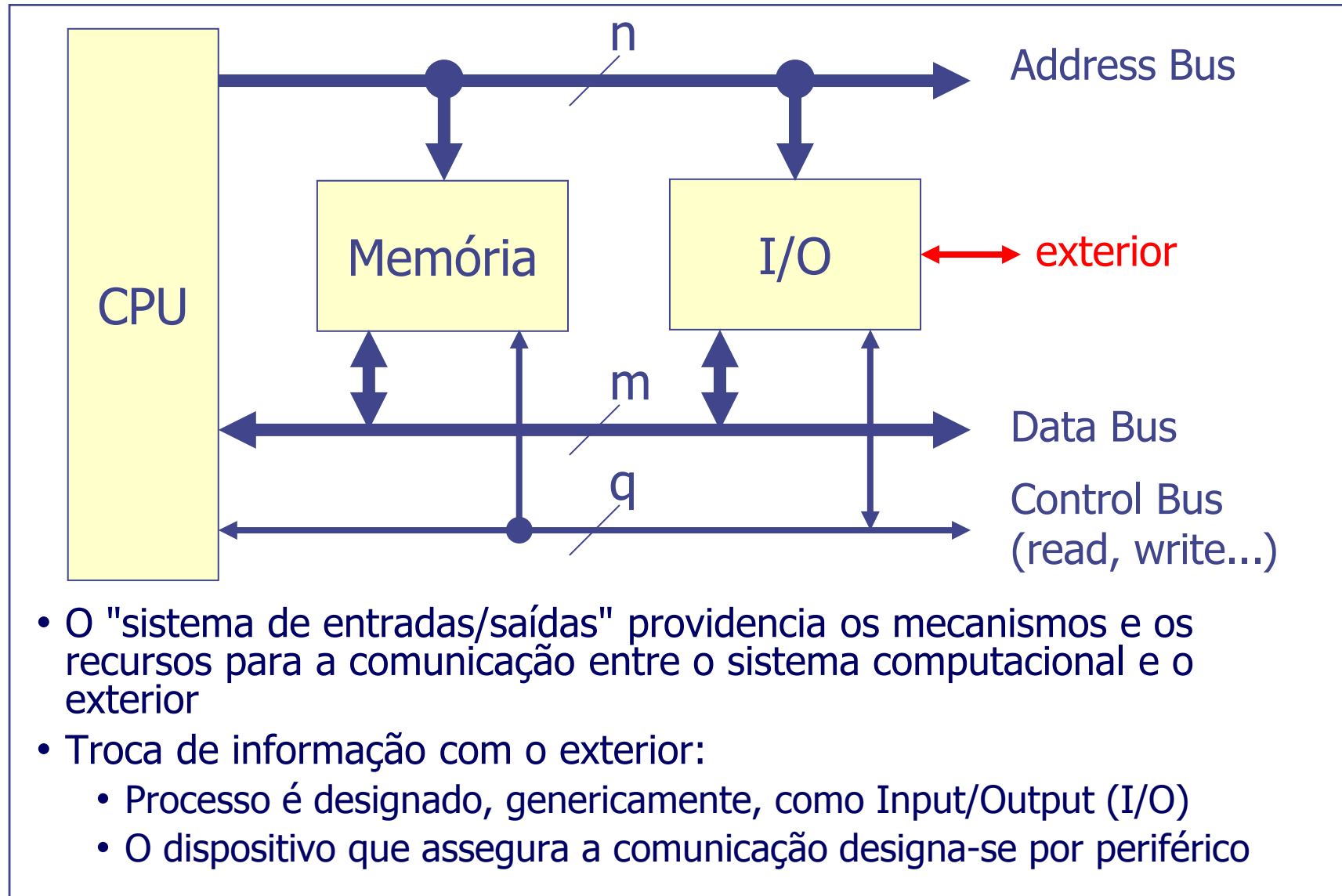


## Aulas 2 e 3

- Noção de periférico; estrutura básica de um módulo de I/O; modelo de programação
- Endereçamento das unidades de I/O
- Descodificação de endereços e geração de sinais de seleção de memória e unidades de I/O
- Mapeamento no espaço de endereçamento de memória
- Exemplo de um gerador de sinais de seleção programável

José Luís Azevedo, Bernardo Cunha, Tomás O. Silva, P. Bartolomeu

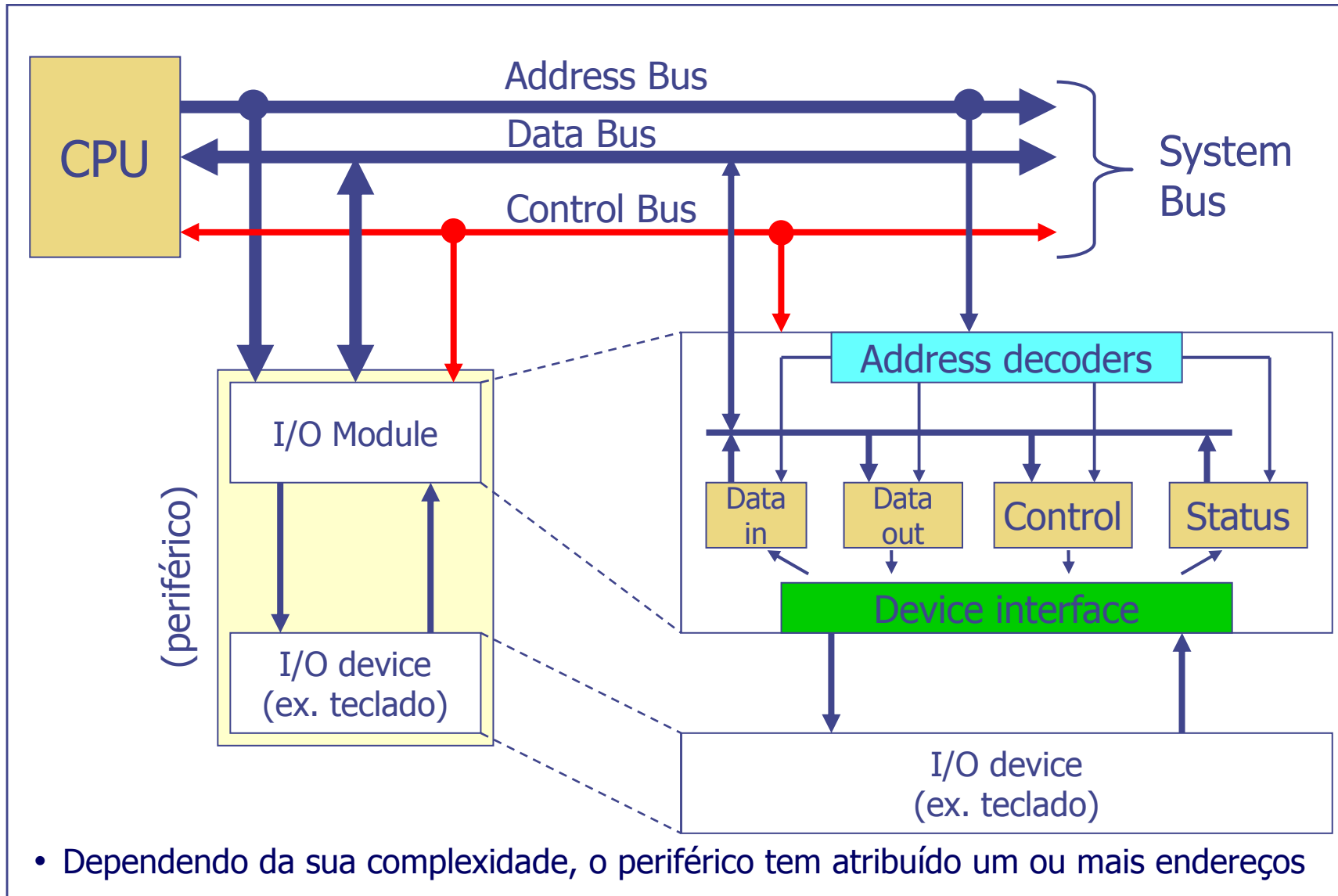
# Introdução



# Introdução

- Dispositivos periféricos:
  - grande variedade (por exemplo: teclado, rato, unidade de disco ...)
  - com métodos de operação diversos
  - assíncronos relativamente ao CPU
  - geram diferentes quantidades de informação com diferentes formatos a diferentes velocidades (de alguns bits/s a dezenas de Megabyte/s)
  - mais lentos que o CPU e a memória
- É necessária uma interface que providencie a adaptação entre as características intrínsecas do dispositivo periférico e as do CPU / memória
  - **Módulo de I/O**

# Módulo de I/O



# Módulo de I/O

- O módulo de I/O pode assim ser visto como um módulo de compatibilização entre as características e modo de funcionamento do sistema computacional e o dispositivo físico propriamente dito
- Ao nível do hardware:
  - Adequa as características do dispositivo físico de I/O às características do sistema digital ao qual tem que se ligar. O periférico liga-se ao sistema através dos barramentos, do mesmo modo que todos os outros dispositivos (ex. memória)
- Na interação com o dispositivo físico:
  - Lida com as particularidades do dispositivo, nomeadamente, formatação de dados, deteção e gestão de situações de erro, ...
- Ao nível do software:
  - Adequa o dispositivo físico à forma de organização do sistema computacional, disponibilizando e recebendo informação através de registos; esta solução esconde do programador a complexidade e os detalhes de implementação do dispositivo periférico

# Módulo de I/O

- O módulo de I/O permite ao processador ver um modelo simplificado do periférico, escondendo os detalhes de funcionamento interno
- Com a adoção do módulo de I/O, o dispositivo periférico, independentemente da sua natureza e função, passa a ser encarado pelo processador como uma coleção de registos de dados, de controlo e de *status*
- A comunicação entre o processador e o periférico é assegurada por operações de escrita e de leitura, em tudo semelhantes a um acesso a uma posição de memória
  - Ao contrário do que acontece na memória, o valor associado a estes endereços pode mudar sem intervenção do CPU
- O conjunto de registos e a descrição de cada um deles são específicos para cada periférico e constituem o que se designa por **modelo de programação do periférico**

# Módulo de I/O – modelo de programação

- **Data Register(s)** (*Read/Write*)
  - Registo(s) onde o processador coloca a informação a ser enviada para o periférico (*write*) e de onde lê informação proveniente do periférico (*read*)
- **Status Register(s)** (*Read only*)
  - Registo(s) que engloba(m) um conjunto de bits que dão informação sobre o estado do periférico (ex. operação terminada, informação disponível, situação de erro, ...)
- **Control Register(s)** (*Write only* ou *Read/Write*)
  - Registo(s) onde o CPU escreve informação sobre o modo de operação do periférico (comandos)
- É comum um só registo incluir as funções de controlo e de *status*. Nesse caso, um conjunto de bits desse registo está associado a funções de controlo (*read/write* ou *write only bits*) e outro conjunto a funções de status (*read only bits*)

# Comunicação entre o CPU e outros dispositivos

- A iniciativa da comunicação é sempre do CPU, no contexto da execução das instruções
- A comunicação entre o CPU e um dispositivo genérico envolve a existência de um **protocolo** que ambas as partes conhecem e respeitam
- Apenas duas operações podem ser efetuadas no sistema:
  - **Write** (fluxo de informação: CPU → dispositivo externo)
  - **Read** (fluxo de informação: CPU ← dispositivo externo)
- Uma operação de acesso do CPU a um dispositivo externo envolve:
  - Usar o barramento de endereços para especificar o **endereço do dispositivo a aceder**
  - Usar o barramento de controlo para sinalizar qual a operação a realizar (*read* ou *write*)
  - O barramento de dados assegura a transferência de dados, no sentido adequado, entre as duas entidades envolvidas na comunicação

