I/O Devices

- Types of I/O devices
  - Block devices
    - Stores information in fixed-size blocks, each one with its own address space
    - Common block size: 512 bytes to 32,768 bytes
    - E.g. disks
  - Character devices
    - Delivers or accepts a stream of characters, without regard to any block structure
    - Not addressable, no seek operation
    - E.g. printers, network interfaces, mice
  - Other devices
    - Clocks, memory-mapped screens

I/O Devices

- Some typical device, network, and bus data rates

Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Scanner at 300 dpi	1 MB/sec
Digital camcorder	3.5 MB/sec
4x Blu-ray disc	18 MB/sec
802.11n Wireless	37.5 MB/sec
USB 2.0	60 MB/sec
FireWire 800	100 MB/sec
Gigabit Ethernet	125 MB/sec
SATA 3 disk drive	600 MB/sec
USB 3.0	625 MB/sec
SCSI Ultra 5 bus	640 MB/sec
Single-lane PCIe 3.0 bus	985 MB/sec
Thunderbolt 2 bus	2.5 GB/sec
SONET OC-768 network	5 GB/sec

Device Controllers

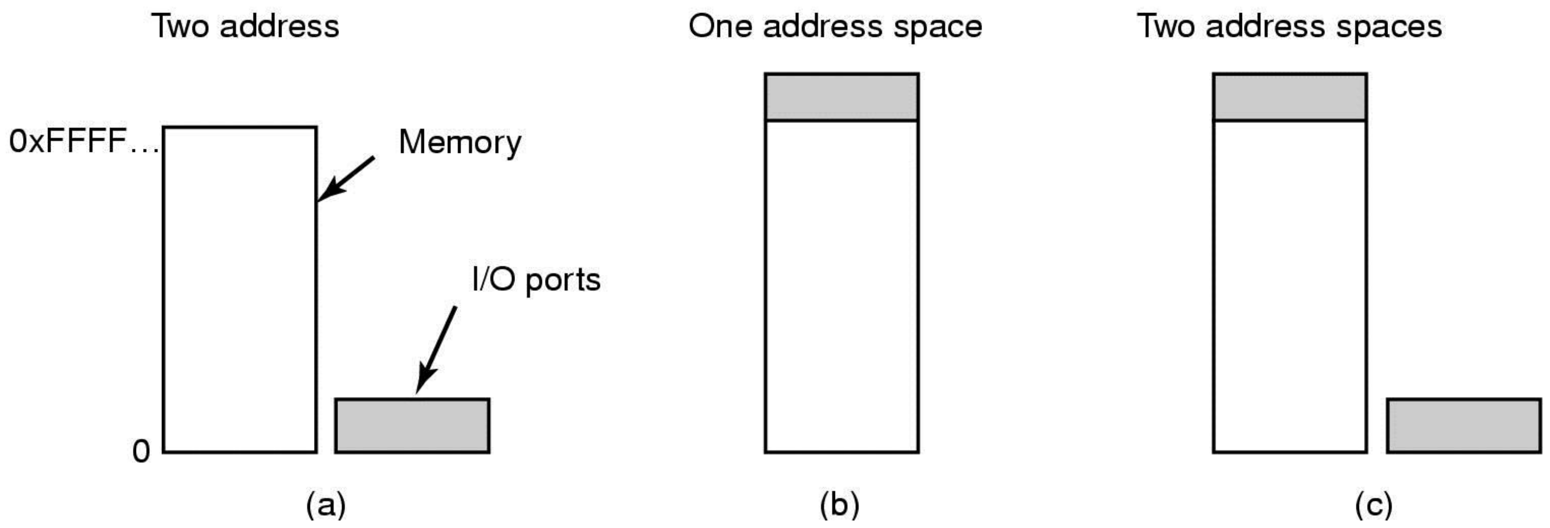
- I/O devices have components:
  - mechanical component
  - electronic component
- The electronic component is the device controller
  - may be able to handle multiple devices
- Controller's tasks
  - convert serial bit stream to block of bytes
  - perform error correction as necessary
  - make available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

Memory-Mapped I/O (1)

- Each controller has a few registers for communicating with the CPU
  - Control registers
    - By writing into control registers, the OS can command the device to deliver data, accept data, etc.
    - By reading from control registers, the can learn what the device's state is, whether it is prepared to accept a new command, etc.
  - Data buffer
    - The OS can read data from it or write data to it

Memory-Mapped I/O (2)

