Arquitetura de Computadores

Aula T23 – 30 de Maio de 2023

Dispositivos de entrada/saída:

- Organização do hardware e software

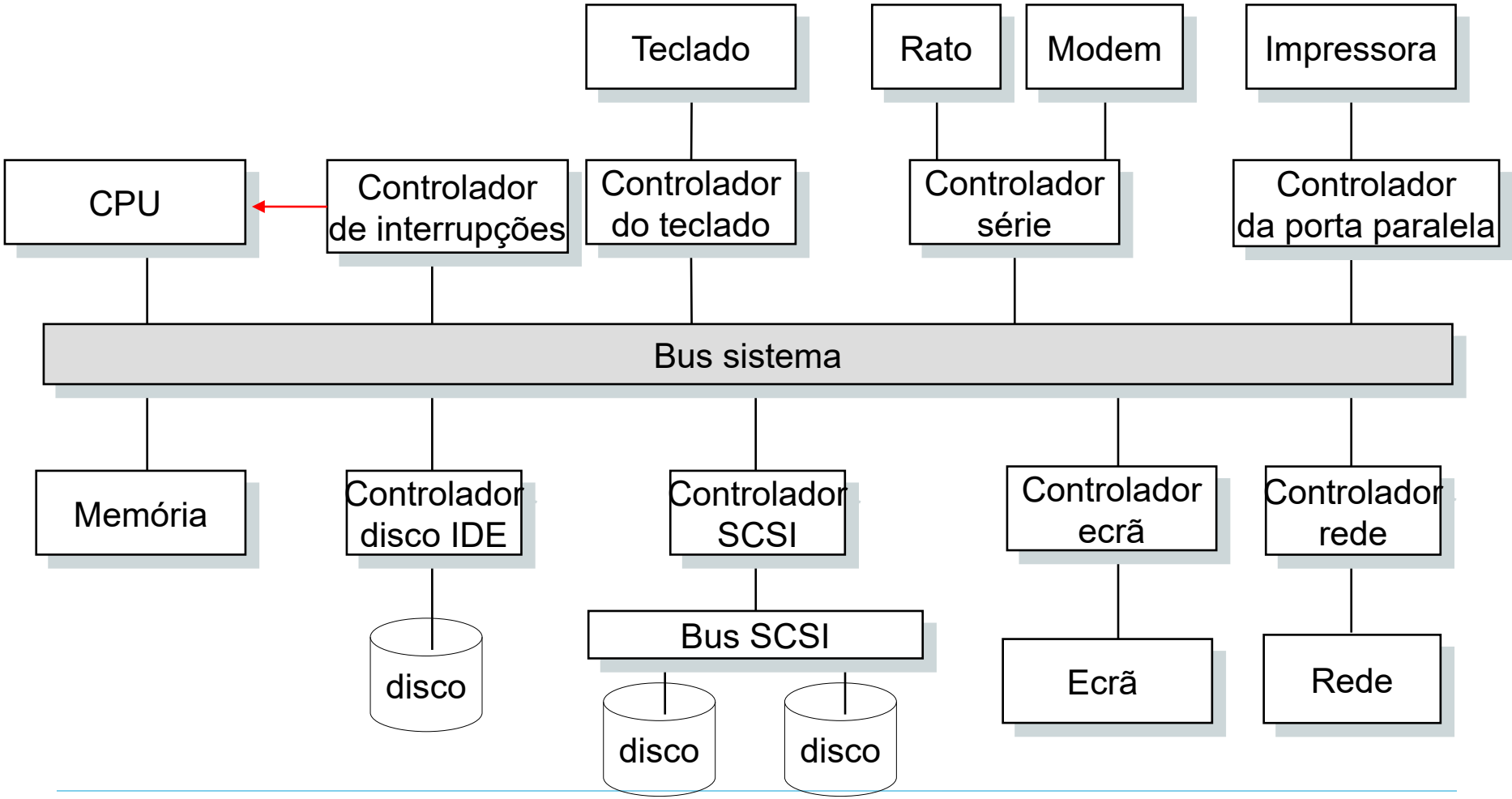
Bibliografia:

OSTEP Cap. 36, secções 36.1 e 36.2

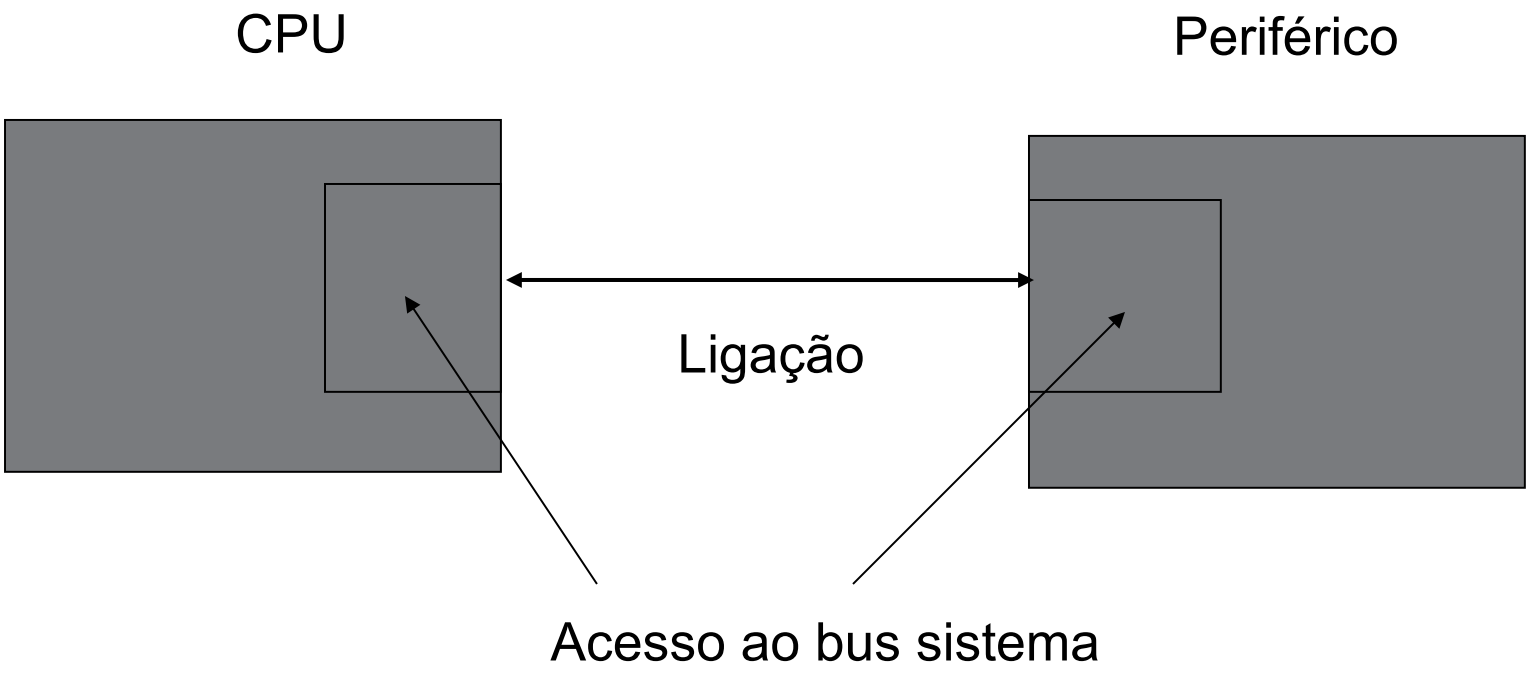
Dispositivos de entrada/saída

- ***Dispositivos de comunicação***:
  - *Dispositivos de entrada*: permitem introduzir no sistema dados gerados por um agente exterior que são transformados em formato binário de forma a poderem ser processados;
    - teclado, rato, controlador de rede (entrada)
  - Dispositivos de saída: recebem dados do CPU/memória em formato digital e tornam-nos acessíveis a um agente exterior
    - ecrã, impressora, controlador de rede (saída)
- ***Dispositivos de armazenamento***: mantêm internamente ao sistema dados em forma não volátil. Podem ser de entrada/saída ou só de entrada
    - Discos magnéticos, discos óticos, discos de estado sólido, “tapes”

Hardware


Teclado
Rato
Modem
Impressora
CPU
Controlador de interrupções
Controlador do teclado
Controlador série
Controlador da porta paralela
Bus sistema
Memória
Controlador disco IDE
Controlador SCSI
Controlador ecrã
Controlador rede
disco
Bus SCSI
Ecrã
Rede
disco
disco

Interligação CPU-Periférico



Para transferir dados entre o CPU e o periférico

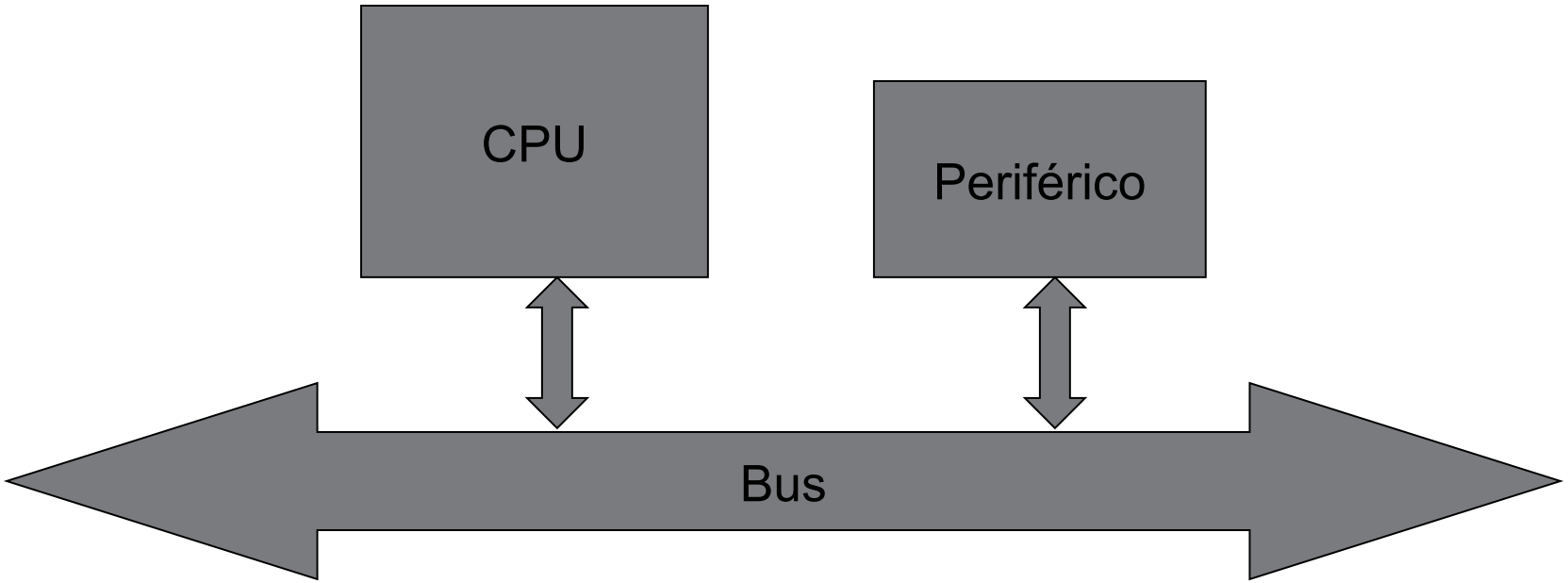
Visão do CPU em relação às entradas/saídas

- O CPU não faz acesso aos periféricos directamente
- Usa uma interface programável para passar pedidos ao controlador de interface e para receber as repostas a esses pedidos
- O controlador de interface converte esses pedidos em sinais eléctricos
- Esses sinais são interpretados pelo controlador do lado do periférico
- Este é que interage directamente com o dispositivo físico.

