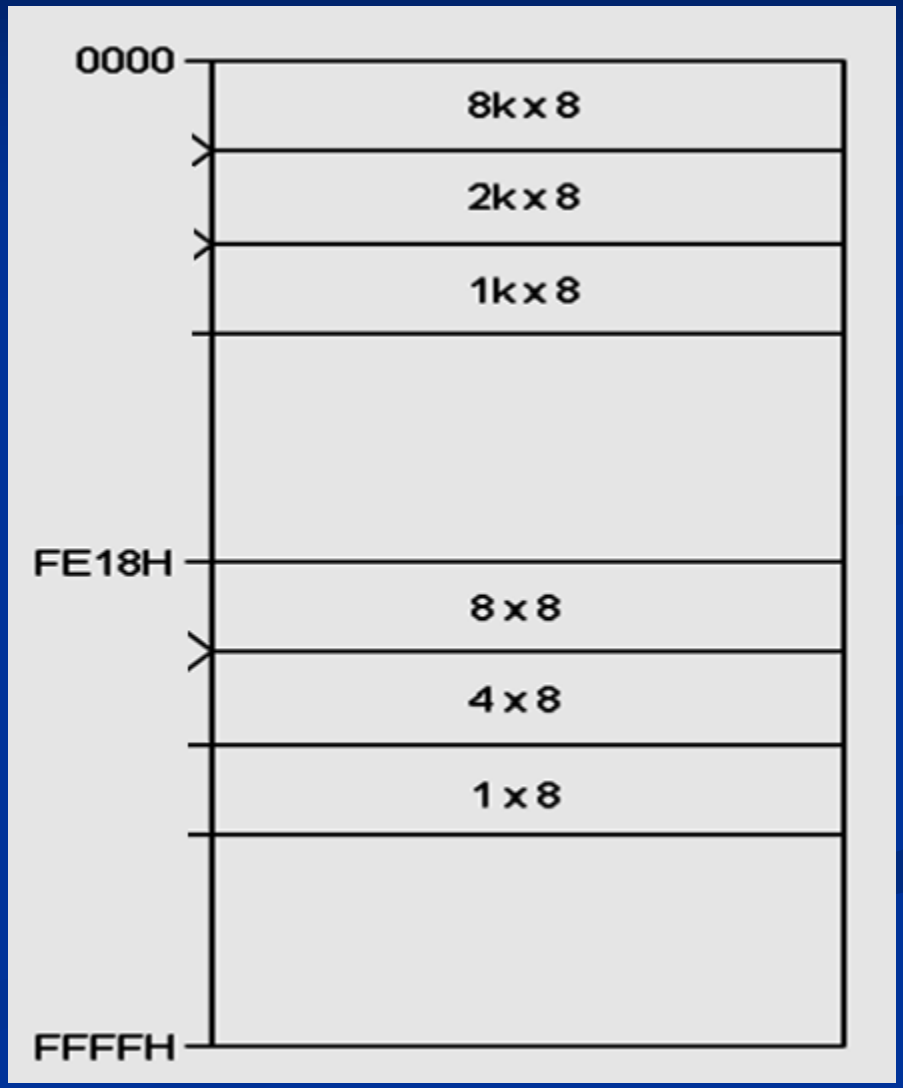
Relação entre os blocos de memórias e as linhas de endereço de um microprocessador de 16 linhas (bits) de endereço

Exemplo: considerando endereço inicial = 8000H

	end. Inicial	**end. Final**
32Kx8	8000H	FFFFH
8KX8	8000H	9FFFH
1KX8	8000H	83FFH
8X8	8000H	8007H
4X8	8000H	8003H
2X8	8000H	8001H
1X8	8000H	8000H

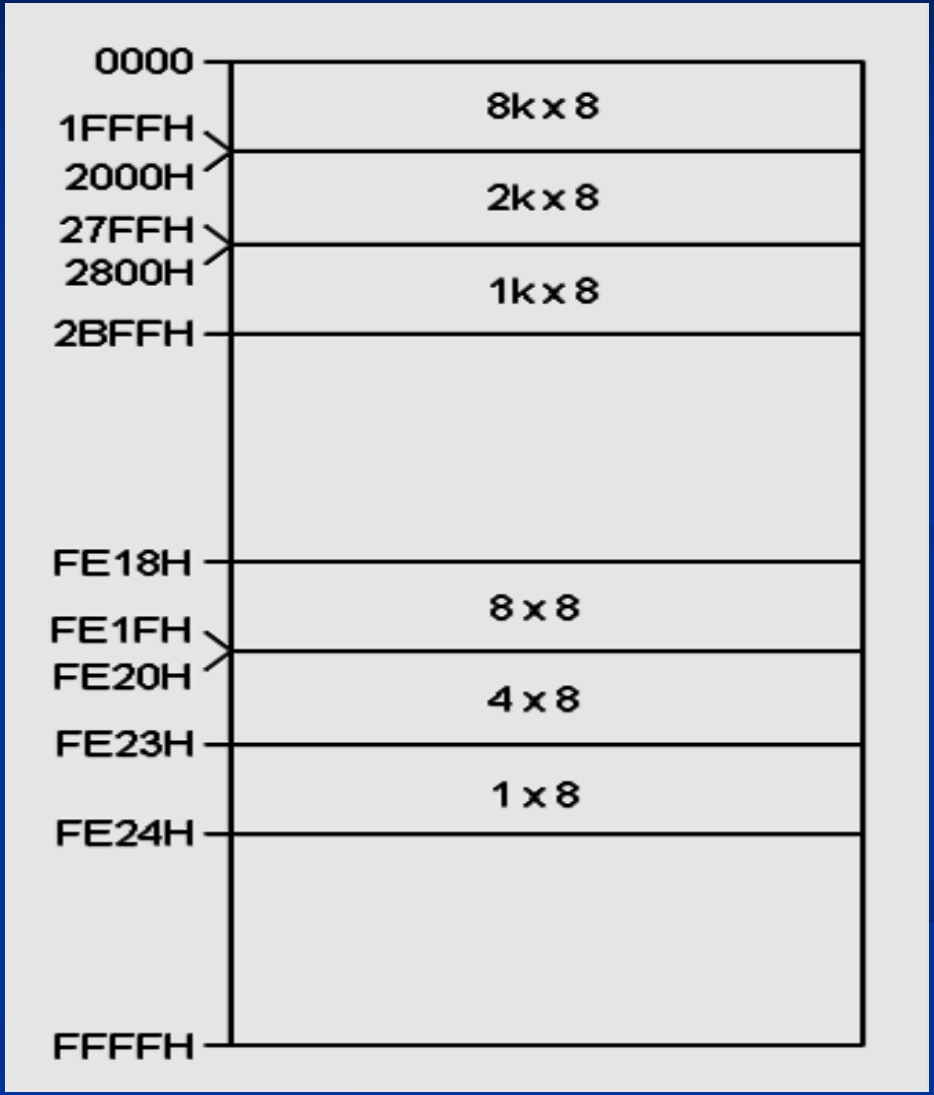
LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

Exercício: Encontrar o endereço inicial e final, para as organizações de memória e de I/O mapeadas na Figura ao lado



LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

Resposta: Endereço inicial e final para as organizações de memória e de I/O



LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

A lógica de seleção implementada com circuitos decodificadores garante a seleção de **uma única memória ou interface**, que se comunicará com o microprocessador.

• Cada decodificador fica "dentro" de um espaço de endereço, e divide esse espaço em blocos menores.

• O tamanho da divisão depende de qual é o **bit de seleção** menos significativo conectado na entrada do decodificador (ver tabela 1) .
Exemplo:

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

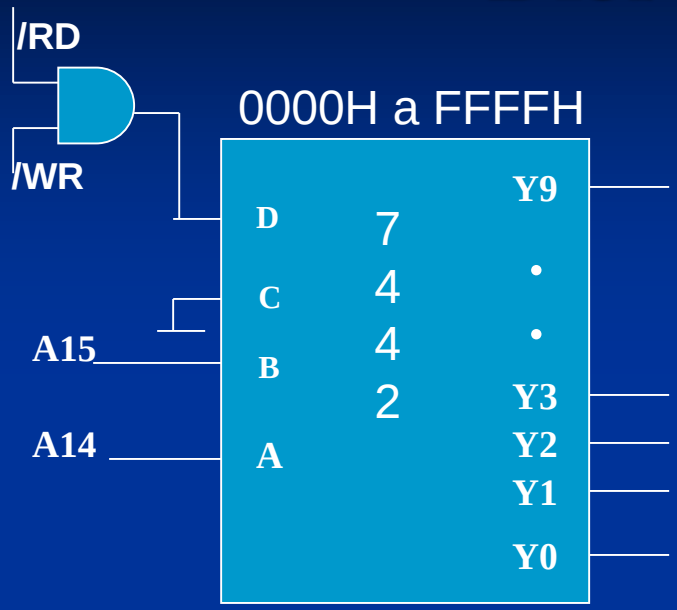
- Espaço de endereço: **0000 a FFFFH**

(o decodificador não é selecionado por nenhum outro decodificador)

- tamanho da divisão: **16 Kx8**, pois o bit menos significativo é o **A14**
- pode-se **conectar direto** nesse decodificador organizações que tem linhas de endereço de A0 - A13

Quais saídas do decodificador que podem ser ativadas ?
Qual a faixa de endereço associada a cada saida?

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O



- Espaço de endereço: **0000 a FFFFH**
(o decodificador não é selecionado por nenhum outro decodificador)
- tamanho da divisão: **16 Kx8**, pois o bit menos significativo é o **A14**
- pode-se **conectar direto** nesse decodificador organizações que tem linhas de endereço de A0 - A13

Saídas válidas para serem usadas como /CS:	**faixa de endereço**
Y0 : (A15, A14) = (0,0)	.... 0000H até 3FFFH
Y1 : (A15, A14) = (0,1)	.... 4000H até 7FFFH
Y2 : (A15, A14) = (1,0)	.... 8000H até BFFFH
Y3 : (A15, A14) = (1,1)	.... C000H até FFFFH

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

Cada saída **válida** do decodificador, que pode ser usada como saída de seleção **(/CS)**, tem a ela associada uma **faixa de endereço**, determinada pelos bits de endereço (seleção) conectados nas entradas deste decodificador.

Há duas maneiras de se determinar a faixa de endereço da saída:

a . Soma do bloco divisor ao endereço inicial de cada saída válida do decodificador

b. determina-se o endereço inicial e final associado à saída do decodificador como segue:

- **endereço inicial :** valor dos bits de seleção conectados no decodificador, que ativa a saída. Os demais bits em "**0**".
- **endereço final:** valor dos bits de seleção conectados no decodificador, que ativa a saída. Os demais bits em "**1**"

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

Utilizando o mesmo exemplo do decodificador anterior:

a . Soma do bloco divisor ao endereço inicial de cada saída
tamanho do bloco divisor = **3FFFH** (16 K x 8)
Y0 = 0000 a 3FFFH Y1 = 4000H a 7FFFH
Y2 = 8000H a BFFFH Y3 = C000H a FFFFH

b. Valor dos bits de seleção (A15 e A14)		**A15** **A0**
Saída Y0	endereço inicial: 0000H →	**00**00 0000 0000 0000
	endereço final : 3FFFH →	**00**11 1111 1111 1111
Saída Y1	endereço inicial: 4000H →	**01**00 0000 0000 0000
	endereço final: 7FFFH: →	**01**11 1111 1111 1111
Saída Y2	endereço inicial: 8000H →	**10**00 0000 0000 0000
	endereço final : BFFFH →	**10**11 1111 1111 1111
Saída Y3	endereço inicial: C000H →	**11**00 0000 0000 0000
	endereço final : DFFFH →	**11**11 1111 1111 1111

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

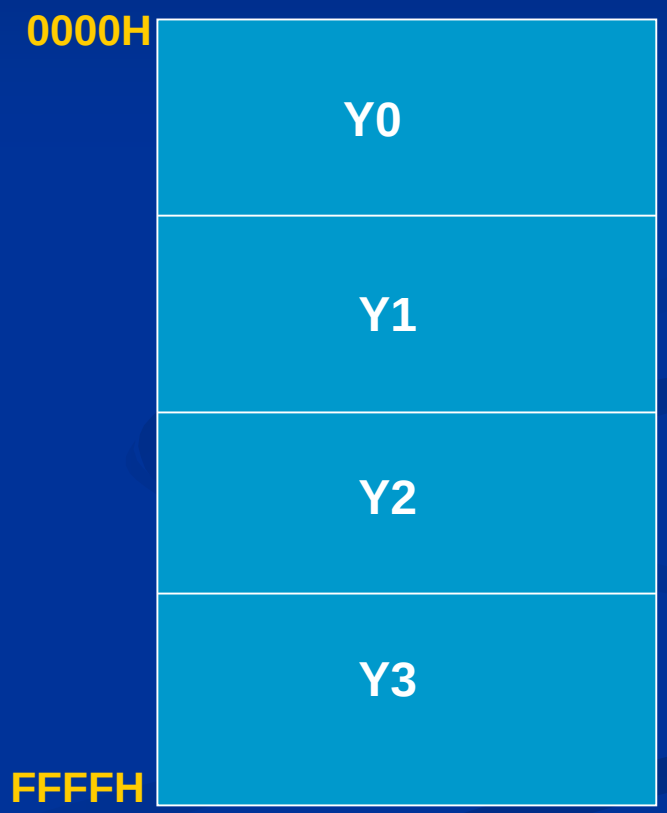
Dado o mapa de endereço ao lado:

a. como seria a ligação do decodificador(7442) para obter a divisão do espaço de endereços de 64Kbytes em 4 espaços iguais?

b. Qual o tamanho de cada bloco em hexadecimal?

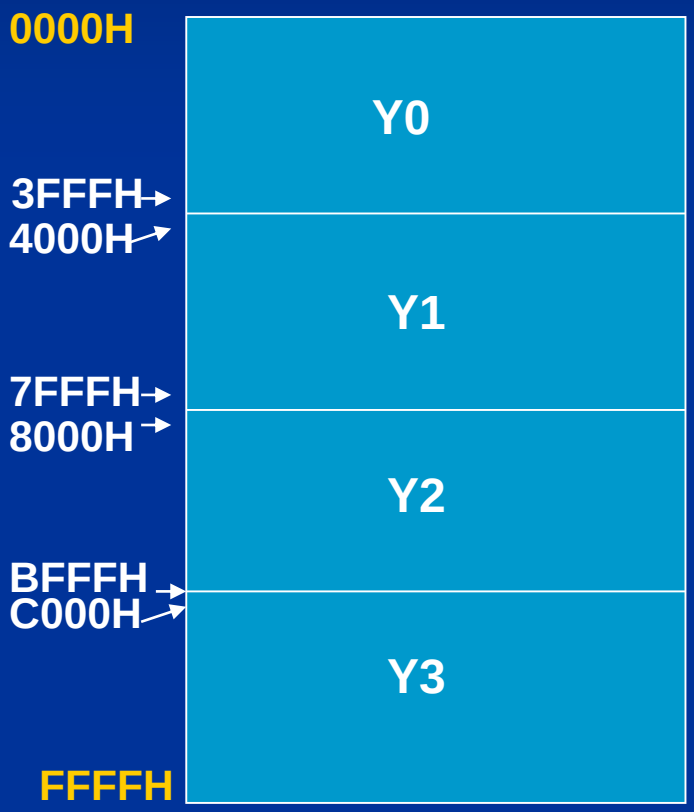
c. Qual o endereço inicial e final em (hexadecimal) de cada bloco?

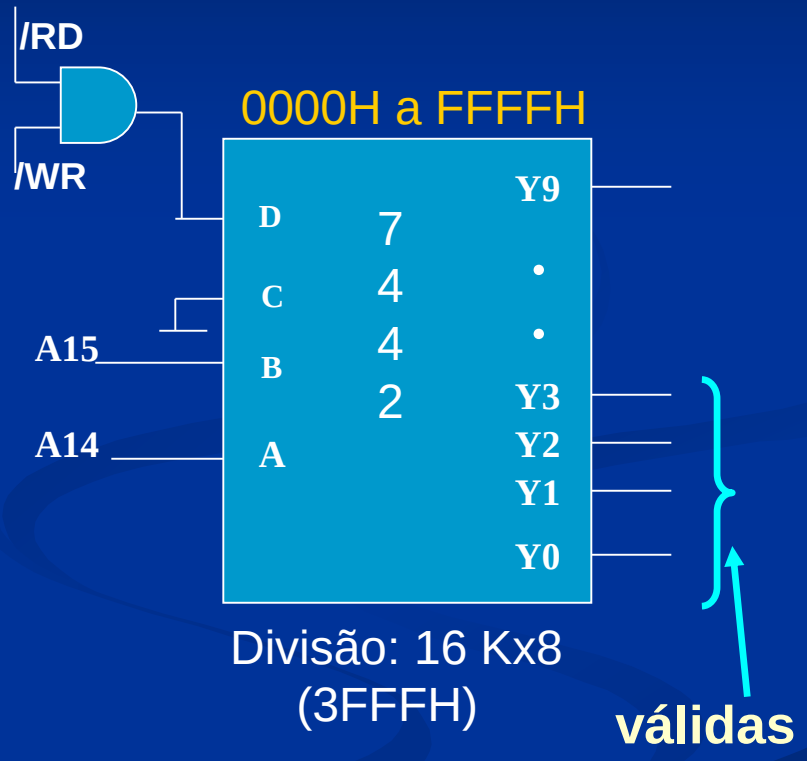
Mapa de endereços



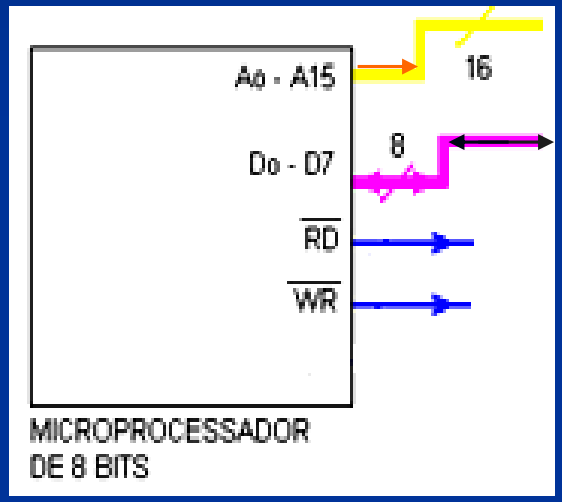
LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

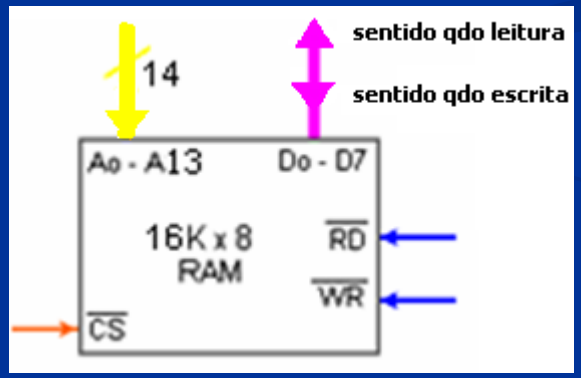
Mapa de endereços

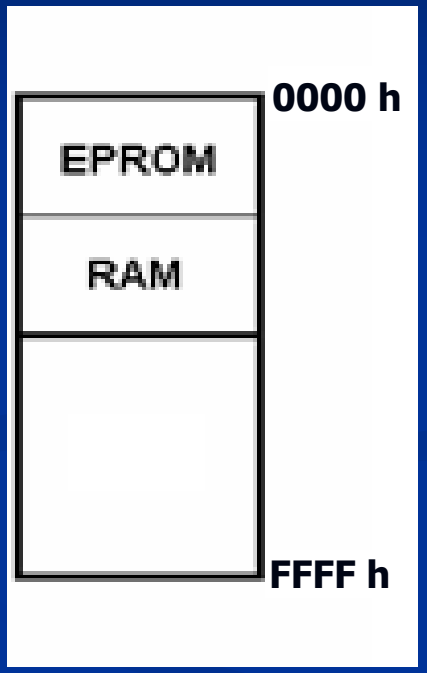




Como ligar o µP às Memórias?







