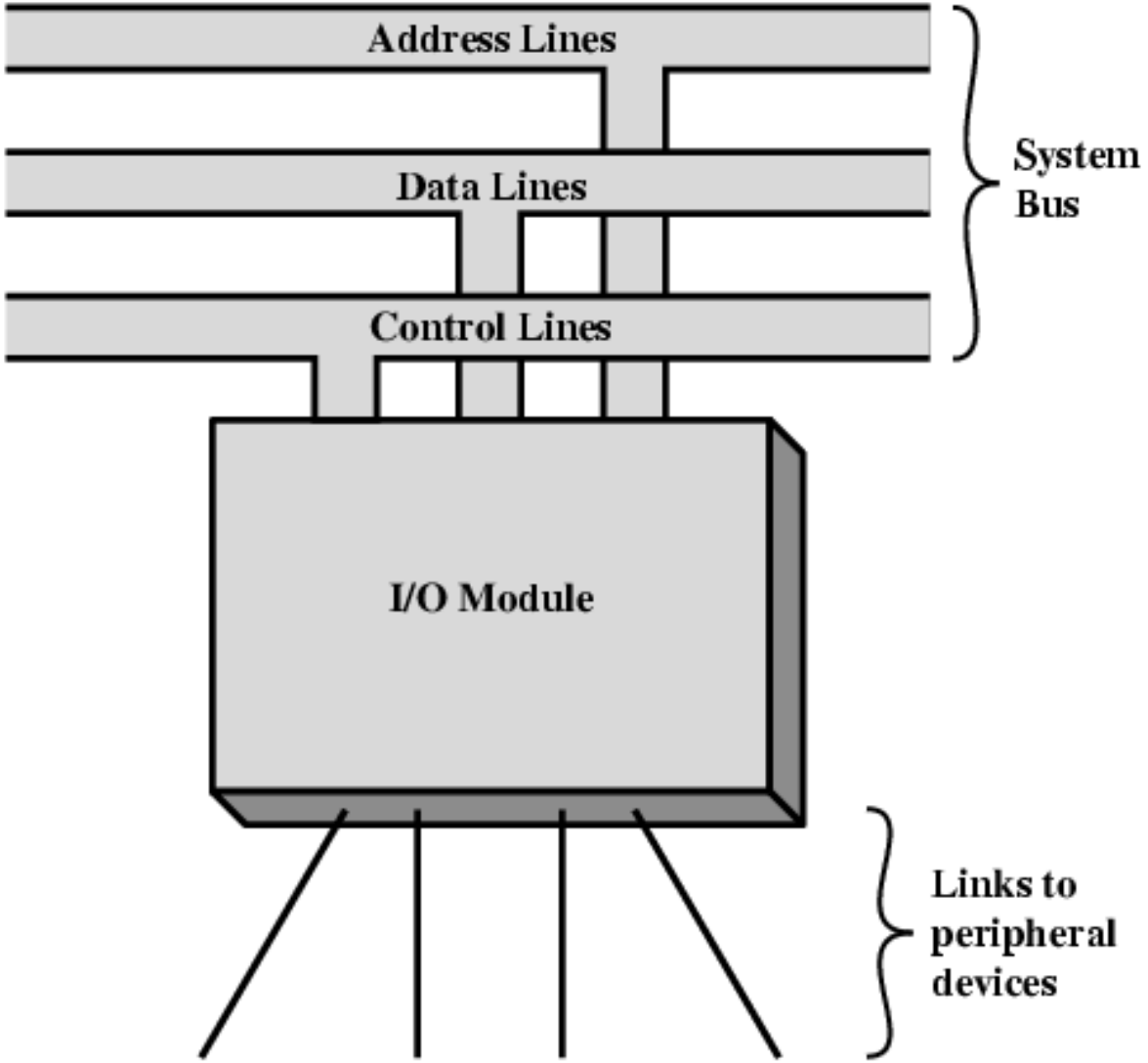
Input/output

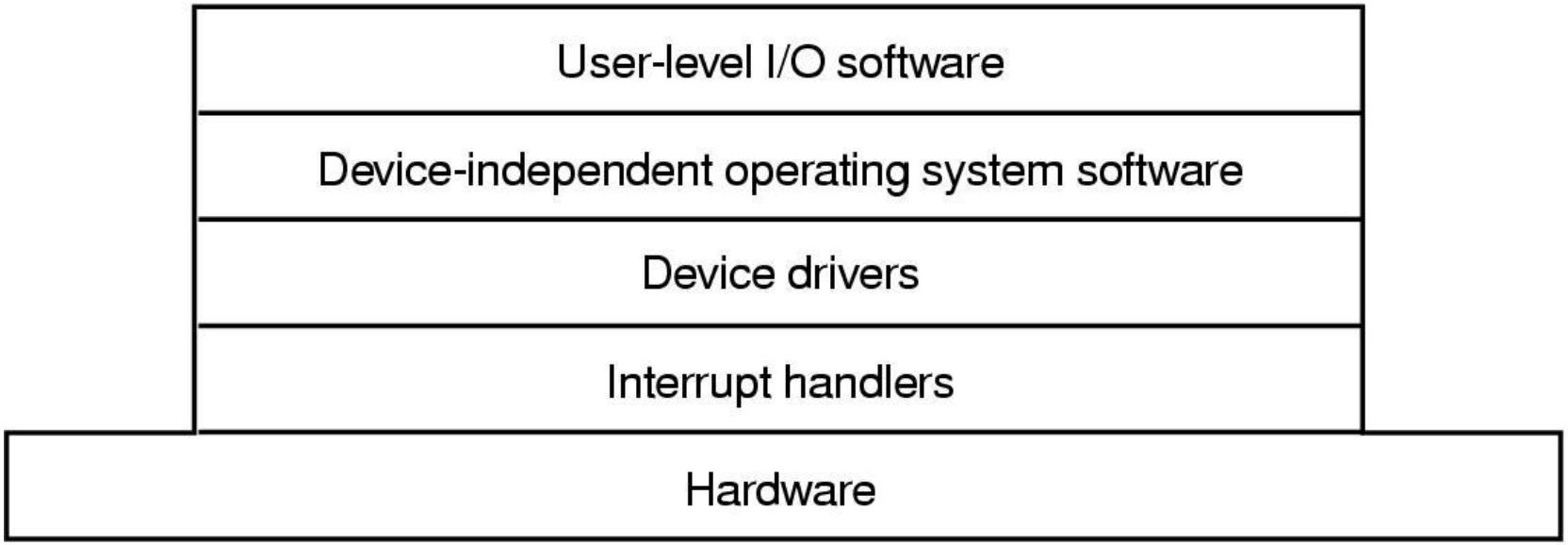
Input/Output Problems

- Wide variety of peripherals
  - Delivering different amounts of data
  - At different speeds
  - In different formats
- All slower than CPU and RAM
- Need I/O modules

Generic Model of I/O Module



I/O Software Layers



Layers of the I/O Software System

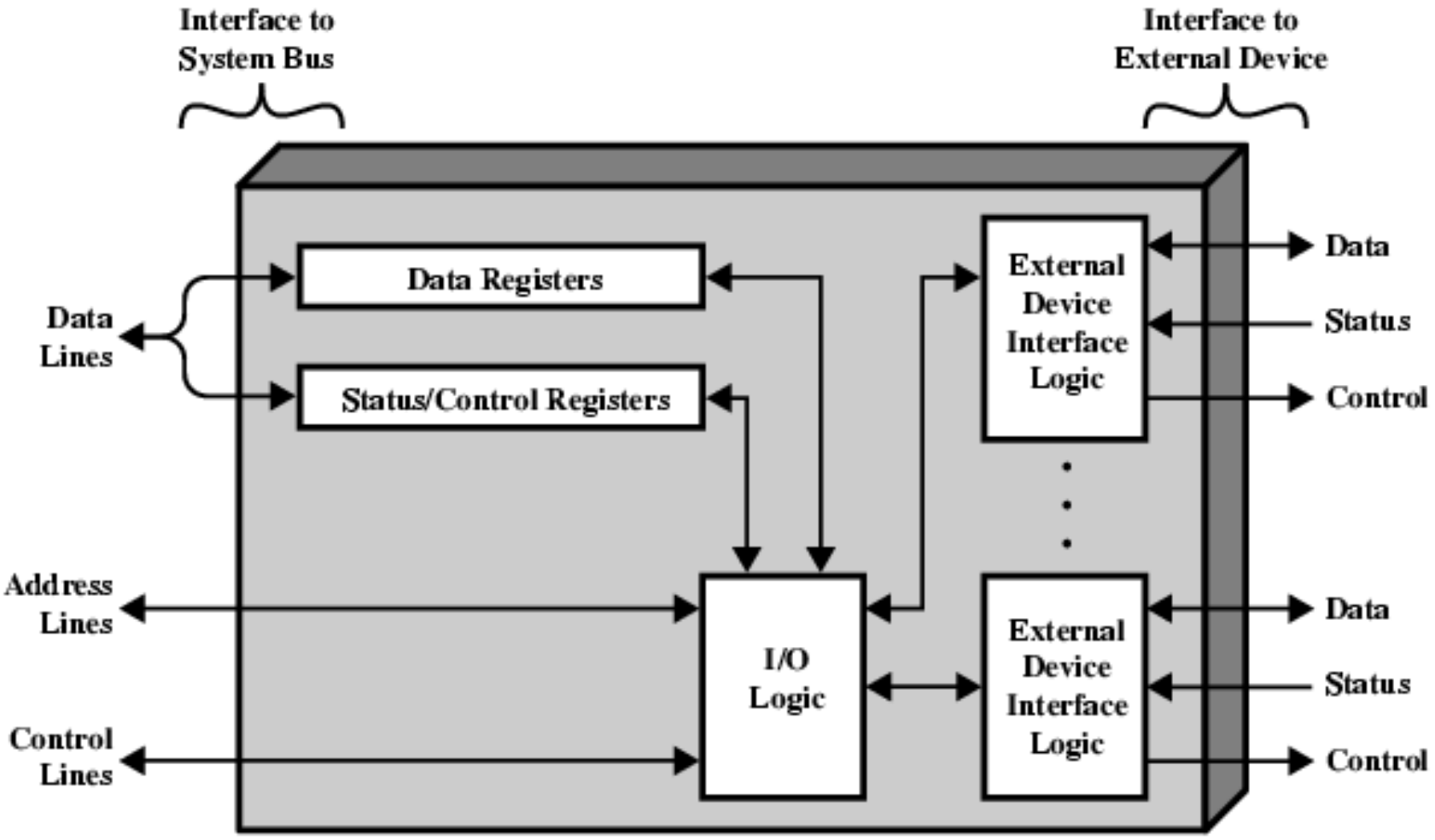
I/O Module Function

- Support single or multiple devices
- Hide or reveal device properties

Provides:

- Control & Timing
- CPU Communication
- Device Communication
- Data Buffering
- Error Detection

I/O Module Diagram



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

Principles of I/O Software

- Device independence
  - programs can access any I/O device
  - without specifying device in advance
    - (floppy, hard drive, or CD-ROM)
- Buffering
  - data coming off a device cannot be stored in final destination
- Error handling
  - handle as close to the hardware as possible

Input Output Techniques

- Programmed
- Interrupt driven
- Direct Memory Access (DMA)

Programmed I/O

- CPU has direct control over I/O
  - Sensing status
  - Read/write commands
  - Transferring data
- CPU waits for I/O module to complete operation
- Usually not a good use of CPU time

Programmed I/O - detail

- CPU requests I/O operation
- I/O module performs operation
- I/O module sets status bits
- CPU checks status bits periodically
- CPU may wait or come back later

