

Entrada e Saída: Princípios de Hardware e Software

Prof. Dr. Márcio Castro
marcio.castro@ufsc.br



1

Principios de hardware

Princípios de Hardware

- **Tipos de dispositivos de E/S**

- Dispositivos de blocos
- Dispositivos de caracteres

Dispositivos de blocos

- Armazenam informação em **blocos de tamanho fixo**
- Cada **bloco possui um endereço**
- Transferências em unidades de **um ou mais blocos consecutivos**
- **Exemplos:** disco rígido, pen drive, leitor de DVD, ...



Dispositivos de caractere

- Envia ou recebe um **fluxo de caracteres**
- Não é endereçável
- Não dispõe de operação de posicionamento
- **Exemplos:** impressora, interface de rede, mouse, ...



Controladores de dispositivos

- Unidades de E/S são compostas por componentes **eletrônicos**
 - Muitas delas também possuem componentes **mecânicos**
- **Componentes eletrônicos**
 - Controlador do dispositivo
 - Demais componentes (memórias, resistores, capacitores, indutores, ...)
- **Componentes mecânicos**
 - Motores, discos, ...

Controlador

- **Funções do controlador**

- Converter um **fluxo serial de bits** oriundo do dispositivo em um **bloco de bytes**
- Bloco é montado em um **buffer no próprio controlador**
- Executar correções de erro (e.g., ECC)

- **Interfaces conectam dispositivos de E/S aos computadores**

- **Exemplos:** IDE, SATA, SCSI, USB, ...

Comunicação com controladores

- Controladores possuem **registradores** para comunicação com a CPU
 - **Registradores de controle**: envio de comandos para o dispositivo e a verificação do estado do dispositivo
 - **Buffers de dados**: leitura e escrita de dados
- **Comunicação com os registradores dos controladores**
 - Portas de E/S
 - E/S mapeada na memória
 - Abordagem híbrida

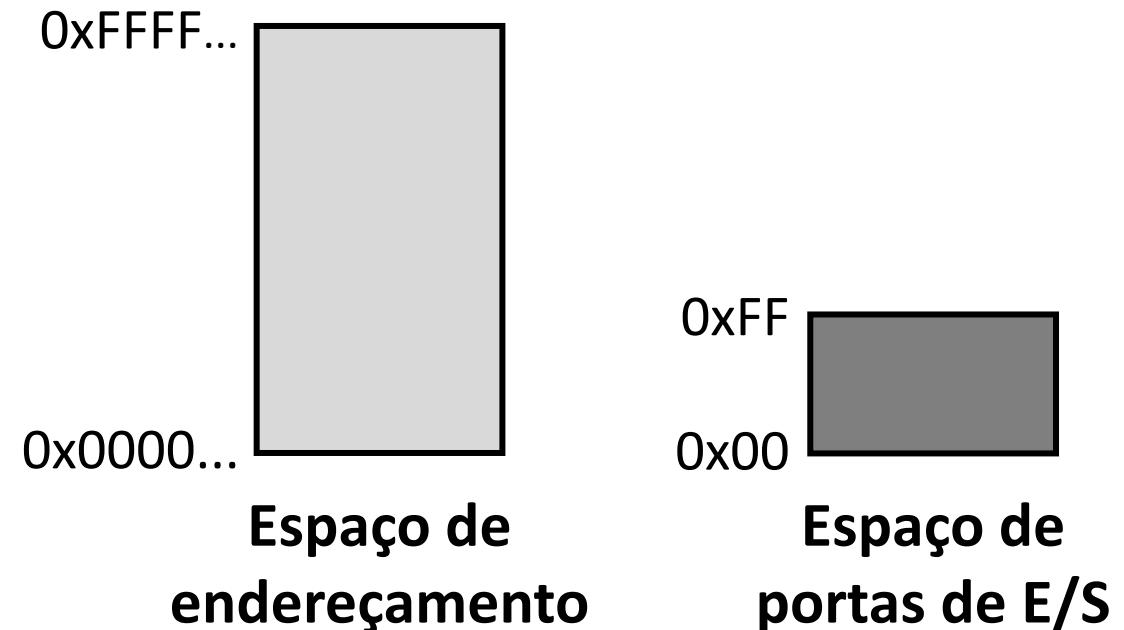
Comunicação via portas de E/S

- Cada registrador de controle é associado a uma porta de E/S
- Espaços de endereçamento e de portas de E/S distintos
- Espaço de portas de E/S é acessível somente através de instruções especiais

