


Unit 6 – I/O Device Management





Introduction

- A function of OS
- Responsible for managing all the hardware devices of the computer system
- Both storage devices and I/O devices
- Device Management generally performs:
 - Installing devices, component drivers and related software dependencies
 - Configuring the device to perform at its peak
 - Implementing security measures and processes



Classification of I/O devices

- The I/O units which consist mechanical components are called I/O devices e.g. hard-disk drive, printer etc.
- There are two types of devices
 - Block devices or Machine readable
 - Character devices or User readable



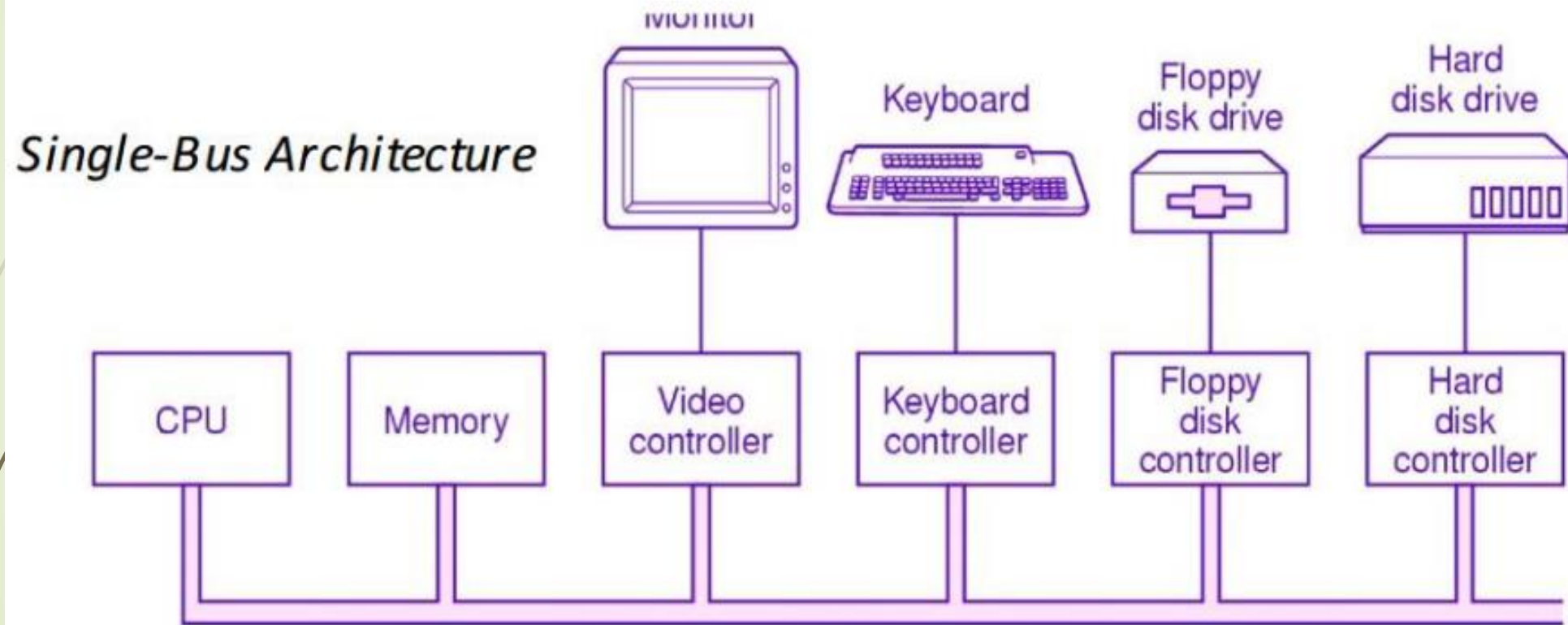
Classification of I/O devices (contd.)

