Part 5
Input/Output

5.1 Principles of I/O hardware
5.2 Principles of I/O software
5.3 I/O software layers
5.4 Disks
5.5 Clocks
5.6 User Intrefaces: Keyboard, Mouse, Monitor
5.7 Power Management

5.1 Principles of I/O hardware

Principles of I/O Hardware
Types of I/O devices

Two main groups: Block and Character Devices

- Block devices include disk drives
  - Commands include read, write, seek
  - Raw I/O or file-system access
  - Memory-mapped file access possible
- Character devices include keyboards, mice, serial ports
  - Commands include `get, put`
  - Libraries layered on top allow line editing

Principles of I/O Hardware
Types of I/O devices

Some typical device, network, and data base rates

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

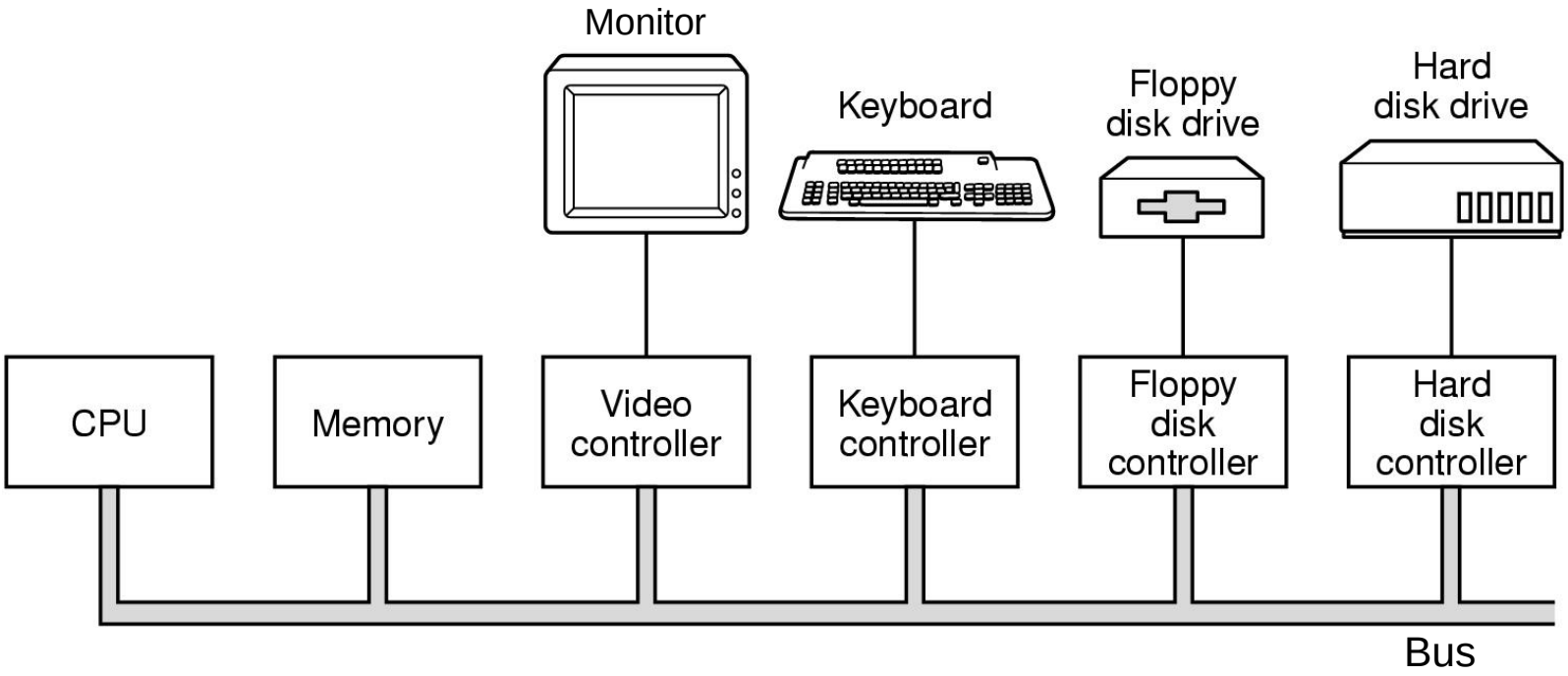
Principles of I/O Hardware

Common concepts

Common concepts

- I/O Device Controller
- I/O Port
- I/O Bus

Principles of I/O Hardware
Device Controllers



- Components of a simple personal computer

Principles of I/O Hardware

Device Controllers

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

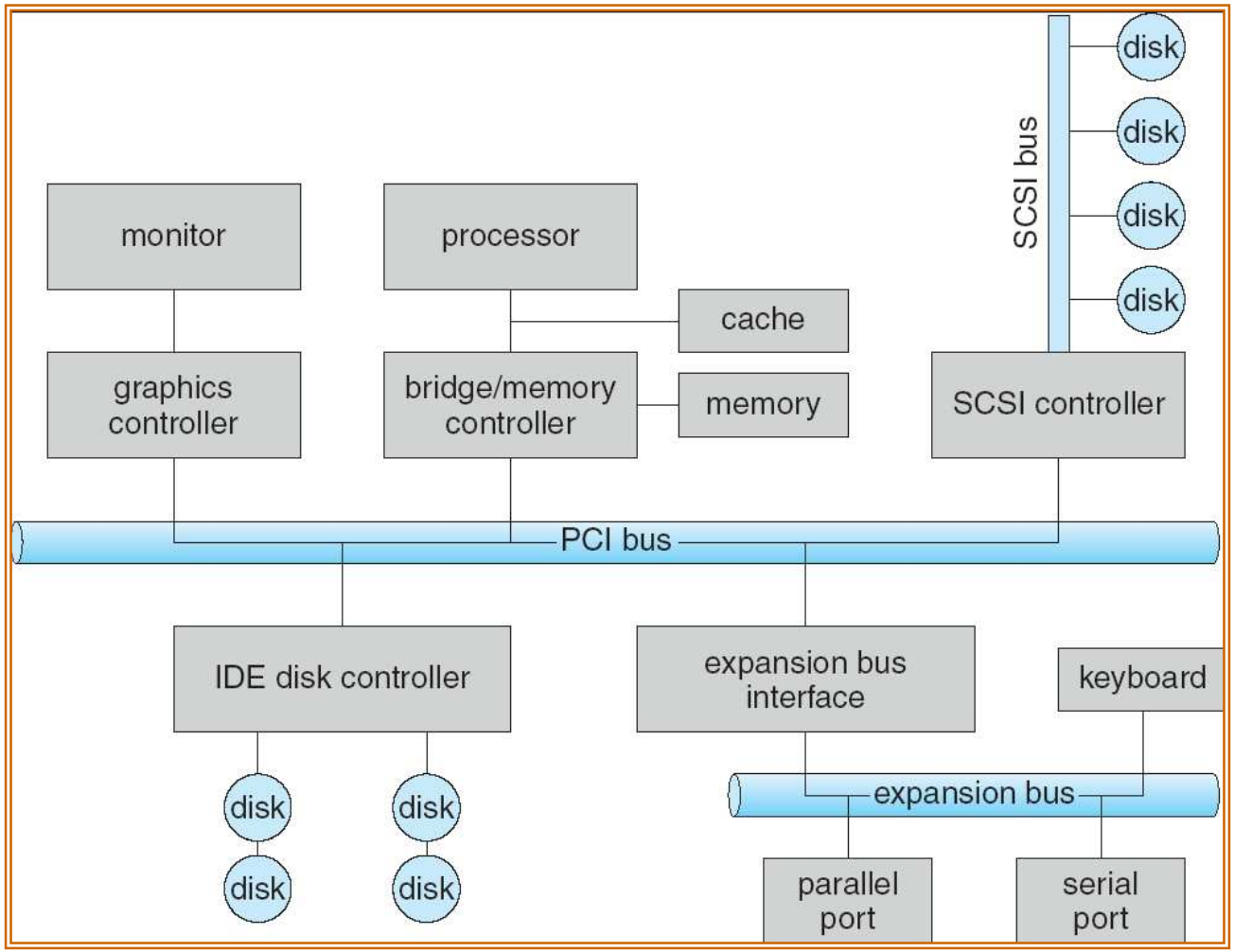
Principles of I/O Hardware
I/O Port

I/O Port is a register in device interface. Example: Device I/O Port Locations on PCs (partial)

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

Principles of I/O Hardware

A Typical PC Bus Structure

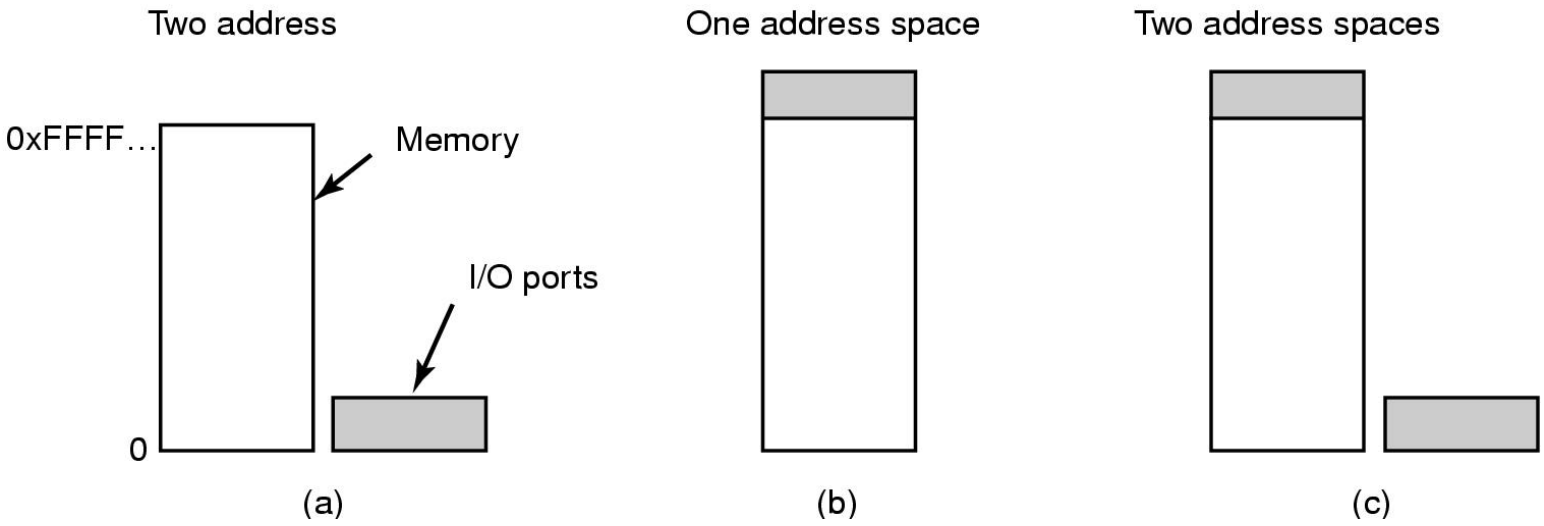


Principles of I/O Hardware

I/O address

- I/O instructions control devices
- Devices have addresses, used by
  - Direct I/O instructions
  - Memory-mapped I/O

Principles of I/O Hardware



- (a) Separate I/O and memory space
- (b) Memory-mapped I/O
- (c) Hybrid

Principles of I/O Hardware
Separating I/O and memory space

- Device drivers must be written using assembly language
- Programs must use 2 instructions or more to test whether the device is ready
- There is special protection mechanism to keep user processes from performing I/O

Principles of I/O Hardware
Memory-mapped I/O (1)

- Since the control registers of devices are mapped into the memory space, device drivers can be written in C
- Programs can use 1 instructions to test whether the device is ready
- Device control register may be memory pages
- But Caching device control register would be disastrous

Principles of I/O Hardware

Data transfer Method between CPU and I/O device

Three Data I/O transfer Methods:

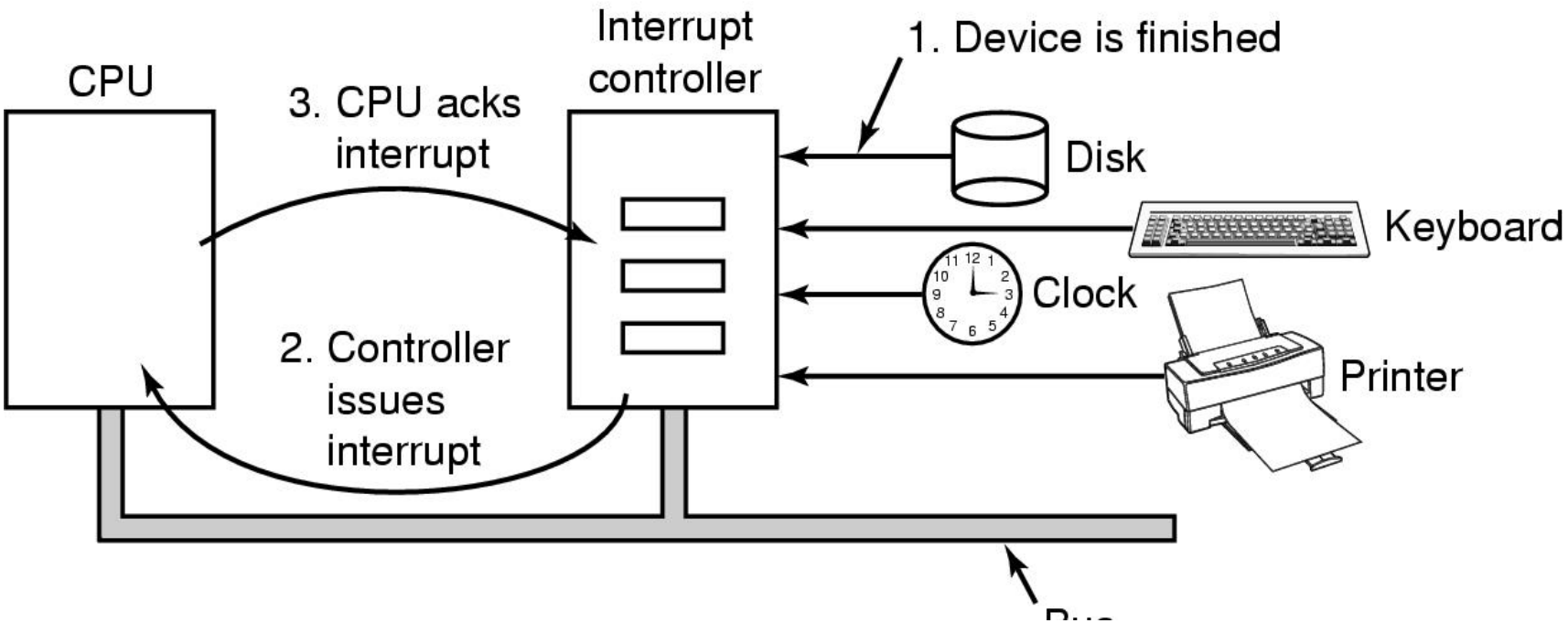
- Programmed I/O
- Interrupt-Driven I/O
- Direct Memory Access