Mapeamento

Lógica de Seleção

1.o exemplo:

Lógica de Seleção do µP – Linhas de Endereços																	Memória		
Tipo	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Início (H)		Fim (H)
ROM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	16k	
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1			3FFF
RAM																	4000	16k	
																			7FFF
I/0																	8000	32k	
																			FFFF

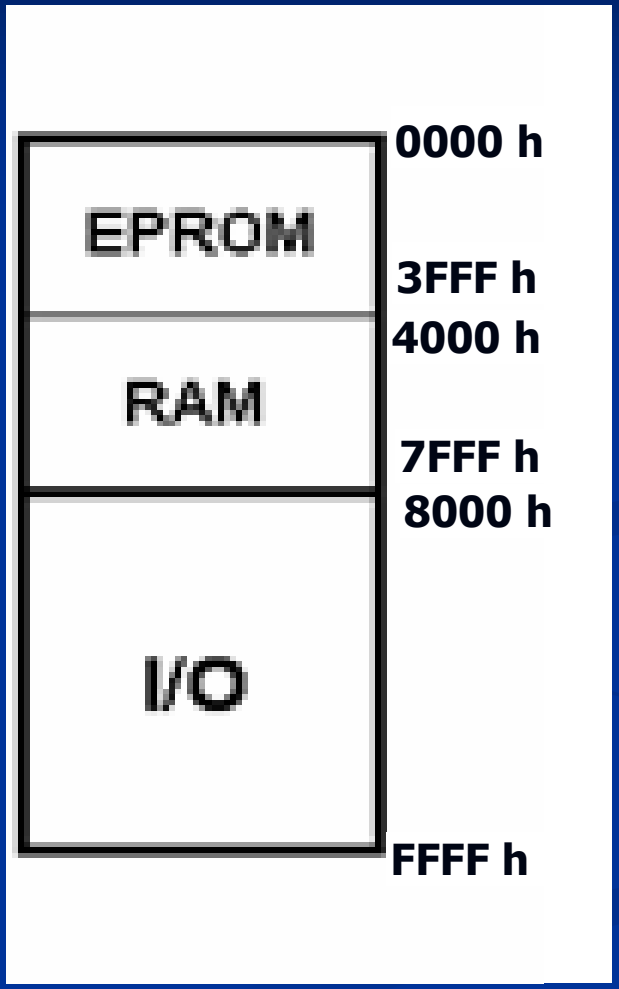
Lógica de Seleção

1.o exemplo:

Lógica de Seleção do µP – Linhas de Endereços																	Memória		
Tipo	**A15**	**A14**	**A13**	**A12**	**A11**	**A10**	**A9**	**A8**	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**	**Início (H)**		**Fim (H)**
ROM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0000**	16k	
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1			**3FFF**
RAM	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**4000**	16k	
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			**7FFF**
I/0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**8000**	32k	
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			**FFFF**

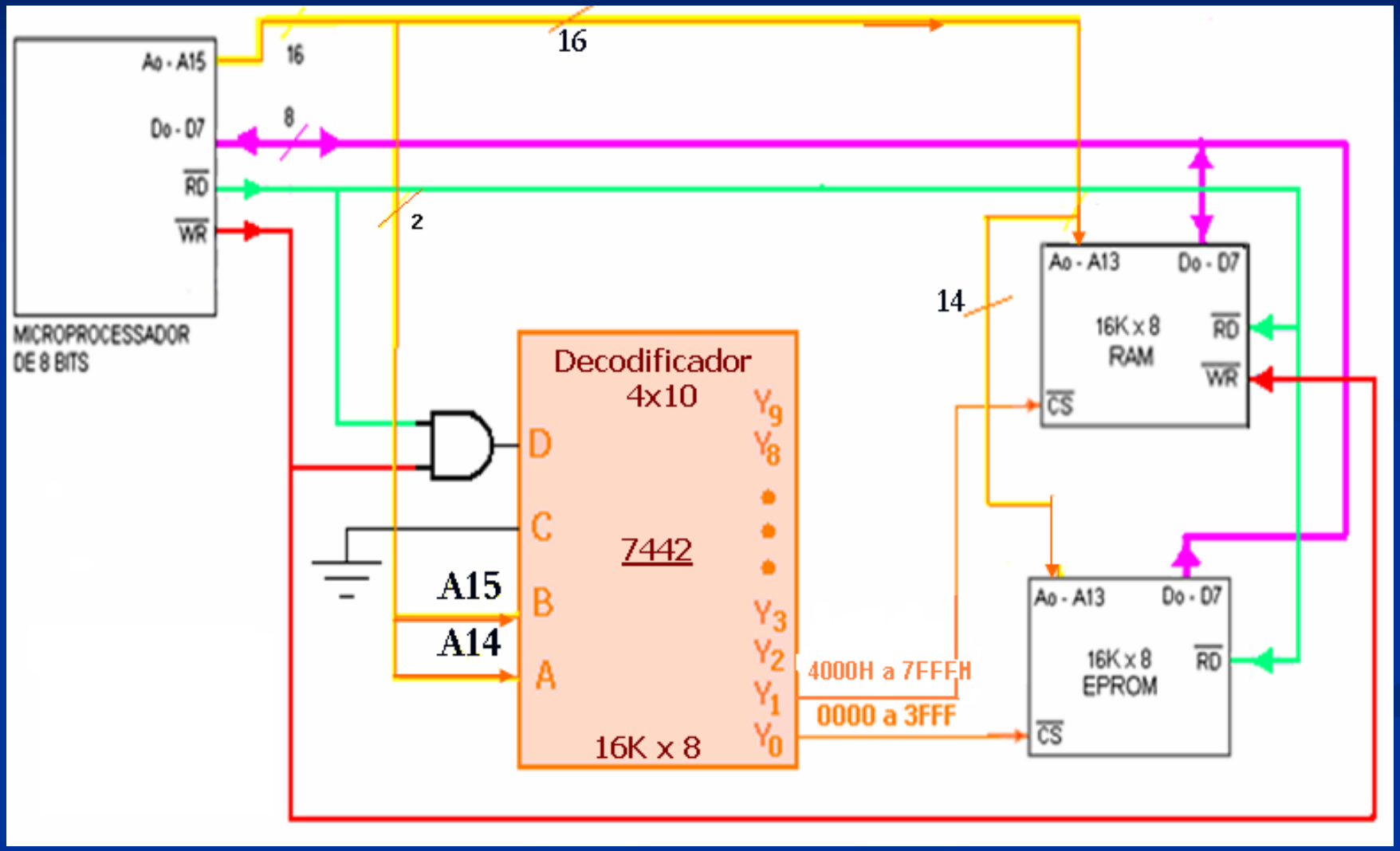
Mapeamento

1.o exemplo:



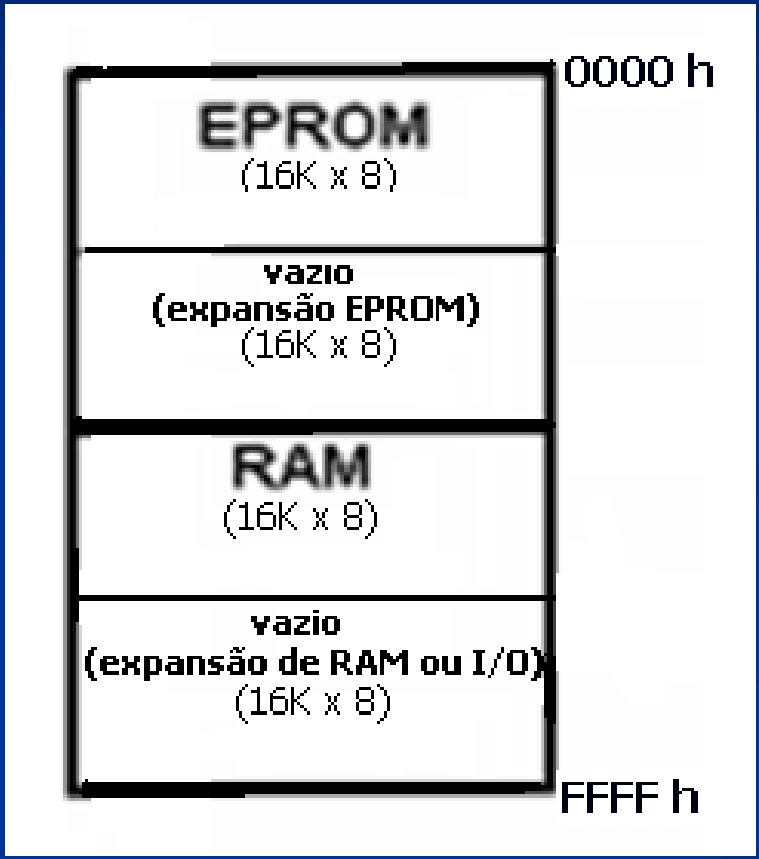
Como ligar o μP às Memórias

1.o exemplo:



Mapeamento

2.o exemplo:



Lógica de Seleção

2.o exemplo:

Lógica de Seleção do µP – Linhas de Endereços																	Memória		
Tipo	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Início (H)		Fim (H)
ROM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	16k	
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1			3FFF
vazio expansão de ROM																	4000	16k	
																			7FFF
RAM																	8000	16k	
																			BFFF
vazio expansão de RAM ou I/O																	C000	16k	
																			FFFF

Lógica de Seleção

2.o exemplo:

Lógica de Seleção do µP – Linhas de Endereços																	Memória		
Tipo	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Início (H)		Fim (H)
ROM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	16k	
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1			3FFF
vazio expansão de ROM	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000	16k	
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			7FFF
RAM																	8000	16k	
																			BFFF
vazio expansão de RAM ou I/O																	C000	16k	
																			FFFF

Lógica de Seleção

2.o exemplo:

Lógica de Seleção do µP – Linhas de Endereços																	Memória		
Tipo	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Início (H)		Fim (H)
ROM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	16k	
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1			3FFF
vazio expansão de ROM	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000	16k	
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			7FFF
RAM	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000	16k	
	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1			BFFF
vazio expansão de RAM ou I/O																	C000	16k	
																			FFFF

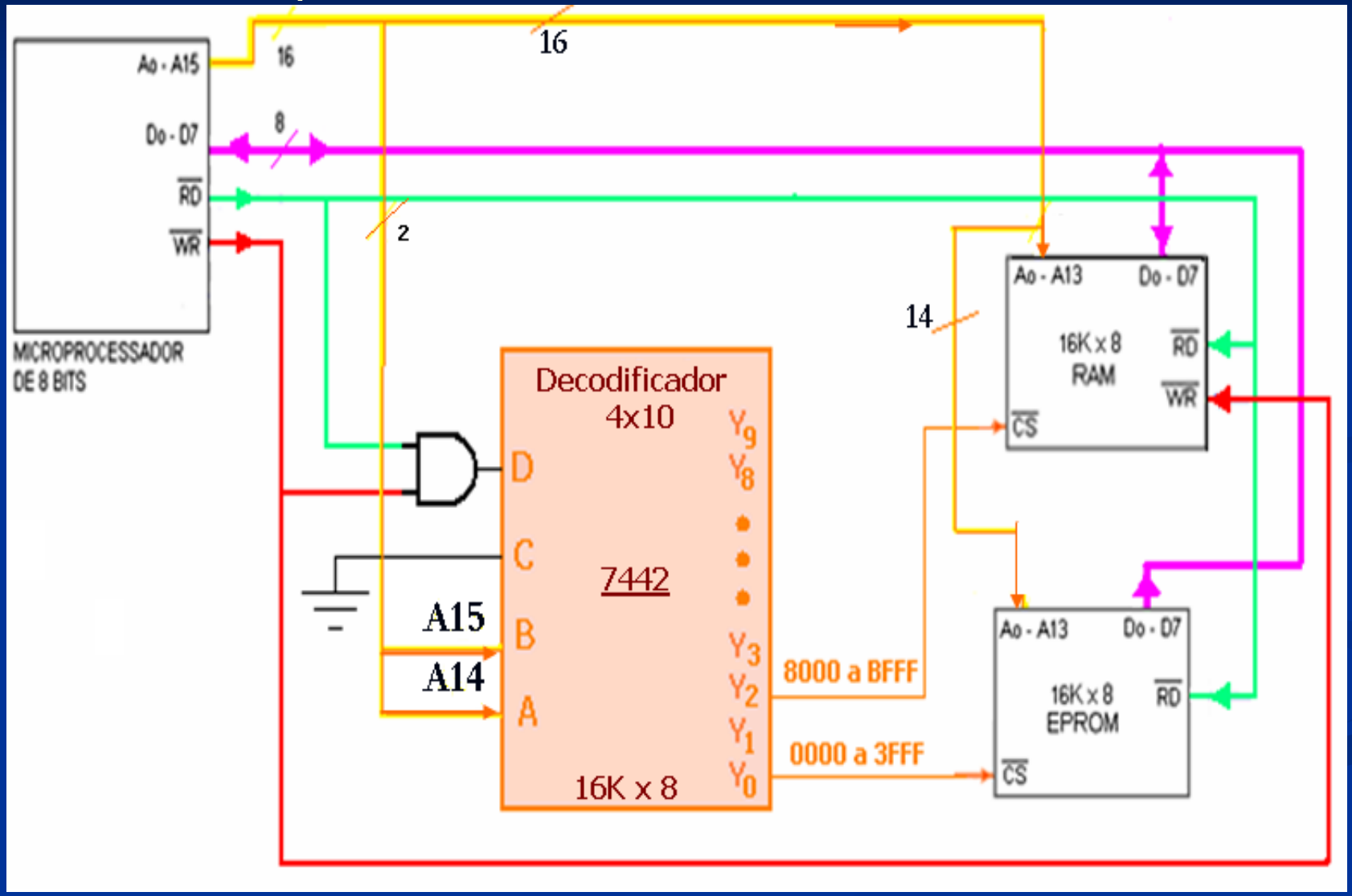
Lógica de Seleção

2.o exemplo:

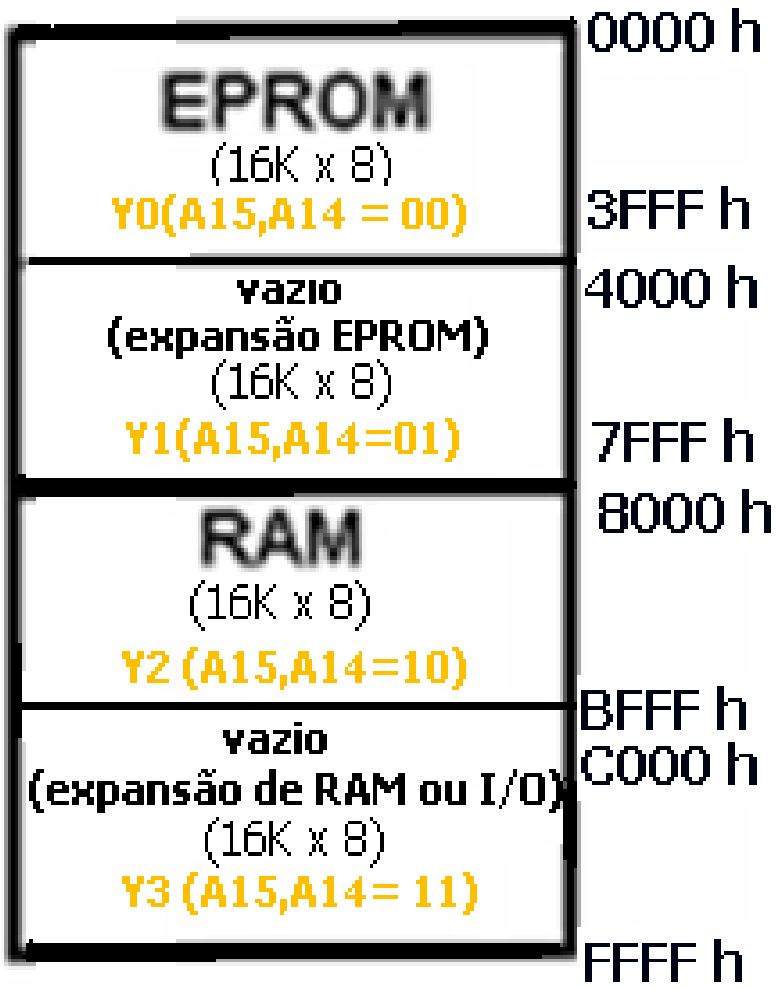
Lógica de Seleção do µP – Linhas de Endereços																	Memória		
Tipo	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Início (H)		Fim (H)
ROM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	16k	
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1			3FFF
vazio expansão de ROM	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000	16k	
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			7FFF
RAM	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000	16k	
	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1			BFFF
vazio expansão de RAM ou I/O	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C000	16k	
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			FFFF

Como ligar o µP às Memórias?

