(Cap. 5 Modern Operating Systems)

Input/Output

1 Princípios de hw de I/O
2 Princípios de sw de I/O
3 Camadas de sw de I/O
4 Discos
5 Relógio
6 Terminais orientados a caracteres
7 Interfaces gráficas
8 Terminais de rede
9 Gestão de consumo de energia

Princípios de hw de I/O

Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Telephone channel	8 KB/sec
Dual ISDN lines	16 KB/sec
Laser printer	100 KB/sec
Scanner	400 KB/sec
Classic Ethernet	1.25 MB/sec
USB (Universal Serial Bus)	1.5 MB/sec
Digital camcorder	4 MB/sec
IDE disk	5 MB/sec
40x CD-ROM	6 MB/sec
Fast Ethernet	12.5 MB/sec
ISA bus	16.7 MB/sec
EIDE (ATA-2) disk	16.7 MB/sec
FireWire (IEEE 1394)	50 MB/sec
XGA Monitor	60 MB/sec
SONET OC-12 network	78 MB/sec
SCSI Ultra 2 disk	80 MB/sec
Gigabit Ethernet	125 MB/sec
Ultrium tape	320 MB/sec
PCI bus	528 MB/sec
Sun Gigaplane XB backplane	20 GB/sec

Taxa de acesso de alguns dispositivos típicos

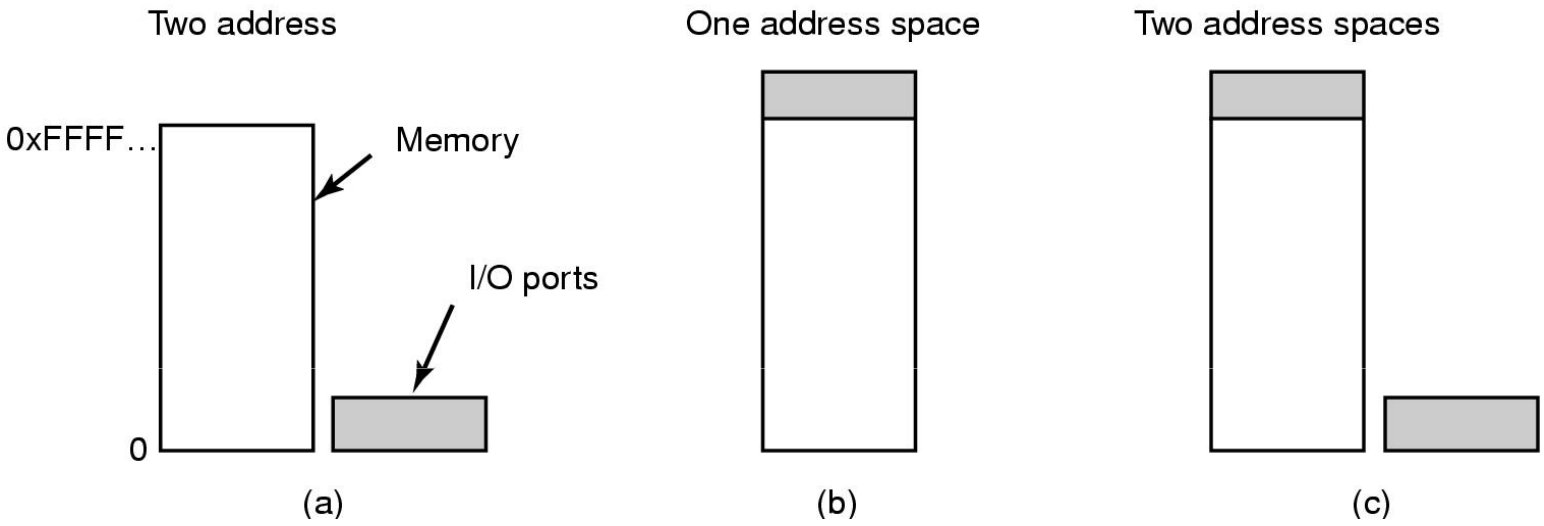
Controladores de dispositivos (device controllers)

- Componentes de um dispositivo de I/O
  - mecânico
  - eletrônico
- O componente eletrônico é o controlador do dispositivo
  - Pode gerir múltiplos dispositivos
- Tarefas do controlador
  - Converter sequencia de bits para blocos de bytes
  - Executar correção de erro, se necessário
  - Disponibilizar os dados em memória principal

Controladores de dispositivos (device controllers)

- Um dispositivo (device) comunica-se com um computador através de um ponto de conexão: **portas (I/O ports)**
- Uma porta de I/O pode ser serial ou paralela e, normalmente, consiste de 4 registradores:
  - Data-in: lido pelo controlador
  - Data-out: escrito pelo controlador
  - Status
  - Control: enviar comandos ou mudar o estado ou modo de um dispositivo (por exemplo, verificação de paridade, tamanho da palavra etc)

Memory-Mapped I/O (1)

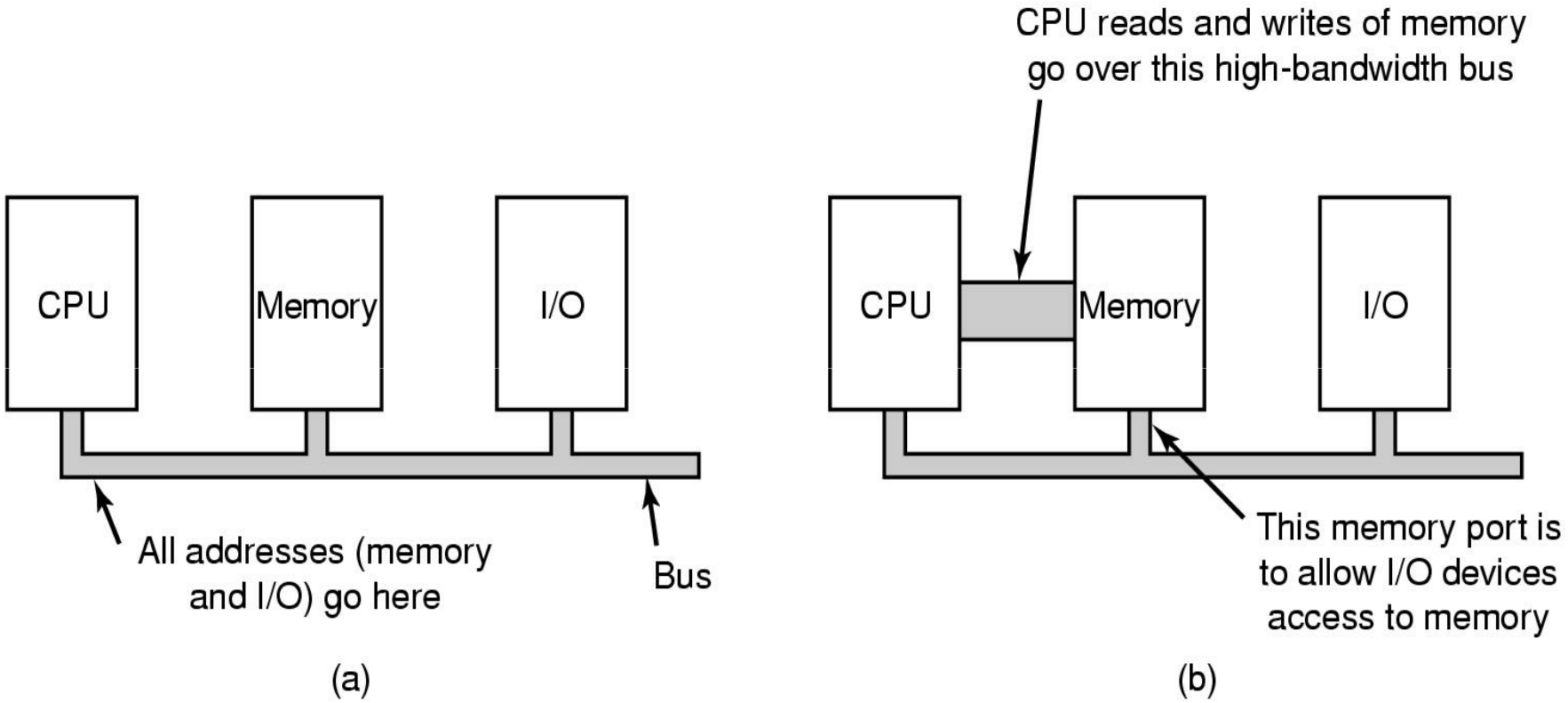


- (a) Espaço de I/O e memória separados
- (b) Memory-mapped I/O
- (c) Híbrido (pe: controlador de placas gráficas)

Memory-Mapped I/O (2)

- Espaço de I/O e memória separados requer instruções especiais para I/O: IN and OUT, que não acedem a memória e, sim, o espaço de I/O.
- Portanto, código para device drivers precisa ser escrito em “assembly”
- `MOV R0,4 ≠ IN R0,4`

Memory-Mapped I/O (3)



(a) Arquitetura com barramento único

(b) Arquitetura com barramento duplo

Memory-Mapped I/O (4)

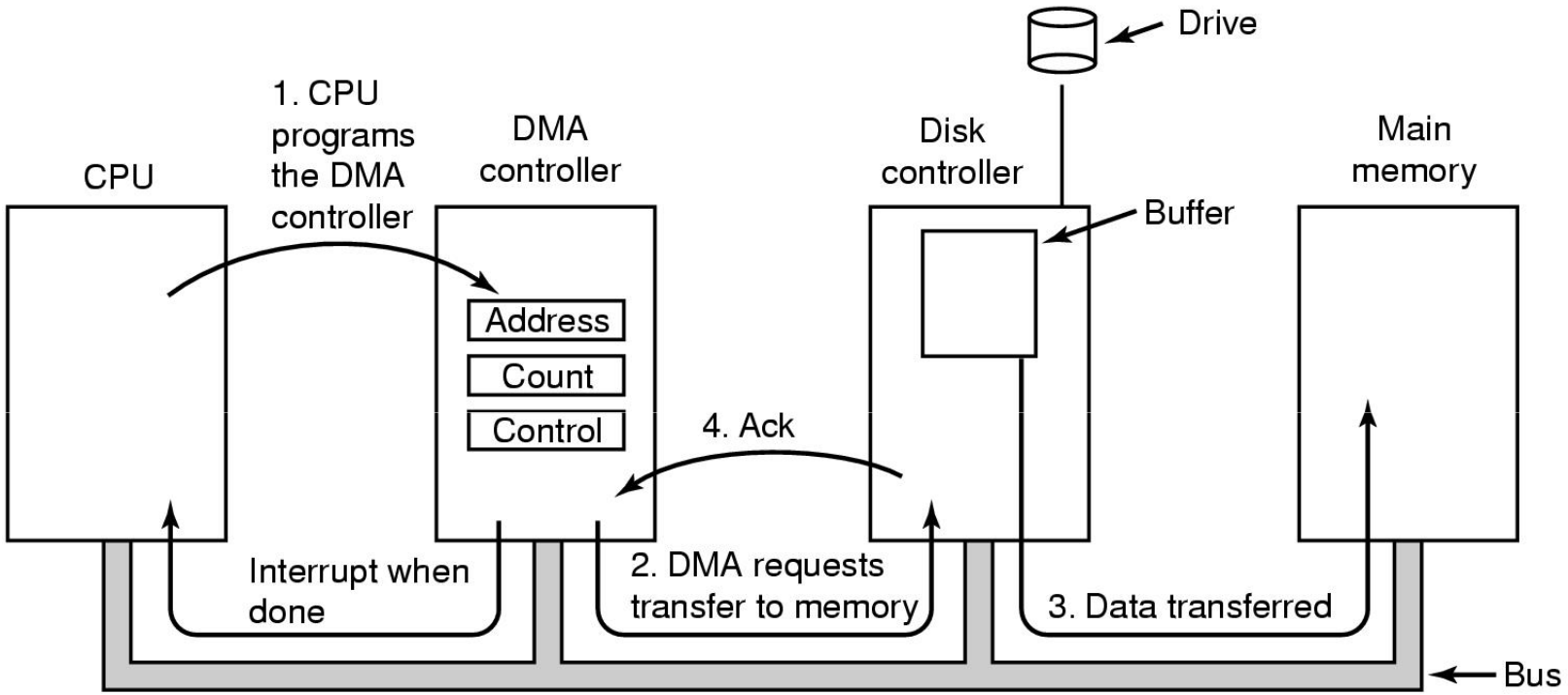
- Vantagens:
  - Registradores de controle são apenas variáveis em memória
  - Device driver pode ser escrito normalmente em C e não precisa de código “assembly”
- Desvantagens:
  - Execução errônea na presença de “caching”

```
Loop:   TEST PORT_4  // check if port 4 is 0
        BEQ Ready    // if it is 0, goto Ready
        BRANCH Loop // otherwise, cont testing
Ready: .....
```

Memory-Mapped I/O (5)

- Solução:
  - Desabilitar caching...
  - Mas adiciona complexidade extra no hw e sw
- Outra Desvantagem:
  - Em sistemas que utilizam arquitetura com memória com duplo barramento (que é comum nos procs hoje em dia), dispositivos não vêem os endereços colocados no barramento extra.
- Solução: ordenar acessos enviando primeiro todos os endereços para a memória. Se a memória não responder, enviar para outros barramentos.

