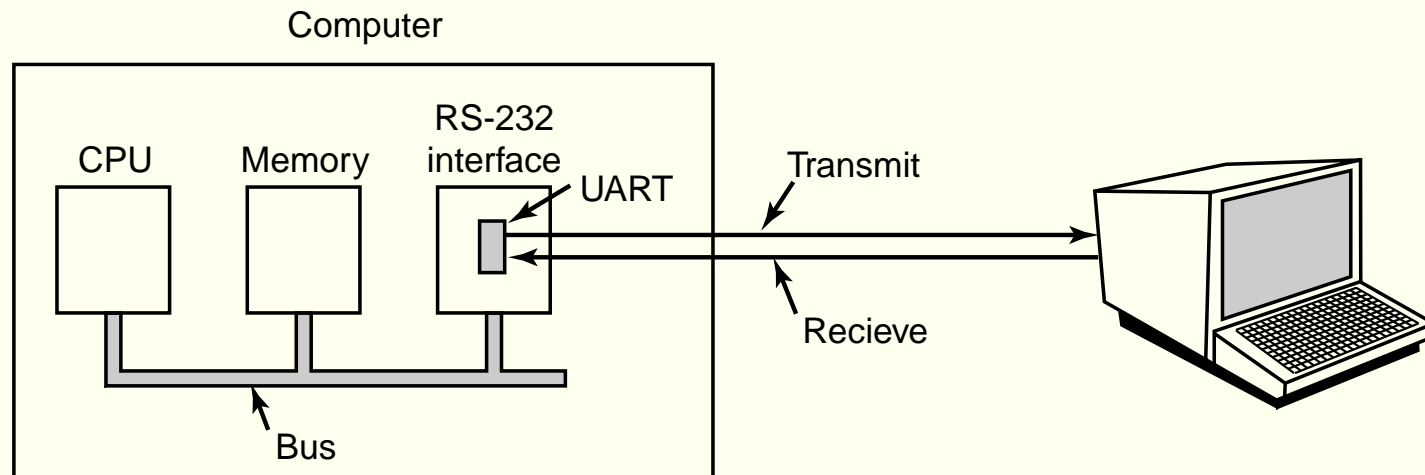


Gerenciamento de Entrada e Saída

Dispositivos de I/O e velocidades

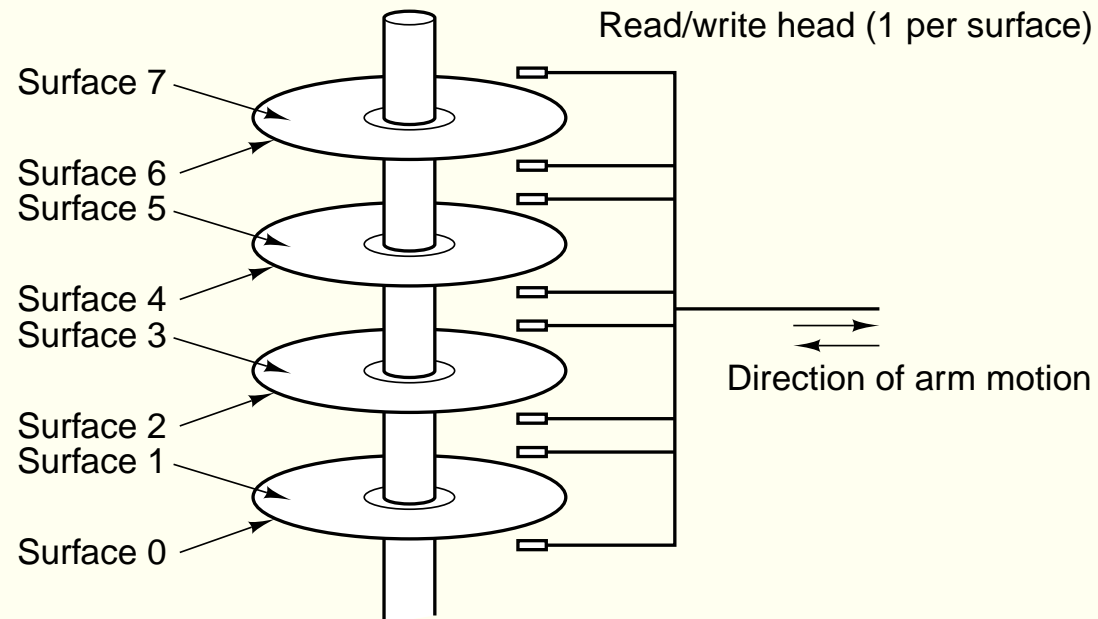
Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Telephone channel	8 KB/sec
Dual ISDN lines	16 KB/sec
Laser printer	100 KB/sec
Scanner	400 KB/sec
Classic Ethernet	1.25 MB/sec
USB (Universal Serial Bus)	1.5 MB/sec
Digital camcorder	4 MB/sec
IDE disk	5 MB/sec
40x CD-ROM	6 MB/sec
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

