Entradas/Saídas (Input/Output)

Como são movimentados os dados?
Quais são os meios?

Dispositivos de (Input/Output)

Os dispositivos de E/S podem ser divididos em duas categorias:

- dispositivos de bloco;
- dispositivos de caractere.

Dispositivos de Bloco

Um dispositivo de bloco é aquele que armazena informações em blocos de tamanho fixo, cada um com seu próprio endereço. Os tamanhos comuns dos blocos variam de 512 a 65.536 bytes. Todas as transferências são em unidades de um ou mais blocos inteiros (consecutivos). A propriedade essencial de um dispositivo de bloco é que é possível ler ou escrever cada bloco independentemente de todos os outros. Discos rígidos, discos óticos (ex.: Blu-ray) e dispositivos USB são dispositivos de bloco comuns.

Dispositivos de Caractere

Um dispositivo de caracteres fornece ou aceita um fluxo de caracteres, sem considerar qualquer estrutura de bloco. Não é endereçável e não possui nenhuma operação de busca. Impressoras, interfaces de rede, mouses, e a maioria dos outros dispositivos que não são semelhantes a discos rígidos podem ser vistos como dispositivos de caracteres.

Dispositivos de E/S (I/O)

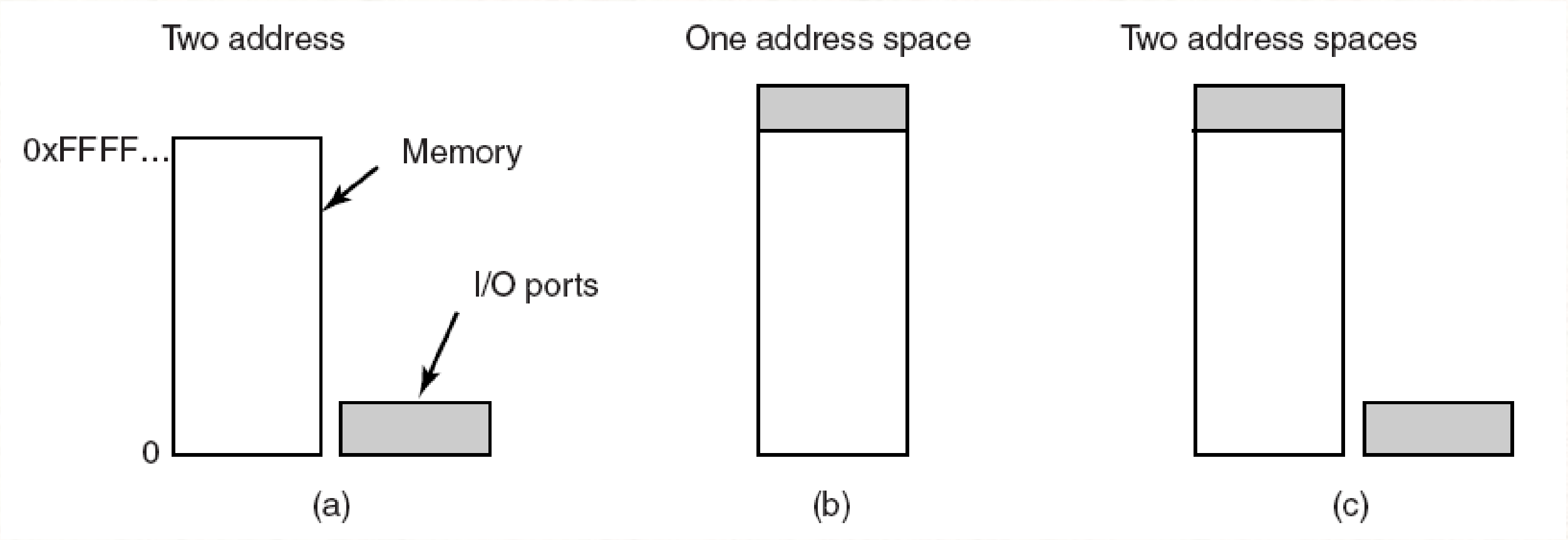
Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Scanner	400 KB/sec
Digital camcorder	3.5 MB/sec
802.11g Wireless	6.75 MB/sec
52x CD-ROM	7.8 MB/sec
Fast Ethernet	12.5 MB/sec
Compact flash card	40 MB/sec
FireWire (IEEE 1394)	50 MB/sec
USB 2.0	60 MB/sec
SONET OC-12 network	78 MB/sec
SCSI Ultra 2 disk	80 MB/sec
Gigabit Ethernet	125 MB/sec
SATA disk drive	300 MB/sec
Ultrium tape	320 MB/sec
PCI bus	528 MB/sec

Memória mapeada (E/S)

Cada controlador de dispositivo de E/S possui alguns registradores que são usados para comunicação com a CPU. Ao escrever nesses registros, o sistema operacional pode comandar o dispositivo para entregar dados, aceitar dados, ativar ou desativar, ou realizar alguma ação. Ao ler esses registros, o sistema operacional pode saber qual é o estado do dispositivo, se está preparado para aceitar um novo comando e assim por diante.

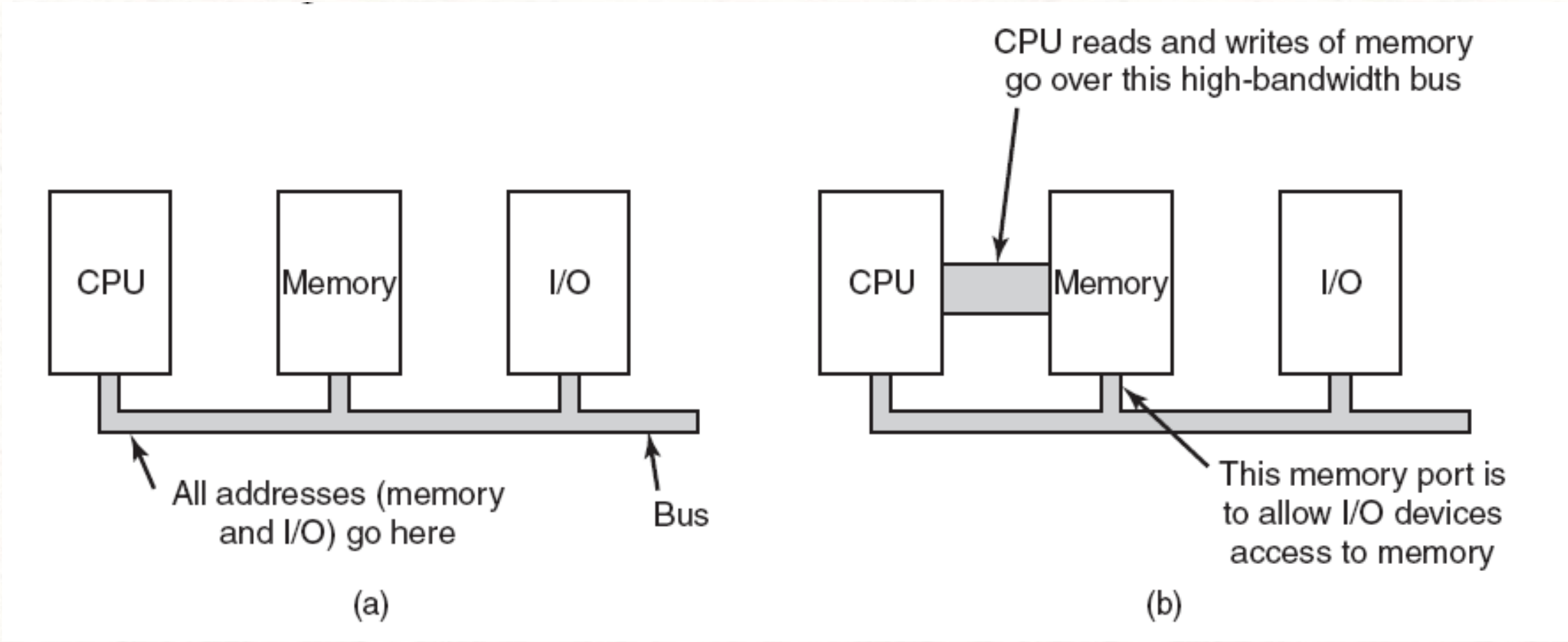
Além dos registros de controle, muitos dispositivos possuem um buffer de dados que o sistema operacional pode ler e gravar. Por exemplo, uma maneira comum de os computadores exibirem pixels na tela é ter uma RAM de vídeo, que é basicamente apenas um buffer de dados, disponível para programas ou sistema operacional para gravação.