How the CPU communicates with the control registers and the device data buffers



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

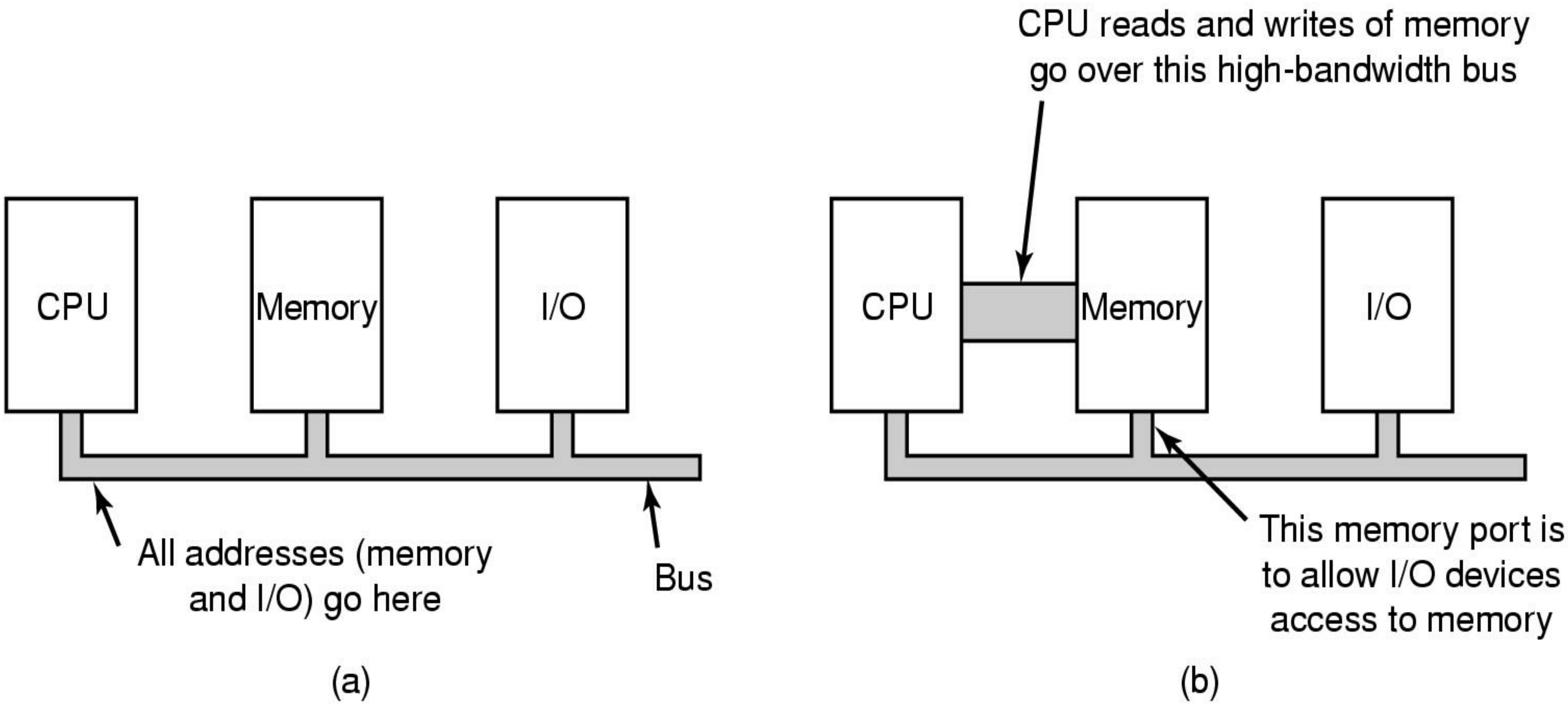
Memory-Mapped I/O (3)

- Separate I/O and memory space
  - Each control register is assigned an I/O port number
  - Special I/O instructions
    - IN REG, PORT
      - e.g. IN R0, 4 (The CPU reads in control register 4 and store the result in CPU register R0) It is different from MOV R0, 4 (The CPU reads the contents of the memory word 4 and puts it in R0)
    - OUT PORT, REG
      - e.g. OUT 7, R1 (The CPU writes the contents of CPU register R1 to a control register 7)

Memory-Mapped I/O (4)

- Memory-mapped I/O
  - Maps all the control register into the memory space
  - Each control register is assigned a unique memory address to which no memory is assigned such as at the top of the memory
- Hybrid
  - Memory-mapped I/O data buffers
  - Separate I/O ports for the control registers
  - e.g. Pentium
    - Addresses 640K to 1M reserved for device data buffers in IBM PC compatibles
    - I/O ports 0 through 64K

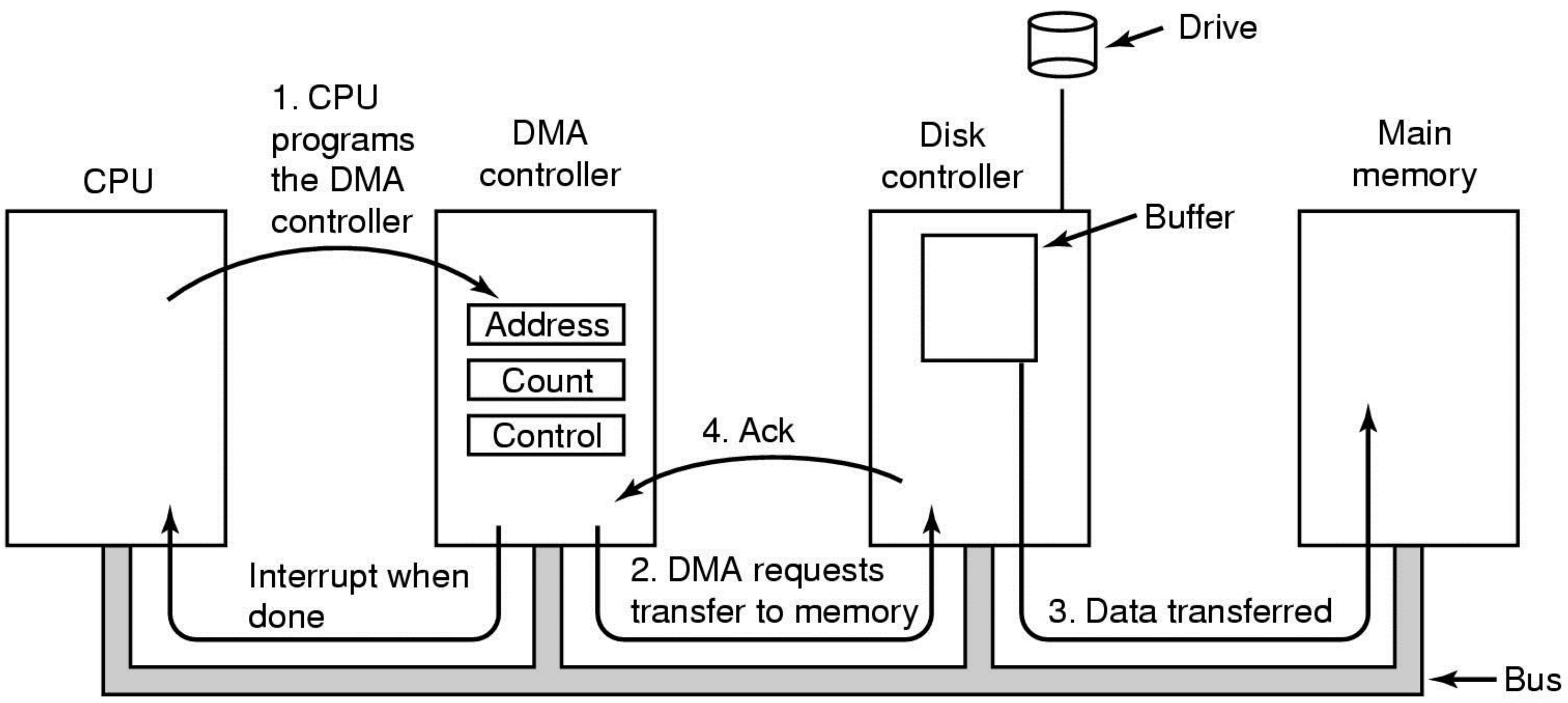
Memory-Mapped I/O (5)