Exemplo de instruções de E/S

IN REG, PORT

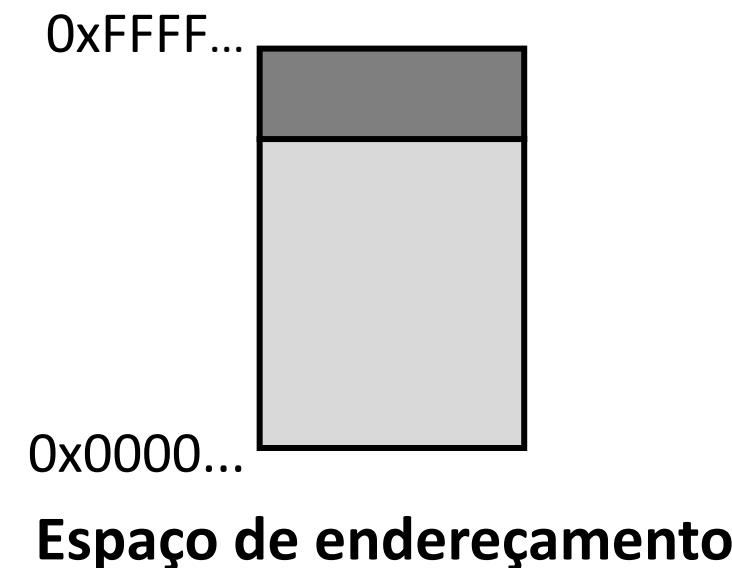
OUT PORT, REG



Adaptado de TANENBAUM, Andrew S. *Sistemas operacionais modernos*. 3. ed. Rio de Janeiro: Pearson Prentice Hall, 2010. xiii, 653 p. ISBN 9788576052371.

Comunicação via E/S mapeada em memória

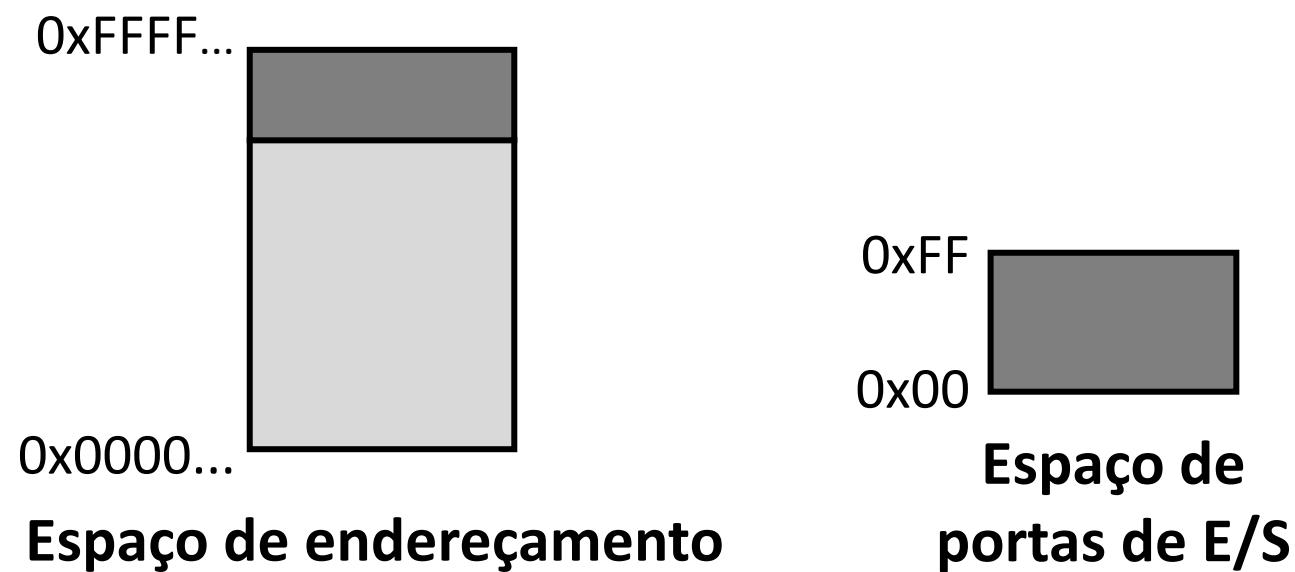
- Mapeia os **registradores de controle** dos dispositivos no **espaço de endereçamento da memória** do computador
- Cada registrador de controle é associado a um **endereço de memória**
- Acessível através de **instruções de escrita e leitura da memória**



Adaptado de TANENBAUM, Andrew S. *Sistemas operacionais modernos*. 3. ed. Rio de Janeiro: Pearson Prentice Hall, 2010. xiii, 653 p. ISBN 9788576052371.

Abordagem híbrida

- Mapeia os **registradores de controle** dos dispositivos no **espaço portas de E/S**
- Mapeia **buffers de dados** dos dispositivos no **espaço de endereçamento em memória**