Memória mapeada (E/S)



(a) Espaço de memória separados para E/S. (b) Memória Mapeada de E/S. (c) Hibrido.

Mapeamento - memória (E/S)



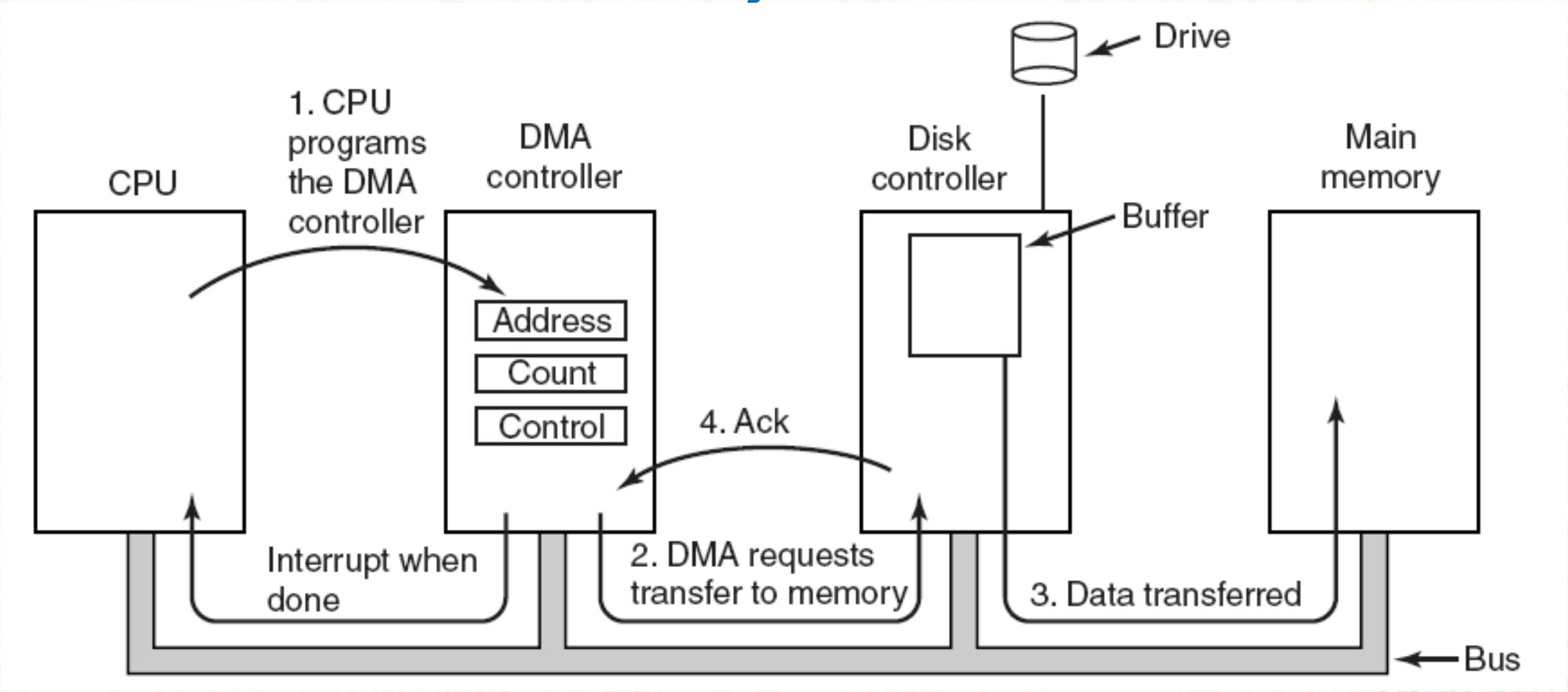
(a) Arquitetura single-bus .

(b) Arquitetura de memória Dual-bus.

Direct Memory Access (DMA)

Cada dispositivo pode possuir um DMA , porém, é comumente, um único controlador de DMA que está disponível (por exemplo, na placa-mãe) para regular as transferências para vários dispositivos, muitas vezes simultaneamente.
Não importa onde esteja fisicamente localizado, o controlador de DMA tem acesso ao barramento do sistema independente da CPU. Ele contém vários registros que podem ser escritos e lidos pela CPU.

Direct Memory Access (DMA)



Operação de uma transferência com DMA

Interrupções

Quando um dispositivo de E/S termina o trabalho dado a ele, ele causa uma interrupção (supondo que as interrupções tenham sido ativadas pelo sistema operacional). Ele faz isso com um sinal em uma linha de barramento que foi atribuída. Este sinal é detectado pelo chip do controlador de interrupção na placa-mãe, que decide o que fazer.

Se nenhuma outra interrupção estiver pendente, o controlador de interrupção manipula a interrupção imediatamente. No entanto, se outra interrupção estiver em andamento ou outro dispositivo tiver feito uma solicitação simultânea em uma linha de solicitação de interrupção de prioridade mais alta no barramento, o dispositivo será ignorado por enquanto. Neste caso, ele continua a afirmar um sinal intermitente no barramento até que ele seja atendido pela CPU.

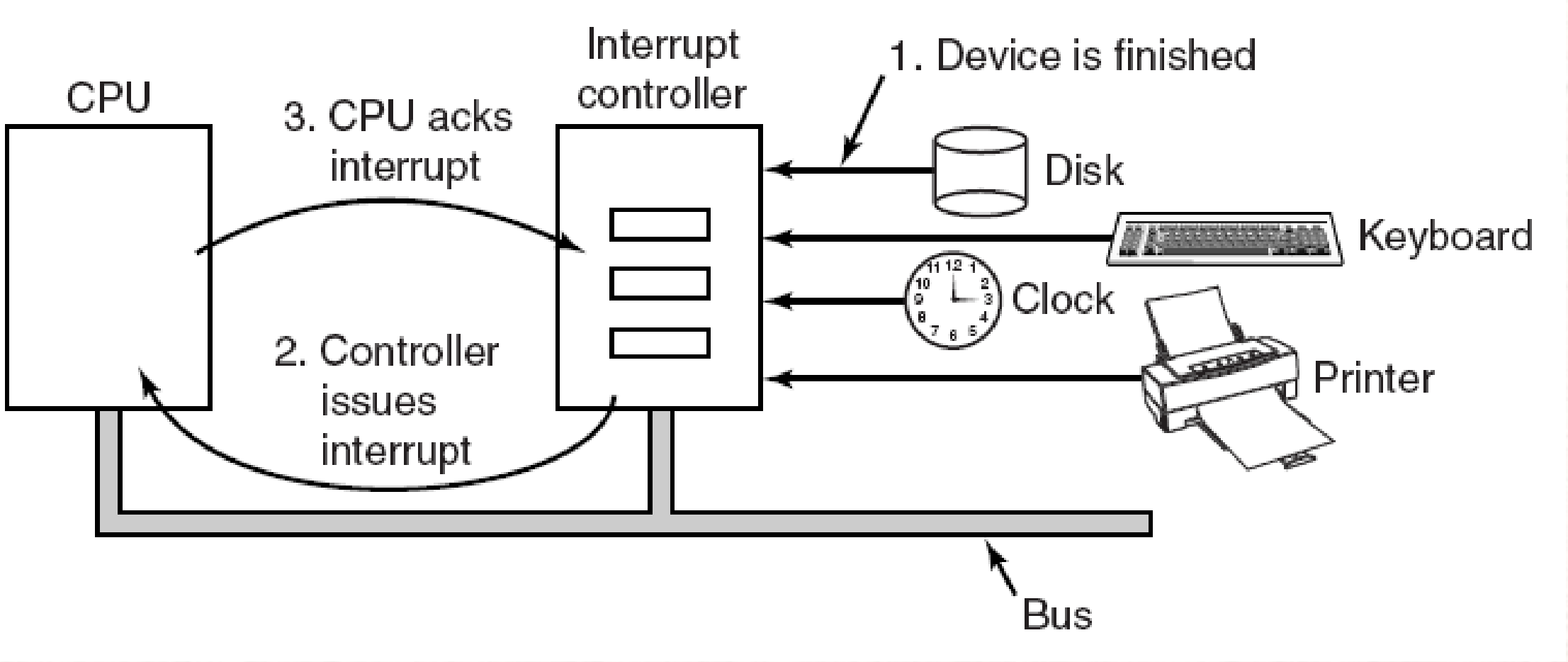
Controlador de Interrupções

Para lidar com a interrupção, o controlador coloca um número nas linhas de endereço especificando qual dispositivo quer atenção e ativa um sinal para interromper a CPU.

