Distributed Operating Systems

Input/Output

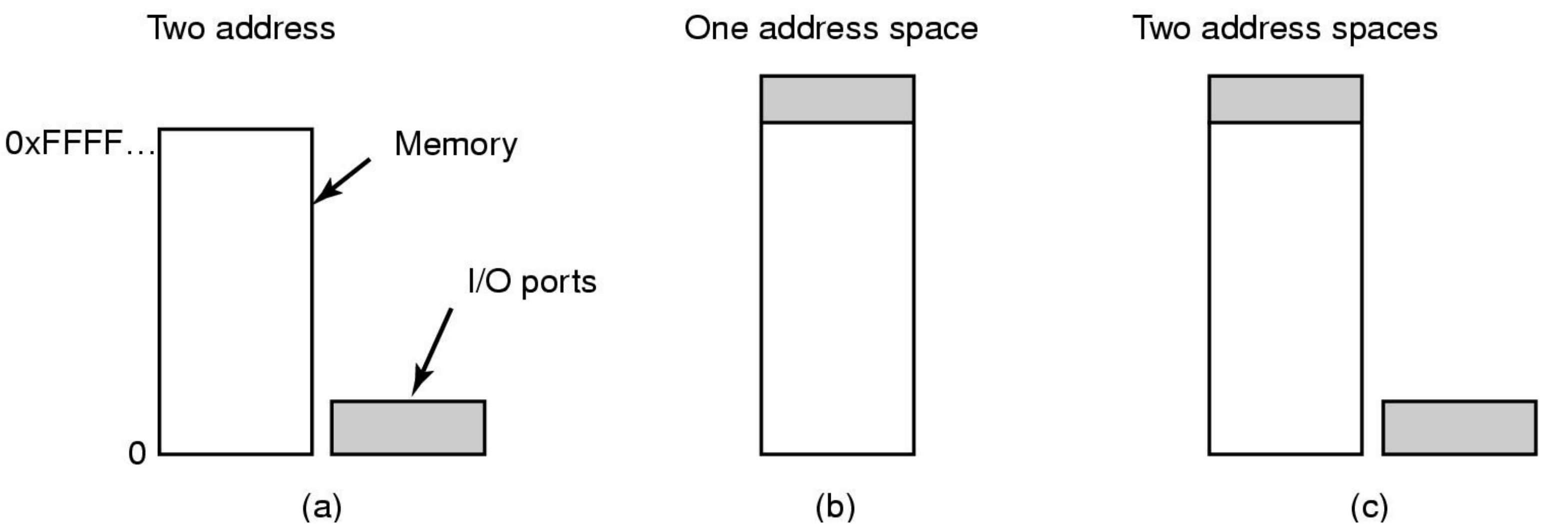
Topics

- Principles of I/O hardware
- Principles of I/O software
- I/O software layers
- Disks
- Clocks

Device Controllers

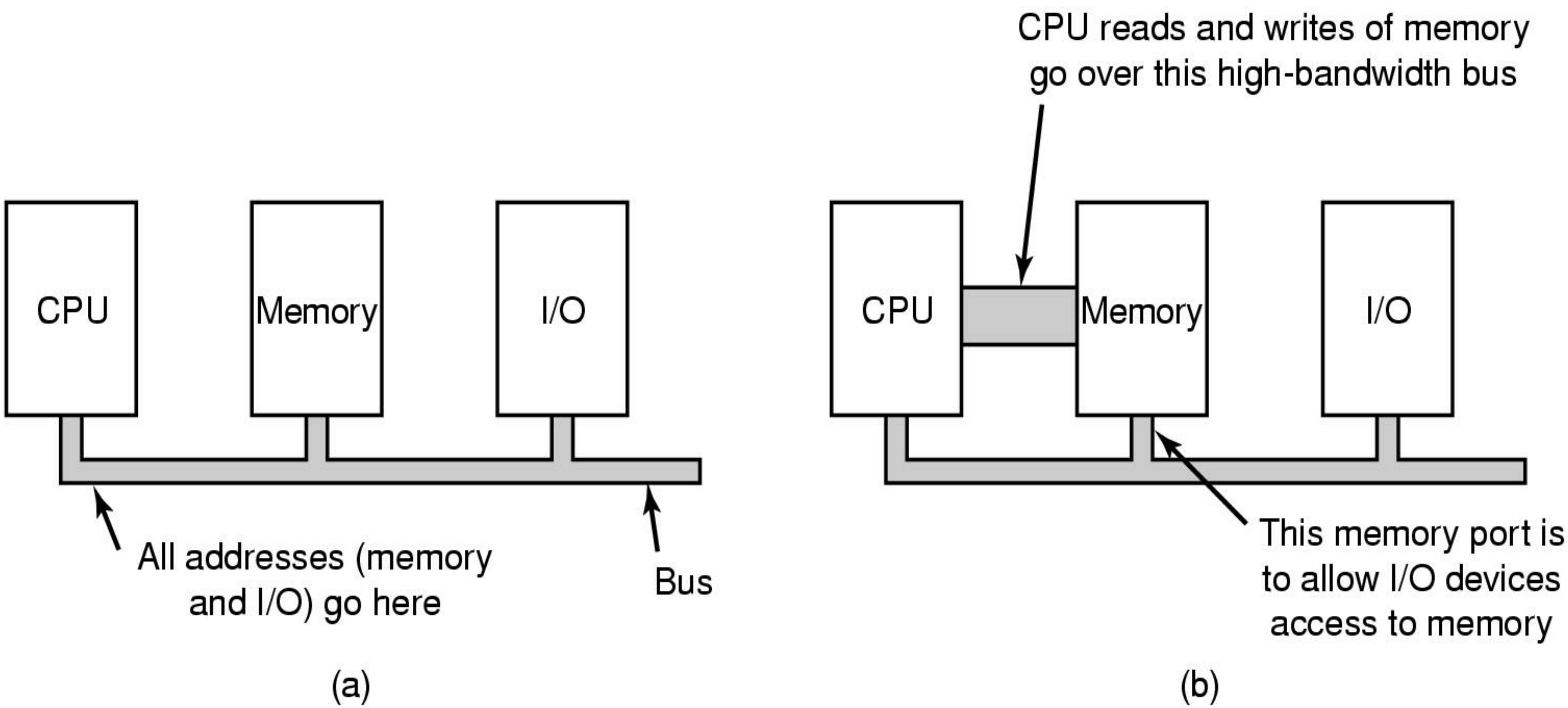
- I/O devices have components:
  - mechanical component
  - electronic component
- The electronic component is the device controller
  - may be able to handle multiple devices
- Controller's tasks
  - convert serial bit stream to block of bytes
  - perform error correction as necessary
  - access the main memory

Memory-Mapped I/O (1)