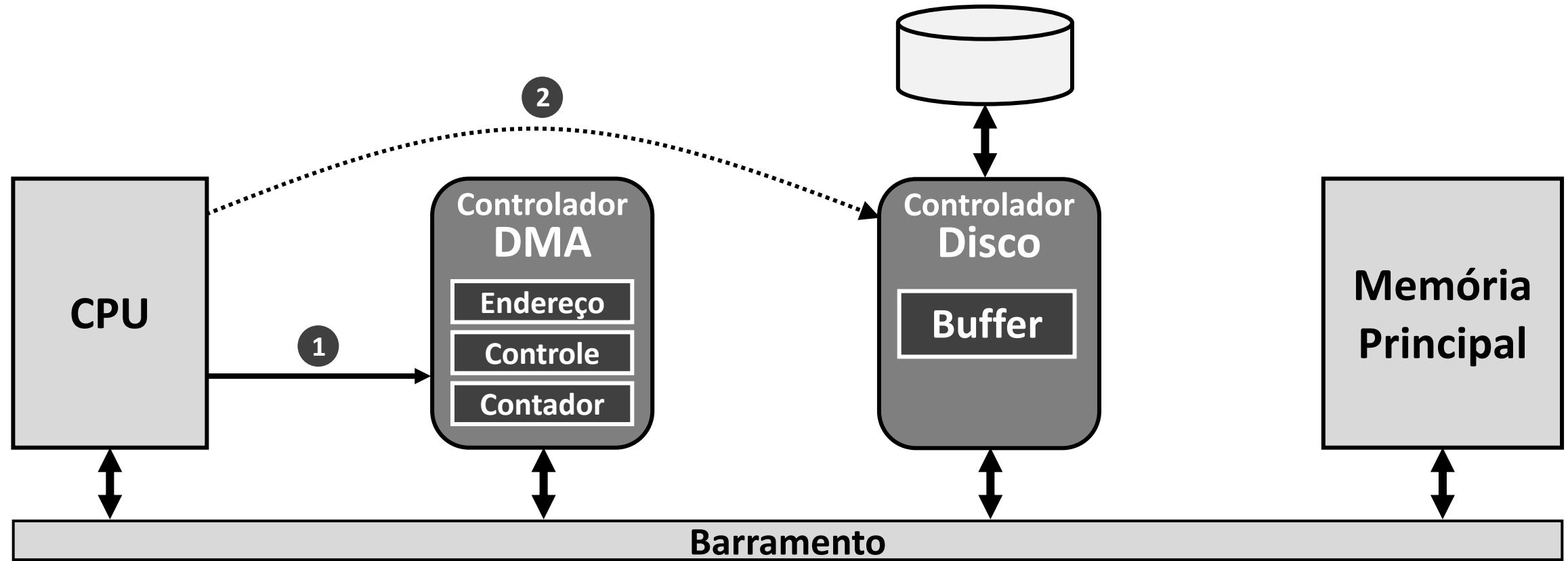


Adaptado de TANENBAUM, Andrew S. *Sistemas operacionais modernos*. 3. ed. Rio de Janeiro: Pearson Prentice Hall, 2010. xiii, 653 p. ISBN 9788576052371.

Direct Memory Access (DMA)

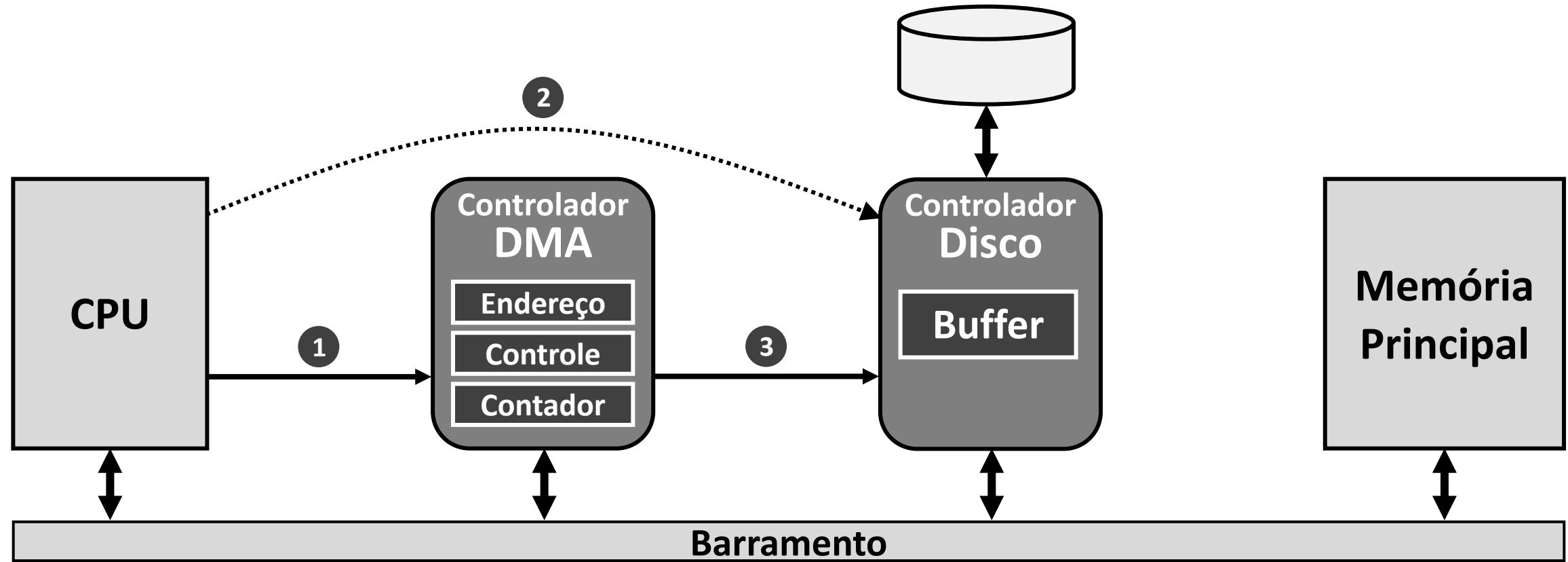
- Independentemente da forma de comunicação, a CPU precisa **endereçar os controladores de dispositivo** para troca de dados
- O desempenho geral do sistema fica **comprometido** se a **CPU** for encarregada de **realizar as transferências de dados** entre buffers do controlador e a memória RAM
- **Direct Memory Access (DMA)**
 - Um controlador específico realiza as **transferências de dados**
 - A CPU fica **livre** para executar outras tarefas