O sinal de interrupção faz com que a CPU pare o que está fazendo e comece a fazer outra coisa. O número nas linhas de endereço é usado como um índice em uma tabela chamada vetor de interrupção para buscar um novo contador de programa. Este contador de programa aponta para o início do procedimento de serviço de interrupção correspondente. Normalmente, interceptações e interrupções usam o mesmo mecanismo a partir desse ponto, geralmente compartilhando o mesmo vetor de interrupção. A localização do vetor de interrupção pode ser conectada à máquina ou pode estar em qualquer lugar na memória, com um registro da CPU (carregado pelo sistema operacional) apontando para sua origem.

Logo após a execução, o procedimento de interrupção de serviço confirma a interrupção gravando um determinado valor em uma das portas de E/S do controlador de interrupção. Esta confirmação diz ao controlador que está livre para emitir outra interrupção. Tendo a CPU que atrasar essa confirmação até que esteja pronta para lidar com a próxima interrupção, as condições de corrida envolvendo várias interrupções (quase simultâneas) podem ser evitadas.

Revisão de Interrupções



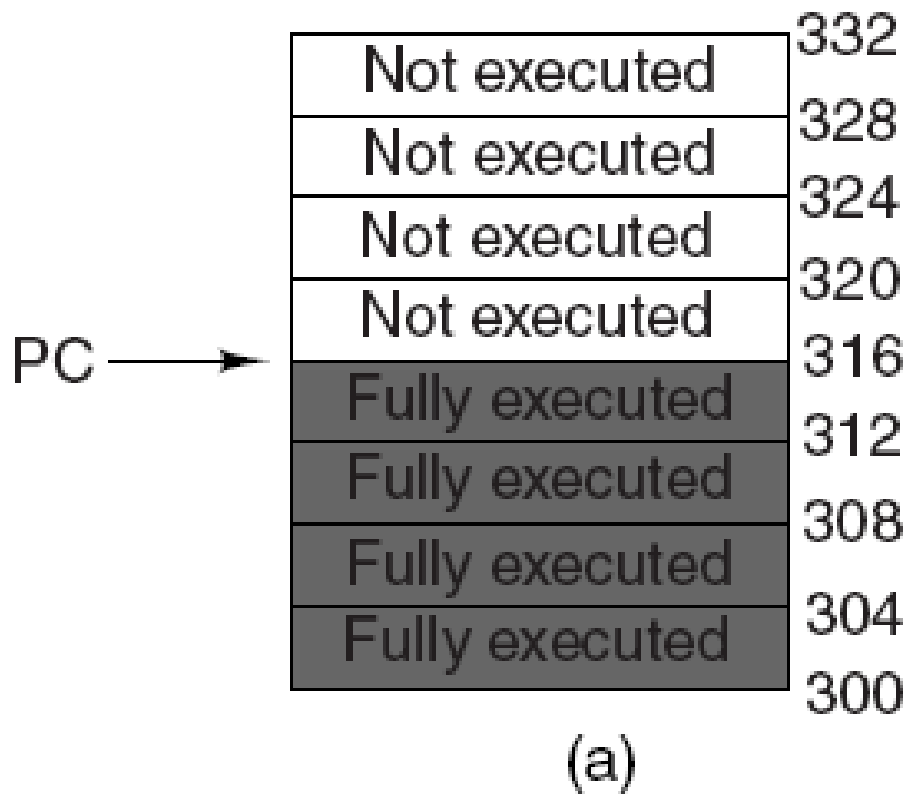
Como uma interrupção acontece. As conexões entre os dispositivos e o controlador de interrupção realmente usam linhas de interrupção no barramento, em vez de fios dedicados.

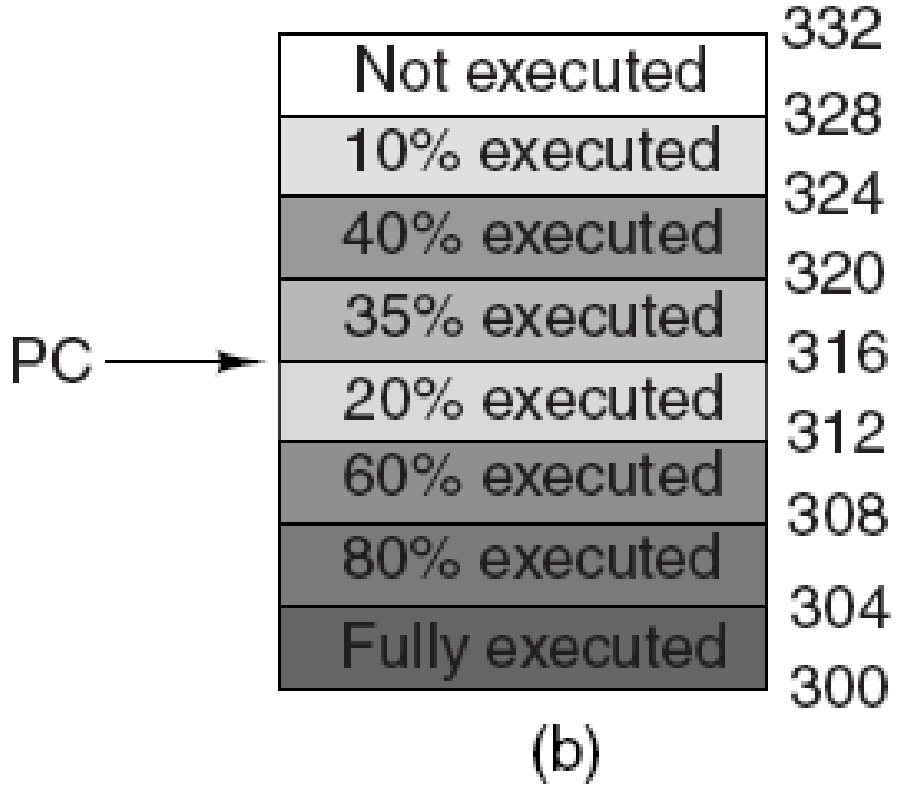
Interrupções precisas e imprecisas

Propriedades de uma **interrupção precisa**:

1. PC (Program Counter) é salvo em um local conhecido.
2. Todas as instruções antes da apontada pelo PC foram totalmente executadas.
3. Nenhuma instrução além daquela apontada pelo PC foi executada.
4. O estado de execução da instrução apontada pelo PC é conhecido