- Separate I/O and memory space
- Memory-mapped I/O
- Hybrid

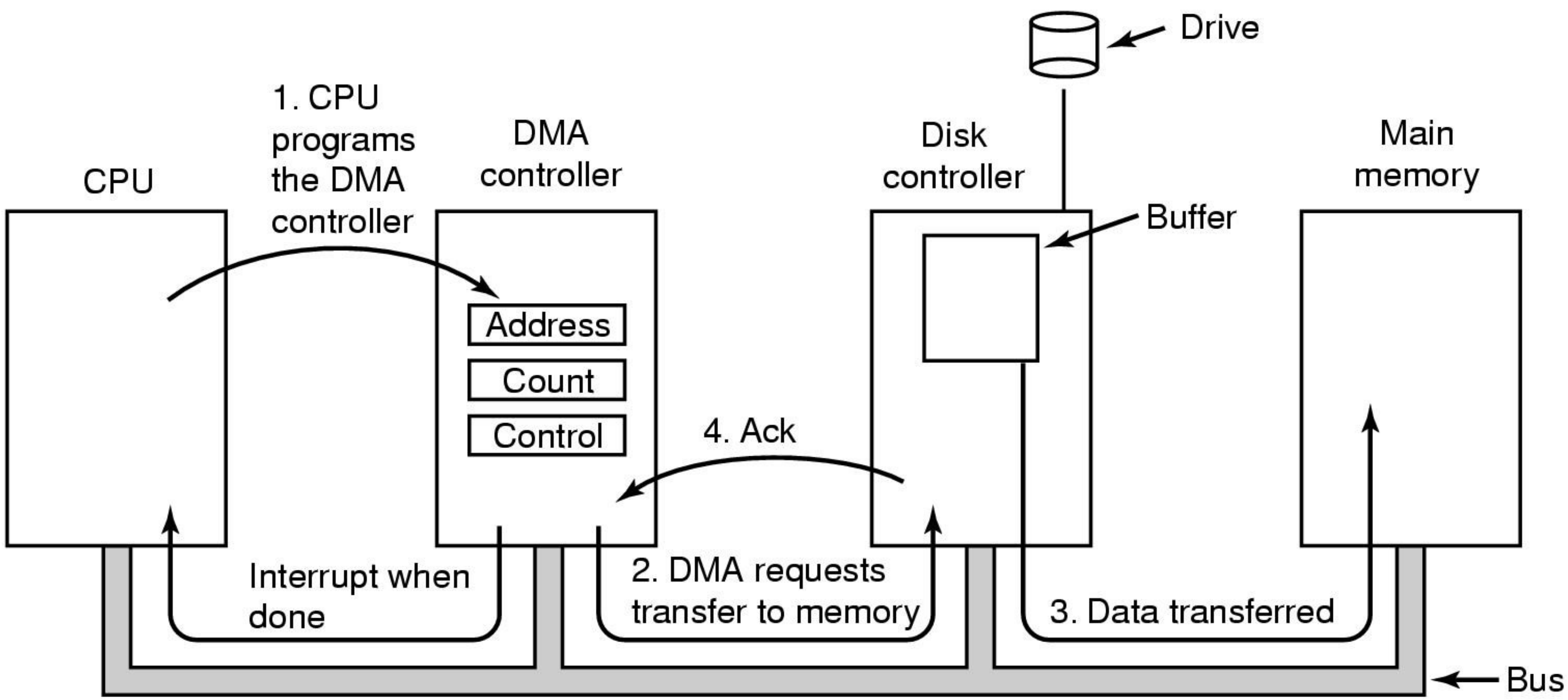
Memory-Mapped I/O (2)



(a) A single-bus architecture

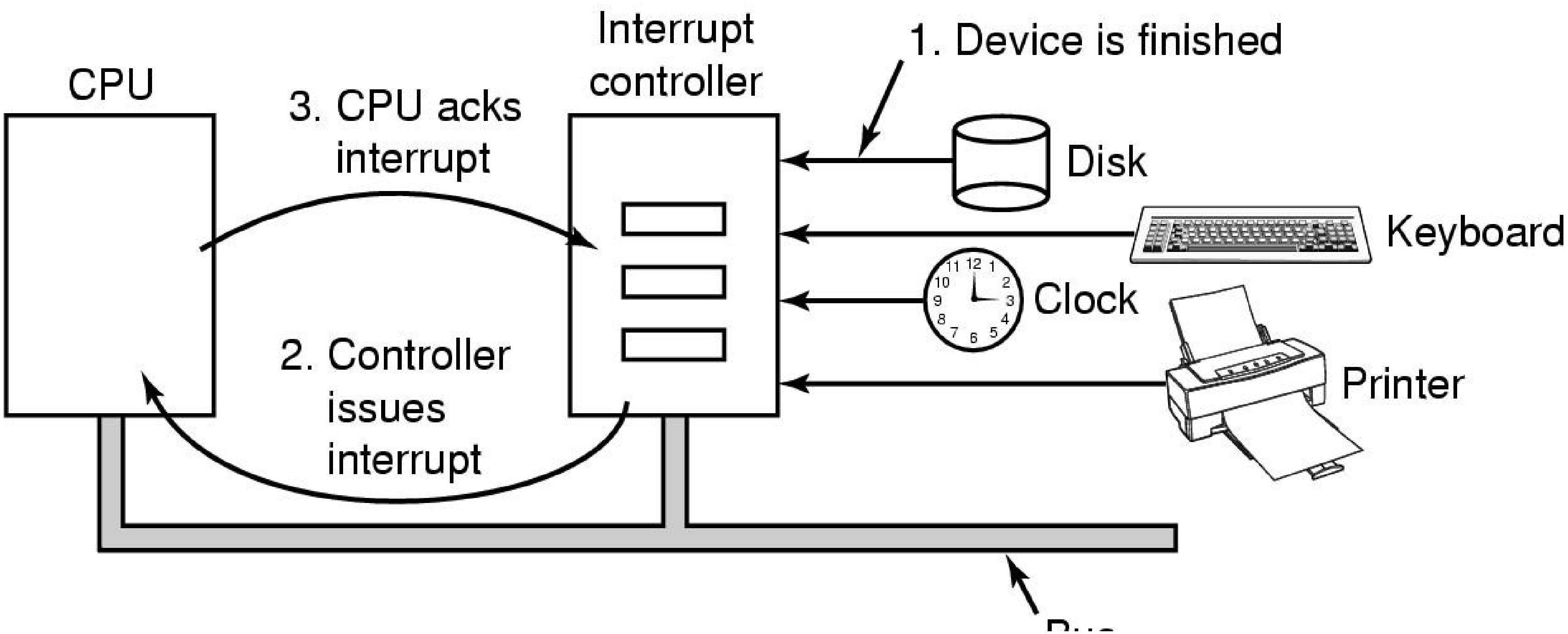
(b) A dual-bus memory architecture

Direct Memory Access (DMA)



Operation of a DMA transfer

Interrupts Revisited



How interrupts happens. Connections between devices and interrupt controller actually use interrupt lines on the bus rather than dedicated wires