Principles of I/O Hardware

Programmed I/O, Polling

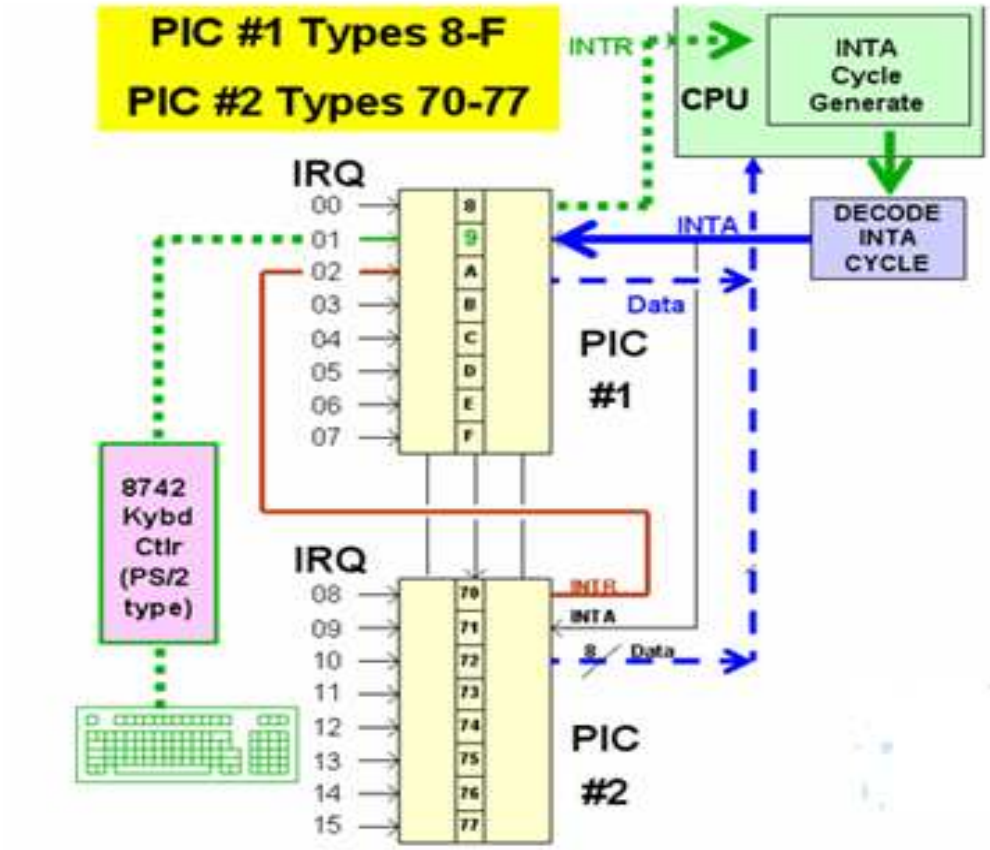
- Determines state of device
  - ready
  - busy
  - Error
- **Busy-wait** cycle to wait for I/O from device

Principles of I/O Hardware
Interrupt-Driven I/O



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

Principles of I/O Hardware
Example: Interrupts of PC computer



INT. NO	Interrupt Source
NMI	Parity Error Detected
IRQ0	Interval Timer (PIT)
IRQ1	Keyboard Output Buffer
IRQ2	PIC 2 (Cascade)
IRQ8	Real-time Clock Int
IRQ9	S/W Redirected To Int 0ah (IRQ2)--free
IRQ10	Free
IRQ11	Free
IRQ12	Free (PS/2 Mouse)
IRQ13	reserved--math coprocessor
IRQ14	Pri EIDE Fixed Disk
IRQ15	Sec EIDE Fixed Disk
IRQ3	COM2
IRQ4	COM1 (Primary)
IRQ5	Free (Sound Card, LPT2)
IRQ6	Floppy Disk Controller
IRQ7	LPT1 (Primary) Or Free

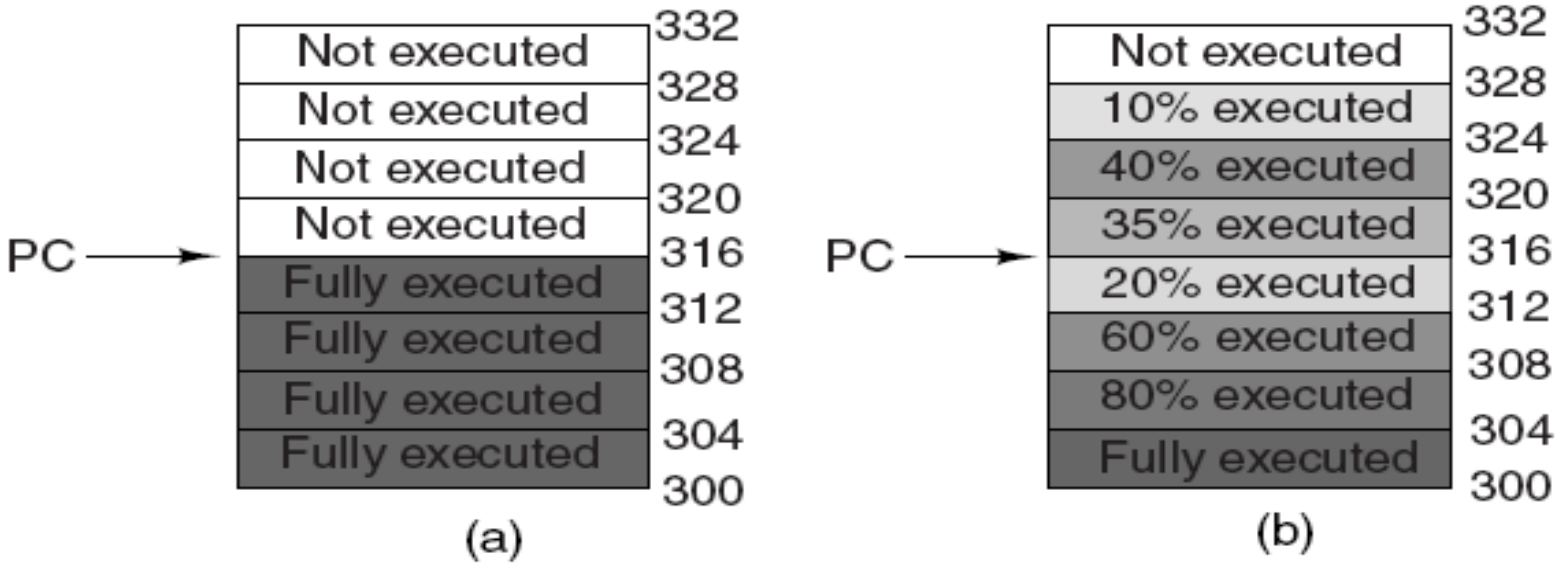
Principles of I/O Hardware
Precise and Imprecise Interrupts (1)

Properties of a *precise interrupt*

1. PC (Program Counter) is saved in a known place.
2. All instructions before the one pointed to by the PC have fully executed.
3. No instruction beyond the one pointed to by the PC has been executed.
4. Execution state of the instruction pointed to by the PC is known.

Principles of I/O Hardware

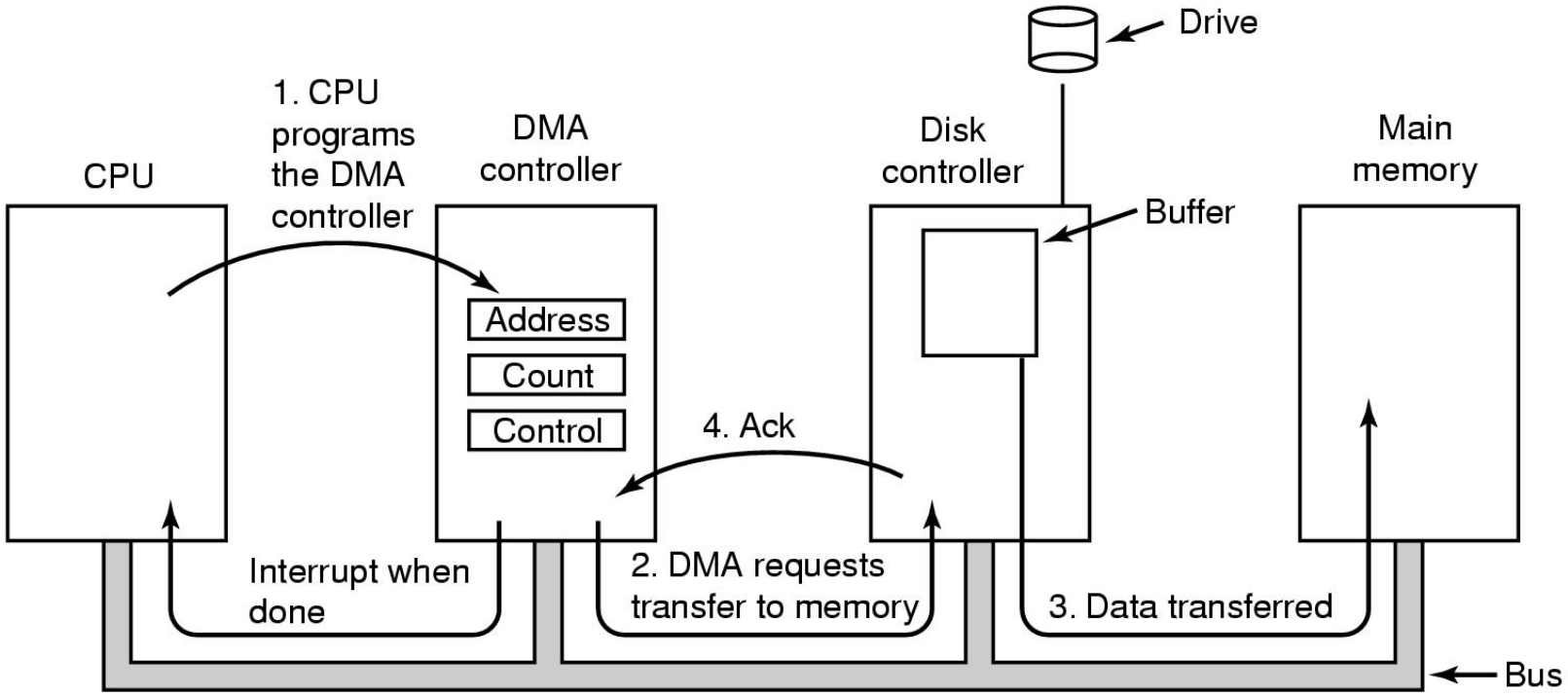
Precise and Imprecise Interrupts (2)



(a) A precise interrupt.

(b) An imprecise interrupt.

Principles of I/O Hardware
Direct Memory Access (DMA) (1)



Operation of a DMA transfer

Principles of I/O Hardware
Direct Memory Access (DMA) (2)

- DMA mechanism
  - In bock mode
  - Cycle Stealing
  - Fly-by mode
  - Third Party DMA

5.2 Principles of I/O software

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

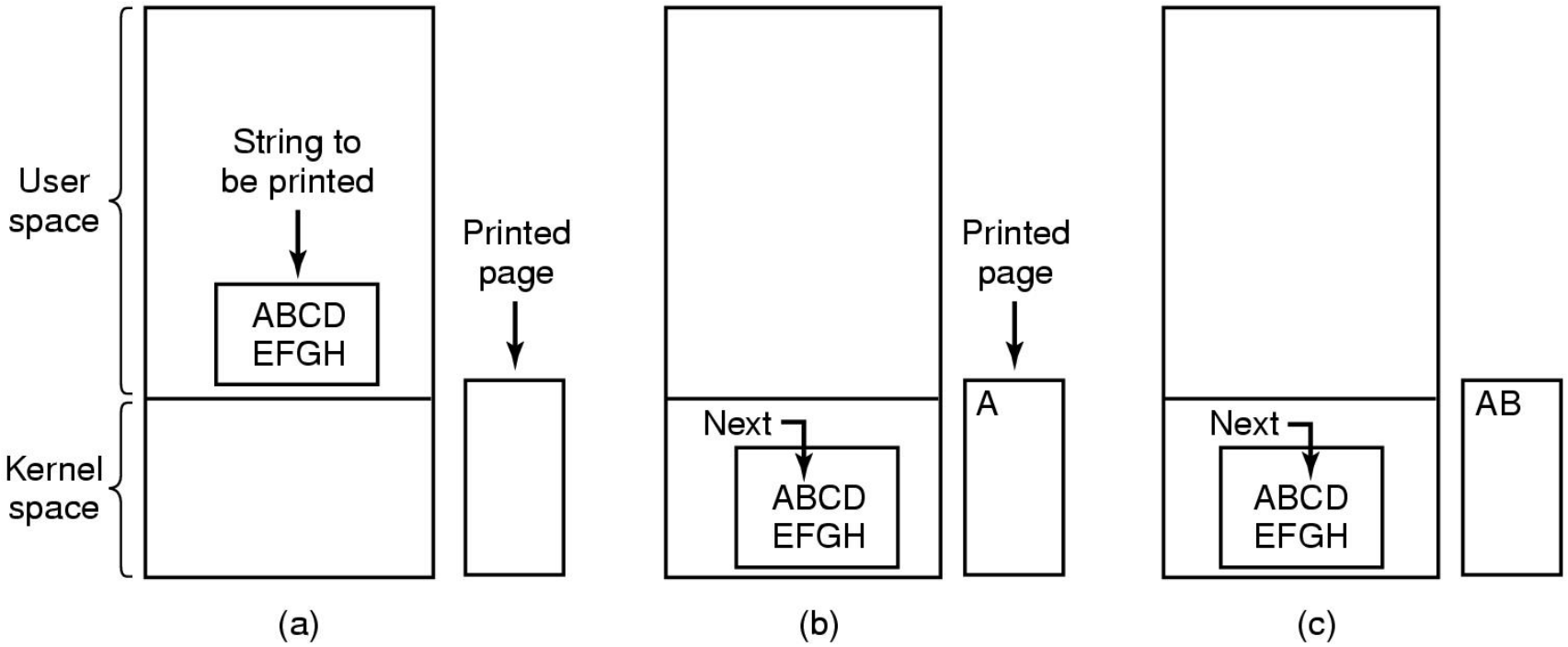
- Device independence
  - programs can access any I/O device
  - without specifying device in advance