Interrupções precisas e imprecisas





(a) Uma interrupção Precisa

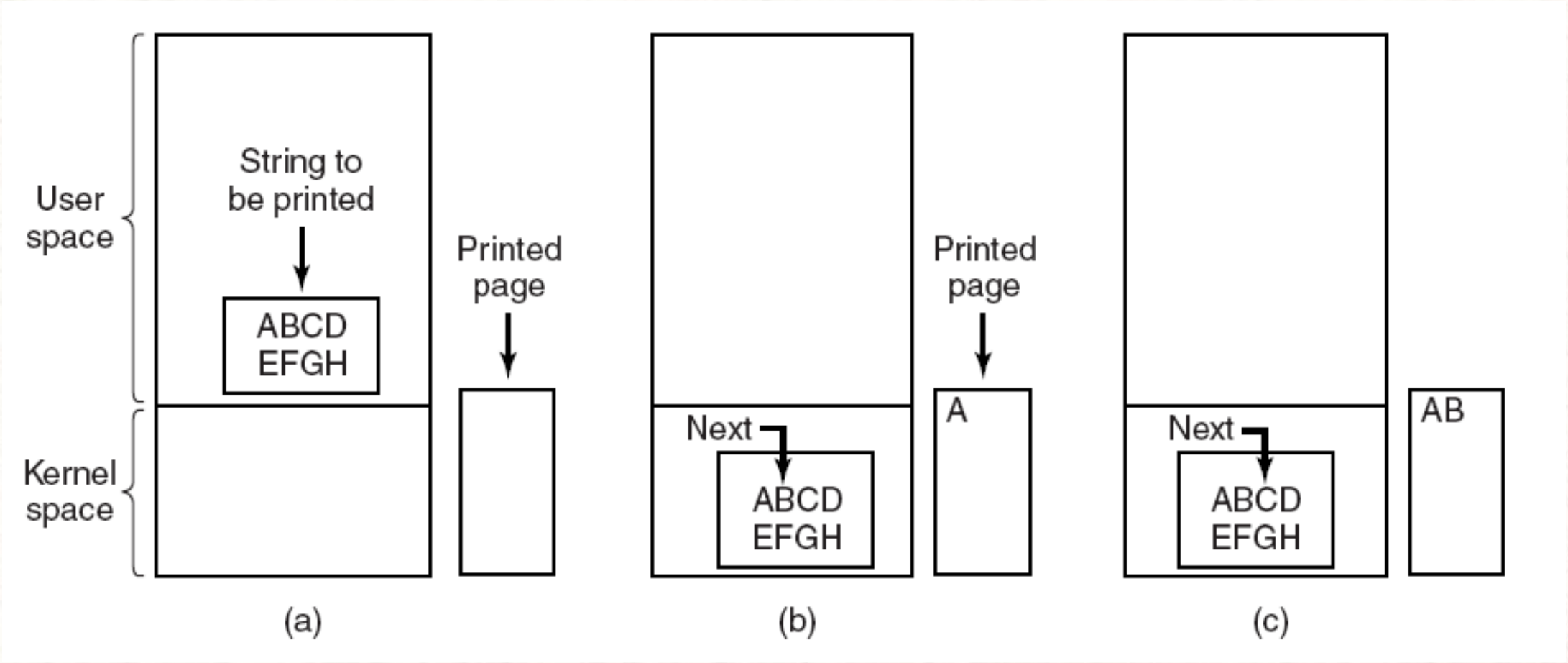
(b) Uma Interrupção Imprecisa

Execução de E/S

Existem três maneiras fundamentalmente diferentes que a E/S pode ser executada:

- E/S Programada (a cargo da CPU)
- Interrupção de E/S
- E/S baseadas em interrupção usando DMA

Programação de E/S



Etapas na impressão de uma string.

Programação de E/S

```
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( );
```

Escrevendo uma string para a impressora usando E / S programada.

Controlando Interrupções de E/S

Vamos considerar um caso em que a impressão em uma impressora, que não armazena caracteres no buffer, mas imprime cada um à medida que chega. Se a impressora puder imprimir, digamos 100 caracteres / seg, cada caractere demora 10 ms a imprimir. Isso significa que, após cada caractere ser gravado no registro de dados da impressora, a CPU ficará em um loop ocioso por 10 mseg, aguardando permissão para produzir o próximo caractere. Isso é mais do que tempo suficiente para fazer uma troca de contexto e executar algum outro processo para os 10 ms que seriam desperdiçados.

A maneira de permitir que a CPU faça outra coisa enquanto espera que a impressora fique pronta é usar interrupções. Quando a chamada do sistema para imprimir a string é feita, o buffer é copiado para o espaço do kernel, como mostramos anteriormente, e o primeiro caractere é copiado para a impressora assim que estiver disposto a aceitar um caractere. Nesse ponto, a CPU chama o Scheduler e algum outro processo é executado. O processo que solicitou que a cadeia seja impressa é bloqueado até que toda a cadeia seja impressa.

Quando a impressora imprime o caractere e está preparada para aceitar o próximo, ele gera uma interrupção. Essa interrupção interrompe o processo atual e salva seu estado. Em seguida, o procedimento de interrupção de serviço da impressora é executado.

Controlando Interrupções de E/S

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

(a)

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

(b)

Escrevendo uma string para a impressora usando E/S controlada por interrupção. (a) Código executado no momento em que a chamada do sistema de impressão é feita. (b) Procedimento de serviço de interrupção para a impressora.

E/S utilizando DMA

Aqui, a ideia é deixar o controlador DMA (Direct Memory Access) alimentar os caracteres para a impressora em um momento, sem que a CPU seja incomodada. Em essência, o DMA faz o trabalho de um programador de E/S, somente com o controlador de DMA fazendo todo o trabalho, em vez da CPU principal. Essa estratégia requer um hardware especial (o controlador de DMA), mas libera a CPU durante a E/S para realizar outro trabalho.

E/S utilizando DMA

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

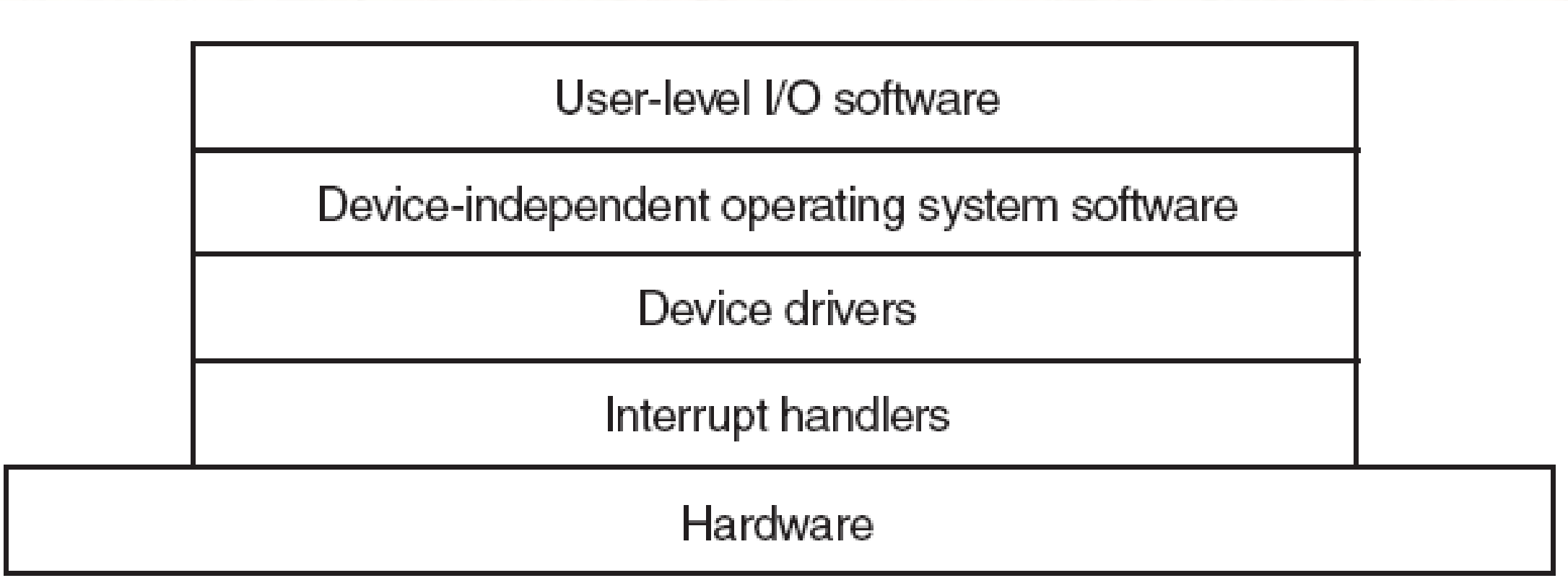
(a)

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

(b)

Imprimindo uma string usando DMA.
(a) Código executado quando a chamada do sistema de impressão é feita.
(b) Procedimento de interrupção de serviço

Camadas se Software de E/S



Camadas de um sistema de Software de E/S

Manipuladores de Interrupções (1)

Série de etapas que devem ser executadas no software após a interrupção do hardware ter sido concluída. Deve-se notar que os detalhes são altamente dependentes do sistema, portanto, algumas das etapas listadas abaixo podem não ser necessárias em uma máquina específica, e etapas não listadas podem ser necessárias. Além disso, as etapas que ocorrem podem estar em uma ordem diferente em algumas máquinas.

1. Salve todos os registradores (incluindo o PSW) que ainda não tenham sido salvos pelo hardware de interrupção.

2. Configure um contexto para o procedimento de serviço de interrupção. Isso pode envolver a configuração do TLB, da MMU e de uma tabela de páginas.

Manipuladores de Interrupções (2)

3. Configure uma pilha para o procedimento de serviço de interrupção.

4. Confirme o controlador de interrupção. Se não houver um controlador de interrupção centralizado, reative as interrupções.

5. Copie os registros de onde eles foram salvos (possivelmente alguma pilha) para a tabela de processos.

6. Execute o procedimento de serviço de interrupção. Ele extrairá informações dos registros do controlador de dispositivo de interrupção.

Manipuladores de Interrupções (3)

7. Escolha qual processo será executado em seguida. Se a interrupção tiver causado algum processo de alta prioridade que foi bloqueado para ficar pronto, ele pode ser escolhido para ser executado agora.