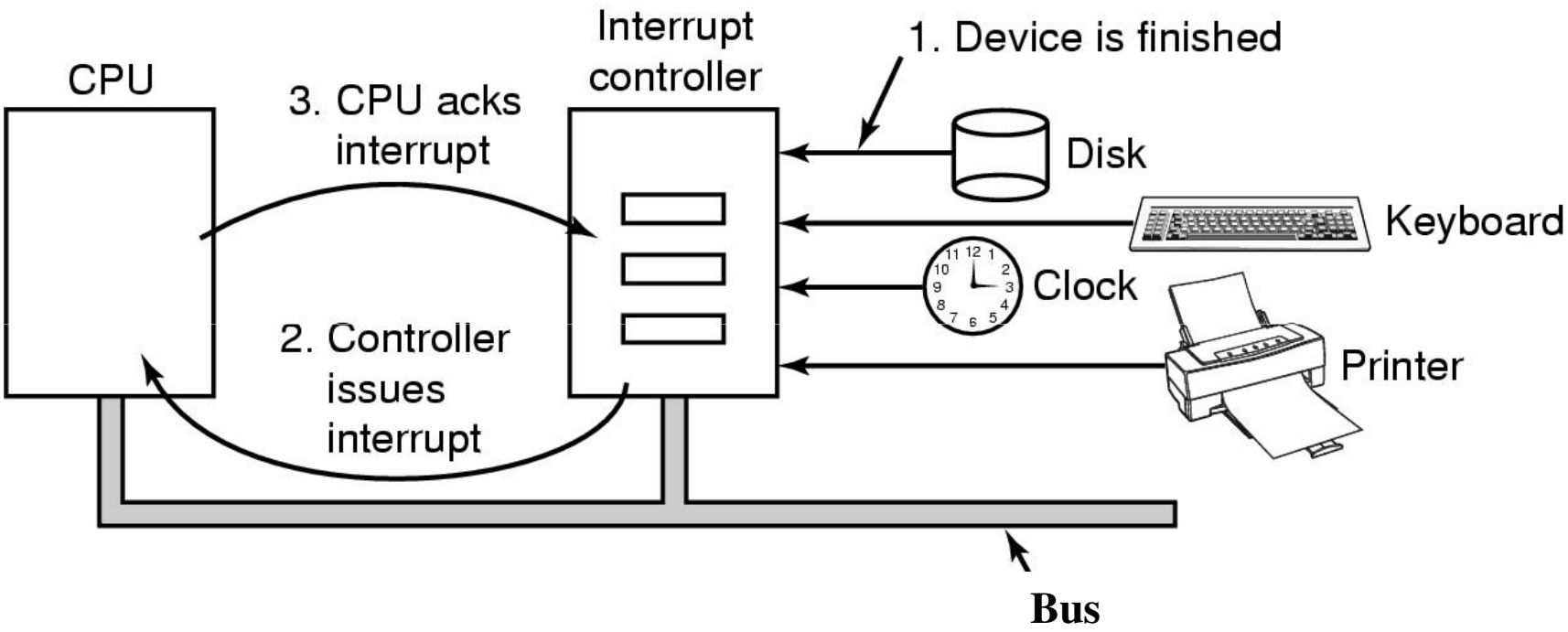
Direct Memory Access (DMA)



Cpu uses read or write control lines, data lines to communicate address of the I/O device and the starting location in memory to read from or write to, number of words to be read or written via the data lines and stored in the Count register

Transferência via controlador DMA

Revisão de Interrupções



Como ocorre uma inerrupção? Conexões entre os dispositivos e o controlador de interrupções utilizam linhas de interrupção no barramento invés de linhas dedicadas.

Princípios de sw de I/O
Objetivos (1)

- Independência do dispositivo
  - programas deveriam poder aceder qualquer dispositivo de I/O
  - Sem precisar especificar o dispositivo
    - (floppy, hard drive, or CD-ROM)
    - Pe: sort < input > output
- Uniform naming
  - nome de ficheiro ou dispositivo: string ou inteiro
  - Nao dependente de máquina
- Manipulação de erros
  - Tão próximo do hw quanto possível

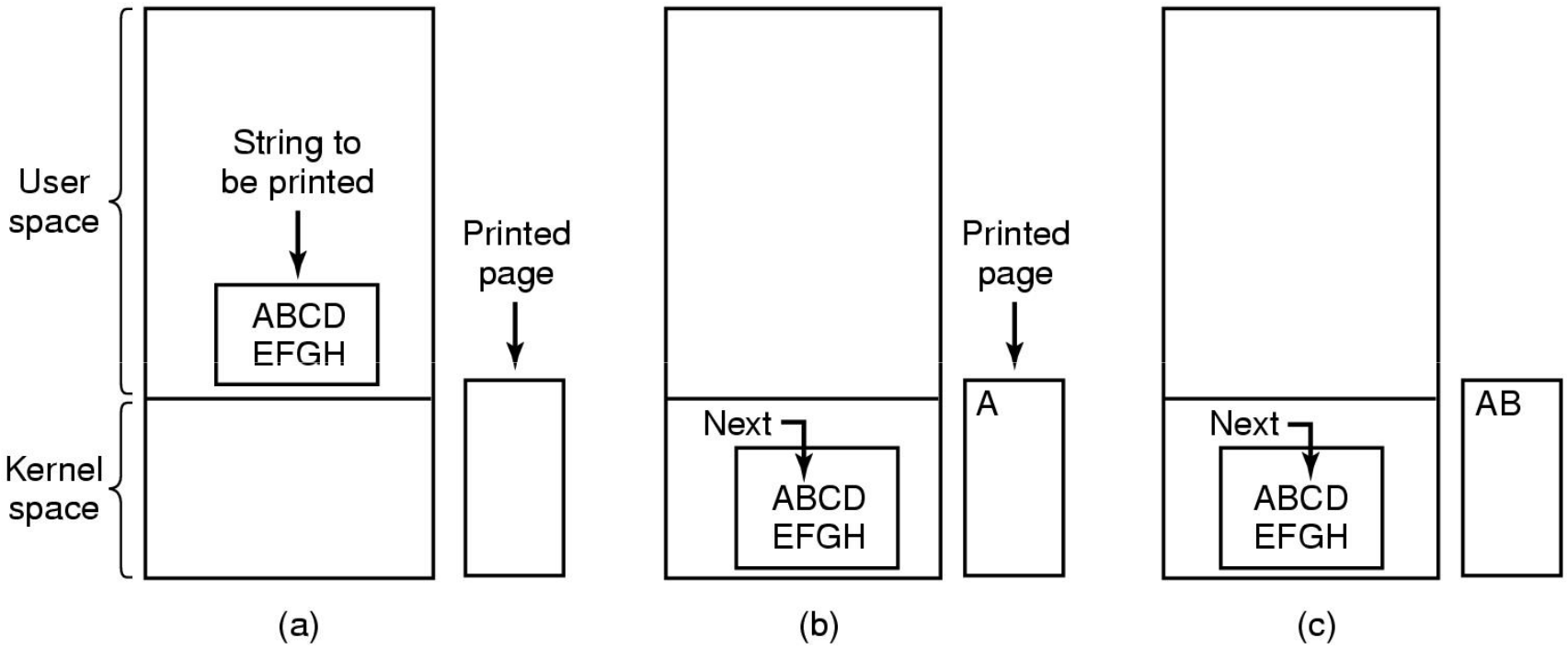
Princípios de sw de I/O
Objetivos (2)

- Transf síncronas vs. assíncronas
  - blocking vs. interrupt-driven
- Buffering
  - Dados que vêm do dispositivo não podem ser armazenados no destino final sem uma inspeção prévia
- Dispositivos compartilháveis vs. dedicados
  - Discos são compartilháveis
  - Fitas não são compartilháveis

Princípios de sw de I/O (3)

- 3 formas de executar I/O
  - I/O programado (programmed I/O)
  - Interrupt-driven
  - Utilização de DMA

Programmed I/O (1)



Passos para imprimir uma string: CPU faz todo o trabalho

Programmed I/O (2)

```
copy_from_user(buffer, p, count);                /* p is the kernel bufer */
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( );
```