    (floppy, hard drive, or CD-ROM)
- Uniform naming
  - name of a file or device is a string or an integer
  - not depending on which machine
- Error handling
  - handle as close to the hardware as possible

Principles of I/O Software
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

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



Steps in printing a string

Principles of I/O Software
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

Principles of I/O Software
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
  - (a) Code executed when print system call is made
  - (b) Interrupt service procedure

Principles of I/O Software
I/O Using DMA

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

(a)

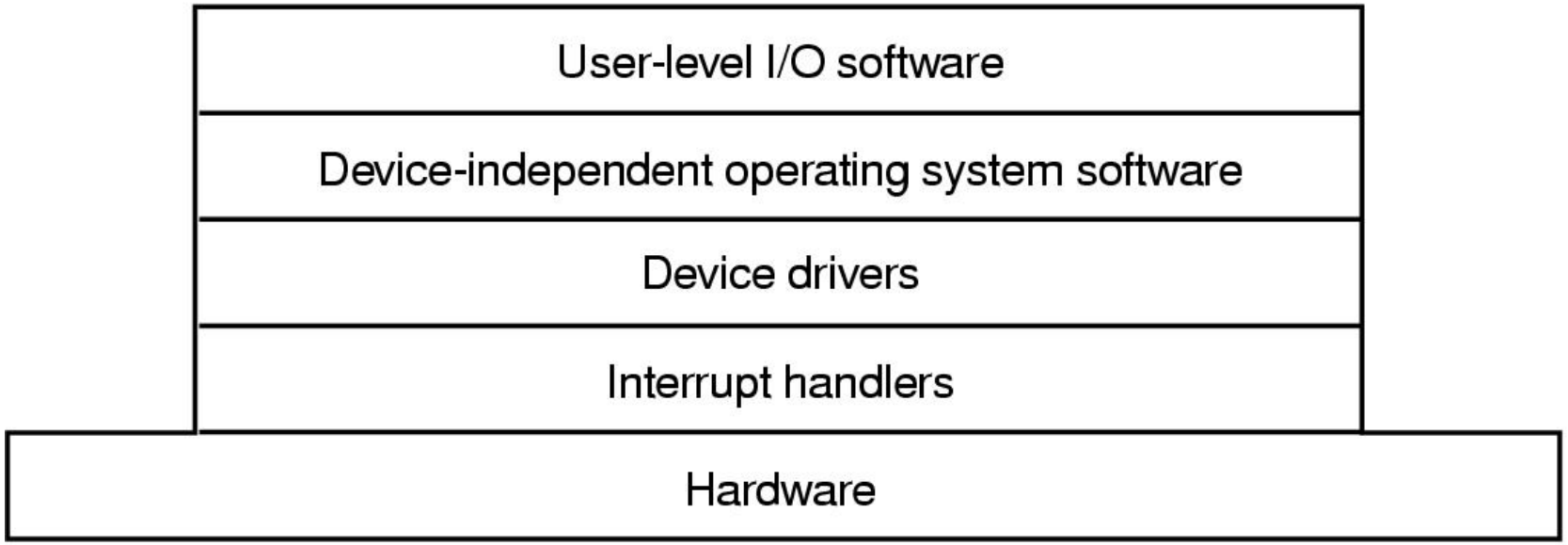
```
acknowledge_interrupt( );
unblock_user( );
return_from_interrupt( );
```

(b)

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

5.3 I/O software layers

I/O Software Layers



Layers of the I/O Software System

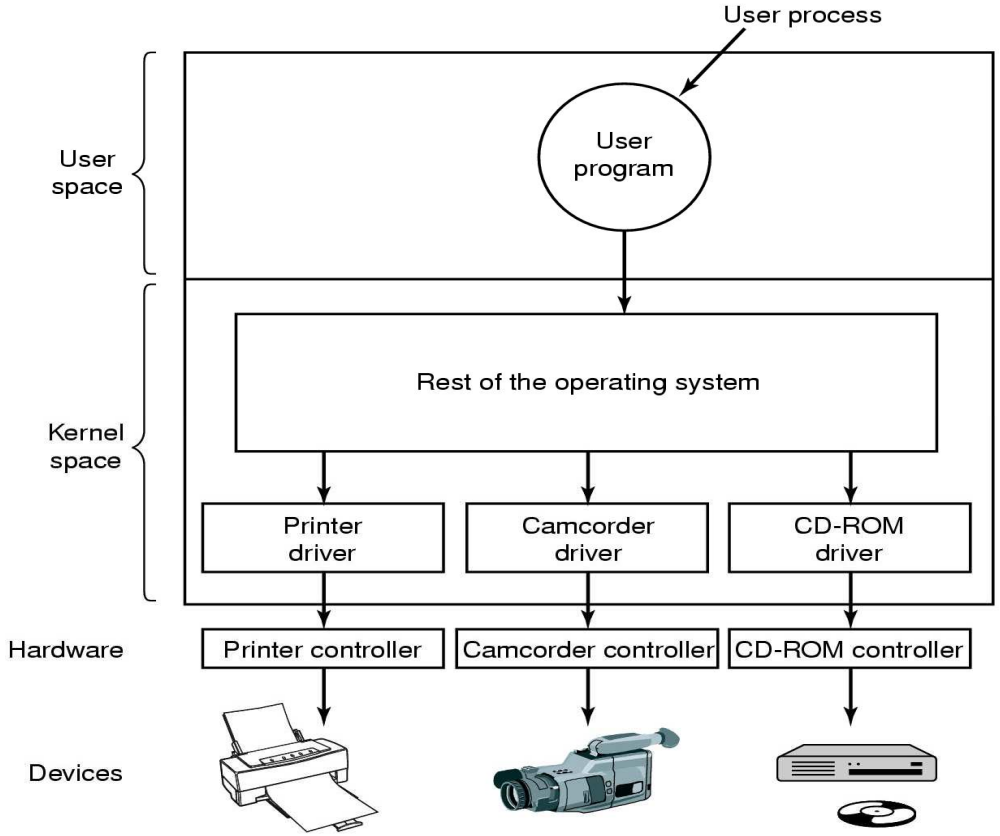
I/O Software Layers
Interrupt Handlers (1)

- Interrupt handlers are best hidden
  - have driver starting an I/O operation block 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 regs not already saved by interrupt hardware
  2. Set up context for interrupt service procedure

I/O Software Layers
Interrupt Handlers (2)

3. Set up stack for interrupt service procedure
4. Ack interrupt controller, reenable interrupts
5. Copy registers from where saved to process table
6. Run service procedure
7. Set up MMU context for process to run next
8. Load new process' registers
9. Start running the new process

I/O Software Layers
Device Drivers (1)



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

I/O Software Layers
Device Drivers (2)

- Component of general structure of device drivers
  - Checking the input parameters to see if they are valid
  - Checking if the device is currently in use
  - Writing command sequence into controller's device registers

I/O Software Layers
Device Drivers (3)

- Function of device drivers
  - To accept abstract read and write request from device independent software above it and see that they are curried out
  - To initialize the device, if needed
  - To manage its power requirements and log events

I/O Software Layers

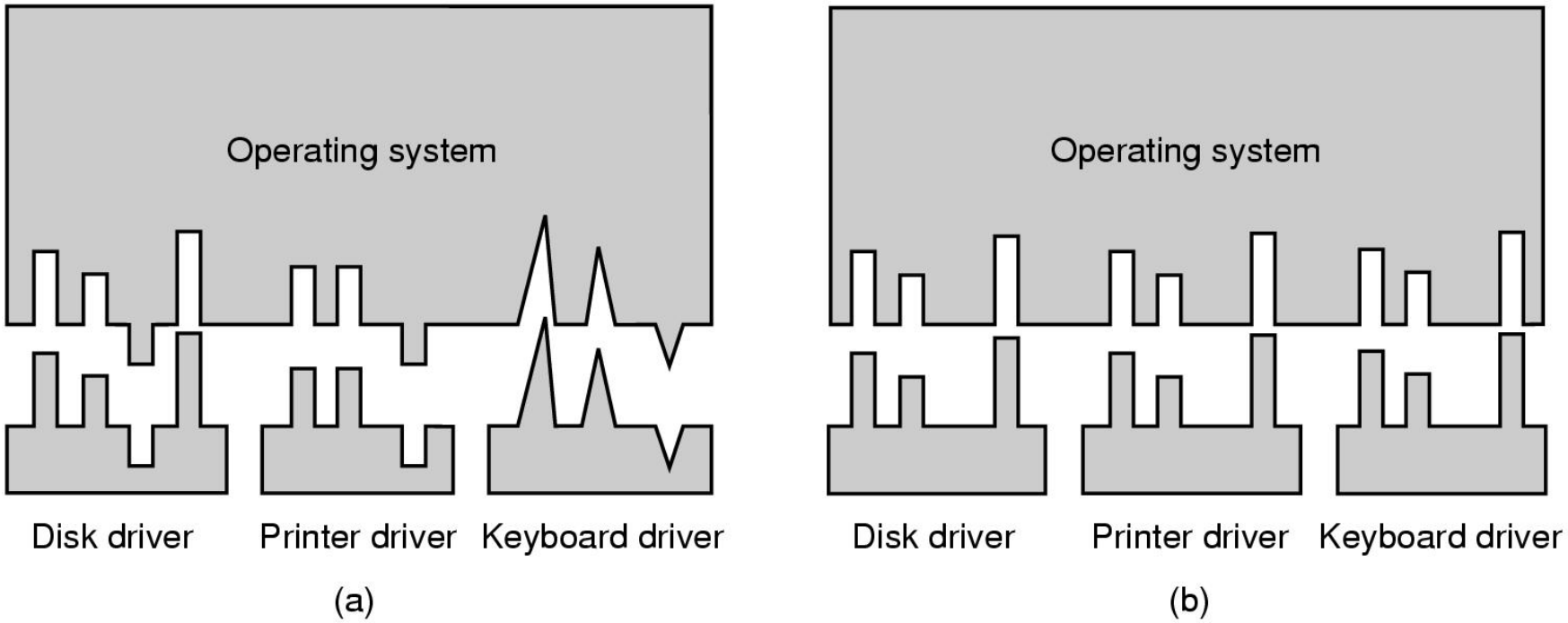
Device-Independent I/O Software (1)

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

Functions of the device-independent I/O software

I/O Software Layers

Device-Independent I/O Software (2)

