3140702
Operating System

Unit – 7
I/O Management & Disk Scheduling

Subject Faculty: Prof. Suhag Baldaniya

Disclaimer

- ***It is hereby declared that the production of the said content is meant for non-commercial, scholastic and research purposes only.***

- ***We admit that some of the content or the images provided in this channel's videos may be obtained through the routine Google image searches and few of them may be under copyright protection. Such usage is completely inadvertent.***

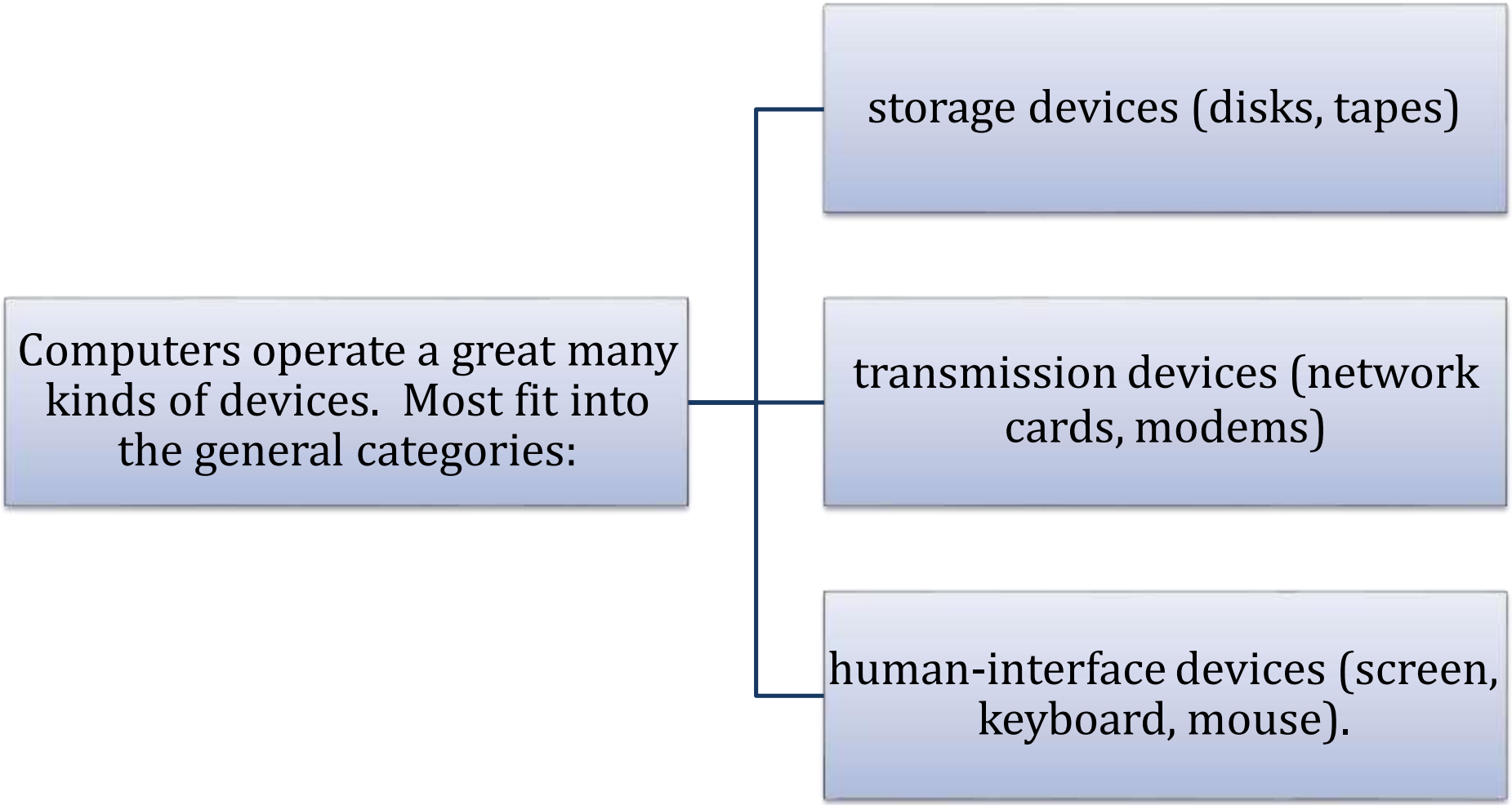
- ***It is quite possible that we overlooked to give full scholarly credit to the Copyright Owners. We believe that the non-commercial, only-for-educational use of the material may allow the video in question fall under fair use of such content. However we honor the copyright holder's rights and the video shall be deleted from our channel in case of any such claim received by us or reported to us.***

Topics to be covered

Marwadi University

- I/O Management
- Principles of I/O Hardware:
  - I/O devices
  - Device controllers
  - Direct memory access
- Secondary-Storage Structure:
  - Disk structure
  - Disk scheduling algorithm

Principles of I/O Hardware: I/O device



Principles of I/O Hardware: I/O device

Marwadi University

Two types of I/O devices	• block, character
Block device:	• It is one that stores information in fixed-size blocks each one with its on address. • can read blocks independently of one another • Hard disks, CD-ROMs, USB sticks • 512 bytes to 32,768 bytes
Character device:	• accepts characters without regard to block structure • Printers, mouse, network interfaces
Not everything fits:	• e.g. clocks don't fit

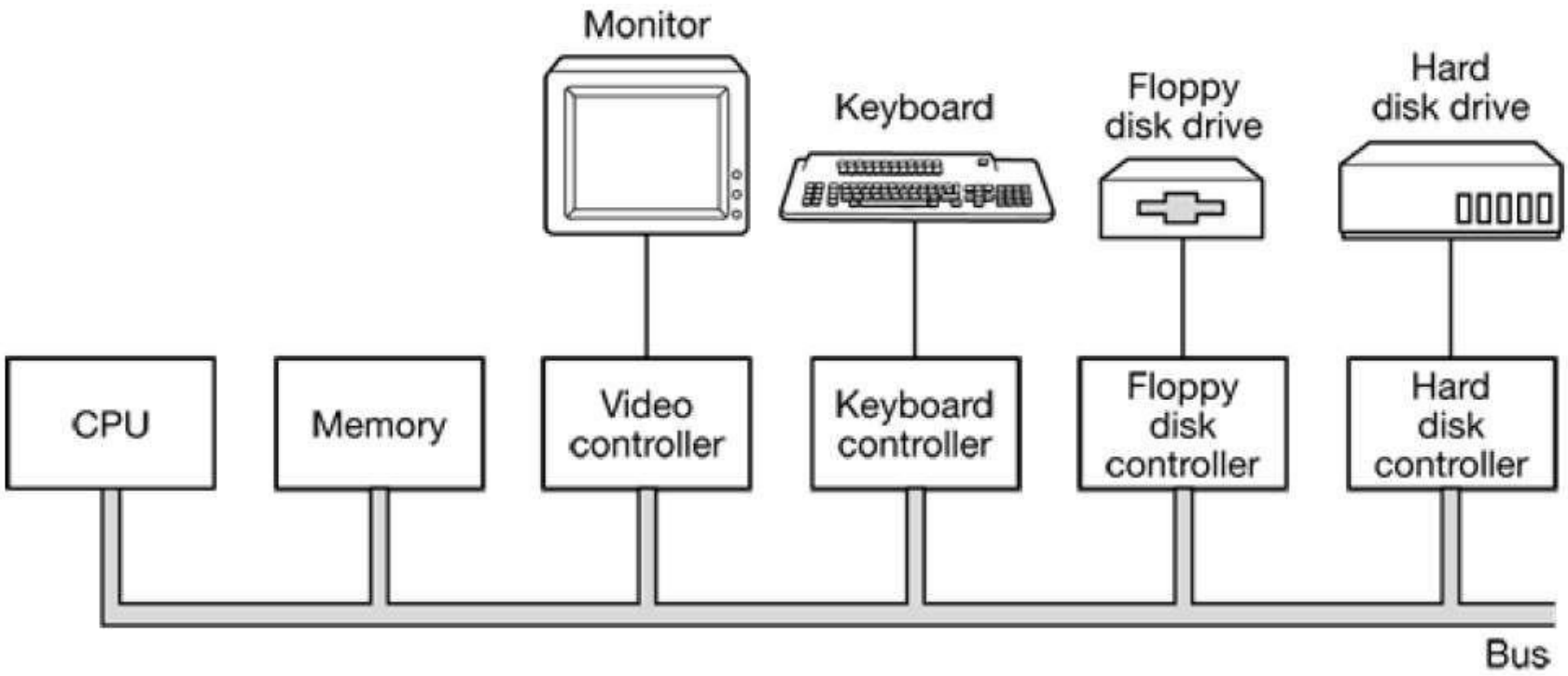
Components of I/O devices

- I/O devices have two components

1. **Mechanical component -** Device itself.
2. **Electronic Component-**

- Known as device controller or adapter .
- The controller card usually has a connector on it, into which a cable leading to the device itself can be plugged