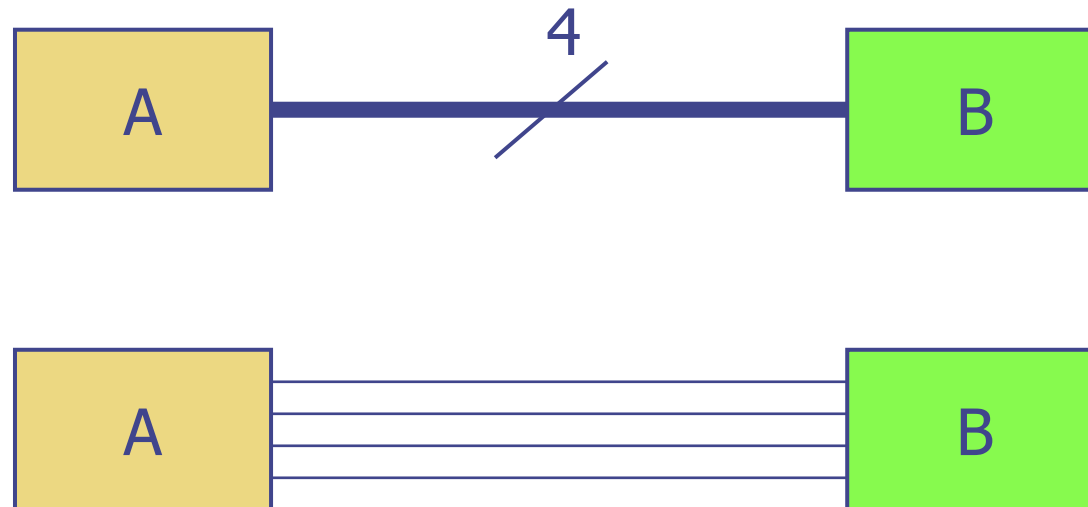


# Seleção do dispositivo externo

- **Operação de escrita** (CPU → dispositivo externo)
  - apenas 1 dispositivo deve receber a informação colocada pelo CPU no barramento de dados
- **Operação de leitura** (CPU ← dispositivo externo)
  - apenas 1 dispositivo pode estar ativo no barramento de dados
  - os dispositivos, quando inativos, têm que estar eletricamente desligados do barramento de dados
  - é obrigatório utilizar portas **Tri-State** na ligação do dispositivo ao barramento de dados
- Num sistema computacional há múltiplos circuitos ligados ao barramento de dados
  - unidades de I/O
  - circuitos de memória
- Há, pois, necessidade de, a partir do endereço gerado pelo CPU, **selecionar apenas um** dos vários dispositivos existentes no sistema

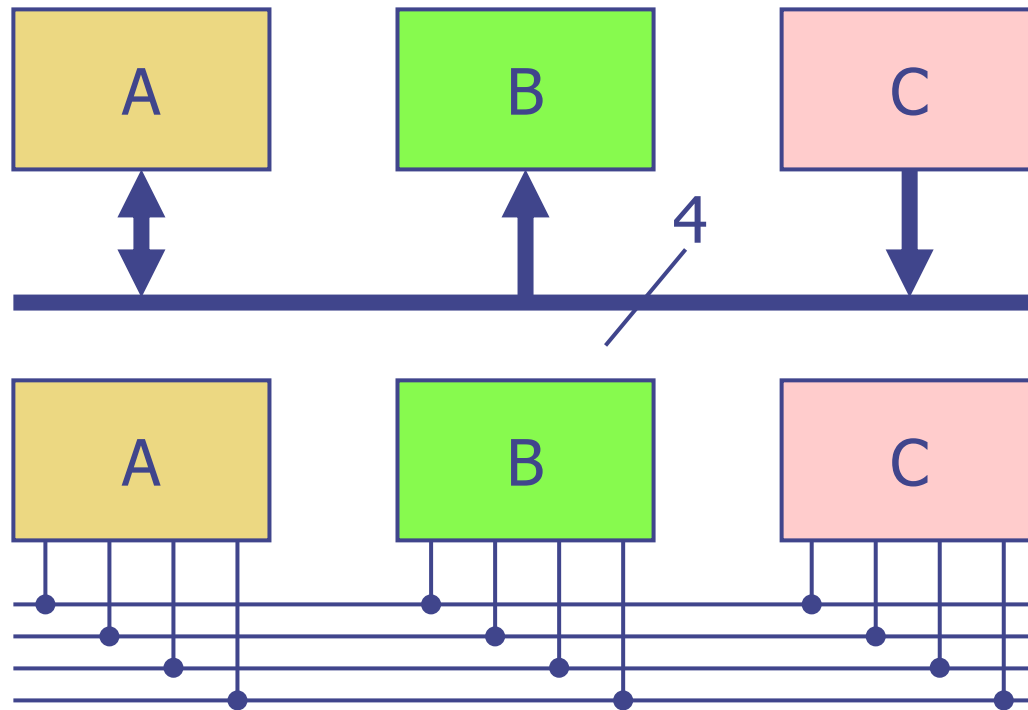
# Barramento simples (revisão)

- Barramento (bus) - conjunto de ligações (fios) agrupadas, geralmente, segundo uma dada função; cada ligação transporta informação relativa a 1 bit.
- Exemplo – barramento de 4 bits que liga os dispositivos A e B



# Barramento partilhado (revisão)

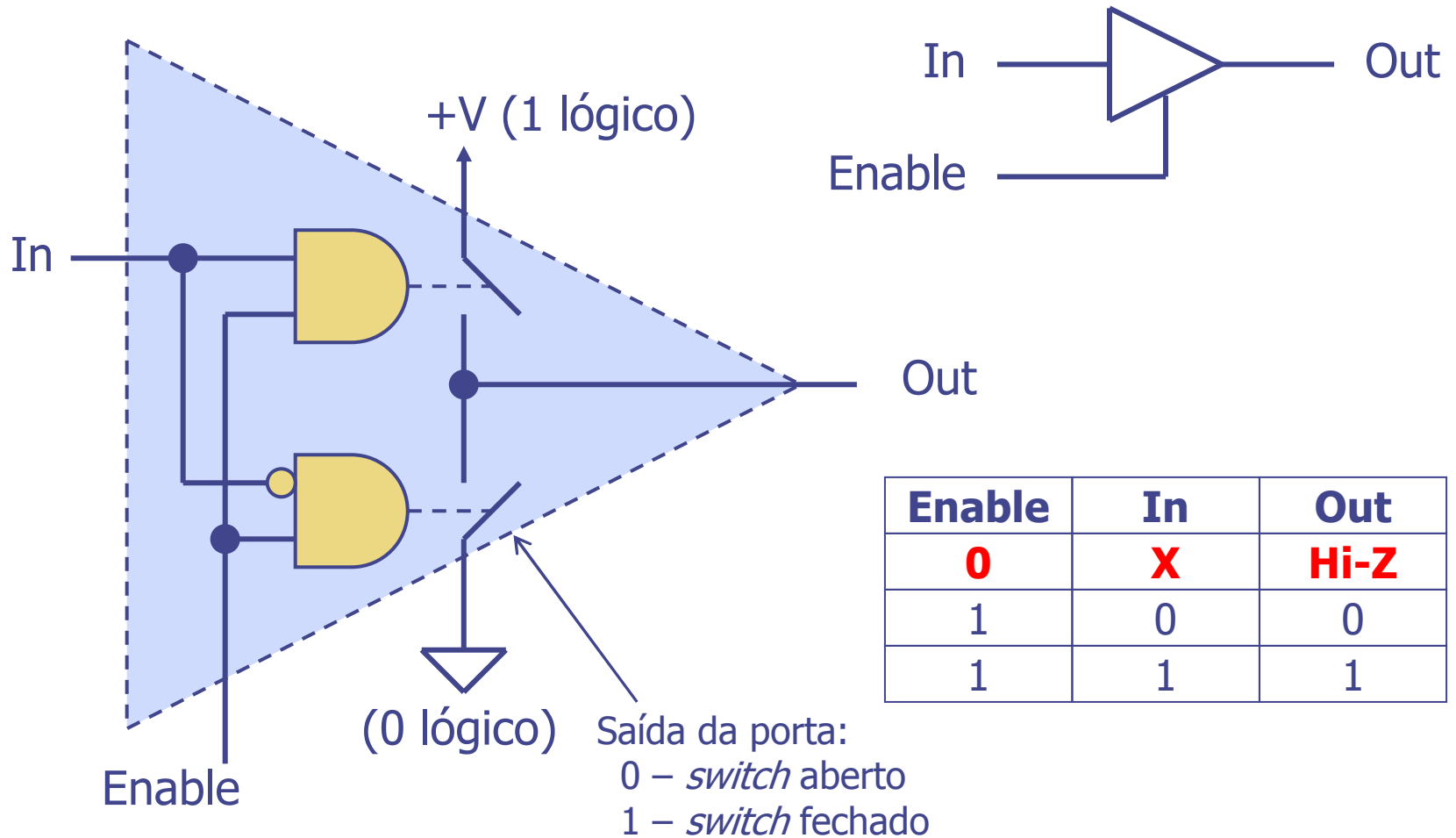
- Barramento partilhado (*shared bus*) - barramento que interliga vários dispositivos
- Exemplo: barramento de interligação entre os dispositivos A, B e C



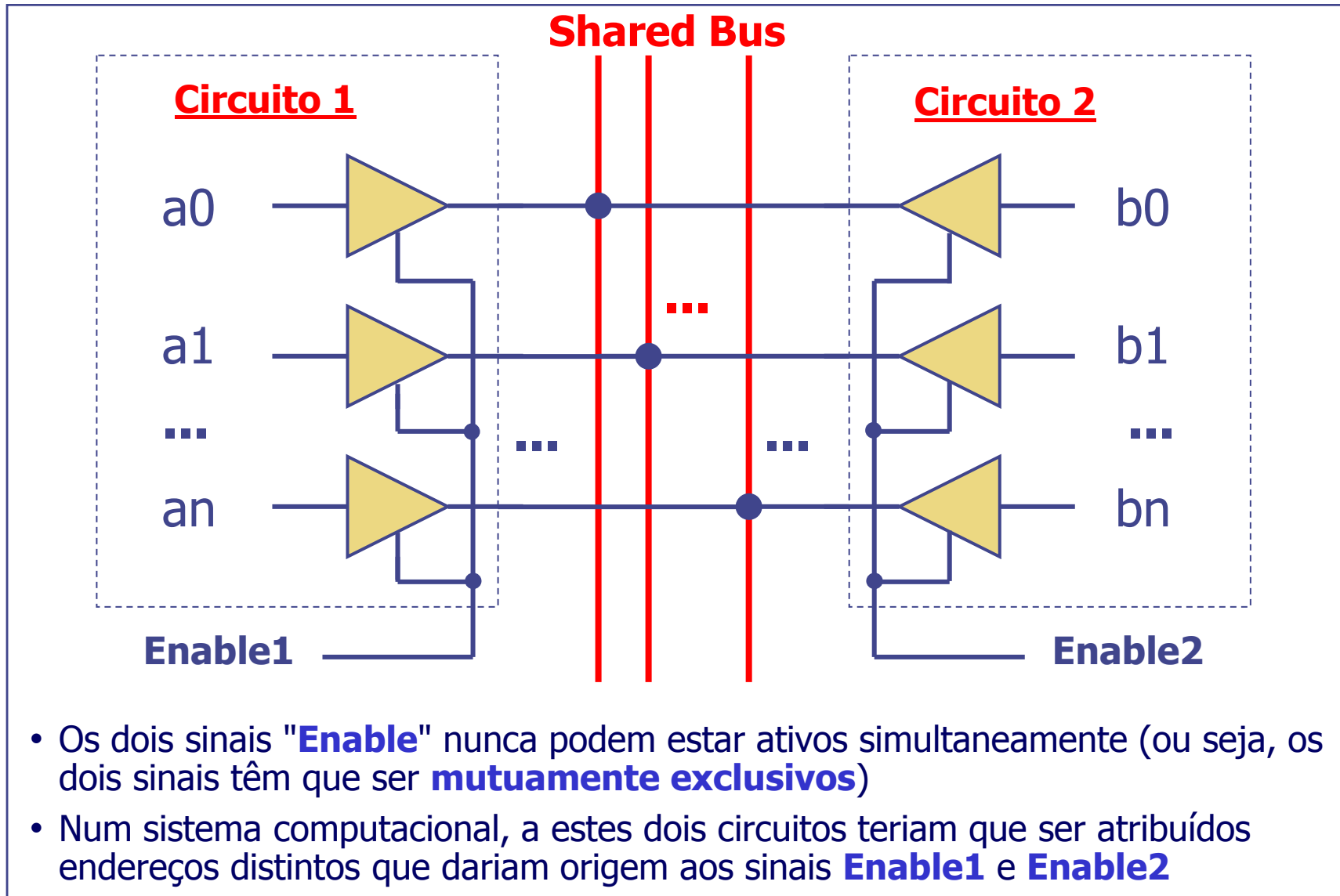
- A comunicação pode realizar-se de A para B, de C para B ou de C para A