Character device



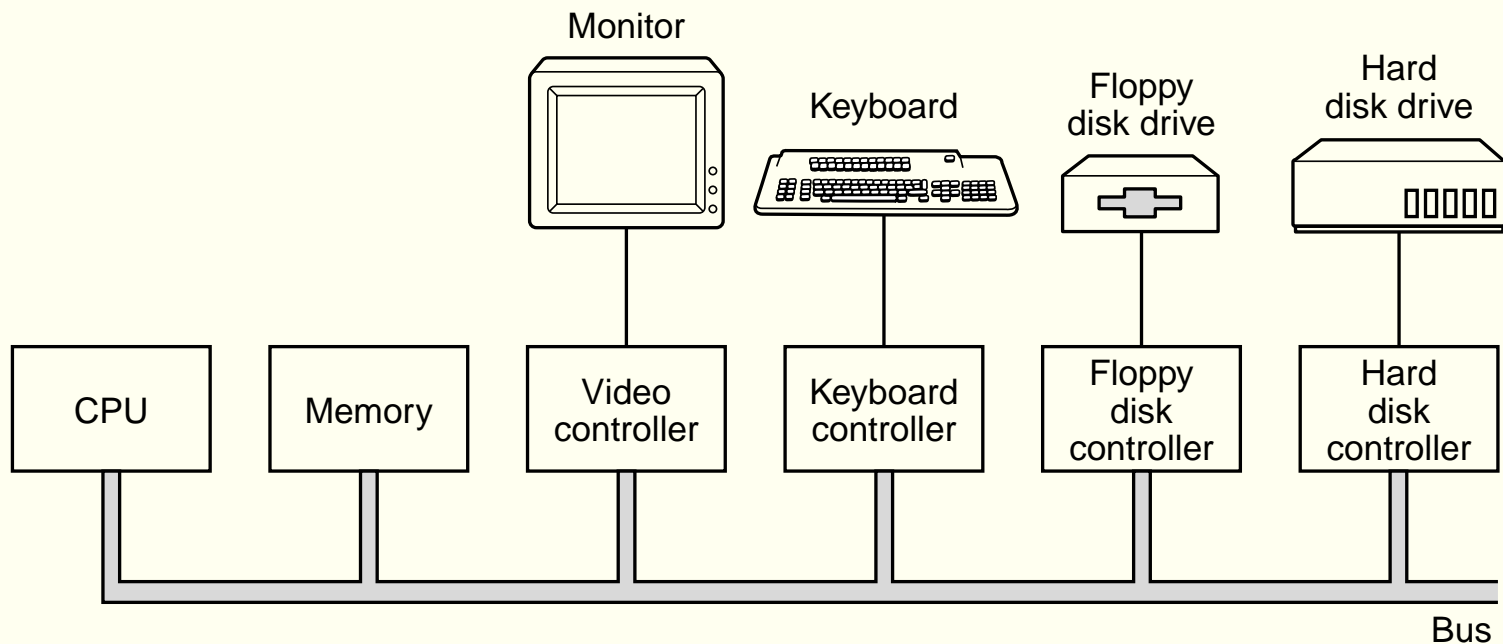
Acesso sequencial, caracter a caracter