(a) A single-bus architecture

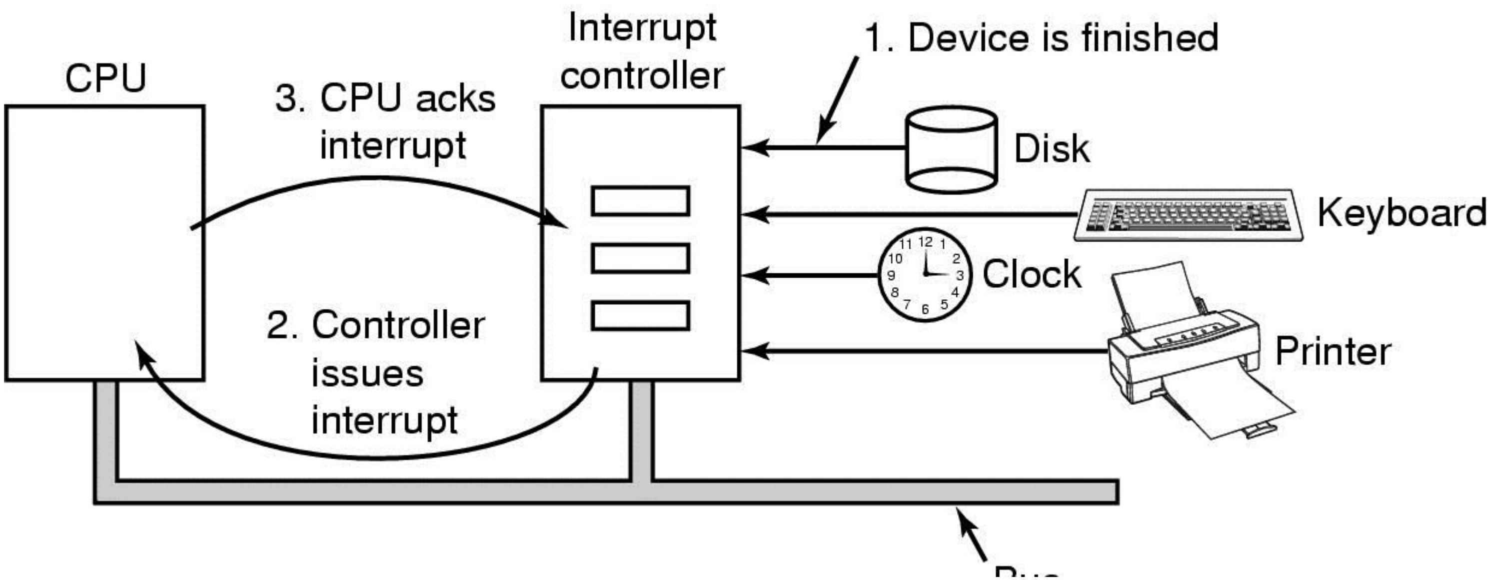
(b) A dual-bus memory architecture

Direct Memory Access (DMA)



Operation of a DMA transfer

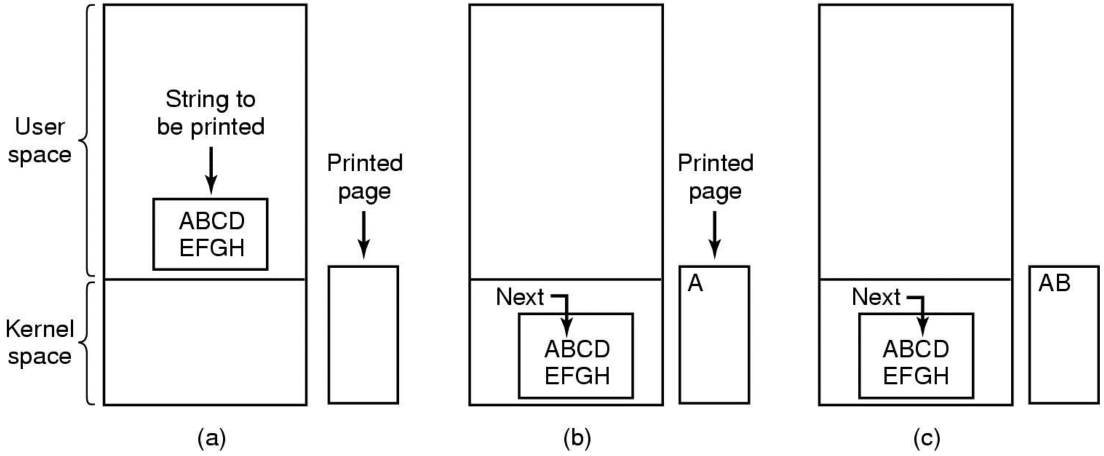
Interrupts Revisited



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

Programmed I/O

- Steps in printing a string



- Writing a string to the printer using programmed I/O

```
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( );
```