# Porta *Tri-State*

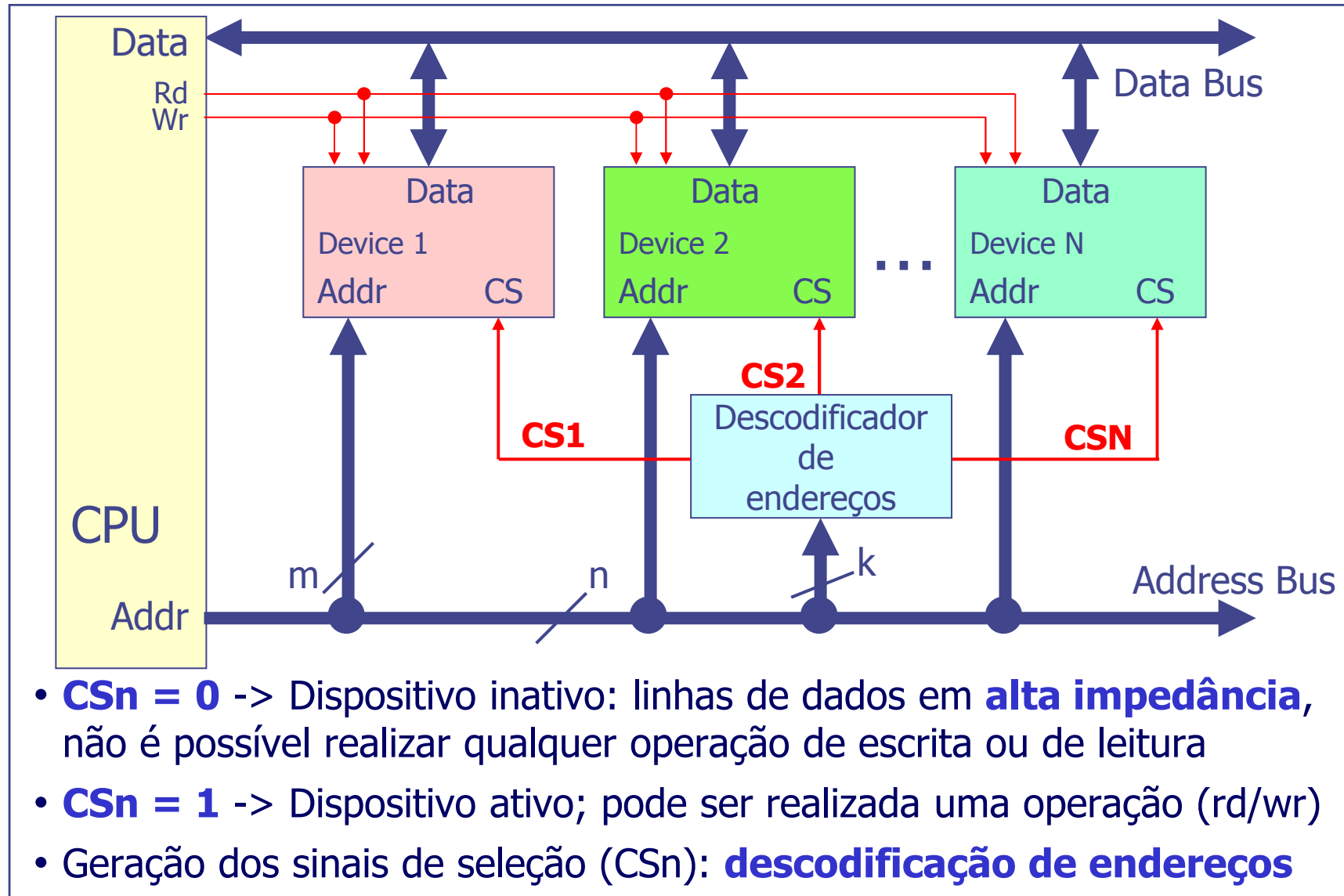
- Modelo de funcionamento de uma porta *Tri-State*



# Ligação a um barramento partilhado



# Seleção do dispositivo externo



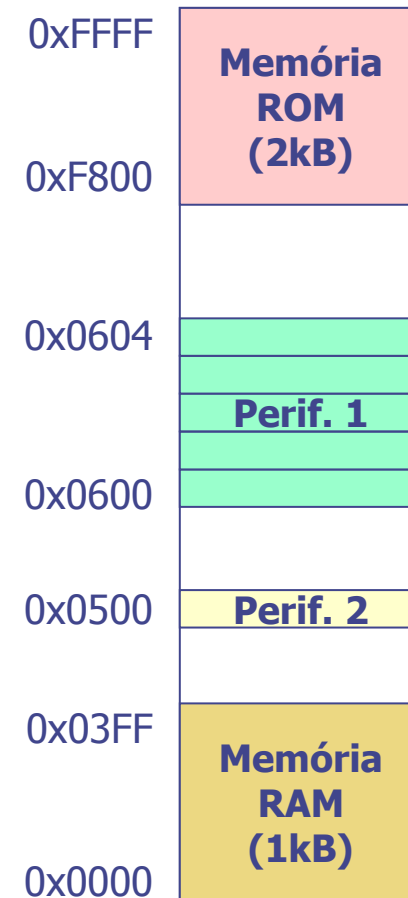
# Seleção do dispositivo externo

- Para ser possível o acesso a todos os recursos disponíveis, o dispositivo externo pode necessitar de apenas um endereço ou de uma gama contígua de endereços
- Exemplos (supondo uma organização de memória do tipo *byte-addressable*) :
  - Para aceder a todas as posições de uma memória de 1kB são necessários 1024 endereços consecutivos (10 bits do barramento de endereços)
  - Para ser possível o acesso aos 5 registos (de 1 byte cada) de um periférico são necessários 5 endereços consecutivos (3 bits do barramento de endereços)
  - Para aceder a um porto de saída de 1 byte (por exemplo implementado como um registo de 8 bits) será apenas necessário 1 endereço
- A implementação do descodificador de endereços é feita a partir de um mapa de endereços que mapeia no espaço de endereçamento do processador a gama de endereços necessária para cada dispositivo do sistema

# Mapeamento no espaço de endereçamento

- Exemplo de mapeamento de dispositivos, considerando um espaço de endereçamento de 16 bits ( $2^{16} = 64k$ ,  $A_{15}-A_0$ ), e uma organização do tipo *byte-addressable*:
  - memória RAM de 1k x 8 (1 kB), memória ROM de 2k x 8 (2 kB), periférico1 com 5 registos de 1, byte, periférico2 com 1 registo de 1 byte
- Possível mapa de endereços:

Dispositivo	Dimensão (bytes)	Endereço Inicial	Endereço Final	Nº bits do <i>address bus</i>
RAM, 1k x 8	1024	0x0000	0x03FF	10
ROM, 2k x 8	2048	0xF800	0xFFFF	11
Periférico 1	5	0x0600	0x0604	3
Periférico 2	1	0x0500	0x0500	0

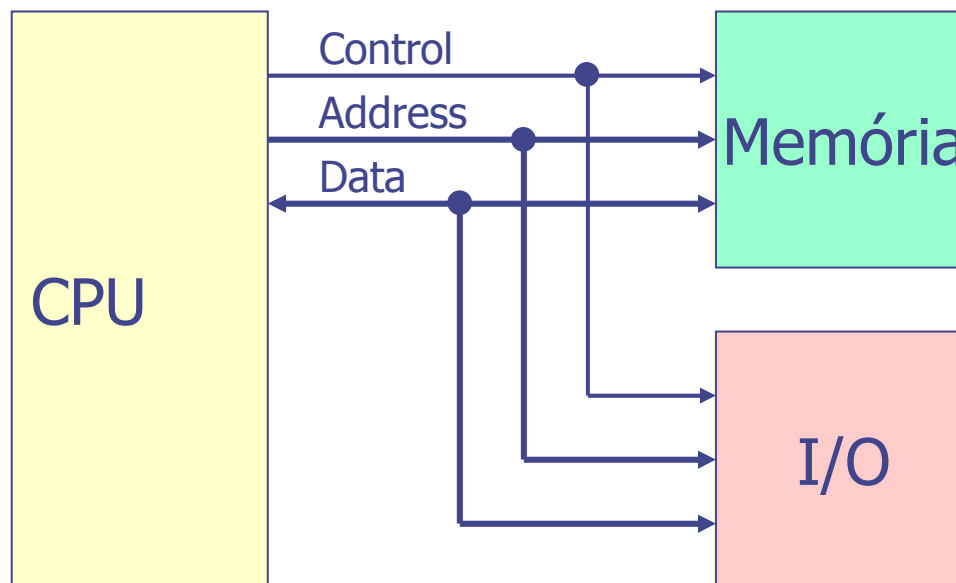


Se os registos dos periféricos fossem de 32 bits, quais seriam as gamas de endereços necessárias ?