Block device



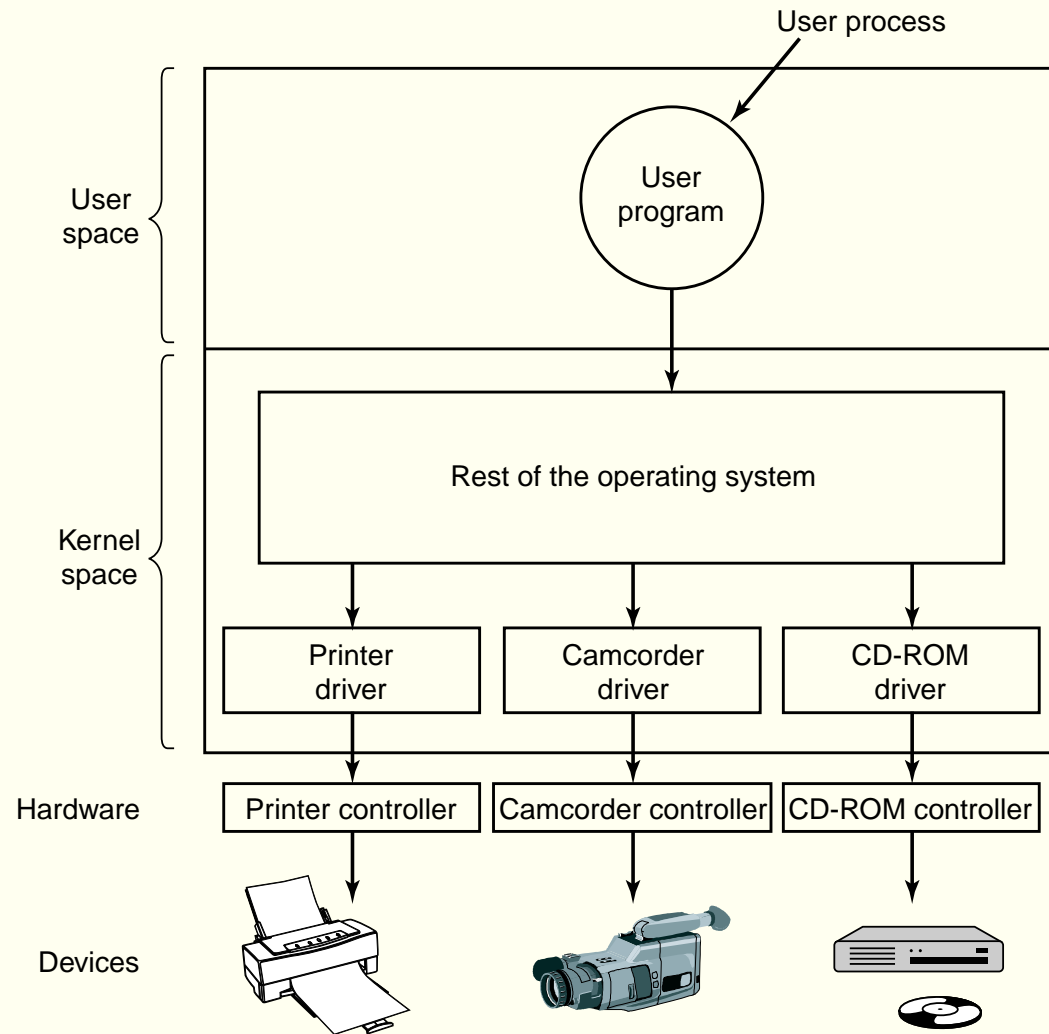
Acesso não sequencial a blocos de informação