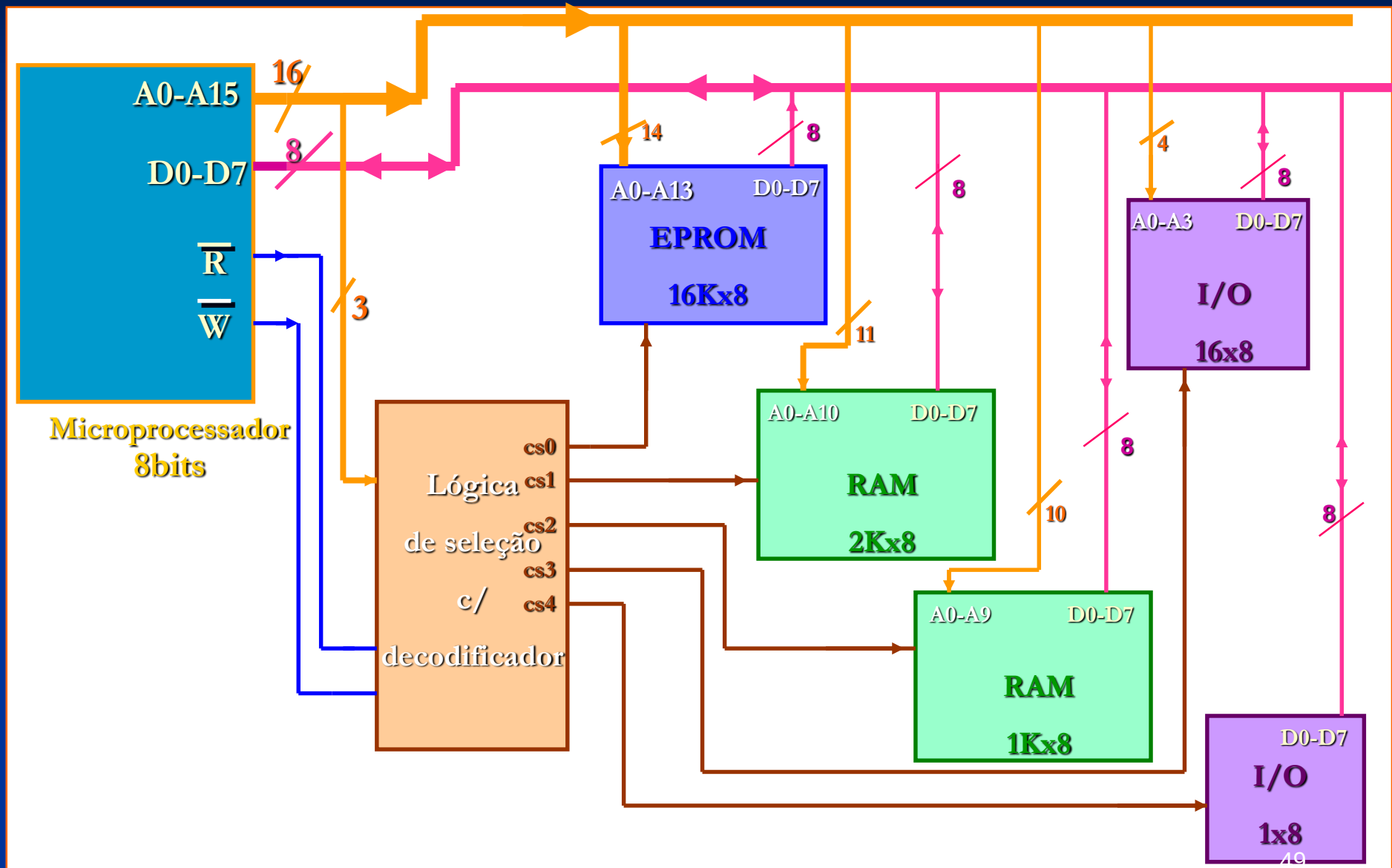
2.o exemplo:



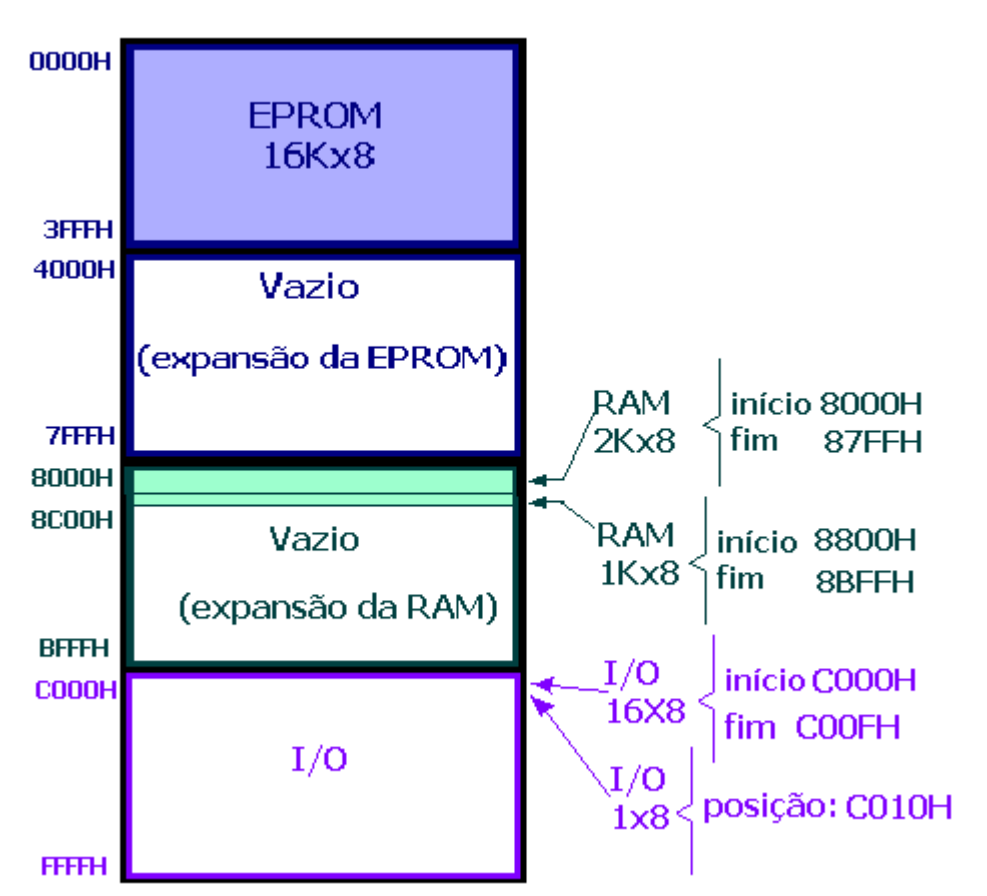
Mapeamento 2º exemplo



Como ligar o µP às Memórias? Exemplo 3

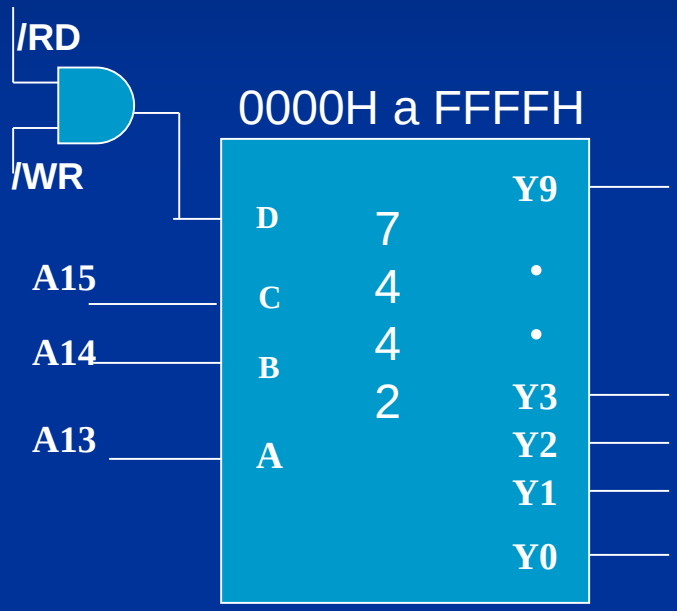


Mapeamento do Exemplo2



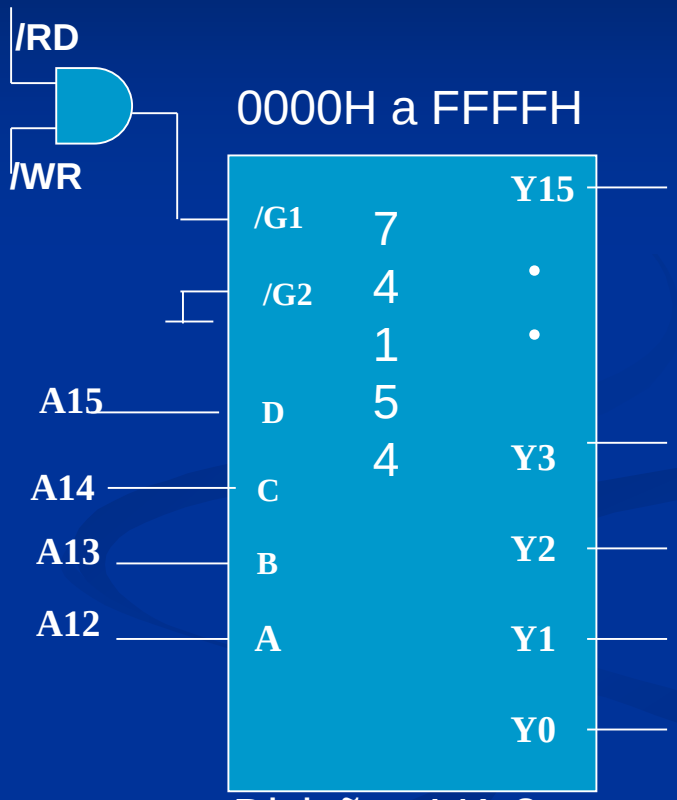
LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

Outros Exemplos:



Divisão: 8 Kx8
(1FFFH)

Saídas válida como /CS: Y0 a Y7



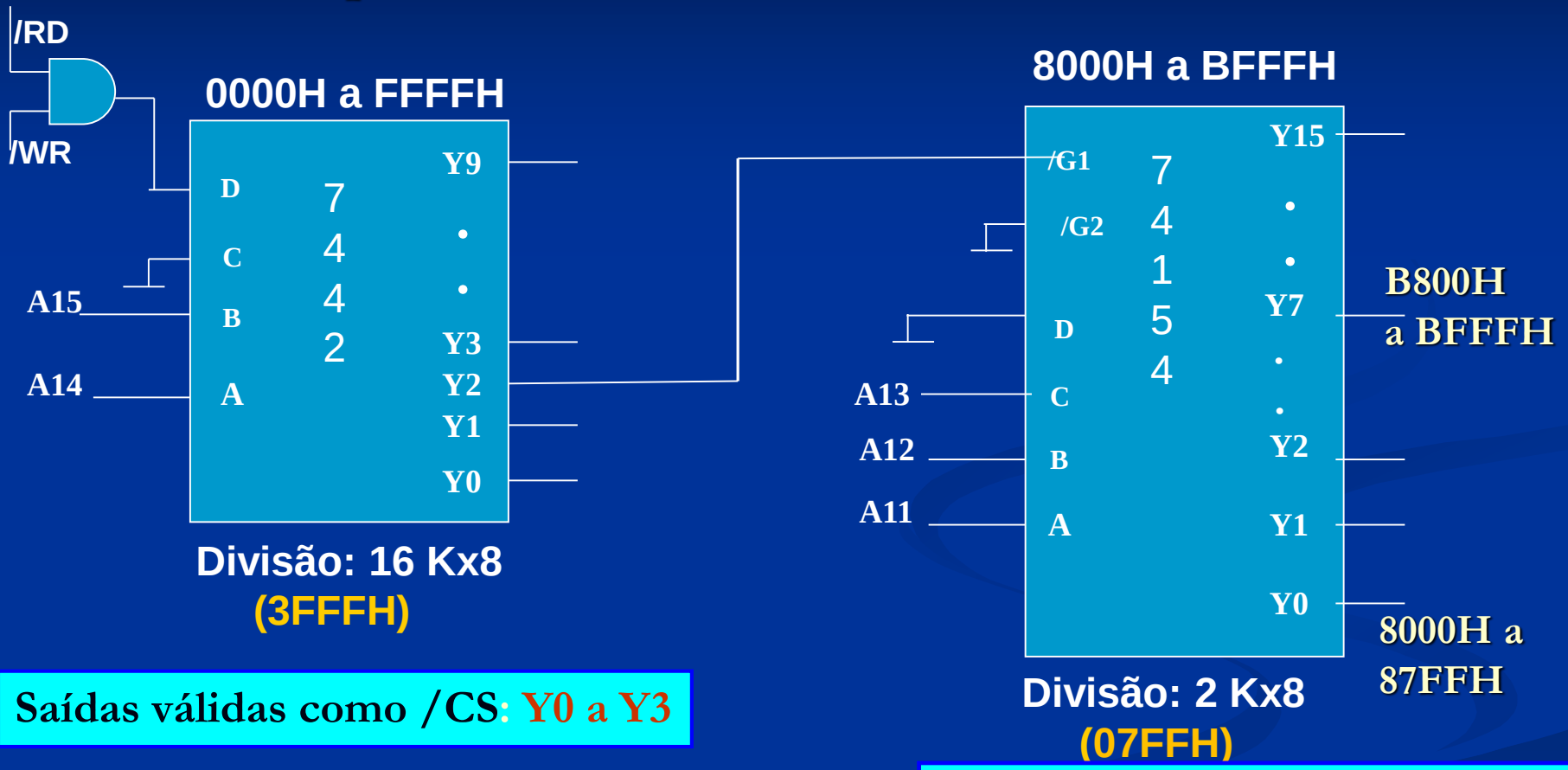
Divisão: 4 Kx8
(0FFFH)

Saídas válida como /CS: Y0 a Y15

Exercício: determinar a faixa de endereço associada a cada saída

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

Outros Exemplos:



Saídas válidas como /CS: Y0 a Y3

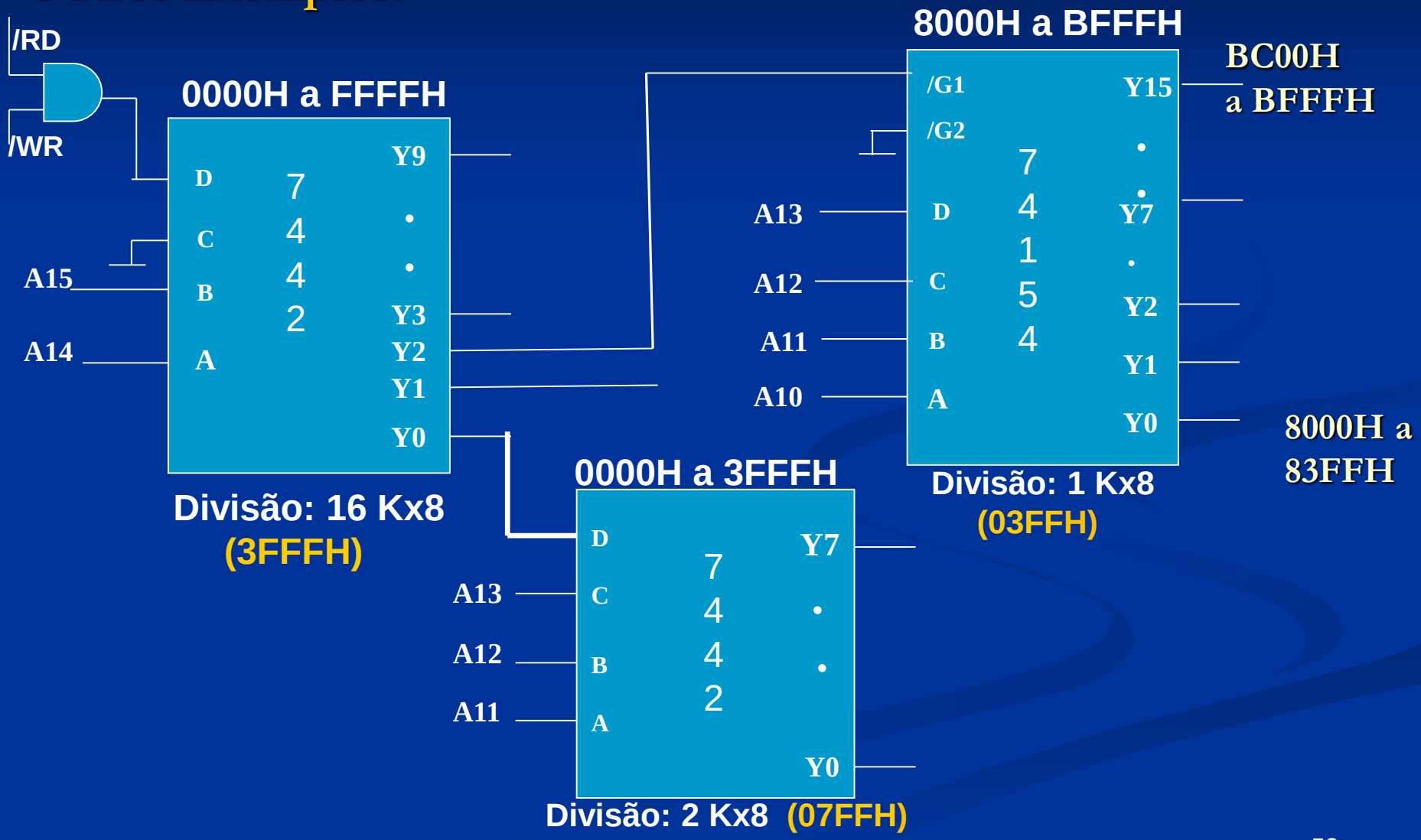
Saídas válidas como /CS: Y0 a Y7

Exercício: 1. determinar a faixa de endereço associada a cada saída válida

2. Substituir o decodificador 74154 pelo 7442 e efetuar as ligações necessárias para selecionar a mesma faixa de endereço.

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

Outros Exemplos:



Exercício: 1. determinar a faixa de endereço associada a cada saída válida.