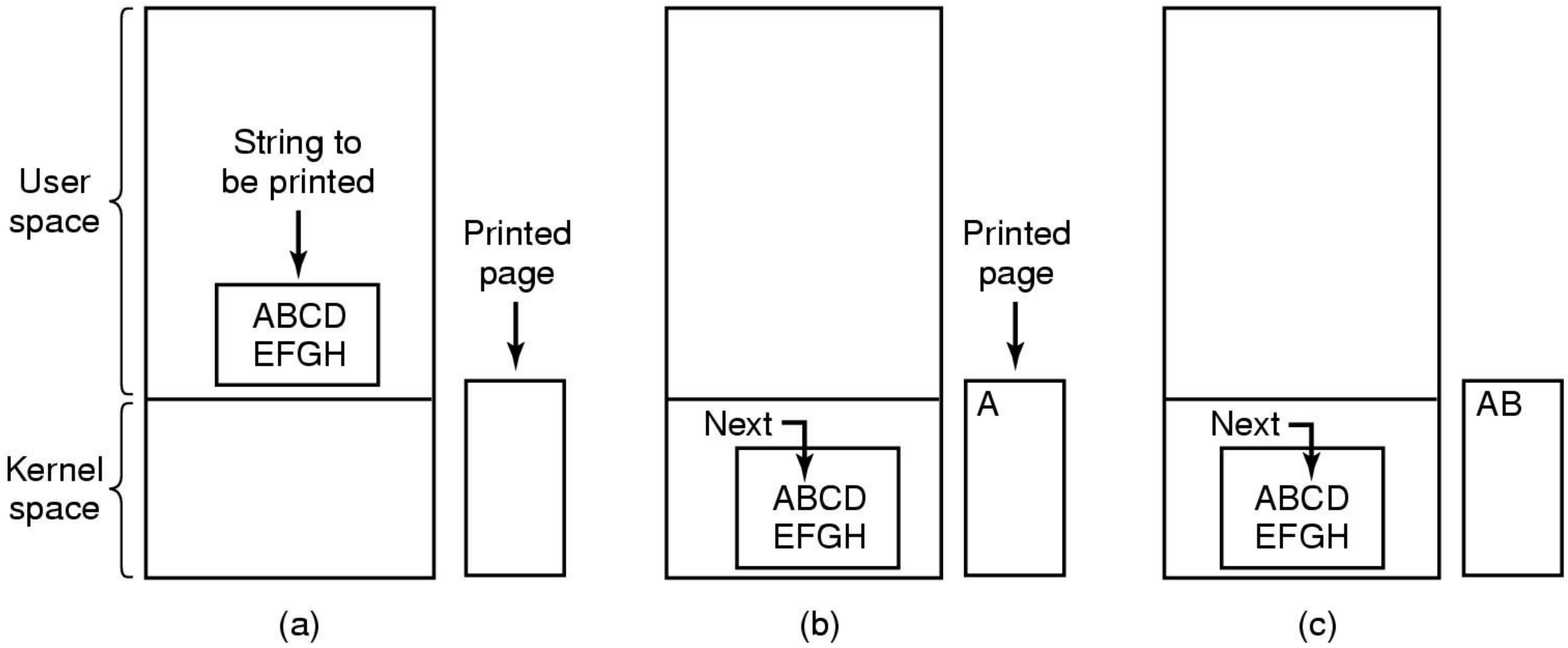
Principles of I/O Software
Goals of I/O Software (1)

- Device independence
  - programs can access any I/O device
  - without specifying device in advance
    - (floppy, hard drive, or CD-ROM)
- Uniform naming
  - name of a file or device a string or an integer
  - not depending on which machine
- Error handling
  - handle as close to the hardware as possible

Goals of I/O Software (2)

- Synchronous vs. asynchronous transfers
  - blocked transfers vs. interrupt-driven
- Buffering
  - data coming off a device cannot be stored in final destination
- Sharable vs. dedicated devices
  - disks are sharable
  - tape drives would not be

Programmed I/O (1)