- Block devices or Machine readable
 - Stores information in fixed-sized blocks, each one with its own address
 - A block device is one with which the driver communicates by sending entire blocks of data.
 - Read or write is possible independent to other blocks-direct access
 - Example: Hard disk, USB, tape etc.
- Character devices or User readable
 - Delivers or accepts a stream of characters without regard to any block structure
 - It is not addressable and does not have any seek operation
 - Example: keyboards, mice, terminals, line printers, network interfaces, and most other devices that are not disk-like.

Device Controllers

- Device drivers are software modules that can be plugged into an OS to handle a particular device. Operating system takes help from device driver to handle I/O devices.
- A controller is a collection of electronics that can operate a bus or a device
- Controller works like an interface between a device and a device driver. I/O units typically consist of a mechanical and electronic component where electronic component is called the device controller.
- A single controller can handle multiple devices; some devices have their own built-in controller
- The controller has one or more registers for data and signals
- The processor communicates with the controller by reading and writing bit patterns in these registers

Device Controllers (contd.)





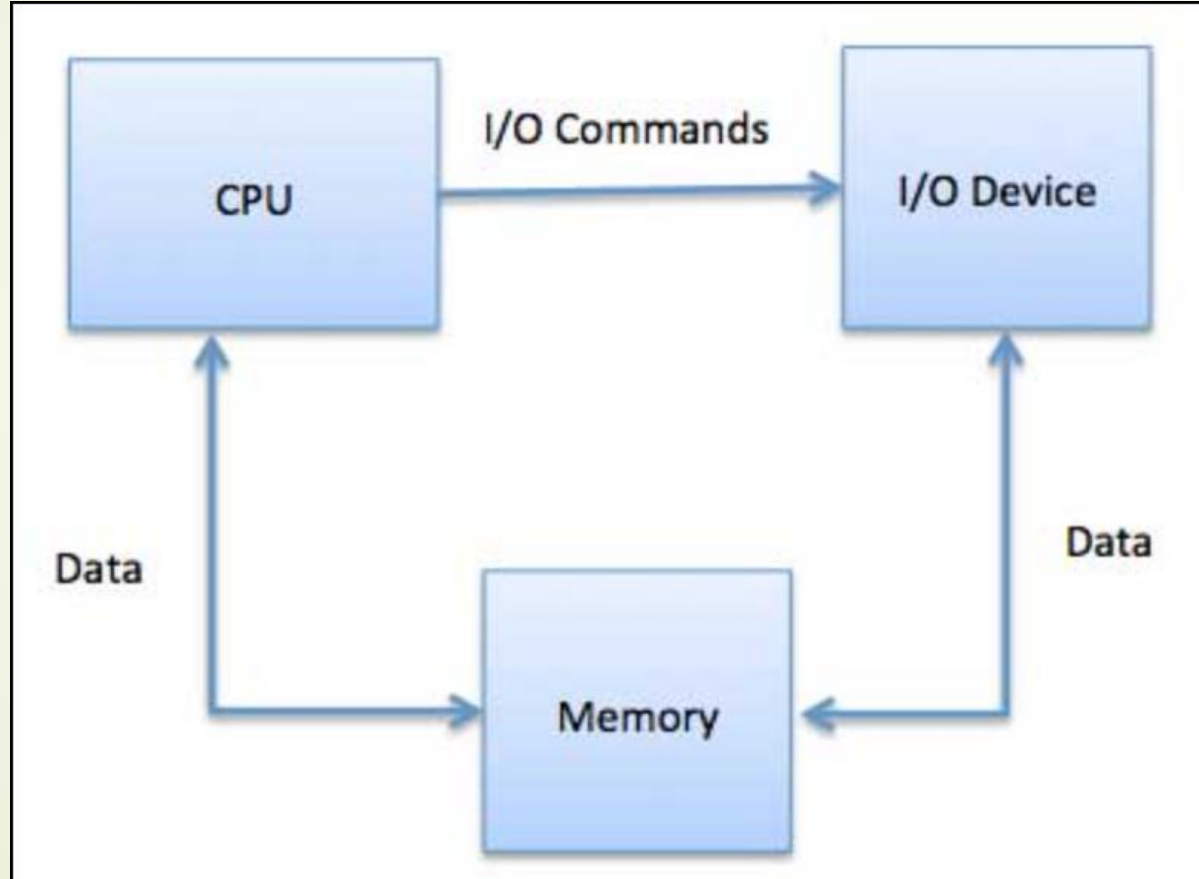
Communication of CPU to I/O Devices

- The CPU must have a way to pass information to and from an I/O device.
- There are three approaches available to communicate with the CPU and Device.
 1. **Special Instruction I/O:** These instructions typically allow data to be sent to an I/O device or read from an I/O device
 2. **Memory-mapped I/O**
 3. **Direct memory access (DMA)**