# Endereçamento das unidades de I/O

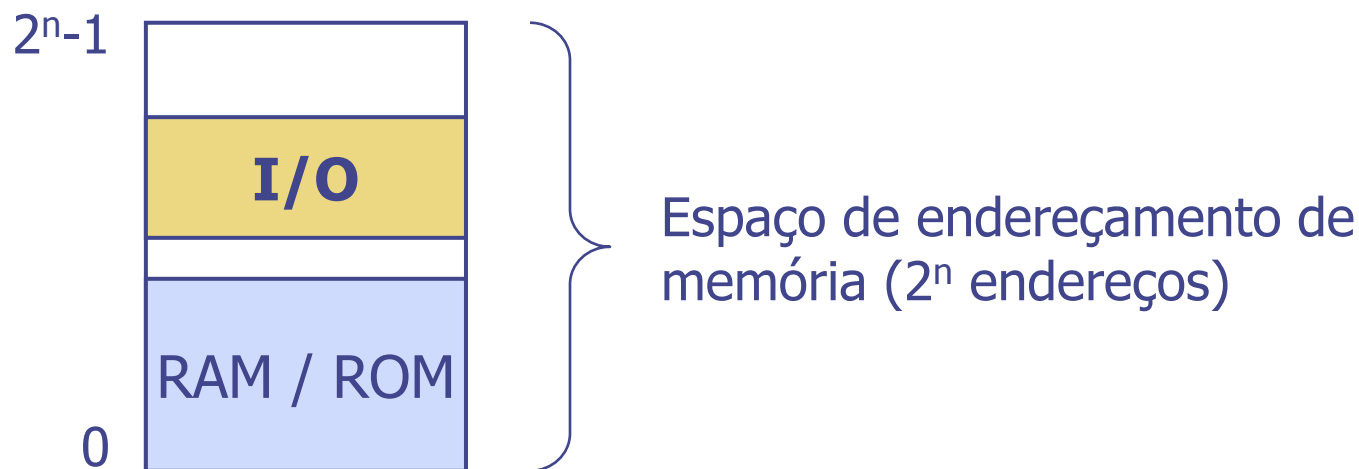
- *Memory-mapped I/O*



- Memória e unidades de I/O coabitam no mesmo espaço de endereçamento
- Uma parte do espaço de endereçamento é reservada para periféricos

# Endereçamento das unidades de I/O

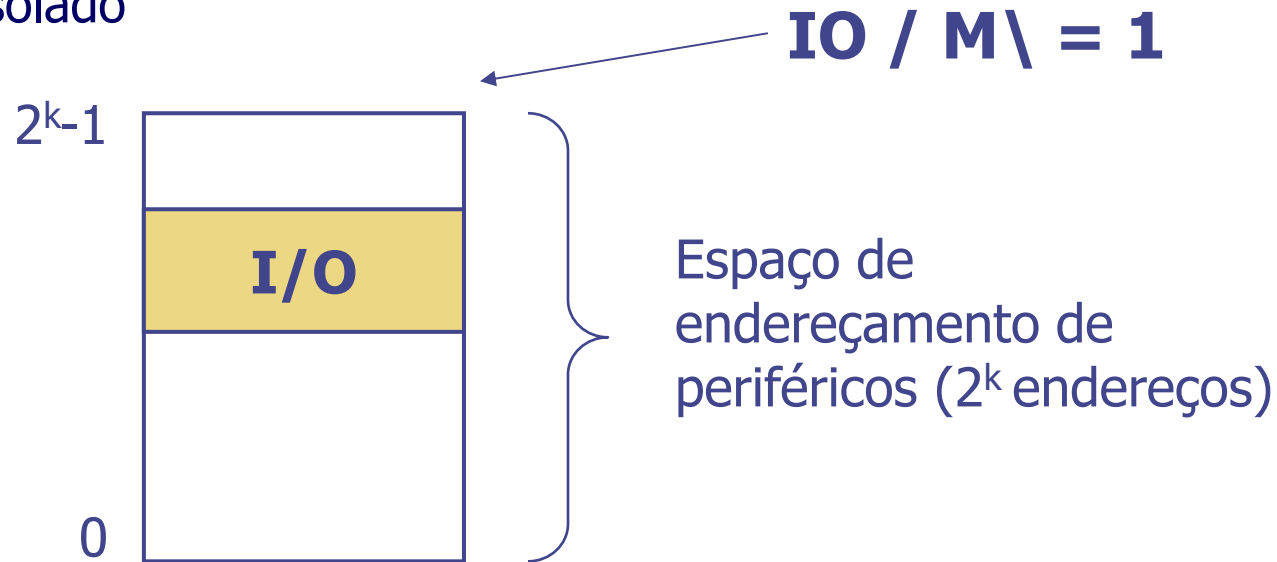
- *Memory-mapped I/O*



- Às unidades de I/O são atribuídos endereços do espaço de endereçamento de memória
- O acesso às unidades de I/O é feito com as mesmas instruções com que se acede à memória (**lw** e **sw** no caso do MIPS)

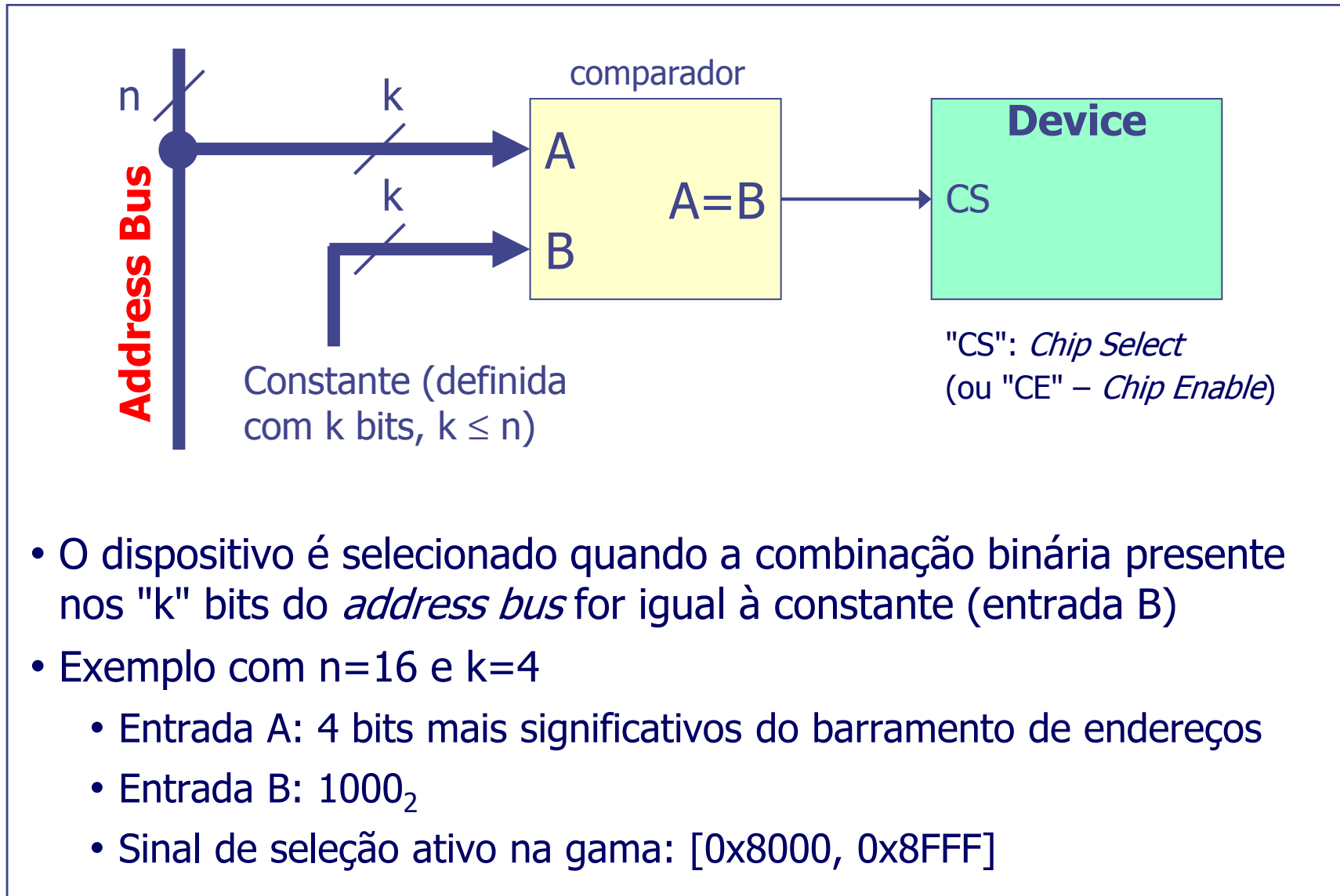
# Endereçamento das unidades de I/O

- I/O Isolado



- Memória e periféricos em espaços de endereçamento separados
- Sinal do barramento de controlo indica a qual dos espaços de endereçamento (I/O ou memória) se destina o acesso; por exemplo  $\mathbf{IO/M\}$ :
  - $\mathbf{IO/M\ = 1}$  -> acesso ao espaço de endereçamento de I/O
  - $\mathbf{IO/M\ = 0}$  -> acesso ao espaço de endereçamento de memória
- O acesso às unidades de I/O é feito com instruções específicas

# Descodificação de endereços

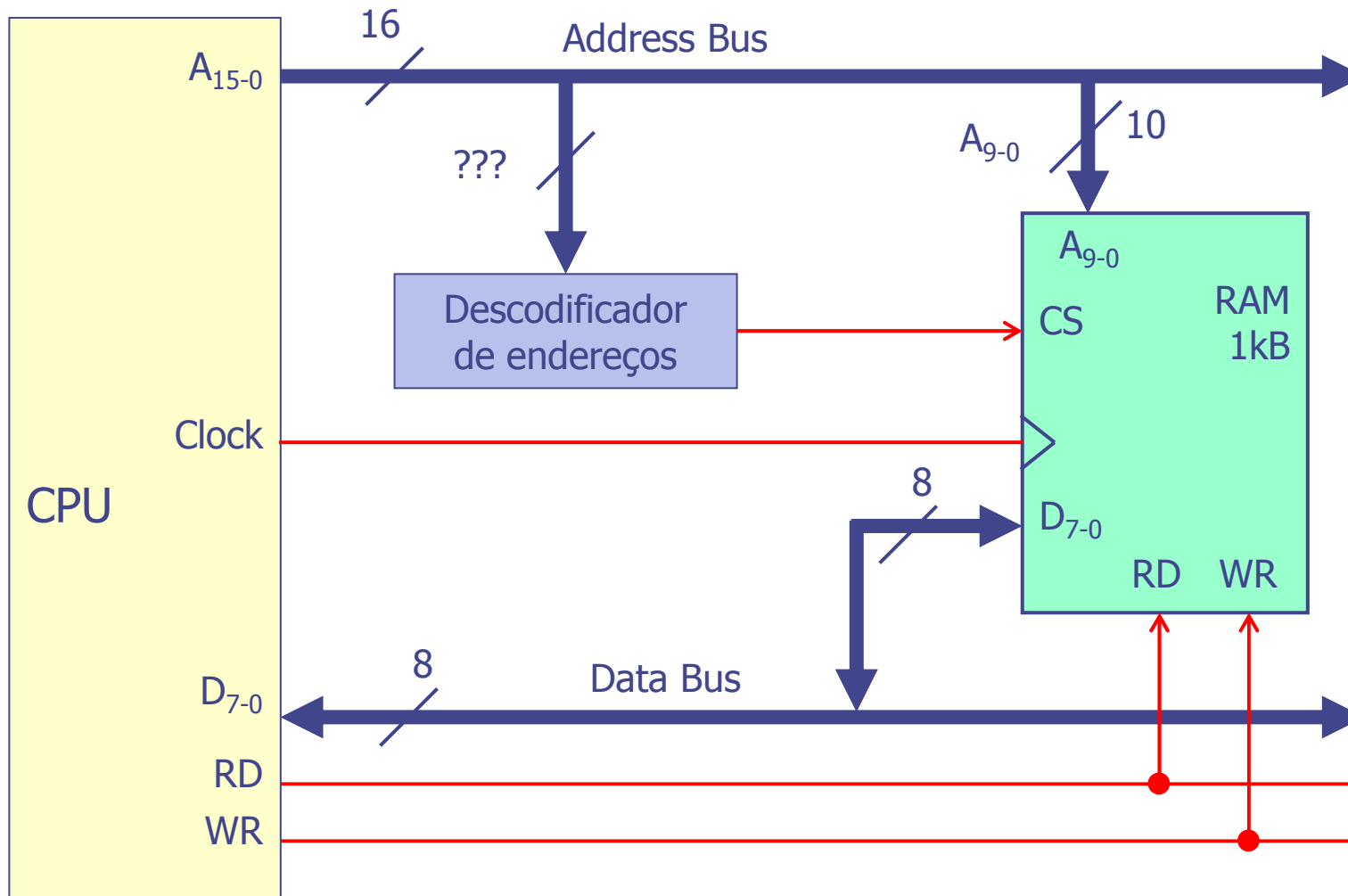


# Descodificação de endereços

- Supondo um CPU com um espaço de endereçamento de 16 bits, memória *byte-addressable*, e um barramento de dados de 8 bits
  - 16 bits ( $A_{15}-A_0$ )  $\rightarrow$  ( $2^{16}=64$  k)
  - 8 bits ( $D_7-D_0$ )
- **Exemplo 1:** ligação de uma memória RAM de 1 kByte ao CPU
  - 1 kByte (1k x 8) - 10 bits de endereço ( $2^{10}=1$ k)
- **Exemplo 2:** ligação de um porto de saída de 1 byte ao CPU