Interrupt-Driven I/O

- Writing a string to the printer using interrupt-driven I/O

  (a) Code executed when print system call is made

  (b) Interrupt service procedure

```
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)

I/O Using DMA

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

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

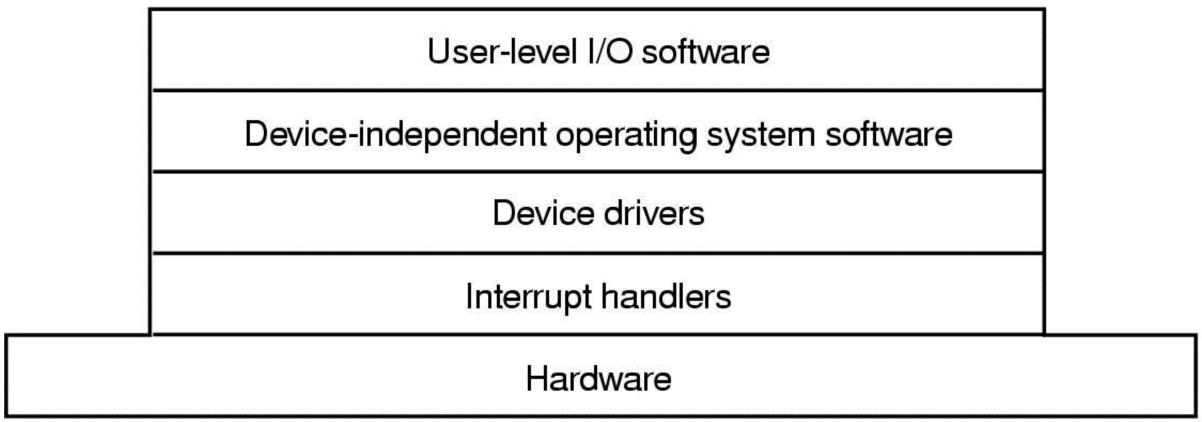
(a)

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

(b)

I/O Software Layers

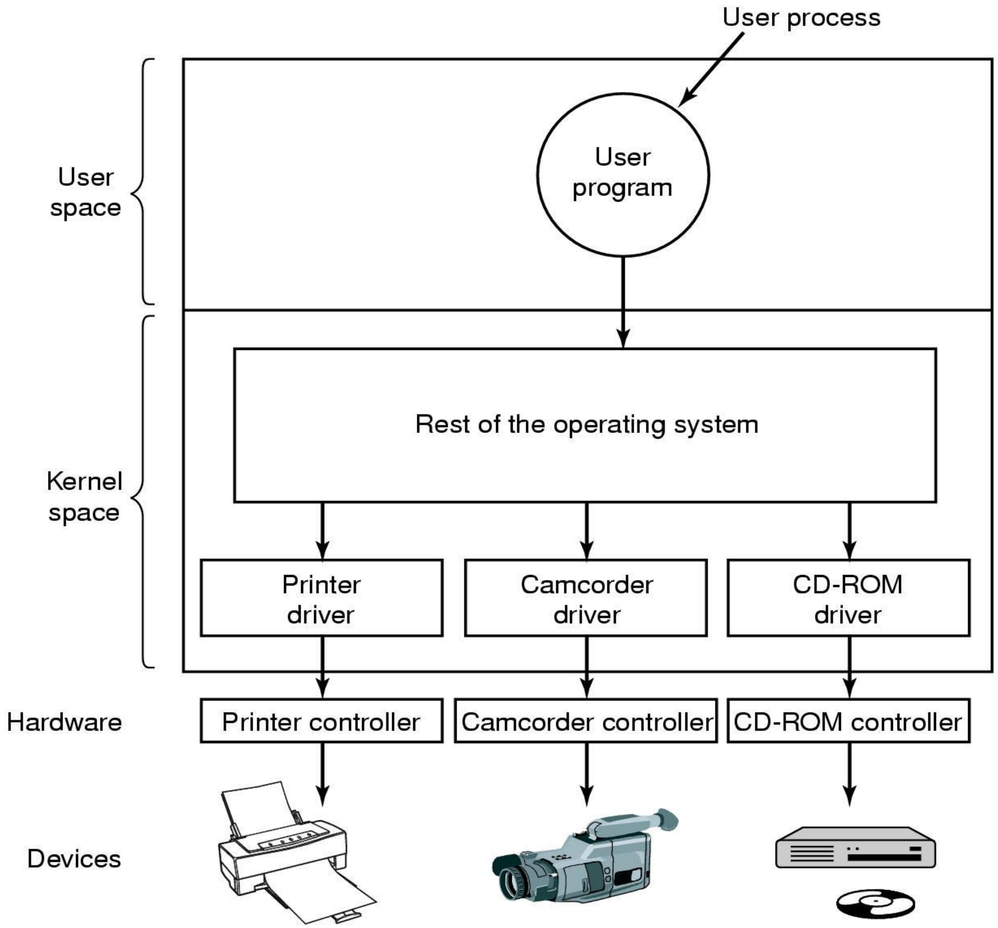
- Layers of the I/O Software System



Interrupt Handlers

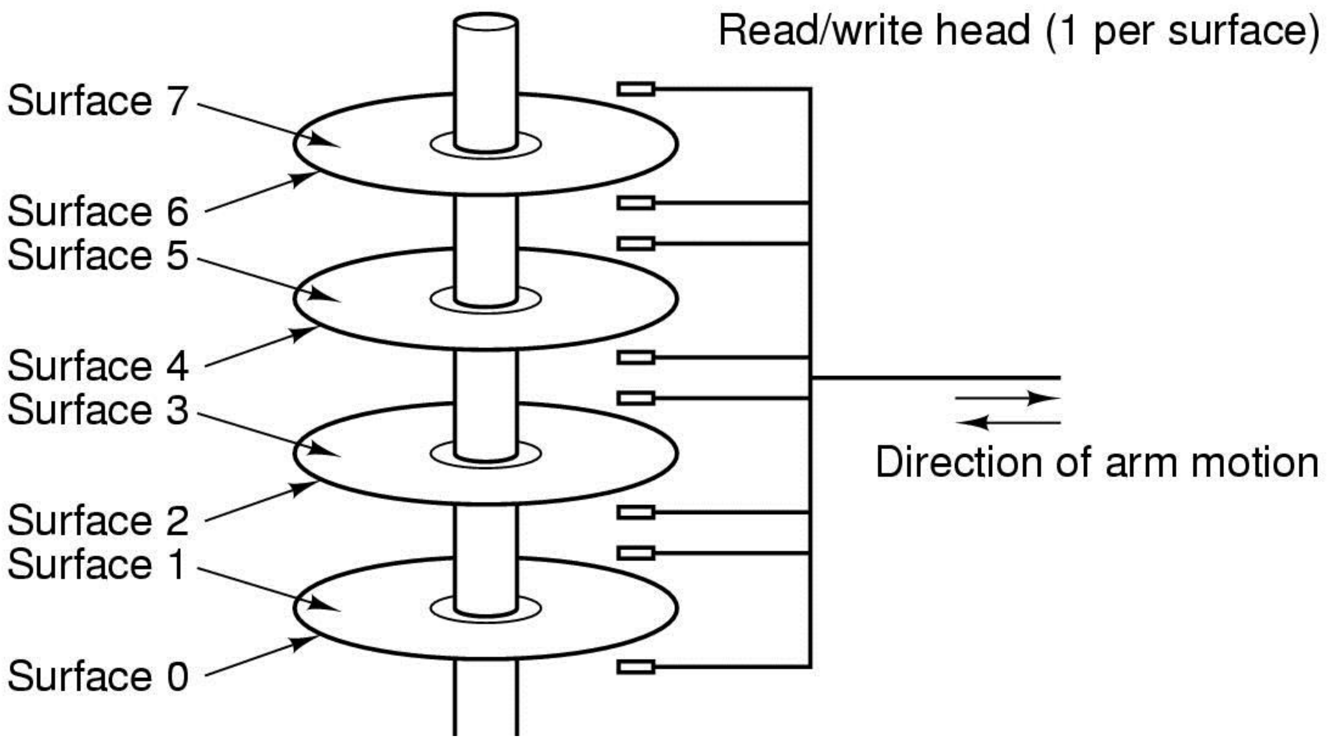
- Interrupt handlers are best hidden
  - have driver starting an I/O operation block until interrupt notifies of completion
- Interrupt procedure does its task
  - then unblocks driver that started it
- Steps must be performed in software after interrupt completed
  - Save regs not already saved by interrupt hardware
  - Set up context for interrupt service procedure
  - Set up stack for interrupt service procedure
  - Copy registers from where saved
  - Run service procedure
  - Set up MMU context for process to run next
  - Load new process' registers
  - Start running the new process

Device Drivers



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

Structure of a Disk Drive

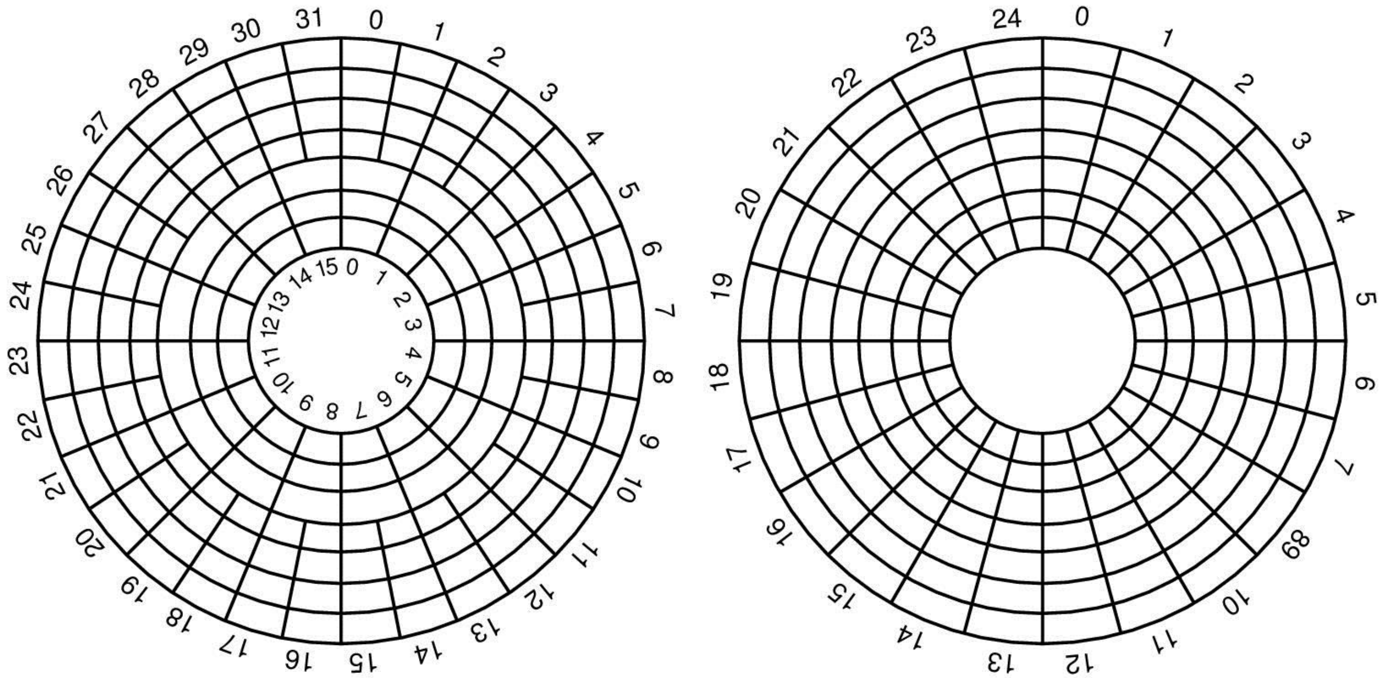


Disks