Interrupt driven I/O - CPU Viewpoint

- Issue I/O command
- Do other work
  - Check for interrupt at end of each instruction cycle
- When interrupt request is granted:-
  - Save context (registers)
  - Process interrupt
    - Execute “service routine”
- Continue other work

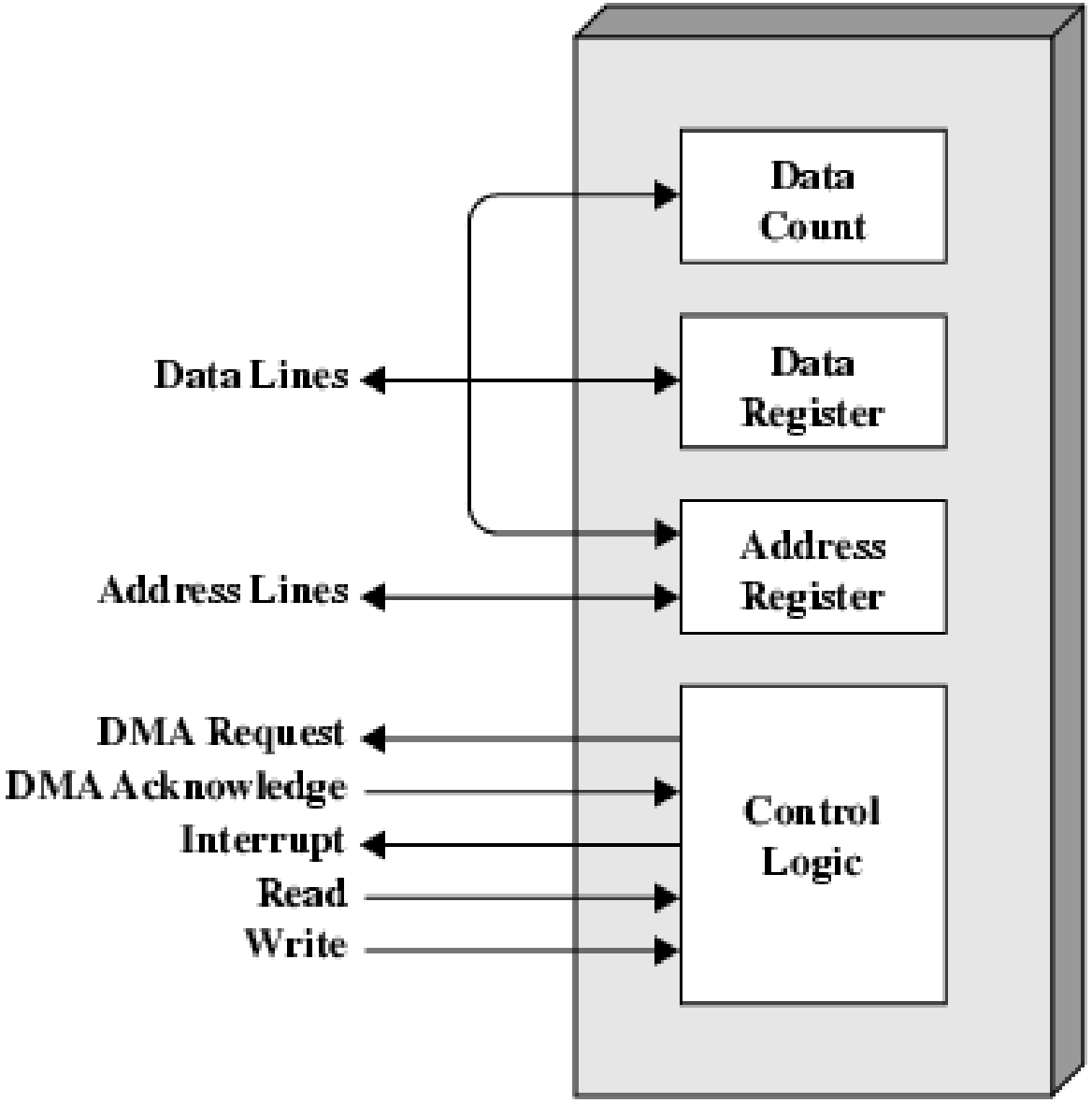
Interrupt Driven I/O – Device Perspective

- CPU issues I/O command (enable interrupt)
- I/O module gets data from peripheral while CPU does other work
- I/O module interrupts CPU (Interrupt request)
- Device serviced by CPU

DMA Function

- DMA controller(s) takes over from CPU for I/O
- Additional Module(s) attached to bus