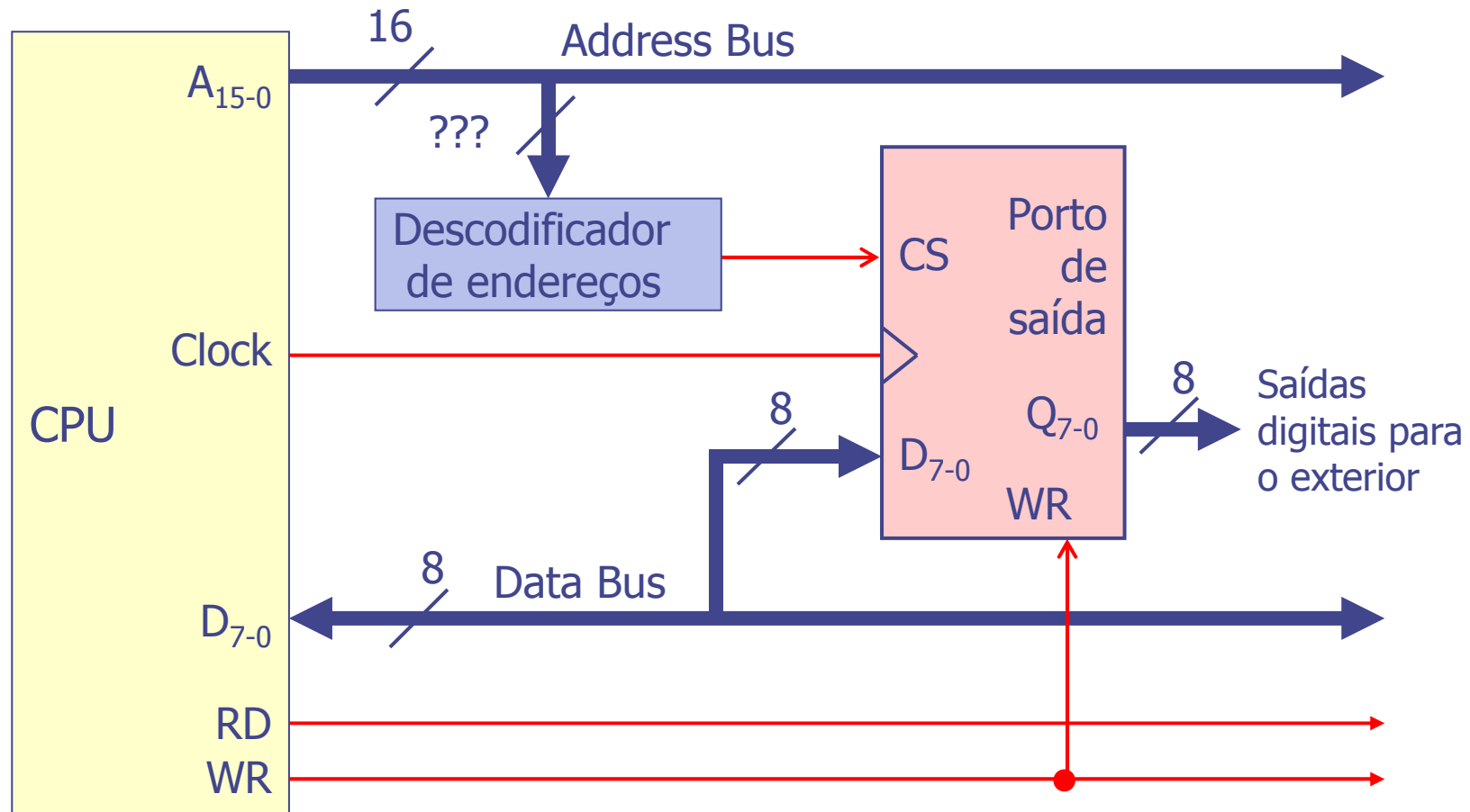
# Descodificação de endereços

- Exemplo 1: ligação de uma memória RAM de 1 kByte ao CPU



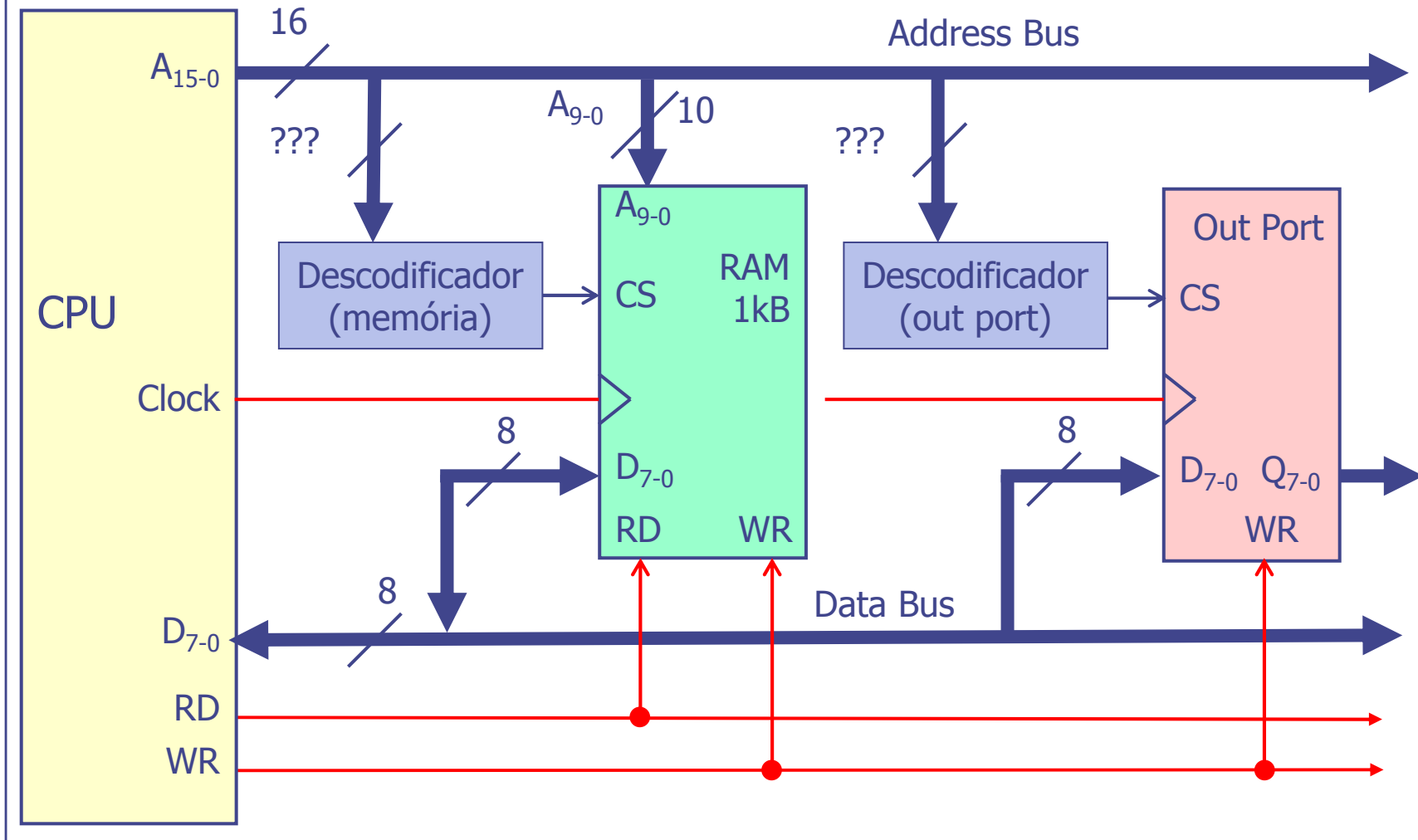
# Descodificação de endereços

- Exemplo 2: ligação de um porto de saída de 1 byte ao CPU



# Descodificação de endereços

- Ligação do porto de saída e da memória





# Descodificação de endereços

- Descodificação total
  - Para uma dada posição de memória / registo de periférico existe apenas um endereço possível para acesso
  - Todos os bits relevantes são descodificados
- Descodificação parcial
  - Vários endereços possíveis para aceder à **mesma posição de memória/registo de um periférico**
  - Apenas alguns bits são descodificados
  - Conduz a circuitos de descodificação mais simples (e menores atrasos)

# Descodificação de endereços

- Mapa de endereços, num espaço de endereçamento de 16 bits (para o exemplo dos slides anteriores):

<b>Dispositivo</b>	<b>Dimensão (bytes)</b>	<b>Endereço Inicial</b>	<b>Endereço Final</b>	<b>Nº bits do <i>address bus</i></b>
RAM, 1k x 8	1024	0x0000	0x03FF	10
Porto de saída	1	0x4100	0x4100	0

- Descodificador de endereços do porto de saída
  - Quais os bits a usar no descodificador de endereços?
- Descodificador de endereços da memória RAM
  - Quais os bits a usar no descodificador de endereços?

# Descodificação de endereços

Dispositivo	Dimensão (bytes)	Endereço Inicial	Endereço Final	Nº bits do <i>address bus</i>
RAM, 1k x 8	1024	0x0000	0x03FF	10
Porto de saída	1	0x4100	0x4100	0

- Porto de saída – **descodificação total:**

$$0x4100 = 0100\ 0001\ 0000\ 0000$$

$$A_n \setminus \Leftrightarrow \overline{A_n}$$

$$CS = A_{15} \setminus .A_{14} .A_{13} \setminus .A_{12} \setminus .A_{11} \setminus .A_{10} \setminus .A_9 \setminus .A_8 .A_7 \setminus .A_6 \setminus .A_5 \setminus .A_4 \setminus .A_3 \setminus .A_2 \setminus .A_1 \setminus .A_0 \setminus$$

- Porto de saída – **descodificação parcial:**

- Não usar, por exemplo, os dois bits menos significativos

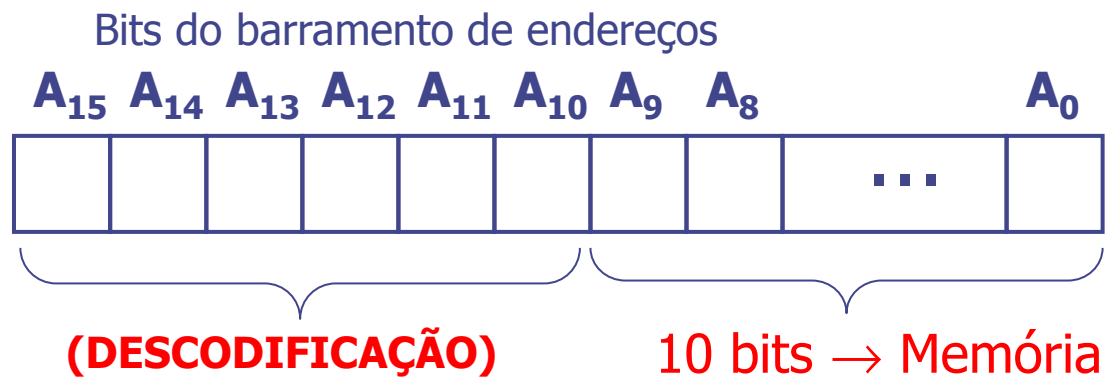
$$CS = A_{15} \setminus .A_{14} .A_{13} \setminus .A_{12} \setminus .A_{11} \setminus .A_{10} \setminus .A_9 \setminus .A_8 .A_7 \setminus .A_6 \setminus .A_5 \setminus .A_4 \setminus .A_3 \setminus .A_2 \setminus$$