DMA: exemplo de leitura de um bloco do disco



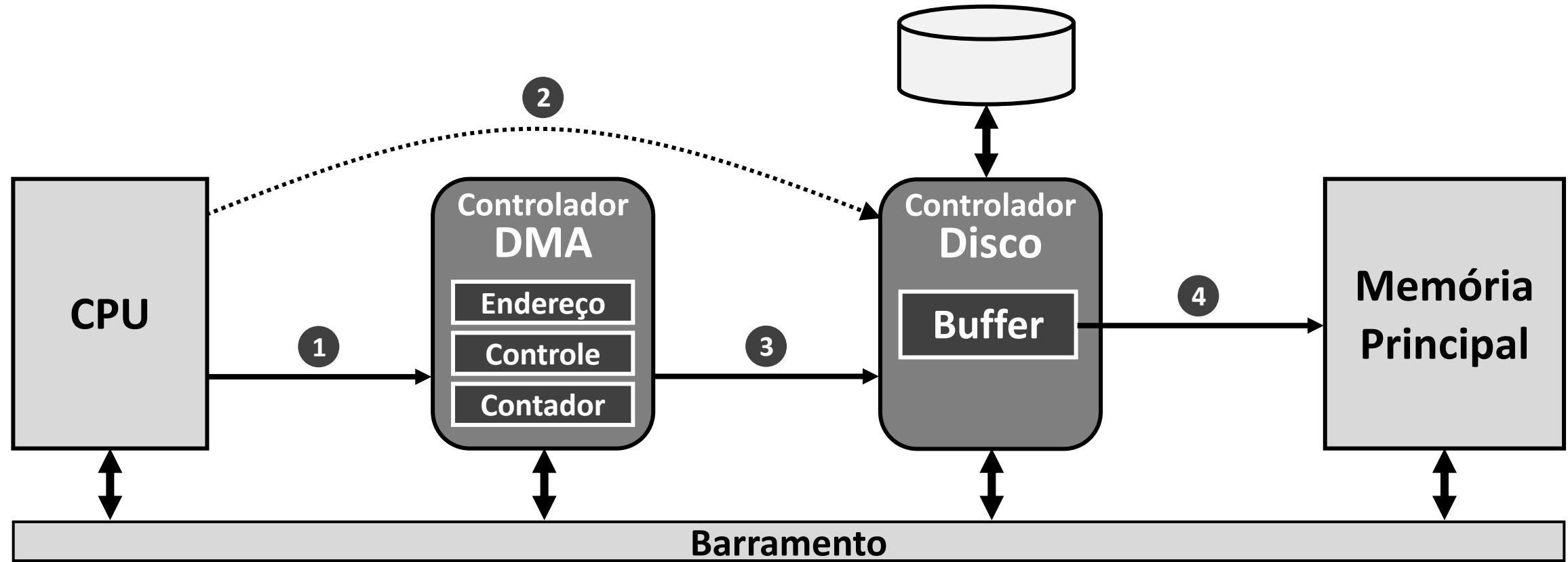
- ① A CPU programa o controlador DMA, informando endereço de memória que irá receber os dados do bloco do disco e a quantidade de bytes a serem lidos
- ② A CPU requisita leitura ao controlador do disco, dados são trazidos para o buffer