8. Configure o contexto MMU para o processo ser executado a seguir. Algumas configurações de TLB também podem ser necessárias.

9. Carregue os registros do novo processo, incluindo seu PSW.

10. Comece a executar o novo processo.

Device Drivers

Cada dispositivo de E/S conectado a um computador precisa de algum código específico do dispositivo para controlá-lo. Esse código, chamado de driver de dispositivo, é geralmente gravado pelo fabricante do dispositivo e entregue junto com o dispositivo. Como cada sistema operacional precisa de seus próprios drivers, os fabricantes de dispositivos geralmente fornecem drivers para vários sistemas operacionais populares.

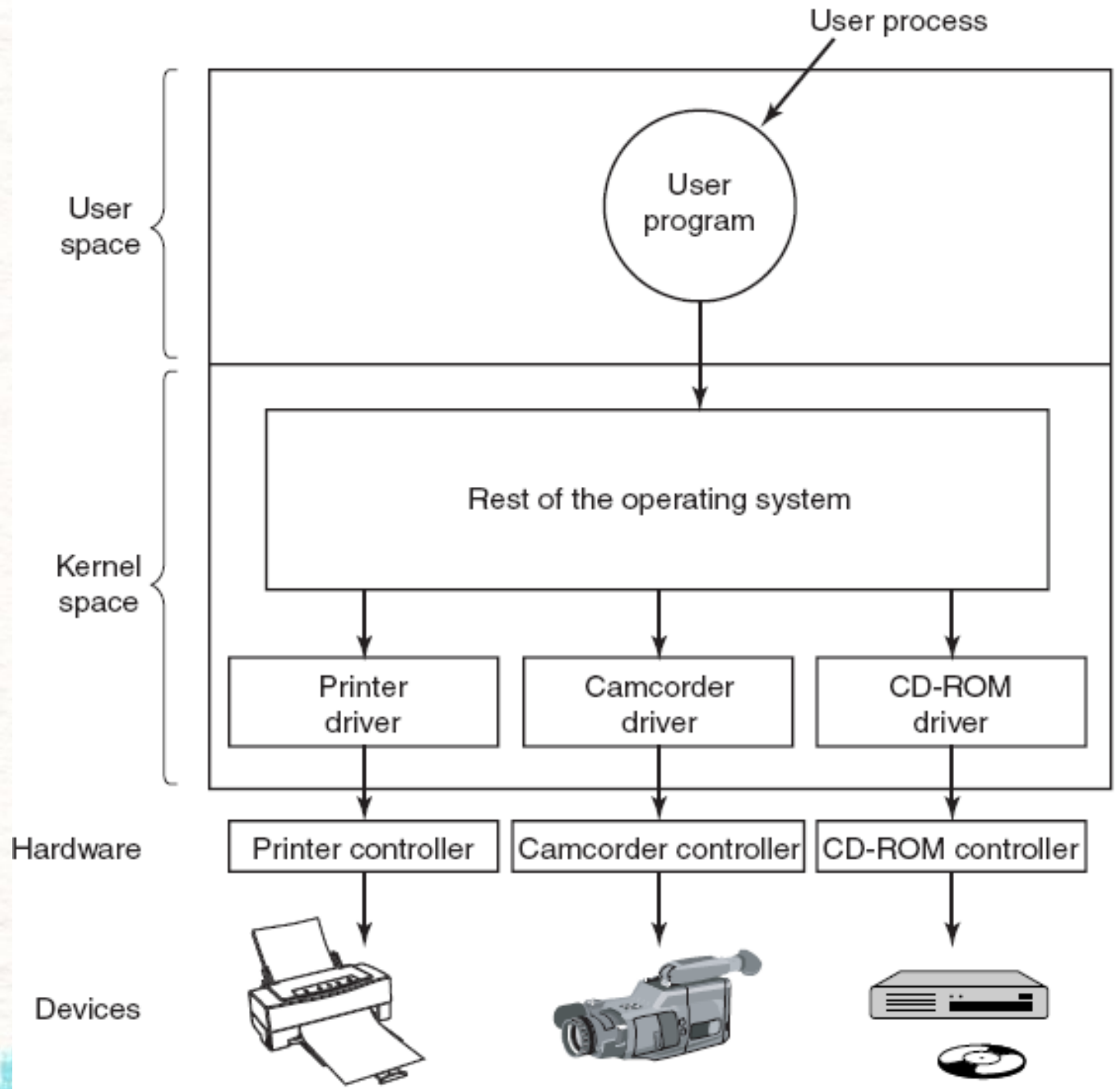
Cada driver de dispositivo normalmente lida com um tipo de dispositivo ou, no máximo, uma classe de dispositivos intimamente relacionados. Por exemplo, um driver de disco SCSI geralmente pode manipular vários discos SCSI de diferentes tamanhos e velocidades diferentes, e talvez um disco SCSI Blu-ray também. Por outro lado, um mouse e um joystick são tão diferentes que normalmente são necessários drivers diferentes. No entanto, não há restrição técnica para que um driver de dispositivo controle vários dispositivos não relacionados.

Device Drivers

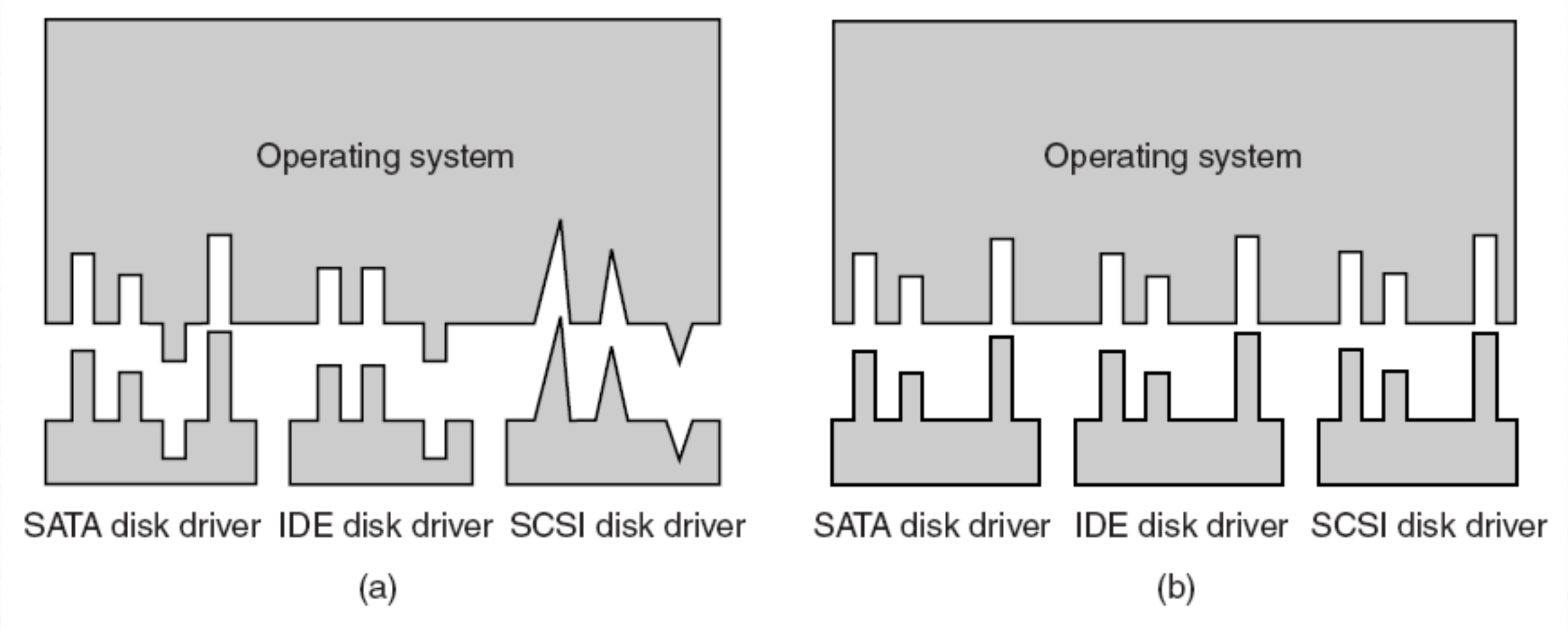
Às vezes, porém, dispositivos diferentes são baseados na mesma tecnologia subjacente. O exemplo mais conhecido é provavelmente o USB.
O truque é que os drivers USB são tipicamente empilhados, como uma pilha TCP / IP nas redes. Na parte inferior, geralmente em hardware, encontramos a camada de enlace USB (E / S serial) que lida com material de hardware como sinalização e decodificação de um fluxo de sinais para pacotes USB. Ele é usado por camadas superiores que lidam com os pacotes de dados e a funcionalidade comum para USB compartilhada pela maioria dos dispositivos. Além disso, finalmente, encontramos as APIs de camada superior, como as interfaces para armazenamento em massa, câmeras etc. Assim, ainda temos drivers de dispositivo separados, mesmo que eles compartilhem parte da pilha de protocolos.