- Gama de ativação do CS: [0x4100, 0x4103]

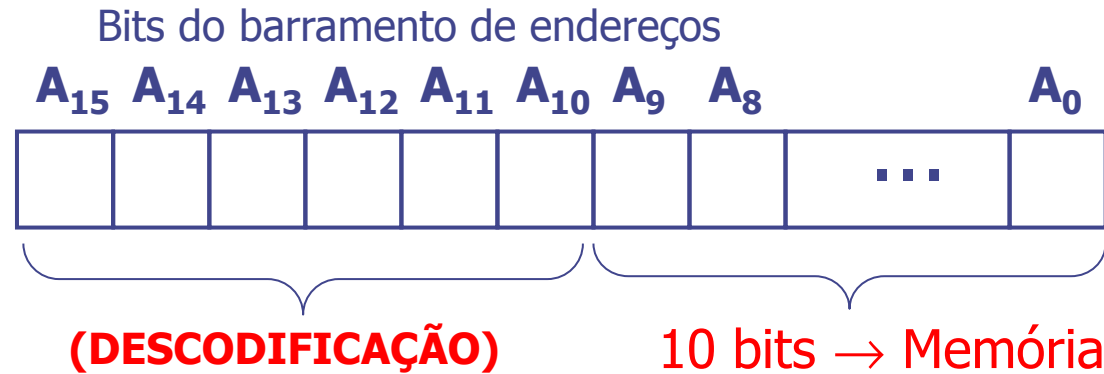
# Descodificação de endereços

Dispositivo	Dimensão (bytes)	Endereço Inicial	Endereço Final	Nº bits do <i>address bus</i>
RAM, 1k x 8	1024	0x0000	0x03FF	10
Porto de saída	1	0x4100	0x4100	0

- Memória RAM



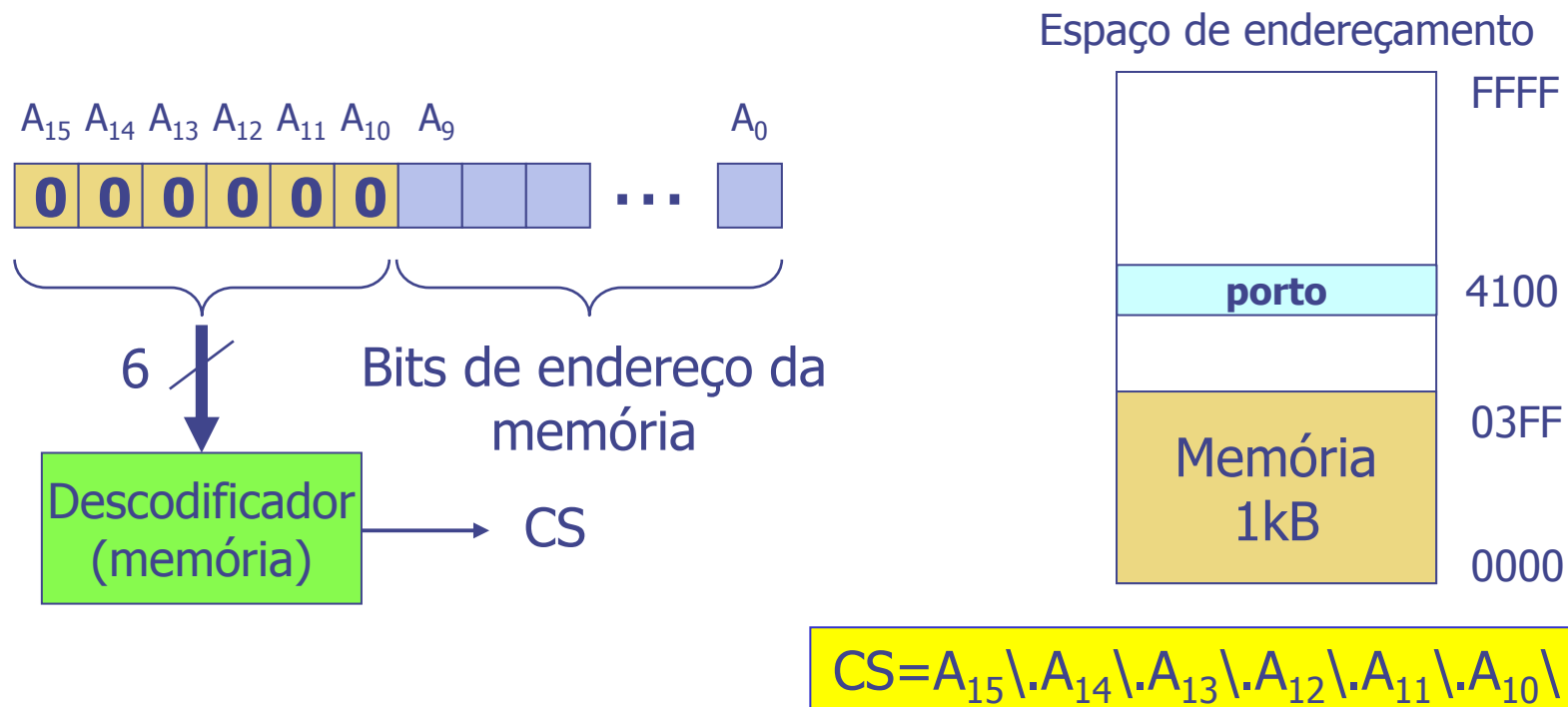
# Descodificação de endereços



- Para garantir que a memória de 1kB está mapeada a partir do endereço 0x0000 (na gama 0x0000-0x03FF), há várias soluções possíveis; vamos analisar as seguintes 3:
  - 1) descodificação total** – usar os 6 bits A15 a A10 (e.g. **000000**)
  - 2) descodificação parcial** – usar A15, A14, A13 e A12 e ignorar A11 e A10 (e.g. **0000xx**)
  - 3) descodificação parcial** – usar apenas A13, A12, A11 e A10 e ignorar A15 e A14 (e.g. **xx0000**)
- Que implicações têm estas escolhas? Quais garantem zonas de endereçamento exclusivas para o porto de saída e para a memória?

# Descodificação de endereços – descodificação total

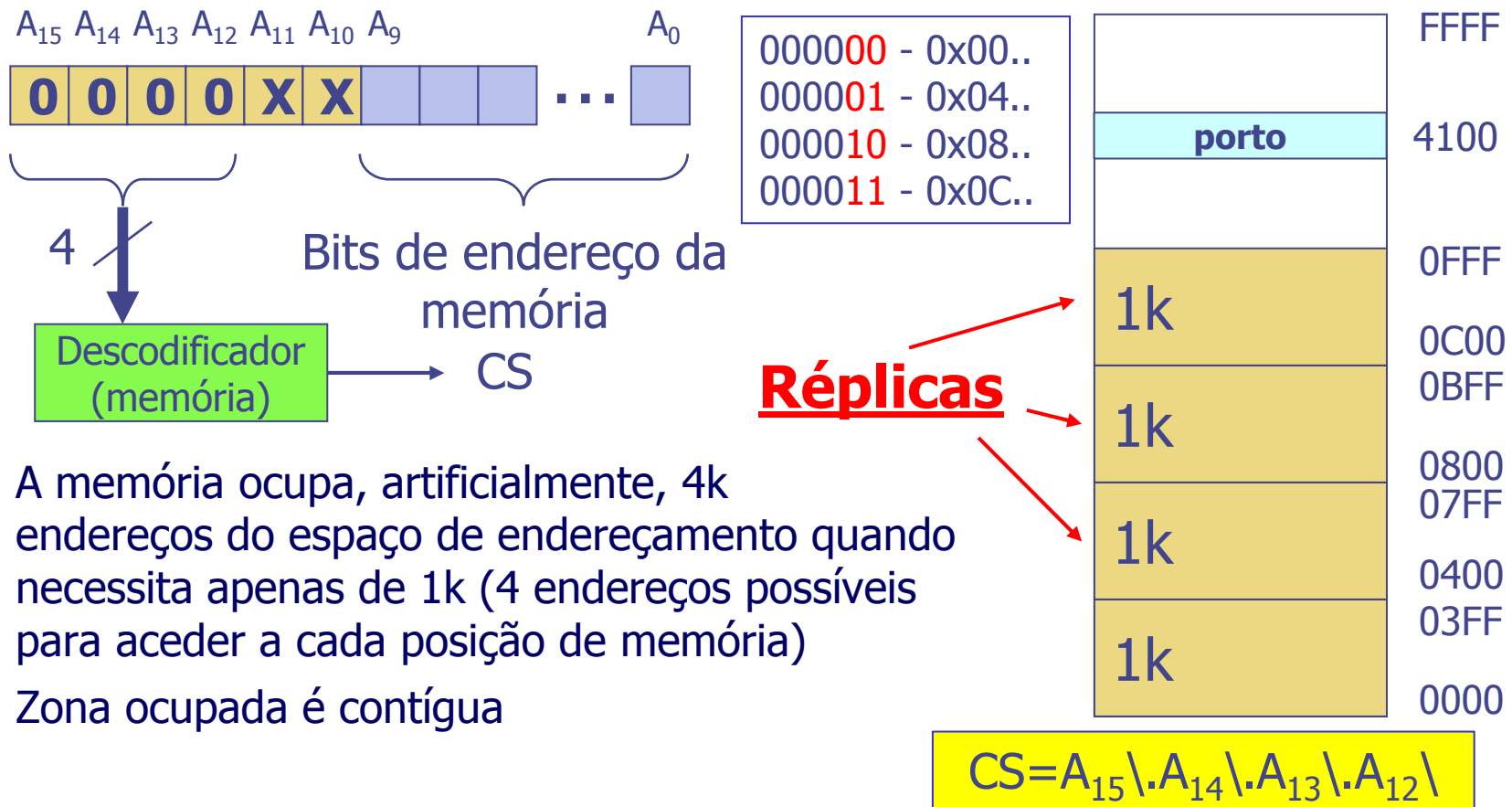
- Solução 1 – utilizar todos os bits possíveis, e.g. **000000**. Isto significa que um endereço só é válido para aceder à memória se tiver os 6 bits mais significativos a 0



- A memória ocupa 1k do espaço de endereçamento
- Apenas 1 endereço possível para aceder a cada posição de memória