DMA: exemplo de leitura de um bloco do disco



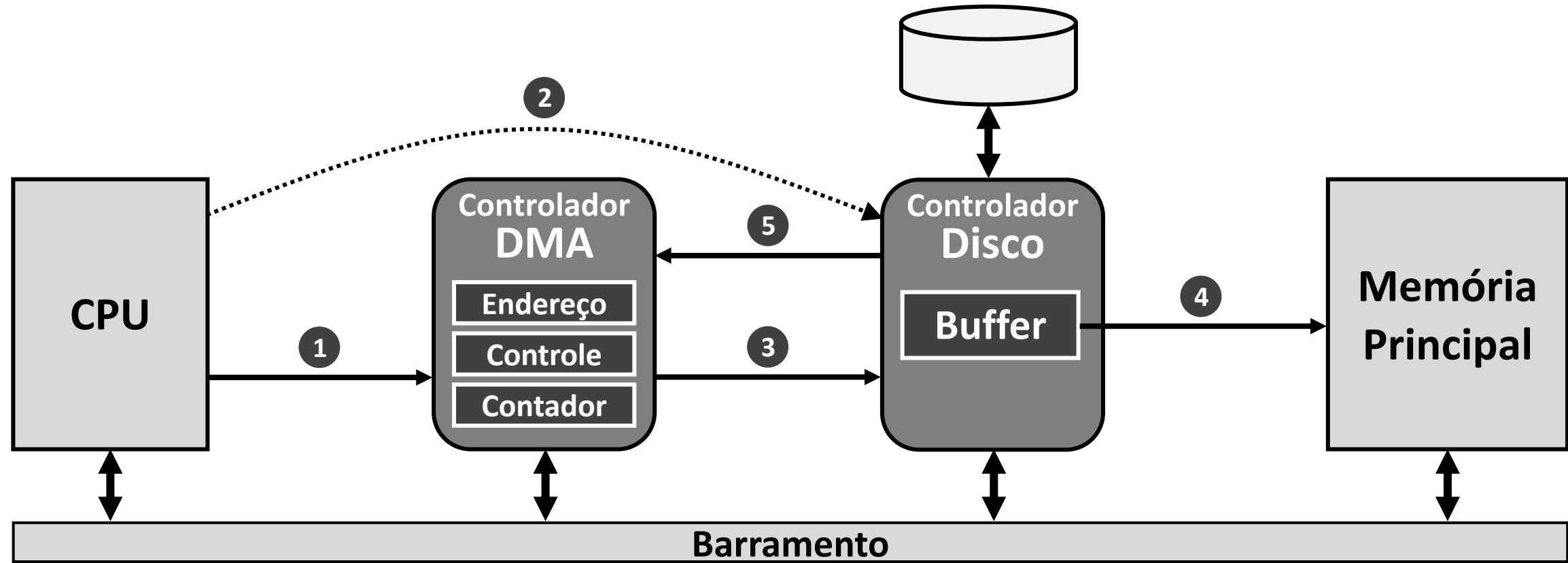
- ③ Controlador DMA requisita transferência de uma palavra do bloco armazenado no buffer ao controlador do disco**

DMA: exemplo de leitura de um bloco do disco



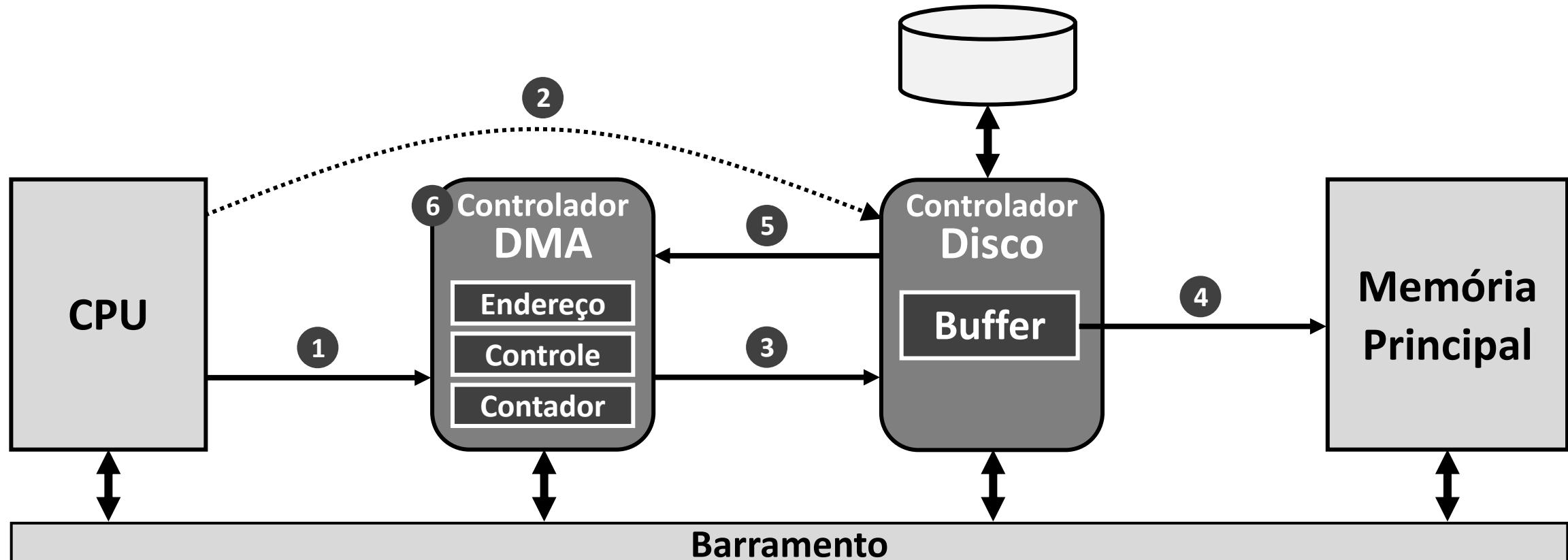
- ④ A palavra é transferida do buffer do controlador do disco para um endereço na memória principal