Device Controller



Device Controller

- **Electronic component** which **controls the device**.
- It may handle multiple devices.
- There **may be more than one controller** per mechanical component (example: hard drive).
- Controller's tasks are:
  - It **converts serial bit stream to block of bytes**
  - **Perform error correction** if necessary
  - **Block of bytes is** first **assembled bit by bit** in buffer inside the controller
  - After verification, the block has been declared to be error free, and then it can be **copied to main memory**

Device Controller

- Each device controller **has a few registers** that are **used for communicating with the CPU**.
- By **writing into these registers**, the OS can command the device to **deliver data, accept data, switch itself on or off**, or perform some action.
- By **reading from these registers** OS can learn **what the device's status is**, whether it is prepared to accept a new command and so on.
- There are two ways to communicate with control registers and the device buffers:
  1. I/O Port
  2. Memory mapped I/O

Disk Architecture

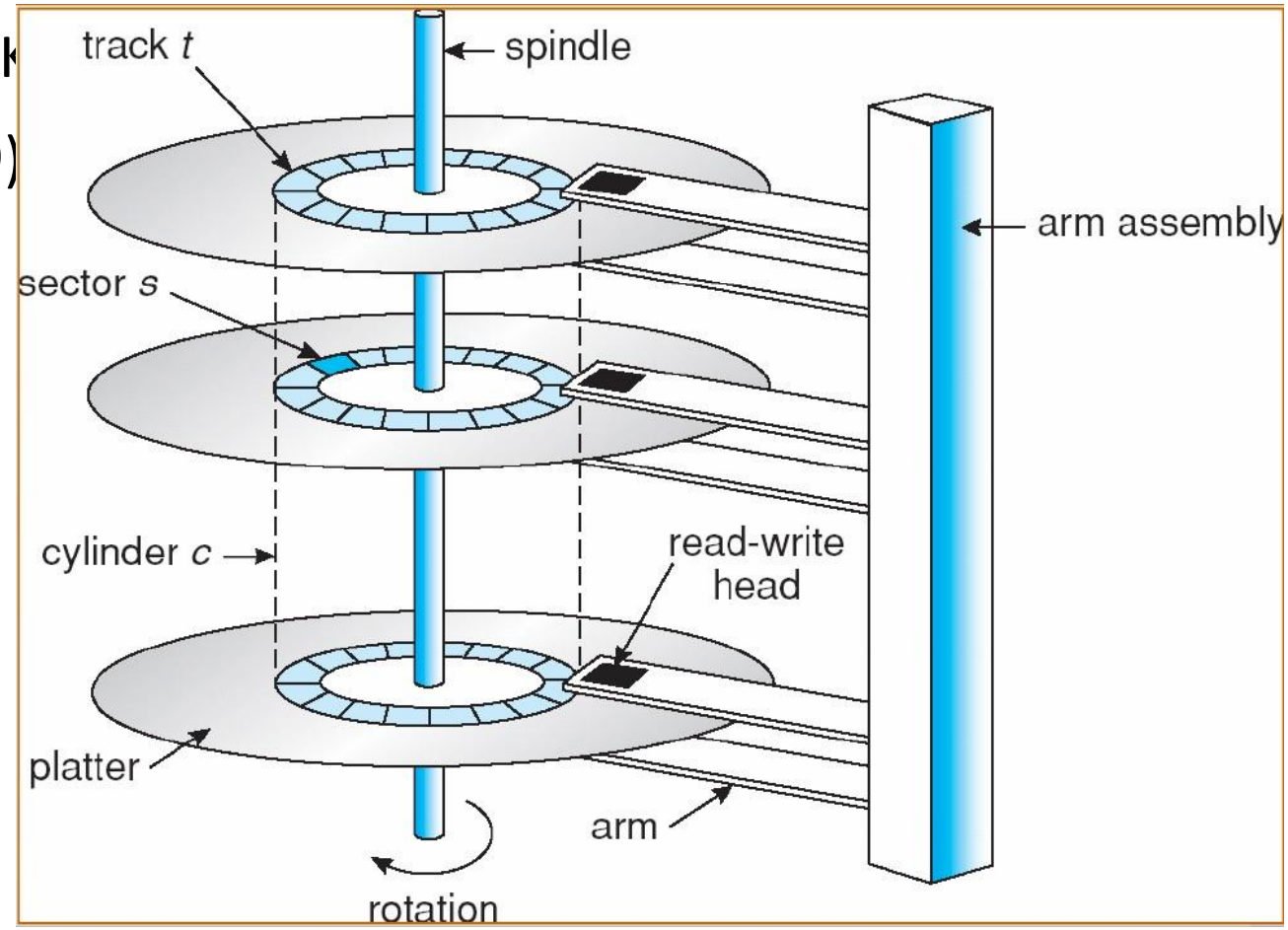
- **Platter → Surface → Track → Sector → Data .**

8*2*256*512*512K

2(3)*2(1)*2(8)*2(9)

2(40)

1TB



Disks

- It is relatively **permanent** and can hold **large quantities** of data. Its **access time is slow** compared with that of main memory and magnetic disk. provide the bulk of secondary storage for modern computer system.
- **disk platter:** has a flat circular shape, like a CD. The two surfaces of a platter are covered with a magnetic material. We store information by recording it magnetically on the platters.
- The surface of a platter is logically divided into circular **tracks** which are subdivided into **sectors**. The set of tracks that are at one arm position makes up a **cylinder**.

Disk Access time

- Seek time – the time taken by r/w head to reach desired track.
- Rotation time – the time for one full rotation (360◦).

How does CPU communicate with registers and buffers?

Marwadi University

First scheme

- If CPU want to read word,
- Puts read SIGNAL on control line
- Put address on address line
- A second signal line is used to tell weather I/O space or memory space is needed.

Memory mapped approach

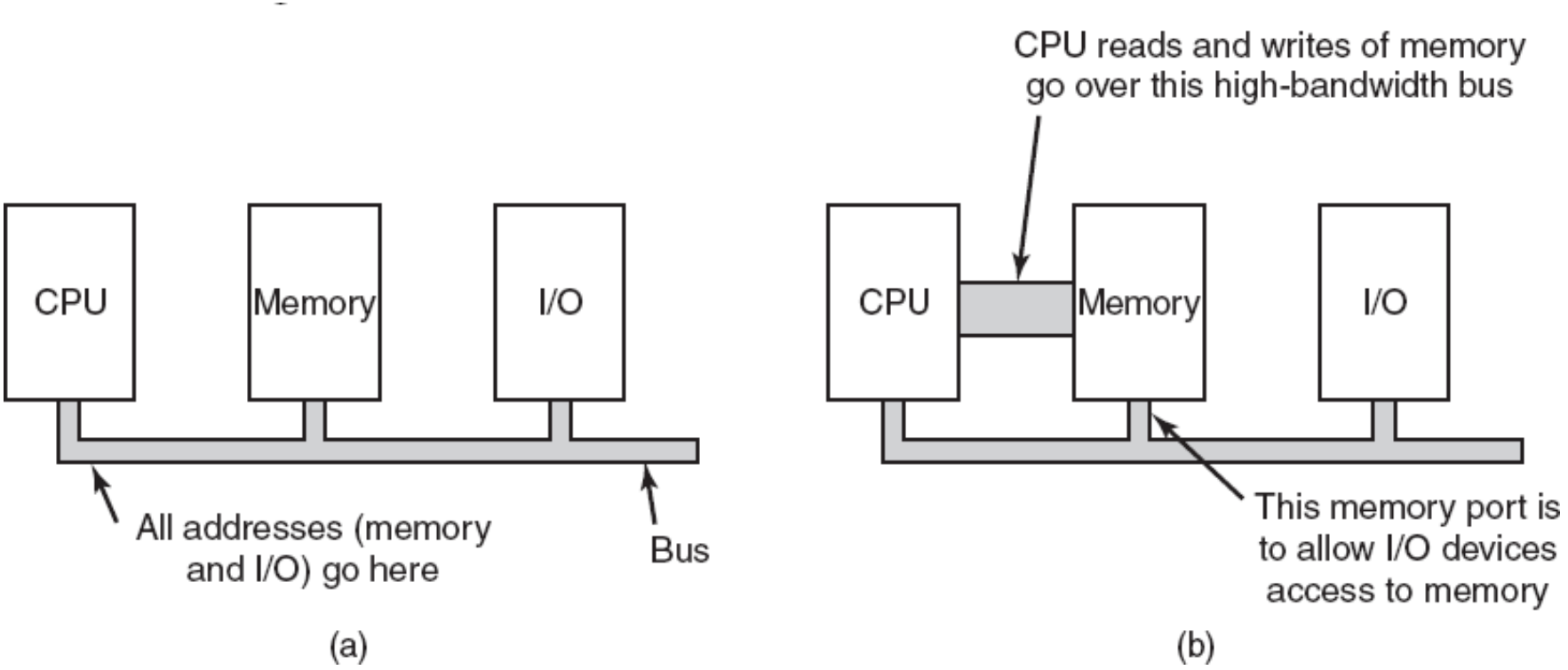
- If CPU want to read word,
- Puts read SIGNAL on control line
- Put address on address line and let memory module and I/O devices compare address with the ranges they serve.