Device Drivers

Posicionamento lógico de drivers de dispositivos.
Na realidade, toda a comunicação entre drivers e controladores de dispositivos passa pelo barramento.



Uniformizando Interfaces de Device Drivers



(a) Sem uma interface padrão

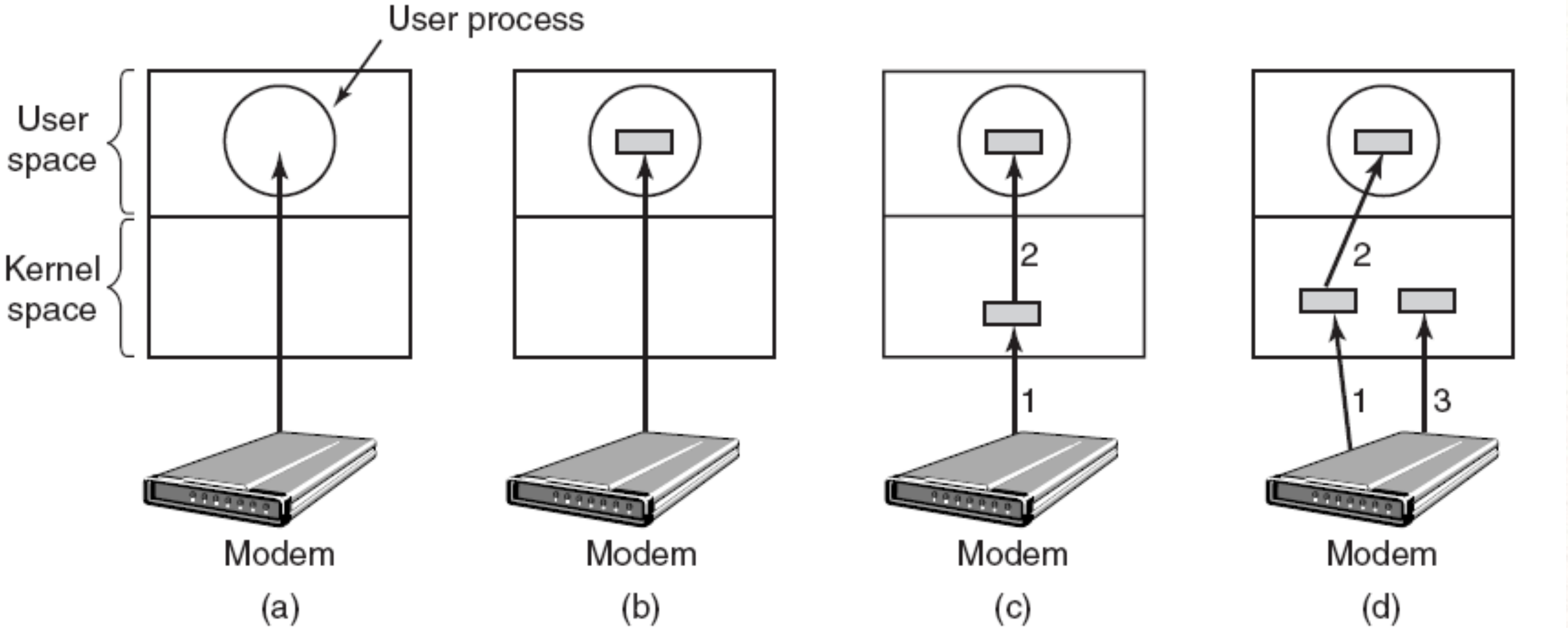
(b) Com uma interface padrão

Buffer

O termo buffer designa uma memória intermediária auxiliar cuja finalidade é a de agilizar as tarefas e guardar os dados antes de serem usados.

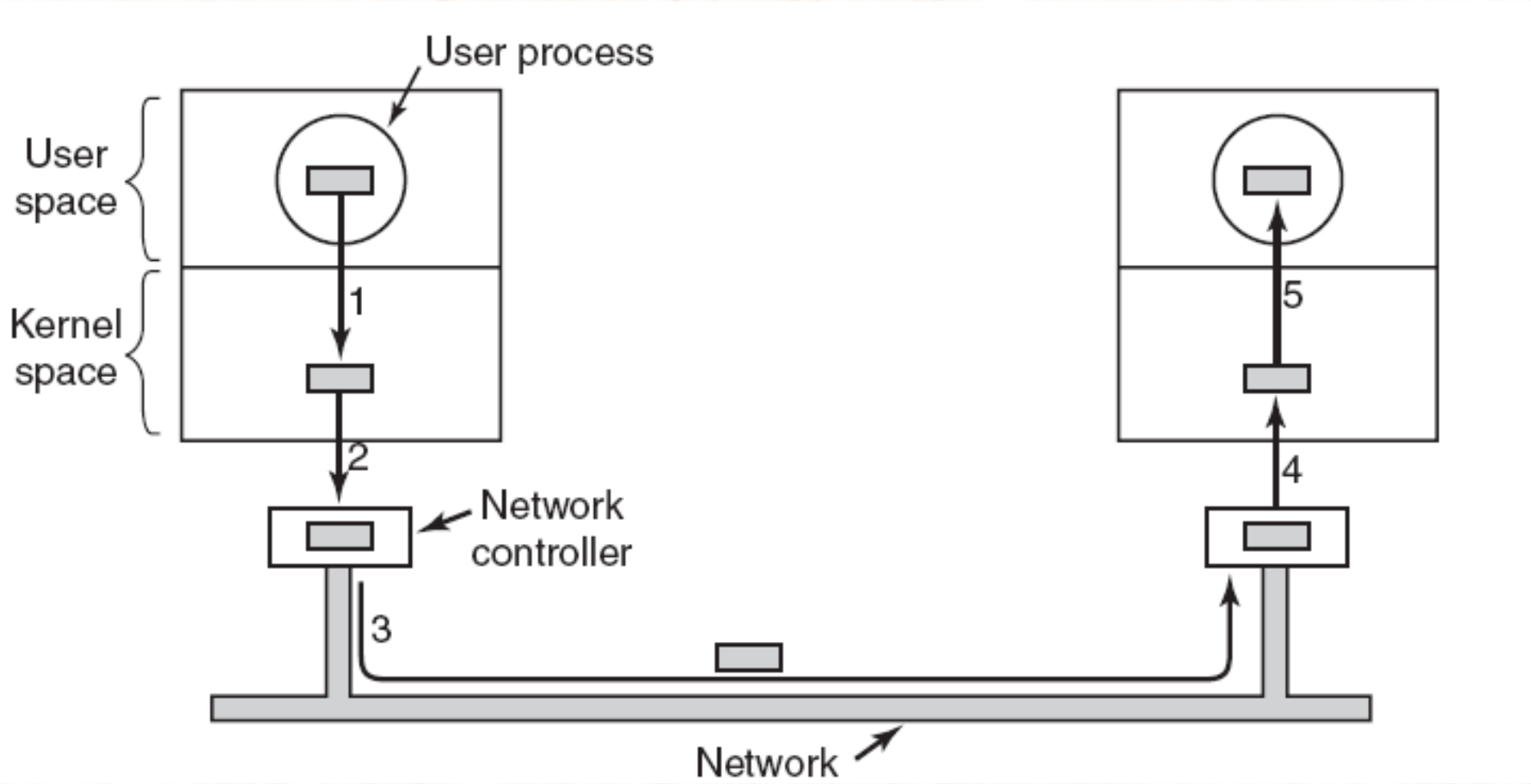
Um buffer contém dados que são armazenados por um curto período de tempo, tipicamente na memória do computador (RAM). A finalidade de um buffer é guardar os dados antes de serem usados. Por exemplo, quando transfere um arquivo de áudio ou vídeo a partir da Internet, pode ser carregado os primeiros 20% no buffer e, em seguida, começar a reproduzir. Enquanto o parte do arquivo é reproduzido, o computador transfere continuamente o resto desse arquivo e armazena-lo no buffer. Como o arquivo está sendo reproduzido a partir do buffer, e não diretamente a partir da Internet, há menos probabilidade de que o áudio ou o vídeo pare quando há um congestionamento da rede.