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

Pode-se escolher qualquer endereço inicial ?

Resp: Não!!! há endereços que simplificam a lógica de seleção.

- Se a memória está **"alinhada"** com o endereço inicial, os bits de seleção tem o mesmo valor para qualquer posição da memória, o **que simplifica a lógica de seleção**

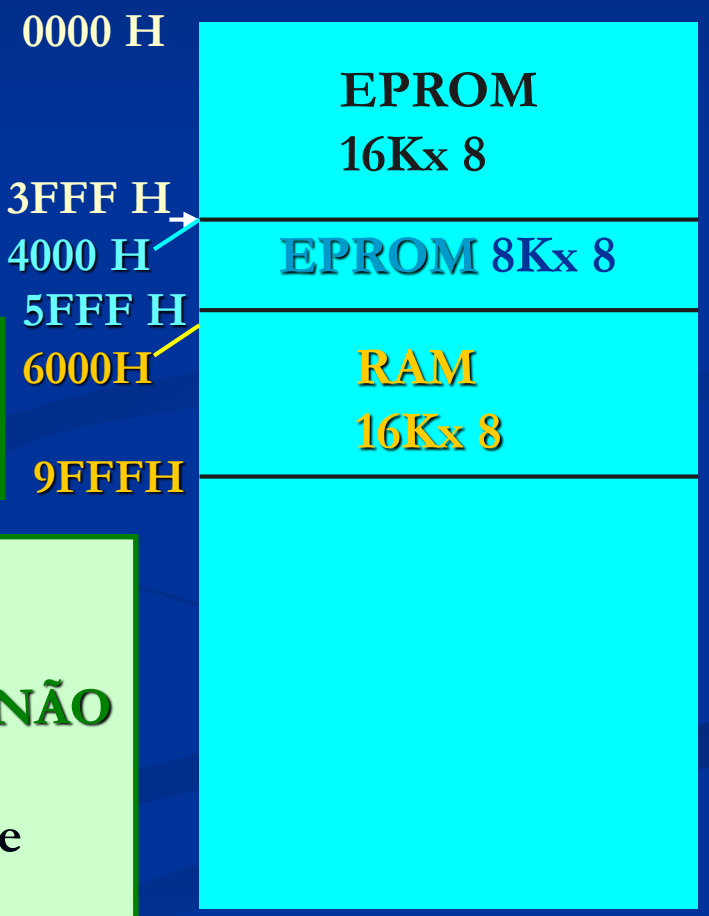
- a memória está alinhada com o endereço inicial se os **bits de endereçamento do bloco de memória** tem valor **zero** para o endereço inicial.

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

Exemplo:

a) Memória não alinhada - RAM de 16Kx8

	A13				
END. INICIAL:	0 1	10	0000	0000	0000
END. FINAL :	1 0	01	1111	1111	1111



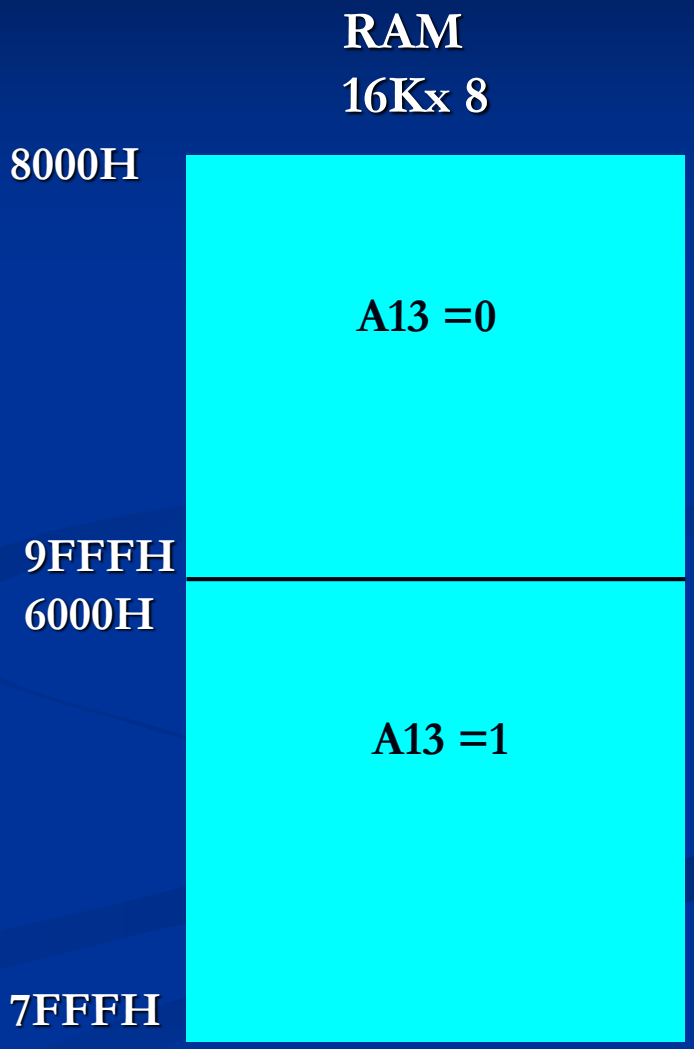
- A13 que é bit de endereçamento da memória, tem valor 1, para o end. inicial.
- Os bits de seleção para endereço inicial e final **NÃO** são os mesmos
- A lógica de seleção deve ser feita com divisão de 8Kx8 (ou menor) , pois para esse bloco os bits de Endereço se mantém em 0.

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

END. INICIAL:
6000 H : 0 1 10 0000 0000 0000

END. FINAL :
9FFFH: 1 0 01 1111 1111 1111

Obs: O chip da memória é acessado da metade até o fim (qdo A13=0) e depois do início até a metade(qdo A13=1)



LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

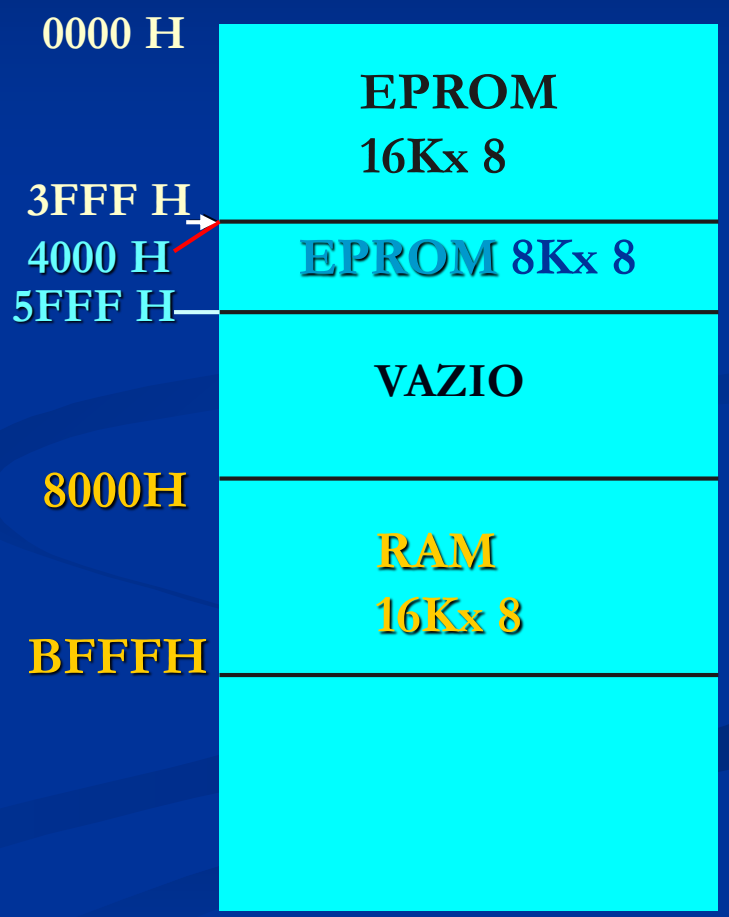
B) Memória alinhada - RAM de 16Kx8

Pode ser criado um "buraco" (vazio) para o alinhamento

A13

END. INICIAL: 1 0 00 0000 0000 0000

END. FINAL : 1 0 11 1111 1111 1111



- 0s bit de endereçamento da memória, A0 até A13, tem valor " 0".
- Os bits de seleção, para endereço inicial e final, são os mesmos
- O lógica de seleção deve ser feita com divisão de 16Kx8 , o mesmo tamanho da memória.

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

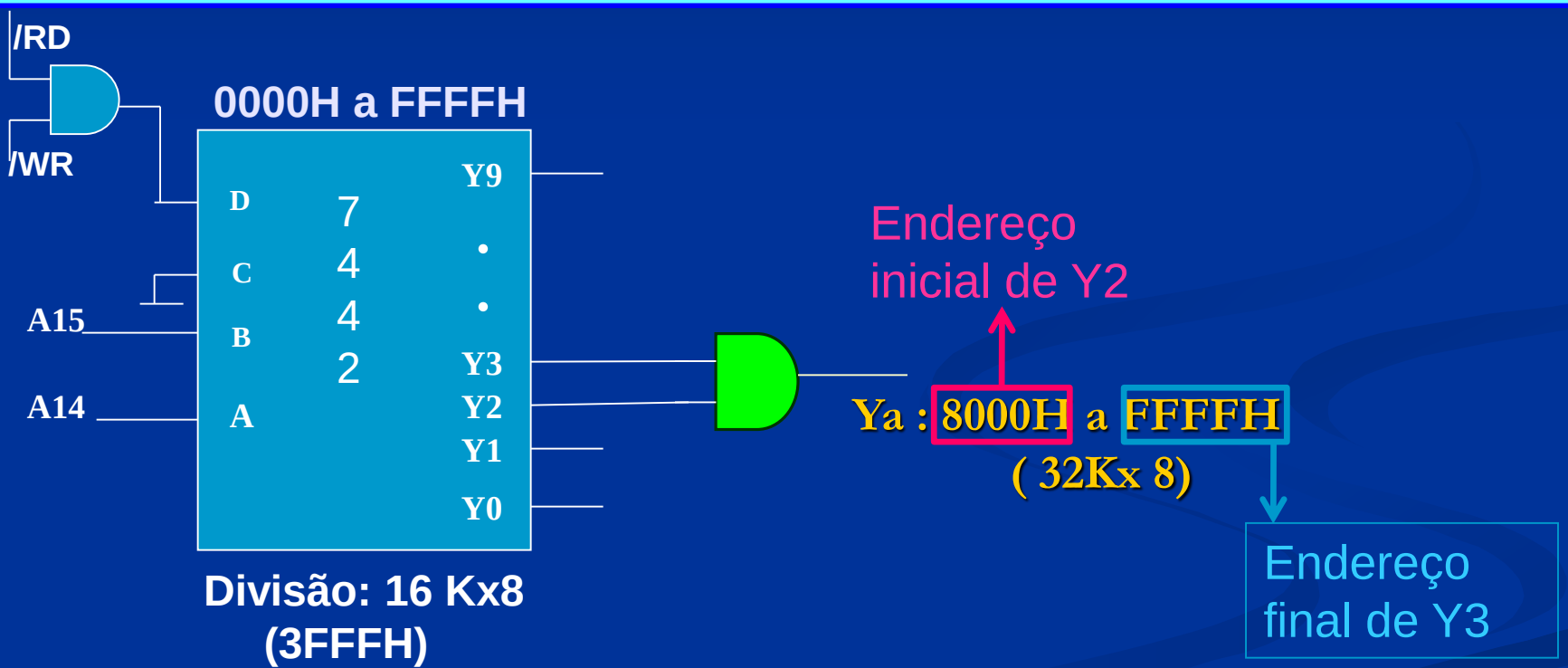
- **No mapeamento das memórias, é interessante que as memórias do tipo EPROM sejam mapeadas em sequência, para se ter continuidade do programa armazenado.**

- **Memórias do tipo RAM também são mapeadas em sequência para se ter continuidade na área de dados.**

- **Para evitar a ocorrência de memória não alinhada, além da escolha do endereço inicial adequado, as memórias devem ser mapeadas de forma que as de maior organização ocupem os blocos iniciais no mapa de endereços**

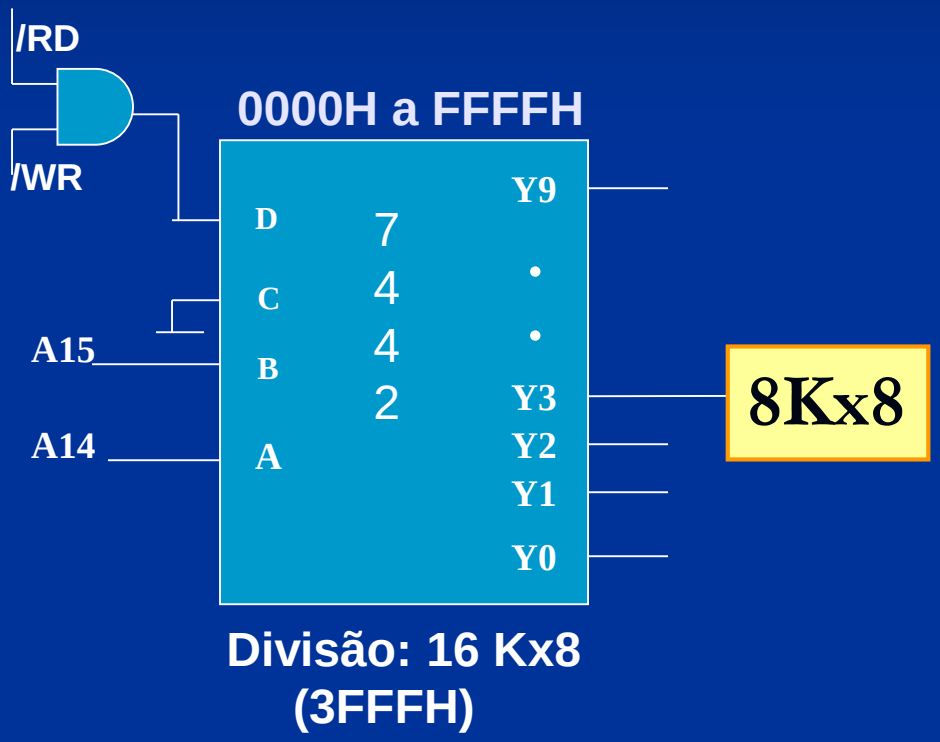
LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

Pode-se combinar linhas de seleção através da lógica AND, para selecionar organizações de memória MAIORES do que a faixa de endereço das saídas de seleção

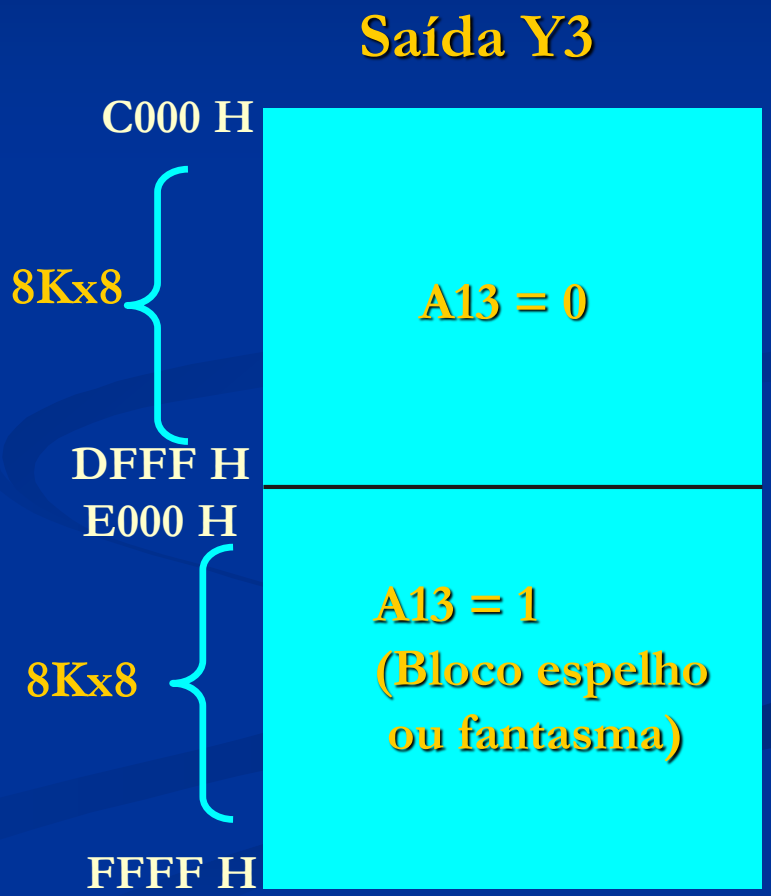


LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

O que ocorre se uma memória com organização menor que a da saída de seleção, for interligada na saída Y3?