Memory Mapped I/O

- **Advantages:**
- As the I/O space is mapped into memory so **no need of any special instructions** to read/write control register, can write a device driver in C
- Don’t need **special protection** to keep users from doing I/O directly. Just don’t put I/O memory in any user space.
- **Disadvantages:**
- If there is only one address space, then all memory modules and all I/O devices must examine all memory references to see which ones to respond to.
- solution: A dual-bus memory

Memory-Mapped I/O

- (a) A single-bus architecture. (b) solution: A dual-bus memory architecture.



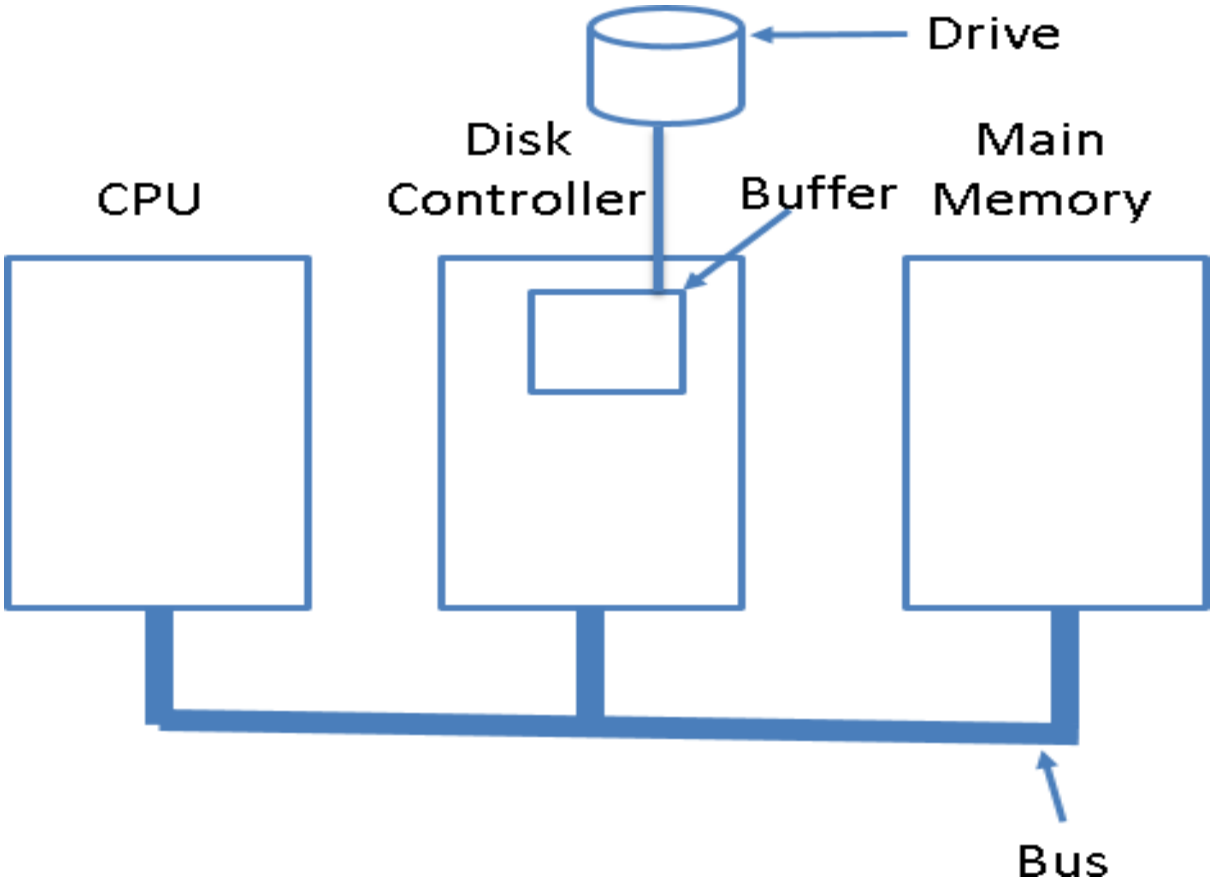
Direct Memory Access

- **CPU** could request **data one byte at a time** from **I/O controller**, Big waste of time. So other schema can be used DMA.
- **DMA controller can access system bus independent of CPU**. DMA contains serval registers that can be written and read by CPU.
- Memory address register
- Byte count register
- Control registers-I/O port, direction of transfer, transfer units (byte/word), number of bytes to transfer in a burst.

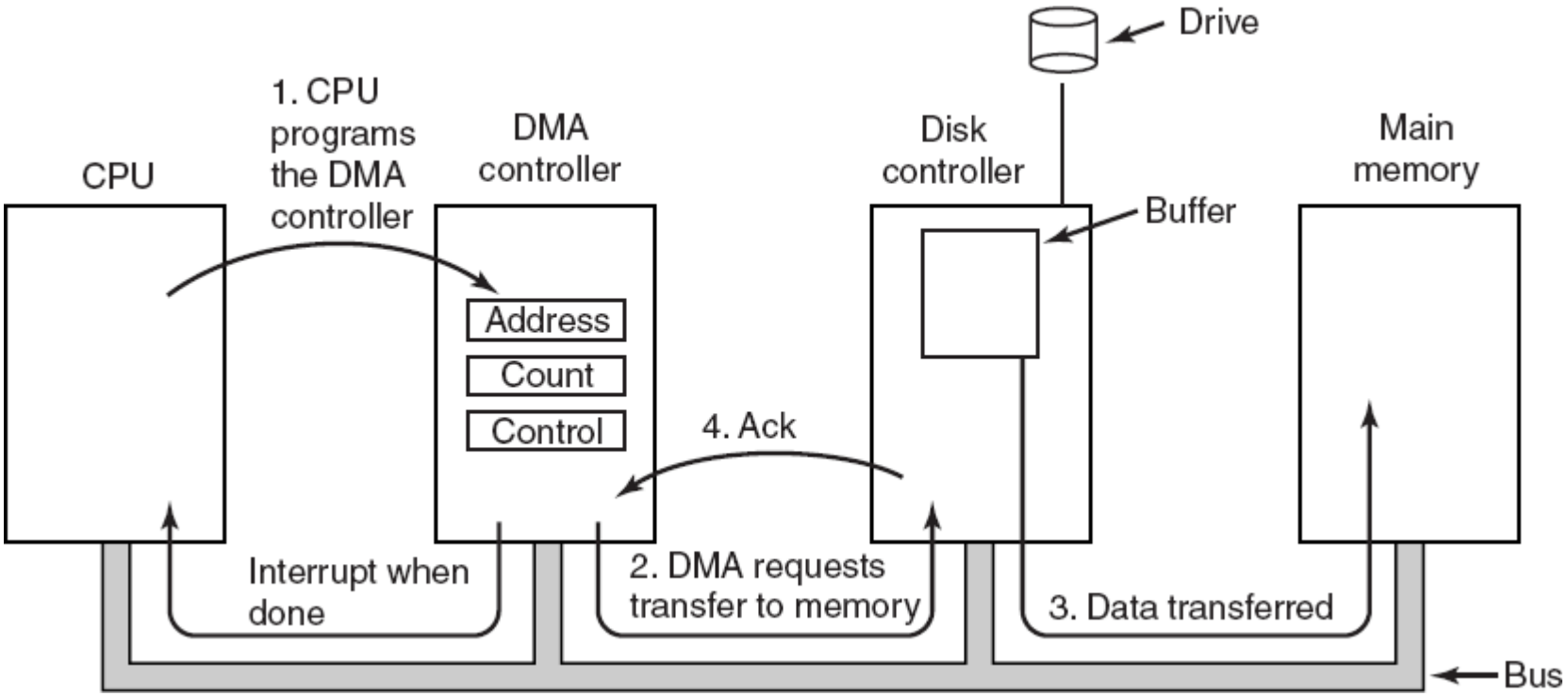
How Disk read occurs when DMA is not used??

- **First the controller reads the block** (one or more sectors) from the drive serially, bit by bit, until the **entire block is in the controller's internal buffer.**
- Next**, it computes** the **checksum to verify** that **no read errors have occurred.** Then the controller causes an interrupt.
- When the operating system starts running, it can read the disk block from the controller's buffer a byte or a word at a time by executing a loop, with each iteration reading one byte or word from a controller device register and storing it in main memory.