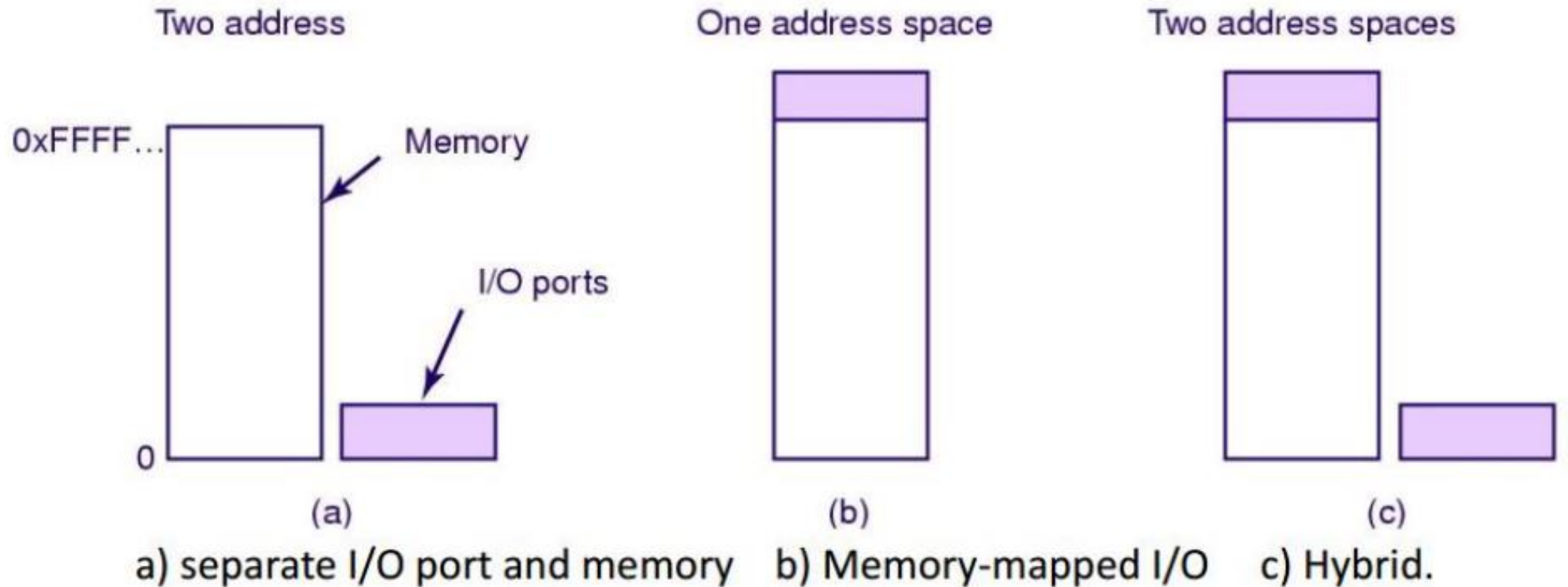
Memory Mapped I/O

- The device is connected directly to certain main memory locations so that I/O device can transfer block of data to/from memory without going through CPU.
- Device controller have their own register (Control Register) and Data Buffer for communicating with the CPU
- When using memory-mapped I/O, the same address space is shared by memory and I/O devices.
- The device is connected directly to certain main memory locations so that I/O device can transfer block of data to/from memory without going through CPU.
- While using memory mapped IO, OS allocates buffer in memory and informs I/O device to use that buffer to send data to the CPU.
- I/O device operates asynchronously with CPU, interrupts CPU when finished.
- Each control register is assigned a unique memory address to which no memory is assigned
- The device control registers are mapped into memory space, called memory I/O mapped.
- OS allocates buffer in memory and informs I/O device to use that buffer to send the data to the CPU
- It is used for high-speed I/O devices
- CPU communicated with control register and data buffer:
 - Using I/O ports
 - Using Memory mapped I/O

Memory Mapped I/O (contd.)



Memory Mapped I/O (contd.)