Saída Y3: C000H a FFFFH



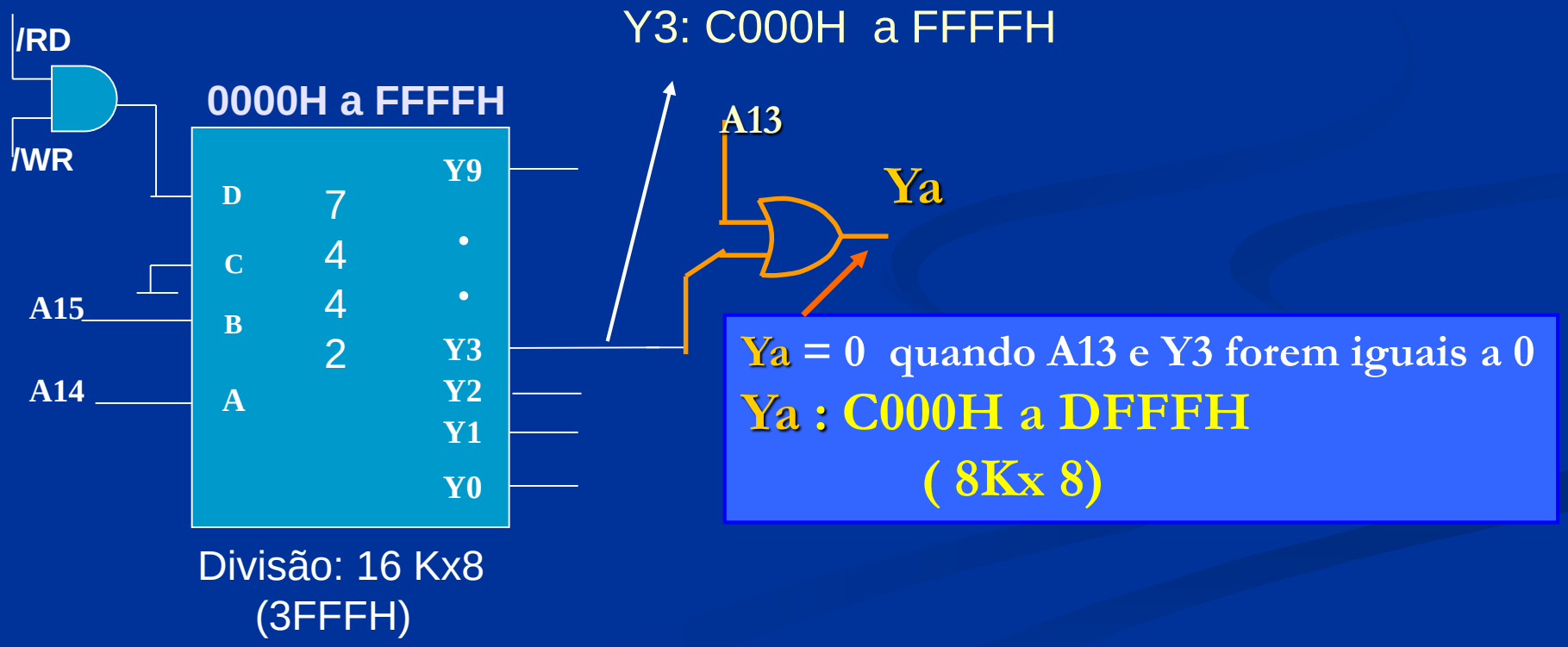
LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

A13 é bit de seleção da memória de 8Kx8, mas NÃO está presente no decodificador, portanto é irrelevante, podendo valer 1 ou 0.

- **Os dois blocos de 8K podem ser usados para selecionar a memória de 8Kx8 : C000H a DFFFH e E000H a FFFFH.**
- **Os dois blocos acessam as mesmas posições físicas das memórias, p. ex., C000H e E000H acessam a mesma posição da memória.**

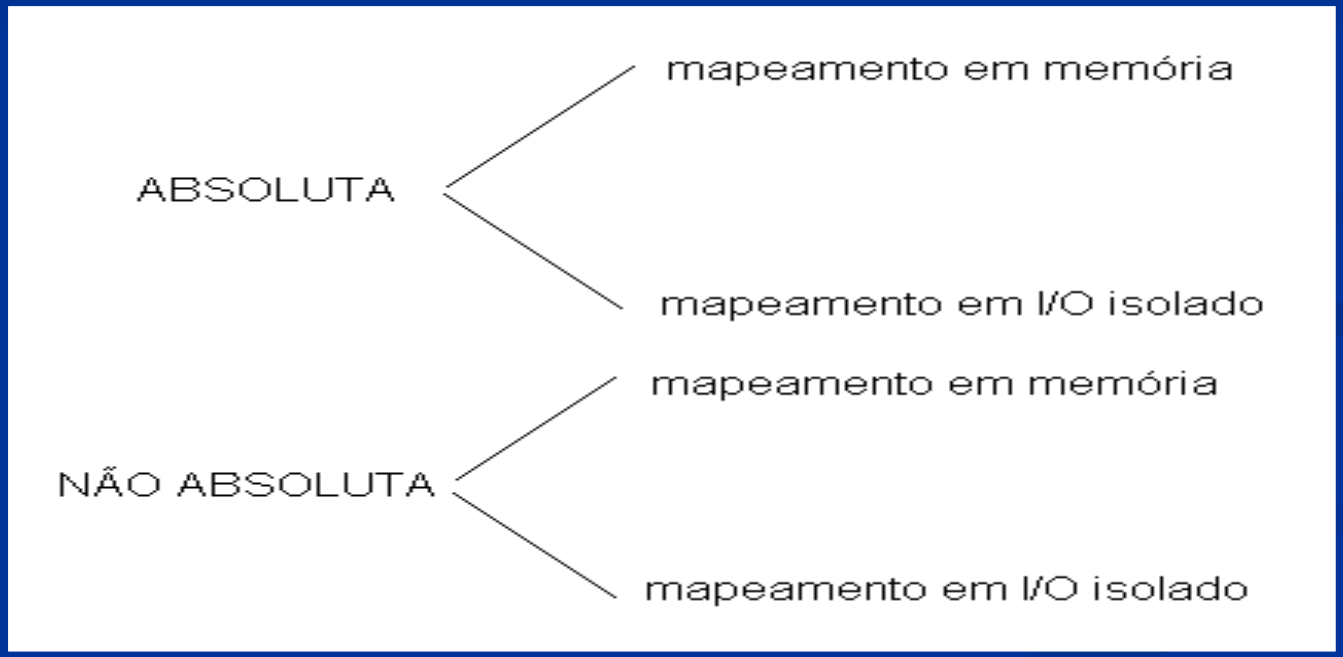
LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

Pode-se combinar linhas de seleção através da lógica OR, combinada com linhas de endereço adequadas, para selecionar organizações de memória MENORES do que a faixa de endereço das saídas de seleção.



LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

3. Tipos de Lógica de Seleção



LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

3. Tipos de Lógica de Seleção

Mapeamento em Memória: no espaço de 64K bytes são mapeadas as memórias e os dispositivos de I/O

- os sinais de controle de leitura e gravação são os mesmos para memórias e para I/O

Mapeamento em I/O isolado: os dispositivos de I/O são mapeados num espaço de I/O separado do espaço de memória.

- os sinais de controle de leitura e gravação são diferentes para os dois espaços
- É feita uma lógica de seleção separada para cada espaço

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

3.1 Lógica de Seleção Absoluta

- São usados todos os bits de seleção do CHIP
- É selecionado um espaço que tem exatamente as dimensões do chip. Exemplo: para memória de 1k x 8 é selecionado um bloco de 1k x 8.

• Se a memória está **"alinhada"** com o endereço inicial, os bits de seleção tem o mesmo valor para qualquer posição da memória, o que simplifica a lógica de seleção

• a memória está alinhada com o endereço inicial se os **bits de endereçamento do chip** tem valor **zero** para o endereço inicial.

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

3.1 Lógica de Seleção Absoluta

Vantagem: não há possibilidade de conflito de espaços de endereço.

Desvantagem: o hardware é mais complexo que o da lógica não absoluta, principalmente para interfaces que ocupam poucas posições de memória

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

3.1.a Exemplos de lógica de seleção absoluta, Mapeamento em Memória

Para o circuito de seleção da figura 2 é determinada a faixa de endereço associada a cada saída de seleção CSi (tabela 2 e 3)

No espaço de 64K bytes são mapeadas as memórias e os dispositivos de I/O

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

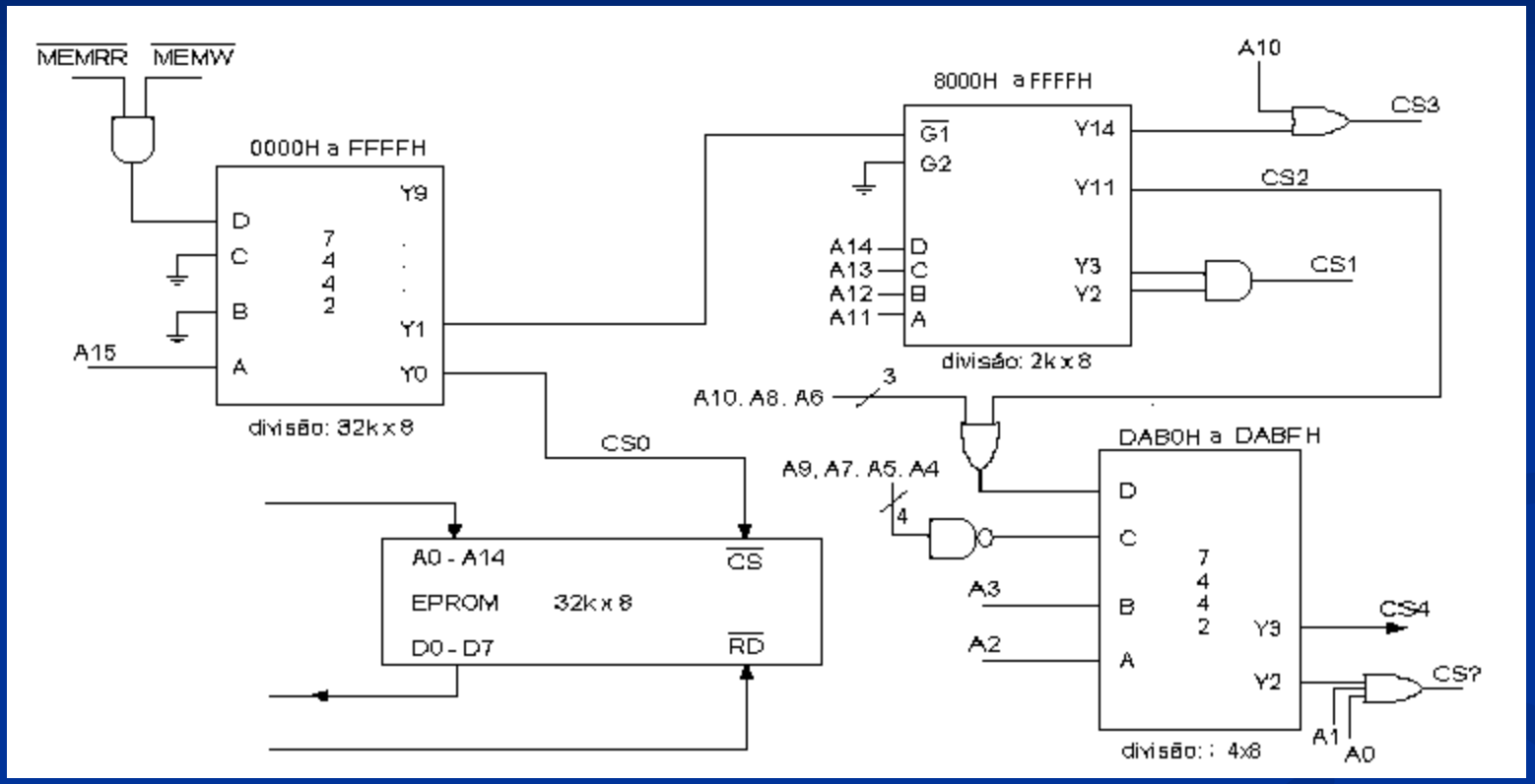


Figura 2 - Exemplo de lógica de seleção absoluta , mapeamento em memória

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

Tabela 2 - Faixa de endereços para cada saída de seleção

	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
CS0 (32k x 8)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
CS1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
CS2	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0
	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
CS3	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1
CS4	1	1	0	1	1	0	1	0	1	0	1	1	1	1	0	0
	1	1	0	1	1	0	1	0	1	0	1	1	1	1	1	1

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

- CS0 : 0000 -> 7FFFH ; 32k x 8
- CS1 : 9000H -> 9FFFH ; 4k x 8
- CS2 : D800H -> DFFFH ; 2k x 8
- CS3 : F000H -> F3FFH ; 1k x 8
- CS4 : DABCH -> DABFH ; 4 x 8

Tabela 3 - Faixa de endereços em hexadecimal, referentes à tabela 2

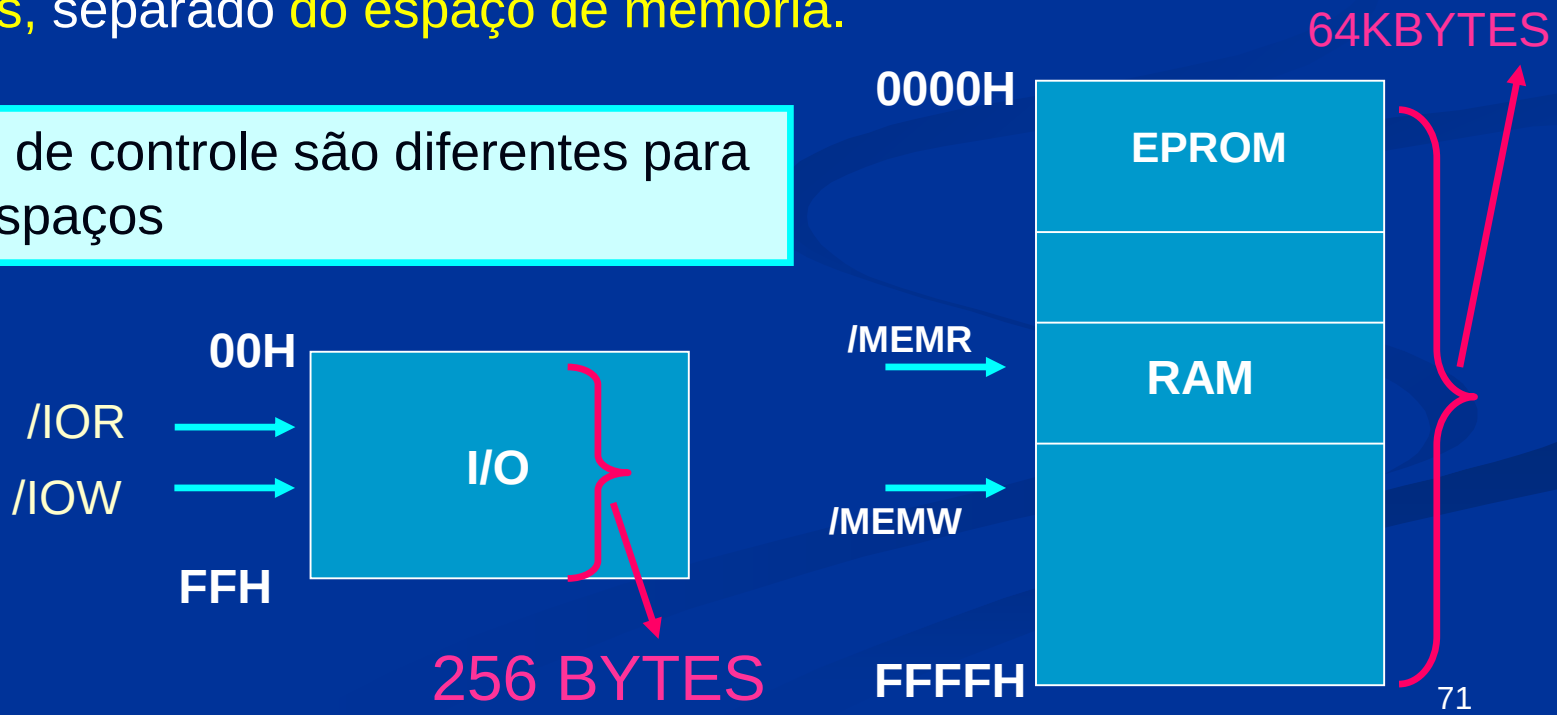
LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

3.1.b Exemplos de lógica de seleção absoluta, Mapeamento em I/O Isolado

Para o circuito de seleção da figura 3 é determinada a faixa de endereço associada a cada saída de seleção CSi (tabela 4) .

Os dispositivos de I/O são mapeados num espaço de I/O de 256 bytes, separado do espaço de memória.

Os sinais de controle são diferentes para os dois espaços



LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

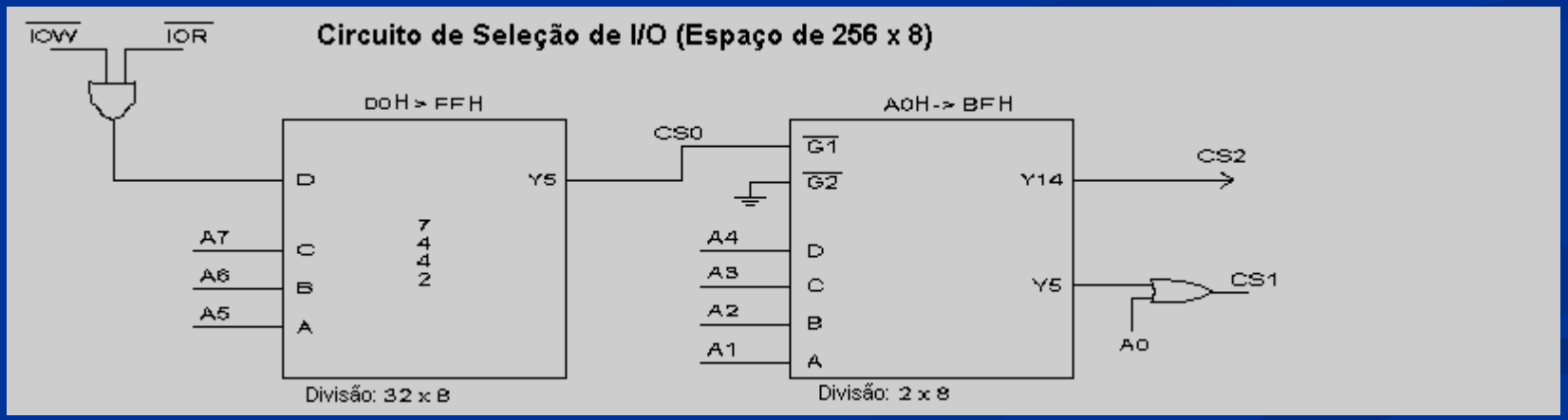


Figura 3 - exemplo de lógica de seleção absoluta, mapeamento em I/O isolado

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

	A7	A6	A5	A4	A3	A2	A1	A0
CS0	1	0	1	0	0	0	0	0
	1	0	1	1	1	1	1	1
CS1	1	0	1	0	1	0	1	0
	1	0	1	0	1	0	1	0
CS2	1	0	1	1	1	1	0	0
	1	0	1	1	1	1	0	1

CS0 : A0H -> BFH -> espaço: 32 x 8
CS1 : AAH -> espaço: 1 x 8
CS2 : BCH -> BDH -> espaço: 2 x 8

Tabela 4 : mapa de endereços para o circuito de seleção de I/O da figura 3

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

3.2 Lógica de Seleção NÃO Absoluta

- Usa-se parte dos bits de seleção, na lógica de seleção
- os bits de seleção não usados geram espaços de endereço extra, associados ao dispositvo, denominados **espaços de endereço fantasma**;
  - esse hardware de seleção é inadequado para área de dados seqüenciais;
  - os endereços fantasmas não podem ser usados por outros chips, pois não são espaços livres.