Steps in printing a string

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( );
```

Writing a string to the printer using 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)

- Writing a string to the printer using interrupt-driven I/O
  - Code executed when print system call is made
  - Interrupt service procedure

I/O Using DMA

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

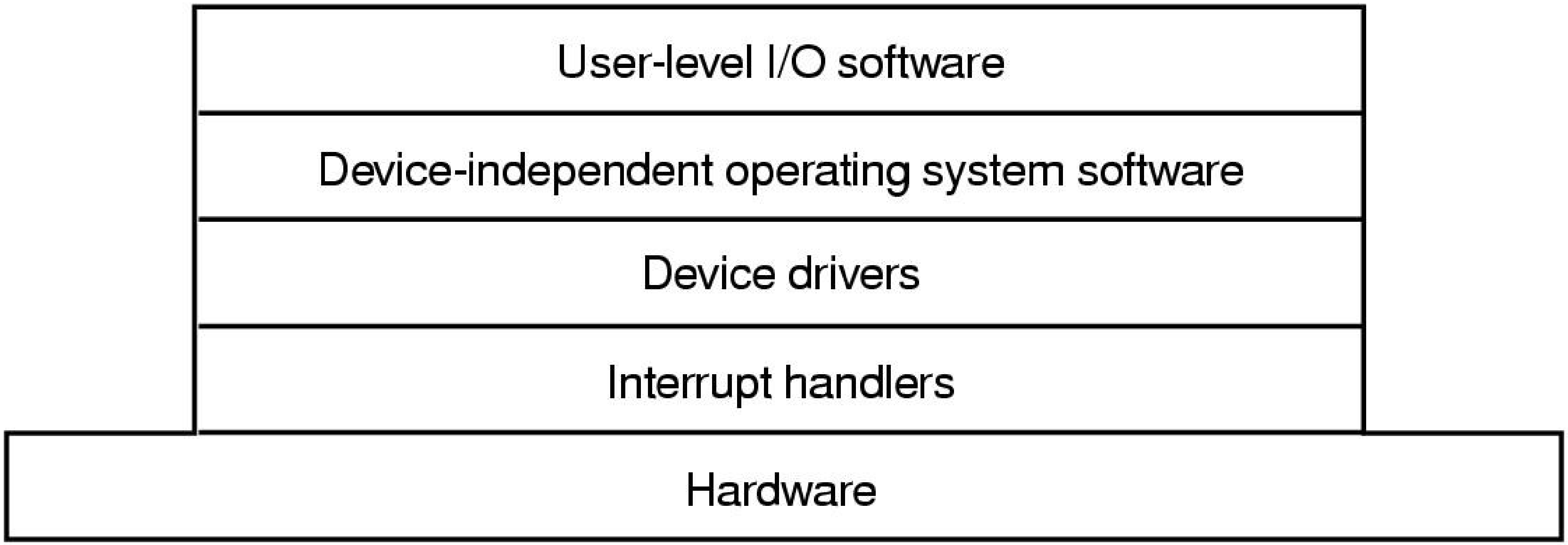
(a)

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

(b)

- Printing a string using DMA
  - code executed when the print system call is made
  - interrupt service procedure

I/O Software Layers



Layers of the I/O Software System

Interrupt Handlers

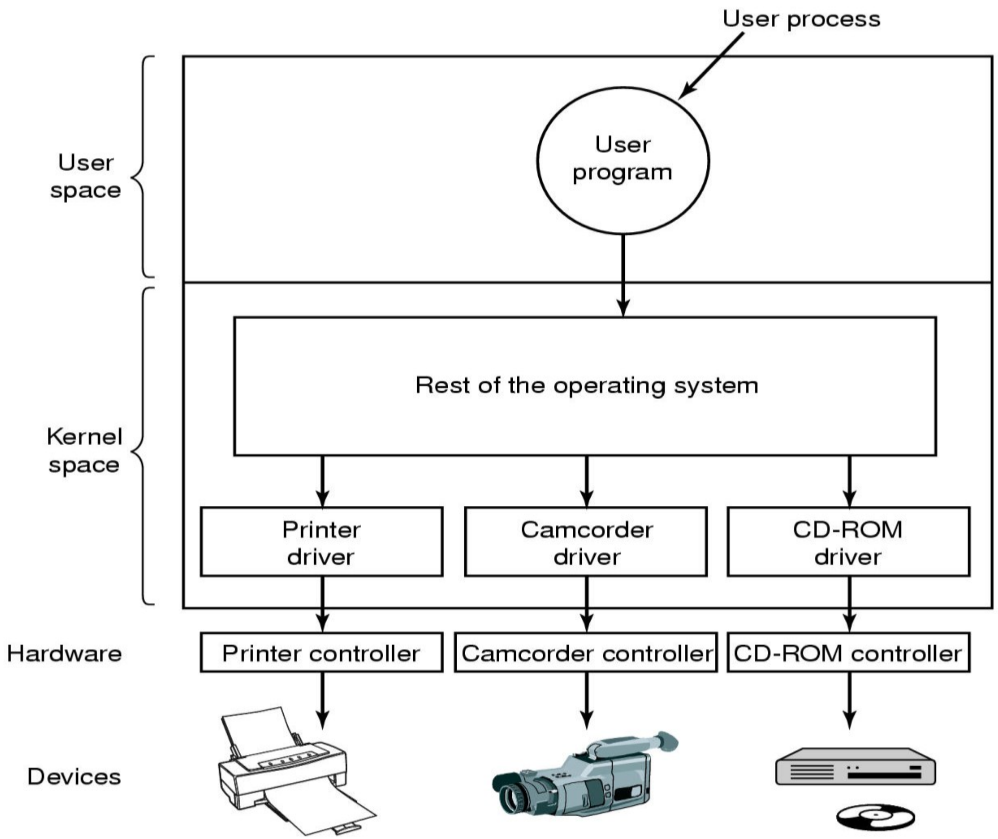