Bus (barramento)

- Um “bus” é um mecanismo de comunicação digital que permite a duas ou mais unidades funcionais transferir dados e sinais de controle
- Um computador pode conter vários “buses” que estão optimizados para um determinado objectivo
  - “bus” de memória: liga o CPU ao subsistema de memória
  - “bus” de entrada/saída (i/o bus) interliga o CPU a um conjunto de dispositivos de entrada/saída

Bus (barramento)



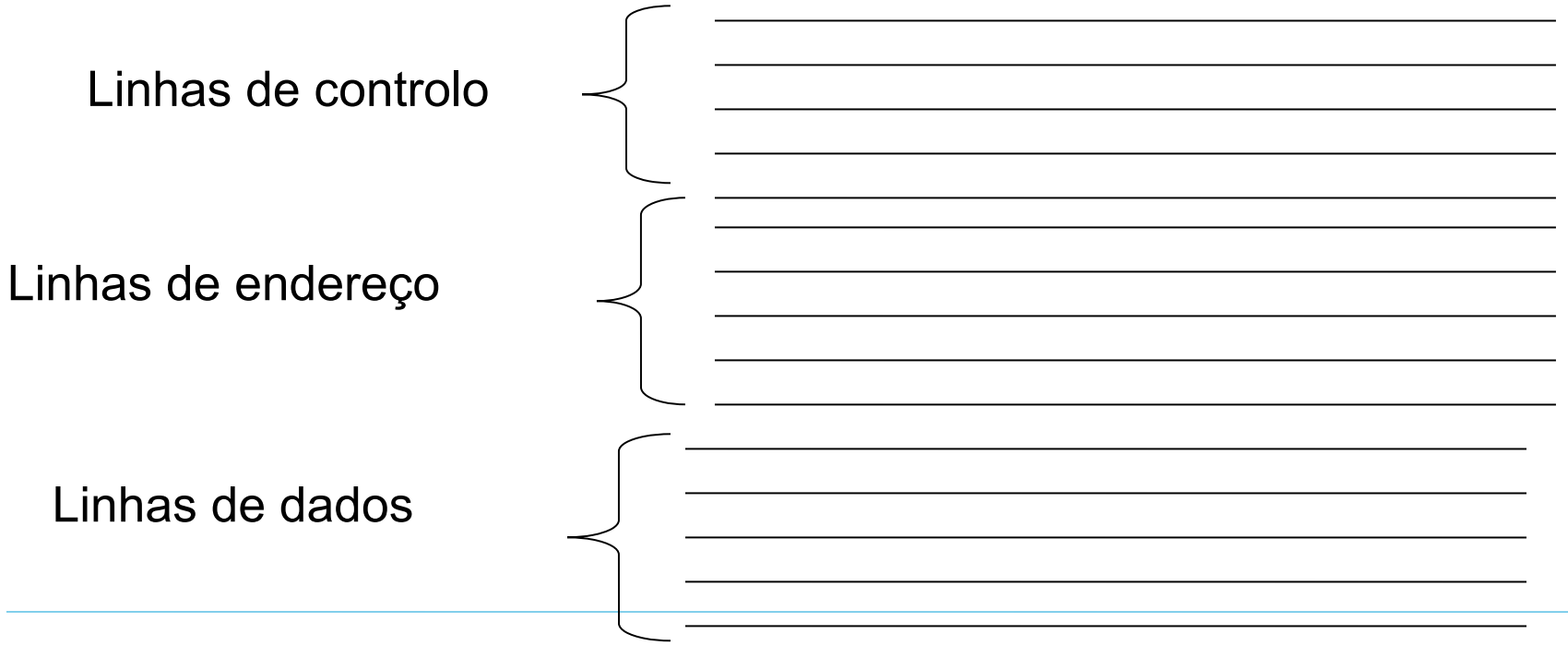
Um “bus” suporta as ligações entre o CPU e os periféricos referidas anteriormente.Na maior parte dos casos transfere múltiplos bits em simultâneo (paralelo)

Características do “bus”

- Transferência de dados em paralelo
  - Pode transferir múltiplos bits em simultâneo
  - Largura típica: 32 ou 64 bits
- O “bus” é normalmente passivo
  - Não contém muita lógica própria; pode ser visto como um mero conjunto de fios (linhas)
  - São os dispositivos a ele ligados que suportam a comunicação

Organização do “bus”

- Bus está separado funcionalmente em 3 partes
  - Controlo
  - Especificação da localização do endereço
  - Dados a transferir