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

3.2.a Lógica de Seleção NÃO Absoluta, mapeamento em memória

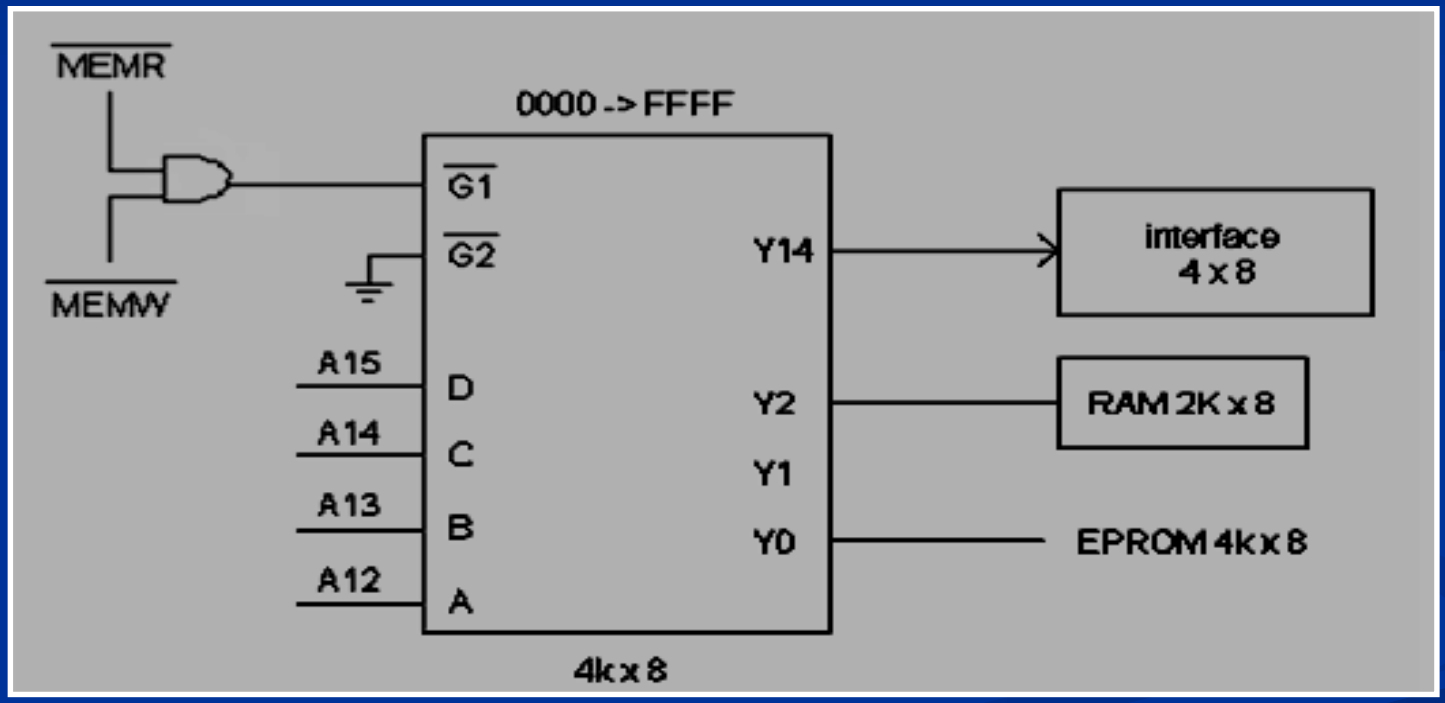


Figura 4 - Exemplo de lógica de seleção não absoluta , mapeamento em memória

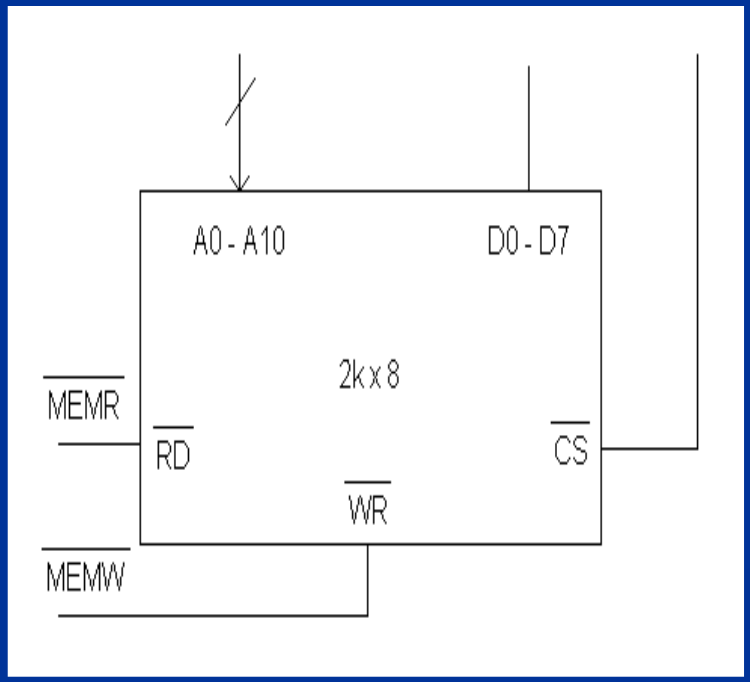
LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

Faixas de endereço para a figura 4

a)Y0 : 0000H -> 0FFF H(4k x 8)

b) Y2 : 2000H -> 2FFFH (4k x 8)

A memória conectada a Y2 é de 2k x 8 , organização menor do que o espaço gerado pela lógica de seleção.



O bit de endereço **A11** não está presente na lógica de seleção para a memória de **2k x 8**, o que leva essa memória a ter dois espaços de endereço associados a ela

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

Faixa de endereço para a memória RAM de 2Kx8

	A15	A14	A13	A12	A11	A10	A 9	A 8	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0
Y2	0	0	1	0	X	0	0	0	0	0	0	0	0	0	0	0
	0	0	1	0	X	1	1	1	1	1	1	1	1	1	1	1

Se **X = 0** **a faixa de endereço = 2000H a 27FFH**
Se **X = 1** **a faixa de endereço = 2800H a 2FFFH**

- Um dos espaços de 2k x 8 é denominado de **espaço fantasma (espelho)**.
- **Qualquer uma** das duas faixas de endereço pode ser escolhida como a fantasma.
- endereço 2000H e 2800H endereçam a mesma posição física do CHIP
- (linhas de endereçamento do chip são iguais a zero para esses dois endereços).

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

c. Faixa de endereço para interface de 4 x 8

Y14: E000H -> EFFFH (4Kx8)

1ª. Faixa de endereço da interface: E000 -> E003H

2ª. Faixa de endereço da interface: E004 -> E007H

.

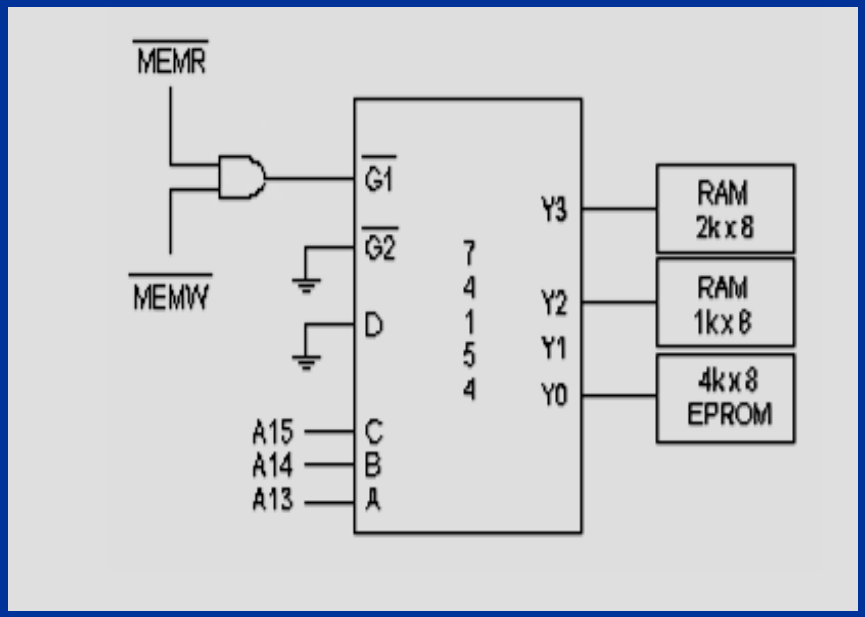
.

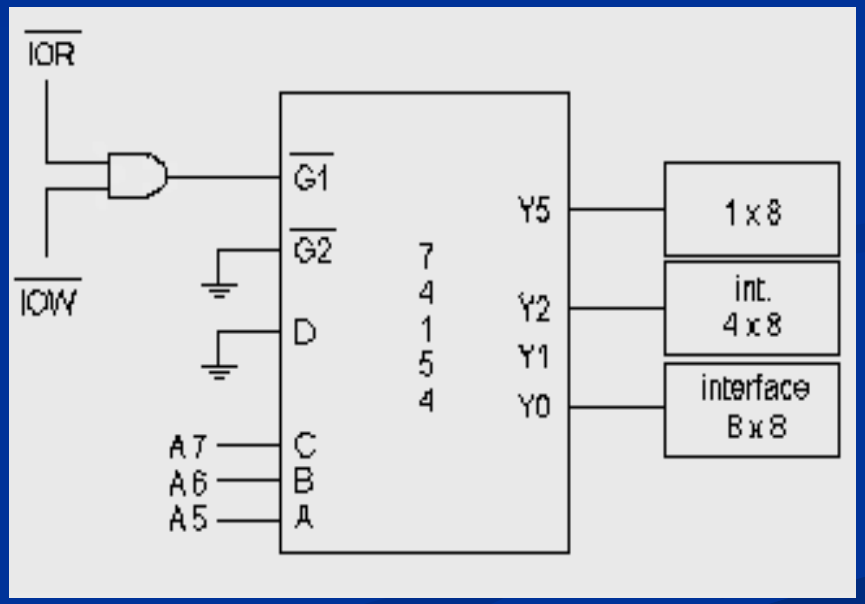
.

Quantas faixas de endereço estarão associadas a essa interface?

LÓGICA DE SELEÇÃO DE MEMÓRIA E DE DISPOSITIVOS DE I/O

3.2.b. - Exemplo de lógica de seleção não absoluta, mapeamento I/O isolado





Determine a faixa de endereço para cada saída de seleção, e os endereços fantasmas

Exemplos

Exercício 1 da Lista nº9

Faça o projeto da lógica de seleção para dividir o espaço de endereço do microprocessador em blocos de 4Kbytes, especificando endereço inicial e final de cada bloco. Divida o bloco que inicia no endereço 4000H, em blocos de 1Kbytes e o bloco que inicia no endereço A000H em blocos de 512 bytes. Usando lógica de seleção absoluta, ligue uma memória de 2kbytes, a partir do endereço 1000H, outra a partir do endereço 4000H e outraa partir do endereço A000H, determinando o endereço final de cada memória. Repita o mesmo para lógica de seleção não absoluta, determinando os endereços fantasmas. Justifique quando a lógica de seleção absoluta não se aplicar.

Ex. 3: Endereçamento de um bloco de memórias utilizando

Decodificação Absoluta

3 Memórias na seqüência

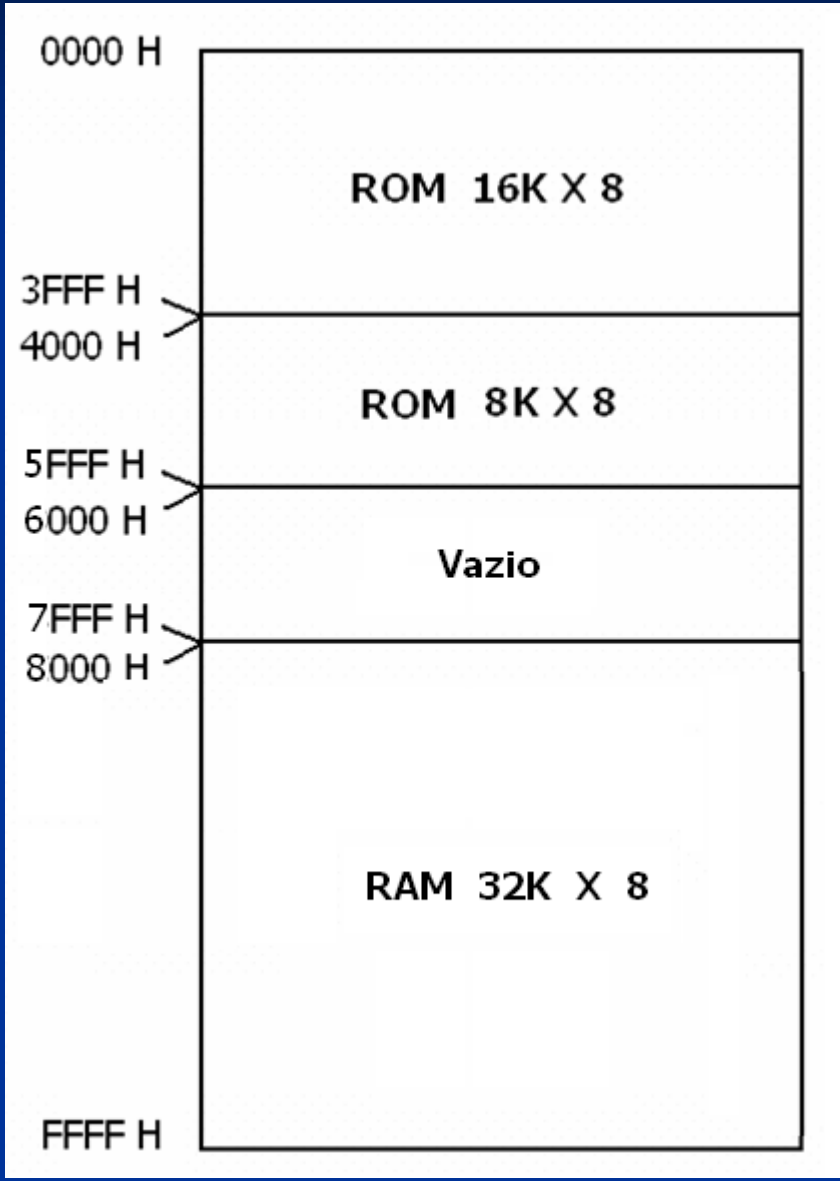
Com alinhamento dos CIs

- ROM 16 k x 8
- ROM 8 k x 8
- RAM 32 k x 8

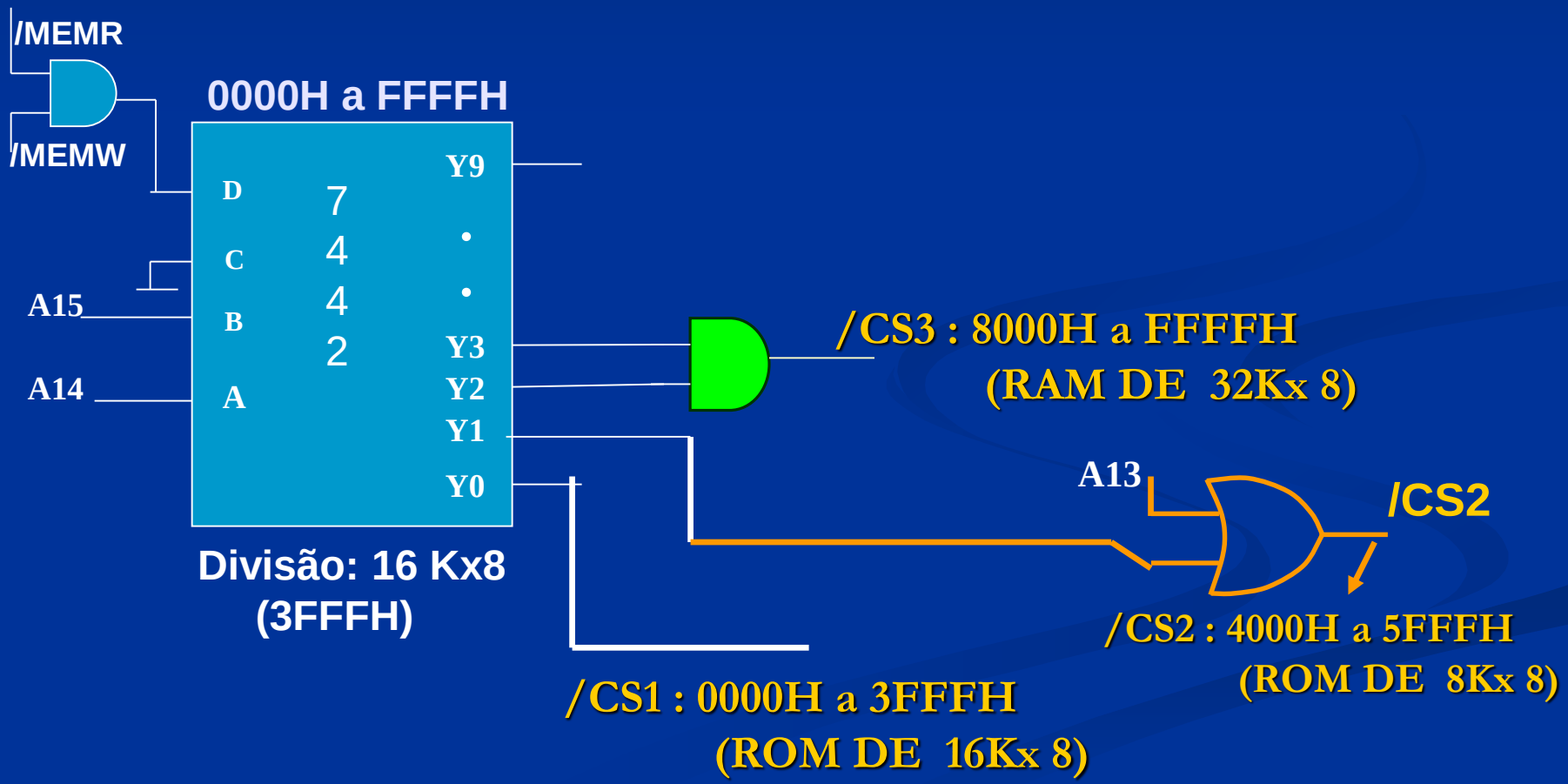
Lógica de Seleção – Decodif. Absoluta com alinhamento das memórias