How Disk read occurs when DMA is not used??



How does DMA work?



How does DMA work?

Marwadi University

- **Step 1:** First the CPU programs the DMA controller by setting its registers so it knows what to transfer where.
- It also issues a command to the disk controller telling it to read data from the disk into its internal buffer and verify the checksum. When valid data are in the disk controller's buffer, DMA can begin.
- **Step 2:** The DMA controller initiates the transfer by issuing a read request over the bus to the disk controller.
- This read request looks like any other read request, and the disk controller does not know (or care) whether it came from the CPU or from a DMA controller.

How does DMA work?

Marwadi University

- Typically, the memory address to write to is on the bus' address lines, so when the disk controller fetches the next word from its internal buffer, it knows where to write it.
- **Step 3:** The write to memory is another standard bus cycle.
- **Step 4:** When the write is complete, the disk controller sends an acknowledgement signal to the DMA controller, also over the bus.
- The DMA controller then increments the memory address to use and decrements the byte count.
- If the byte count is still greater than 0, steps 2 to 4 are repeated until it reaches 0.

Modes of bus operation

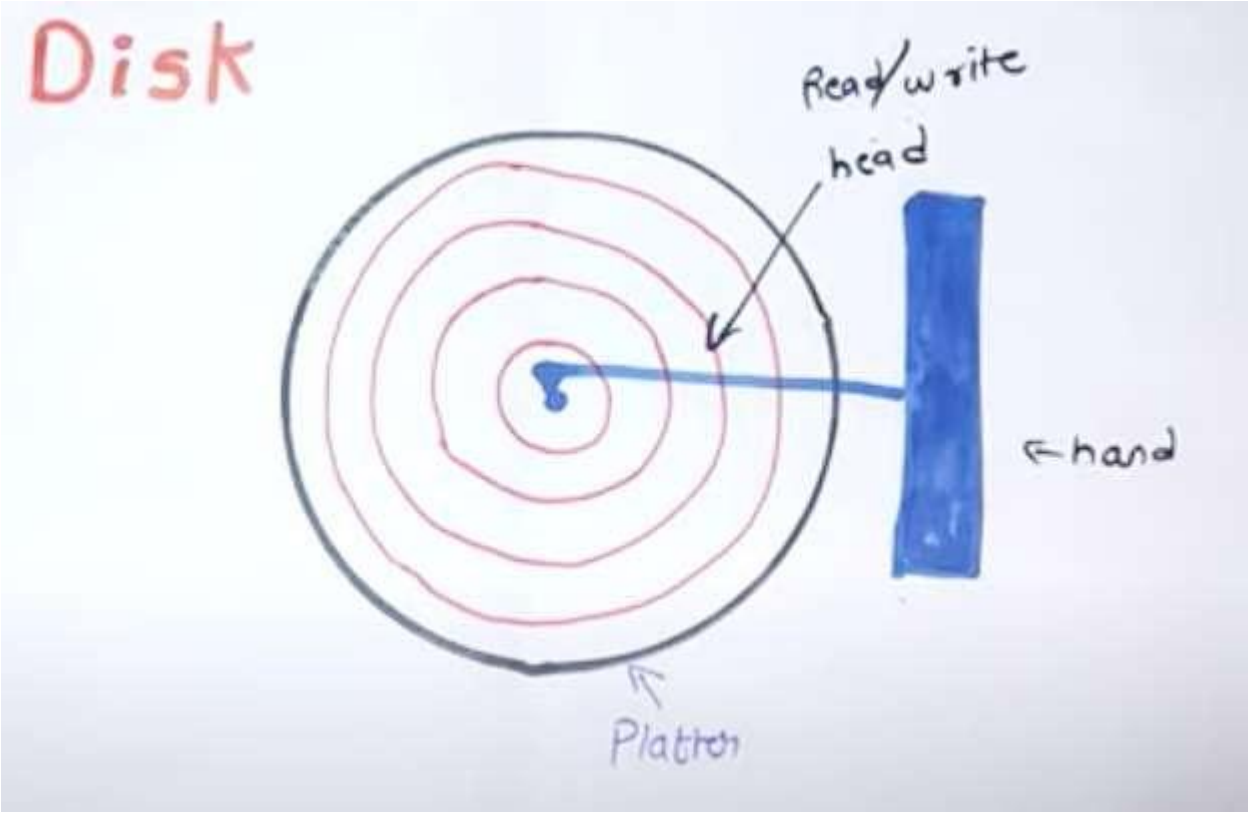
Marwadi University

- The buses can be operated in two modes
  1. **Word-at-a-time mode:** Here the DMA requests for the **transfer of one word** and gets it.
     - If CPU wants the bus at same time then it has to wait.
     - This mechanism is known as Cycle.
  2. **Block mode:** Here the DMA controller tells the device to acquire the bus, issues a **series of transfer** and then releases the bus.
     - This form of the operation is called Burst mode.
     - It is more efficient then cycle stealing.

Disk Arm Scheduling Algorithm

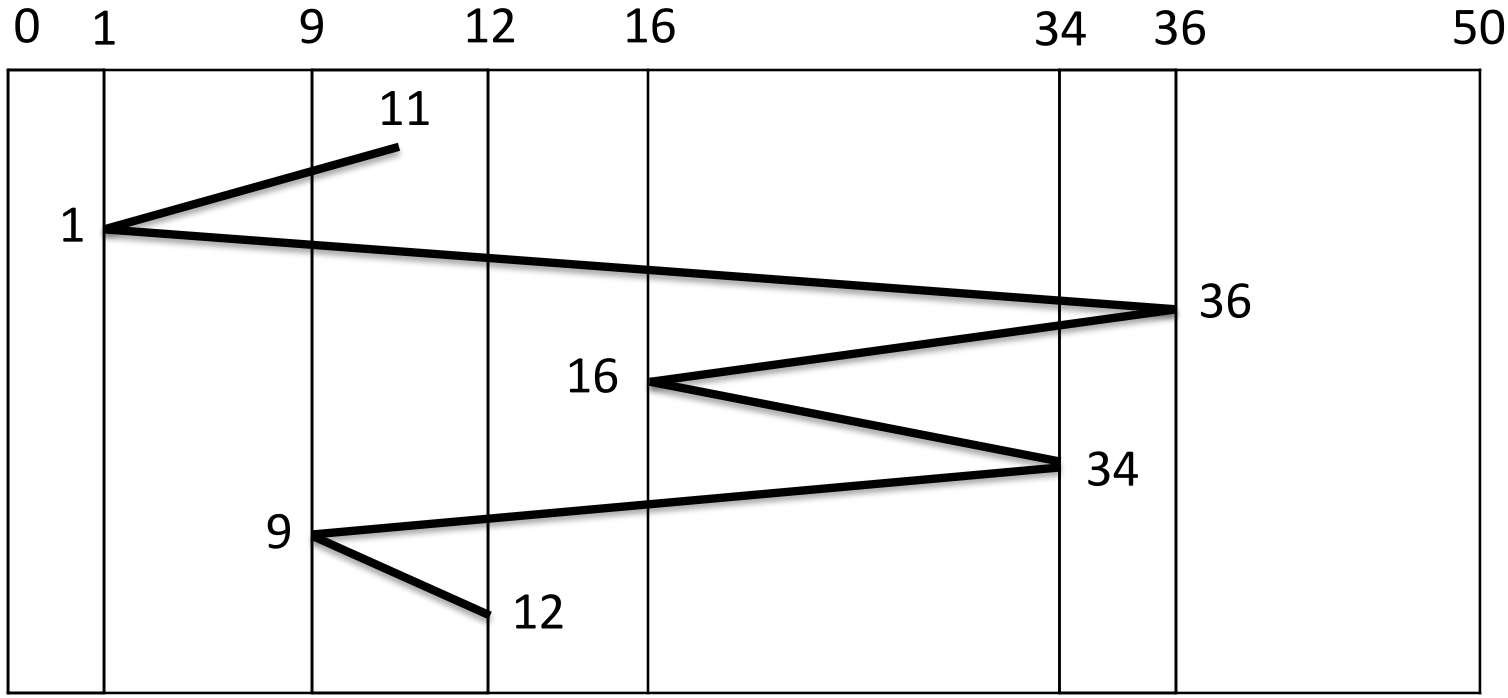
Marwadi University

- Various types of disk arm scheduling algorithms are available to decrease mean seek time.
  1. FCFS (First come first serve)
  2. SSTF (Shorted seek time first)
  3. SCAN
  4. C-SCAN
  5. LOOK (Elevator)
  6. C-LOOK


Disk
Read/write
head
arm
Platter

FCSC (First come first serve)

- Here **requests are served in the order of their arrival**.