Disk Hardware (1)

Parameter	IBM 360-KB floppy disk	WD 3000 HLFS hard disk
Number of cylinders	40	36481
Tracks per cylinder	2	255
Sectors per track	9	63 (avg)
Sectors per disk	720	586,072,368
Bytes per sector	512	512
Disk capacity	360 KB	300 GB
Seek time (adjacent cylinders)	6 msec	0.7 msec
Seek time (average case)	77 msec	4.2 msec
Rotation time	200 msec	6 msec
Time to transfer 1 sector	22 msec	1.4 μsec

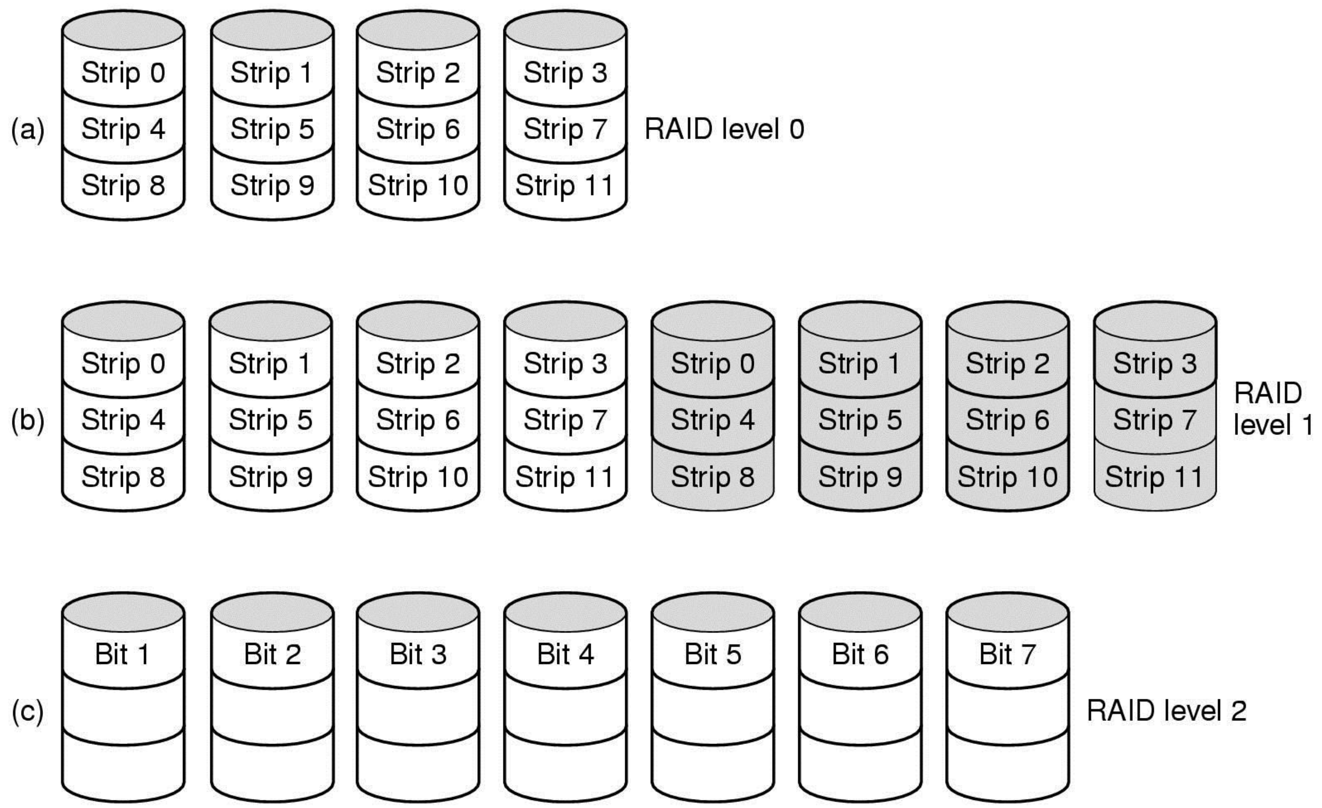
Disk parameters for the original IBM PC floppy disk and a Western Digital WD 3000 HLFS(“Velociraptor) hard disk

Disk Geometry



- Physical geometry of a disk with two zones (left)
  - More sectors on the outer zone than the inner one
- A possible virtual geometry for this disk (right)
  - Maximum values for IBM PC compatibility: (65535 cylinders, 16 heads, 63 sectors per track)
  - Logical block addressing
    - Sector numbers are numbered consecutively starting at 0, without regard to the disk geometry

RAID

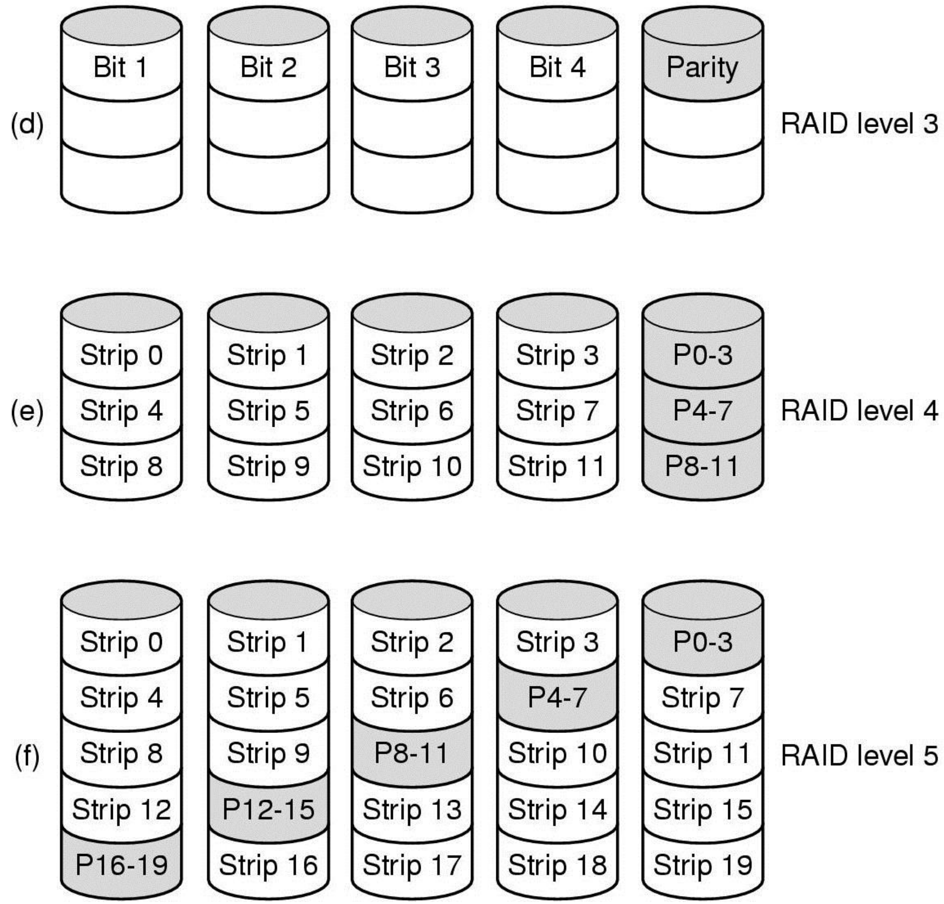


- Redundant Array of Independent Disks
- Raid levels 0 through 2
- Backup and parity drives are shaded

RAID

- Level 0
  - Works best with large requests
  - Works worst with OS's that habitually ask for data one sector at a time : no parallelism, performance gain
  - No redundancy: reliability is worse than a SLED(Single Large Expensive Disk)
- Level 1