DMA: exemplo de leitura de um bloco do disco



- 5 Quando a transferência **termina**, o controlador do disco **envia um sinal de confirmação ao controlador DMA**

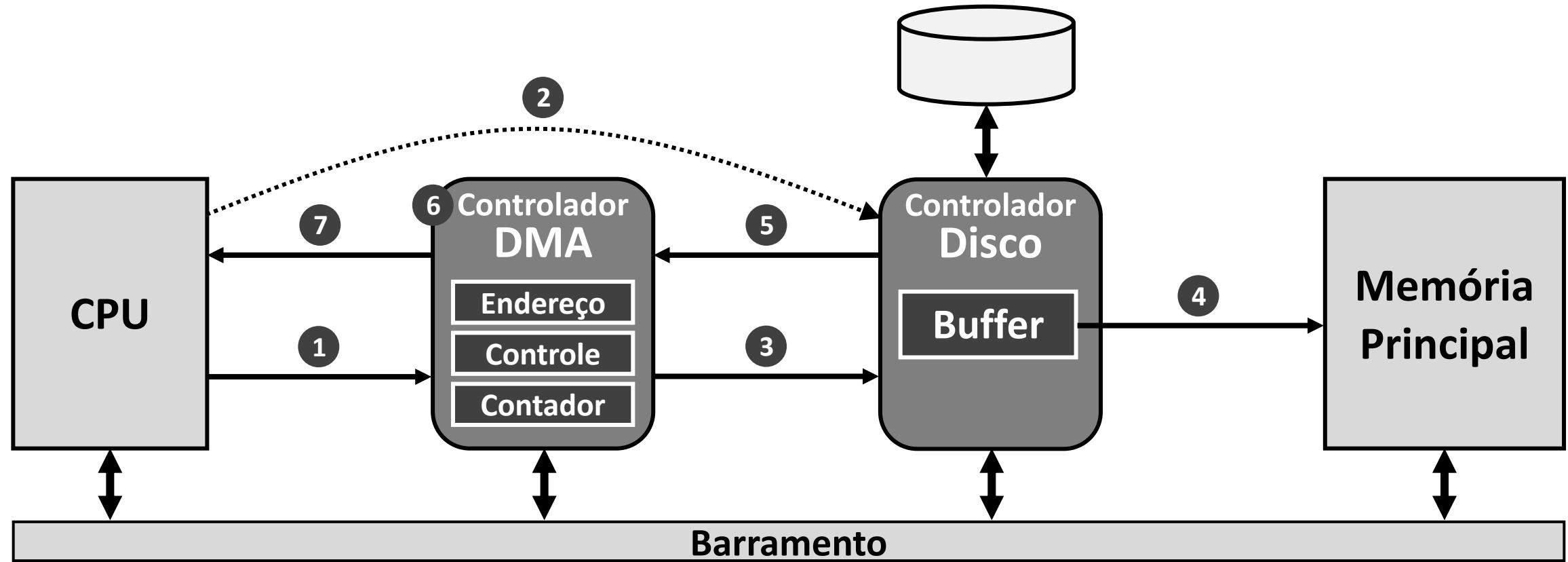
DMA: exemplo de leitura de um bloco do disco



⑥ O controlador DMA **incrementa o endereço de memória fornecido e decrementa o contador de bytes**

Os passos 3–6 são repetidos até que o contador tenha o valor zero

DMA: exemplo de leitura de um bloco do disco



- 7 O controlador DMA envia uma interrupção para a CPU para informar que a transferência do bloco foi finalizada

Direct Memory Access (DMA)

- Controladores DMA mais sofisticados são capazes de **tratar múltiplas transferências simultaneamente**
 - Diversos canais de DMA
 - Um conjunto de registradores para cada **canal de DMA**
- Esses controlados possuem um **escalonador** para lidar com múltiplas transferências simultâneas
- **Escalonadores mais utilizados**
 - Round-robin
 - Prioridades

2

Principios de software

Princípios de Software

- **Exemplo:** passos para comunicação com **impressora** via porta serial
 - ① Enviar um caractere
 - ② Aguardar a impressora estar pronta para receber um novo caractere
 - ③ Repetir os passos 1–2 até que todos os caracteres tenham sido transferidos
- **Três maneiras de realizar E/S**
 - E/S programada
 - E/S orientada à interrupção
 - E/S com DMA