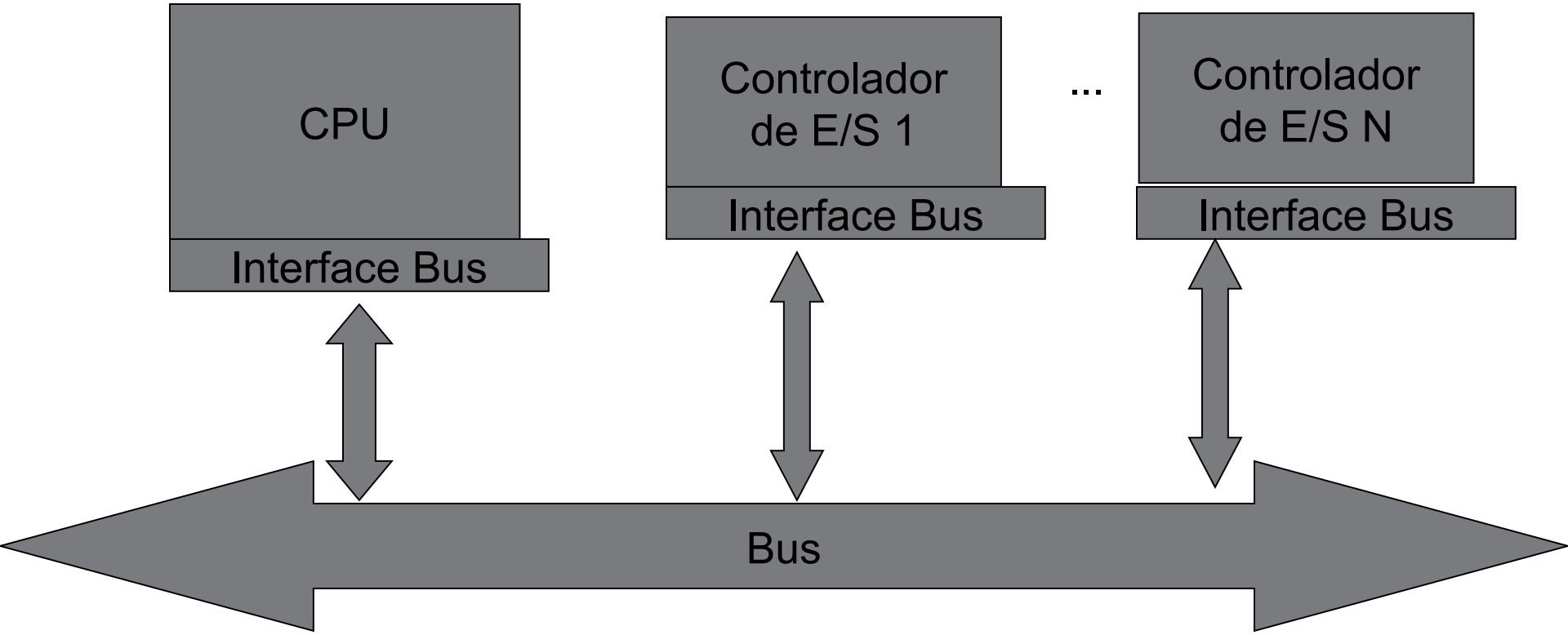


Acesso ao “bus”

- O “bus” só suporta duas operações
  - Leitura
  - Escrita
- O modo de acesso é o mesmo da memória
- Todas as operações de E/S são efectuadas efectuando leituras e escritas no “bus”

Endereçamento no “bus”

- Um “bus” define um espaço de endereçamento



Ponto de vista do CPU

- A interface do bus fornece uma interface de programação, que tem apenas duas operações:
  - Ler e escrever
- Quando há uma referência à memória o CPU deixa tudo a cargo da interface do bus:
  - Leitura – o CPU pede à interface para efectuar a leitura
  - Escrita – idem
- Do ponto de vista do programador, a interface de bus é invisível: ela define um **espaço de endereçamento, em que cada controlador corresponde a uma faixa de endereços distinta**

Acesso em leitura

Instrução máquina do CPU

in endereço, registo do CPU

- Usa as linhas de controlo para obter acesso ao “bus”
- Coloca o endereço *endereço* nas linhas de endereço
- Usa as linhas de controlo para requerer uma operação de leitura
- Testa as linhas de controlo para verificar se a transferência já se concluiu
- Lê um valor das linhas de dados
- Transfere para o registo *registo do CPU*

Acesso em escrita

Instrução máquina do CPU

out registo do CPU, endereço

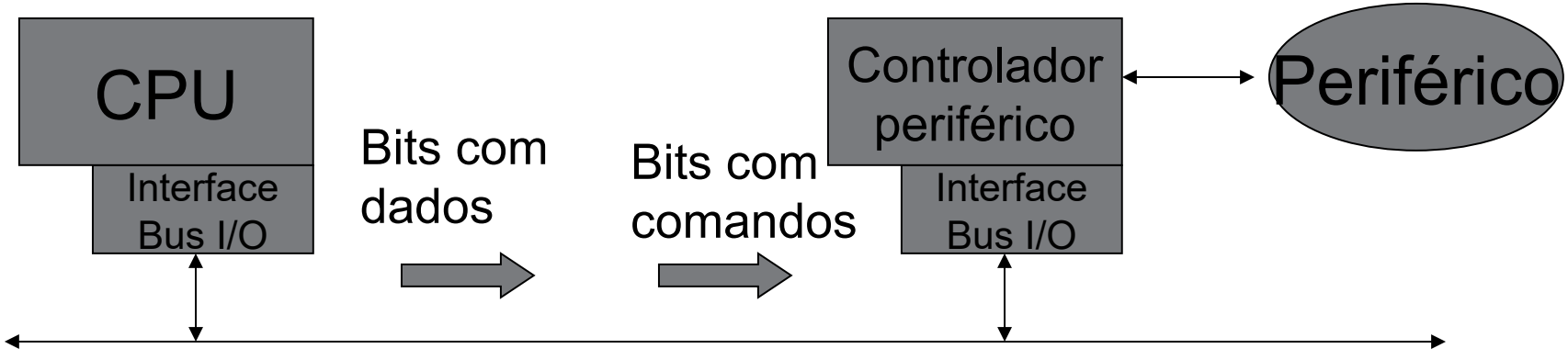
- Usa as linhas de controlo para obter acesso ao “bus”
- Coloca o endereço *endereço* nas linhas de endereço
- Coloca o conteúdo de *registo do CPU* nas linhas de dados
- Usa as linhas de controlo para indicar uma operação de escrita
- Testa as linhas de controlo para verificar se a transferência já se concluiu

Endereçamento no “bus”

- A interface de bus implementa o protocolo do “bus” e trata de todas as transferências
- Usa as linhas de controlo para fazer acesso ao bus
- Posiciona as linhas de dados e endereços
- Embora receba todos os pedidos que passam no “bus”, *a interface só participa nas transferências que contêm os endereços para os quais a interface foi configurada*

Como é que operações de leitura e escrita manipulam periféricos ?