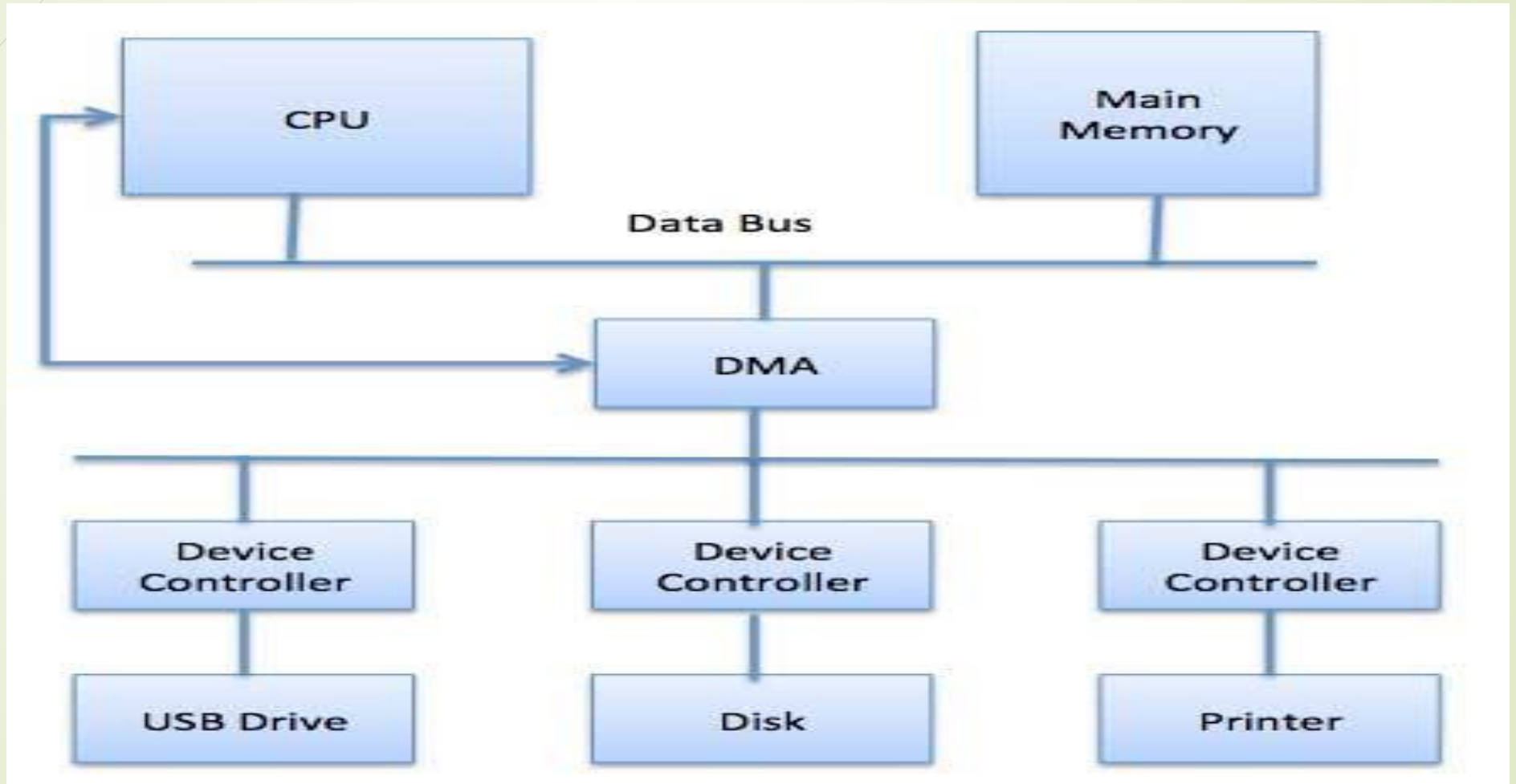