- Disk movement will be 11, 1, 36, 16, 34, 9 and 12.
- Total cylinder movement: (11-1) + (36-1) + (36-16) + (34-16) + (34-9) + (12-9) = 111

Disk Scheduling Algorithm

Marwadi University

Q1 Track no given= 82, 170,43,140,24,16,190

- A disk contains 0-199 tracks
- Current position of r/w head is 50
- Calculate the total no of track movement of your r/w head?

Solution-

FCFS= and **SSTF=**

SCAN=

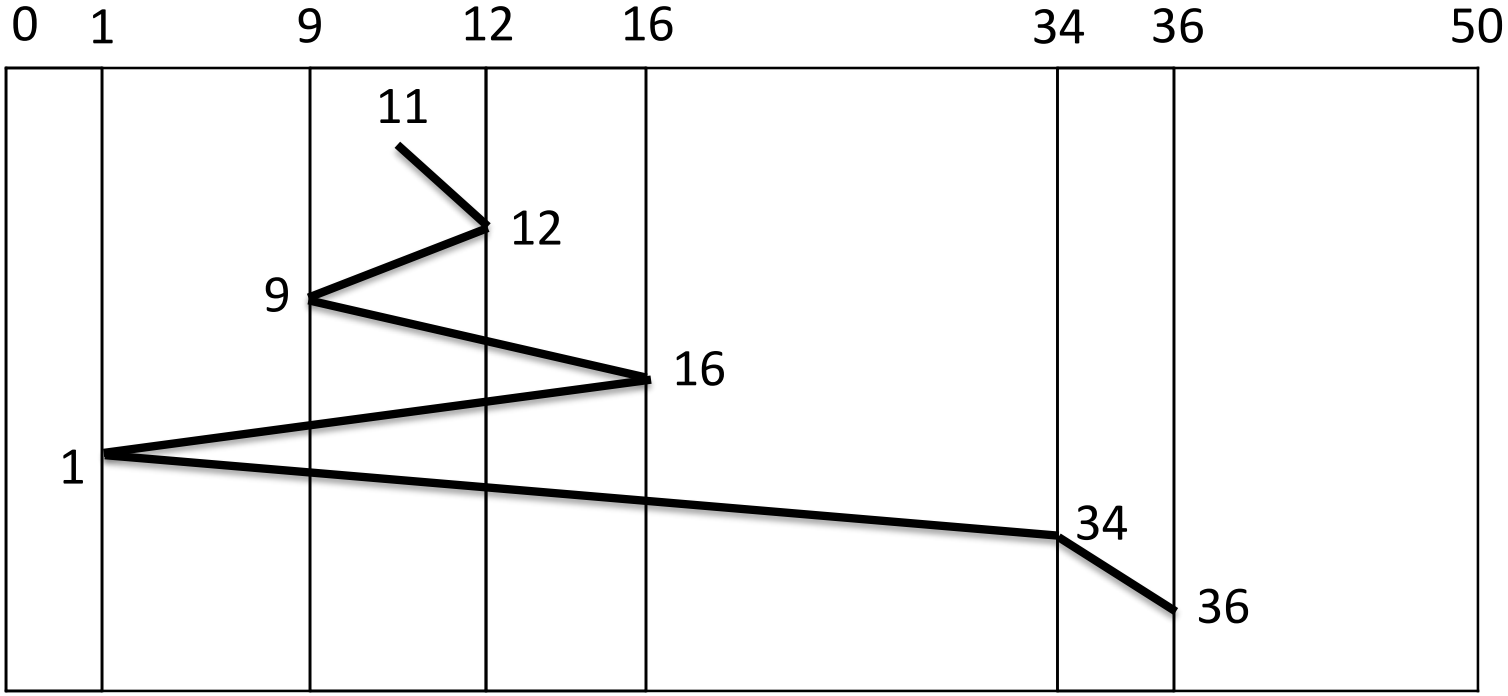
C-SCAN

LOOK

C-LOOK

SSTF (Shortest seek time first)

- We can minimize the disk movement by **serving the request closest to the current position** of the head.



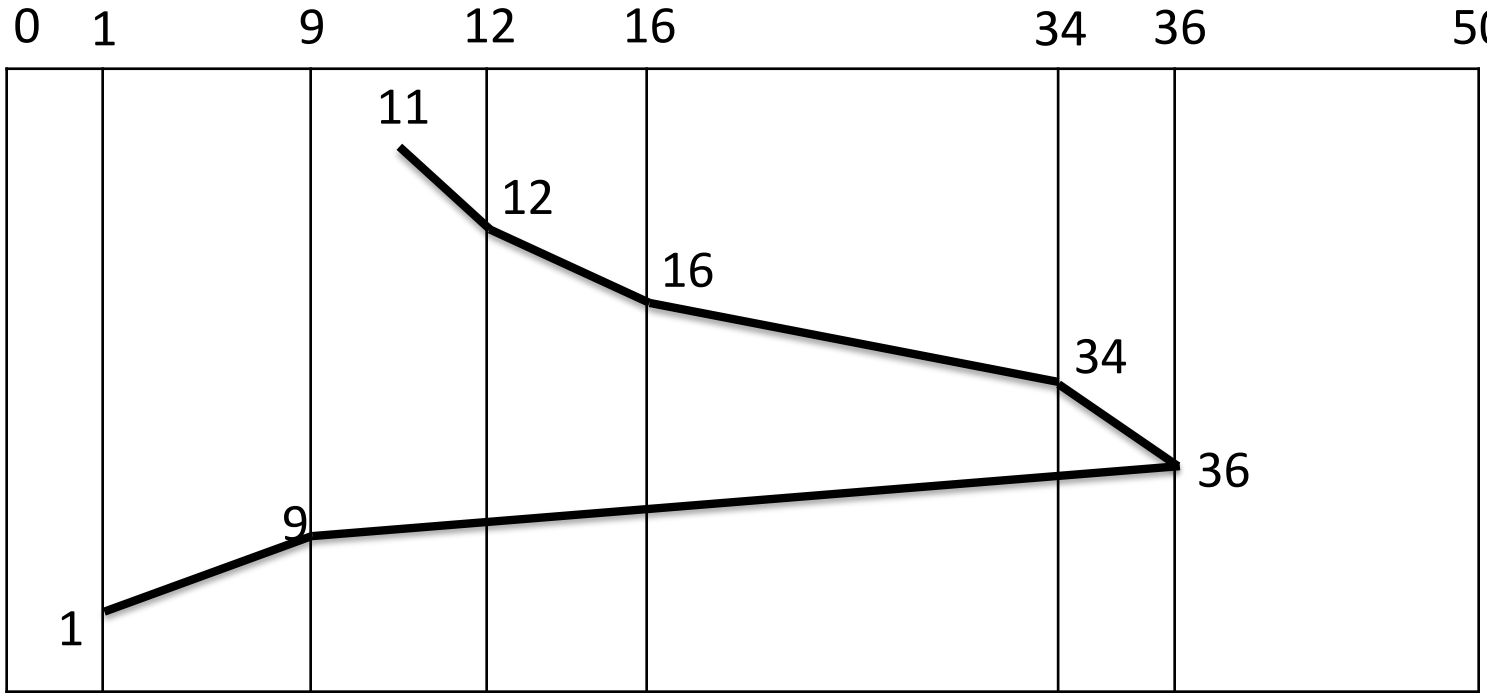
1, 36, 16, 34, 9, 12

- Disk movement will be 11, 12, 9, 16, 1, 34, 36.
- Total cylinder movement: (12-11) + (12-9) + (16-9) + (16-1) + (34-1) + (36-34) = 61

LOOK(Elevator)

- **Keep moving in the same direction until there are no more outstanding requests pending in that direction, then algorithm switches the direction**.
- After switching the direction the arm will move to handle any request on the way. Here **first go it moves in up direction then goes in down direction**.
- This is also called as **elevator algorithm**.
- In the elevator algorithm, the **software maintains 1 bit: the current direction bit, which takes the value either UP or DOWN**.

LOOK (Elevator)



1,
36,
16,
34,
9,
12

- Disk movement will be 11, 12, 16, 34, 36, 9, 1.
- Total cylinder movement: (12-11) + (16-12) + (34-16) + (36-34) + (36-9) + (9-1)=60

Disk Scheduling Algorithm

Marwadi University

Q1 Track no given= 82, 170,43,140,24,16,190

- A disk contains 0-199 tracks
- Current position of r/w head is 50 direction is towards larger value of track
- Calculate the total no of track movement of your r/w head?