Typical DMA Module Diagram



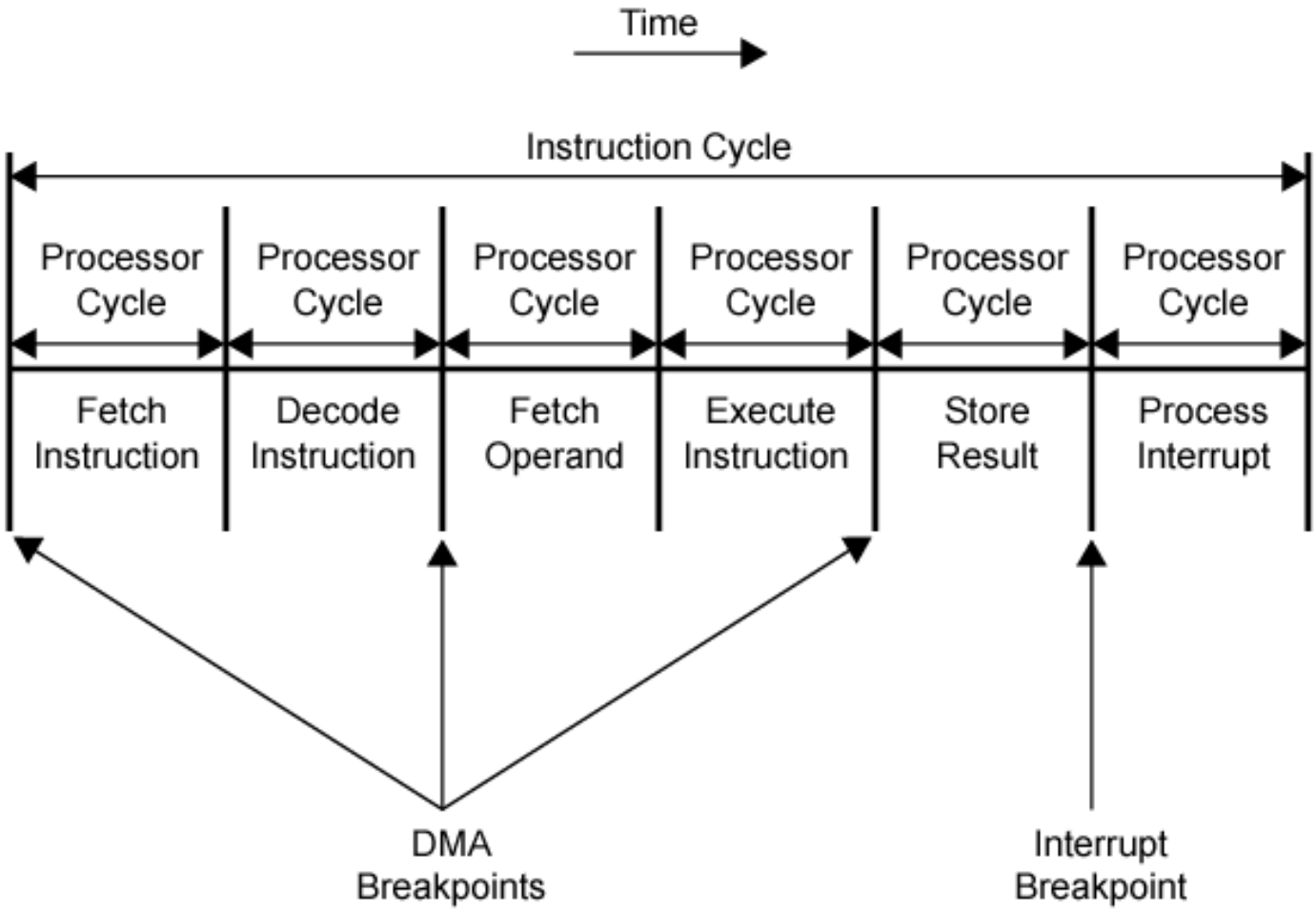
DMA Operation

- CPU tells DMA controller:-
  - Read/Write
  - Device address
  - Starting address of memory block for data
  - Amount of data to be transferred
- CPU carries on with other work
- DMA controller deals with transfer
- DMA controller sends interrupt when finished

DMA Transfer
Cycle Stealing

- DMA controller takes over bus for a cycle
- Transfer of one word of data
- Not an interrupt
  - CPU does not switch context
- CPU suspended just before it accesses bus
  - i.e. before an operand or data fetch or a data write
- Slows down CPU but not as much as CPU doing transfer

DMA and Interrupt Breakpoints During an Instruction Cycle



What is wrong with this?

I/O Channels

- I/O channels are processors dedicated to I/O

  e.g. 3D graphics cards
- CPU instructs I/O controller to do transfer
- I/O controller does entire transfer from one or many devices
- Makes transfers less visible to CPU
- Improves speed
  - Takes load off CPU

Difference between iosolated io and memory mapped io

- **Isolated I/O**
- Isolated I/O uses separate memory space.
- Limited instructions can be used. Those are IN, OUT, INS, OUTS.
- The addresses for Isolated I/O devices are called ports

- **Memory Mapped I/O**
- Memory mapped I/O uses memory from the main memory.
- Any instruction which references to memory can be used.
- Memory mapped I/O devices are treated as memory locations on the memory map.

Difference between iosolated io and memory mapped io

Table 7.1: Comparison between Memory Mapped I/O and I/O Mapped I/O

S. No.	Memory Mapped I/O		I/O Mapped I/O
1.	Address width is 16-bit. A_0 to A_{15} are used to generate address of the device.	1.	Address width is 8-bit. A_0 to A_{15} lines are used to generate address of the device.
2.	$\overline{\text{MEMR}}$ and $\overline{\text{MEMW}}$ control signals are used to control read and write I/O operations respectively.	2.	$\overline{\text{IOR}}$ and $\overline{\text{IOW}}$ control signals are used to control read and write I/O operations respectively.
3.	Instructions available are STA addr, LDA addr, LDAX rp, STAX rp, ADD M, CMP M, MOV r, M, etc.	3.	IN and OUT are the only available instructions.
4.	Data transfer takes place between any register and I/O device.	4.	Data transfer takes place between accumulator and I/O device.
5.	Maximum number of I/O devices that can be addressed is 65536 (theoretically).	5.	Maximum number of I/O devices that can be addressed is 256.
6.	Execution speed using STA addr, LDA addr is 13 T-state and for MOV M, r, etc., it is 7-T states.	6.	Execution speed is 10 T-states.
7.	Decoding 16-bit address will require more hardware circuitry.	7.	Decoding 8-bit address will require less hardware circuitry.

Disk Scheduling

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth.
- Access time has two major components
  - *Seek time* is the time for the disk are to move the heads to the cylinder containing the desired sector.
  - *Rotational latency* is the additional time waiting for the disk to rotate the desired sector to the disk head.
- Minimize seek time
- Seek time ≈ seek distance
- Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.

Disk Scheduling (Cont.)