  - Duplicates all the disks
  - Read performance can be up to twice as good.
  - Fault-tolerance is excellent:
    - if a drive crashes, the copy is simply used instead.
- Level 2
  - Works on a word or byte basis
  - Hamming code is used
  - Drives need to be synchronized in terms of arm position and rotational position
  - The controller must do a Hamming checksum every bit time.

RAID



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

RAID

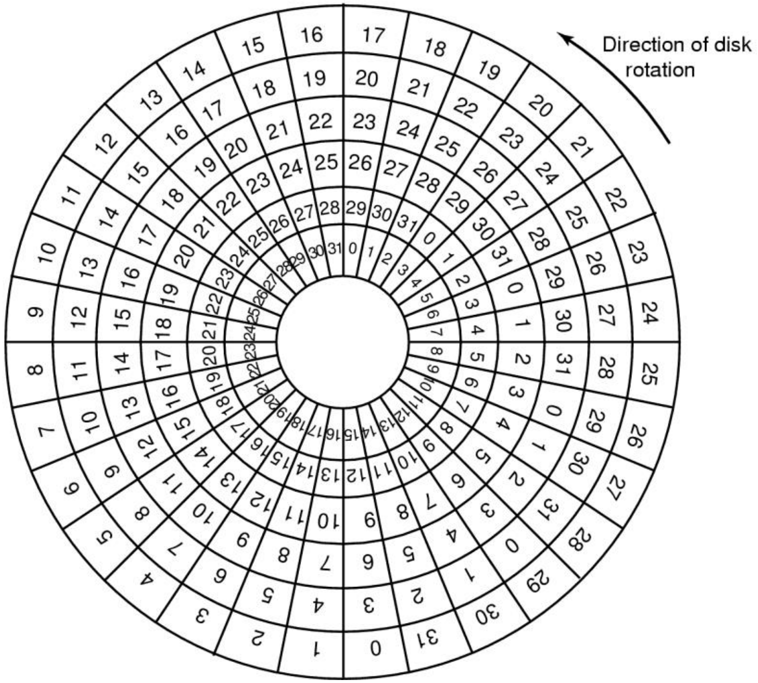
- Level 3
  - Single parity bit is computed for each data word and written to a parity drive
    - 1-bit error detection for random undetected errors
    - 1-bit error correction for a drive crashing
  - Drives must be synchronized.
- Level 4
  - Like level 0 RAID with a strip-for-strip parity written onto an extra drive
  - Do not require synchronized drives
  - If a drive crashes, the lost bytes can be recomputed from the parity drive.
  - Performs poorly for small updates
    - If one sector is changed, it is necessary to read all the drives in order to recalculate the parity, which must then be rewritten.
  - Heavy load on the parity drive
- Level 5
  - Eliminates bottleneck in the parity drive by distributing parity bits uniformly over all the drives, round robin fashion
  - Reconstructing the contents of the failed drive is a complex process.

Low-level Formatting