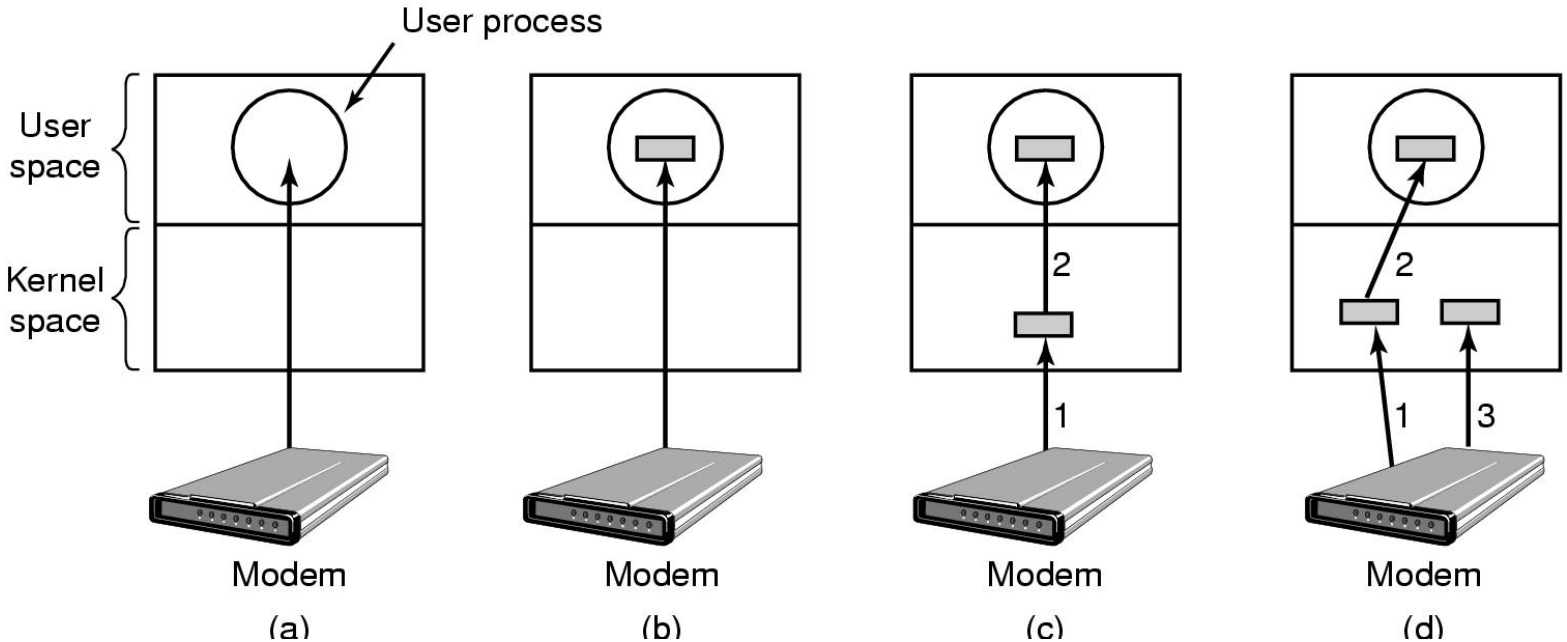


(a) Without a standard driver interface

(b) With a standard driver interface

I/O Software Layers

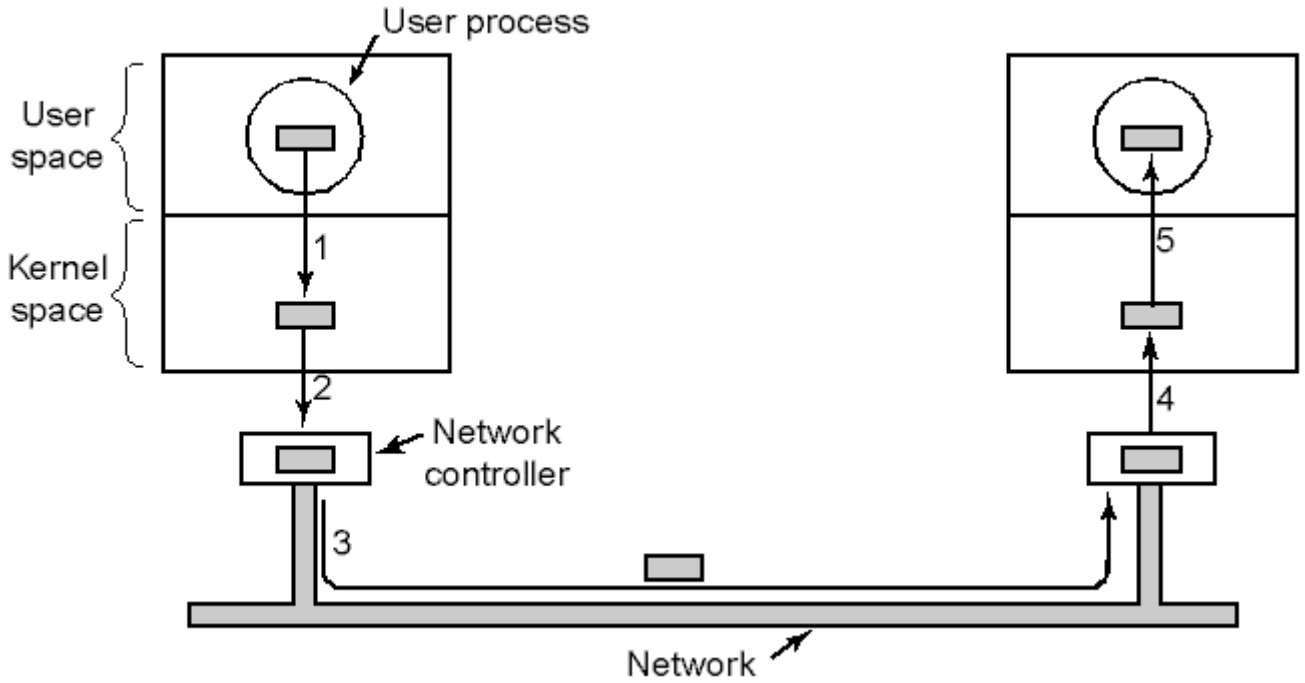
Device-Independent I/O Software (3)



(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

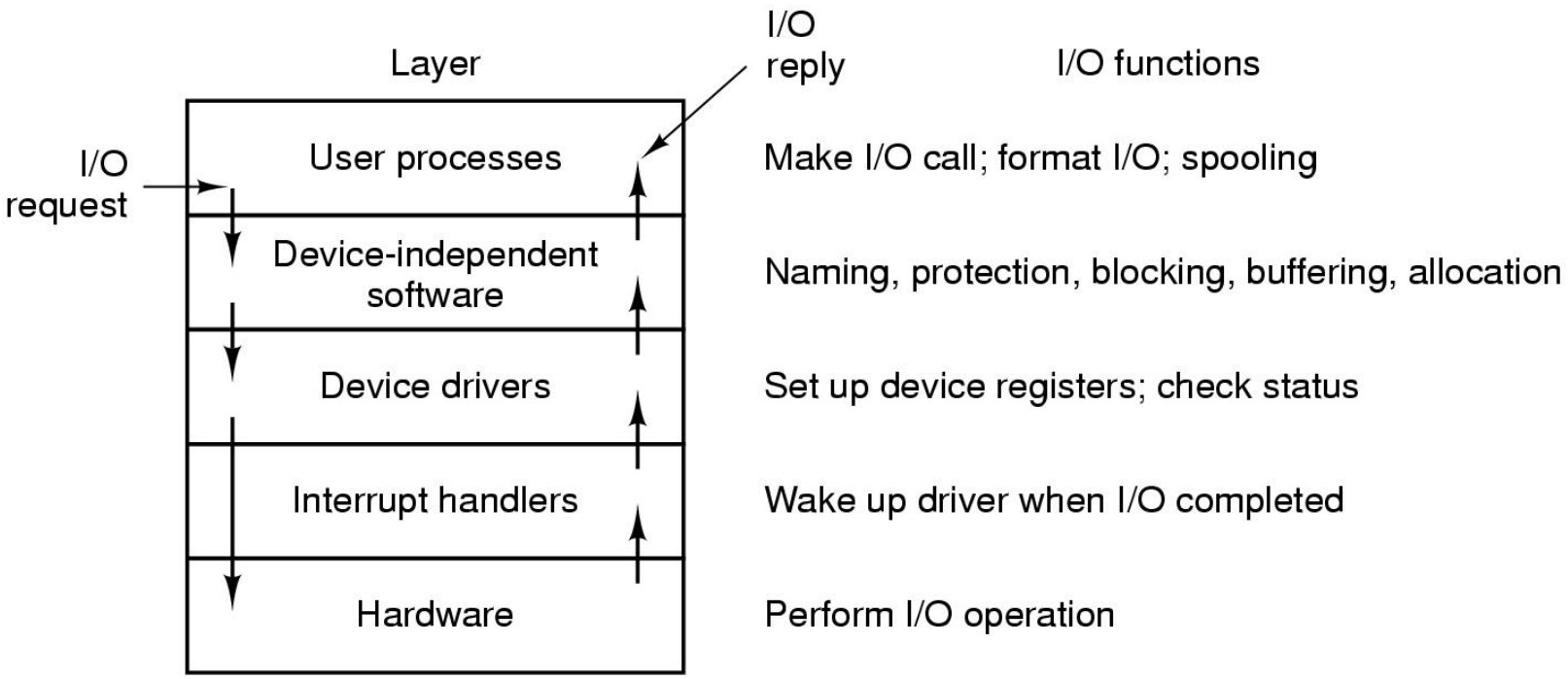
I/O Software Layers

Device-Independent I/O Software (4)



Networking may involve many copies

User-Space I/O Software



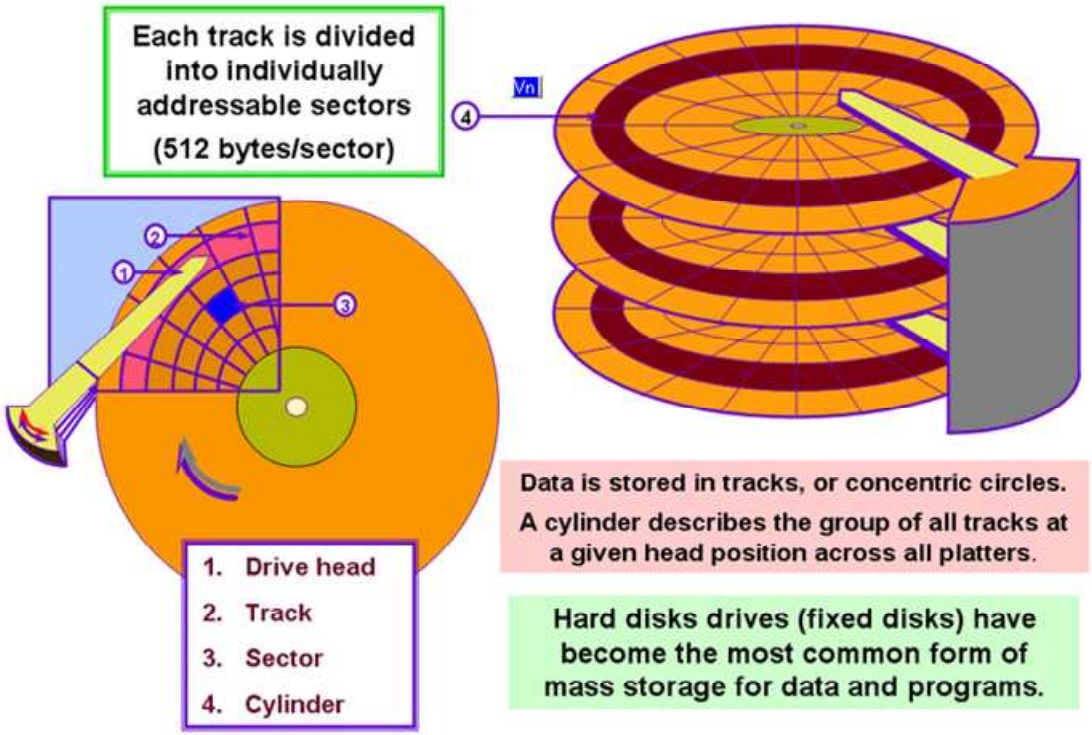
Layers of the I/O system and the main functions of each layer

I/O Software Layers
Examples of a function for each layer

- User-level software
  - Example: Converting binary integers to ASCII for printing
- Device-Independent I/O Software
  - Example: Checking to see if the user is permitted to use the device
- Device Drivers
  - Example: Writing commands to the device registers
- Interrupt Handlers,
  - Example: Indicating that new character can be accepted to print

5.4 Disks

Disk Hardware

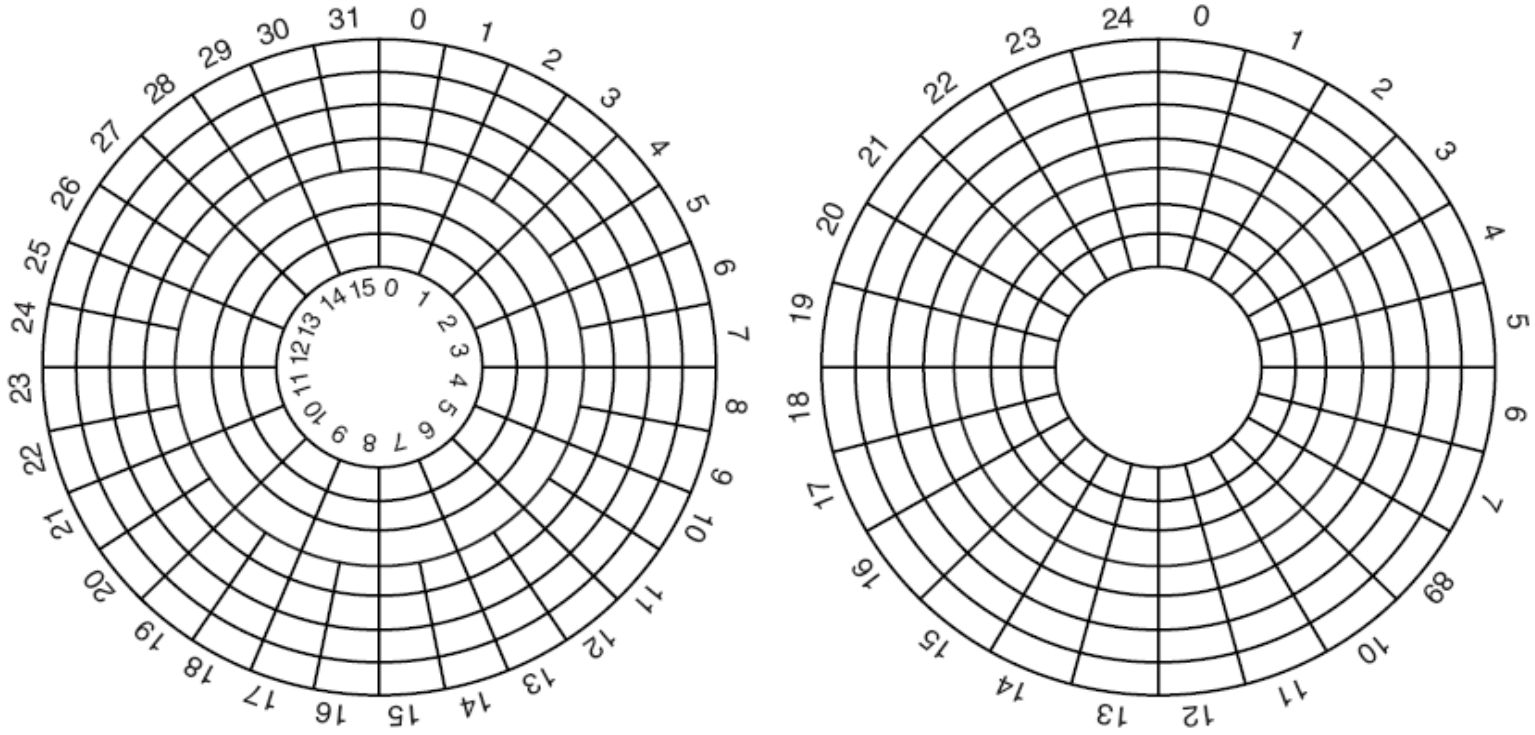


Magnetic Disks (1)

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 360-KB floppy disk and a Western Digital WD 18300 hard disk.