Dispositivos de I/O e controladores



O sistema operacional deve interagir com os controladores

Device drivers

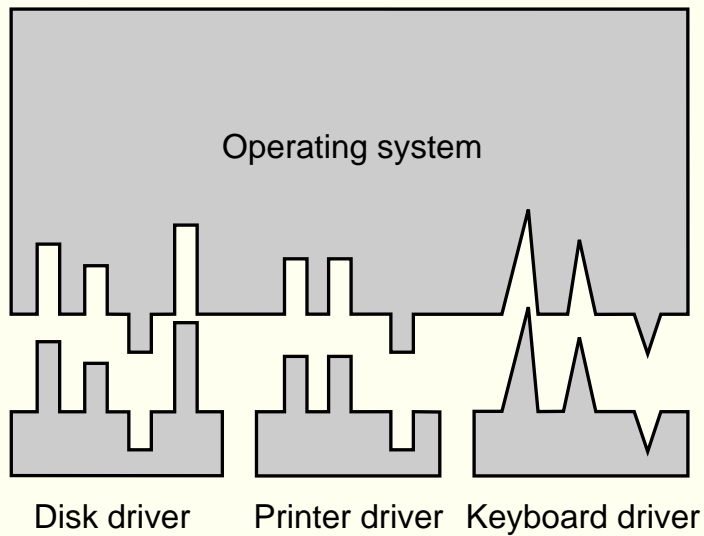


Device drivers

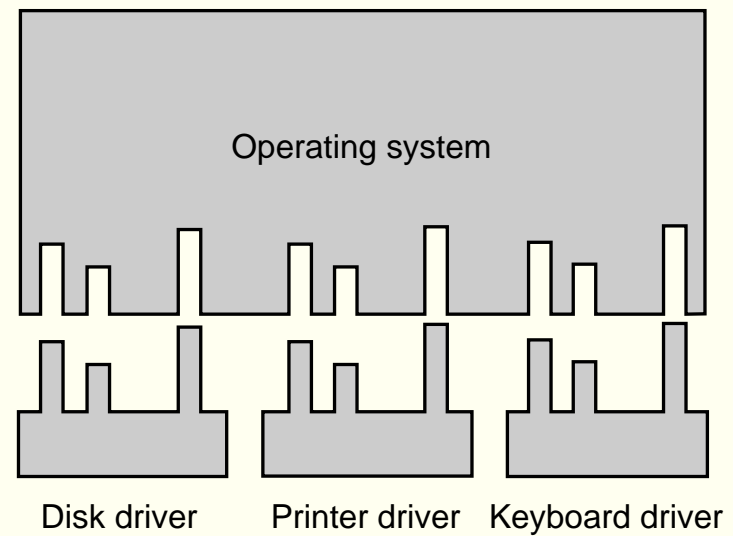
- Software que “conversa” com o controlador
- Os fabricantes devem fornecer device drivers para os sistemas operacionais
- Como acoplar um device driver ao kernel:
 - relink e reboot
 - entrada em um arquivo e reboot
 - on-the-fly

Device drivers

Interface padrão

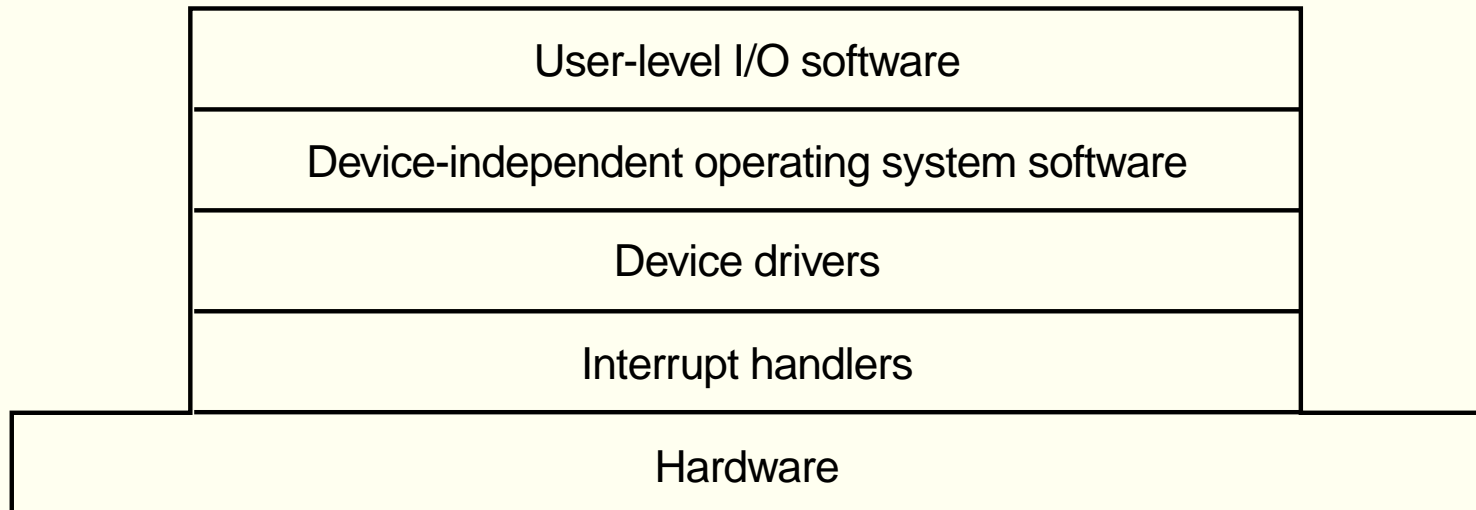


(a)