Imprimindo uma string usando programmed I/O

Interrupt-Driven I/O

```
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)

- Imprimindo uma string usando interrupt-driven I/O
  - (a) Código executado quando a chamada de sistema para imprimir é feita
  - (b) Procedimento de serviço de interrupção

I/O Using DMA

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

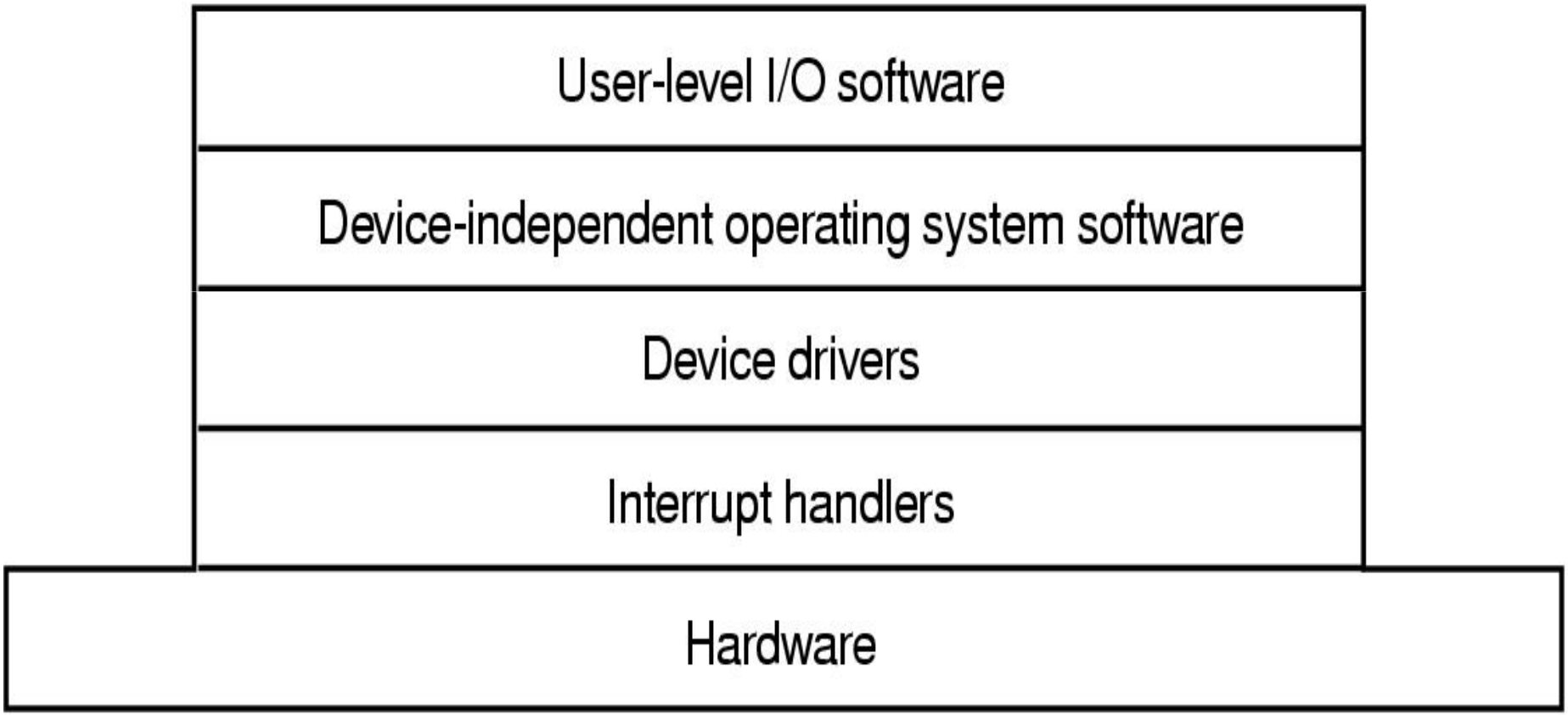
(a)

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

(b)

- Programmed I/O, onde DMA faz todo o trabalho
- Imprimindo usando DMA
  - (a) código executado quando a chamada de sistema é feita
  - Procedimento de serviço de interrupçoes

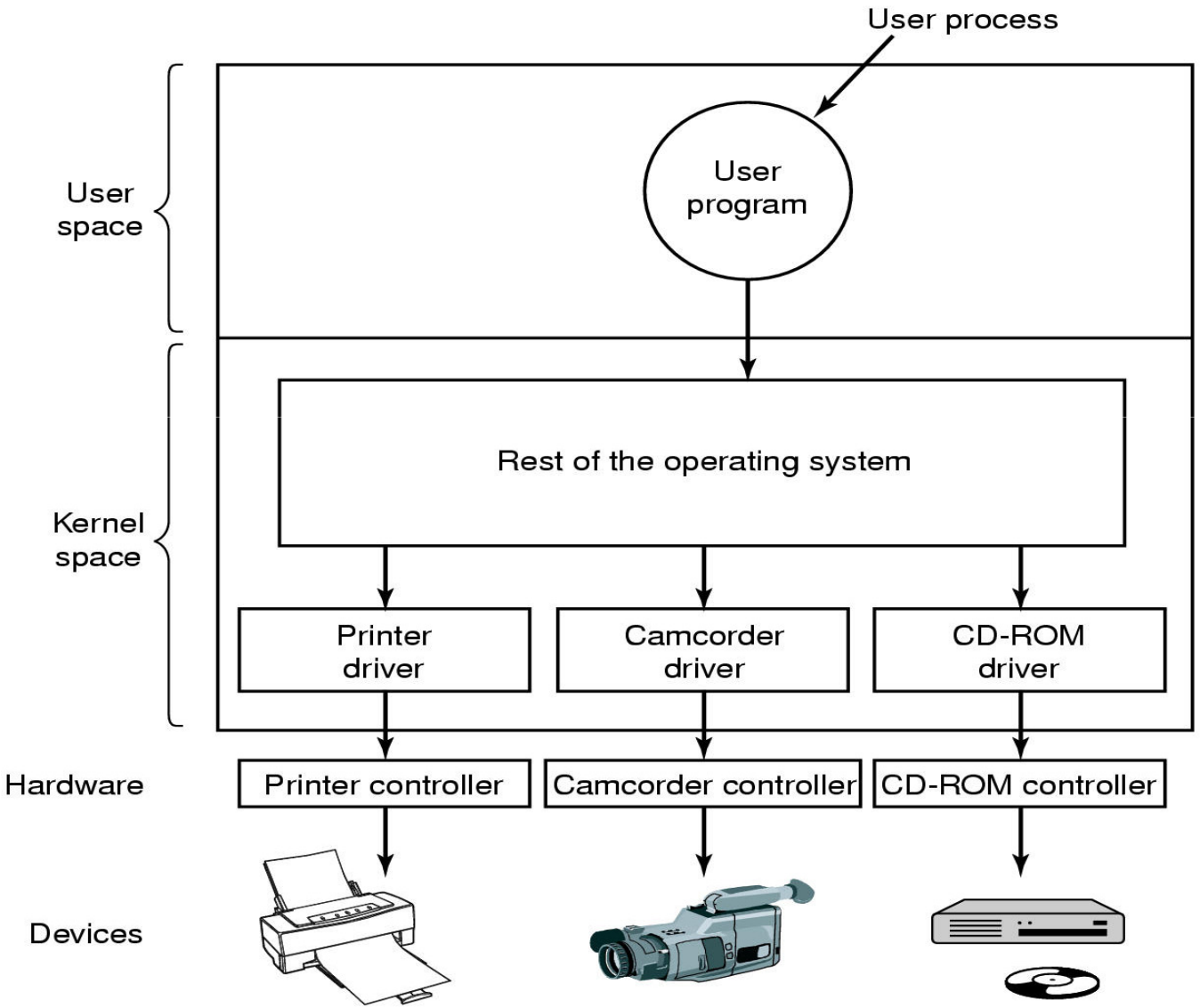
Camadas de sw de I/O



Interrupt Handlers (1)

- Passos que devem ser executados em sw após a interrupção
  1. Guardar regs ainda não guardados pelo hw de interrupção
  2. Preparar contexto para o serviço de interrupção
  3. Preparar pilha para o serviço de interrupção
  4. Ack controlador de interrupções, re-habilitar interrupções
  5. Copiar registos de onde foram salvos para a tabela do processo
  6. Executar o serviço de interrupção
  7. Preparar o contexto da MMU para o próximo processo
  8. Carregar registos do novo processo
  9. Iniciar novo processo

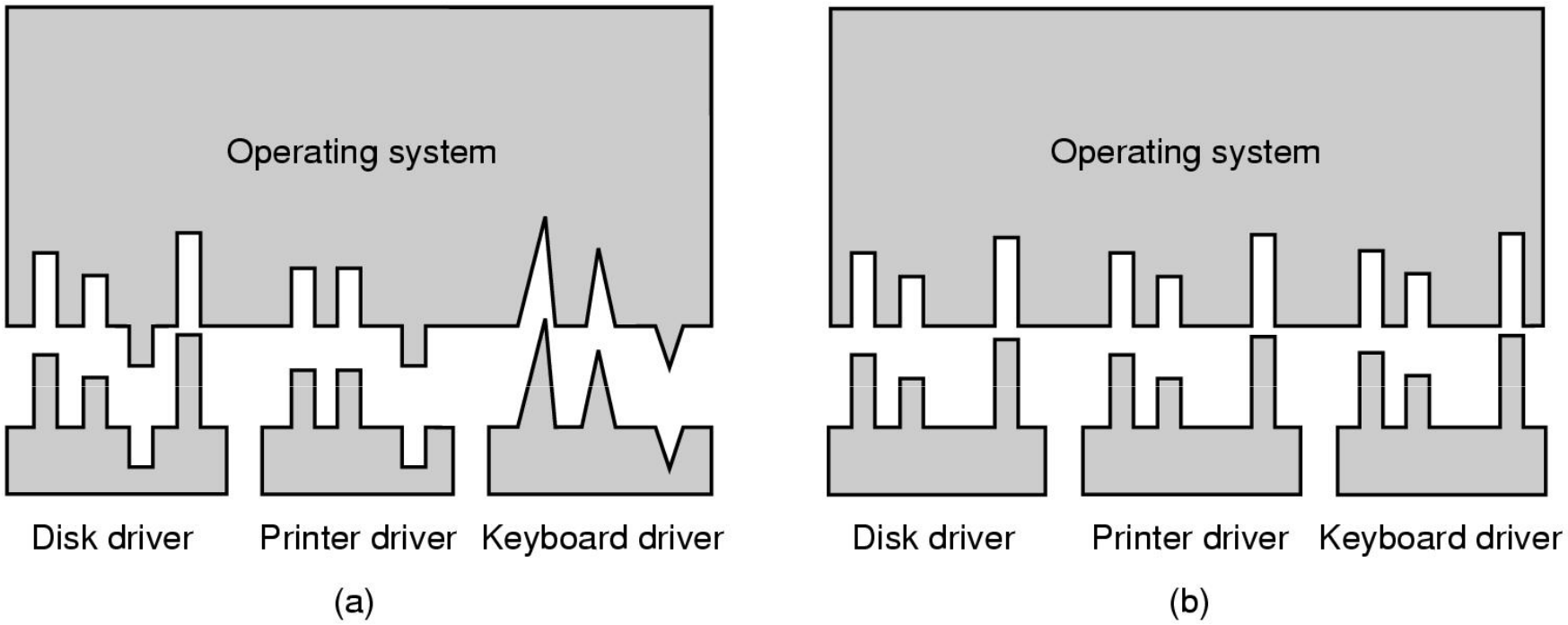
Device Drivers



Funções do sw de I/O (1)

Interface Uniforme para os “drivers” dos dispositivos (“device drivers”)
Buffering
Reportar erros
Alocação e liberação de dispositivos dedicados
Prover tamanho de bloco independente do dispositivo

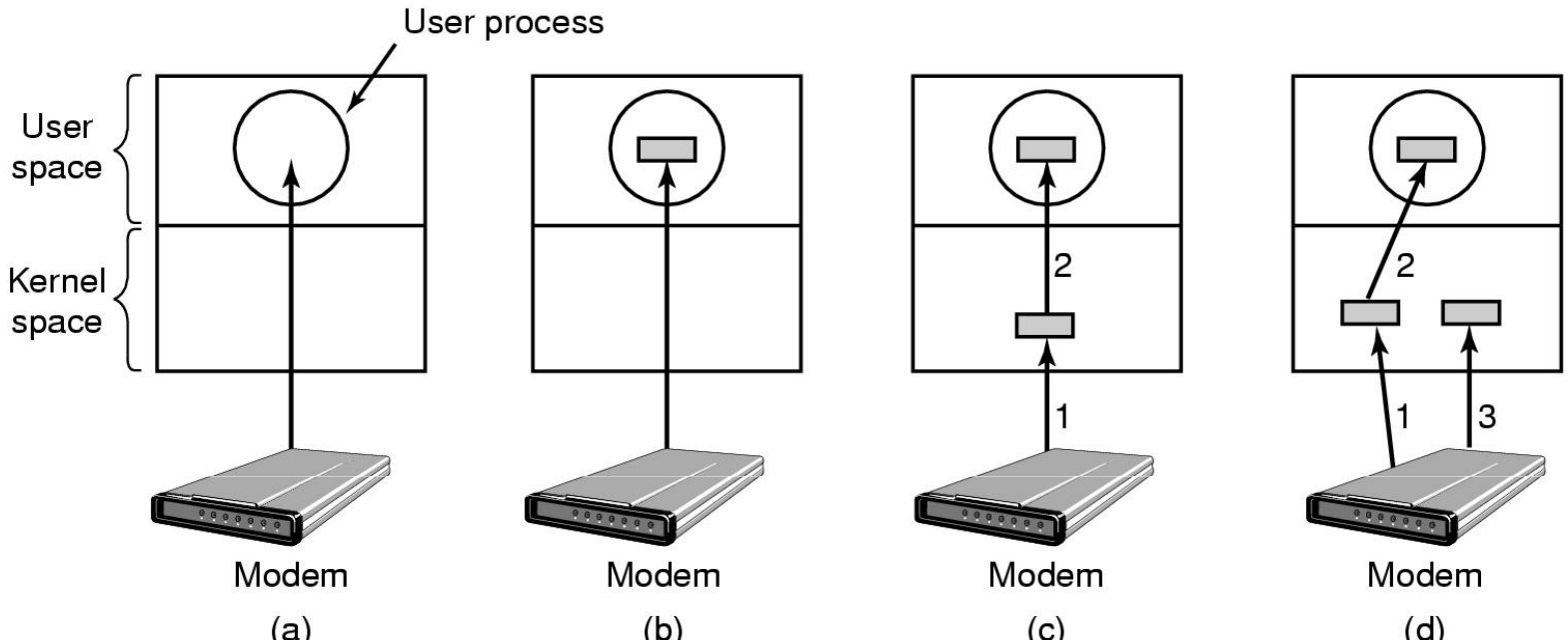
Sw de I/O independente do dispositivo (2)



(a) Driver sem uma interface padrão

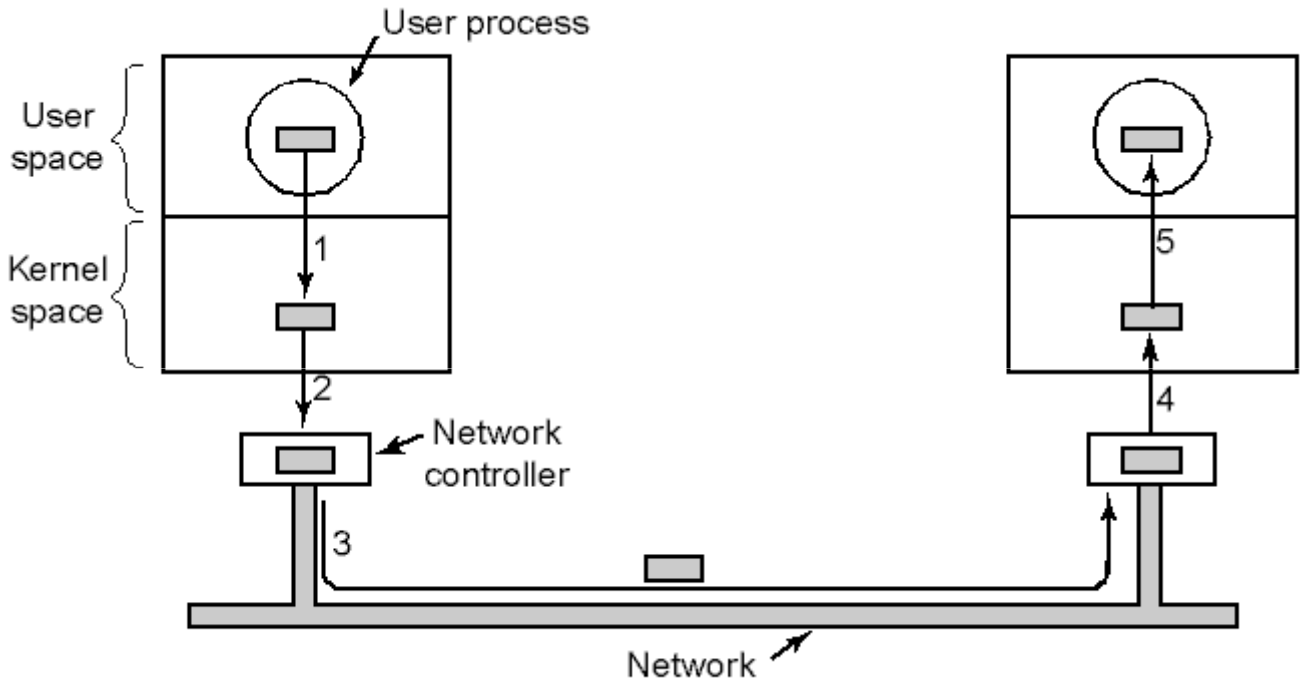
(b) Driver com uma interface padrão