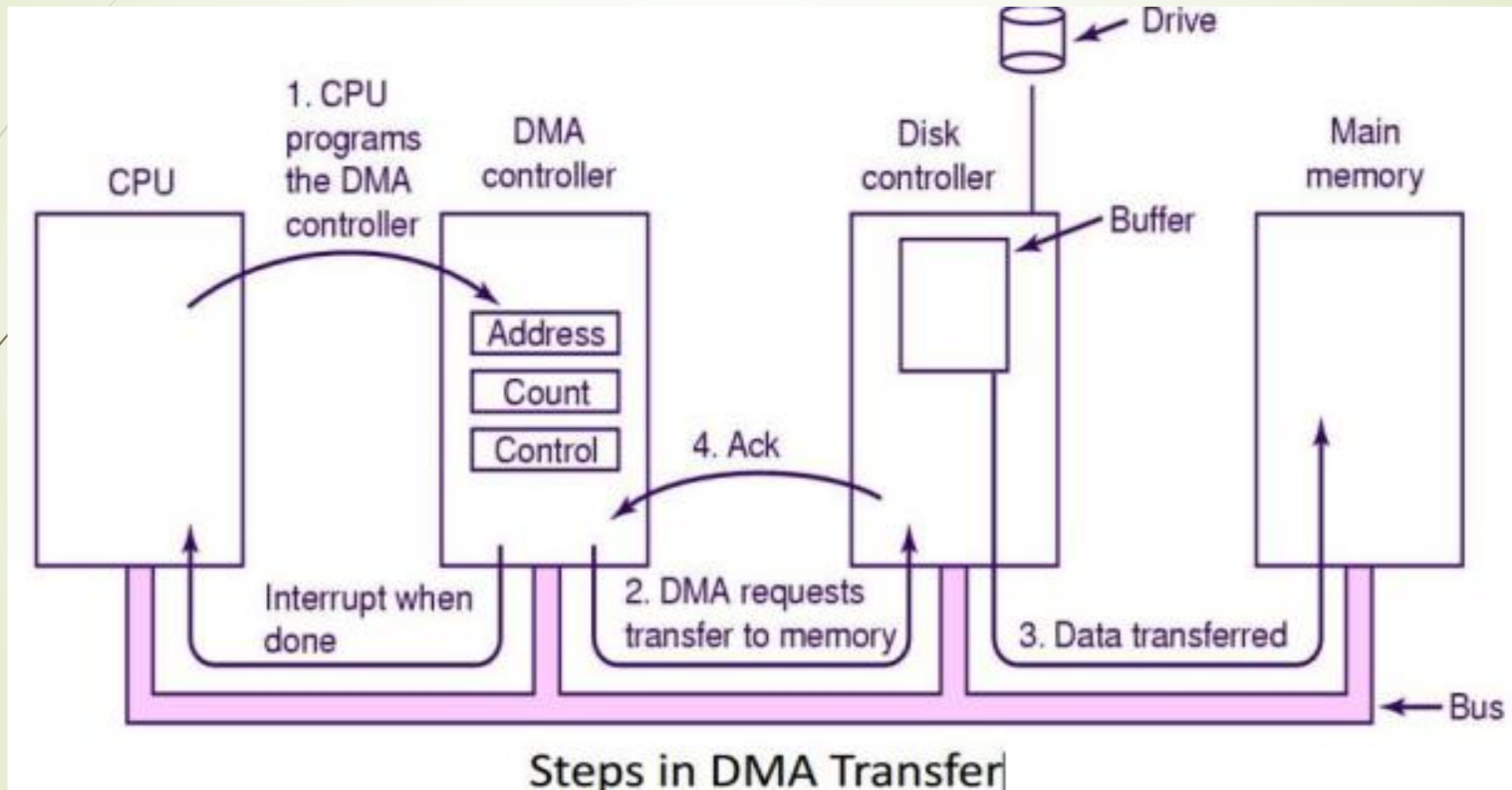
DMA(Direct Memory Access)