- Interrupt handlers blocks the driver starting an I/O operation until interrupt notifies of completion

- Interrupt procedure does its task then unblocks driver that started it

- Steps must be performed in software after interrupt completed

1. Save registers
2. Set up context for interrupt service procedure

Device Drivers



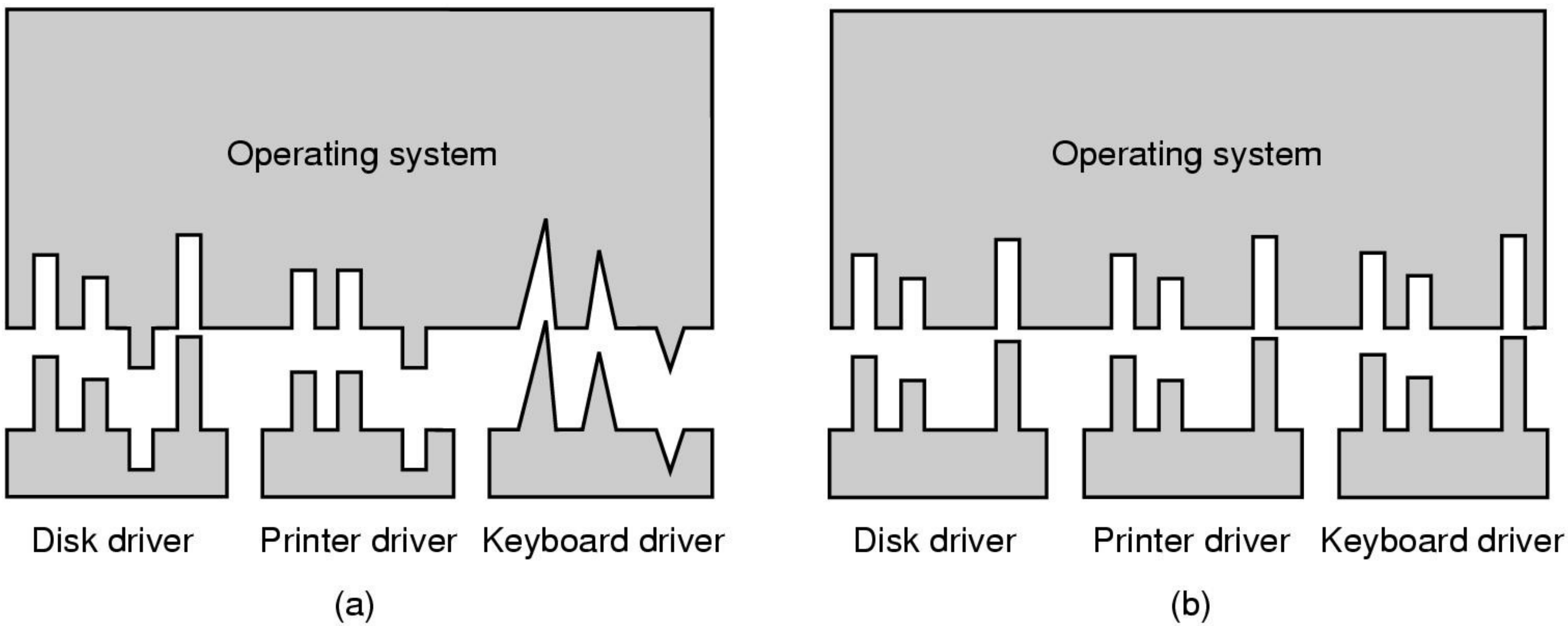
- Logical position of device drivers is shown here
- Communications between drivers and device controllers goes over the bus

Device-Independent I/O Software (1)

Uniform interfacing for device drivers
Buffering
Error reporting
Allocating and releasing dedicated devices
Providing a device-independent block size

Functions of the device-independent I/O software

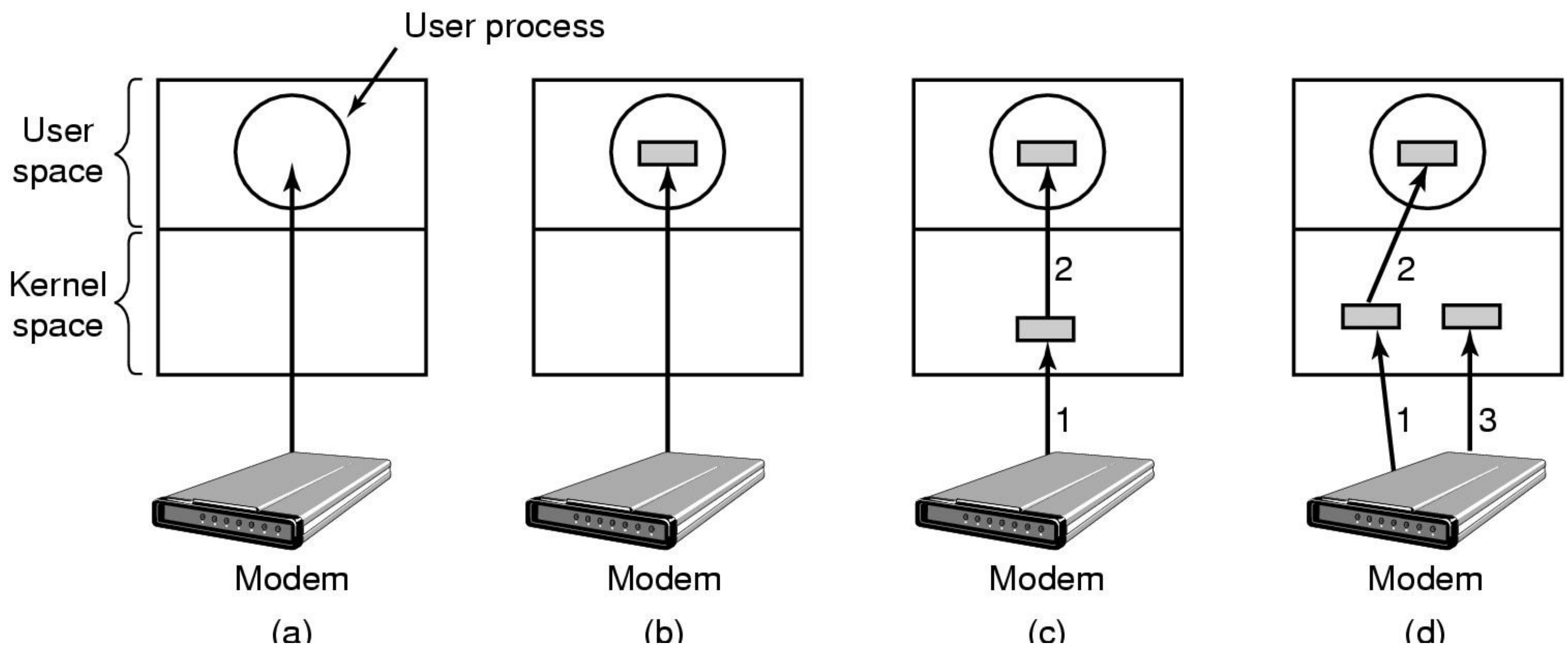