Eficiência com a utilização de buffers (1)



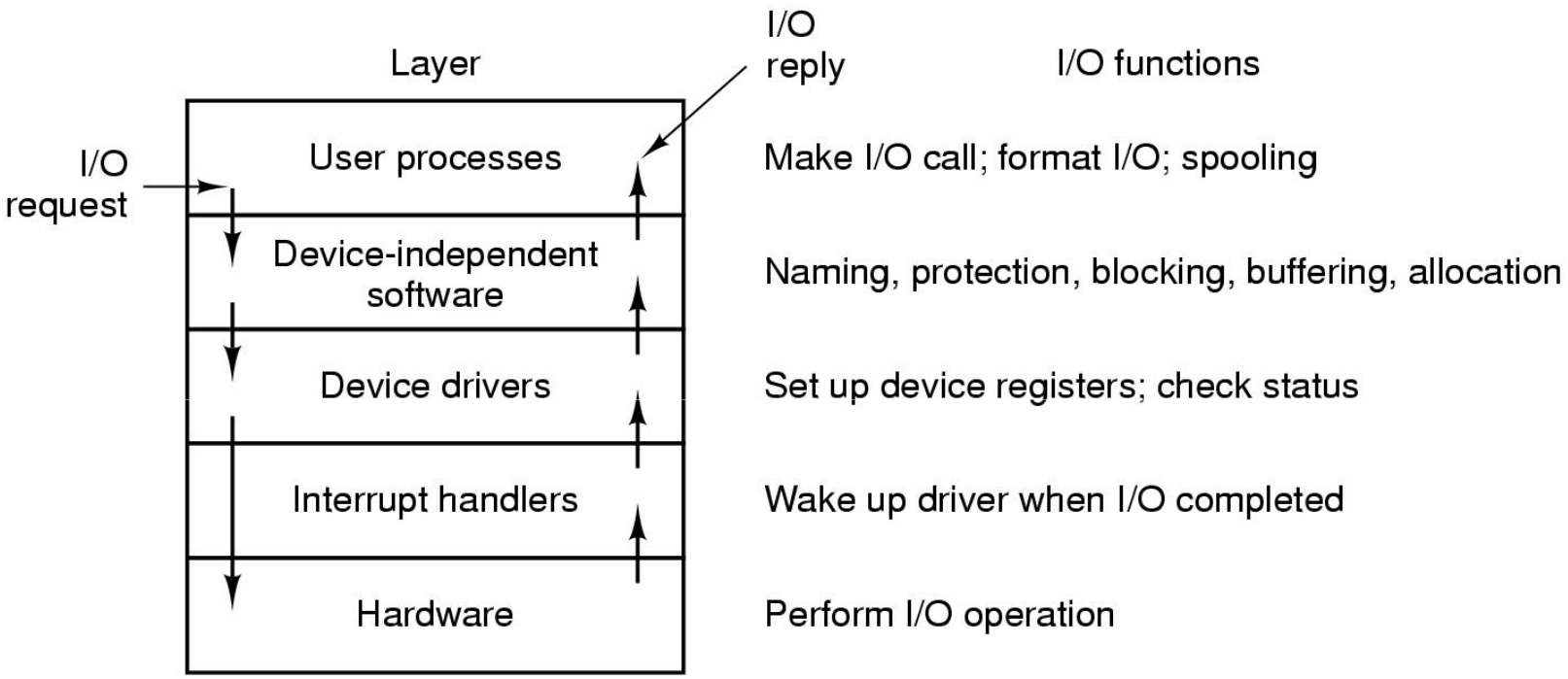
(a) Unbuffered input
(b) Buffering in user space
(c) Buffering in the kernel followed by copying to user space
(d) Double buffering in the kernel

Eficiência com a utilização de buffers (2)



Operações em rede podem envolver muitas cópias

Sw de I/O em espaço do utilizador



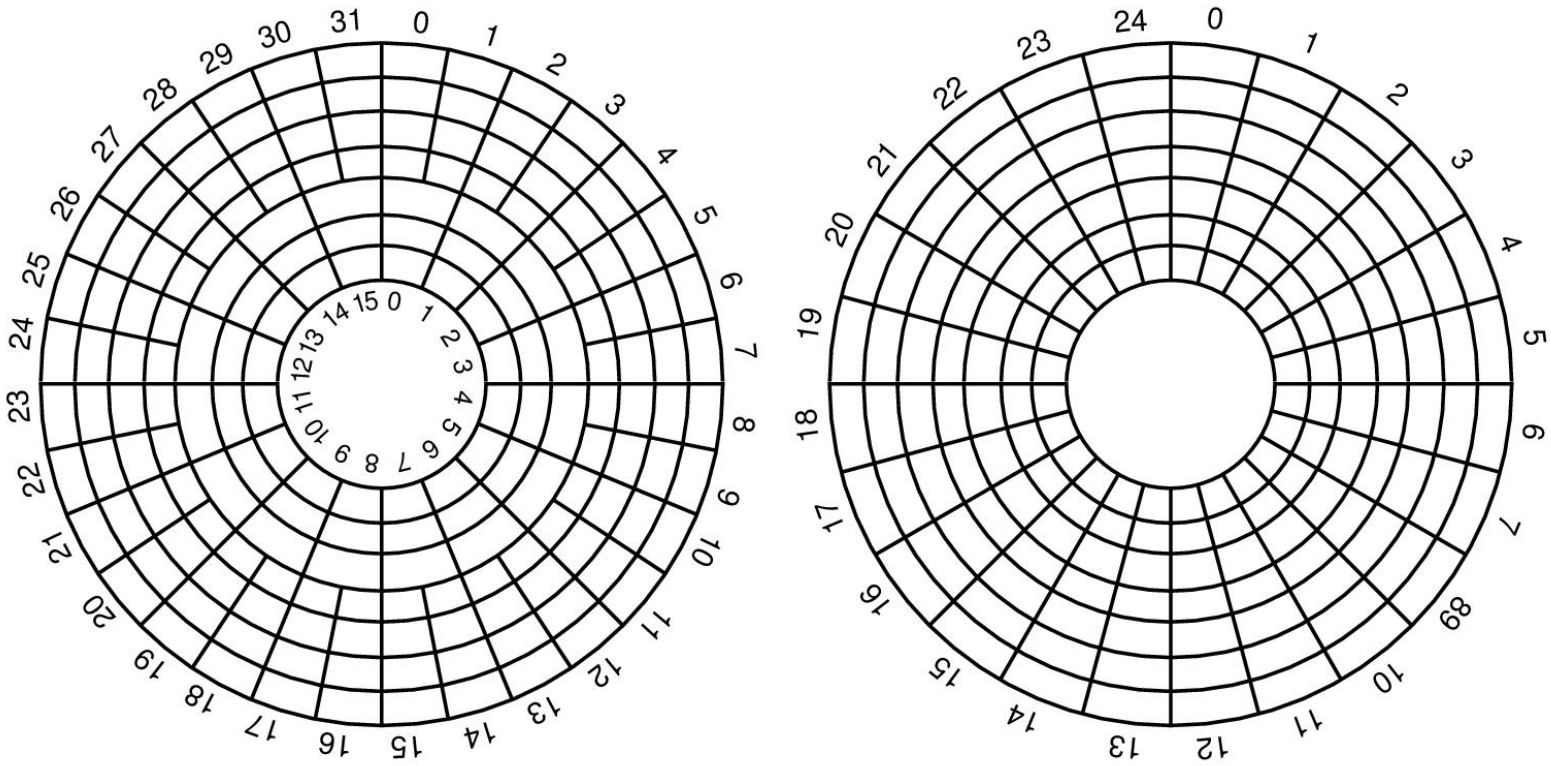
Camadas de sw e suas funções (pe: printf, read, write etc)

Discos

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

Evolução de parms de disco (original IBM PC floppy disk e um Western Digital WD 18300)

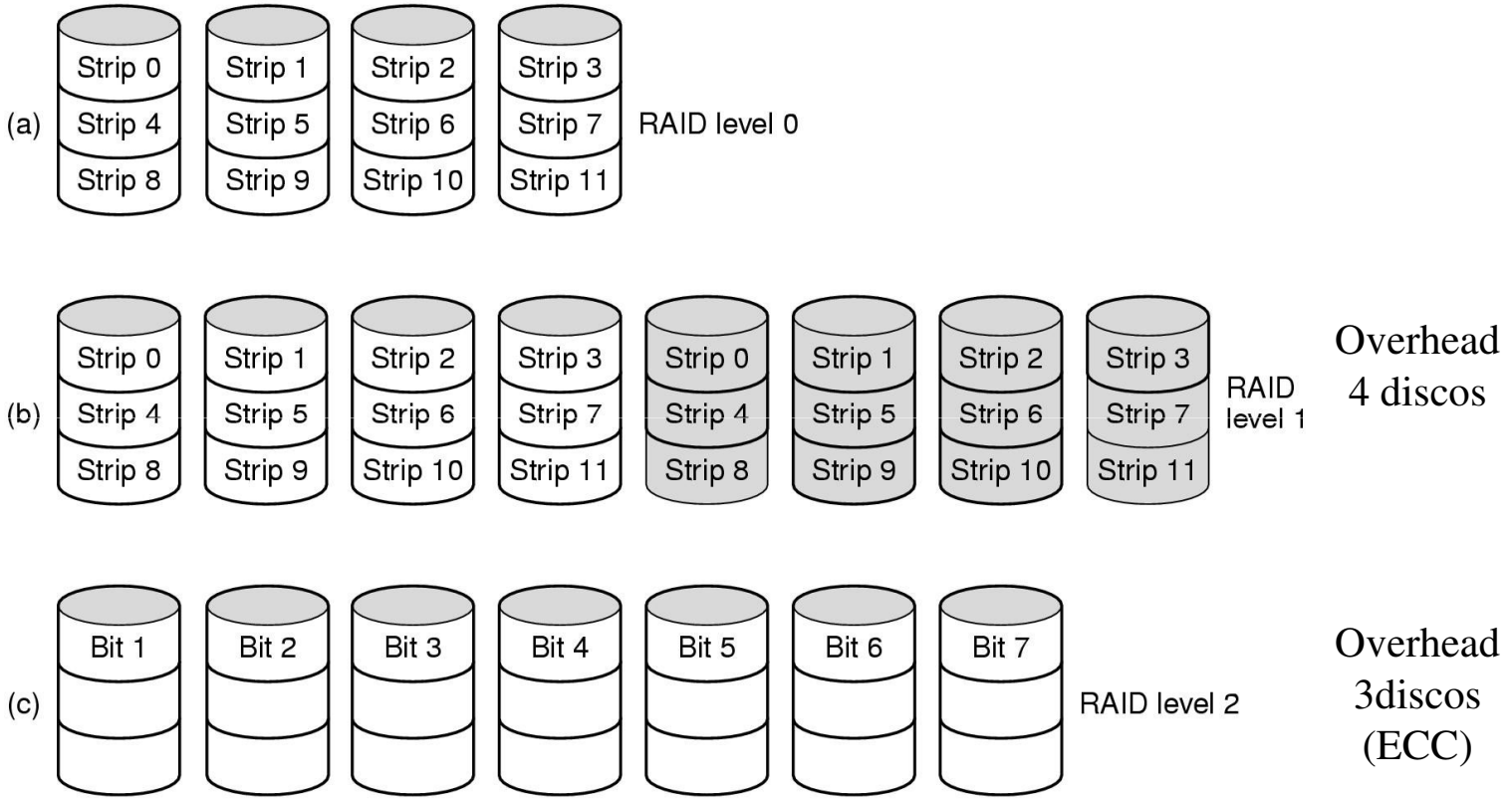
Hw de discos



- Geometria física de um disco com 2 zonas
- Possível geometria virtual apresentada ao SO

Hw de discos

Redundant Array of Independent/Inexpensive Disks (RAID)

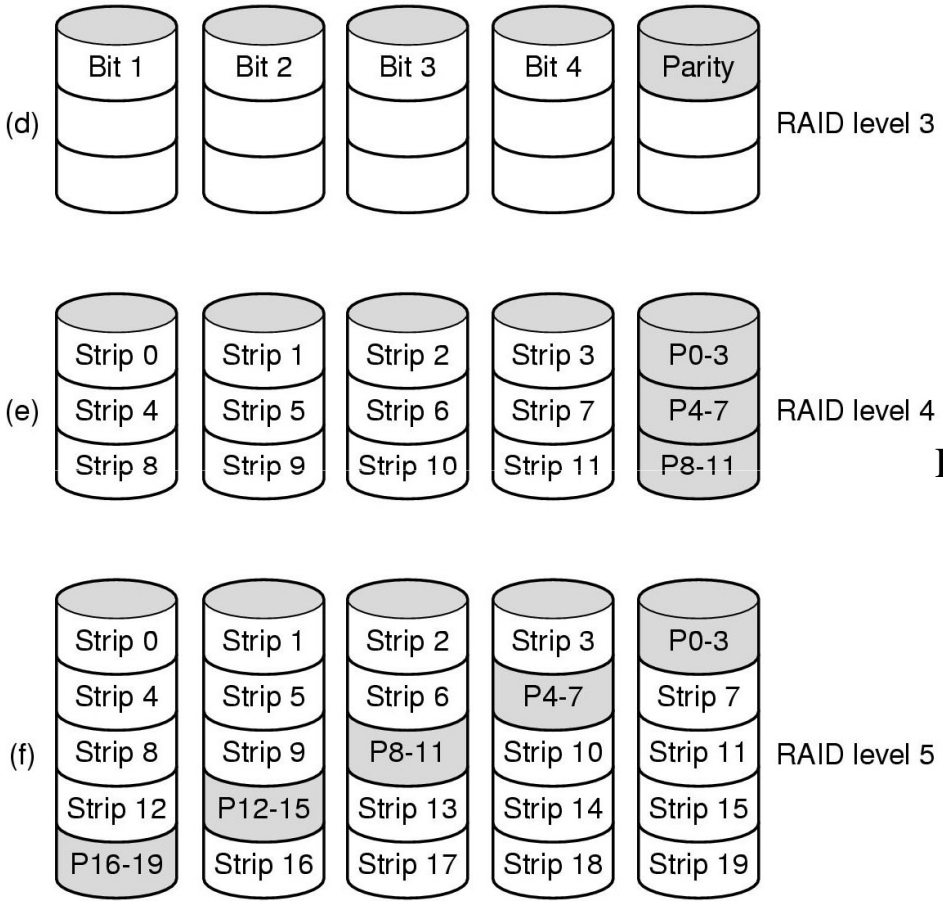


Uses Hamming code

- Raid níveis 0 a 2
- Backup e drives de paridade em cinzento

Hw de discos

Use Parity bits



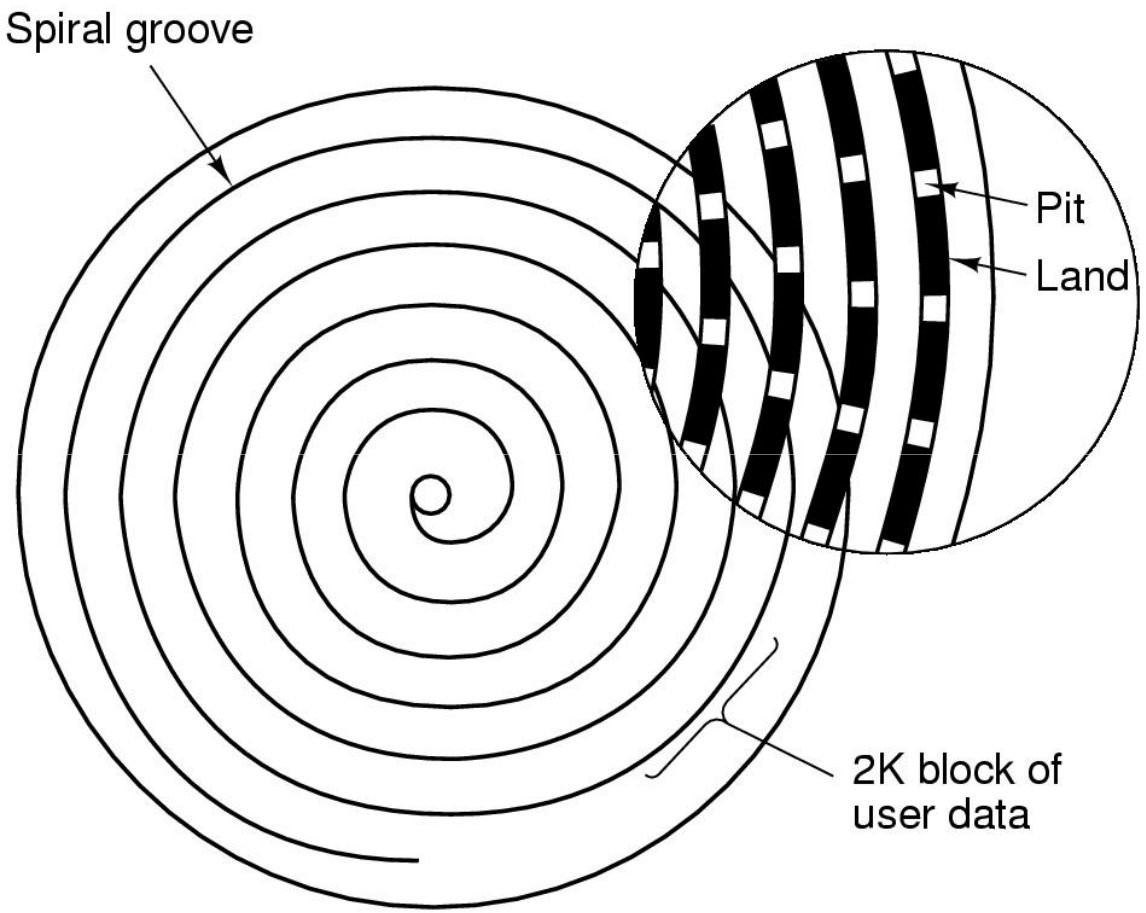
Strips are EXCLUSIVE OR-ed

- Raid levels 3 through 5
- Backup and parity drives are shaded

RAID

- Ganho de desempenho no acesso.
- Redundância em caso de falha em um dos discos.
- Uso múltiplo de várias unidades de discos.
- Facilidade em recuperação de conteúdo "perdido".

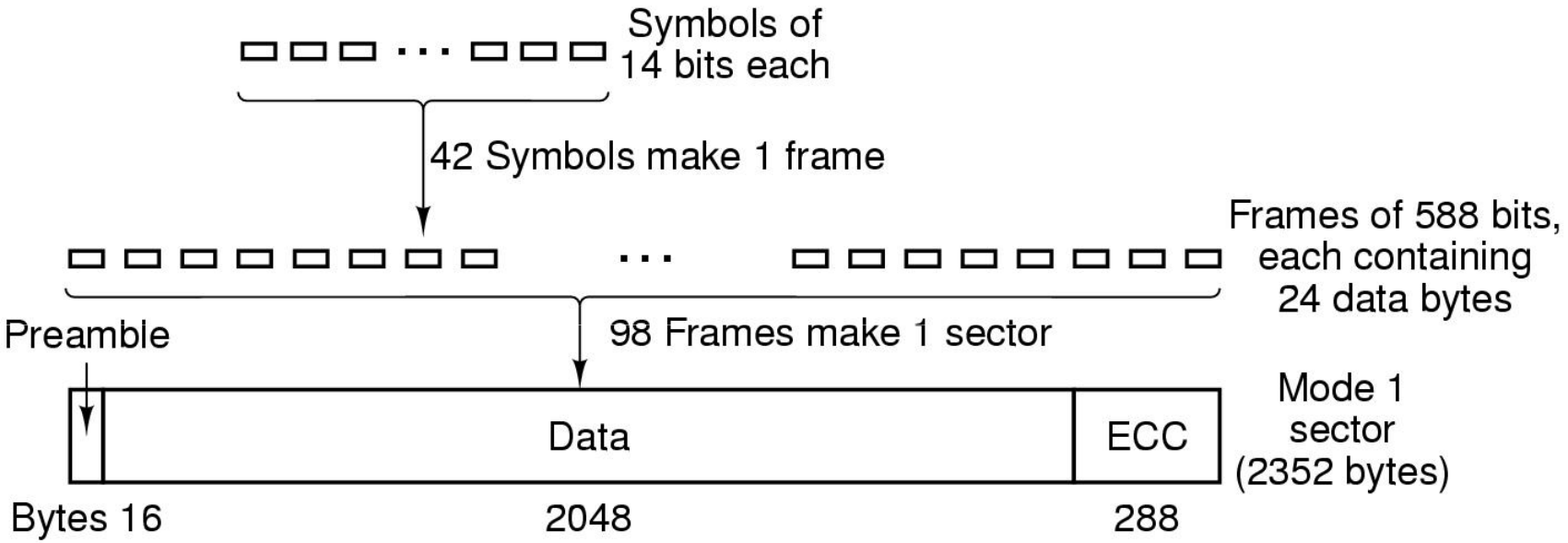
Hw de discos - CD



Superfície de gravação de um CD ou CD-ROM

Espiral pode ter 5,6 Km de comprimento

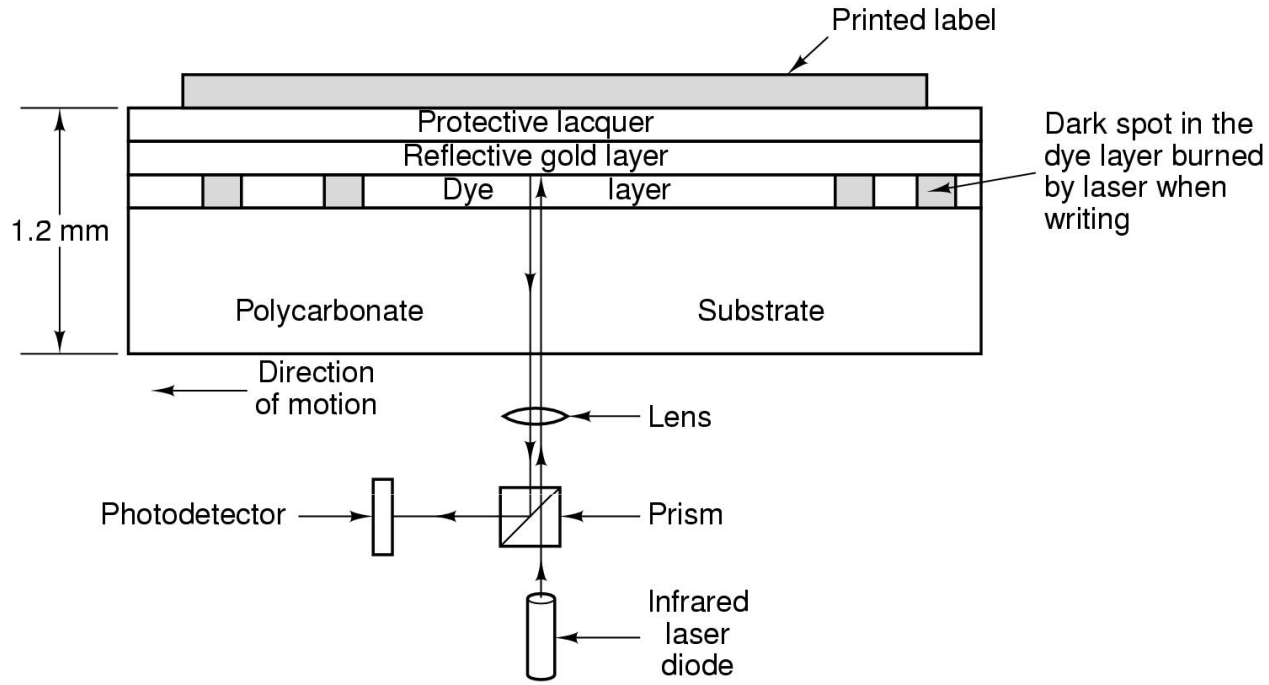
Hw de discos



Layout lógico de dados em um CD-ROM

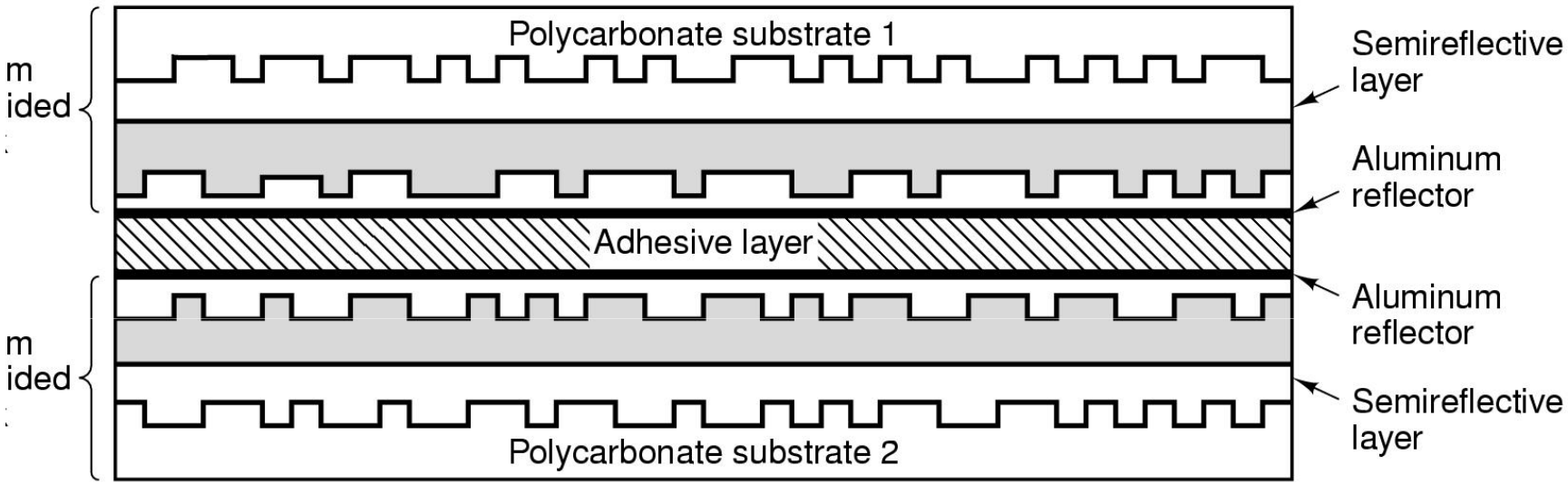
- Similar para música e dados
- Dados: grupos de setores de 98 frames, preâmbulo contém 12 bytes que indicam o início do setor, 3 bytes para indicar o número do setor, e último byte para indicar o modo (1 ou 2)