Lógica de Endereçamento do μP – Endereço de dados																	Memória		
Tipo	**A15**	**A14**	**A13**	**A12**	**A11**	**A10**	**A9**	**A8**	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**	**Início (H)**		**Fim (H)**
ROM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0000**	16k	
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1			**3FFF**
ROM	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**4000**	8 k	
	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1			**5FFF**
RAM	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**8000**	32 k	
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			**FFFF**
Vazio	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	**6000**	8 k	
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			**7FFF**

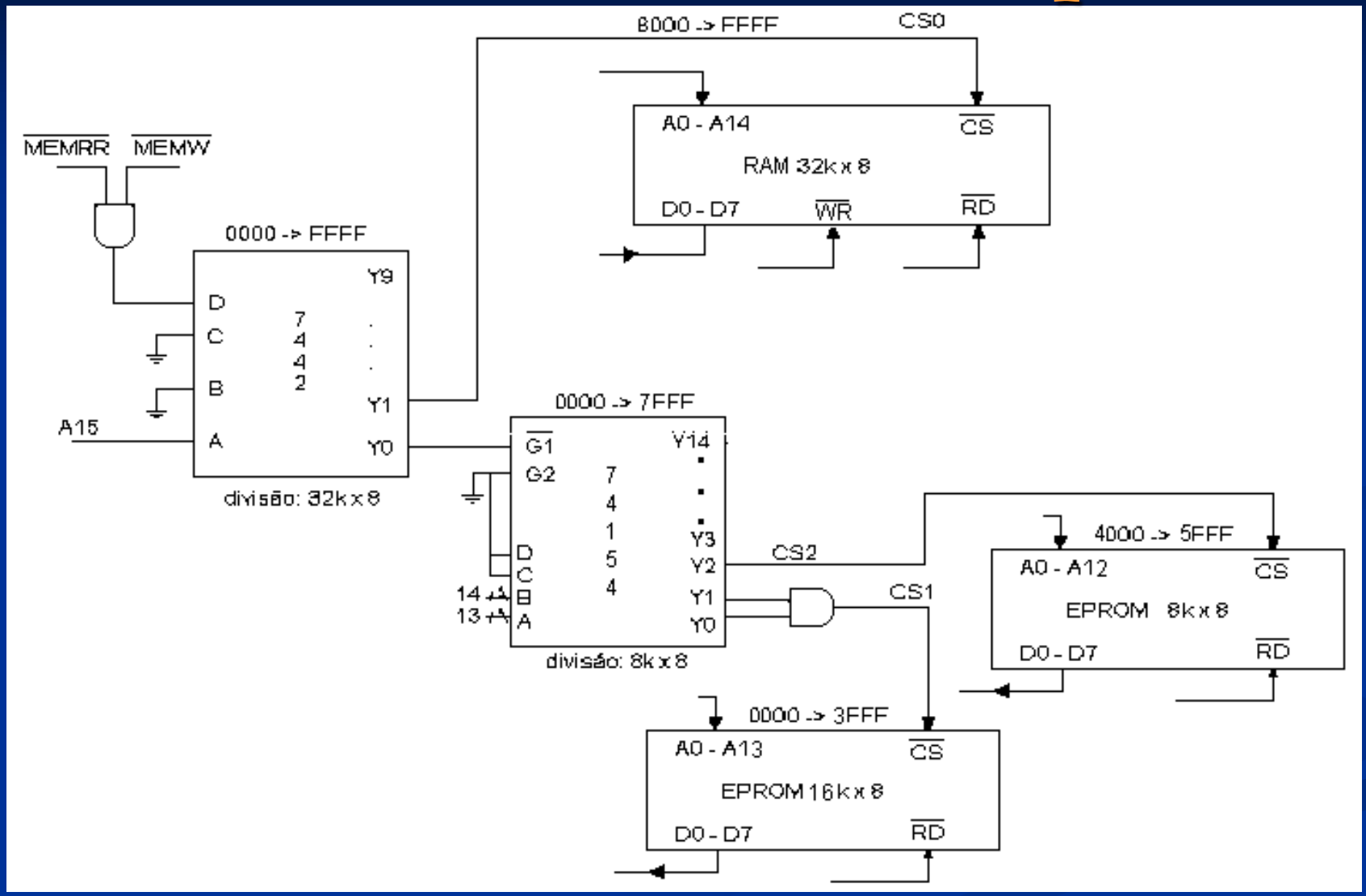
Mapeamento da Memória



Circuito Final - Exemplo 1



Circuito Final - Exemplo 2



Ex. 4: Endereçamento de um bloco de memórias utilizando

Decodificação Absoluta:

5 Memórias na seqüência

Sem o alinhamento dos CIs

- ROM 16 k x 8
- ROM 8 k x 8
- RAM 32 k x 8
- 2 x RAM 4 k x 8

Lógica de Seleção – Decodif. Absoluta

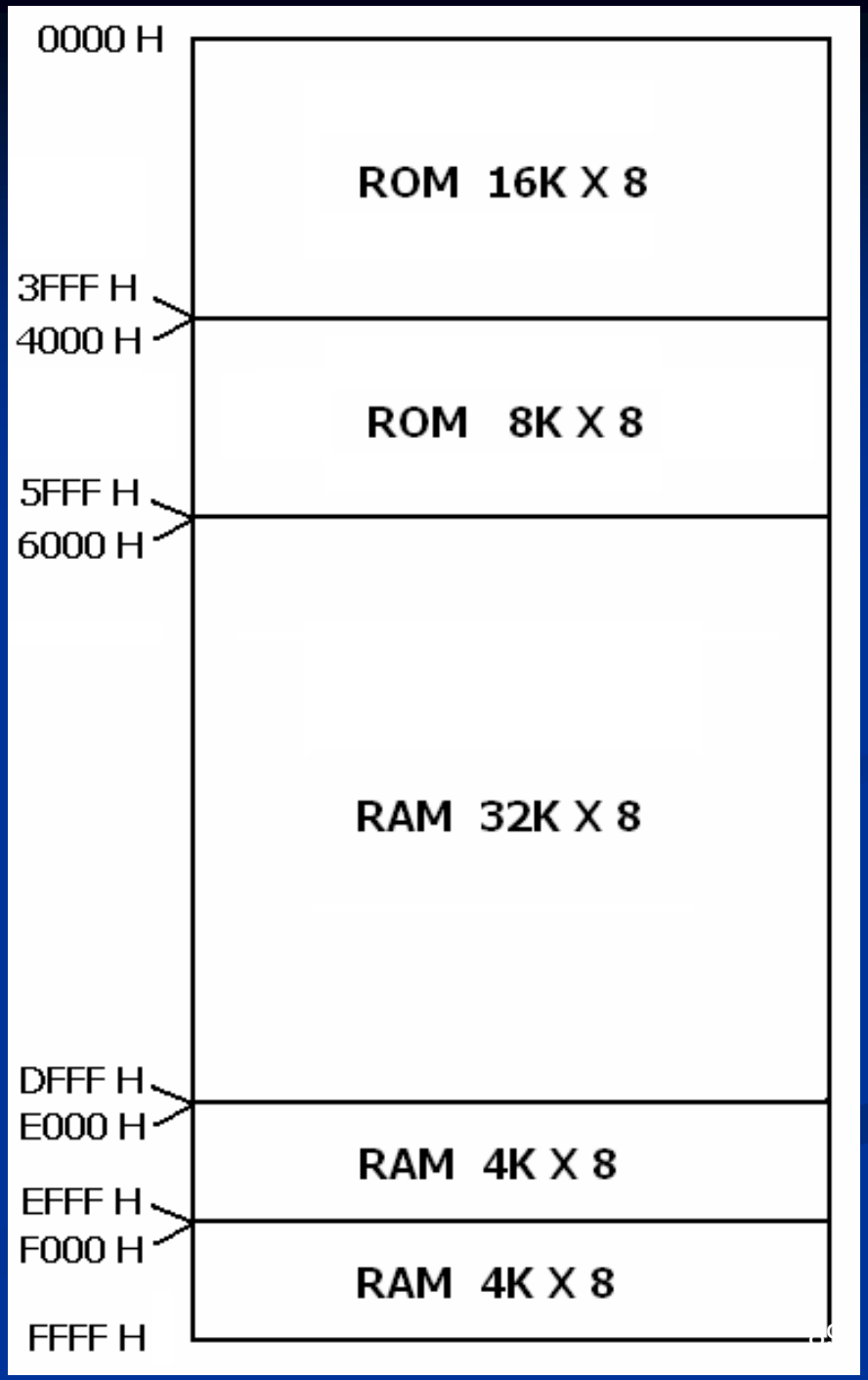
Lógica de Endereçamento do µP – Endereço de dados																	Memória		
Tipo	**A15**	**A14**	**A13**	**A12**	**A11**	**A10**	**A9**	**A8**	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**	**Início (H)**		**Fim (H)**
ROM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0000**	16k	
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1			**3FFF**
ROM	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**4000**	8 k	
	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1			**5FFF**
RAM	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	**6000**	32k	
	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1			**DFFF**
RAM	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	**E000**	4 k	
	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1			**EFFF**
RAM	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	**F000**	4 k	
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			**FFFF**

O CI de memória RAM 32K x 8, nesse caso, não será endereçado na seqüência normal (0000h – 7FFFh). Mesmo assim, TODOS os seus endereços serão utilizados.

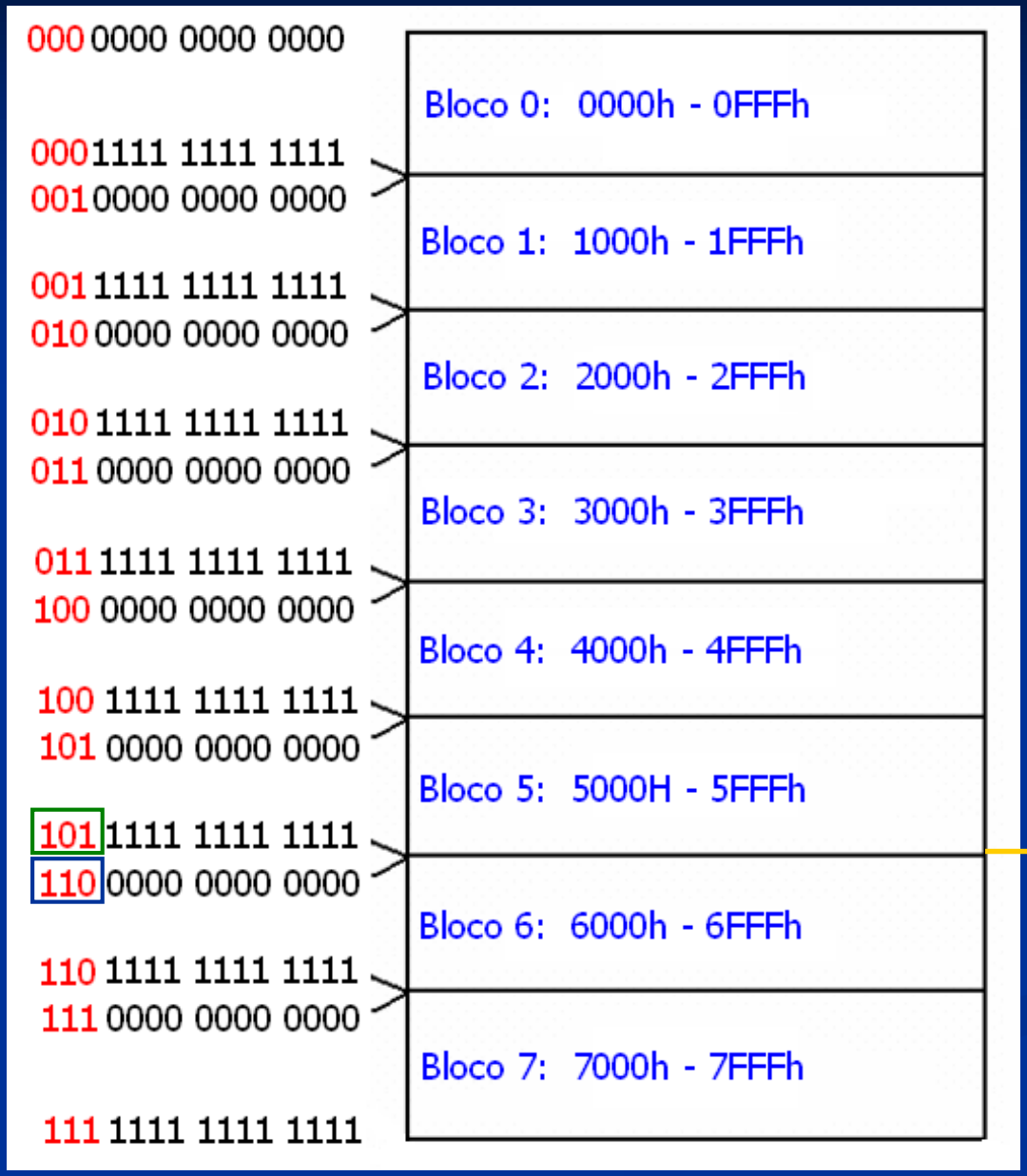
Mapeamento da Memória

Decodificação Absoluta

Obs: Memória de 32Kx8 não alinhada



Como é endereçada a Memória RAM 32K x 8 não alinhada?



A seqüência de seleção da memória RAM 32K x 8 nesse caso será:

Blocos: 6 → 7 → 0 → 1 → 2 → 3 → 4 → 5

A_{15} A_{14} A_{13} A_{12}

1 1 0 1

Bloco 5 será o último a ser selecionado

A_{15} A_{14} A_{13} A_{12}

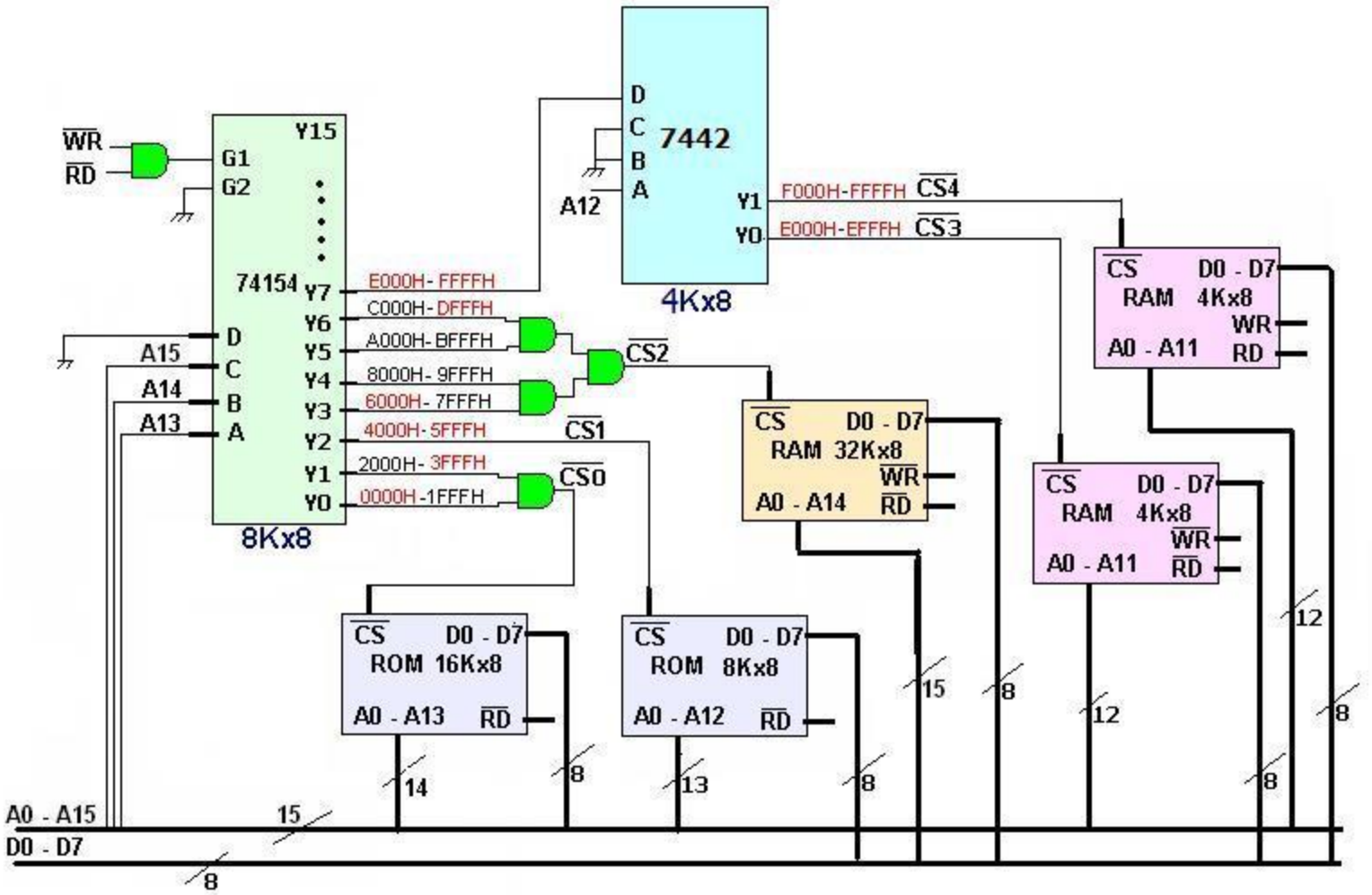
0 1 1 0

Bloco 6 será o primeiro a ser selecionado

Decodificação Absoluta

Lógica de Endereçamento do µP – Endereço de dados																	Memória		
Tipo	**A15**	**A14**	**A13**	**A12**	**A11**	**A10**	**A9**	**A8**	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**	**Início (H)**		**Fim (H)**
ROM	**0**	**0**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	**0000**	16k	
	0	**0**	**1**	**1**	1	1	1	1	1	1	1	1	1	1	1	1			**3FFF**
ROM	**0**	**1**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	**4000**	8 k	
	0	**1**	**0**	**1**	1	1	1	1	1	1	1	1	1	1	1	1			**5FFF**
RAM	**0**	**1**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	**6000**	32k	
	1	**1**	**0**	**1**	1	1	1	1	1	1	1	1	1	1	1	1			**DFFF**
RAM	**1**	**1**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	0	**E000**	4 k	
	1	**1**	**1**	**0**	1	1	1	1	1	1	1	1	1	1	1	1			**EFFF**
RAM	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	**F000**	4 k	
	1	**1**	**1**	**1**	1	1	1	1	1	1	1	1	1	1	1	1			**FFFF**

1º Exemplo de Implementação:



Como é endereçada a Memória RAM 32K x 8 não alinhada para o exemplo 1?