Buffering (1)



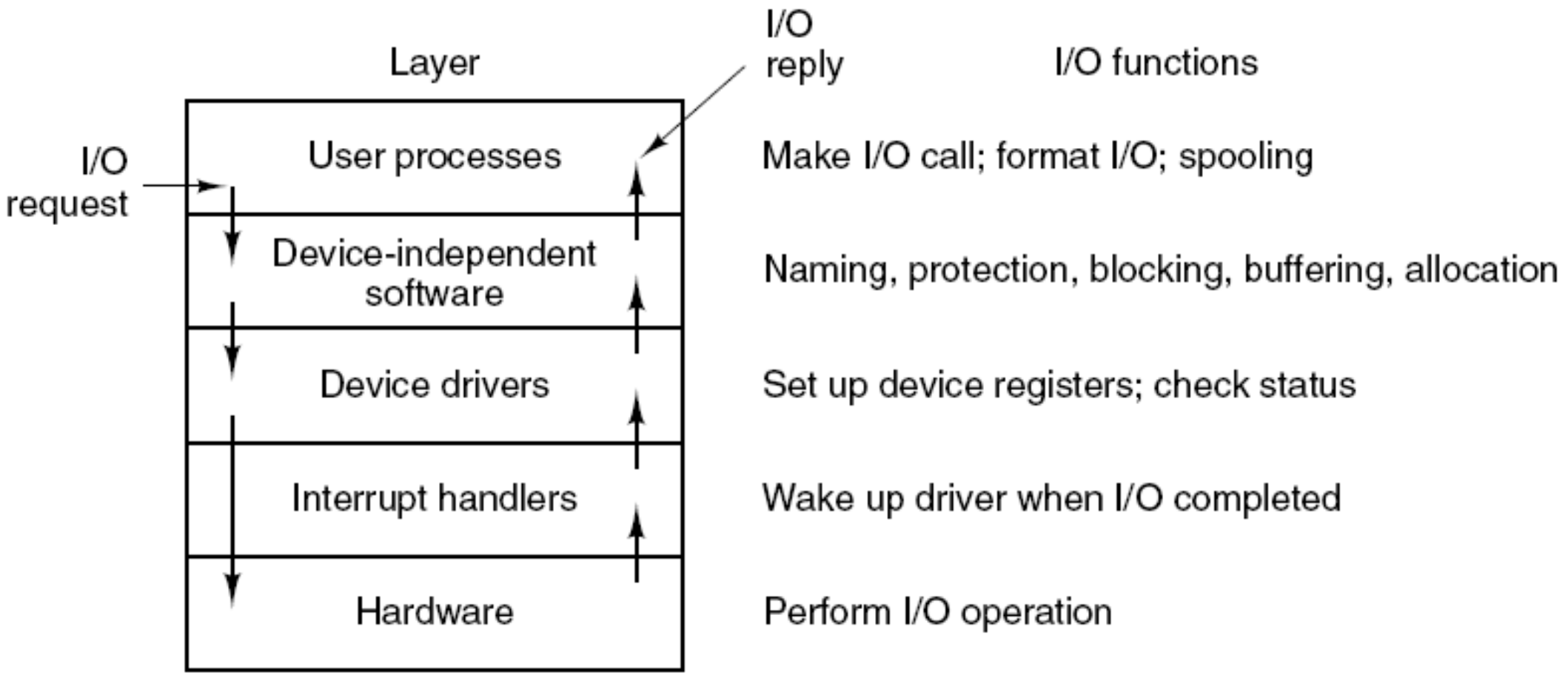
(a) Entrada sem Buffer. (b) Buffer no espaço do usuário. (c) Buffer no kernel seguido de cópia para o espaço do usuário. (d) Buffer duplo no kernel.

Buffering (2)



Situação de transferência de pacotes em rede

Software de E/S - espaço do usuário

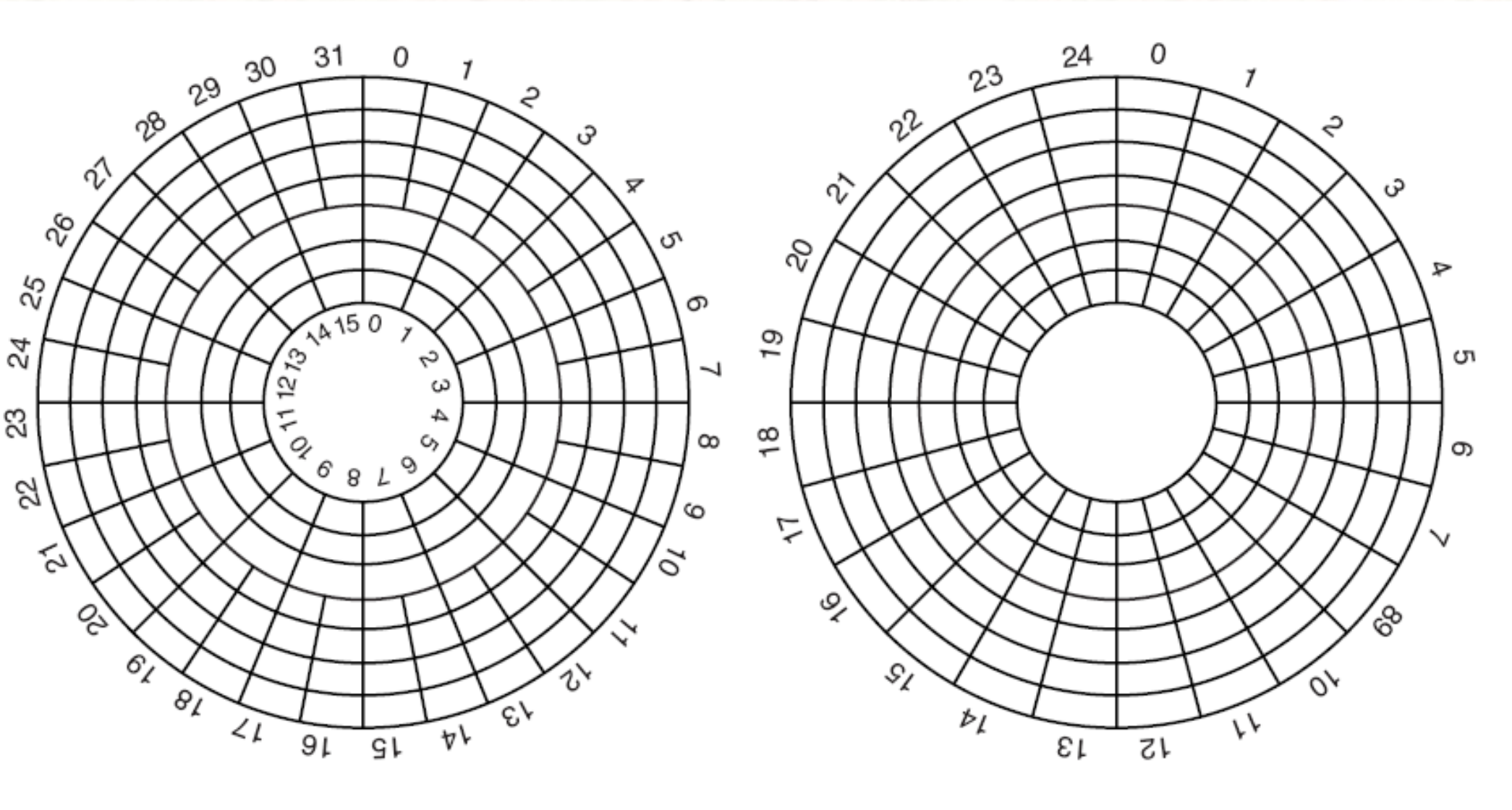


Camadas do sistema de E/S e as principais funções de cada camada.