Device-Independent I/O Software (2)



(a) Without a standard driver interface

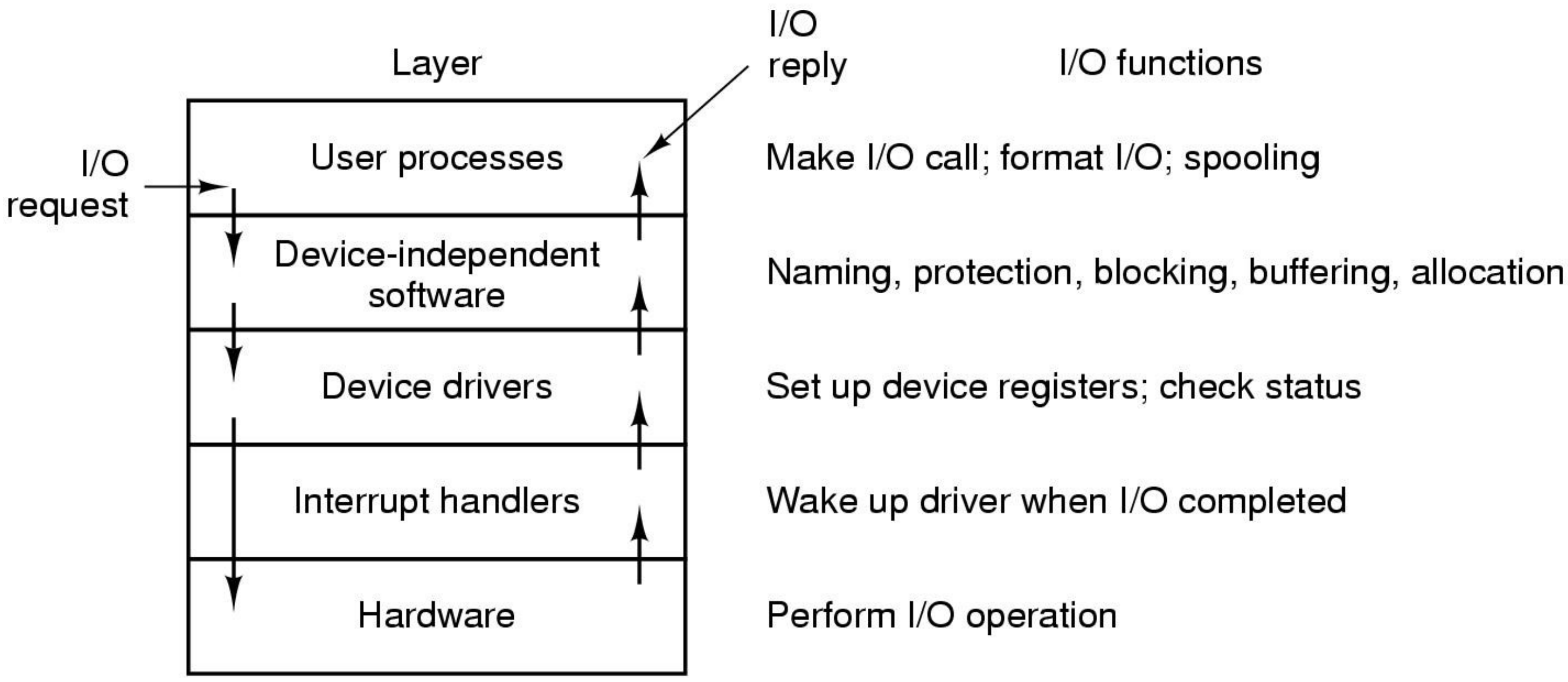
(b) With a standard driver interface

Device-Independent I/O Software (3)



(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

User-Space I/O Software



Layers of the I/O system and the main functions of each layer

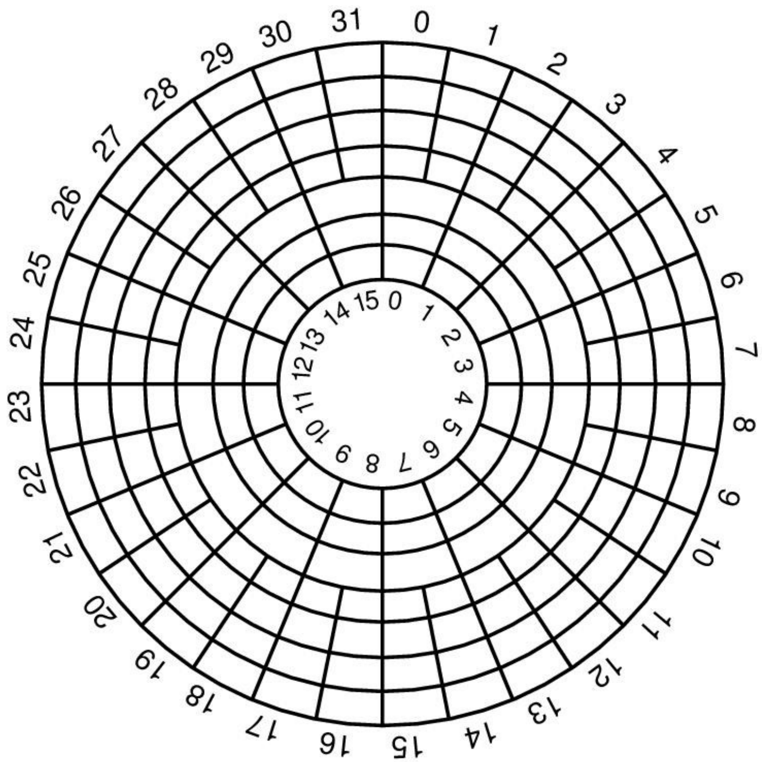
Disks

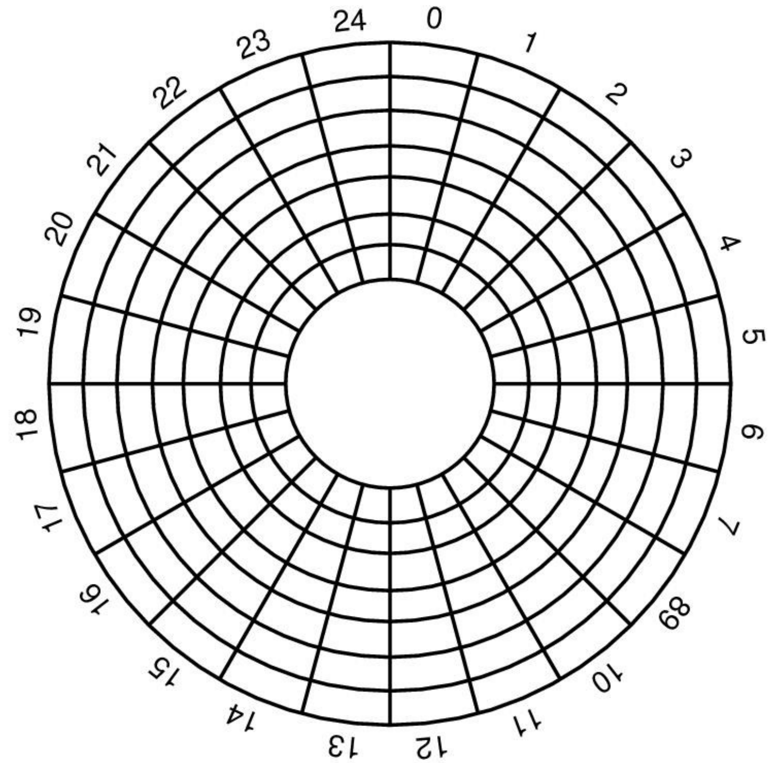
Disk Hardware (1)

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

Disk parameters for the original IBM PC floppy disk and a Western Digital WD 18300 hard disk

Disk Hardware (2)



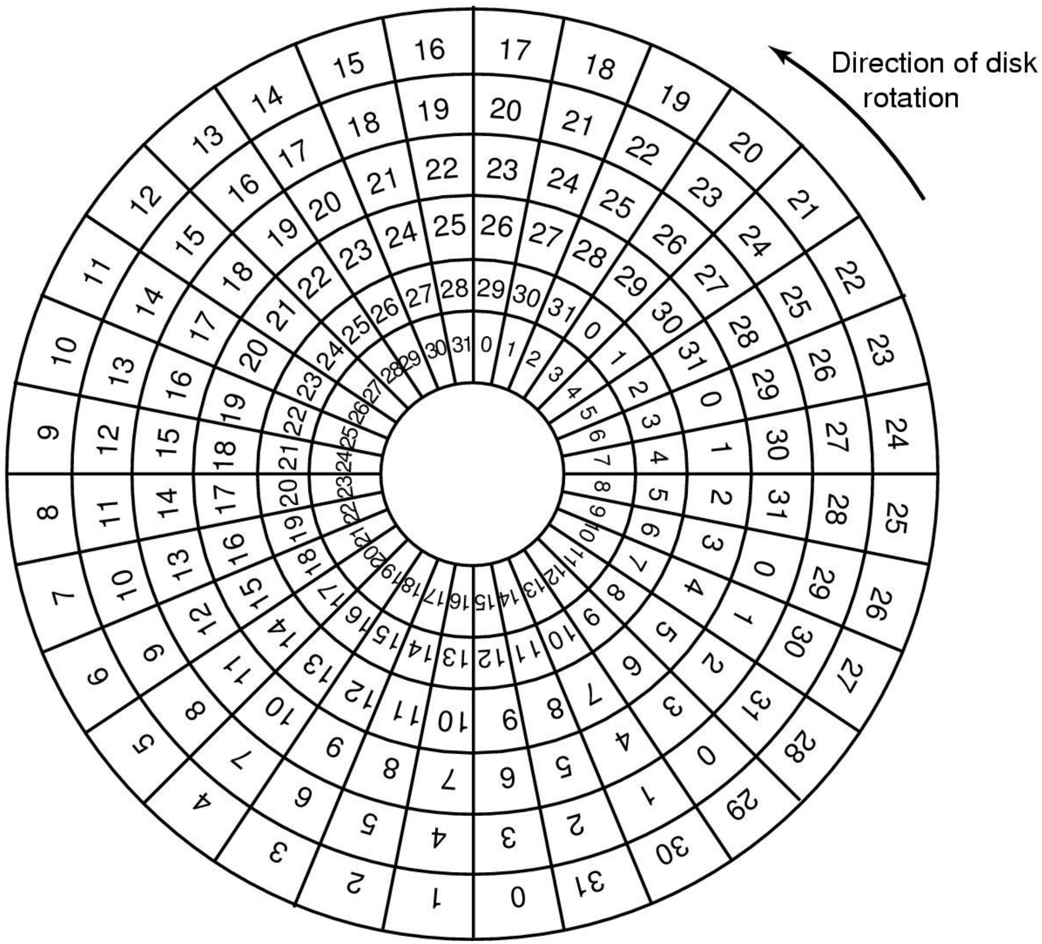


- Physical geometry of a disk with two zones
- A possible virtual geometry for this disk