Magnetic Disks (2)

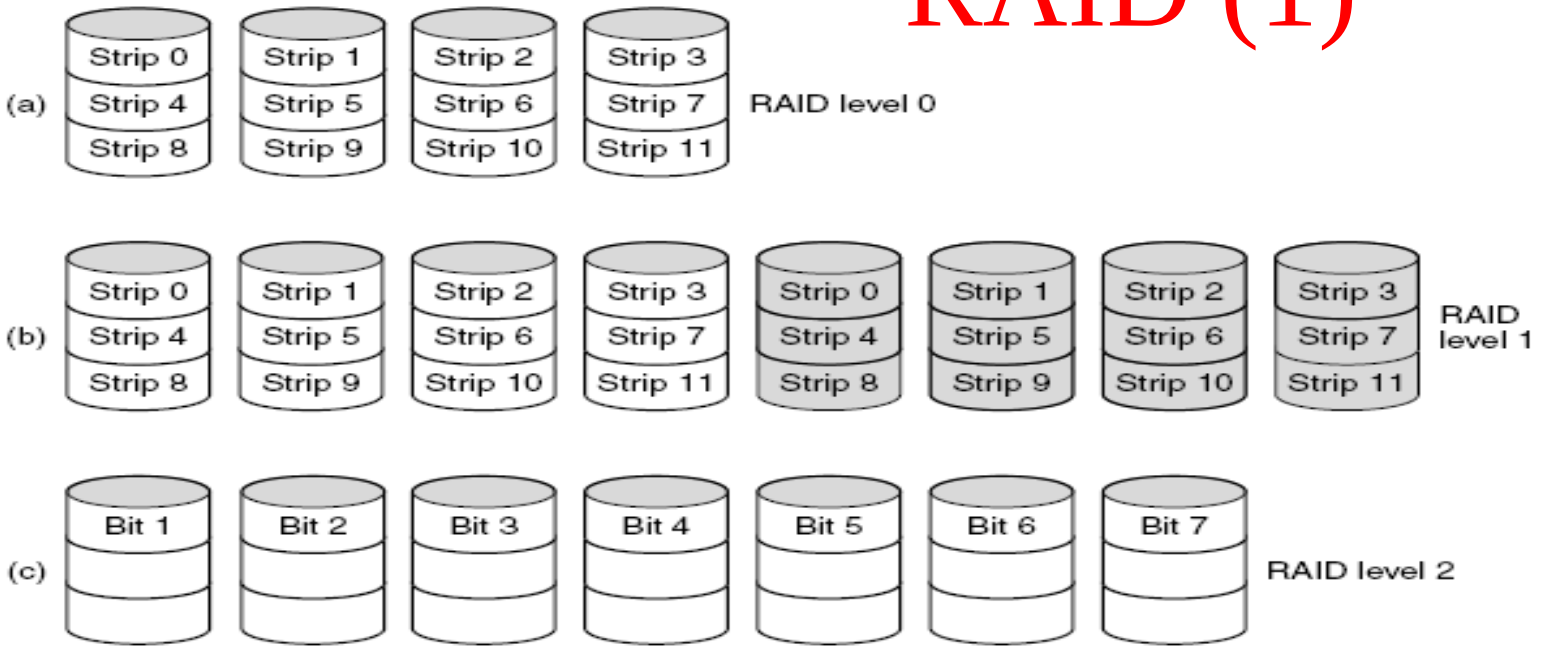


(a) Physical geometry of a disk with two zones.

(b) A possible virtual geometry for this disk.

- Logical block addressing: sectors are numbered consecuitively starting at 0

RAID (1)



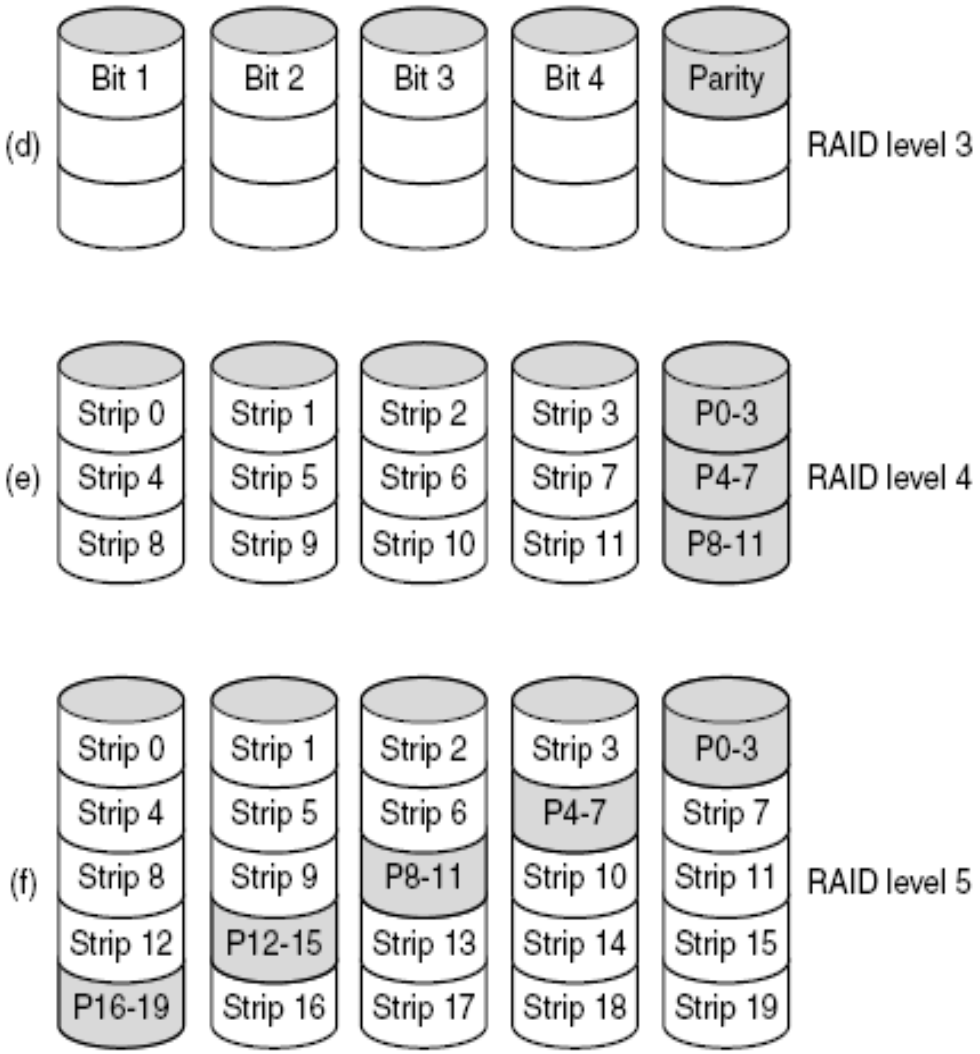
RAID levels 0 distributes data over multiple drives, is no reliability support.

RAID level 1 duplicates all the disks.

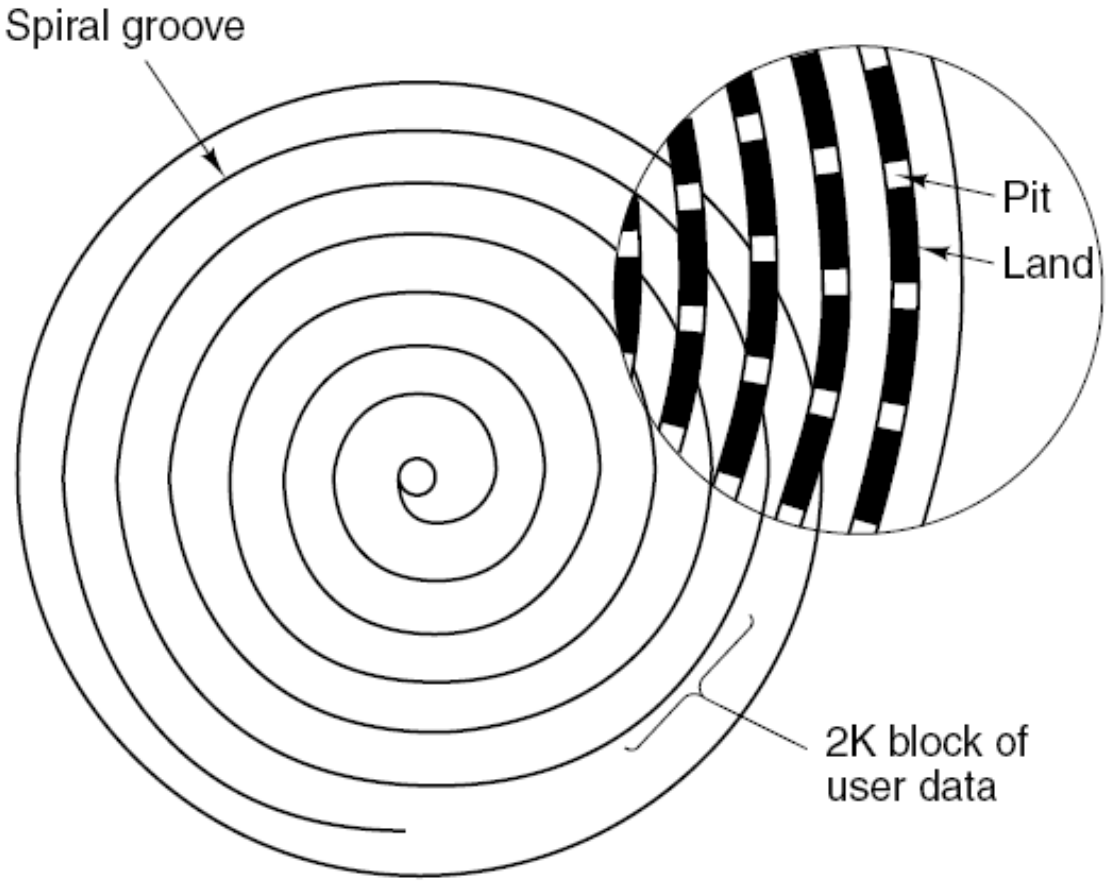
RAID level 2 employs a Hamming code to detect and correct error

RAID (2)

- The RAID level 3 uses word parity for error detection.
- The RAID level 4 uses strip-for-strip parity for error detection
- The RAID level 4 distributes parity strips over all drives

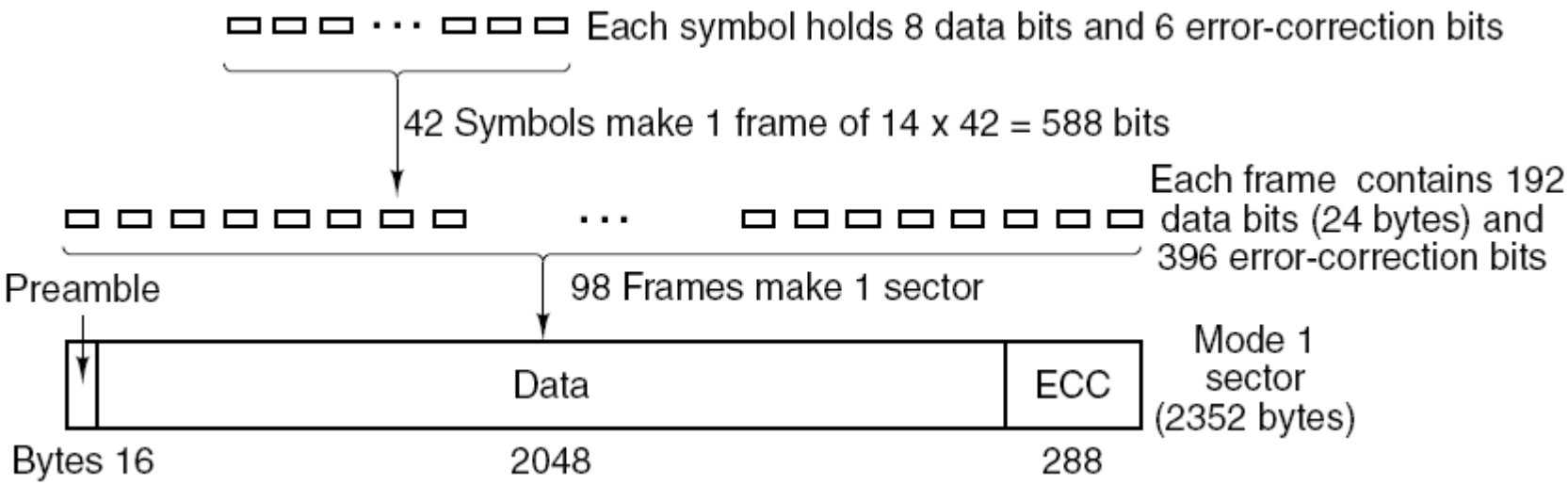


CD-ROMs (1)



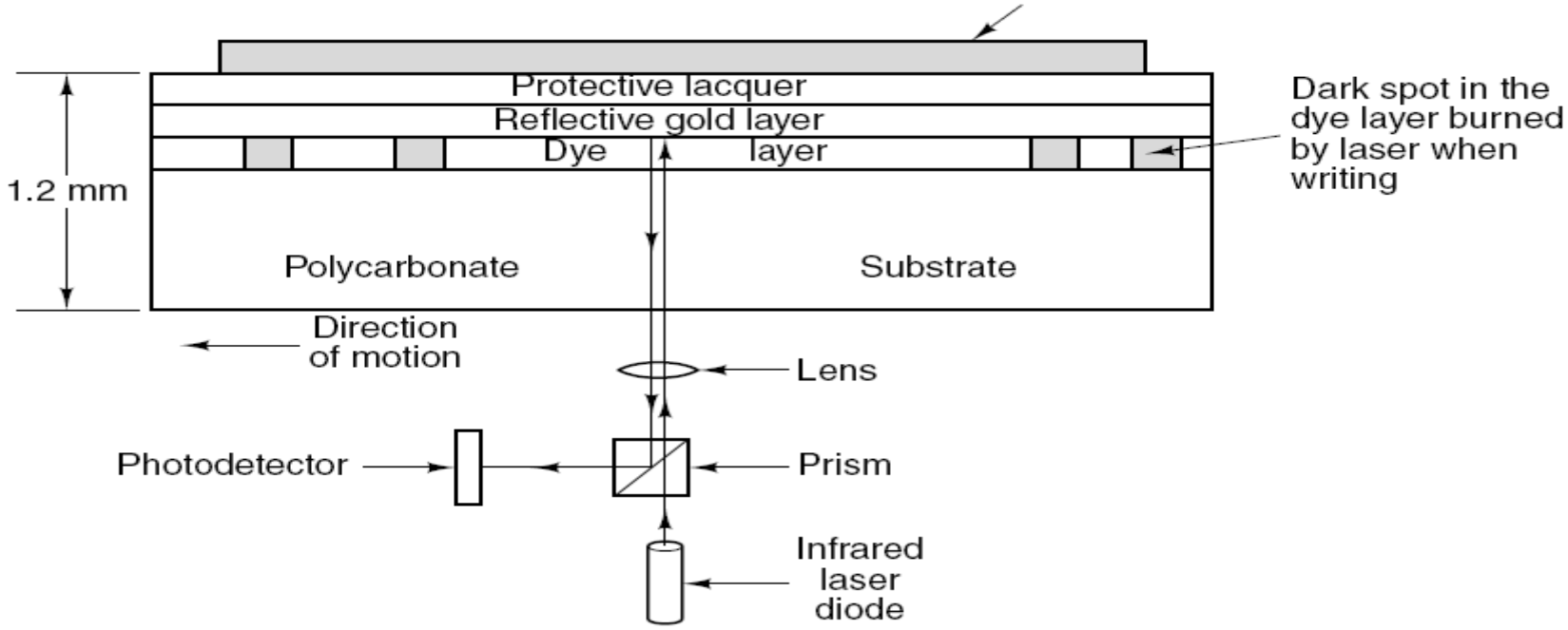
Recording structure of a compact disc or CD-ROM.