# Descodificação de endereços – descodificação parcial

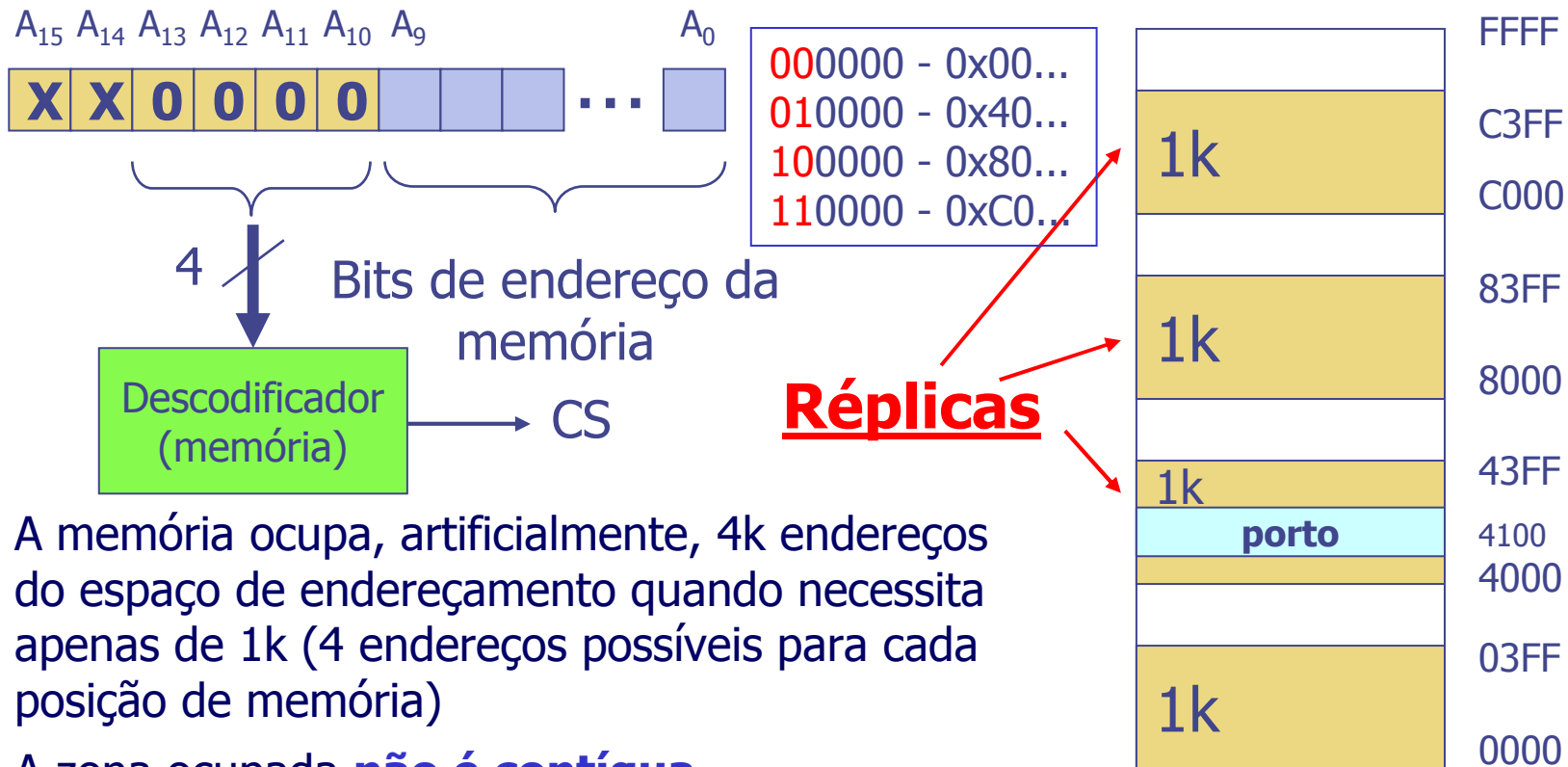
- Solução 2 – usar A15, A14, A13 e A12 e ignorar A11 e A10, e.g. **0000xx**. Isto significa que um endereço válido para aceder à memória não depende do valor dos bits A11 e A10, mas tem que ter os bits A15 a A12 a 0.



- A memória ocupa, artificialmente, 4k endereços do espaço de endereçamento quando necessita apenas de 1k (4 endereços possíveis para aceder a cada posição de memória)
- Zona ocupada é contígua

# Descodificação de endereços – descodificação parcial

- Solução 3 – usar A13, A12, A11 e A10 e ignorar A15 e A14, e.g. **xx0000**. Isto significa que um endereço válido para aceder à memória não depende do valor dos bits A15 e A14, mas tem que ter os bits A13 a A10 a 0



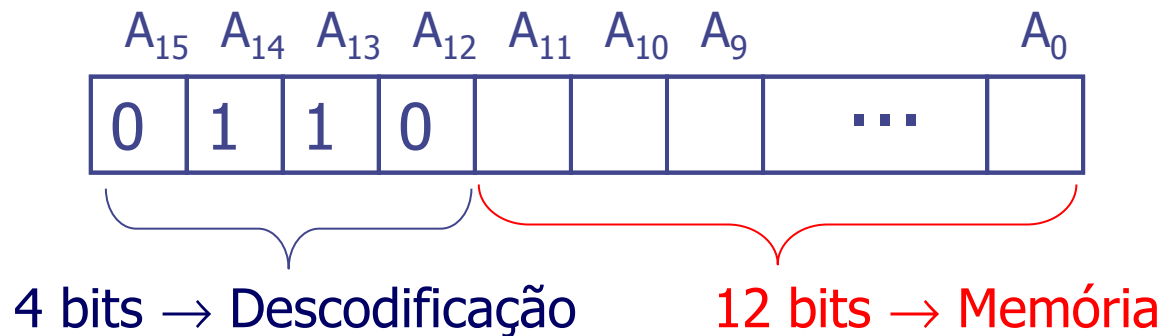
- A memória ocupa, artificialmente, 4k endereços do espaço de endereçamento quando necessita apenas de 1k (4 endereços possíveis para cada posição de memória)
- A zona ocupada **não é contígua**
- Conflito com o endereço do porto (0x4100)



## Descodificação de endereços – exercício

- Escrever a equação lógica do descodificador de endereços para uma memória de 4 kByte, mapeada num espaço de endereçamento de 16 bits, que respeite os seguintes requisitos:
  - Endereço inicial: 0x6000; descodificação total.

$$4 \text{ kByte} = 2^{12} \quad (2^{12} - 1 = 0x0FFF)$$



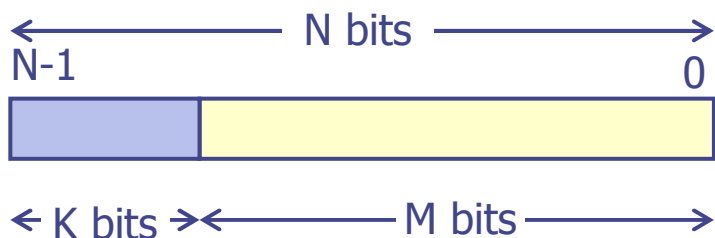
0110**000000000000** (0x6000)

0110**111111111111** (0x6FFF)

- Lógica positiva:  $CS = A_{15} \cdot A_{14} \cdot A_{13} \cdot A_{12}$
- Lógica negativa:  $CS \setminus = A_{15} \setminus + A_{14} \setminus + A_{13} \setminus + A_{12}$

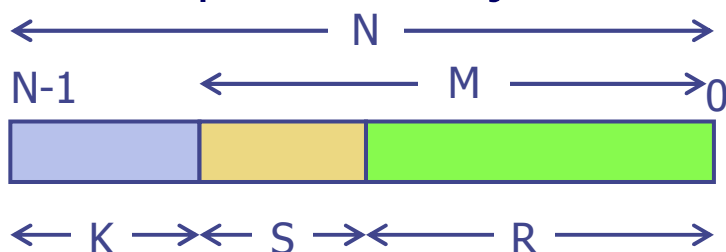
# Gerador de sinais de seleção programável

- Como se viu anteriormente, os N bits do espaço de endereçamento podem, para efeitos de descodificação de endereços e endereçamento, ser divididos em dois grupos: M bits usados para endereçamento dentro da gama descodificada e os restantes K bits usados para descodificação



- Dimensão da gama descodificada:  $2^M$
- Endereço inicial da gama descodificada é definida pela combinação binária dos K bits
- Número de gamas que podem ser descodificadas:  $2^K$

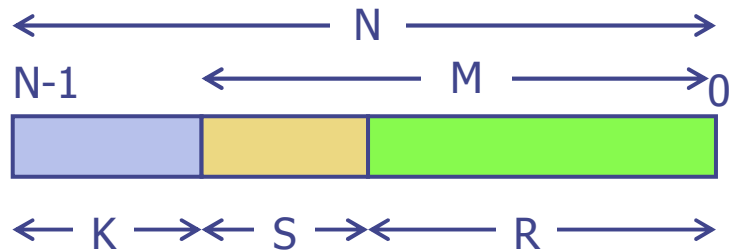
- O mesmo método pode ser aplicado para a sub-divisão dos M bits da gama descodificada: S bits usados para descodificação, R bits usados para endereçamento



- Número de sub-gamas que podem ser descodificadas:  $2^S$
- Dimensão da sub-gama descodificada:  $2^R$
- Endereço inicial da sub-gama descodificada é definido pelo conjunto dos bits K e S

# Gerador de sinais de seleção programável - exemplo

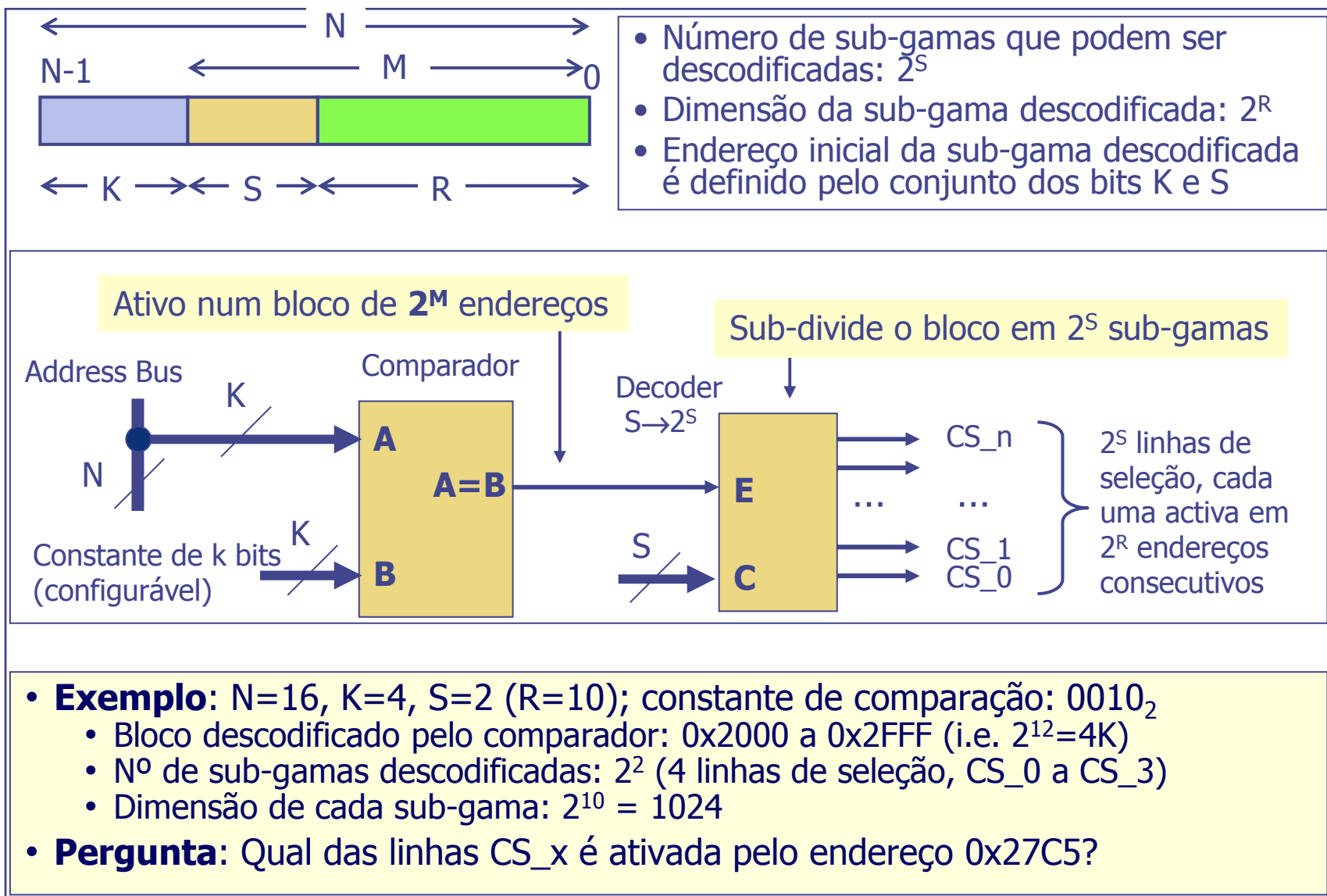
- Exemplo para um espaço de endereçamento de 8 bits ( $N=8$ )