Hw de discos



- Seção transversal de um disco CD-R (tem uma camada dourada)
  - Não está em escala real
  - “pits” e “lands” simulados
- CD-ROM tem uma camada prateada de alumínio

Hw de discos



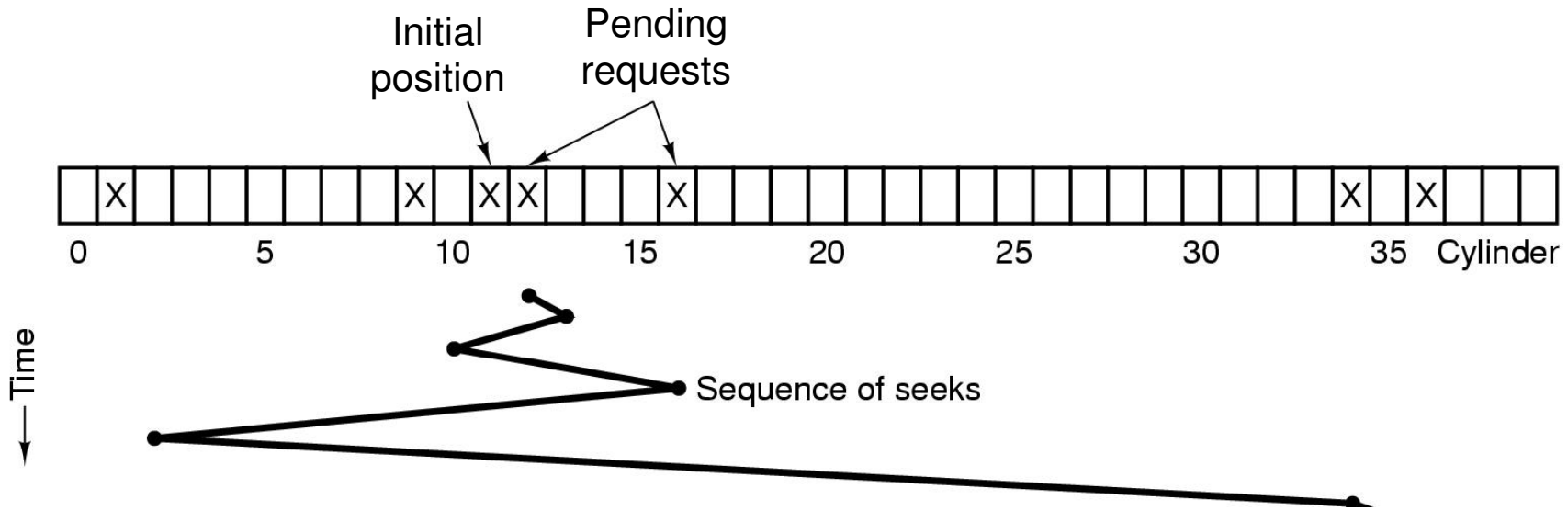
Um DVD de dupla face e camada dupla

- Similar aos CDs, “pits” são menores (0,4 microns)
- Distância menor entre as linhas da espiral
- Laser vermelho
- Capacidade e velocidade maiores

Algoritmos de escalonamento (1)

- Tempo para leitura ou escrita de um bloco de disco é determinado por 3 fatores:
  1. Tempo de posicionamento (Seek time)
  2. Atraso da rotação (Rotational delay)
  3. Tempo de transferência (Actual transfer time)
- Seek time é dominante
- Verificação de erros é feita pelo controlador

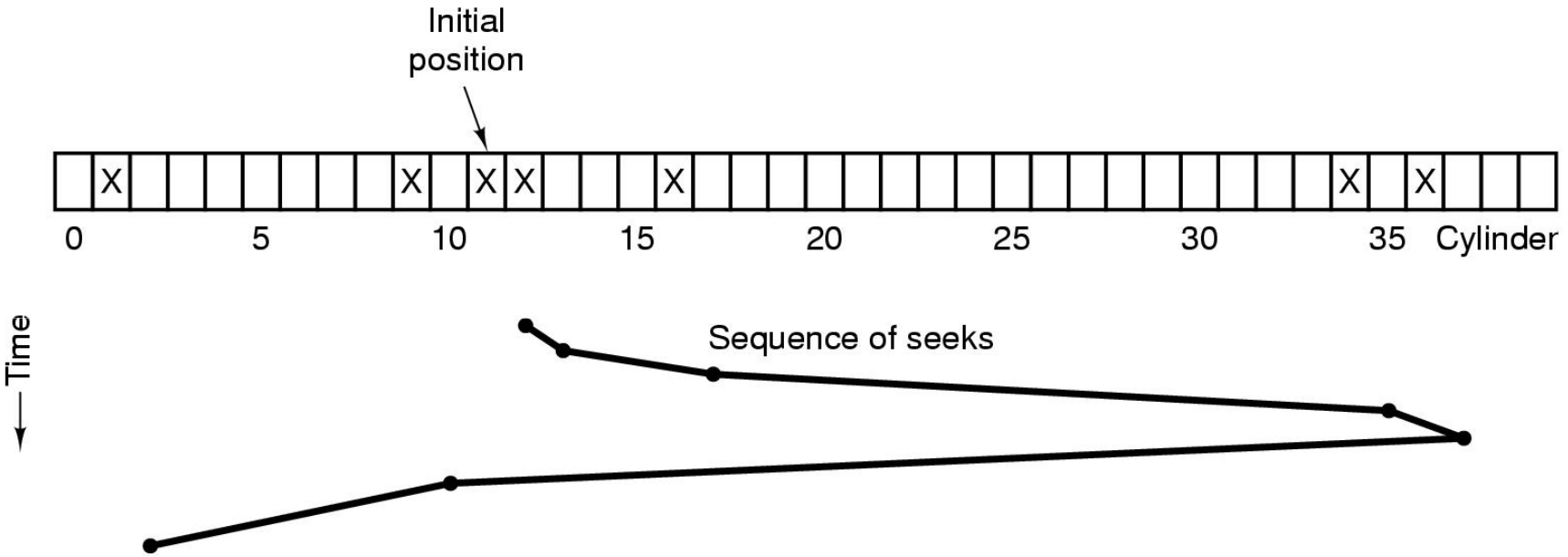
Algoritmos de escalonamento (2)



Shortest Seek First (SSF)

Eqto no cilindro 11, chegam requisições para os cilindros:
1, 36, 16, 34, 9, 12

Algoritmos de escalonamento (3)

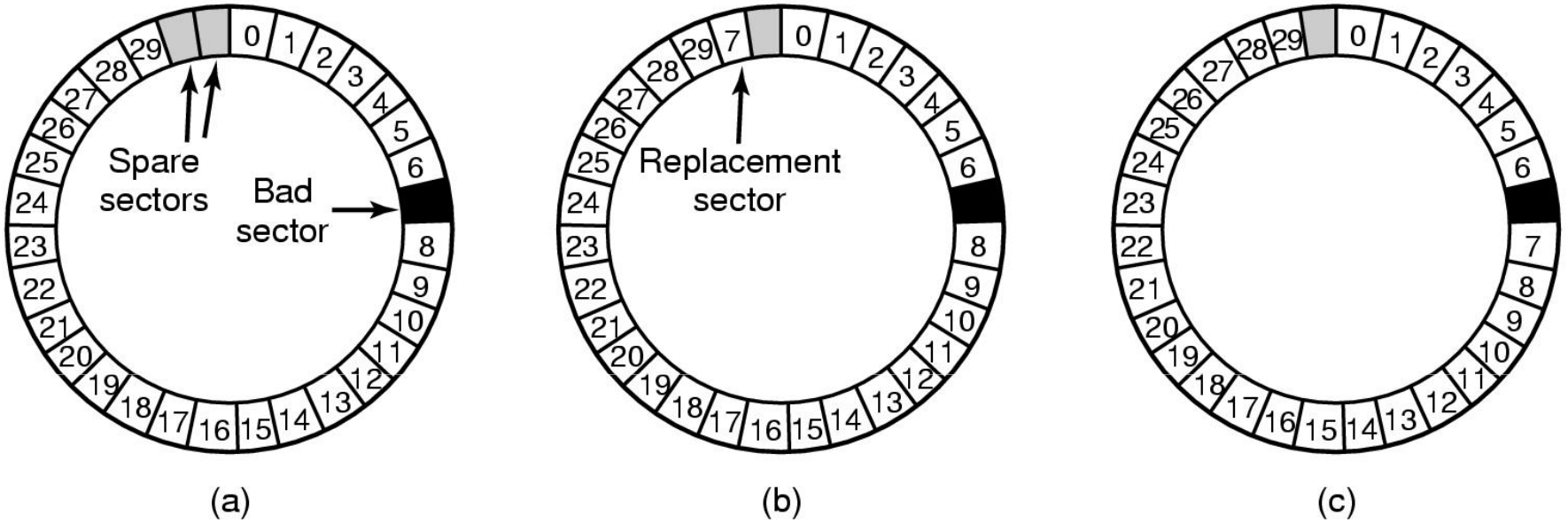


Algoritmo do “elevador” (SCAN)

Algoritmos de escalonamento (4)

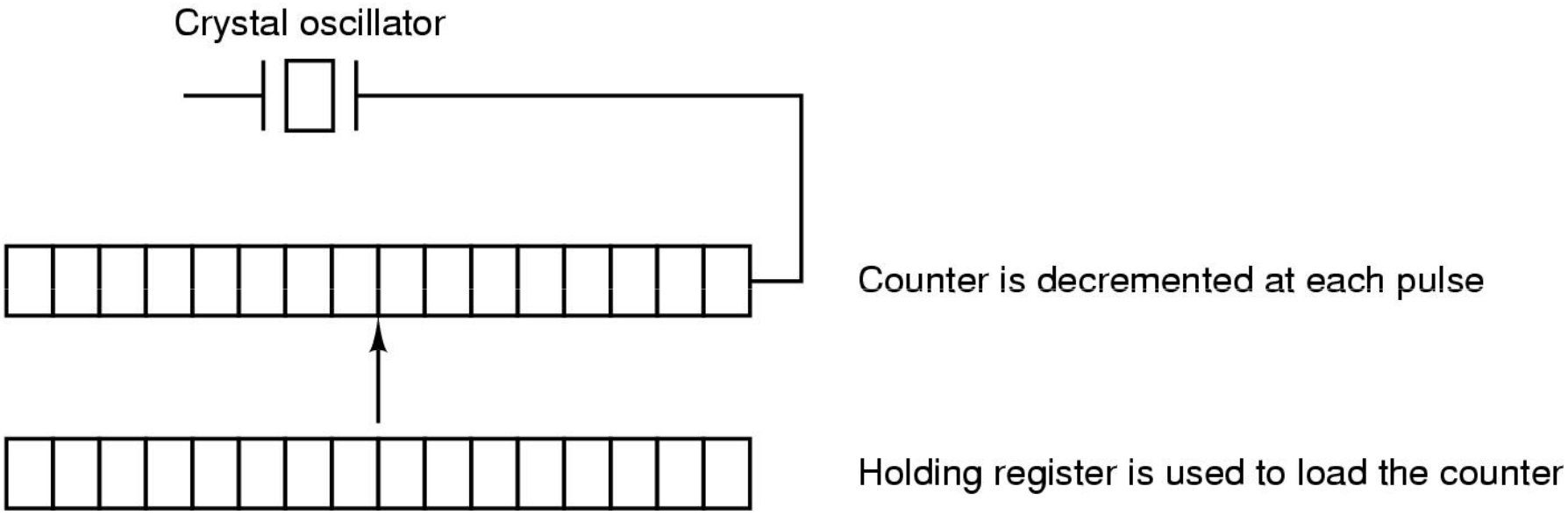
- Desempenho (nº de movimentos):
  - FCFS: 111 cilindros
  - SSF: 61 cilindros
  - Elevador: 60 cilindros
- Elevador normalmente pior, mas tem a propriedade de estabelecer um "upper-bound" para número máximo de movimentos (2 x total de cilindros)

Manipulação de erros



- Uma trilha do disco com um "bad sector" e dois setores "reserva" (a)
- Métodos de correção:
  - Substituição de um setor reserva pelo "bad sector" (b)
  - Reordenação de setores (c)

Relógio
(Clock Hardware)



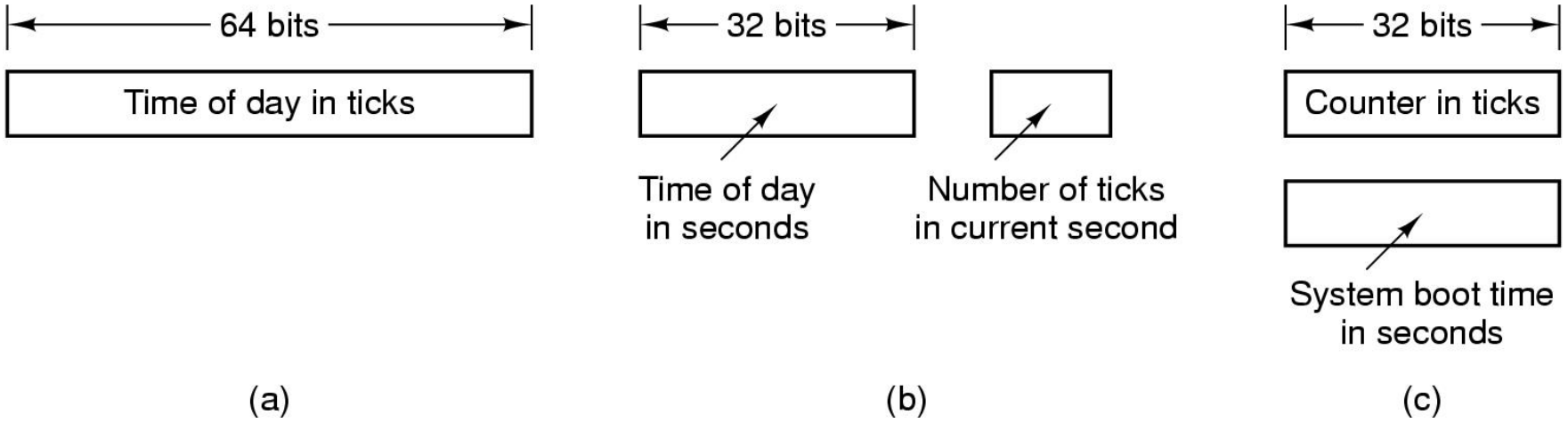
Um relógio programável

Tudo que o relógio em hardware faz é gerar interrupções em intervalos conhecidos