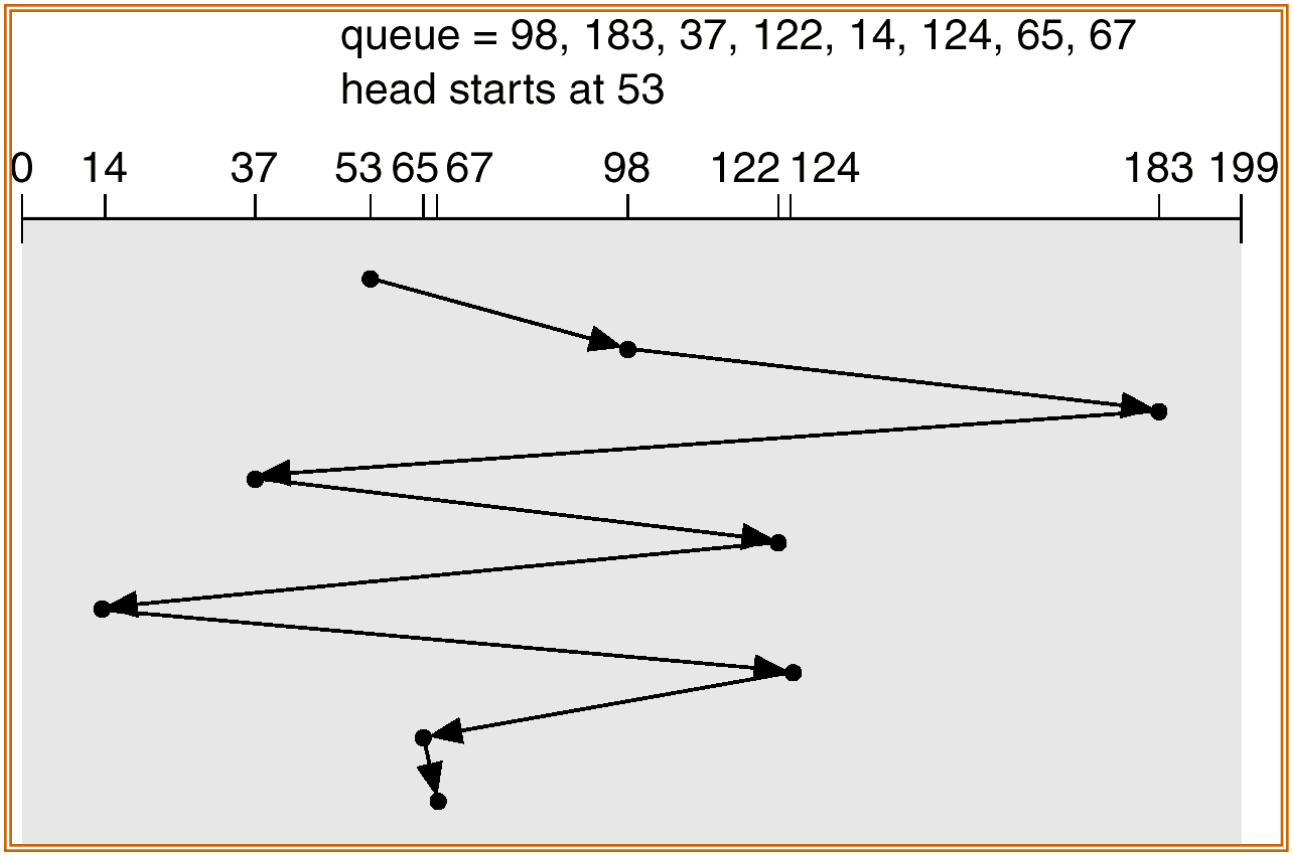
- Several algorithms exist to schedule the servicing of disk I/O requests.
- We illustrate them with a request queue (0-199).

98, 183, 37, 122, 14, 124, 65, 67

Head pointer 53

FCFS

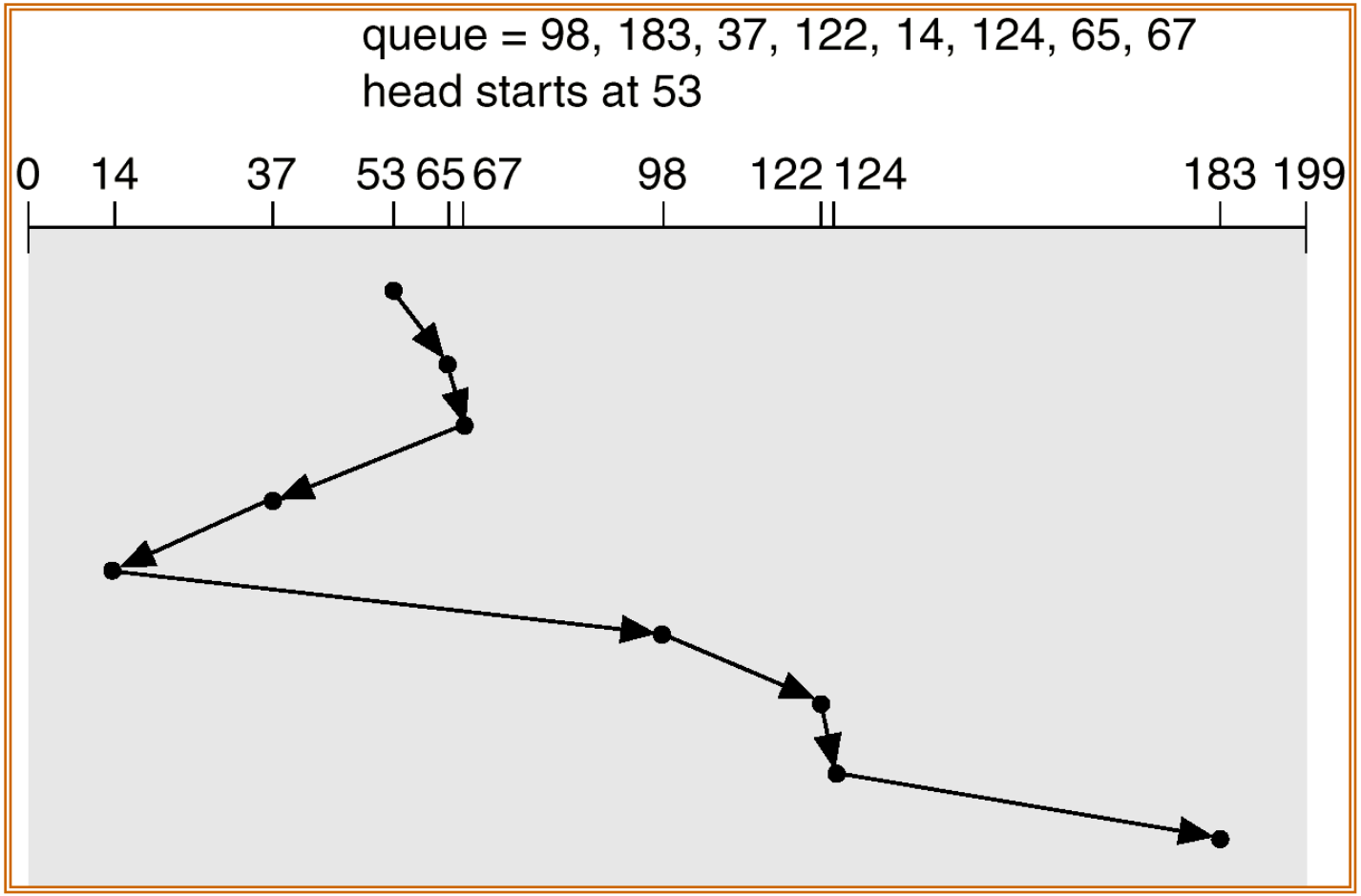
Illustration shows total head movement of 640 cylinders.