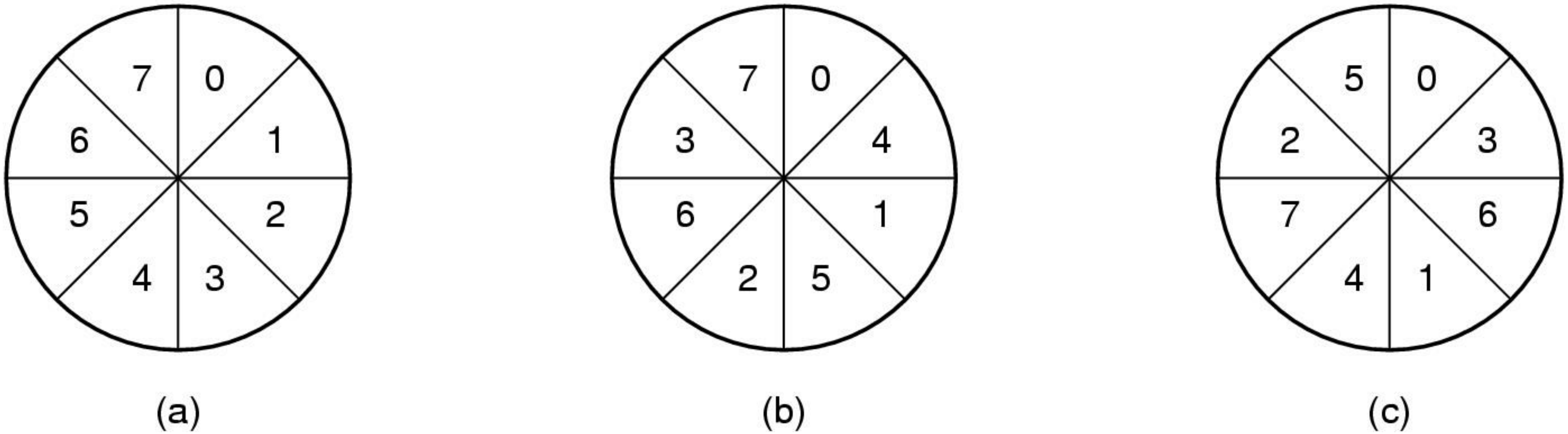
- A disk sector
  - Preamble
    - Starts with a bit pattern that allows the hardware to recognize the start of the sector
    - Contains the cylinder and sector numbers, etc.
  - Data
  - ECC(Error-Correcting Code)
    - Redundant information that can be read to recover from read errors
- Spare sectors

Cylinder Skew

- Cylinder skew
  - The position of sector 0 on each track is offset from the previous track when the low-level format is laid down.
  - To improve performance
    - Allows the disk to read multiple tracks in one continuous operation without losing data.



Disk Interleaving



(a) No interleaving

(b) Single interleaving

(c) Double interleaving

- To avoid the need for interleaving, the controller should be able to buffer an entire track.

Sequence of Disk Formatting

- Low-level formatting
- Partitioning
  - MBR (Master Boot Record)
    - Boot code
    - Partition table
      - Starting sector and size of each partition
- High-level formatting
  - Done for each partition separately
  - Creates a file system
    - Boot block, super block, free storage adm., root dir, etc.

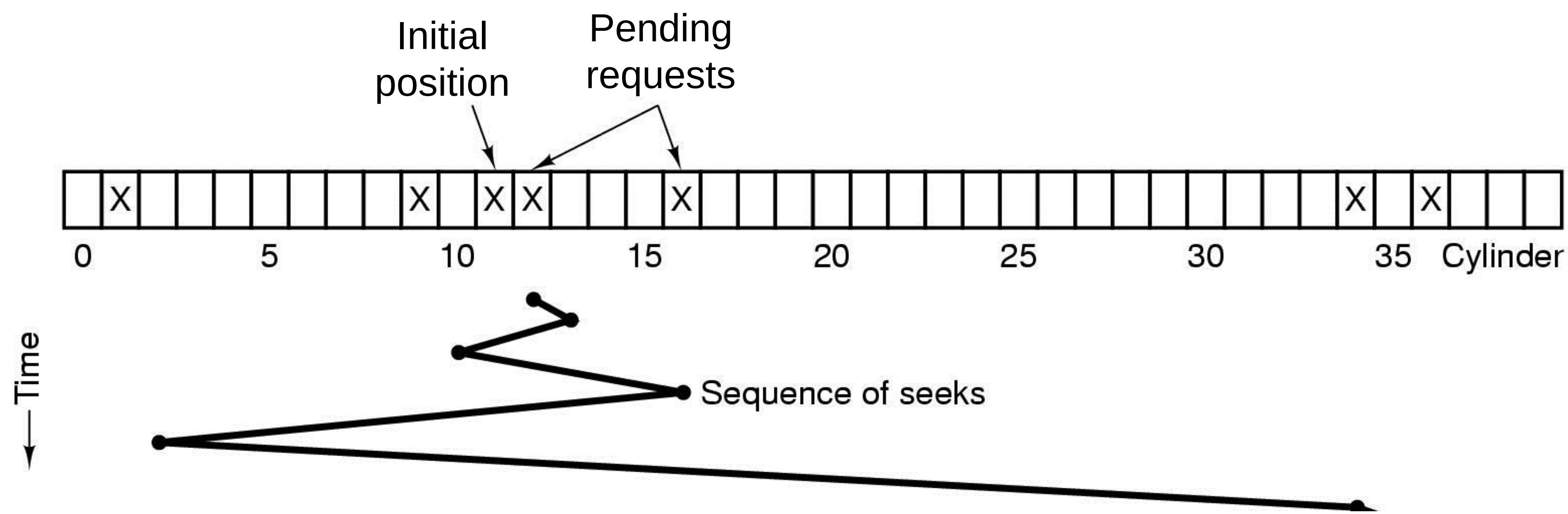
Disk Arm Scheduling Algorithms (1)

- Time required to read or write a disk block determined by 3 factors
  1. Seek time
     - the time to move the arm to the proper cylinder