(geralmente conhecidos por “clock ticks”)

Relógio em Software (1)

- Funções do clock driver:
  - Manter a hora do dia
  - Prevenir processos de rodarem por mais tempo do que são permitidos
  - Manter estatísticas sobre utilização da CPU
  - Prover temporizadores “watchdog” para certas partes do sistema (pe., floppy disks)
  - Fazer “profiling”, monitoração, e coleta de estatísticas

Relógio em Software (2)
manter a hora do dia



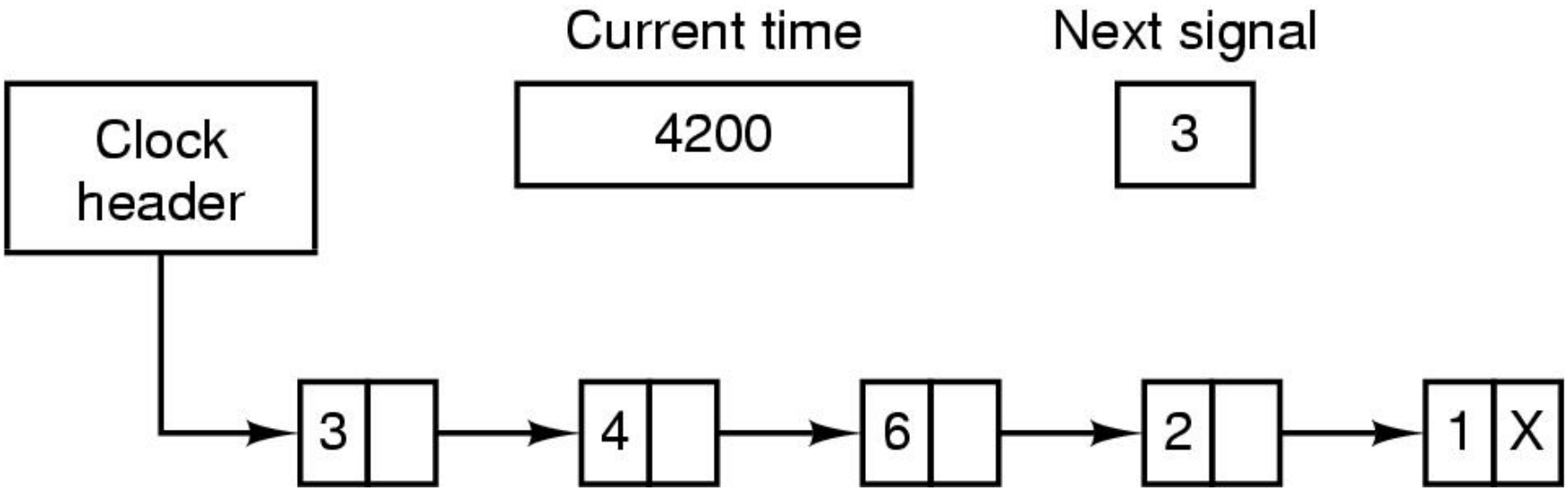
3 formas de manter a hora

Importante: nº de bits do registo de tempo

(a) Manutenção custosa do contador

(b) 2^32 segundos consegue contar 136+ anos

Relógio em Software (3)

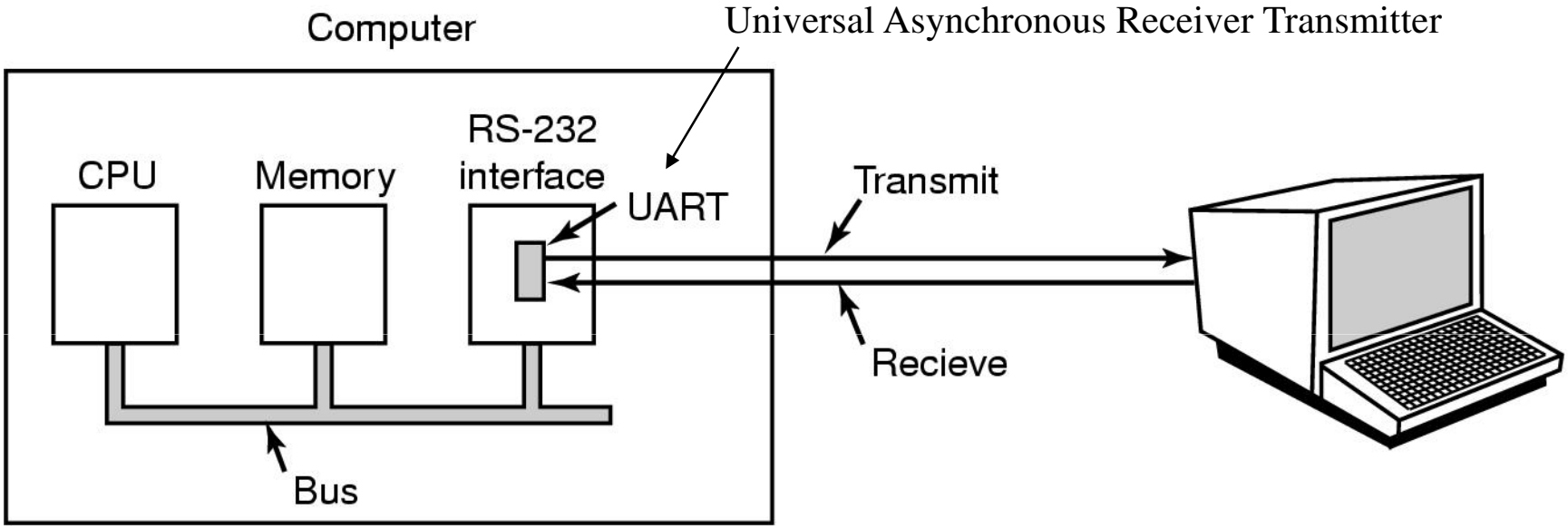


Simulando múltiplos relógios com um único relógio

Por exemplo, para manipulação de sinais entre processos ou eventos de rede

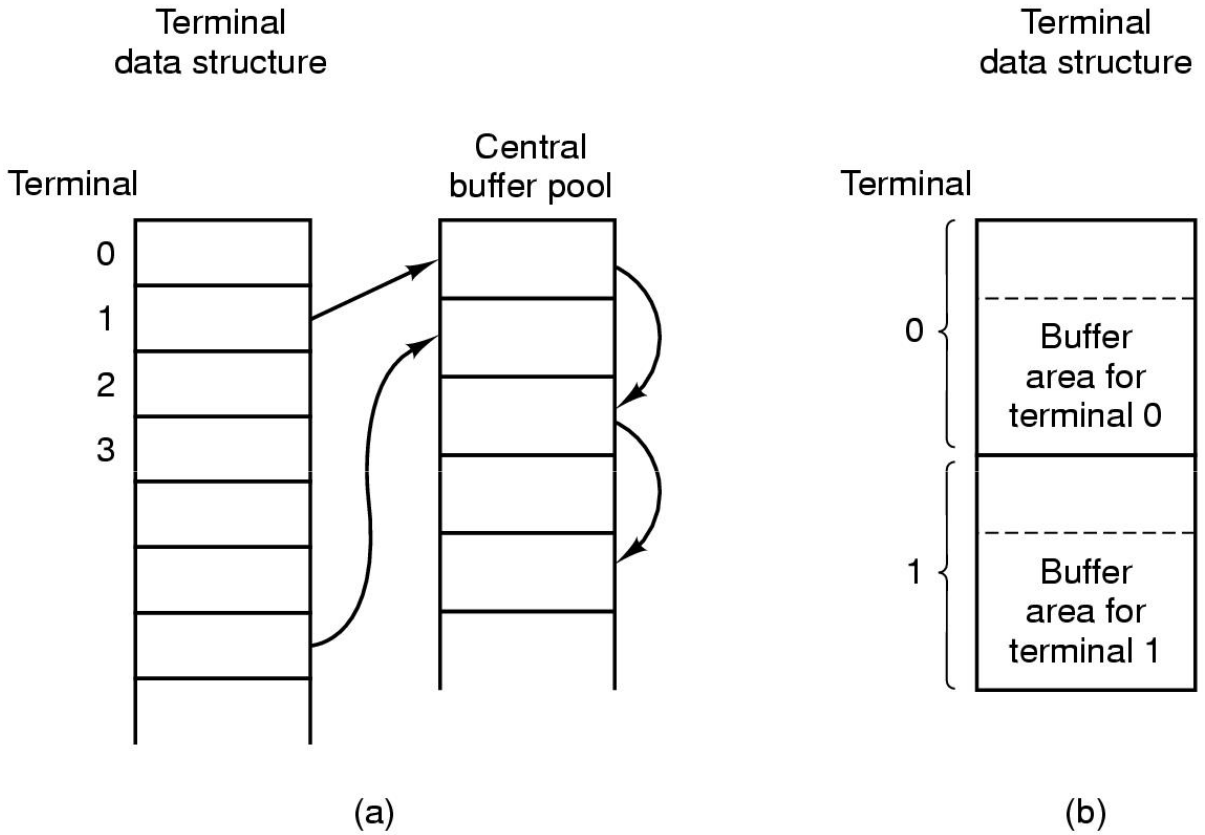
Terminais orientados a caracter

RS-232 Terminal Hardware



- Um terminal RS-232 comunica com um computador de bit em bit
- Linha serial (série) – bits enviados em série, 1 de cada vez
- Windows foi o primeiro a utilizar as portas COM1 e COM2 para portas série
- Computador e terminal são completetamente independentes

Input Software (1)



- Buffer centralizado
- Buffer dedicado para cada terminal

Input Software (2)

Character	POSIX name	Comment
CTRL-H	ERASE	Backspace one character
CTRL-U	KILL	Erase entire line being typed
CTRL-V	LNEXT	Interpret next character literally
CTRL-S	STOP	Stop output
CTRL-Q	START	Start output
DEL	INTR	Interrupt process (SIGINT)
CTRL-\	QUIT	Force core dump (SIGQUIT)
CTRL-D	EOF	End of file
CTRL-M	CR	Carriage return (unchangeable)
CTRL-J	NL	Linefeed (unchangeable)

Caracteres em modo canônico, somente os últimos 2 não podem ser modificados por programa