CD-ROMs (2)



Logical data layout on a CD-ROM.

CD-Recordables (1)



- Cross section of a CD-R disk and laser.
- A silver CD-ROM has similar structure, except without dye layer and with pitted aluminum layer instead of gold layer.
- Group of consecutive sectors written at once called CD-ROM track
- Tracks can be grouped into Section leading to Multisection CD-ROM

DVD (1)

DVD Improvements on CDs

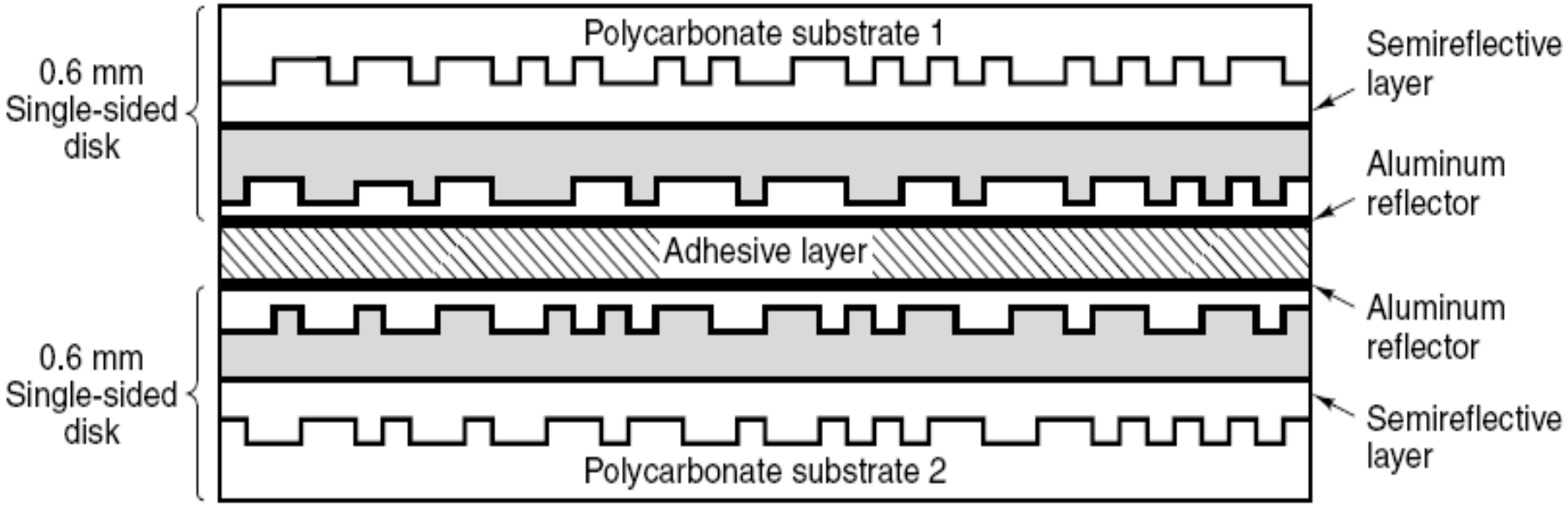
1. Smaller pits
   (0.4 microns versus 0.8 microns for CDs).
2. A tighter spiral
   (0.74 microns between tracks versus 1.6 microns for CDs).
3. A red laser
   (at 0.65 microns versus 0.78 microns for CDs).

DVD (2)

DVD Formats

1. Single-sided, single-layer (4.7 GB).
2. Single-sided, dual-layer (8.5 GB).
3. Double-sided, single-layer (9.4 GB).
4. Double-sided, dual-layer (17 GB).

DVD (3)

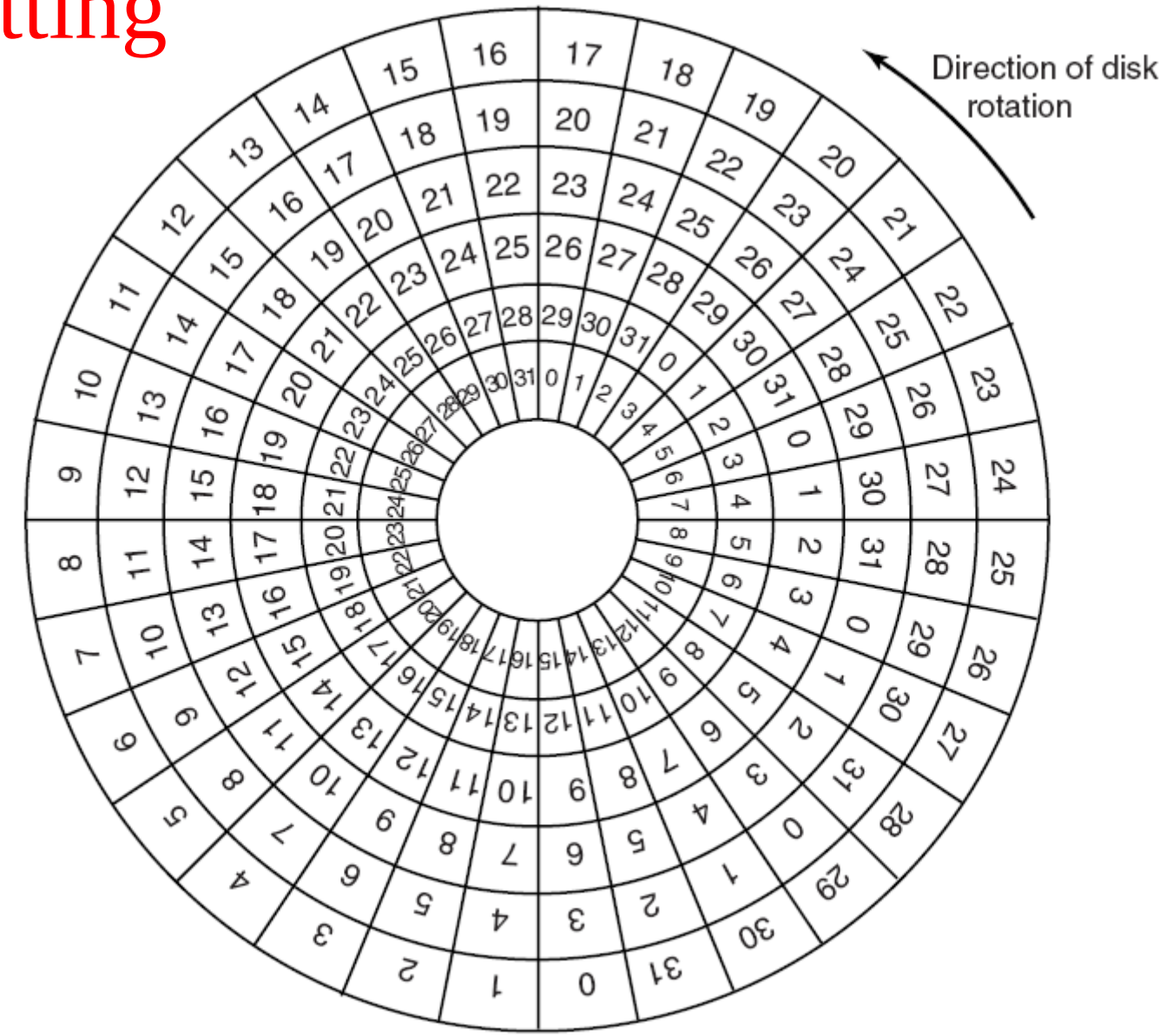


A double-sided, dual-layer DVD disk.

Disk Formatting (1)

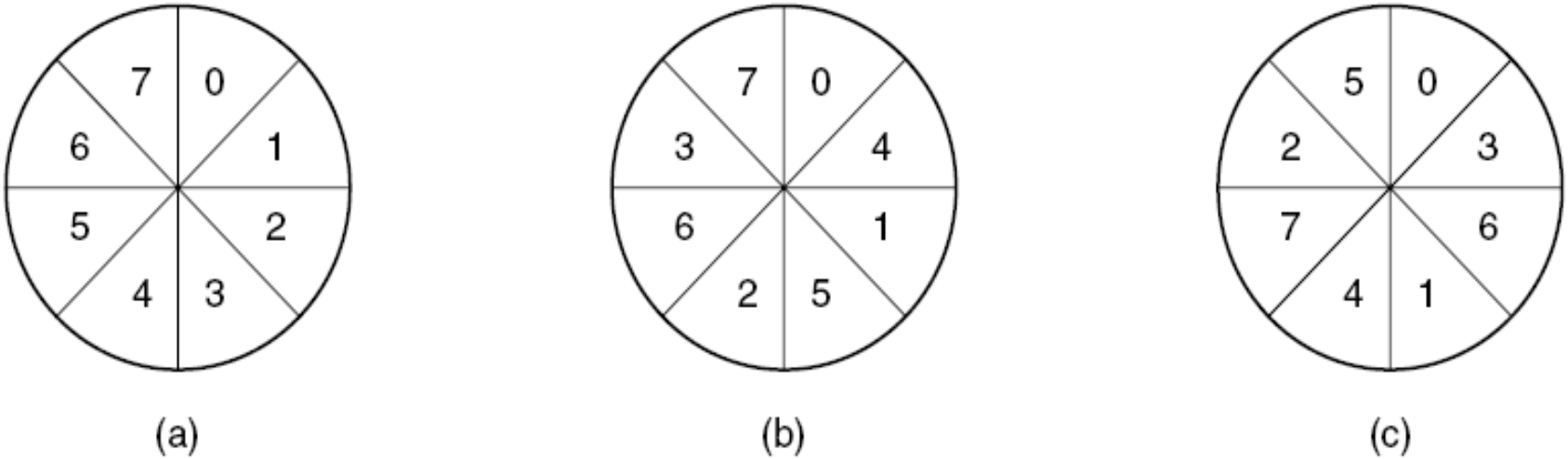
A disk sector includes three fields: Preamble, Data and ECC.

Disk Formatting (2)



An illustration of cylinder skew.

Disk Formatting (3)



(a) No interleaving.

(b) Single interleaving.

(c) Double interleaving.

High- level format of disk lays down

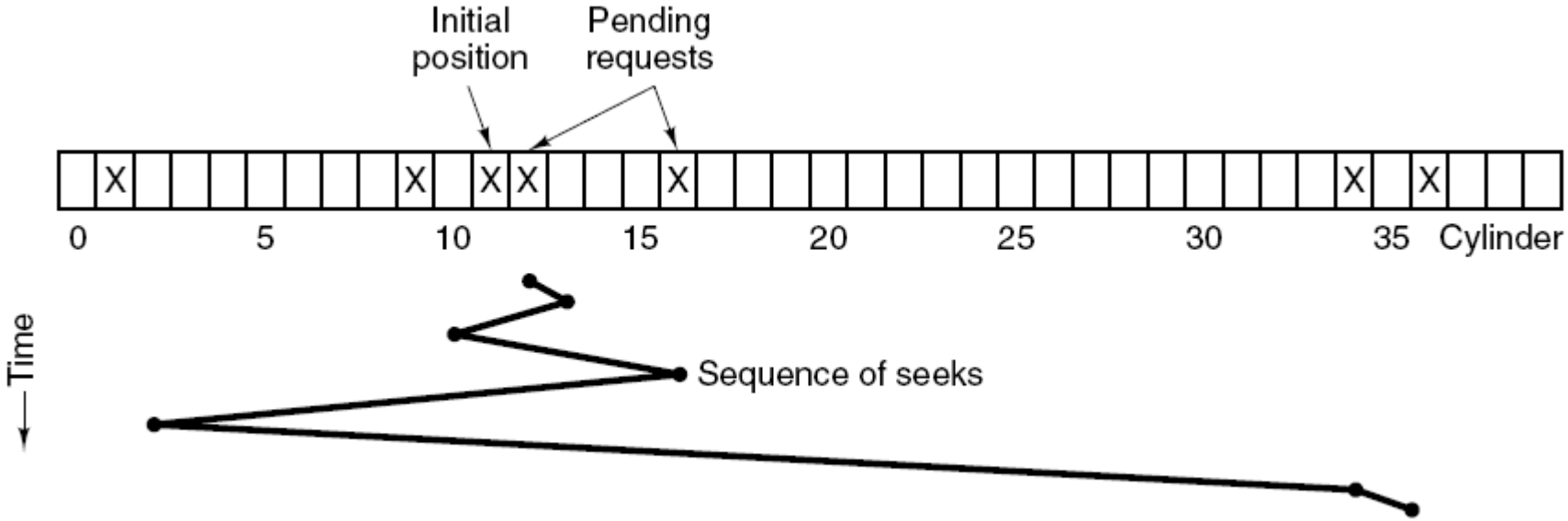
- A boot block
- The free storage administration
- Root directory, and an empty file system

Disk Arm Scheduling Algorithms (1)

Read/write time factors

1. Seek time (the time to move the arm to the proper cylinder).
2. Rotational delay (the time for the proper sector to rotate under the head).
3. Actual data transfer time.