Solution-

FCFS=642 and **SSTF=208**

SCAN=

C-SCAN

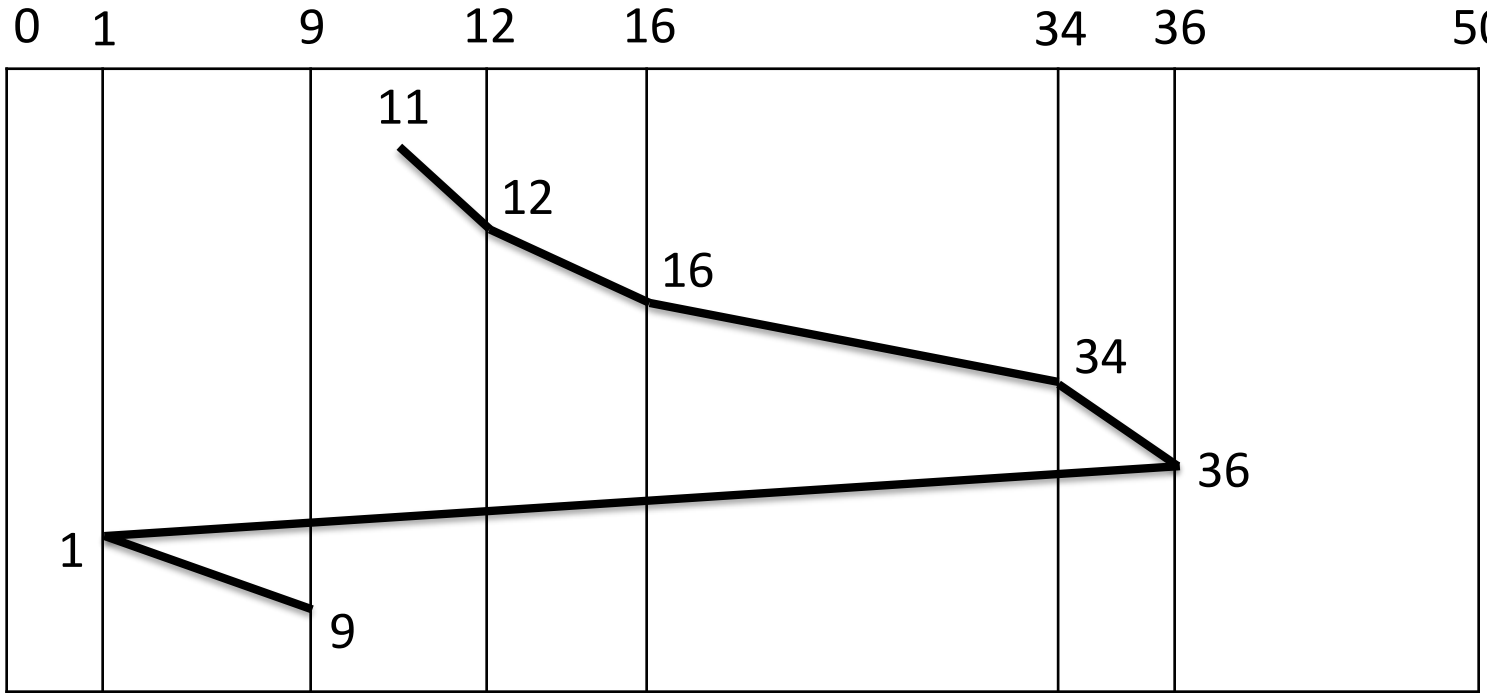
LOOK

C-LOOK

C-LOOK

- **Keep moving in the same direction until there are no more outstanding requests pending in that direction, then algorithm switches direction**.
- **When switching occurs the arm goes to the lowest numbered cylinder** with pending requests and **from there it continues moving in upward direction again**.

C-LOOK



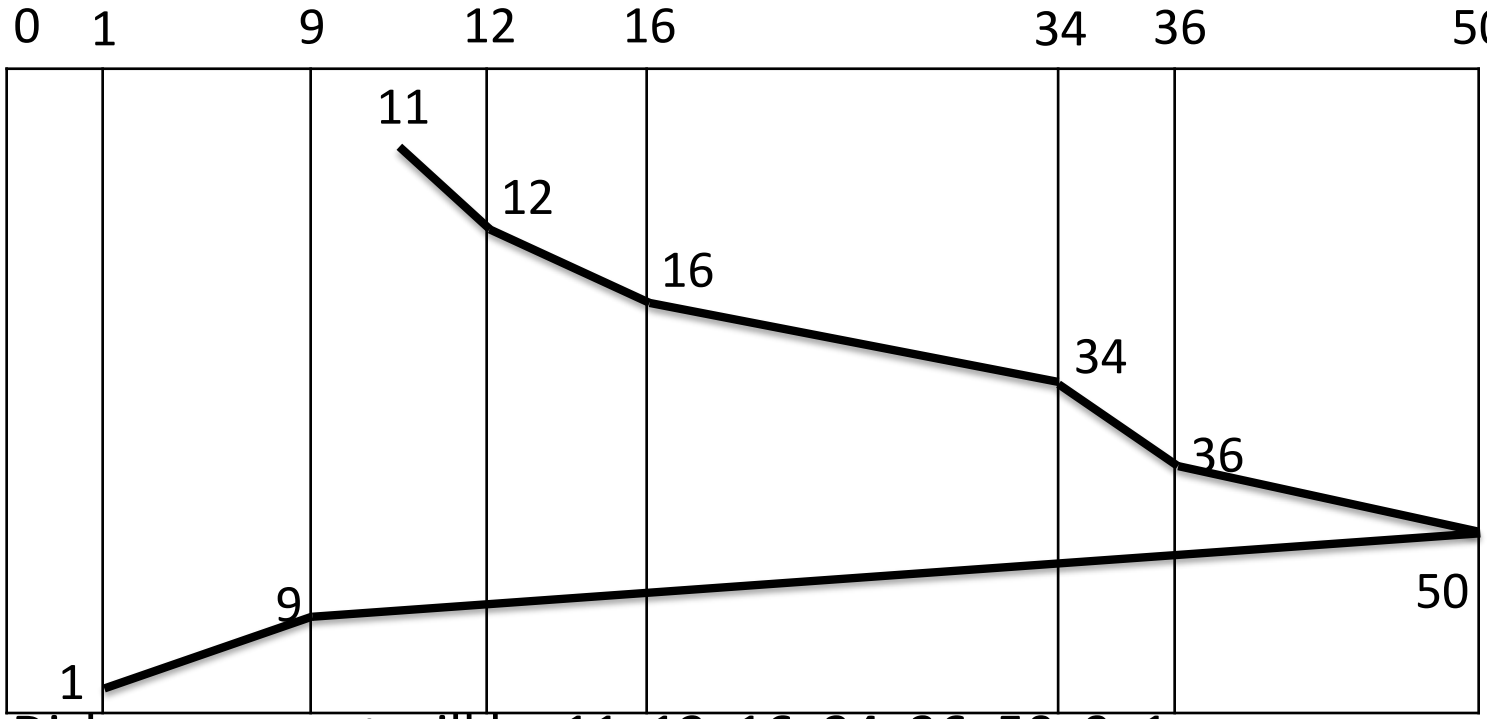
1,
36,
16,
34,
9,
12

- Disk movement will be 11, 12, 16, 34, 36, 1, 9.
- Total cylinder movement: (12-11) + (16-12) + (34-16) + (36-34) +(36-1)+(9-1)=68

SCAN

- **From the current position disk arm starts in up direction and moves towards the end, serving all the pending requests until end**.
- At that end arm **direction is reversed (down) and moves towards the other end serving the pending requests on the way**.