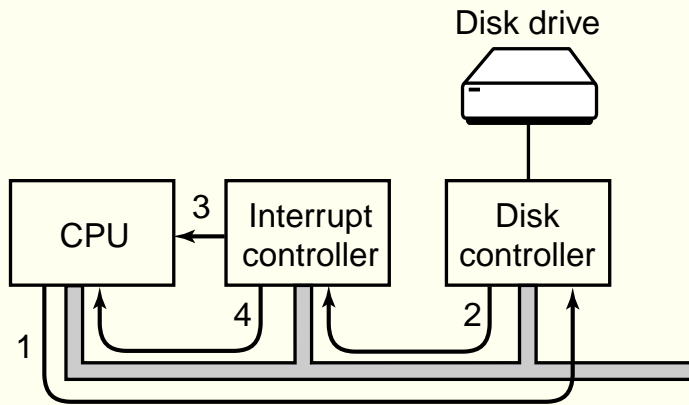


(b)

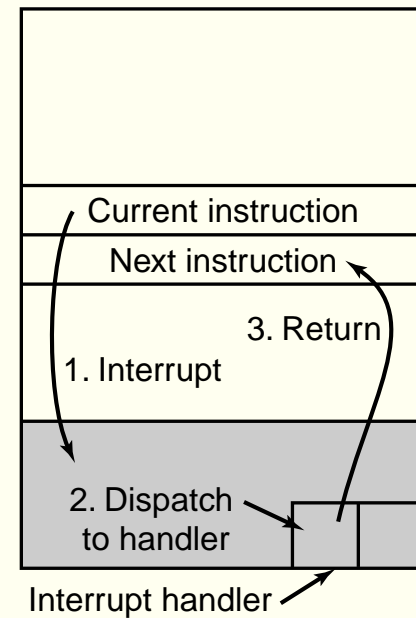
Camadas de software



Tratamento de interrupções

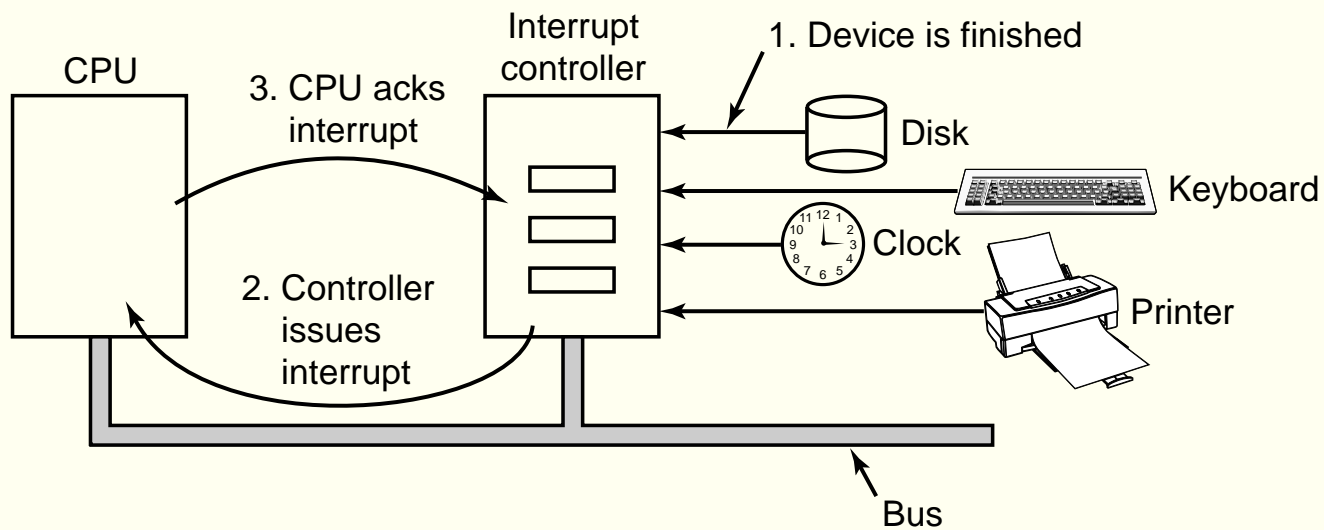


(a)



(b)

Tratamento de interrupções



Como programar os dispositivos?