- O bus apenas fornece um meio de passar bits de uma entidade para outra
- Os bits que passam no bus podem ser:
  - Dados que estão a ser transferidos
  - Representar operações de controlo sobre o periférico; uma determinada configuração de bits é interpretada peloa interface hardware



Exemplo: um “display” de luzes

- Periférico faz as seguintes acções:
  - Ligar/desligar o display
  - Mudar o brilho
  - Luz nº i on e off
- Controlador ocupa endereços de 100 a 103; bus tem 16 bits de dados

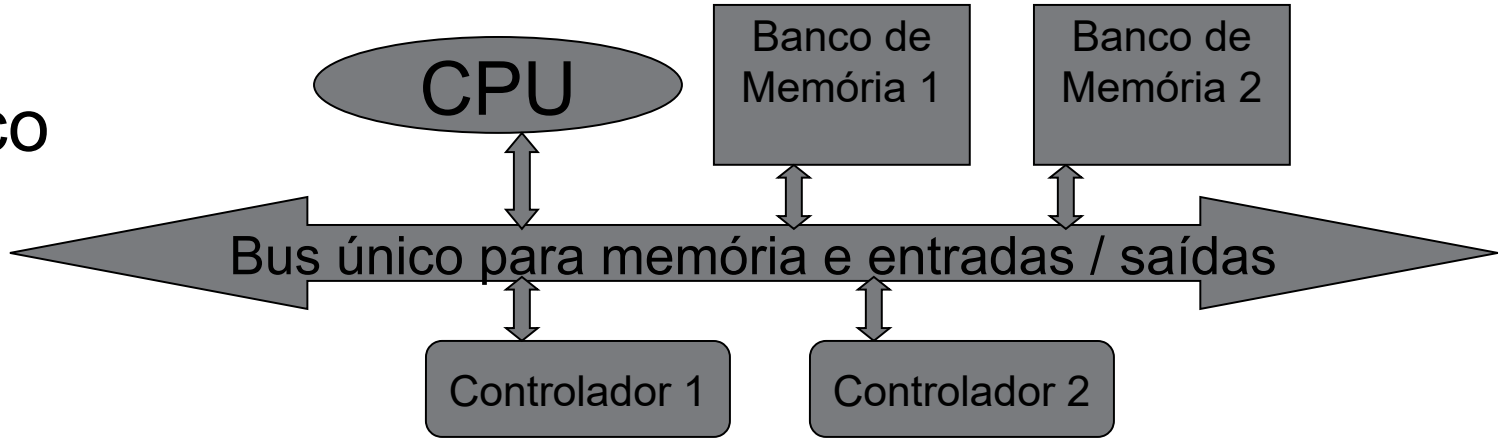
Endereço	Operação	Acções associadas
100	Escrita	Dados a zero desligam o display; dados não nulos ligam o display
101	Leitura	Retorna 0 se o display está desligado; retorna um valor não nulo se está ligado
102	Escrita	Muda o brilho. Os 4 bits menos significativos especificam o brilho de 0 (menor) a 15 (maior)
103	Escrita	“Bitmap” das luzes 0 a 15: bit a 1 luz acesa, bit a 0 luz apagada

Mais detalhe sobre a interação com os controladore de E/S

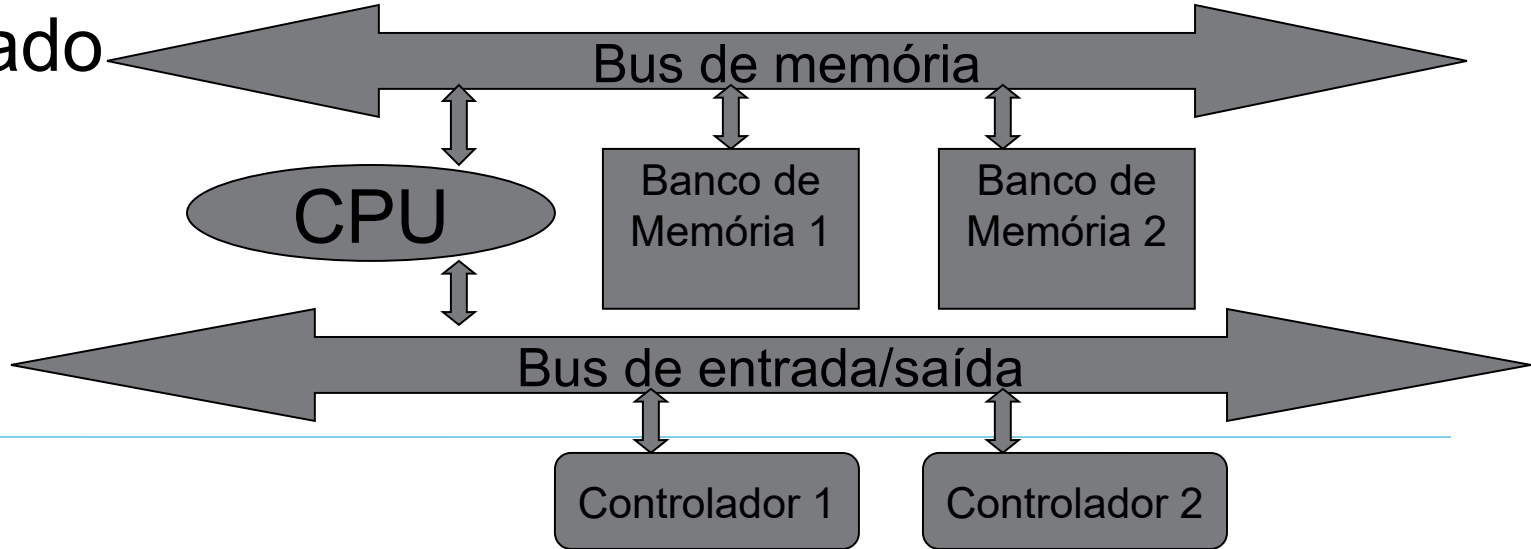
- Como endereçar os vários controladores
- Controlador genérico
- Interação usando espera ativa (exemplo, porta série)
- Interação usando interrupções (próxima aula)