SCAN



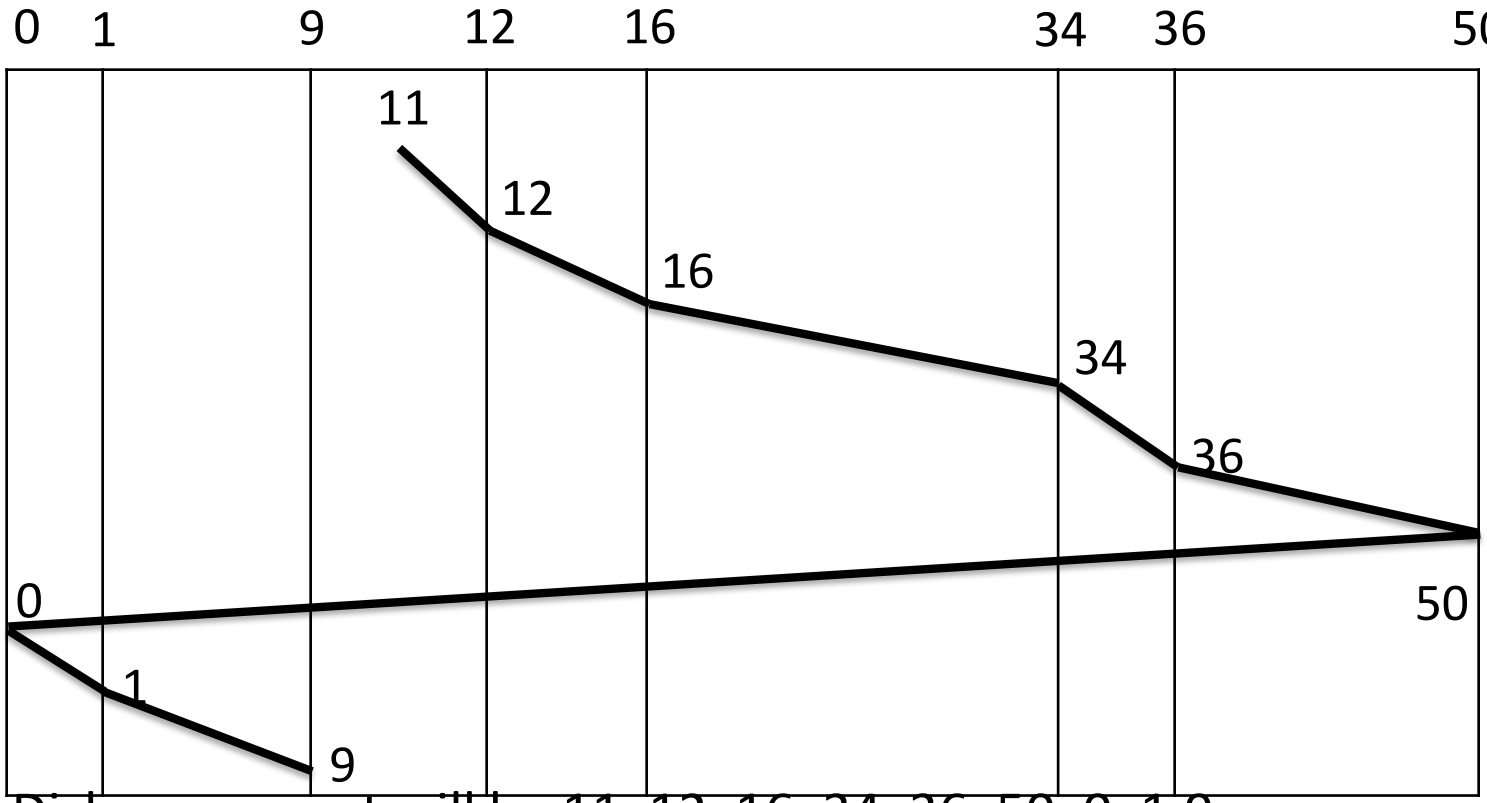
1,
36,
16,
34,
9,
12

- Disk movement will be 11, 12, 16, 34, 36, 50, 9, 1.
- Total cylinder movement: (12-11) + (16-12) + (34-16) +(36-34) +(50-36) + (50-9) + (9-1) = 88

C-SCAN

- From the **current position disk arm starts in up direction and moves towards the end, serving request until end**.
- At the end the arm **direction is reversed (down), and arm directly goes to other end and again continues moving in upward direction**.

C-SCAN



1,
36,
16,
34,
9,
12

- Disk movement will be 11, 12, 16, 34, 36, 50, 0, 1,9.
- Total cylinder movement: (12-11) + (16-12) + (34-16) +(36-34) +(50-36) + (50-0) + (1-0)+ (9-1) = 98

Disk Scheduling Algorithm

Marwadi University

Q- Track no given= 55,58,39,18,90,160,150,38,184

- A disk contains 0-199 tracks
- Current position of r/w head is 100 and direction is towards larger value of track
- Calculate the total no of track movement of your r/w head for FCFS, SSTF, SCAN,C-SCAN, LOOK AND C-LOOK?

Disk Scheduling Algorithm

- Solution

FCFS=498

SSTF=248

SCAN= 280

C-SCAN=388

LOOK=250

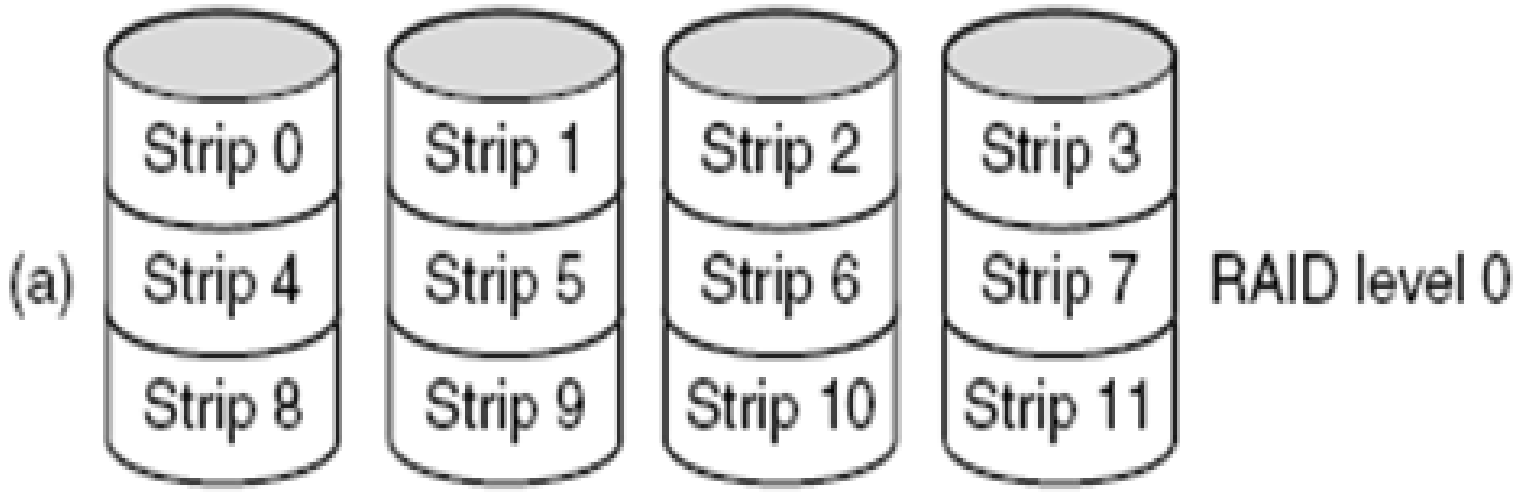
C-LOOK=322

(RAID – Redundant Array of Independent Disks)

- Parallel processing is being used more and more to speed up CPU performance, parallel I/O can be a good idea.
- The basic idea behind RAID is to install a box full of disks next to the computer, replace the disk controller card with RAID controller.
- All RAID have the property that the data are distributed over drives, to allow parallel operation.

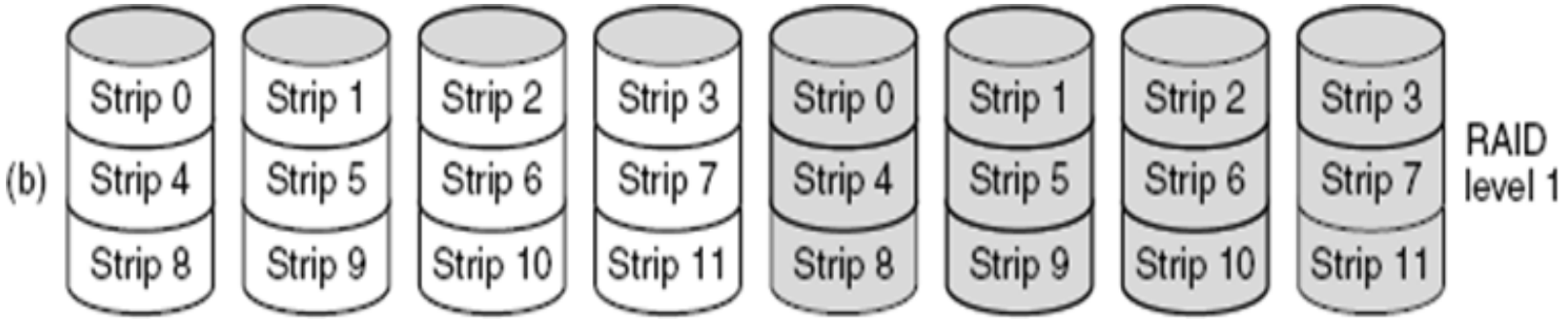
RAID level 0

- It consists of viewing the virtual single disk, as being divided up into strips of k sectors, and distributing data over multiple drives as shown in the figure is called striping.
- The RAID level 0 organization writes consecutive stripes over the drives in round-robin fashion.
- Command to read data block, consisting of four consecutive strips, RAID controller will break, this command up into four separate commands, one for each of the four disks and have them to operate in parallel.
- Thus we have parallel I/O without the software knowing about it.