0110 0000 0000 0000 0111 1111 1111 1111	Bloco 3: 6000 -7FFFF
1000 0000 0000 0000 1001 1111 1111 1111	Bloco 0: 8000 -9FFFF
1010 0000 0000 0000 1011 1111 1111 1111	Bloco 1: A000 -BFFFF
1100 0000 0000 0000 1101 1111 1111 1111	Bloco 2: C000 -DFFFF

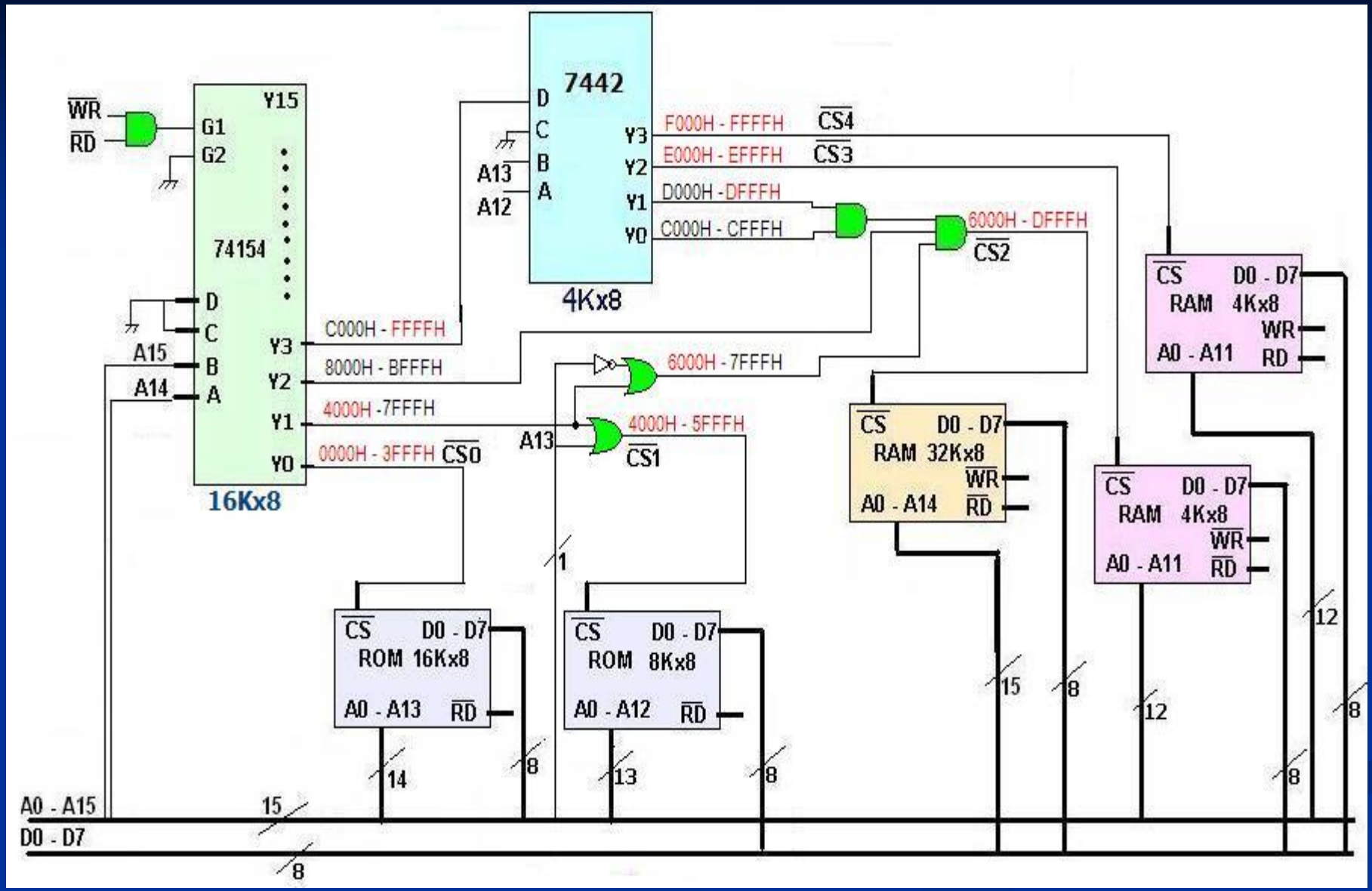
A seqüência de seleção da memória RAM 32K x 8 nesse caso será:

Blocos: 1 → 2 → 3 → 0

Memória de 32Kx8

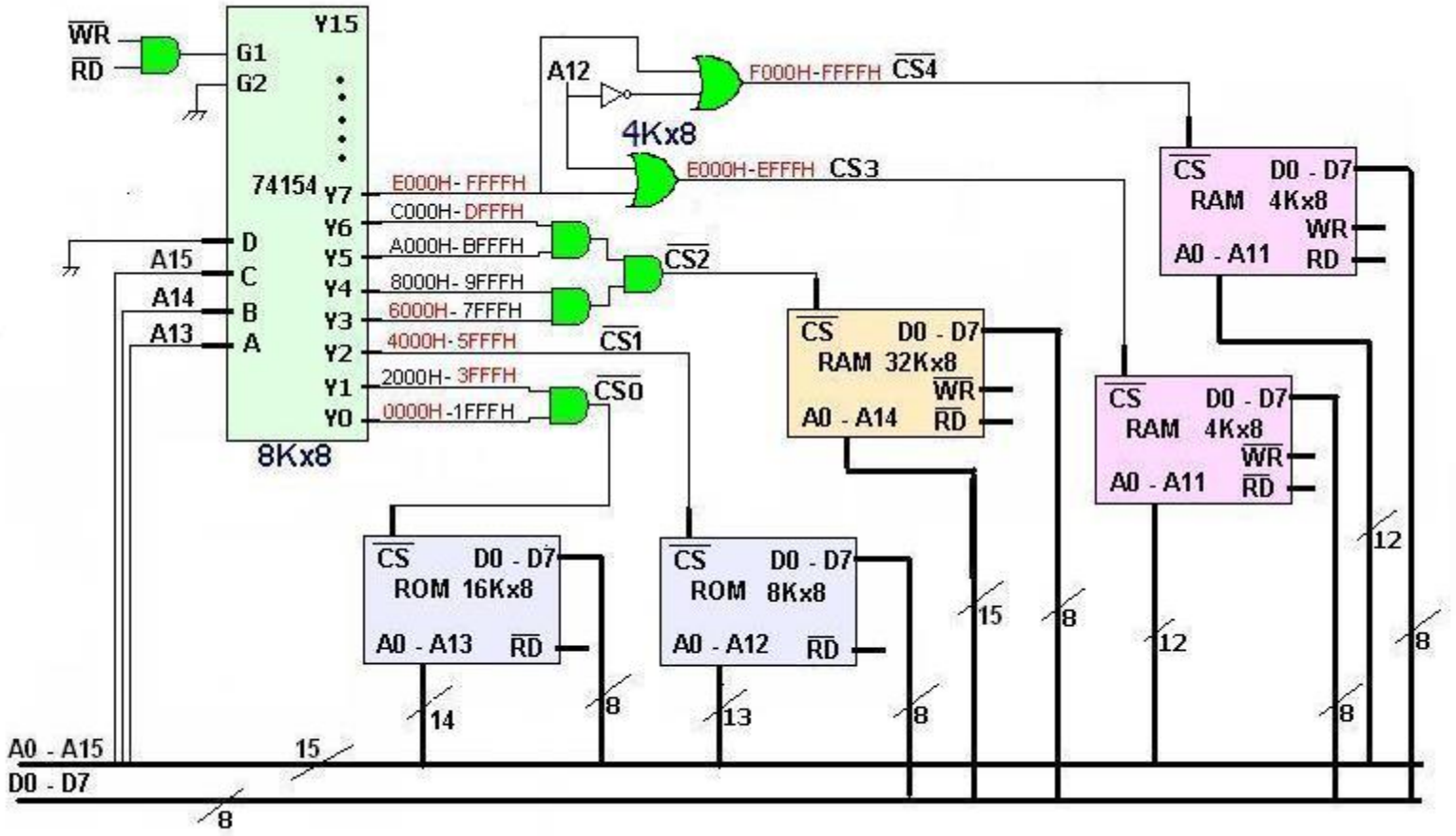
Bloco 0: (A14,A13) = (0,0)
Bloco 1: (A14,A13) = (0,1)
Bloco 2: (A14,A13) = (1,0)
Bloco 3: (A14,A13) = (1,1)

2º Exemplo de Implementação:



Exemplo de Implementação INCORRETA:

Obs: a lógica de seleção deve ser feita com decodificadores e não com uma quantidade grande de portas lógicas OR e AND



Ex. 5: Endereçamento de um bloco de memórias utilizando

Decodificação Não-Absoluta:

3 Memórias na seqüência

Com alinhamento dos CIs

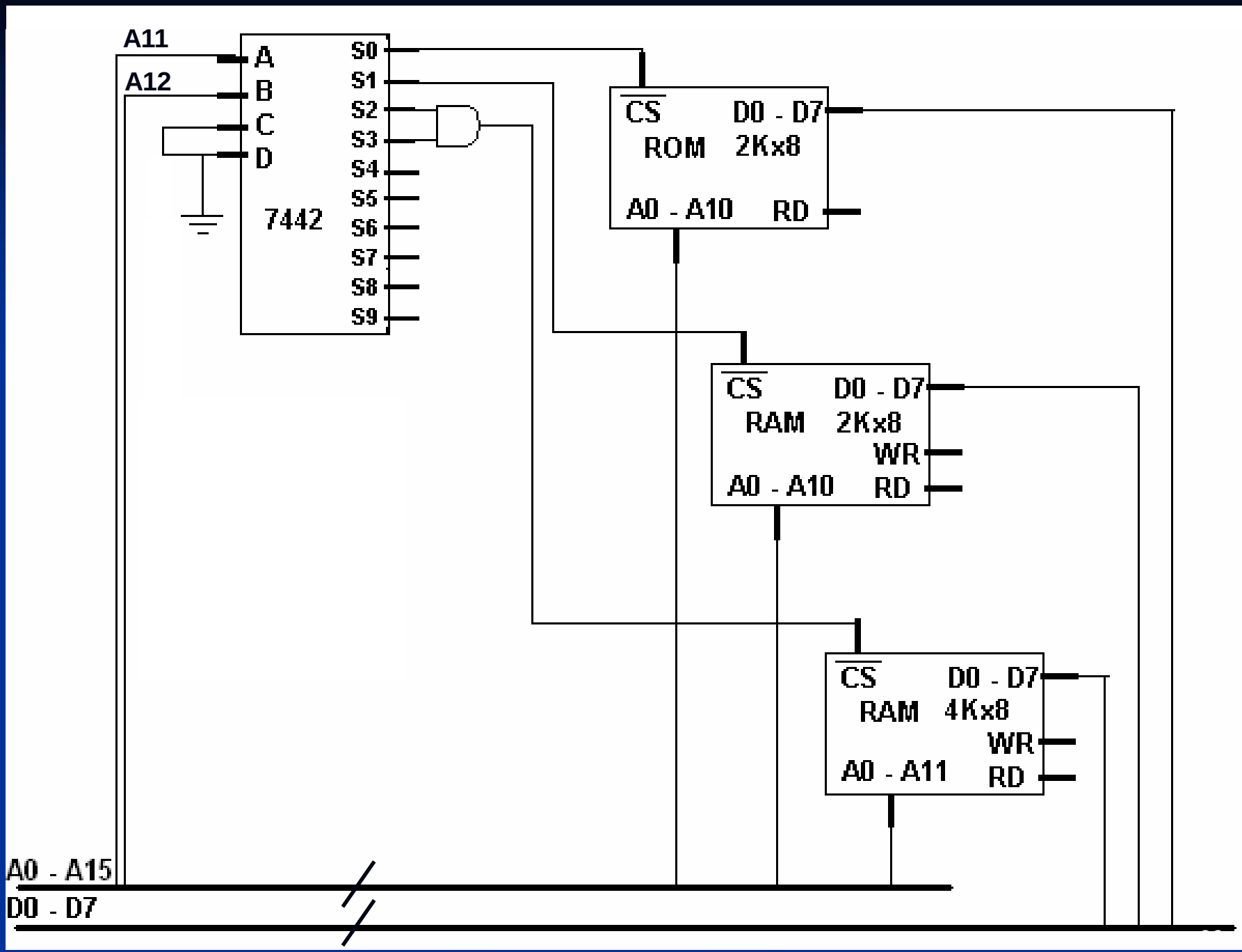
- ROM 2 k x 8
- RAM 2 k x 8
- RAM 4 k x 8

Lógica de Seleção – Decodif. Não-Absoluta

Lógica de Endereçamento do µP – Endereço de dados																	Memória		
Tipo	**A15**	**A14**	**A13**	**A12**	**A11**	**A10**	**A9**	**A8**	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**	**Início (H)**		**Fim (H)**
ROM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**0000**	2 k	
	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1			**07FF**
RAM	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	**0800**	2 k	
	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1			**0FFF**
RAM	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	**1000**	4 k	
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1			**1FFF**

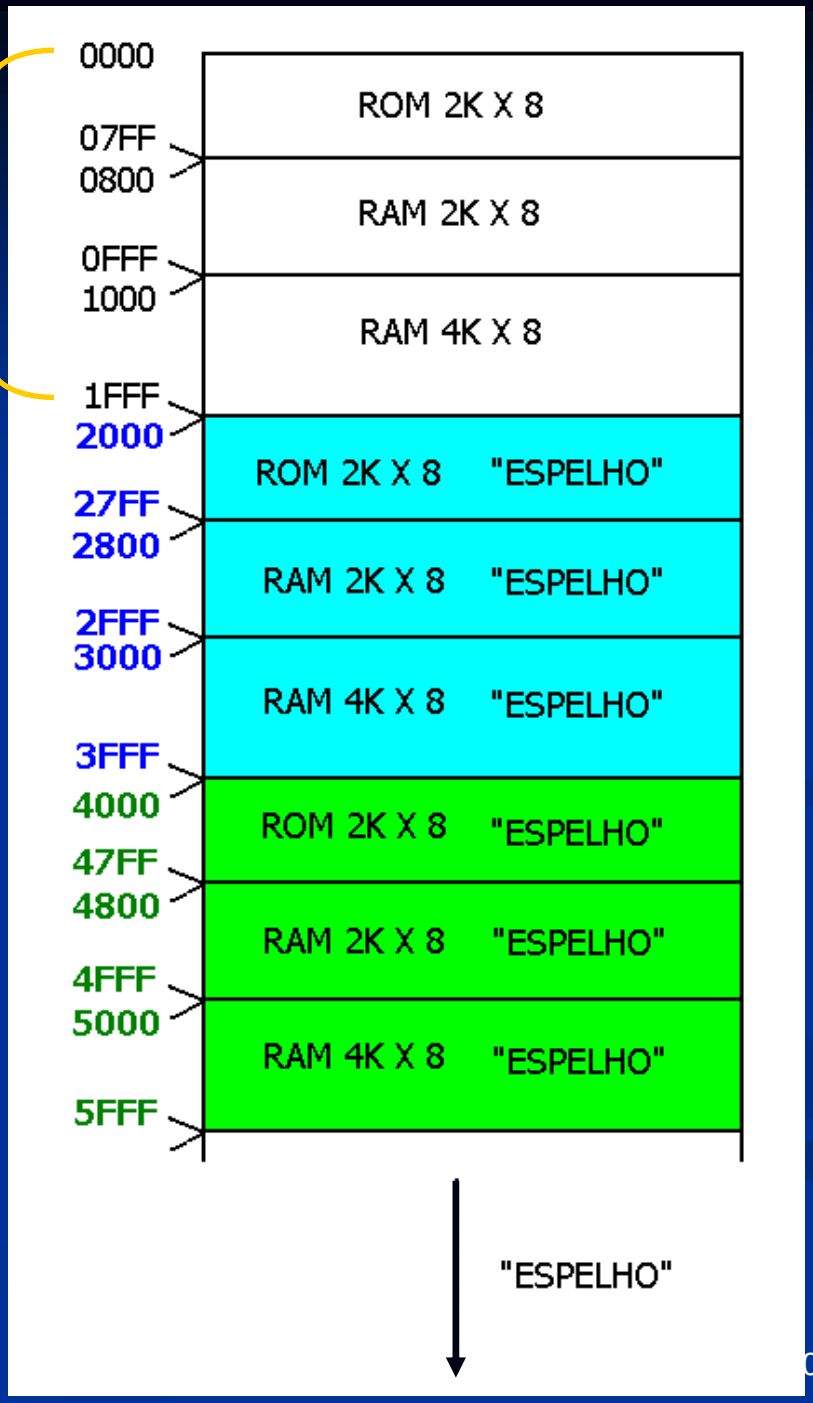
Decodificação Não-Absoluta

Lógica de Endereçamento do µP – Endereço de dados																	Memória		
Tipo	**A15**	**A14**	**A13**	**A12**	**A11**	**A10**	**A9**	**A8**	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**	**Início (H)**		**Fim (H)**
ROM	0	0	0	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	**0000**	2 k	
	0	0	0	**0**	**0**	1	1	1	1	1	1	1	1	1	1	1			**07FF**
RAM	0	0	0	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0	**0800**	2 k	
	0	0	0	**0**	**1**	1	1	1	1	1	1	1	1	1	1	1			**0FFF**
RAM	0	0	0	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	**1000**	4 k	
	0	0	0	**1**	**1**	1	1	1	1	1	1	1	1	1	1	1			**1FFF**


A11
A12
A
B
C
D
7442
S0
S1
S2
S3
S4
S5
S6
S7
S8
S9
CS
ROM 2Kx8
D0 - D7
A0 - A10
RD
CS
RAM 2Kx8
WR
D0 - D7
A0 - A10
RD
CS
RAM 4Kx8
WR
D0 - D7
A0 - A11
RD
A0 - A15
D0 - D7

Mapeamento da Memória

Esse bloco é “espelhado” mais 8 vezes até o endereço final (FFFFh)



FIM