- Instruções especiais

```
IN REG, PORT
```

```
OUT PORT, REG
```

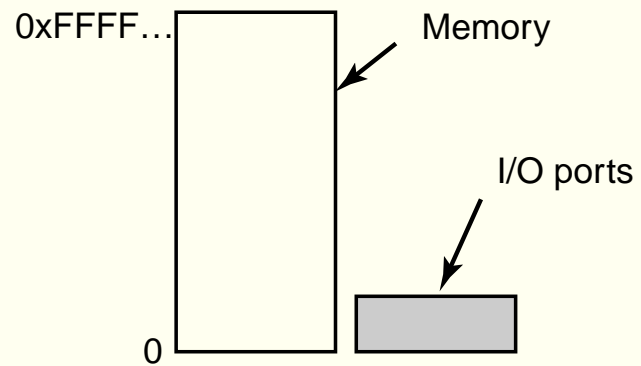
- Memory-mapped I/O

```
MOV REG, ADDR
```

Conforme o valor de ADDR, a instrução MOV fará acesso a uma palavra de memória ou dispositivo

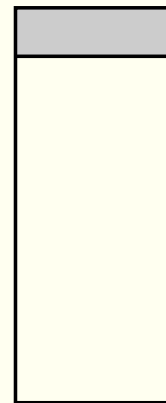
Como programar os dispositivos?

Two address



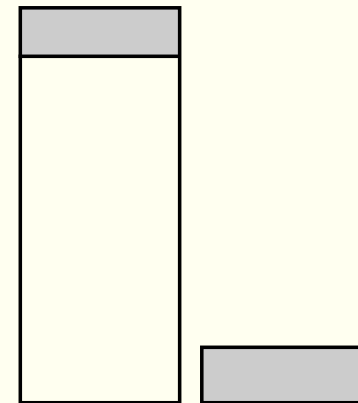
(a)

One address space



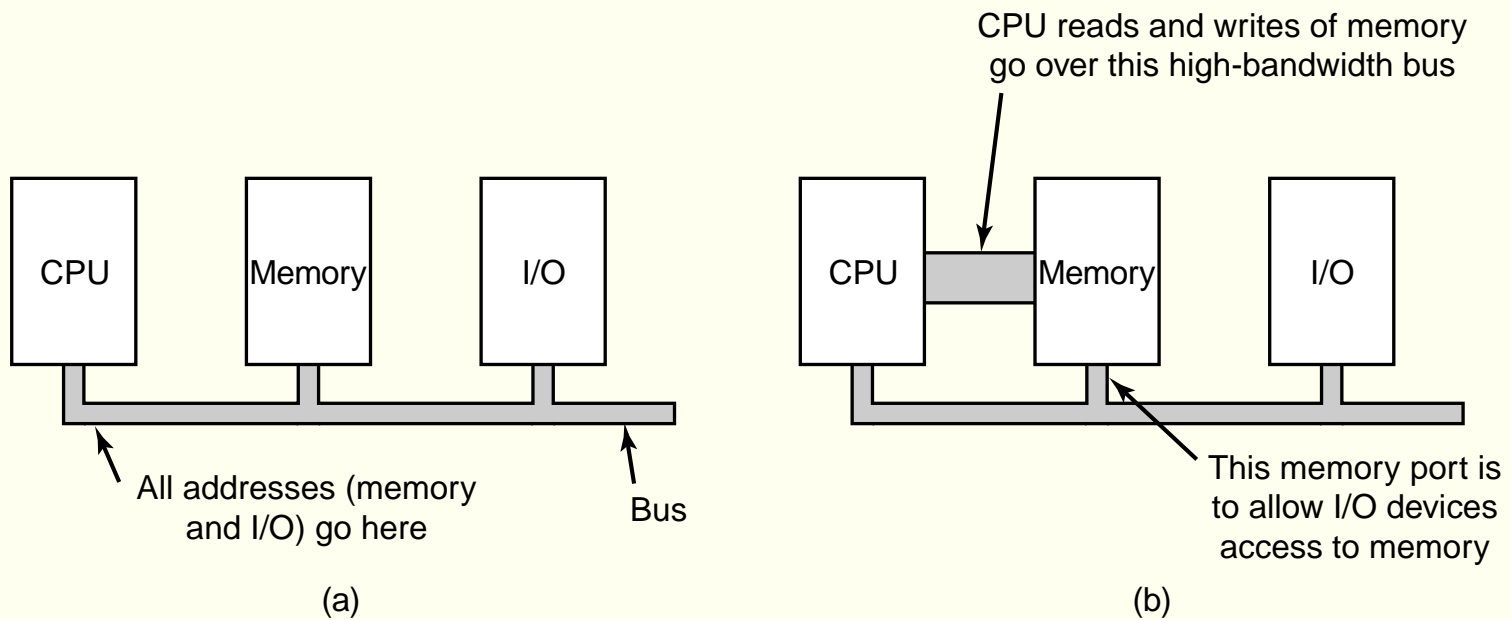
(b)