<b>01000000</b> – <b>01000111</b>	: 0x40 – 0x47
<b>01001000</b> – <b>01001111</b>	: 0x48 – 0x4F
<b>01010000</b> – <b>01010111</b>	: 0x50 – 0x57
<b>01011000</b> – <b>01011111</b>	: 0x58 – 0x5F

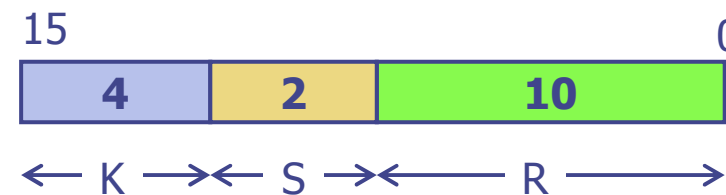
- $M=5$  ( $K=3$ ),  $S=2$  e  $R=3$ 
  - $N = 8$  – espaço de endereçamento com  $2^8 = 256$  endereços
  - $M = 5$  – gama descodificada com  $2^5 = 32$  endereços
  - $S = 2$  – número de sub-gamas  $2^2 = 4$
  - $R = 3$  – dimensão da sub-gama:  $2^3 = 8$  endereços
- A gama de  $2^5$  endereços foi sub-dividida em  $2^2$  gamas iguais, de  $2^3$  endereços cada
- O endereço inicial do bloco de  $2^2$  gamas é definido pela combinação binária usada nos  $K$  bits. Ex: **010** -> endereço inicial = **0x40**
  - gama0: 0x40 – 0x47, gama1: 0x48 – 0x4F, gama2: 0x50 – 0x57, gama3: 0x58 – 0x5F

# Gerador de sinais de seleção programável - implementação



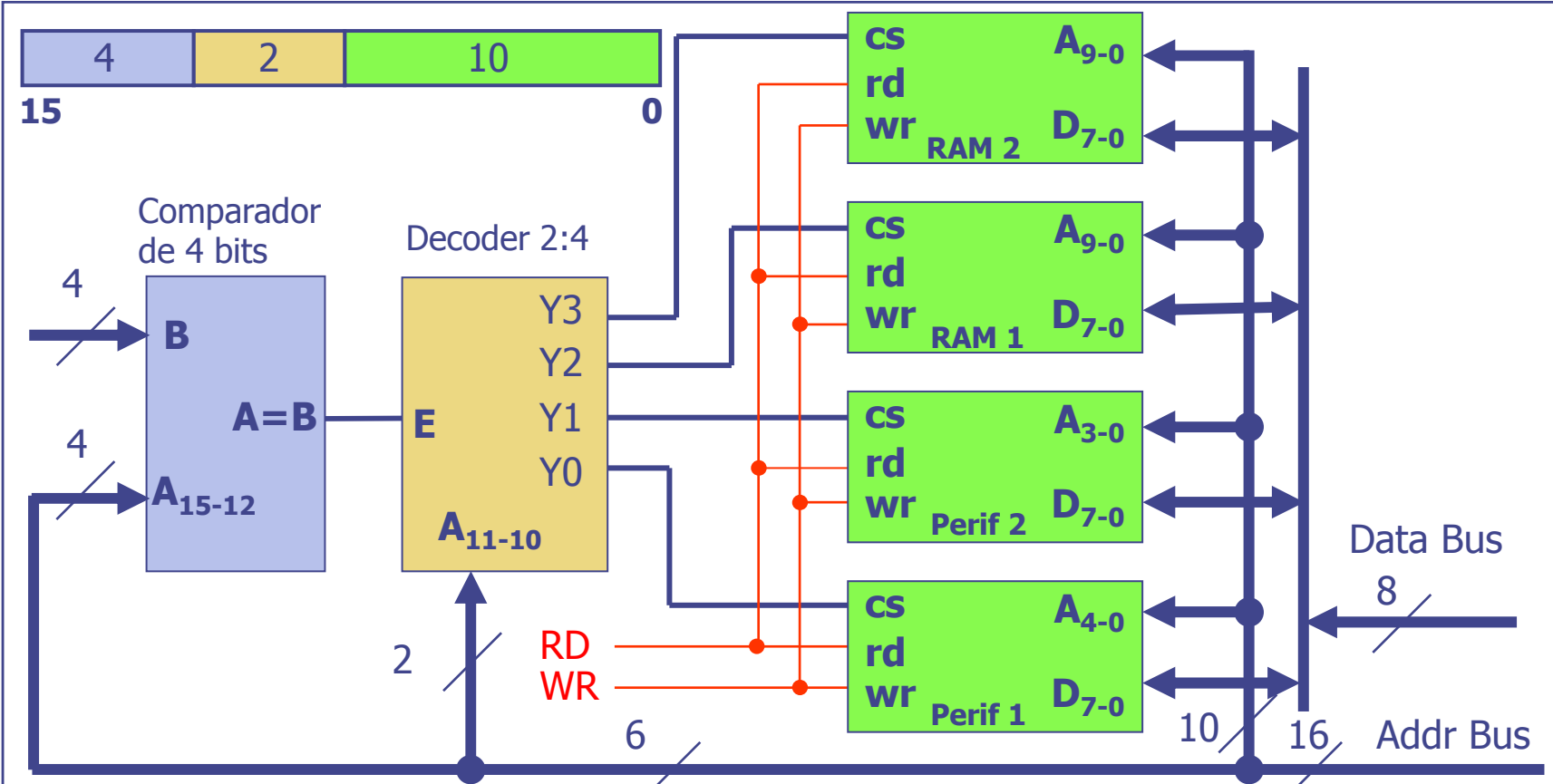
## Gerador de sinais de seleção programável – exercício

- Considerando um espaço de endereçamento de 16 bits, usar o modelo de gerador de sinais de seleção programável para implementar um decodificador de endereços para:
  - Duas memórias RAM de 1 kByte cada
  - Um periférico com 32 registos internos
  - Um periférico com 16 registos internos
- Assuma que o decodificador pode usar o espaço de endereçamento a partir do endereço 0xB000



- Solução:
  - 4 sinais de seleção ( $S=2$ ), cada um ativo em 1024 endereços consecutivos ( $R=10$ )
  - Constante de comparação usada no comparador:  $1011_2$

# Gerador de sinais de seleção programável - exercício



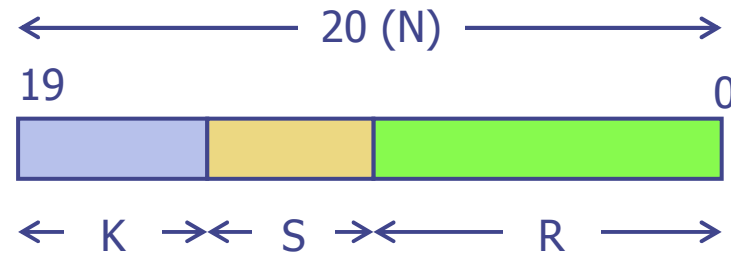
- Indique qual o dispositivo selecionado quando o endereço é 0xB935
- Construa o mapa de memória com os endereços inicial e final em que cada uma das 4 linhas de seleção está ativa
- Construa o mapa de endereços para os 4 dispositivos
- Indique todos os possíveis endereços para aceder ao primeiro registo do periférico 1

# Descodificação de endereços - exercícios

1. Para o exemplo do slide 29, suponha que no descodificador apenas se consideram os bits A15, A13 e A11, com os valores 1, 0 e 0, respetivamente.
  - a) Apresente a expressão lógica que implementa este descodificador: i) em lógica positiva e ii) em lógica negativa.
  - b) Indique os endereços inicial e final da gama-base descodificada e de todas as réplicas.
2. Suponha que, no exercício do slide 33, não se descodificaram os bits A14 e A12, resultando na expressão  $CS = A15 + A13$ 
  - a) Indique as gamas do espaço de endereçamento de 16 bits ocupadas pela memória.
  - b) Indique os endereços possíveis para aceder à 15ª posição da memória.
3. Escreva as equações lógicas dos 4 descodificadores necessários para a geração dos sinais de seleção para cada um dos dispositivos do exemplo do slide 16.
4. Para o exemplo do slide 36, determine a gama de endereços em que cada uma das linhas  $CS_x$  está ativa, com a constante de comparação  $0010_2$

## Gerador de sinais de seleção programável - exercícios

1. Pretende-se gerar os sinais de seleção para 4 memórias de 8 kByte, a mapear em gamas de endereços consecutivas, de modo a formar um conjunto de 32 kByte. O endereço inicial deve ser configurável. Para um espaço de endereçamento de 20 bits:



- Indique o número de bits dos campos K, S e R, supondo descodificação total.
- Esboce o circuito digital que implementa este descodificador
- Indique os endereços inicial e final para a primeira, segunda e última gamas de endereços possíveis de serem descodificadas.
- Para a última gama de endereços, indique os endereços inicial e final atribuídos a cada uma das 4 memórias de 8k
- Suponha que o endereço 0x3AC45 é um endereço válido para aceder ao conjunto de 32k. Indique os endereços inicial e final da gama que inclui este endereço. Indique os endereços inicial e final da memória de 8K à qual está atribuído este endereço



## Gerador de sinais de seleção programável - exercícios

1. Pretende-se gerar os sinais de seleção para os seguintes 4 dispositivos: 1 porto de saída de 1 byte, 1 memória RAM de 1 kByte (*byte-addressable*), 1 memória ROM de 2 kByte (*byte-addressable*), 1 periférico com 5 registos de 1 byte cada um. O espaço de endereçamento a considerar é de 20 bits.
  - a) Desenhe o gerador de linhas de seleção para estes 4 dispositivos, baseando-se no modelo discutido nos slides anteriores e usando a mesma sub-gama para o periférico e para o porto de saída de 1 byte.
  - b) Especifique a dimensão de todos os barramentos e quais os bits que são usados.
  - c) Desenhe o mapa de memória com o endereço inicial e final do espaço efetivamente ocupado por cada um dos 4 dispositivos, considerando para o conjunto um endereço-base por si determinado.
2. O periférico com 5 registos, do exercício anterior, tem um barramento de endereços com três bits. Suponha que esses bits estão ligados aos bits A0, A1 e A2 do barramento de endereços do CPU.
  - a) Usando o decodificador desenhado no exercício anterior, indique os 16 primeiros endereços em que é possível aceder ao registo 0 (selecionado com A0, A1 e A2 a 0)
  - b) Repita o exercício anterior supondo que os 3 bits do barramento de endereços do periférico estão ligados aos bits A2, A3 e A4 do barramento de endereços.