E/S programada

- CPU participa **ativamente** da operação de E/S
- Envio de caracteres para impressora com **espera ocupada**
- Retorno para o espaço de usuário quando a **transferência termina**

```
copy_from_user(buffer, p, count);          /* p is the kernel buffer */
for (i = 0; i < count; i++) {                /* loop on every character */
    while (*printer_status_reg != READY) ;   /* loop until ready */
    *printer_data_register = p[i];           /* output one character */
}
return_to_user();
```

Exemplo

E/S orientada à interrupção

- Transferência iniciada por uma **chamada de sistema**, liberando o processo
- Impressora envia uma **interrupção** quando estiver pronta para receber um novo caractere
- CPU executa o tratador de interrupção **para transferir o próximo caractere**

```
copy_from_user(buffer, p, count);
enable_interrupts();
while (*printer_status_reg != READY) ;
*printer_data_register = p[0];
scheduler();
```

Chamada de sistema

```
if (count == 0) {
    unblock_user();
} else {
    *printer_data_register = p[i];
    count = count - 1;
    i = i + 1;
}
acknowledge_interrupt();
return_from_interrupt();
```

Tratador de
interrupção

E/S com DMA

- CPU não participa das transferências
- Gerenciamento de transferências é feito pelo **controlador de DMA**
- Uma única interrupção quando toda a transferência tiver sido finalizada

```
copy_from_user(buffer, p, count);  
set_up_DMA_controller();  
scheduler();
```

Chamada de sistema

```
acknowledge_interrupt();  
unblock_user();  
return_from_interrupt();
```

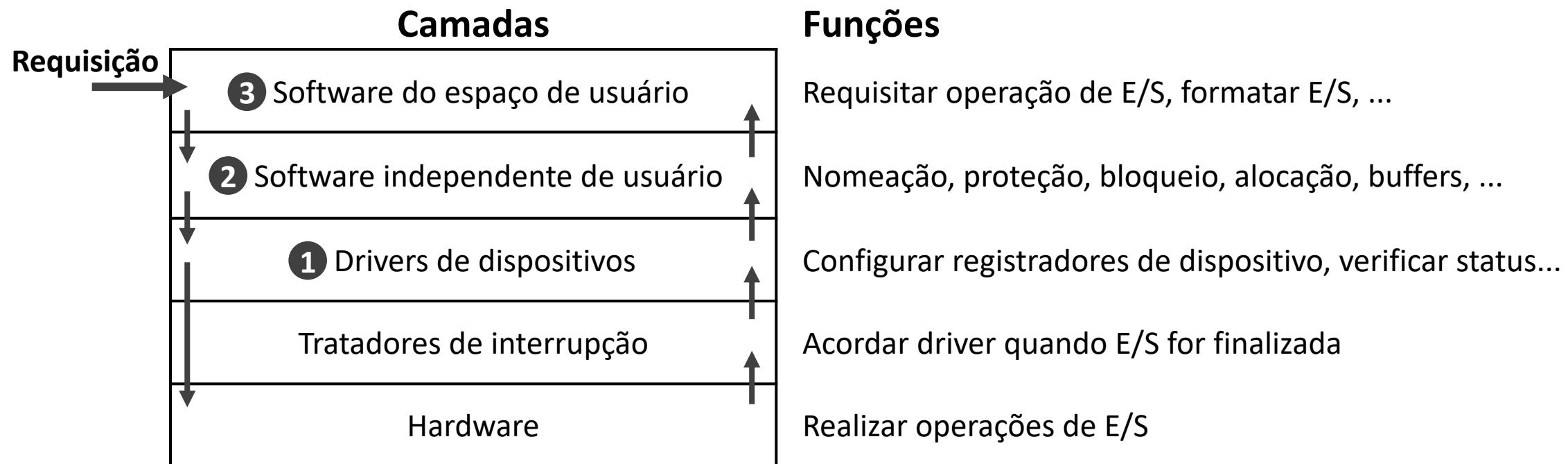
Tratador de interrupção

Camadas de software

1 Drivers de dispositivos

2 Software de E/S independente de dispositivo

3 Software de E/S do espaço de usuário



Adaptado de TANENBAUM, Andrew S. *Sistemas operacionais modernos*. 3. ed. Rio de Janeiro: Pearson Prentice Hall, 2010. xiii, 653 p. ISBN 9788576052371.