Two address spaces

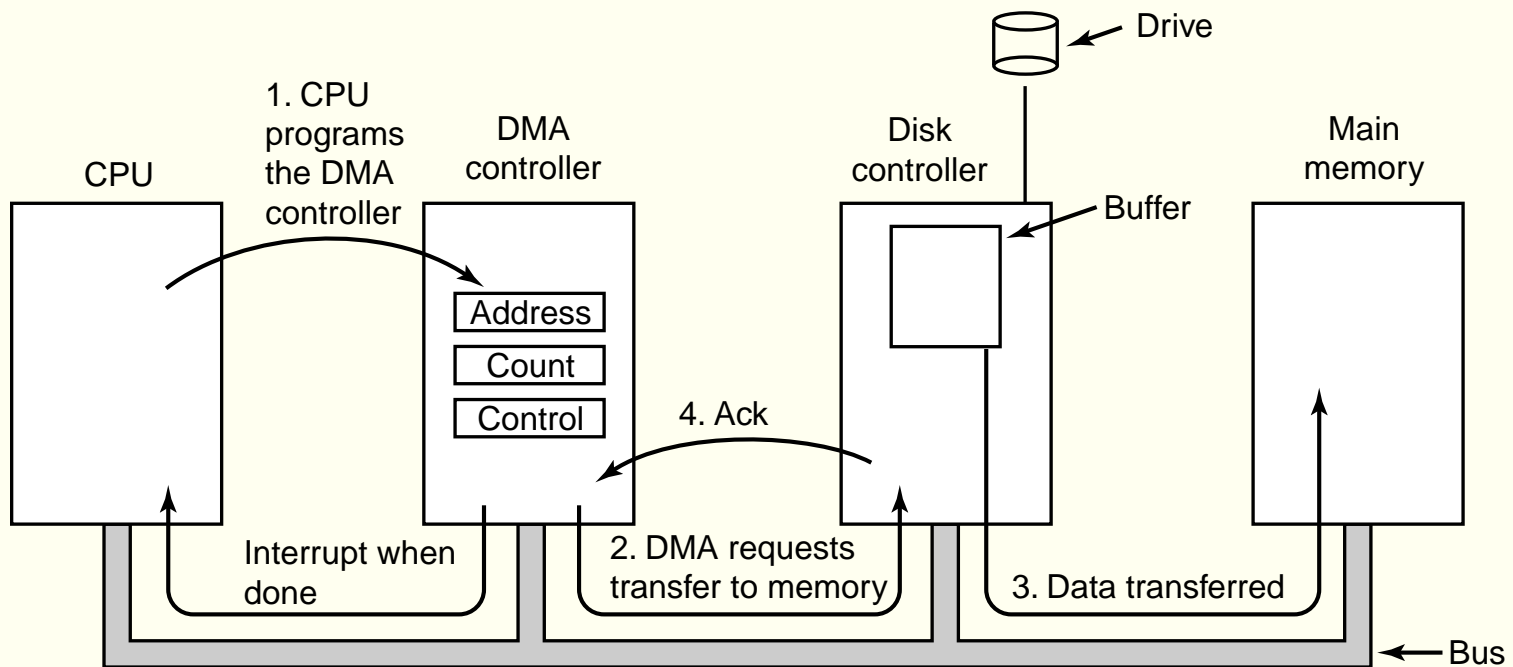


(c)