  2. Rotational delay
     - how long for the proper sector to come under the head
  3. Actual transfer time
- Seek time dominates
- Error checking is done by controllers

First-Come, First-Served (FCFS) Algorithm

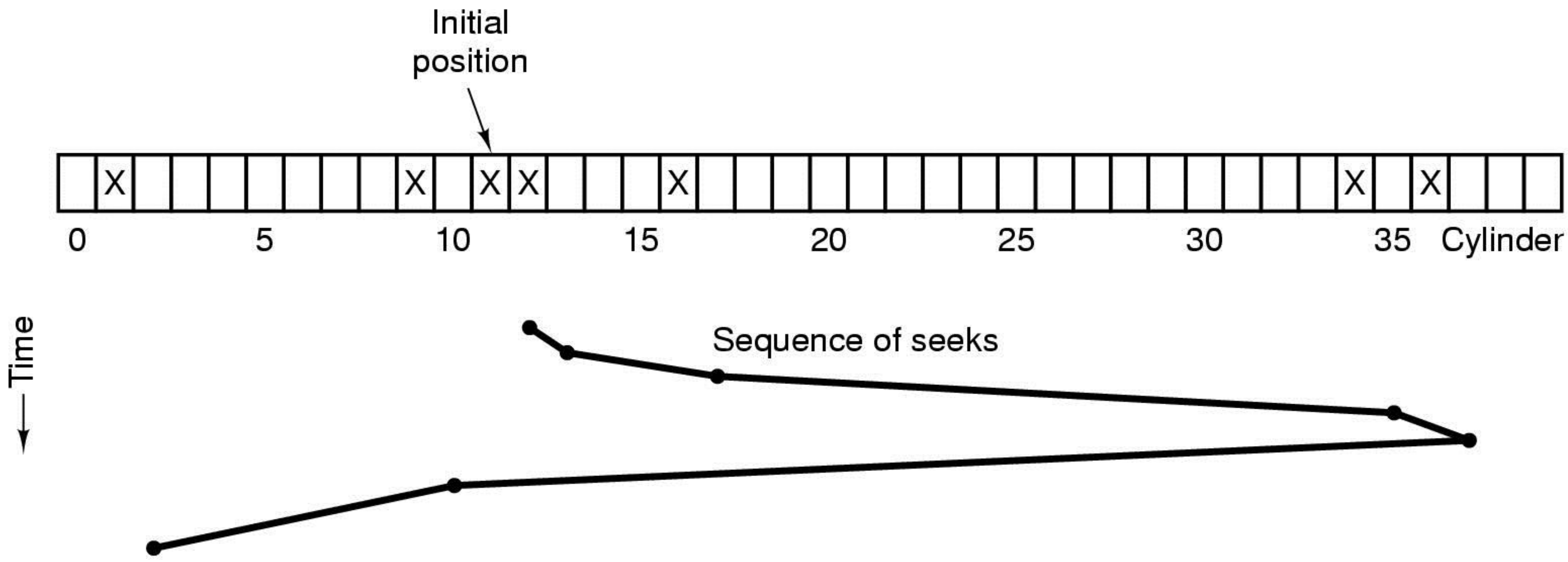
- Handles requests in the order they come in

Shortest Seek First (SSF) Algorithm



- Handles the closest request next, to minimize seek time
- Request far from the middle may get poor service.

Elevator (Scan) Algorithm



- Keeps moving in the same direction until there are no more requests in that direction, then switches direction
- Requires one bit to keep track of the current direction: Up or Down
- The upper bound on the total motion is fixed:
  - 2 * (number of cylinders)

C-Scan (Circular-Scan) Algorithm

- Similar to Scan
- Moves in one direction only (upward only)
- Smaller variance in response times than Scan
- The lowest-numbered cylinder is thought of as being just above the highest-numbered cylinder

Other Optimization

- Any request to read a sector will cause that sector and much or all the rest of the current track to be read, depending upon how much space is available in the controller’s cache memory

- Disk controller’s cache holds blocks that have not actually been requested, while any cache maintained by the operating system will consist of blocks that were explicitly read.