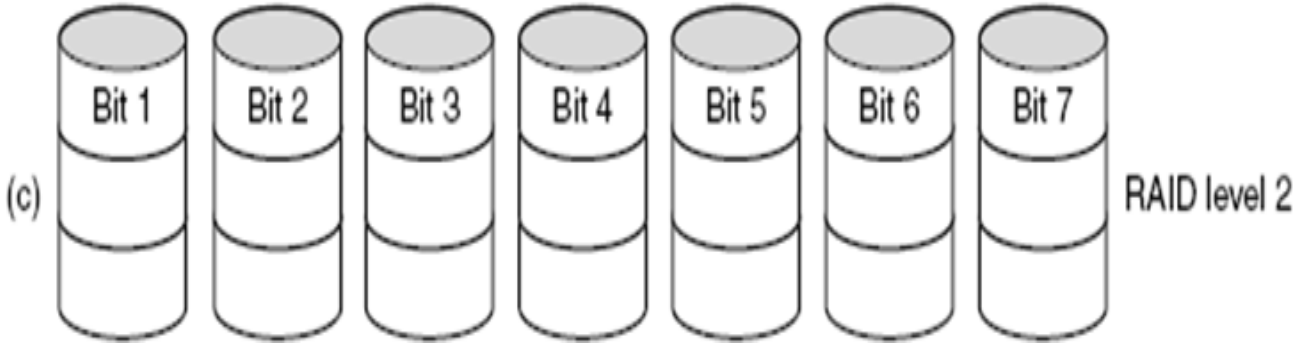


- One disadvantage of RAID level 0 is that the reliability is potentially worse than having SLED (**S**ingle **L**arge **E**xpensive **D**isk).

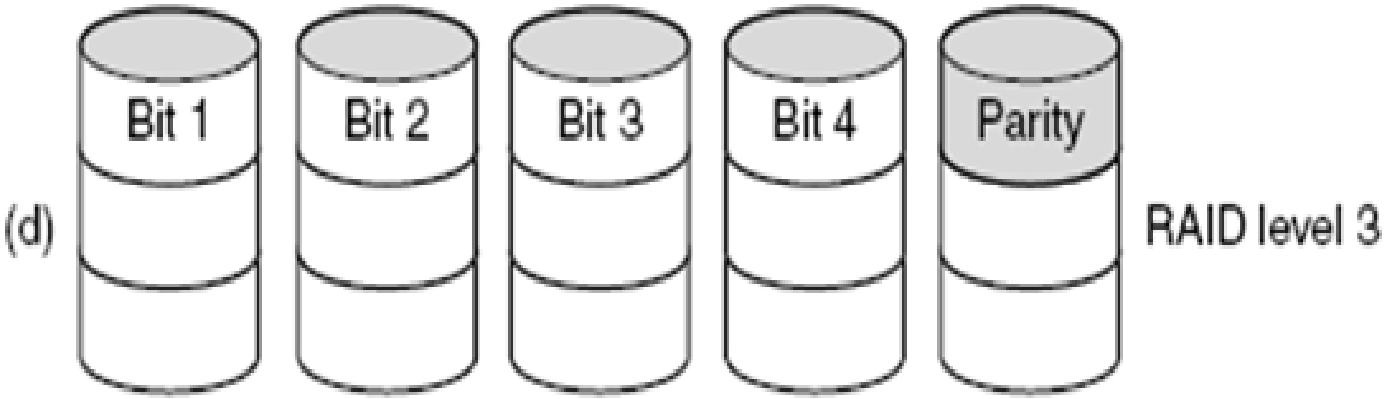
RAID level 1



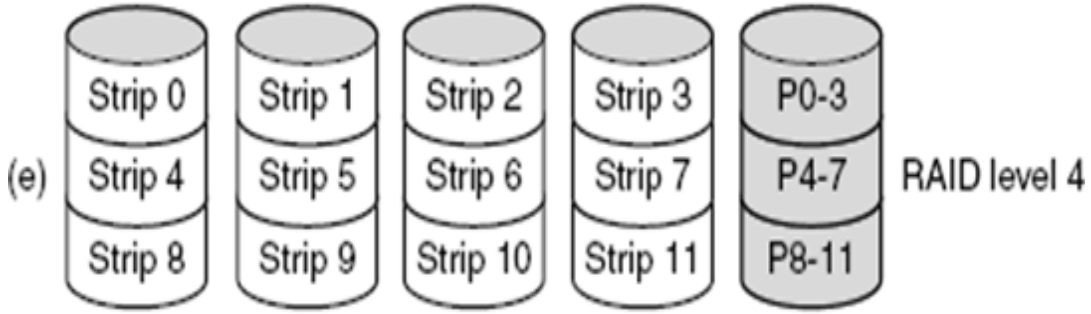
- RAID level 1 is a true RAID.
- It duplicates all the disks, so in figure there are four primary disks and four backup disks.
- On a write operation every strip is written twice.
- On a read operation either copy can be used, distributing the load over more drives.
- Write performance is no better than for a single drive.
- Read performance is twice as good.
- Fault tolerance is excellent: if a drive crashes, a copy is simply used instead.
- Recovery consists of simply installing a new drive and copying the entire backup drive to it.



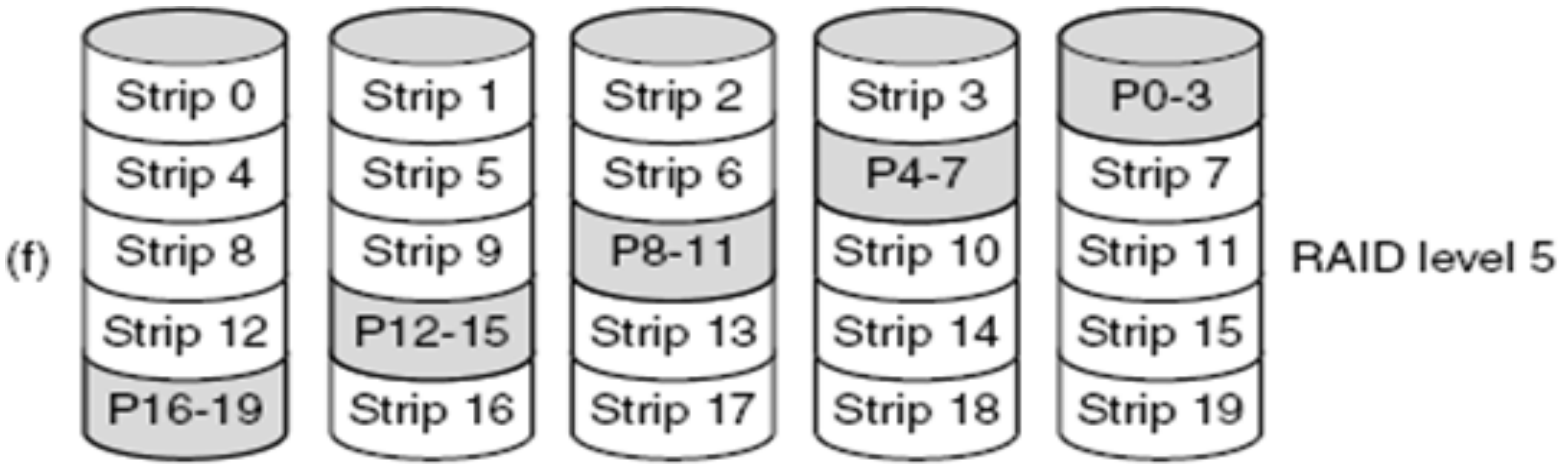
- RAID level 2 works on word, possibly even a byte basis.
- Imagine splitting each byte into a pair of 4-bit nibbles, then adding Hamming code toeach one to form a 7-bit word, of which bit 1, 2 and 4 were parity bits, as shown in figure below.
- In this RAID level 2 each of seven drives needs synchronized in terms of arm position and rotational position, and then it would be possible to write the 7-bit Hamming coded word over the seven drives, one bit per drive.
- Here, losing one drive did not cause problem, which can be handled by Hamming code on the fly.
- But, on the down side this scheme requires all the drives to be rotationally synchronized and it also asks a lot of controller, since it must do a Hamming checksum every bit time.



- RAID level 3 is simplified version of RAID level 2, as shown in the above figure.
- Here single parity bit is computed for each data word and written to a parity drive.
- As in RAID level 2 the drives must be exactly synchronized.
- In case of drive crashing, it provides 1-bit error correction since position of each bad bit is known.



- RAID level 4 works with strips and not individual word with parity.
- They do not require synchronization of drives.
- As shown in the figure RAID level 4 is like RAID level 0, with strip-for-strip parity written onto an extra drive, for example, if each strip is k bytes long, all strips are EXCLUSIVE ORed together, resulting in a parity strip k bytes long.
- If a drive crashes, the lost bytes can be recomputed from the parity drive by reading the entire set of drives.
- This design protects against the loss of a drive but performs poorly for small updates, if one sector is changed, it is necessary to read all the drives in order to recalculate the parity.
- It creates heavy load on parity drive.



- As with RAID level 4, there is a heavy load in the parity drive, it may become bottleneck.
- This bottleneck can be eliminated in RAID level 5 by distributing the parity bits uniformly over all the drives, round robin fashion, as shown in the above figure.
- In the event of drive crash, reconstructing the contents of the failed drive is complex process.


THANK
YOU