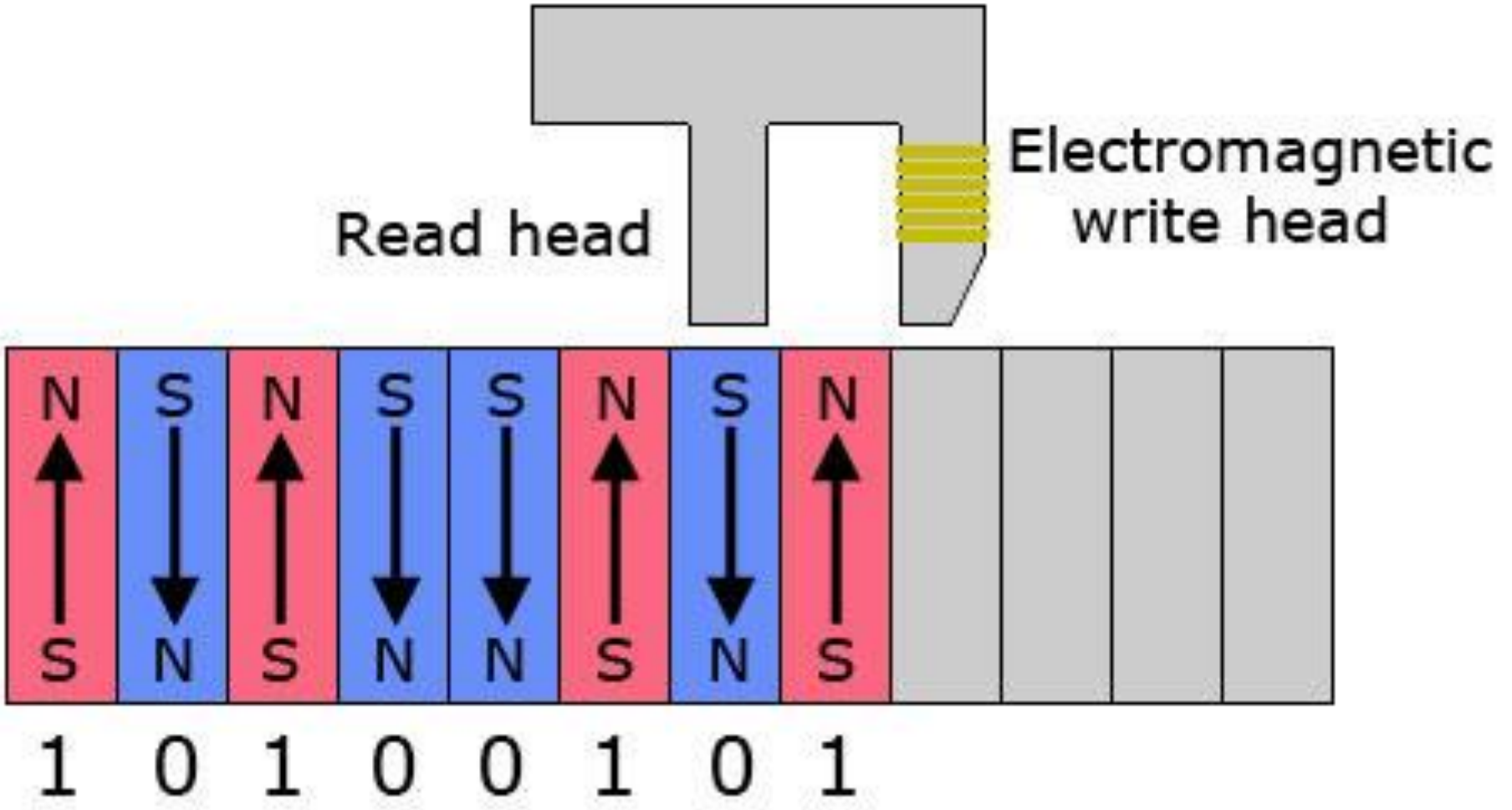
Discos magnéticos

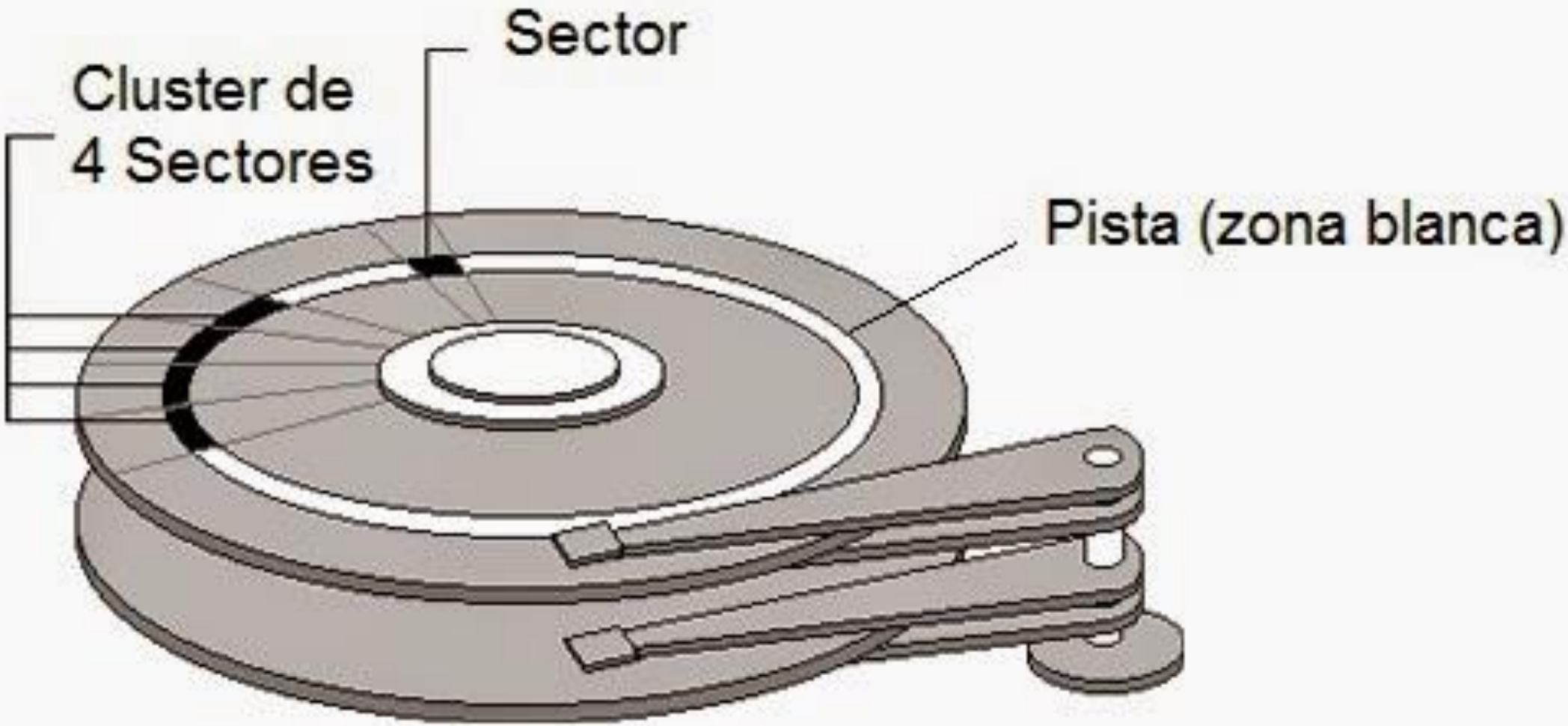
Parameter	IBM 360-KB floppy disk	WD 18300 hard disk
Number of cylinders	40	10601
Tracks per cylinder	2	12
Sectors per track	9	281 (avg)
Sectors per disk	720	35742000
Bytes per sector	512	512
Disk capacity	360 KB	18.3 GB
Seek time (adjacent cylinders)	6 msec	0.8 msec
Seek time (average case)	77 msec	6.9 msec
Rotation time	200 msec	8.33 msec
Motor stop/start time	250 msec	20 sec
Time to transfer 1 sector	22 msec	17 μsec

Discos magnéticos



Hard drive read/write head




Sector
Cluster de
4 Sectores
Pista (zona blanca)

Referências Bibliográficas

- TANENBAUM, Andrew S., BOSS, Herbert. **Sistemas Operacionais Modernos,** Pearson - 4ª ed., 2016.
- SILBERSCHATZ, A., GALVIN, P.B., GAGNE, G. **Fundamentos de Sistemas Operacionais**, Ed. LTC, 8ª ed., 2011
- DEITEL, H.M.; DEITEL, P.J.; CHOFFNES, D.R. **– Sistemas Operacionais**. Prentice Hall, Tradução da 3ª ed., 2005
- DEITEL, H.M.; DEITEL, P.J. **– C How to Program**. Prentice Hall, Tradução da 3ª ed., 2001
- MIZRAHI, Victorine Viviane. **Treinamento em Linguagem C – Curso Completo módulos 1 e 2**, Ed. Person Education.