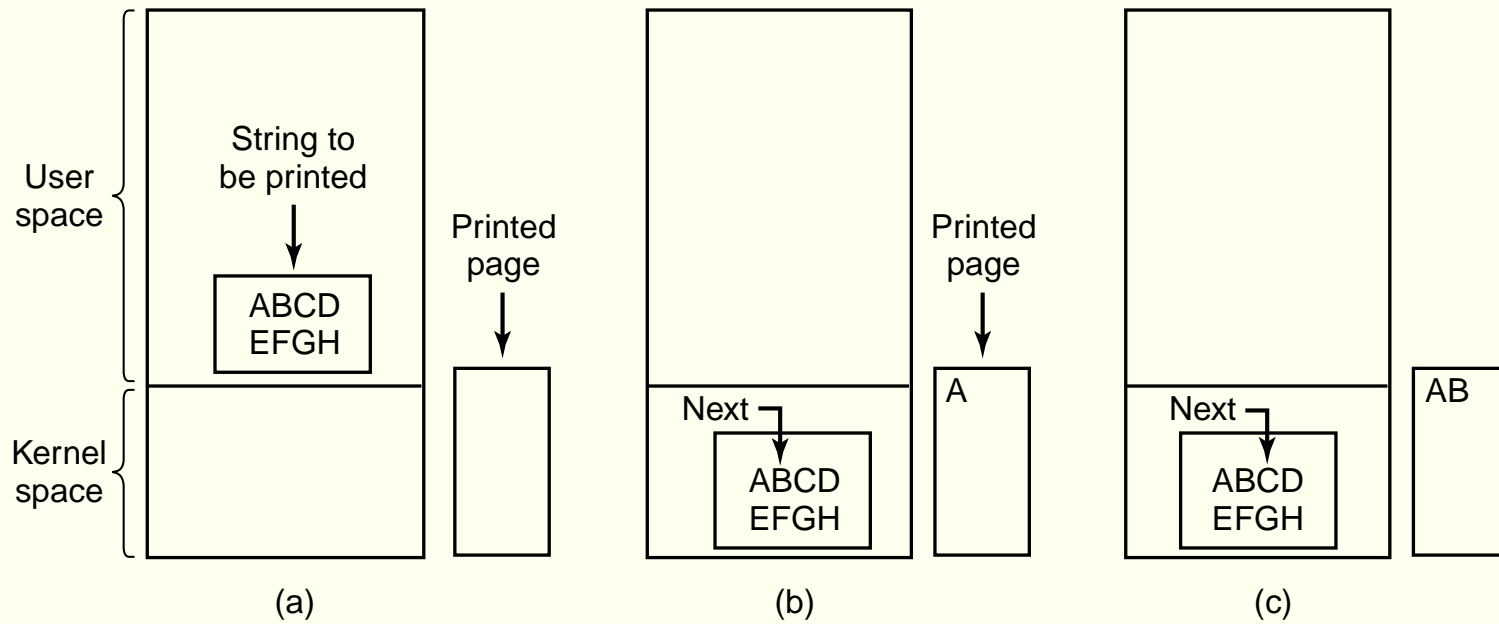
Barramento simples e dual



Direct Memory Access (DMA)



Imprimindo uma string



Imprimindo uma string

Programmed I/O

```
copy_from_user(buffer, p, count);
for (i = 0; i < count; i++) {
    while (*printer_status_reg != READY) ;
    *printer_data_register = p[i];
}
return_to_user();
```

/ p is the kernel bufer */*
/ loop on every character */*
/ loop until ready */*
/ output one character */*

Trecho de código do kernel

Imprimindo uma string

Interrupt-driven I/O

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

(a)

(a) Trecho de código do kernel

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

(b)

(b) Tratador da interrupção

Imprimindo uma string

DMA

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

(a)

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

(b)

(a) Trecho de código do kernel

(b) Tratador de interrupção