- ▶ Slow device like keyboards will generate an interrupt to the CPU each byte is transferred .
- ▶ If a fast device such as disk generated an interrupt for each byte, the OS would spend most of time for handling these interrupts. So to reduce this problem computer system use DMA.
- ▶ DMA is the method of transferring data from the computer's ram to another part of the computer without processing it using the CPU.
- ▶ Direct Memory Access (DMA) means CPU grants I/O module authority to read from or write to memory without involvement.
- ▶ DMA module itself controls exchange of data between main memory and the input/out device .
- ▶ CPU only involves at beginning and end of the transfer and interrupted only after entire block has been transferred.

DMA(Direct Memory Access)



DMA Operation





DMA Operation

1. The CPU programs the DMA controller by 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 to its internal buffer and verify the checksum. When valid data are in the disk's controller buffer, DMA can begin.
2. The DMA controller initiates the transfer by issuing a read request over the bus to the disk controller.
3. Data transferred from disk controller to memory.
4. When transfer is completed, the disk controller sends an acknowledgment signal to the DMA controller. The DMA controller then increments the memory address to use and decrements the byte count. This continues until the byte count is greater than 0.
5. When transfer is completed, the DMA controller interrupts the CPU.

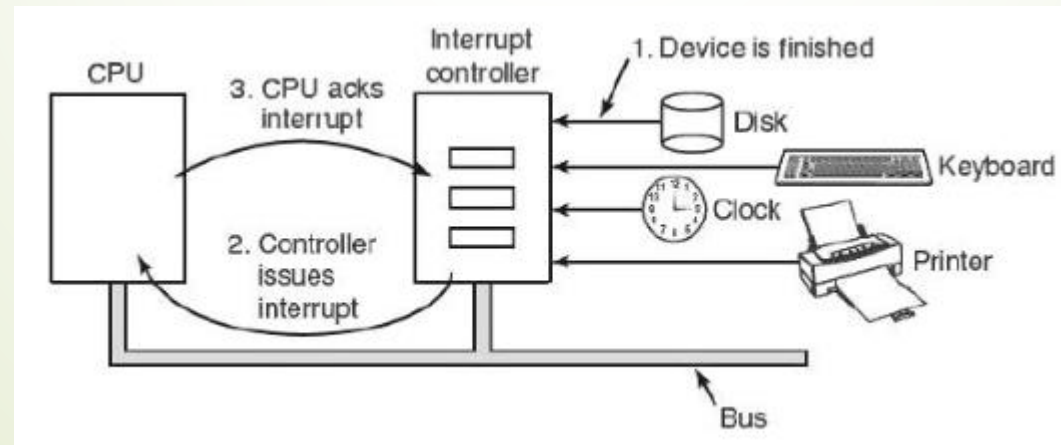


Interrupt

- An interrupt is a signal emitted by hardware or software when a process or an event needs immediate attention.
- It alerts the processor to a high-priority process requiring interruption of the current working process.
- Interrupt force to stop CPU what it is doing and start something else.
- Signal tell the CPU to stop its current activities and execute the appropriate part of the OS.
- There are three types of interrupts
 - **Software Interrupts:** They are generated by software conditions to request specific services from the operating system.
 - They are generated by program when they want to request a system call to be performed by the operating system.
 - **Hardware Interrupts:** Hardware interrupts are generated by external hardware devices to request attention from the CPU.