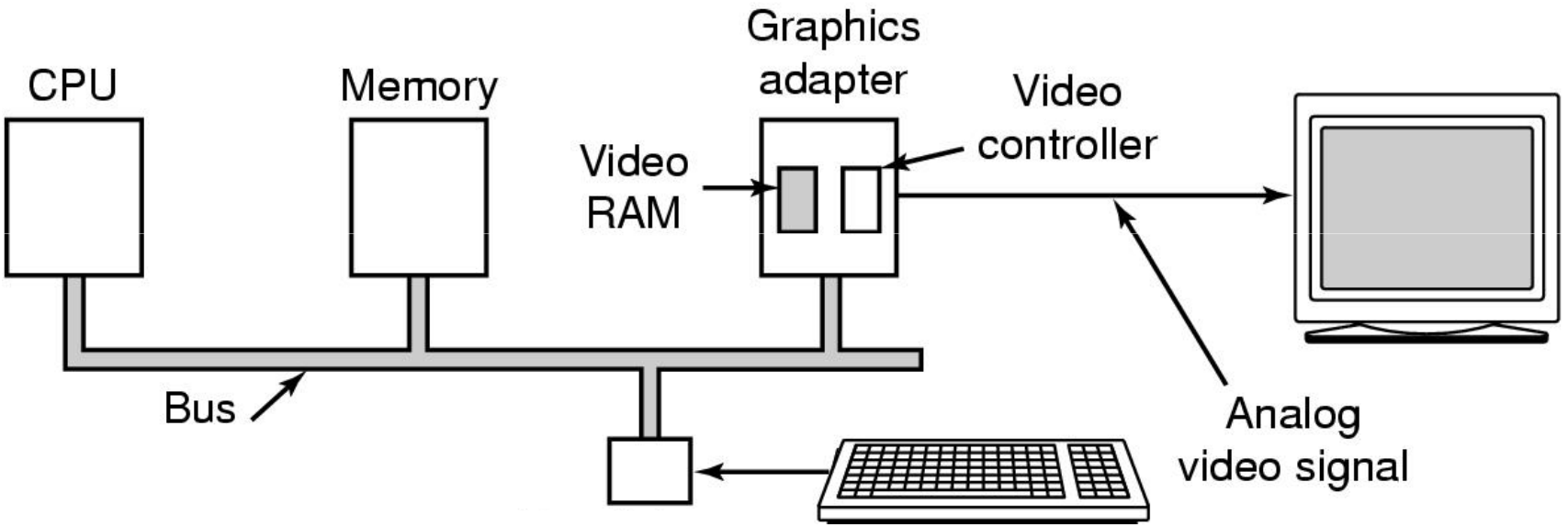
Output Software

Escape sequence	Meaning
ESC [*n* A	Move up *n* lines
ESC [*n* B	Move down *n* lines
ESC [*n* C	Move right *n* spaces
ESC [*n* D	Move left *n* spaces
ESC [*m* ; *n* H	Move cursor to (*m*,*n*)
ESC [*s* J	Clear screen from cursor (0 to end, 1 from start, 2 all)
ESC [*s* K	Clear line from cursor (0 to end, 1 from start, 2 all)
ESC [*n* L	Insert *n* lines at cursor
ESC [*n* M	Delete *n* lines at cursor
ESC [*n* P	Delete *n* chars at cursor
ESC [*n* @	Insert *n* chars at cursor
ESC [*n* m	Enable rendition *n* (0=normal, 4=bold, 5=blinking, 7=reverse)
ESC M	Scroll the screen backward if the cursor is on the top line

Sequências de "escape" ANSI

- Aceitos por um driver de terminal para output
- ESC é o caracter ASCII (0x1B)
- n,m, e s são parâmetros opcionais numéricos

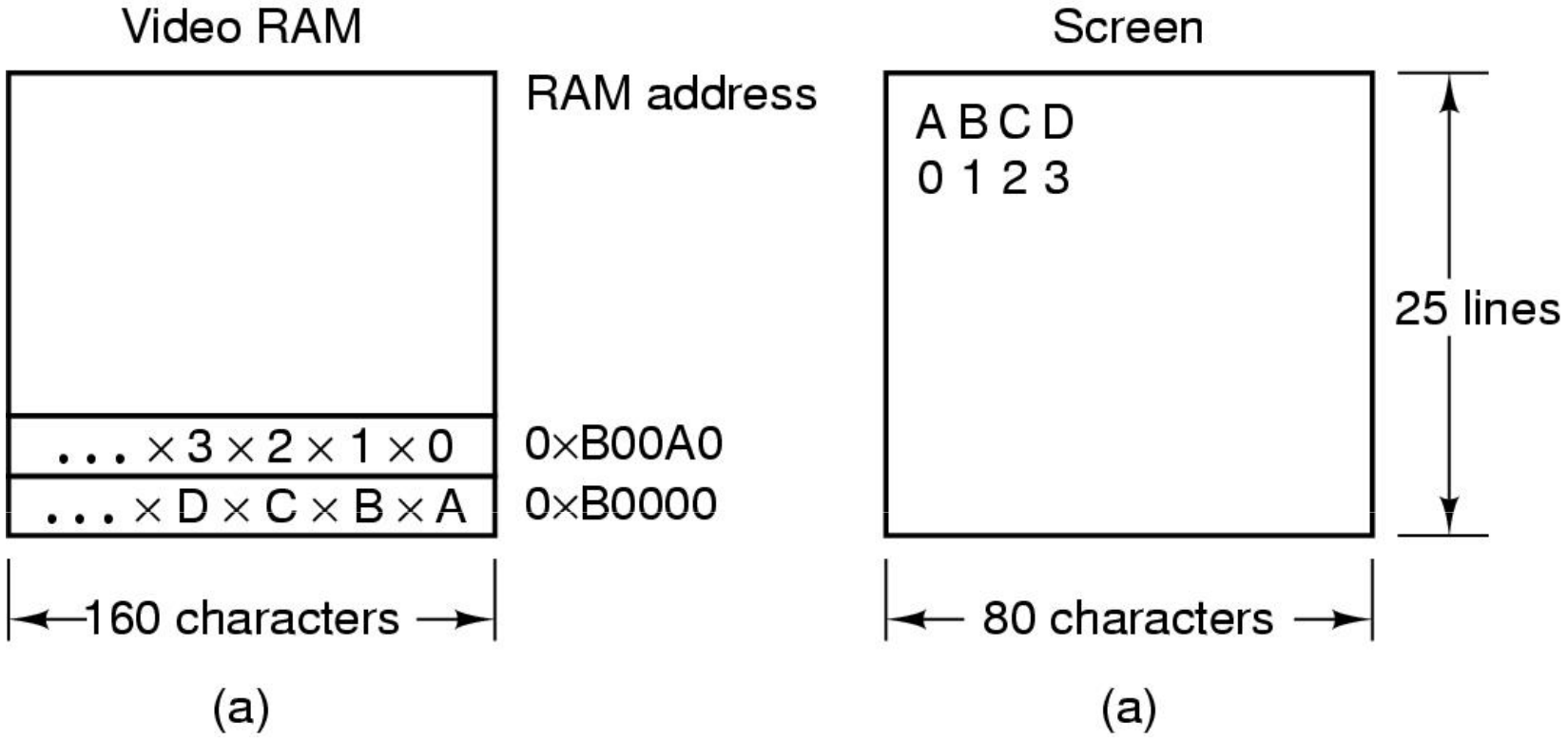
Display Hardware (1)



Displays mapeados em memória

- driver escreve diretamente para a RAM de video

Display Hardware (2)

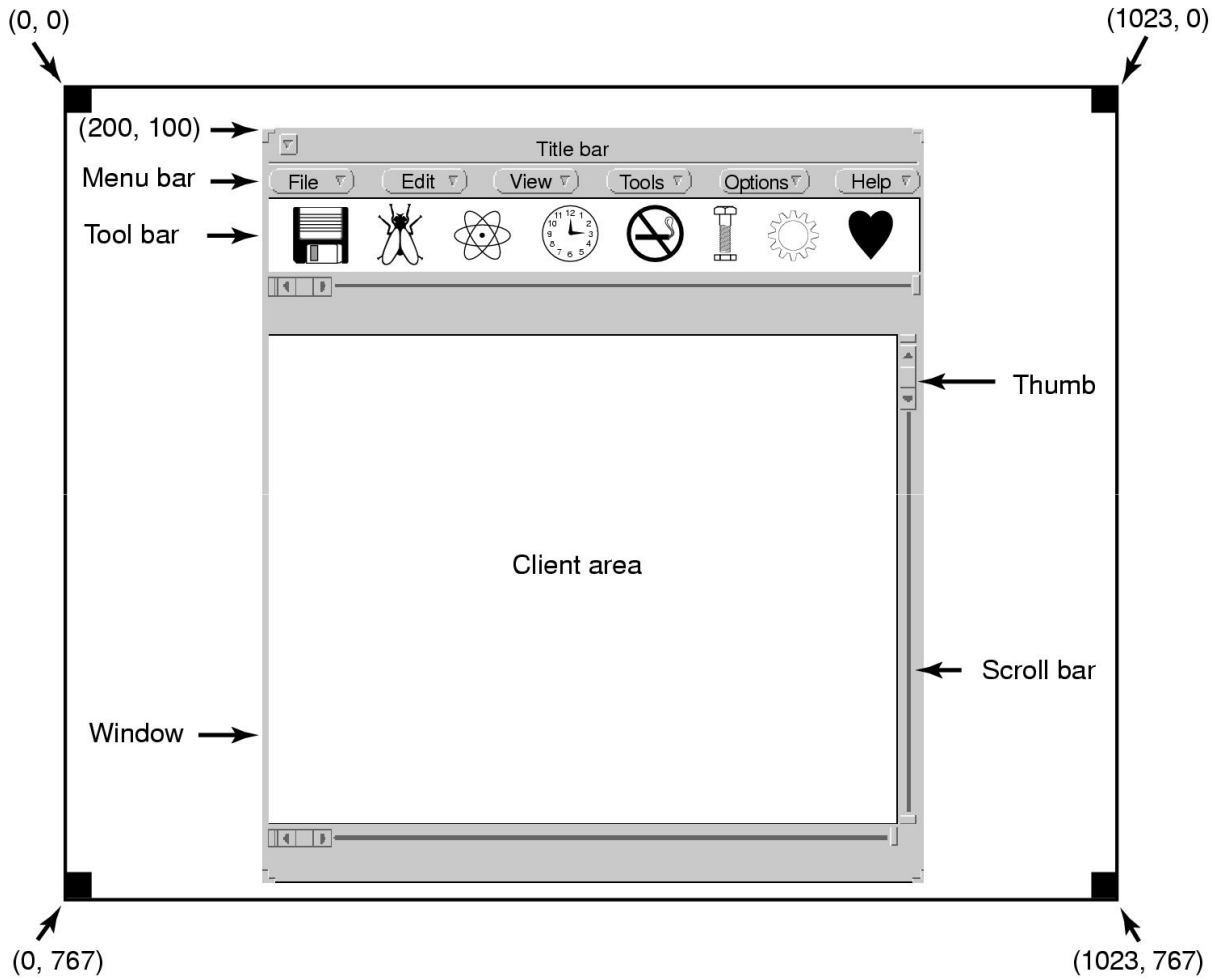


- Uma imagem em RAM de video
  - Display simples monocromático
  - Modo caracter
- Ecrã correspondente
  - (**X**s são bytes de atributos)

Input Software

- Driver de teclado envia um número (código da tecla: “scan code”)
  - driver converte os códigos para caracteres
  - Usa a tabela ASCII
- Exceções e adaptações são necessárias para lidar com as diversas línguas
  - A maioria dos SO provê “loadable keymaps” ou “code pages”

Output Software for Windows (1)



Exemplo de janela localizada em (200,100) num display XGA

Output Software for Windows (2)

- GUI: Graphical User Interface
- 4 elementos essenciais:
  - Windows
  - Icons
  - Menus
  - Pointing device
- Melhor conhecidos como WIMP

Output Software for Windows (3)