Buses de memória e E/S unificados ou separados

Bus único



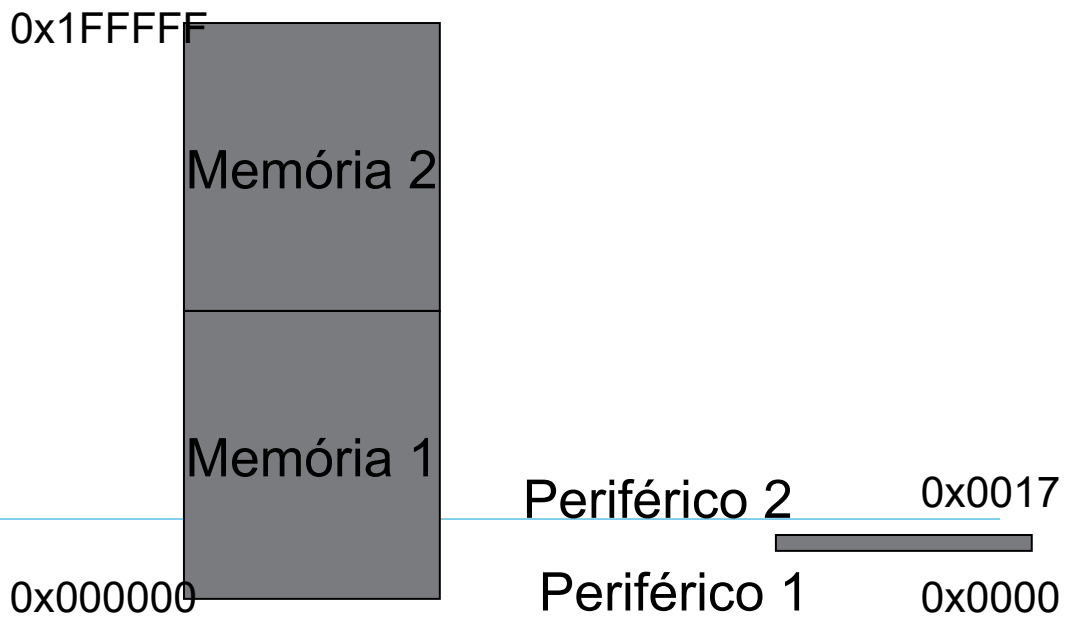
Bus separado



“Buses” separados

- Dois espaços de endereçamento separados
  - Um para a memória
  - Um para os controladores dos dispositivos
- Cada banco de memória é configurado para uma faixa de endereços que não tem intersecção com a faixa de endereços de nenhum outro banco
- Cada controlador é configurado para responder a uma faixa de endereços a que nenhum outro controlador responde

Entidade	Faixa de endereços
Memória 1	0x000000-0x0FFFFF
Memória 2	0x100000-0x1FFFFF
Periférico 1	0x0000-0x000B
Periférico 2	0x000C-0x0017



Instruções máquina para acesso aos controladores de E/S

- Leitura

  Transfere o conteúdo de um endereço de E/S (**porta de E/S**) para um registo do CPU

  in porta e/s, registo

  Pentium in 0x20, al

- Escrita

  Transfere o conteúdo de um registo do CPU para um registo de um controlador de E/S (**porta de E/S**)

  out registo, porta e/s

  Pentium out al, 0x20

Programação dos controladores de dispositivos de E/S

- Controladores podem ser mais ou menos autónomos em relação ao CPU
  - Controladores com pouca inteligência precisam de grande interacção com o CPU
  - Controladores com grande inteligência só necessitam de atenção do CPU no início e fim da transferência
- Sincronização entre o CPU e os controladores
  - Periféricos são muito mais lentos do que o CPU
  - O CPU tem de interactuar com o periférico para saber se este já está disponível para receber novo comando, se já tem o dado pretendido ...

Registos de uma interface (1)

- Uma interface ocupa uma série de endereços (normalmente consecutivos) no espaço de endereçamento
  - Registo(s) de dados (leitura): onde se lêm os dados que vêm do exterior
  - Registo(s) de dados (escrita): onde se escrevem os dados a enviar para o exterior
  - Registo de comando (escrita): onde se dão comandos à interface
  - Registo de estado (leitura): conjunto de bits que dão informação sobre o estado do controlador: livre/ocupado, transferência com/sem erro , ...

Registos de uma interface (2)

- Muitas vezes, o registo de comando e de estado ocupam o mesmo endereço
  - Uma escrita no endereço controla o periférico
  - Uma leitura do mesmo endereço retorna o estado do controlador
- Muitas vezes, uma operação de leitura, implica implicitamente uma ordem para ler mais um valor. Exemplo:
  - Quando um rato se move, há um registo de dados em que fica guardado o movimento relativo em relação à última posição
  - Quando o CPU lê esse registo de dados, desencadeia automaticamente um novo processo de medição de deslocamento, cujo resultado virá para o registo de dados.