Interrupt

- **Trap:** Traps are generated by the CPU itself to indicate that some error or condition occurred for which assistance from the operating system is needed.
- Example: System Calls; System calls are a common use case for trap interrupts.
- **Working mechanism of Interrupt:**
 - When an i/o device has finished the work given to it, it cause an Interrupt.
 - It does this by asserting a signal on a bus, that is has ben assigned.
 - The signal detected by the interrupt controller then decides what to do? If no other interrupt are pending the interrupt controller processes the interrupt immediately, if another interrupt is in progress , then it is ignore for a moment.



Polling

- state of continuous monitoring
- microcontroller keeps checking the status of other devices
- It does no other operation and consumes all its processing time for monitoring.
- Problem Can be addressed by interrupt.
- The interrupt method, the controller responds only when an interruption occurs. Thus, the controller is not required to regularly monitor the status

Goal of I/O Software

- A key concept in the design of I/O software is known as device independence.
- I/O devices should be accessible to programs without specifying the device in advance.
- Some of the major goal of I/O software are:
 - 1) Device independence
 - 2) Uniform naming
 - 3) Error handling
 - 4) Synchronous Vs Asynchronous Transfer
 - 5) Buffering
 - 6) Sharable and dedicated Device

Handling I/O

There are three fundamentally different ways of performing I/O operations which are listed below.

- 1) Programmed I/O
- 2) Interrupted Driven I/O
- 3) I/O using DMA

Handling I/O

1) Programmed I/O

- one of the three fundamentally different ways that I/O can be performed.
- most simple type of I/O technique for the exchanges of data or any types of communication between the processor and the external devices.
- Data are exchanged between the processor and the I/O module.
- The processor executes a program that gives it direct control of the I/O operation, including sensing device status, sending a read or write command, and transferring the data.
- When the processor issues a command to the I/O module, it must wait until the I/O operation is complete.
- If the processor is faster than the I/O module, this is wasteful of processor time.

Handling I/O

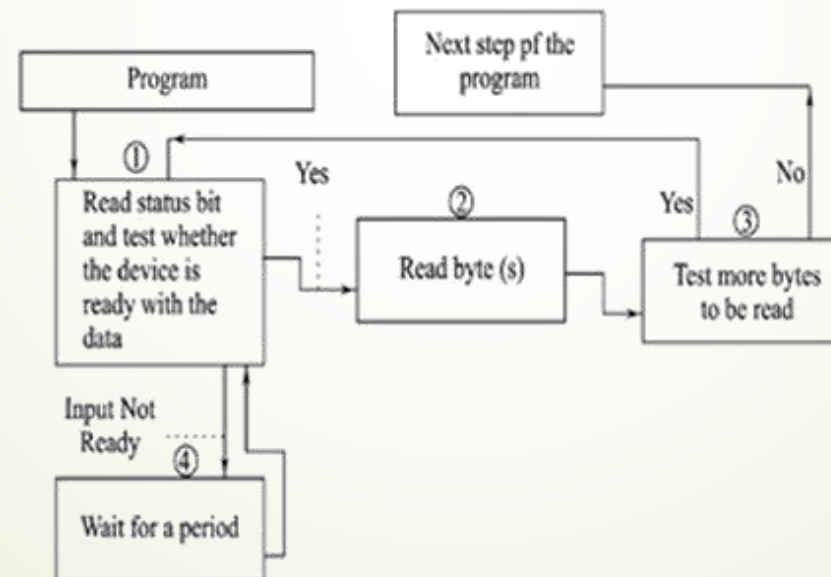
1) Programmed I/O

- overall operation of the programmed I/O can be summaries as
 - The processor is executing a program and encounters an instruction relating to I/O operation.
 - The processor then executes that instruction by issuing a command to the appropriate I/O module.
 - The I/O module will perform the requested action based on the I/O command issued by the processor (READ/WRITE) and set the appropriate bits in the I/O status register
 - The processor will periodically check the status of the I/O module until it find that the operation is complete.