Disk Arm Scheduling Algorithms (2)

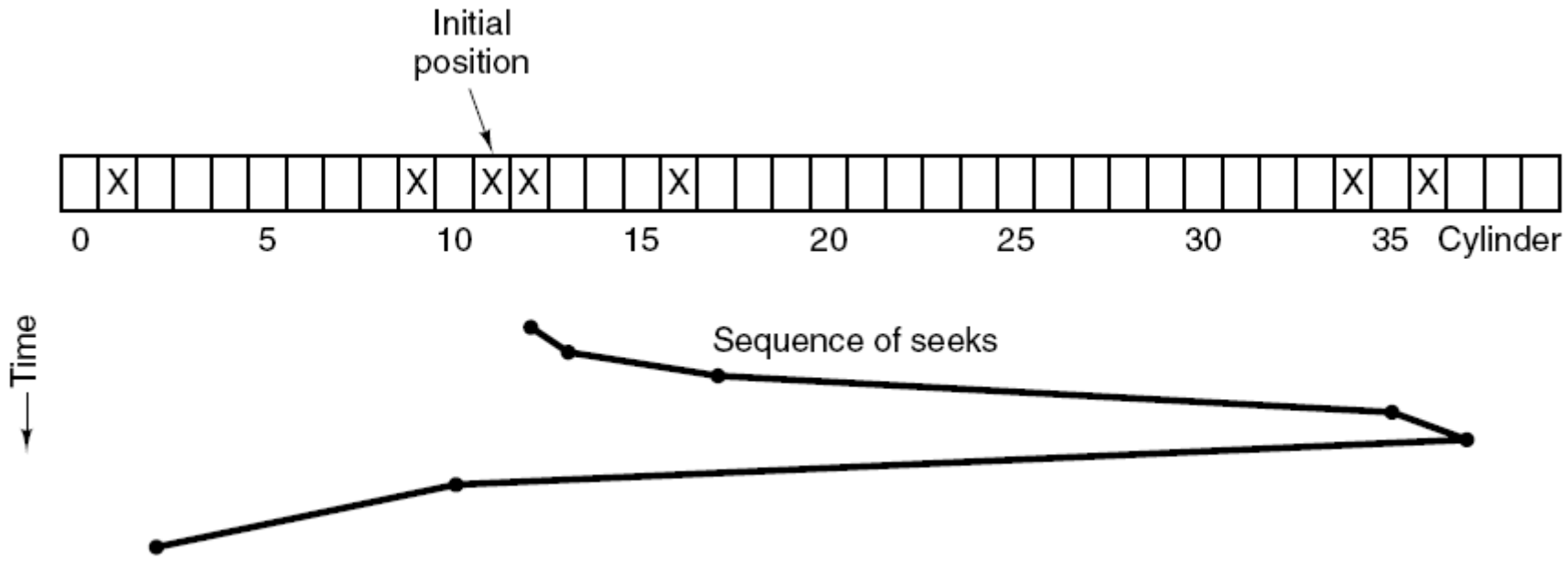


There are four Disk Arm Scheduling Algorithms

1. First-Come, First Served
2. Shortest Seek First (SSF) (Above Figure)

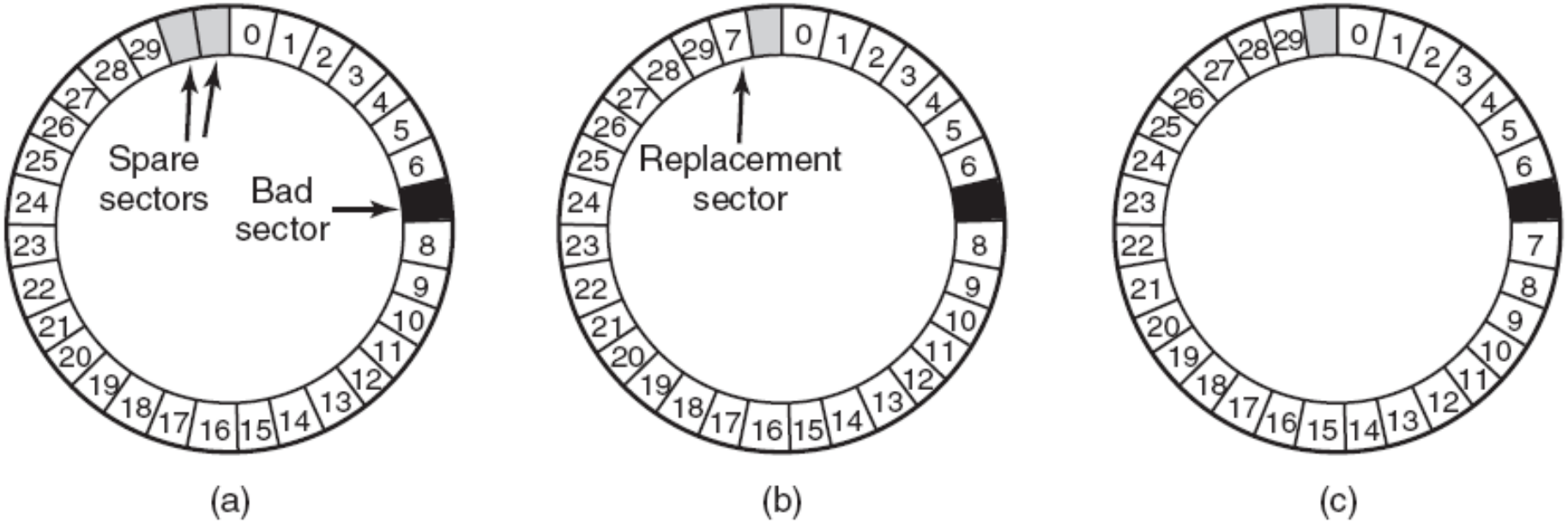
.

Disk Arm Scheduling Algorithms (3)



3. Elevator algorithm for scheduling disk requests. (Above Figure)
4. Modified elevator algorithm

Error Handling



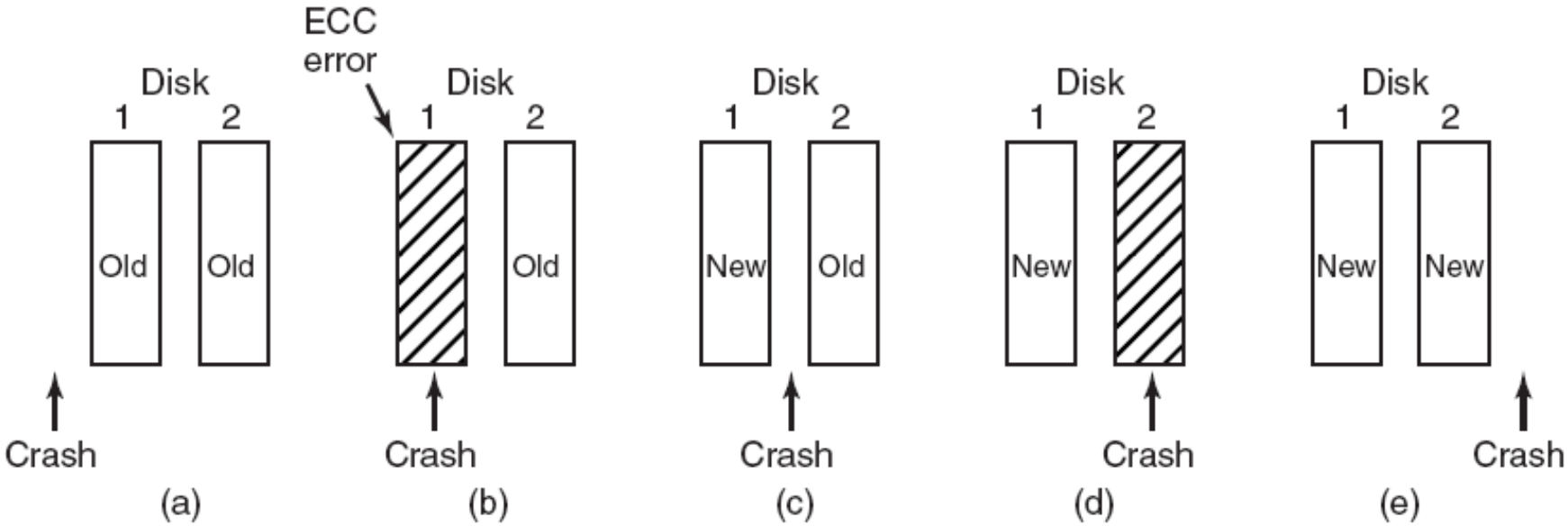
(a) A disk track with a bad sector.
(b) Substituting a spare for the bad sector.
(c) Shifting all the sectors to bypass the bad one.

Stable Storage (1)

Operations for stable storage using identical disks:

1. Stable writes
2. Stable reads
3. Crash recovery

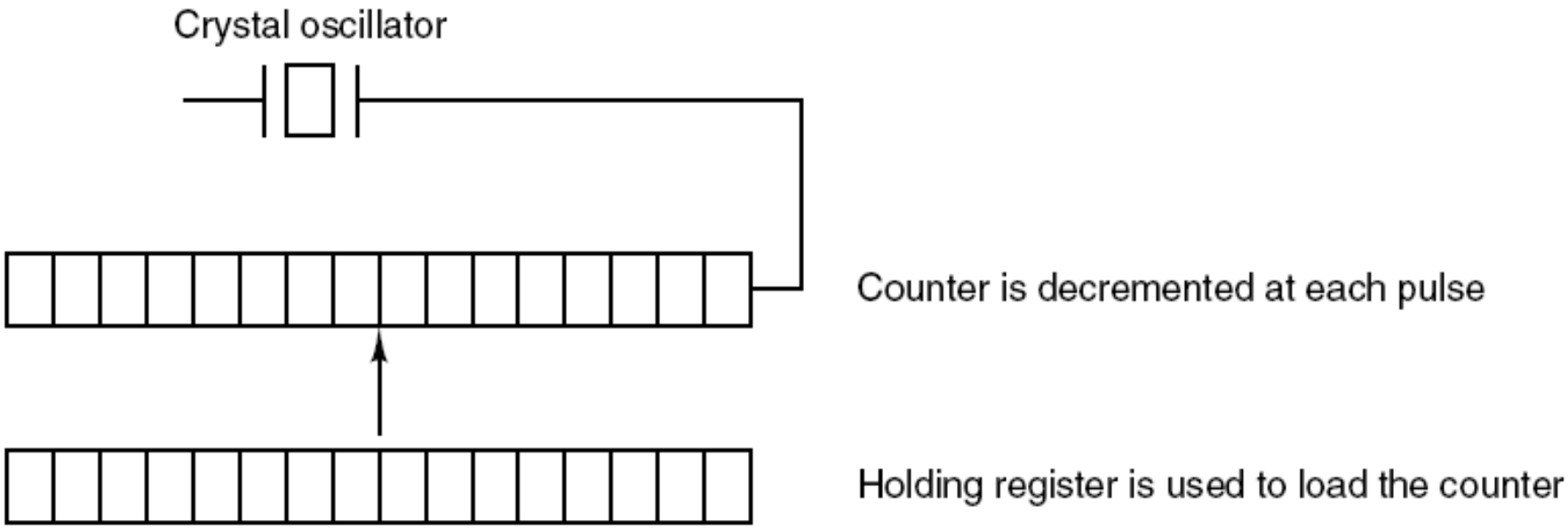
Stable Storage (2)



Analysis of the influence of crashes on stable writes.

5.5 Clocks

Clock Hardware



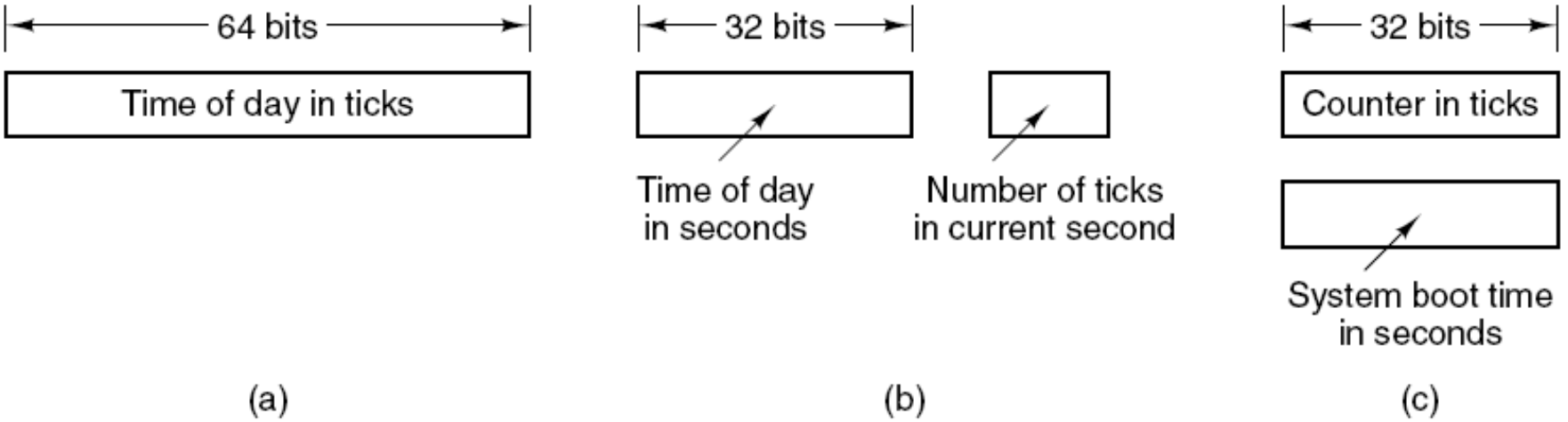
A programmable clock.