SSTF

- Selects the request with the minimum seek time from the current head position.
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests.
- Illustration shows total head movement of 236 cylinders.

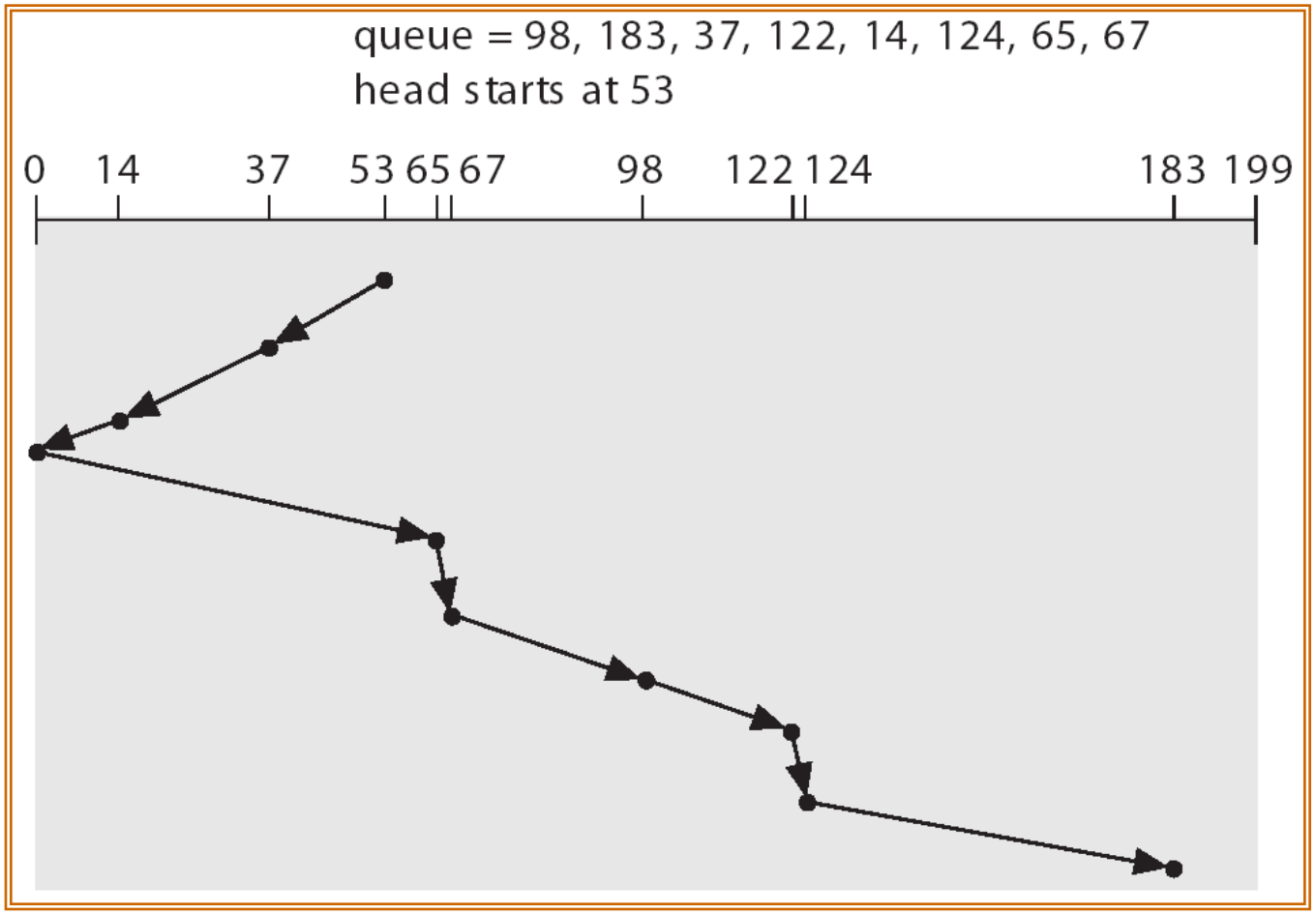
SSTF (Cont.)


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

SCAN

- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
- Sometimes called the *elevator algorithm*.
- Illustration shows total head movement of 208 cylinders.