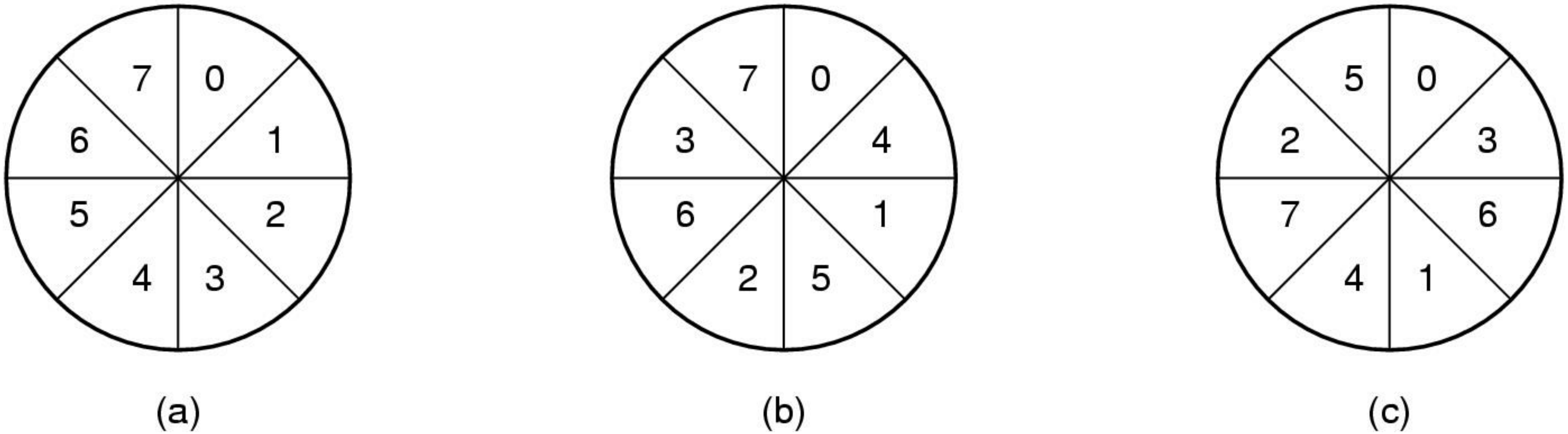
Disk Formatting (1)

A disk sector

Disk Formatting (2)



Disk Formatting (3)

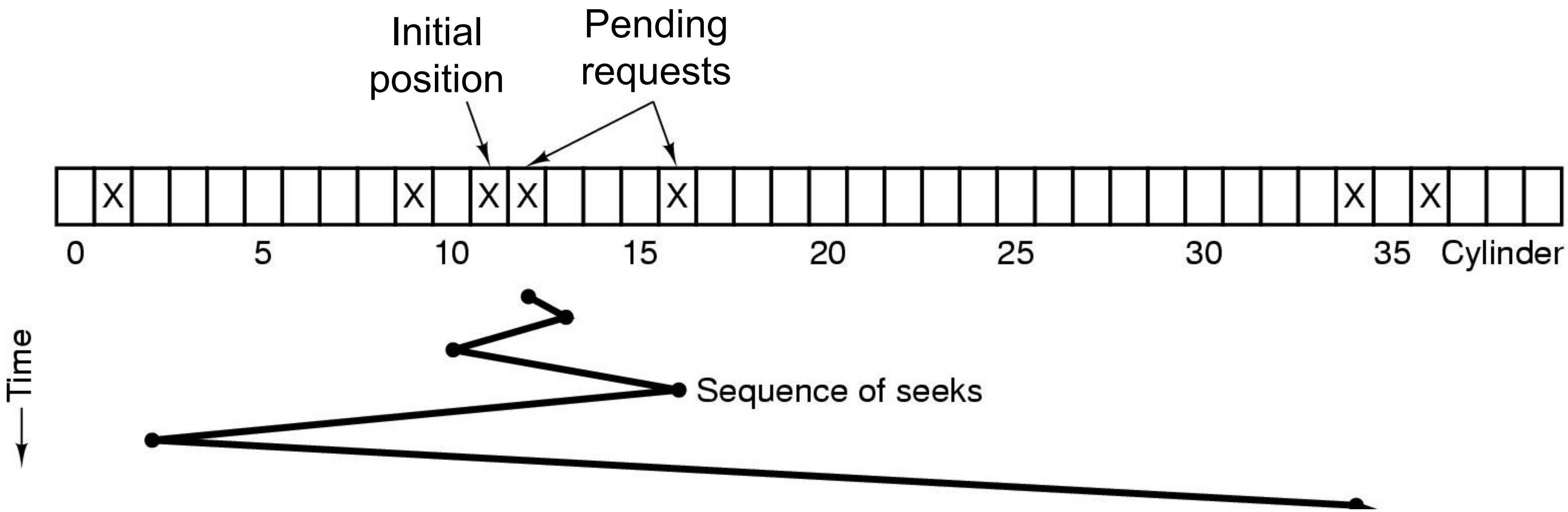


- No interleaving
- Single interleaving
- Double interleaving

Disk Arm Scheduling Algorithms (1)

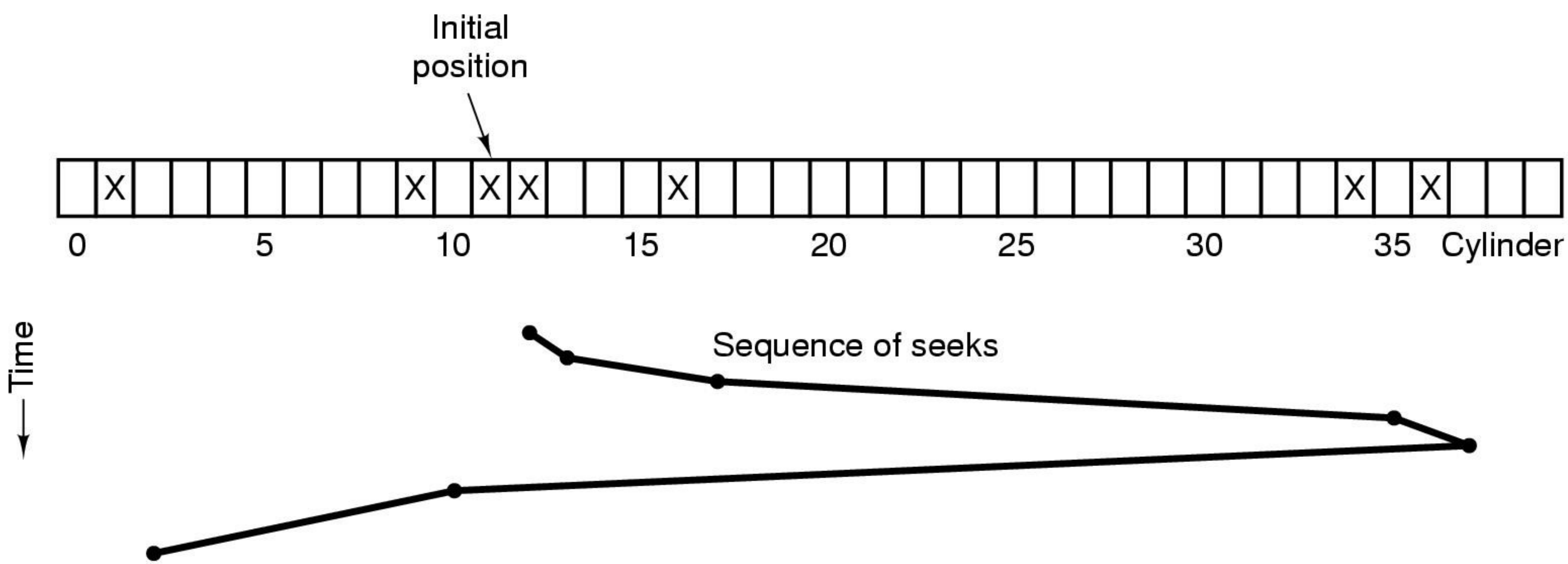
- Time required to read or write a disk block determined by 3 factors
    1. Seek time
    2. Rotational delay
    3. Actual transfer time
- Seek time dominates
- Error checking is done by controllers

Disk Arm Scheduling Algorithms (2)



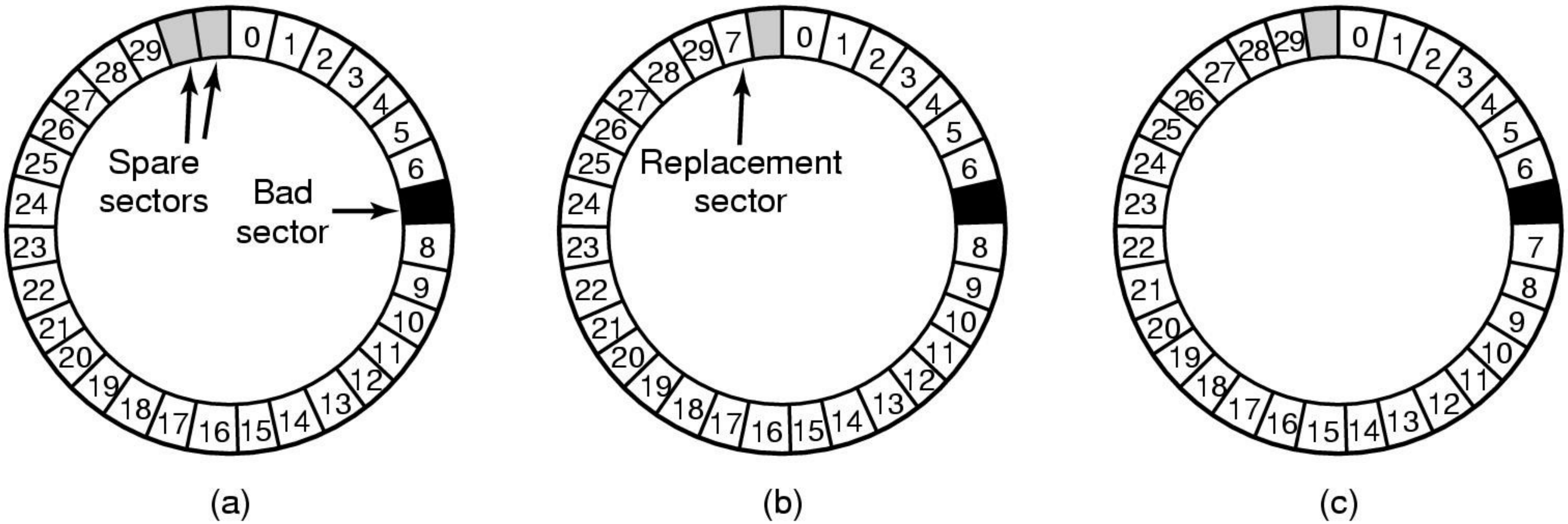
Shortest Seek First (SSF) disk scheduling algorithm

Disk Arm Scheduling Algorithms (3)



The elevator algorithm for scheduling disk requests