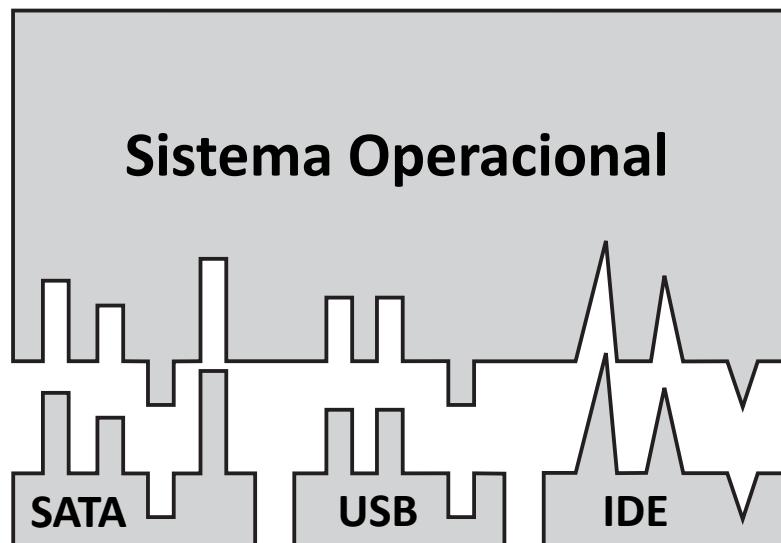
Drivers de dispositivos

- O número de registradores e a natureza dos comandos de cada dispositivo variam **radicalmente**
 - Exemplo: disco vs. mouse
 - Logo, há a necessidade de **drivers diferentes para cada dispositivo**
- Drivers são geralmente feitos pelo **fabricante do dispositivo**
 - Drivers **genéricos** funcionam em dispositivos do **mesmo tipo**

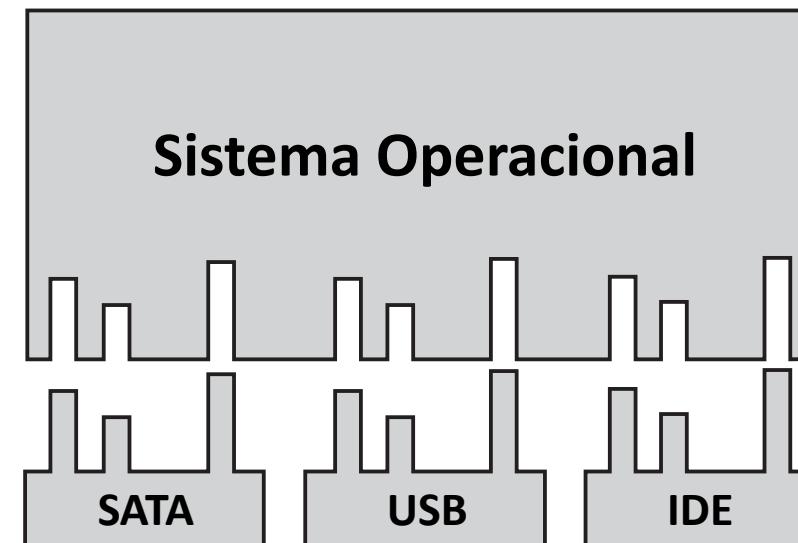
Software de E/S independente de dispositivo

- A camada de E/S independente de dispositivo define uma **interface padrão** a ser implementada pelos **drivers de dispositivo**
- Ligação entre as funções da **interface padrão** e as **implementações dessas funções no driver** é feita através de uma **tabela em memória**

Sem interface padrão



Com interface padrão



Vs.

```

/* Number of character and block devices. */
#define NR_CHRDEV 3
#define NR_BLKDEV 2

/* Character devices table. */
PRIVATE const struct cdev *cdevsw[NR_CHRDEV] = {
    NULL, /* /dev/null */
    NULL, /* /dev/tty */
    NULL /* /dev/klog */
};

/* Block devices table. */
PRIVATE const struct bdev *bdevsw[NR_BLKDEV] = {
    NULL, /* /dev/ramdisk */
    NULL /* /dev/hdd */
};

```

Exemplo: Nanvix

```
/* Character device. */
struct cdev {
    int (*open)(unsigned);                                /* Open.      */
    ssize_t (*read)(unsigned, char *, size_t);           /* Read.      */
    ssize_t (*write)(unsigned, const char *, size_t);     /* Write.     */
    int (*ioctl)(unsigned, unsigned, unsigned);           /* Control.   */
    int (*close)(dev_t);                                 /* Close.     */
};

/* Block device. */
struct bdev {
    ssize_t (*read)(dev_t, char *, size_t, off_t);       /* Read.      */
    ssize_t (*write)(dev_t, const char *, size_t, off_t); /* Write.     */
    int (*readblk)(unsigned, struct buffer *);            /* Read block. */
    int (*writeblk)(unsigned, struct buffer *);           /* Write block. */
};
```

Software de E/S em espaço de usuário

- Composto por **bibliotecas** que são **ligadas aos programas** as quais fornecem **funções básicas** para realizar operações de E/S específicas

Exemplos de funções da biblioteca de E/S padrão do C

Função	Descrição
fread	Lê bytes de uma stream para um buffer em espaço de usuário
fwrite	Escreve dados de um buffer em espaço de usuário em uma stream
fgetc	Lê um caractere de uma stream
fputc	Escreve um caractere em uma stream
fgets	Lê uma string de uma stream
fputs	Escreve uma string em uma stream

!

Obrigado pela atenção!



Dúvidas? Entre em contato:

- marcio.castro@ufsc.br
- www.marciocastro.com



Distributed Systems Research Lab
www.lapesd.inf.ufsc.br