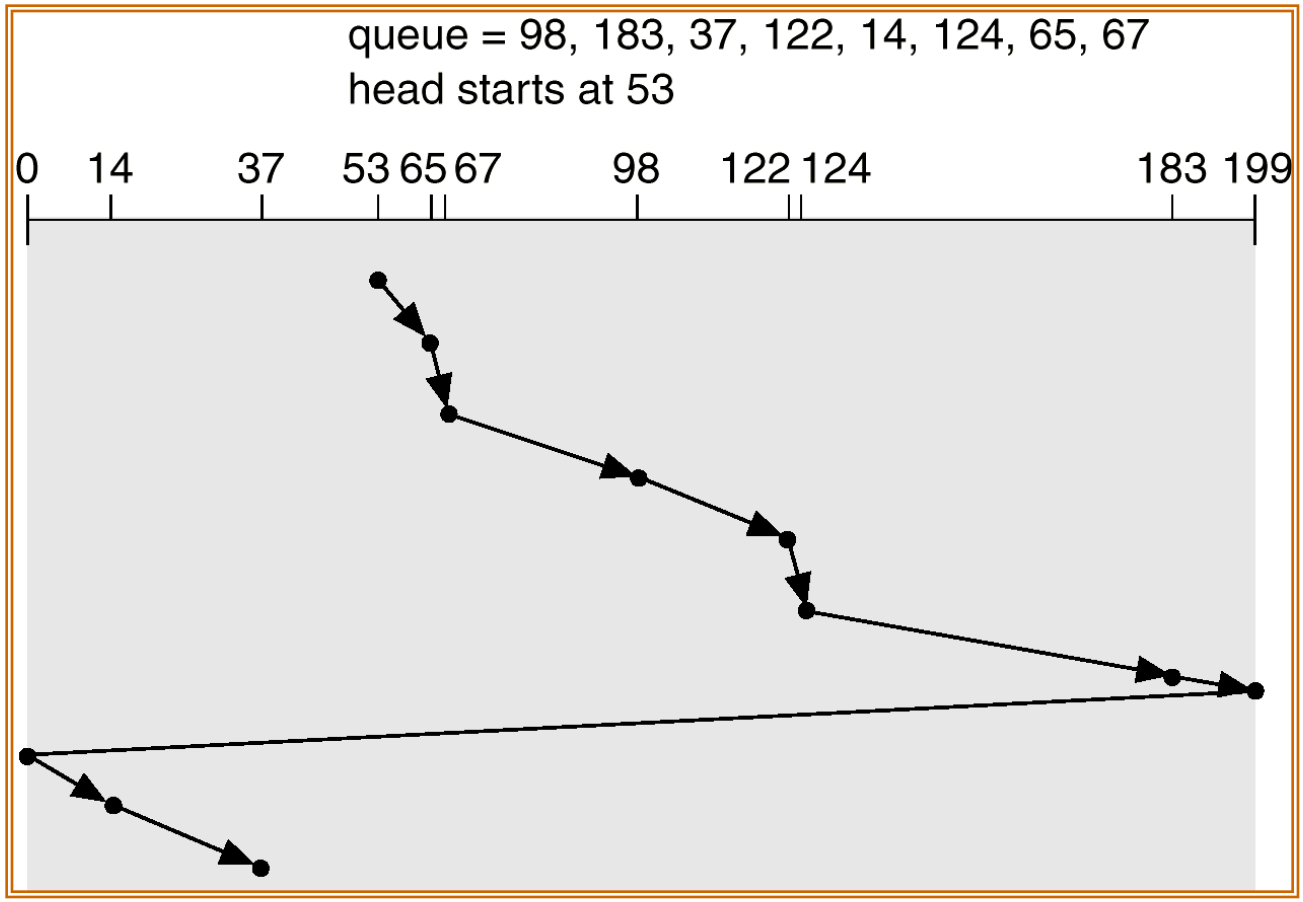
SCAN (Cont.)


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

C-SCAN

- Provides a more uniform wait time than SCAN.
- The head moves from one end of the disk to the other. servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one.