```
#include <windows.h>

int WINAPI WinMain(HINSTANCE h, HINSTANCE, hprev, char *szCmd, int iCmdShow)
{
      WNDCLASS wndclass;              /* class object for this window */
      MSG msg;                        /* incoming messages are stored here */
      HWND hwnd;                      /* handle (pointer) to the window object */

      /* Initialize wndclass */
      wndclass.lpfnWndProc = WndProc;          /* tells which procedure to call */
      wndclass.lpszClassName = "Program name";    /* Text for title bar */
      wndclass.hIcon = LoadIcon(NULL, IDI_APPLICATION);   /* load program icon */
      wndclass.hCursor = LoadCursor(NULL, IDC_ARROW);   /* load mouse cursor */

      RegisterClass(&wndclass);       /* tell Windows about wndclass */
      hwnd = CreateWindow ( ... )     /* allocate storage for the window */
      ShowWindow(hwnd, iCmdShow); /* display the window on the screen */
      UpdateWindow(hwnd);             /* tell the window to paint itself */
```

Parte de um programa Windows para criar janelas (parte 1)

Output Software for Windows (4)

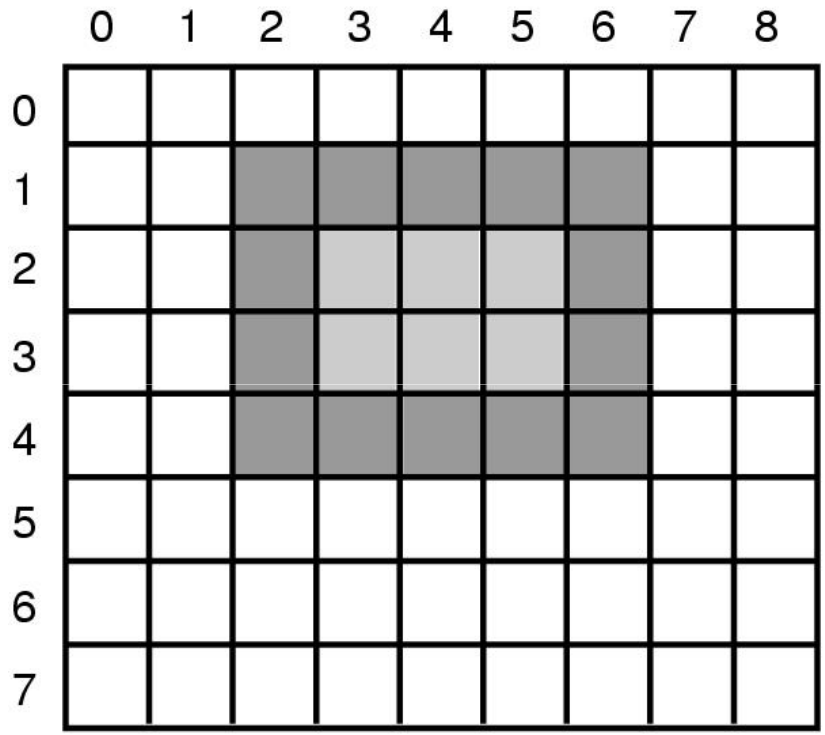
```
    while (GetMessage(&msg, NULL, 0, 0)) {    /* get message from queue */
        TranslateMessage(&msg);    /* translate the message */
        DispatchMessage(&msg);     /* send msg to the appropriate procedure */
    }
    return(msg.wParam);
}

long CALLBACK WndProc(HWND hwnd, UINT message, UINT wParam, long lParam)
{
    /* Declarations go here. */

    switch (message) {
        case WM_CREATE:     ... ;   return ... ;    /* create window */
        case WM_PAINT:      ... ;   return ... ;    /* repaint contents of window */
        case WM_DESTROY:    ... ;   return ... ;    /* destroy window */
    }
    return(DefWindowProc(hwnd, message, wParam, lParam));/* default */
}
```

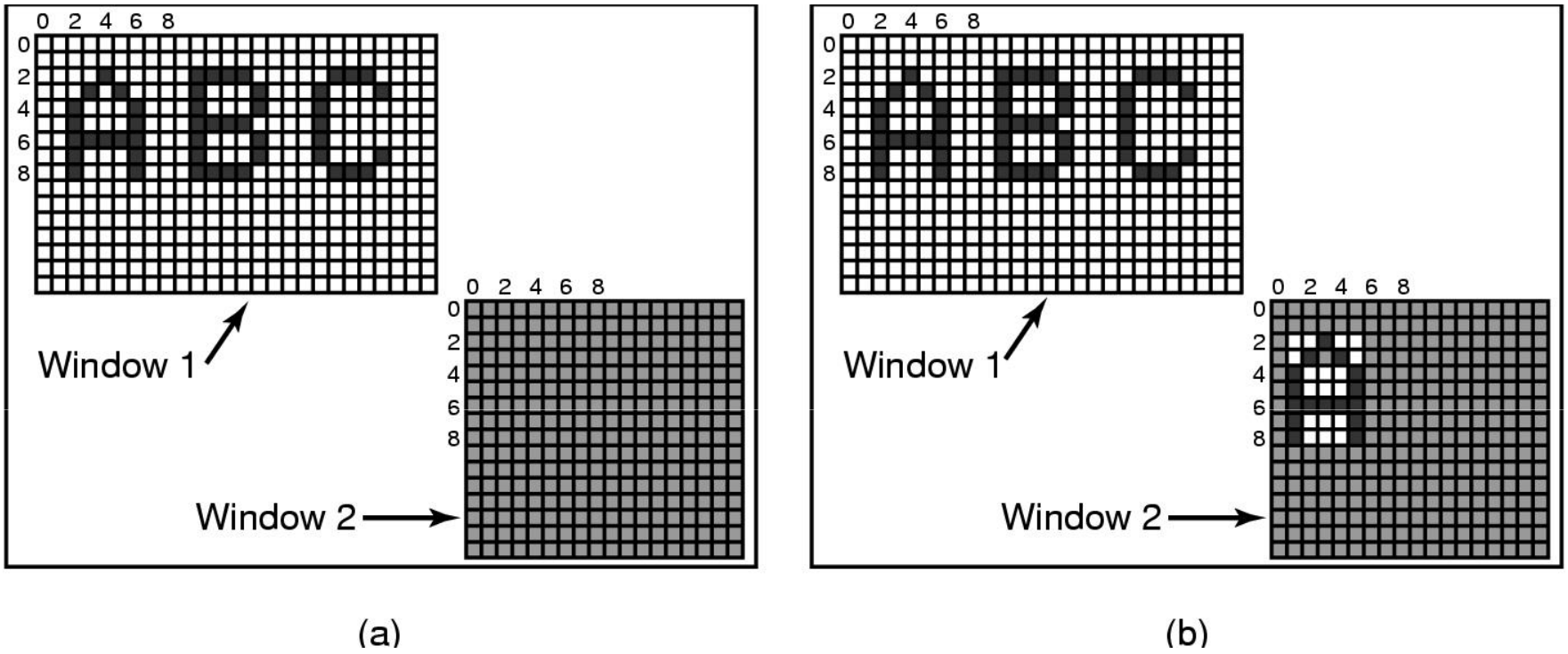
Parte de um programa Windows para criar janelas (parte 2)

Output Software for Windows (5)



Exemplo de retângulo desenhado a partir de uma chamada à função Rectangle(hdc,2,1,6,4) da GDI (Graphics Device Interface) Windows

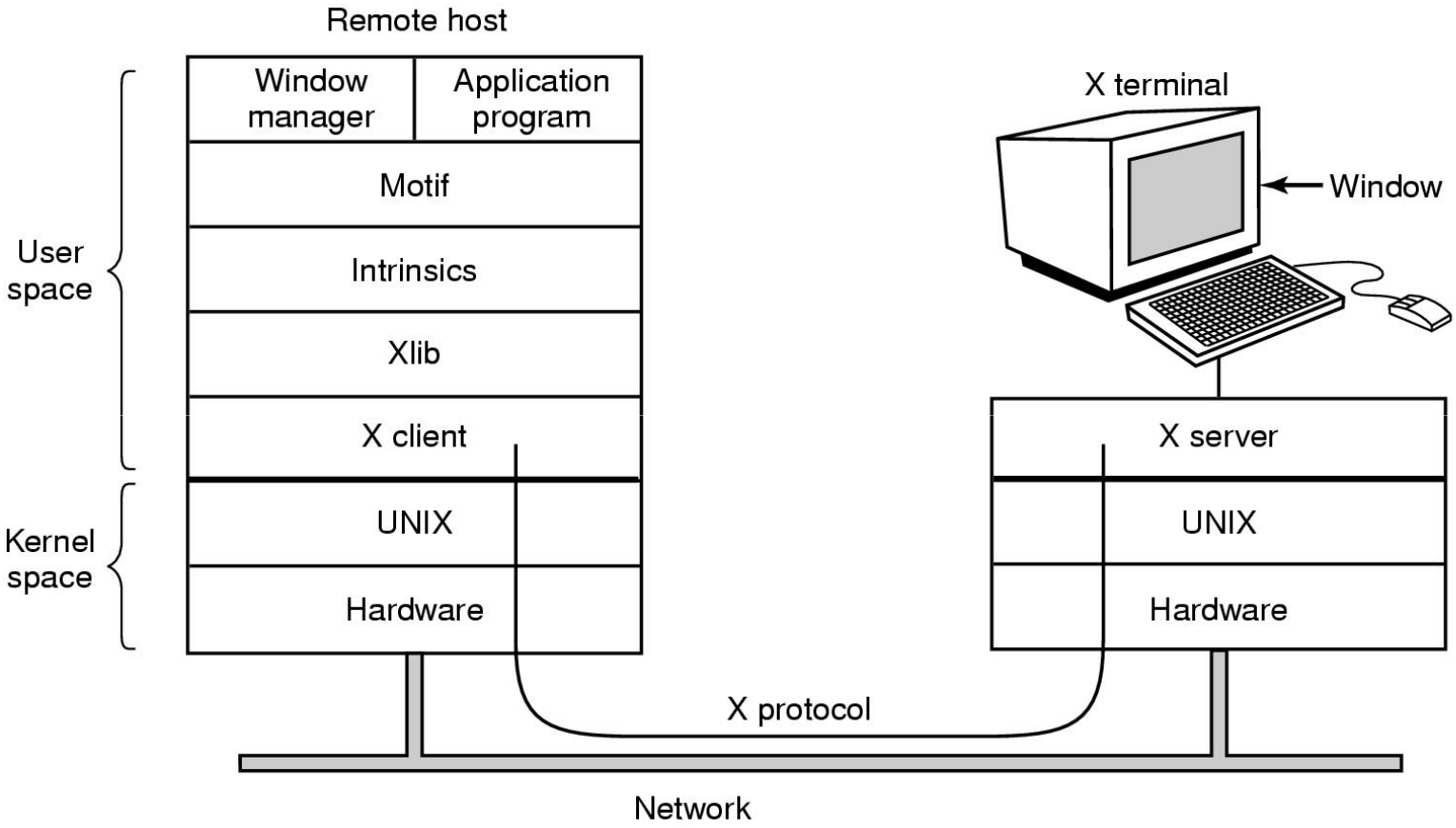
Output Software for Windows (6)



- Cópia de bitmaps usando BitBlt(hdc2,1,2,5,7,hdc1,2,2,SRCCOPY)
  - (a) antes
  - (b) depois

Network Terminals

X Windows (1)



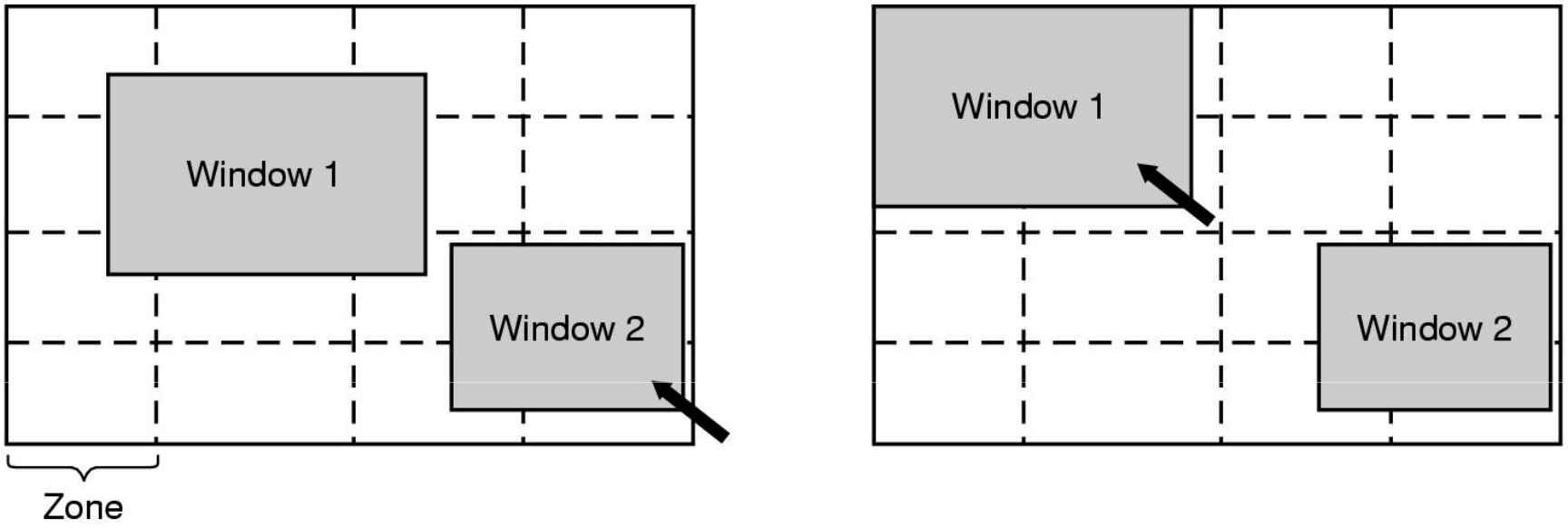
Clientes e servidores em X Windows (created by MIT)

Gestão de energia (1)

Device	Li et al. (1994)	Lorch and Smith (1998)
Display	68%	39%
CPU	12%	18%
Hard disk	20%	12%
Modem		6%
Sound		2%
Memory	0.5%	1%
Other		22%

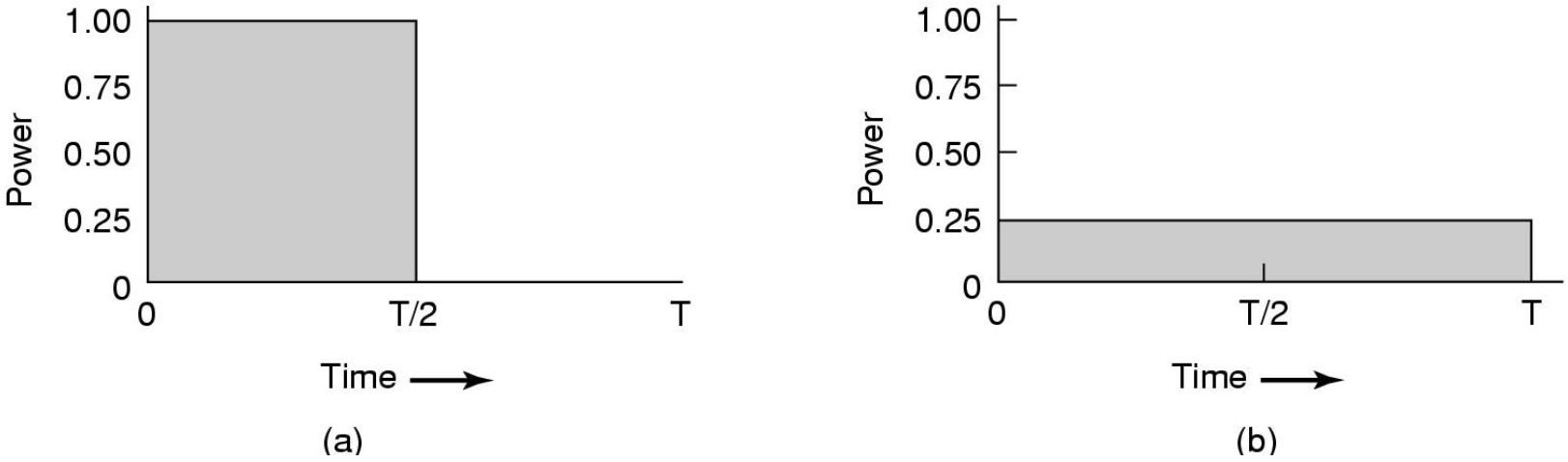
Consumo de energia em várias partes de um portátil

Gestão de energia (2)



Uso de duas zonas para desligar partes do ecrã

Gestão de energia (3)



- Rodando com poder computacional total
- Corte de voltagem à metade
  - Poder computacional reduz-se à metade, mas…
  - …Economiza 4 vezes mais energia

Gestão de energia (4)

- Outra solução seria programar de forma a gastar menos energia, mas isto pode significar uma experiência má para o utilizador

- Exemplos
  - Mudar output de colorido para preto e branco
  - Redução de vocabulário em programas de reconhecimento de voz
  - Reduzir resolução ou detalhes de uma imagem