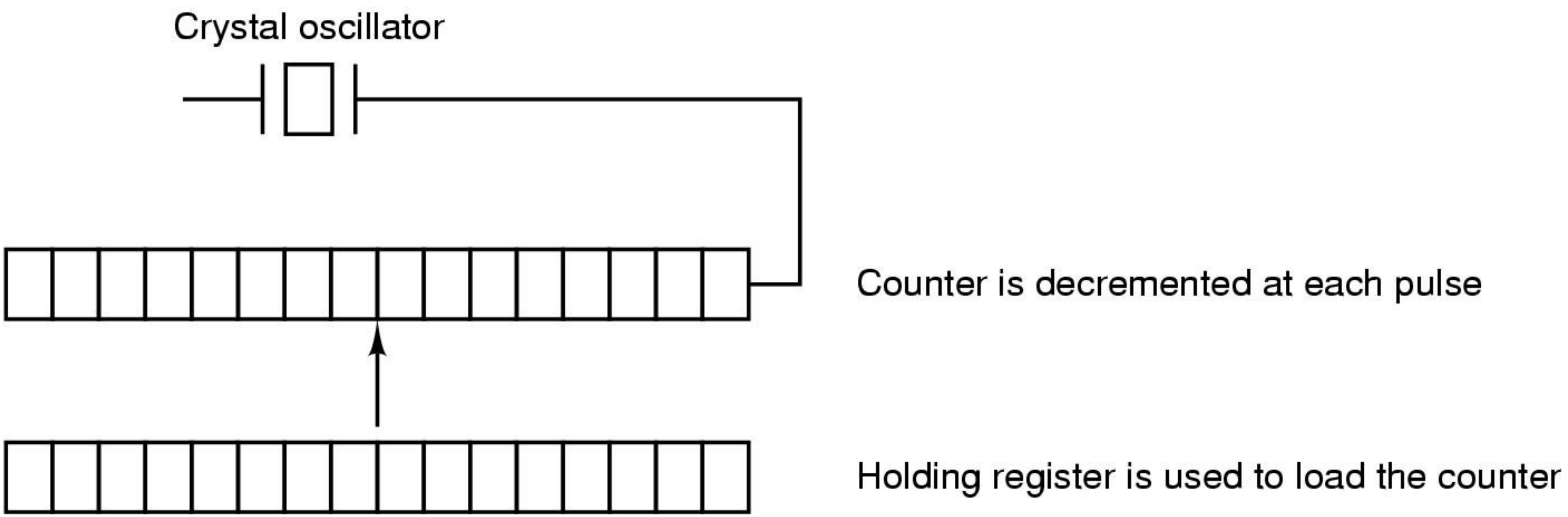
Error Handling



- A disk track with a bad sector
- Substituting a spare for the bad sector
- Shifting all the sectors to bypass the bad one

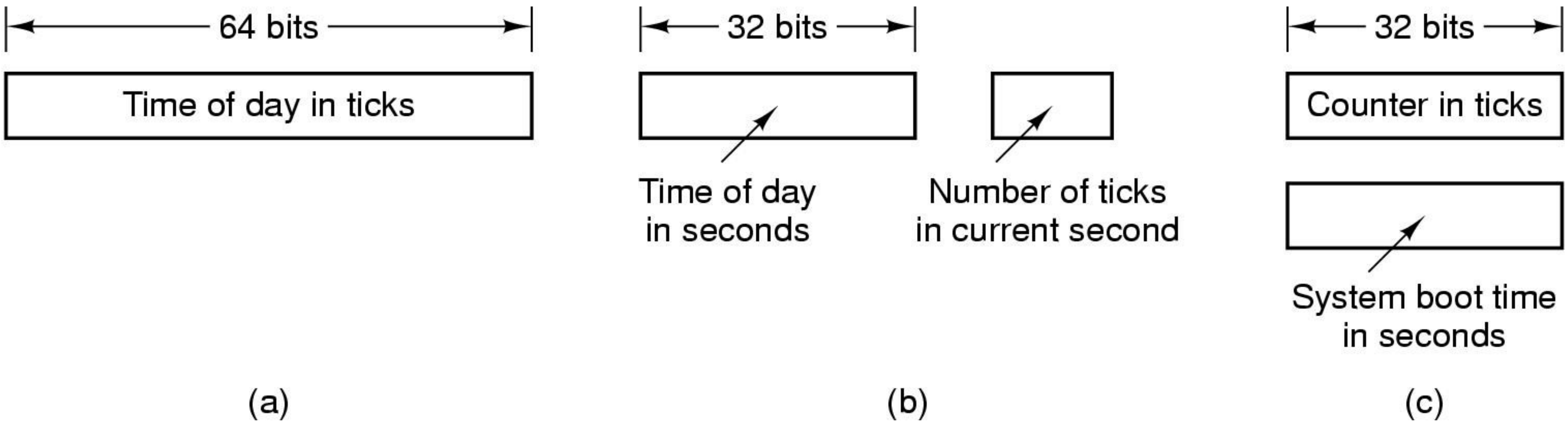
Clocks

Clock Hardware



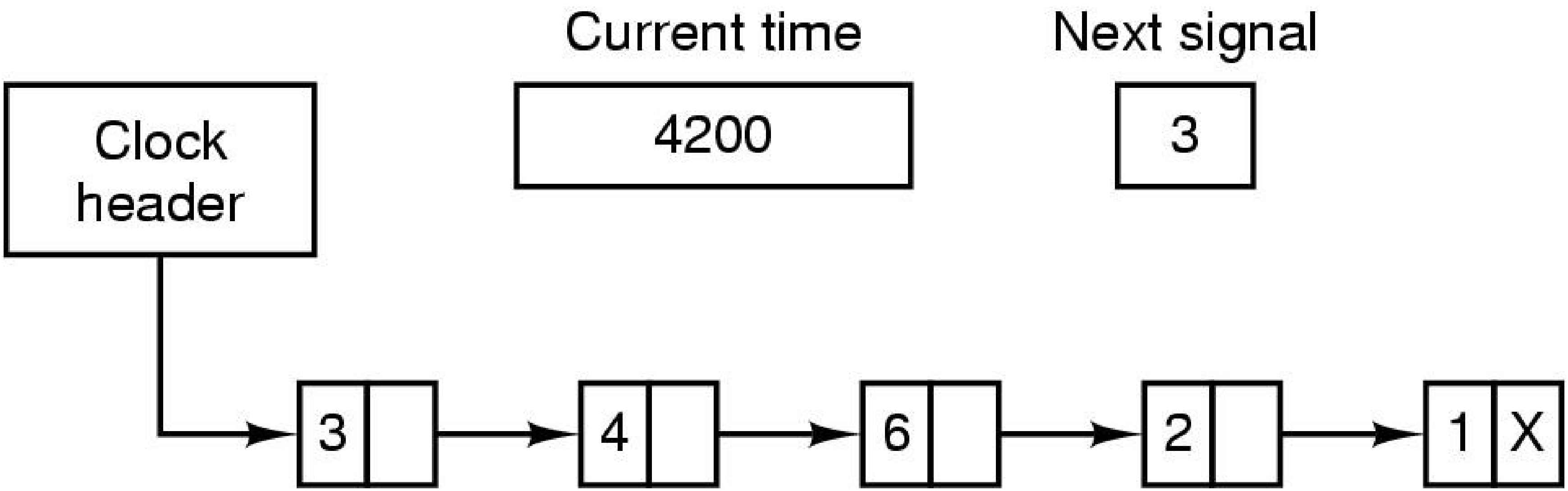
A programmable clock

Clock Software (1)



Three ways to maintain the time of day

Clock Software (2)



Simulating multiple timers with a single clock

Soft Timers

- A second clock available for timer interrupts
  - specified by applications
  - no problems if interrupt frequency is low

- Soft timers avoid interrupts
  - kernel checks for soft timer expiration before it exits to user mode
  - how well this works depends on rate of kernel entries

Questions?