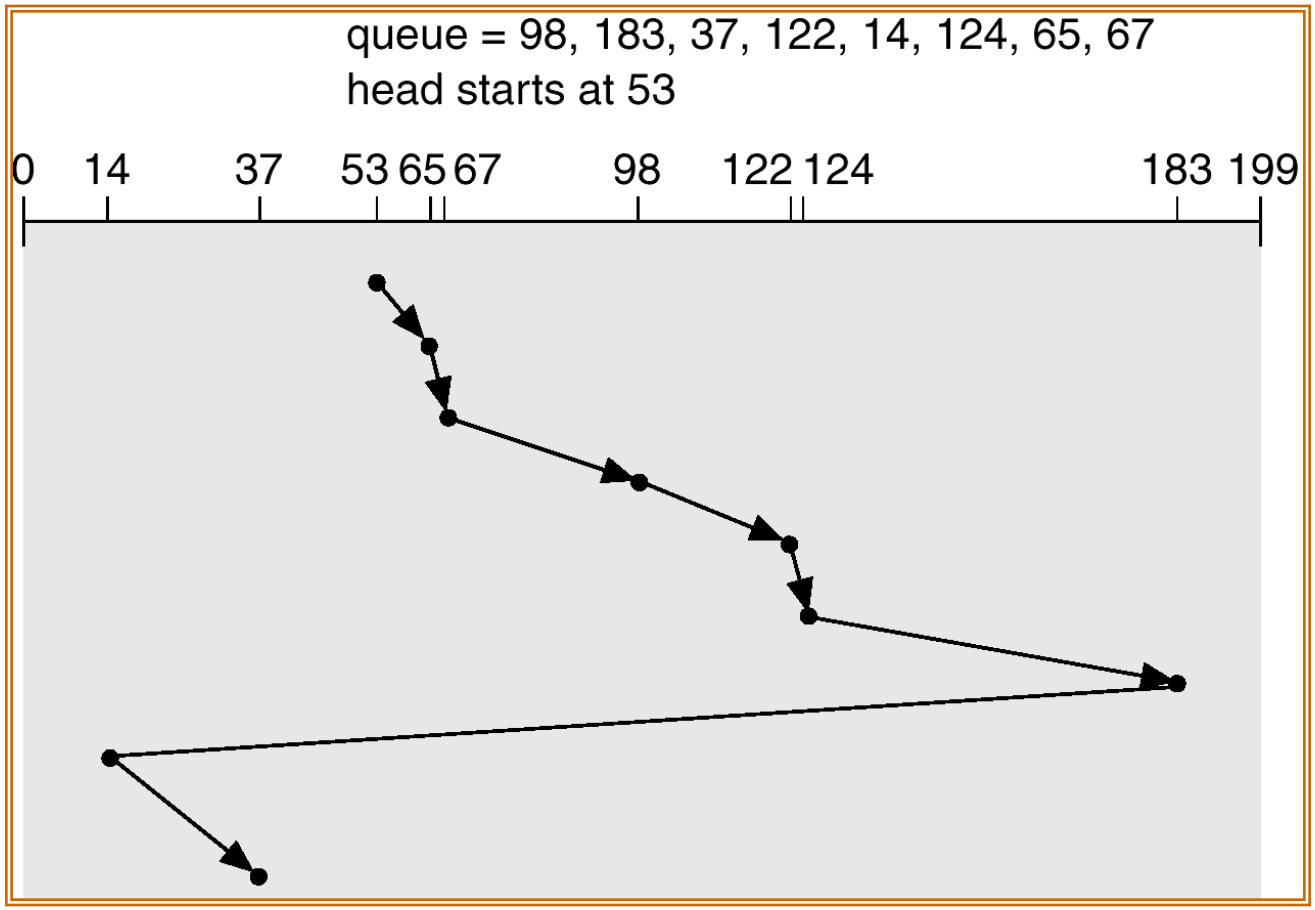
C-SCAN (Cont.)


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

C-LOOK

- Version of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.

C-LOOK (Cont.)


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

Selecting a Disk-Scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk.
- Performance depends on the number and types of requests.
- Requests for disk service can be influenced by the file-allocation method.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary.
- Either SSTF or LOOK is a reasonable choice for the default algorithm.

RAID Structure

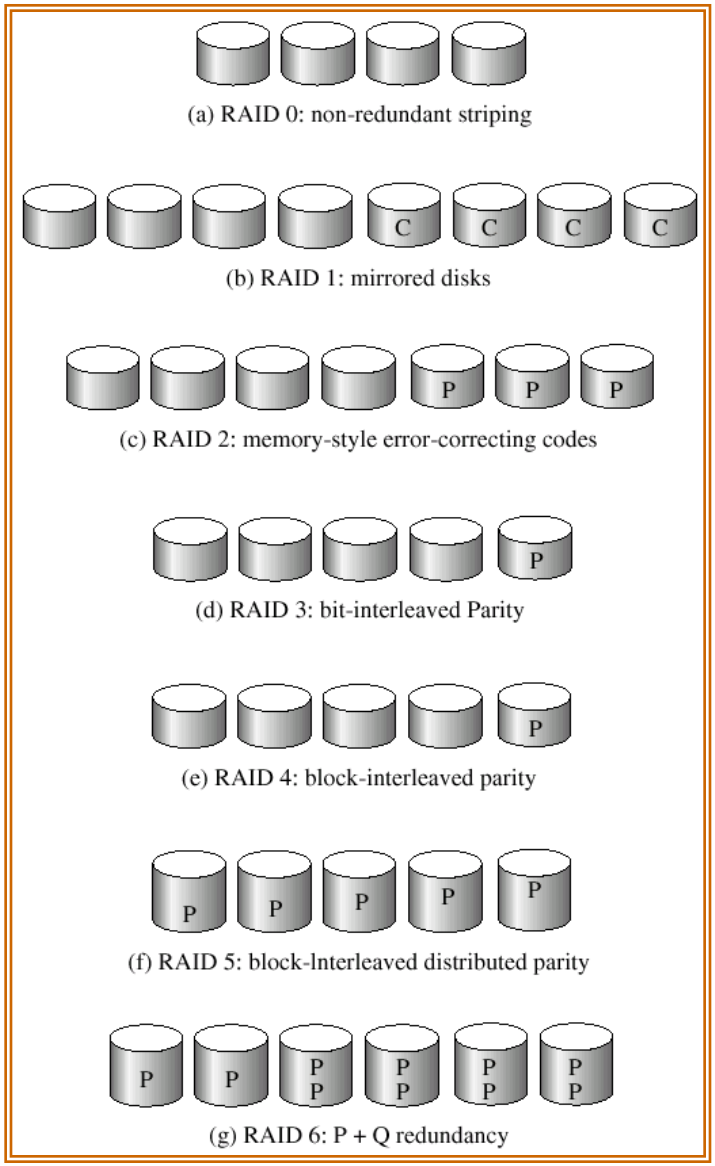
- **RAID** – multiple disk drives provides **reliability** via **redundancy**.

- RAID is arranged into six different levels.

RAID (cont)

- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively.
- Disk striping uses a group of disks as one storage unit.
- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data.
  - *Mirroring* or *shadowing* keeps duplicate of each disk.
  - *Block interleaved parity* uses much less redundancy.

RAID Levels


(a) RAID 0: non-redundant striping
C
(b) RAID 1: mirrored disks
P
(c) RAID 2: memory-style error-correcting codes
(d) RAID 3: bit-interleaved Parity
(e) RAID 4: block-interleaved parity
(f) RAID 5: block-lnterleaved distributed parity
(g) RAID 6: P + Q redundancy