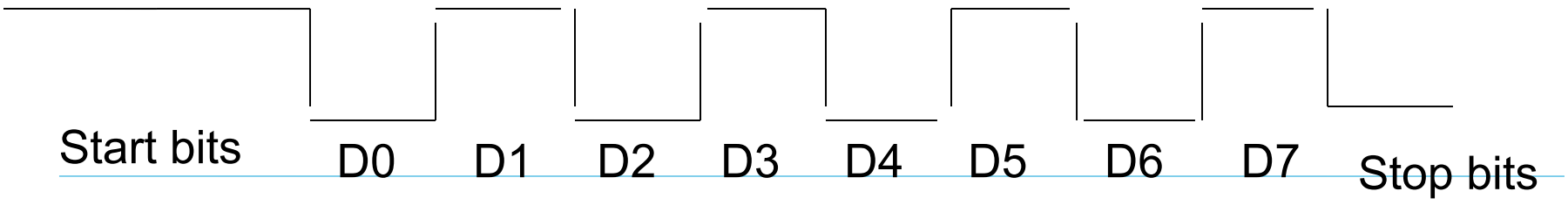
Programação por espera activa (polling)

- O CPU interroga repetidamente o controlador do periférico para saber se a operação anteriormente especificada já terminou
- Exemplo – interacção com uma porta série

Programação por espera activa de um controlador série

Porta série:

- Usada na ligação a modems, impressoras série, ...
- Faz uma conversão paralelo/série (saída) e série/paralelo (entrada)
- Comunicação síncrona
- Comunicação assíncrona
  - Cada conjunto de bits de dados (por exemplo 8) é precedido por 1 ou 2 “start bits” e 1 ou 2 “stop bits”
  - Isto ajuda à sincronização entre emissor e receptor



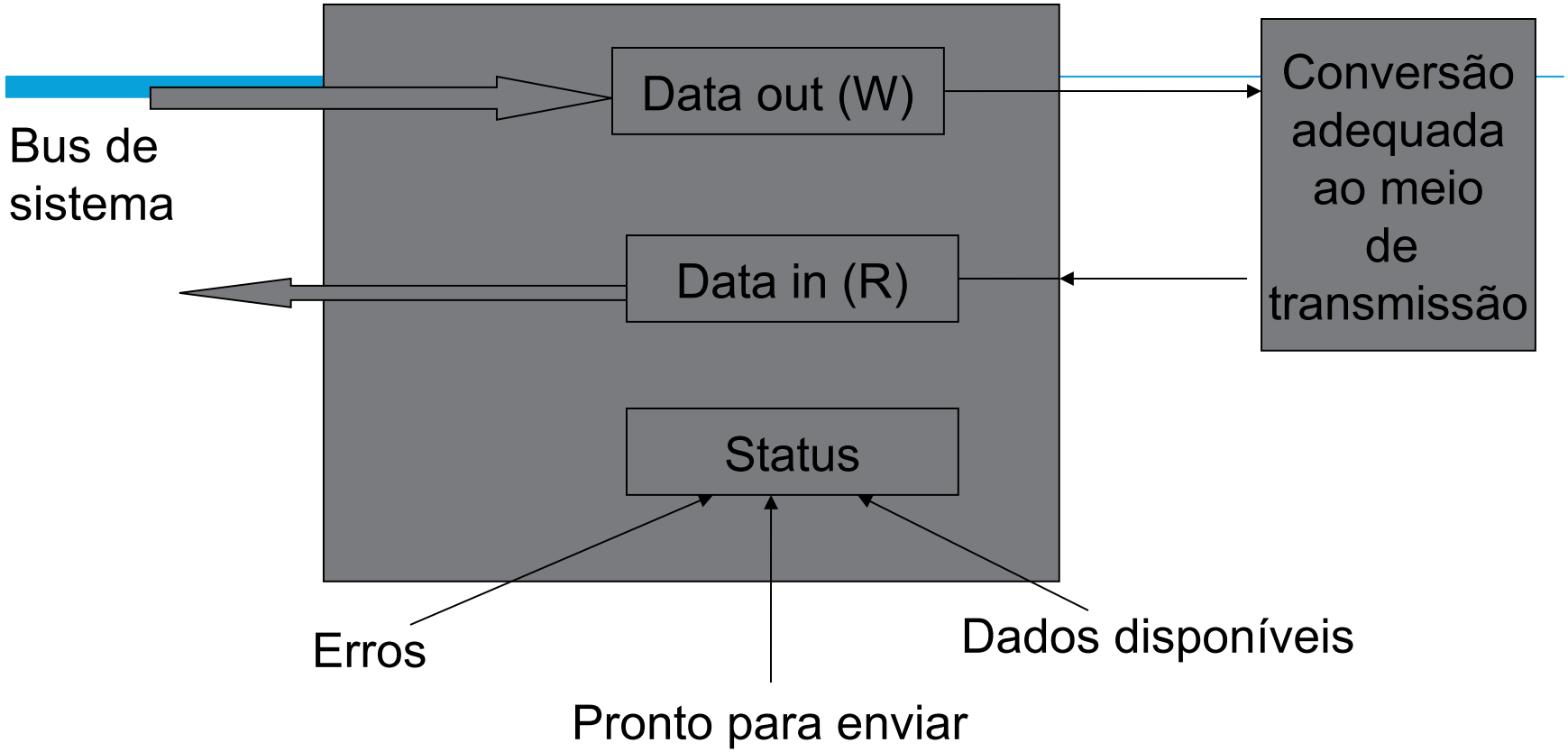
Programação por espera activa de um controlador série

- Porta série:
  - Programação de vários parâmetros:
    - Baud rate (inverso da “duração” de um bit)
    - Paridade (detecção de erros)
    - Número de bits de dados: ASCII (7 ou 8)
    - Número de “start bits” e “stop bits”
  - No MS-DOS

```
mode com1:9600,n,8,1
```