Clock Software (1)

Typical duties of a clock driver

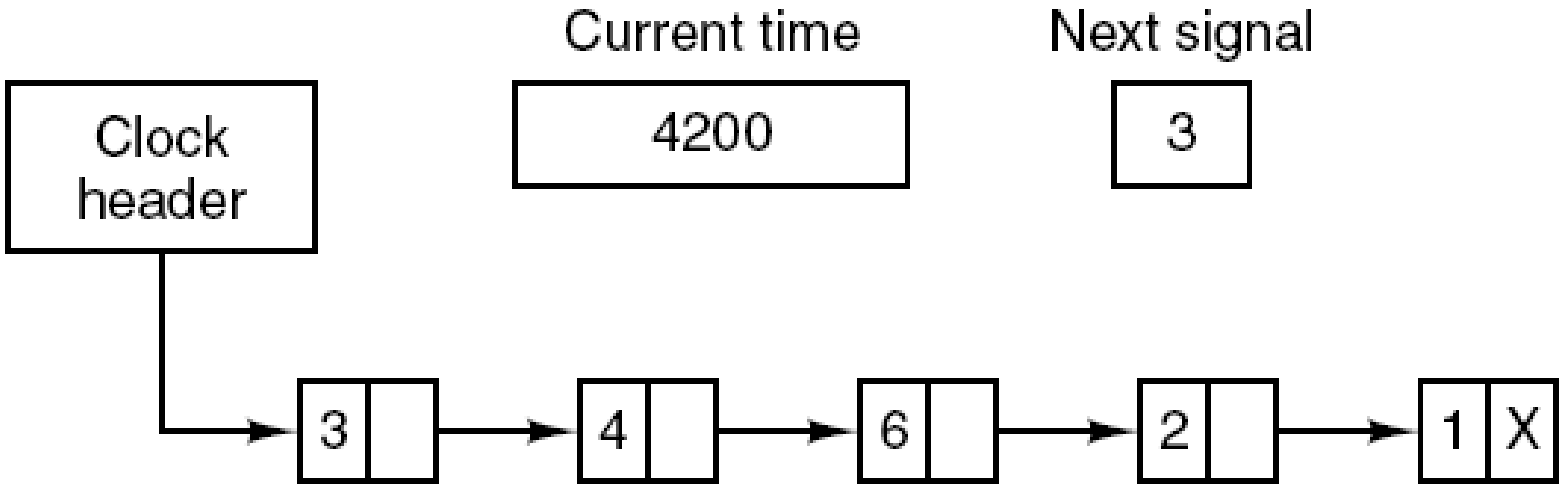
1. Maintaining the time of day.
2. Preventing processes from running longer than they are allowed to.
3. Accounting for CPU usage.
4. Handling alarm system call made by user processes.
5. Providing watchdog timers for parts of the system itself.
6. Doing profiling, monitoring, statistics gathering.

Clock Software (2)



Three ways to maintain the time of day.

Clock Software (3)



Simulating multiple timers with a single clock.

Soft Timers

Soft timers succeed according to rate at which kernel entries are made because of:

1. System calls.
2. TLB misses.
3. Page faults.
4. I/O interrupts.
5. The CPU going idle.

5.6 User Intrefaces

Keyboard Software (1)
keyboard drivers

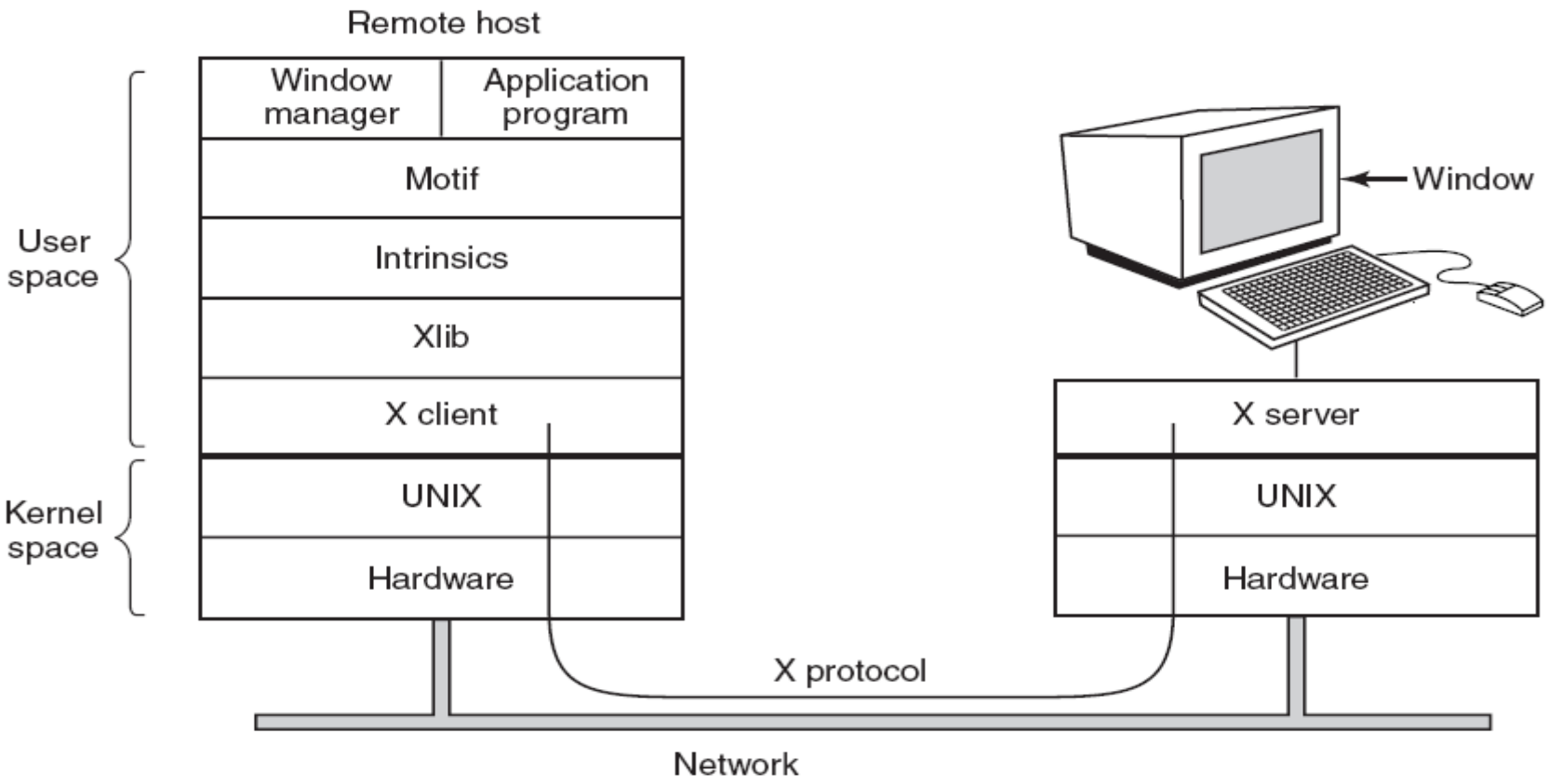
- To keep track of each key and to determinate whether pressed key is lower case or upper case
- In raw mode (noncanonical mode), keyboard drivers accept input key and forward to upper program
- In cooked mode (canonical mode), keyboard drivers handle all intraline and just delivers corrected lines to user program

Keyboard 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)

Characters that are handled specially in canonical mode.

The X Window System (1)



Clients and servers in the M.I.T. X Window System.

The X Window System (2)

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 1from 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

The ANSI escape sequences accepted by the terminal driver on output. ESC denotes the ASCII escape character (0x1B), and *n*, *m*, and *s* are optional numeric parameters.

The X Window System (3)

- Is high portable for nearly all UNIX and LINUX system
- Is highly event driven. Event flows from the workstation to the program
- A key concept in X window is the resource. A resource is a data structure that holds certain information created on the workstation, to be shared among multiple processes

The X Window System (4)

Types of messages between client and server:

1. Drawing commands from the program to the workstation.
2. Replies by the workstation to program queries.
3. Keyboard, mouse, and other event announcements.
4. Error messages.

Graphic adapter (1)

- Contains a special memory called *video RAM*
- Supports some number of screen sizes (resolution)
- Supports different method of coding pixel color

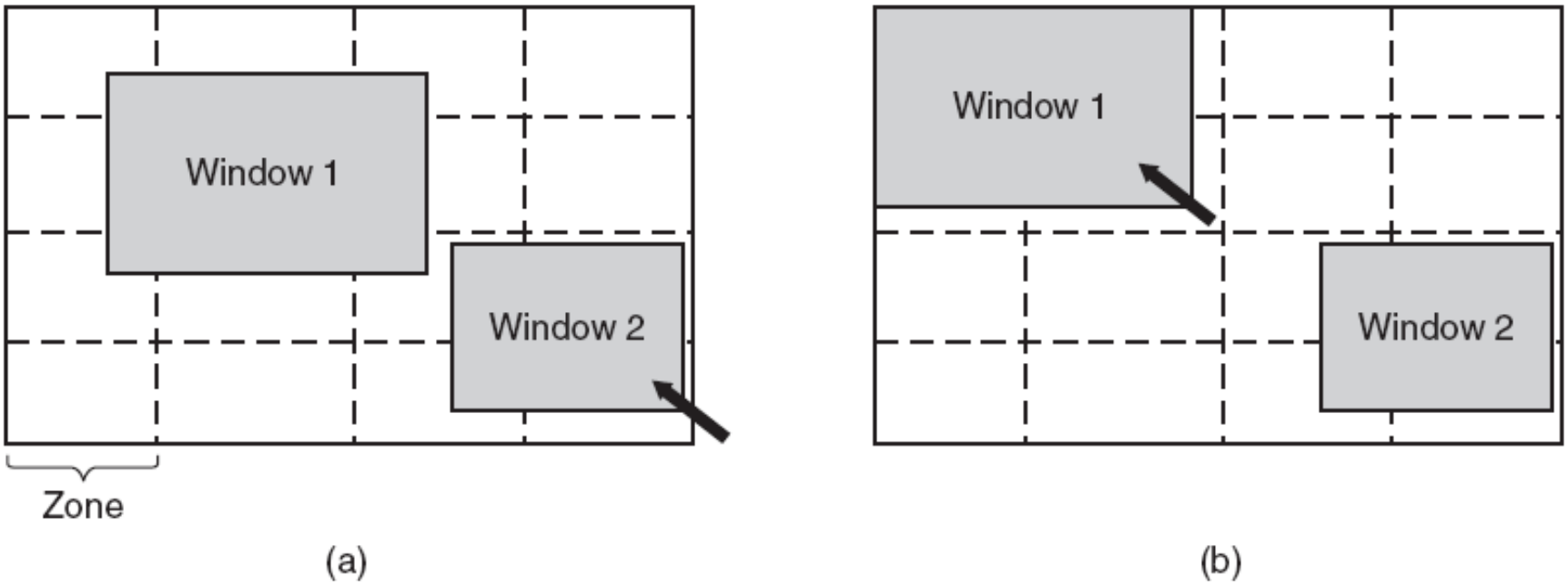
5.7 Power Management

Power Management Hardware Issues

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%

Power consumption of various parts of a notebook computer.

Power Management
The Display

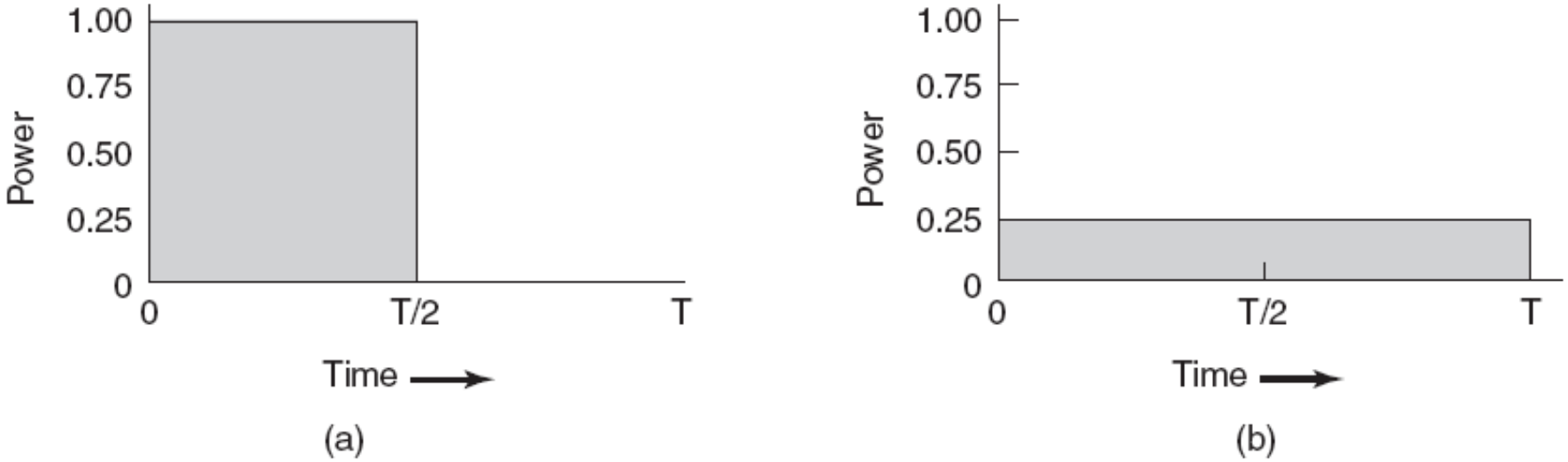


The use of zones for backlighting the display.

(a) When window 2 is selected it is not moved.

(b) When window 1 is selected, it moves to reduce the number of zones illuminated.

Power Management
The CPU



(a) Running at full clock speed.

(b) Cutting voltage by two cuts clock speed by two and power consumption by four.

Power Management

- The Hard Disk
- The Memory
- Wireless Communication