    Ritmo de transmissão: 9600 bit/s; sem paridade; 8 bits de dados; 1 start bit

Porta série

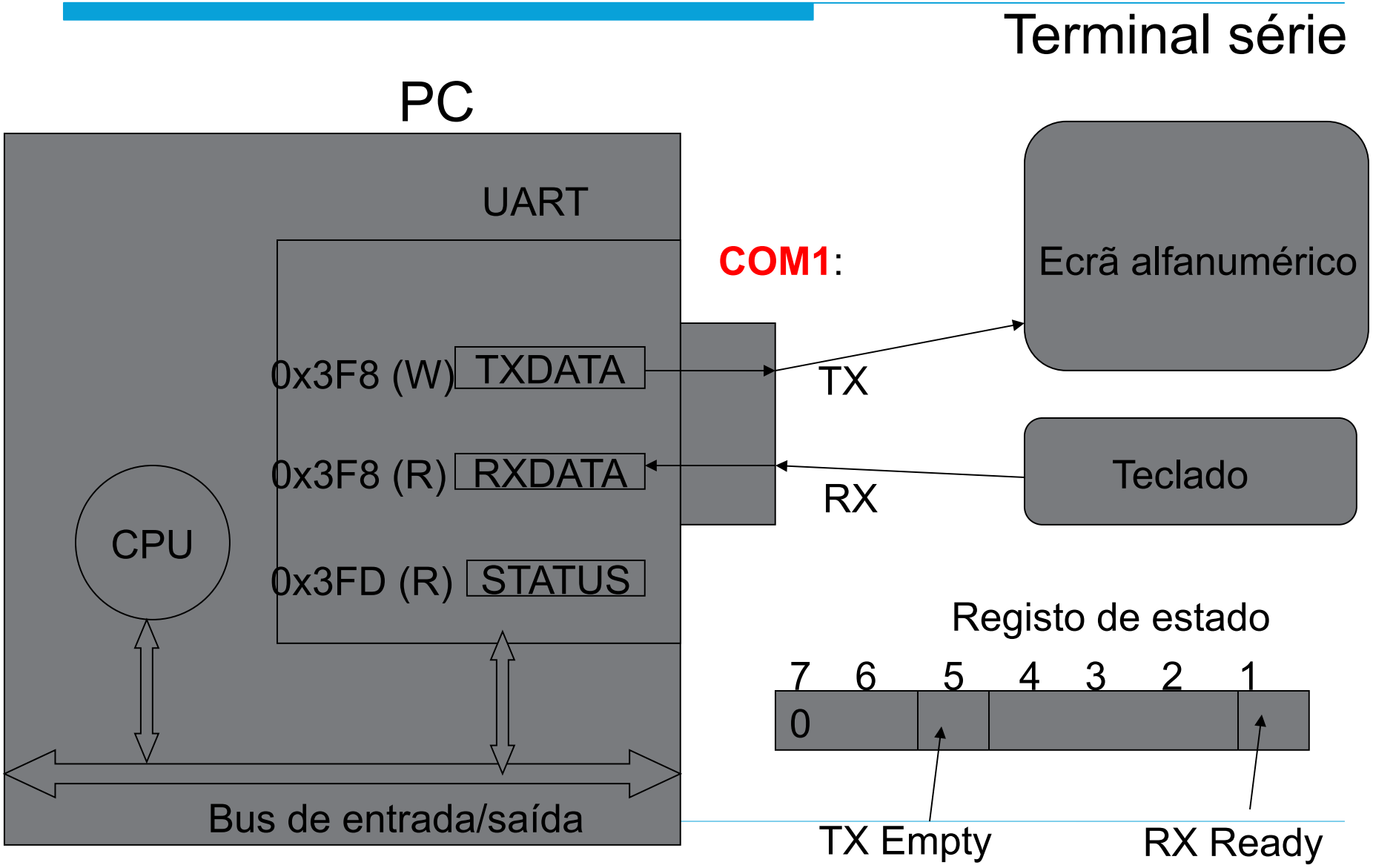


No PC o controlador série é constituído por um único circuito integrado, chamado UART (Universal Asynchronous Receiver Transmiter).
O nome no Windows e MSDOS é COM1:
No Linux /dev/ttys0

Porta série do PC

- Registo de dados de saída (DATA OUT)
  - 0x3F8
  - A escrita neste registo desencadeia a serialização dos dados e saída para o exterior
- Registo de dados de entrada (DATA IN)
  - Ox3F8
  - Este registo acumula um sequência de bits recebida em série
- Registo de estado
  - 0x3FD
  - Bit 0: a 1 indica que há um dado para ler em DATA IN; assim que o dado é lido passa a 0
  - Bit 5: a 1 indica que está pronto a transmitir; assim que é escrito um valor em DATA OUT passa a 0
  - Bits 1,2,3 – vários tipos de erros

Porta série do PC



Porta série

- Suponha-se a existência das duas seguintes funções C que são equivalentes às instruções máquina IN e OUT

```
unsigned char inportb( unsigned short endereço);
void outportb( unsigned short endereço, unsigned char valor);
```

Para enviar um byte teremos

```
void send_serial( unsigned char b ){
      unsigned char s;
      do {
            s = inportb(0x3fd);
      } while(s & 0x20) == 0;
      outportb( 0x3f8, b);
}
```

Porta série

```
unsigned char inportb( unsigned short endereço);
void outportb( unsigned short endereço, unsigned
char valor);
```

Para receber um byte teremos

```
unsigned char receive_serial( ){
     unsigned char s;
     do {
          s = inport(0x3fd);
     } while(s & 0x01) == 0;
     return inport( 0x3f8);
}
```

Inconvenientes do uso do “polling”

- Os controladores que só suportam espera activa são muito simples do ponto de vista hardware, mas implicam um grande desperdício de tempo de CPU
- Os dispositivos de E/S são muito lentos e o CPU vai passar grande parte do tempo nos ciclos do exemplo anterior
- O tempo de espera é fixo (depende do periférico) e independente da velocidade do CPU
- Enquanto se espera há potencial para o CPU executar muitas instruções

O mecanismo de interrupções aumenta a taxa de uso do CPU

- A invenção do mecanismo de interrupções (~1960) permitiu resolver a questão da desadequação das velocidades do CPU e dos periféricos
- Permite que o CPU continue a efectuar computações enquanto espera que a transferência de dados acabe
- O controlador atua autonomamente depois do CPU iniciar a operação de transferência; quando esta termina o controlador envia uma interrupção ao CPU.