Handling I/O

Programmed I/O: Input Data Transfer

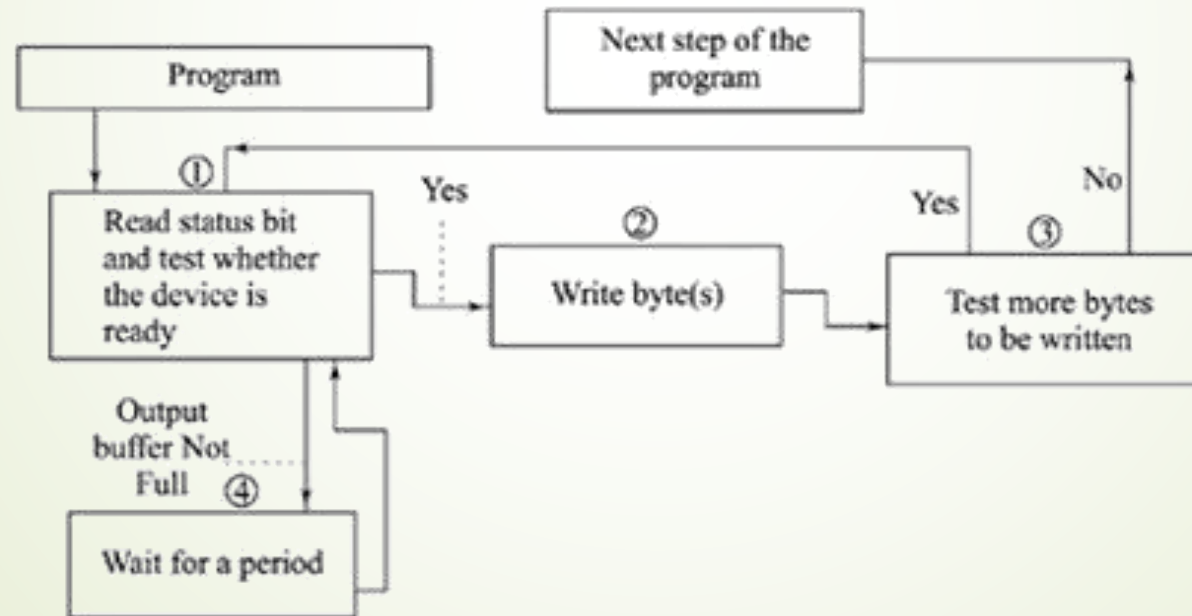
- Each input is read after first testing whether the device is ready with the input (a state reflected by a bit in a status register).
- The program waits for the ready status by repeatedly testing the status bit and till all targeted bytes are read from the input device.
- The program is in busy state only after the device gets ready else in wait state.



Handling I/O

Programmed I/O: Output Data Transfer

- Each output written after first testing whether the device is ready to accept the byte at its output register or output buffer is empty.
- The program waits for the ready status by repeatedly testing the status bit(s) and till all the targeted bytes are written to the device.
- The program in busy state only after the device gets ready else wait state.



Interrupt-Driven I/O

- alternative scheme dealing with I/O
- a way of controlling input/output activity whereby a peripheral or terminal that needs to make or receive a data transfer sends a signal.
- Allow CPU to carry on with other operations until the module is ready to transfer data.
- When CPU wants to communicate with the device, it issues an instruction to the appropriate I/O module, and then continues with other operations.
- When the device is ready, it will interrupt the CPU. The CPU can carry out data transfer.

Advantage:

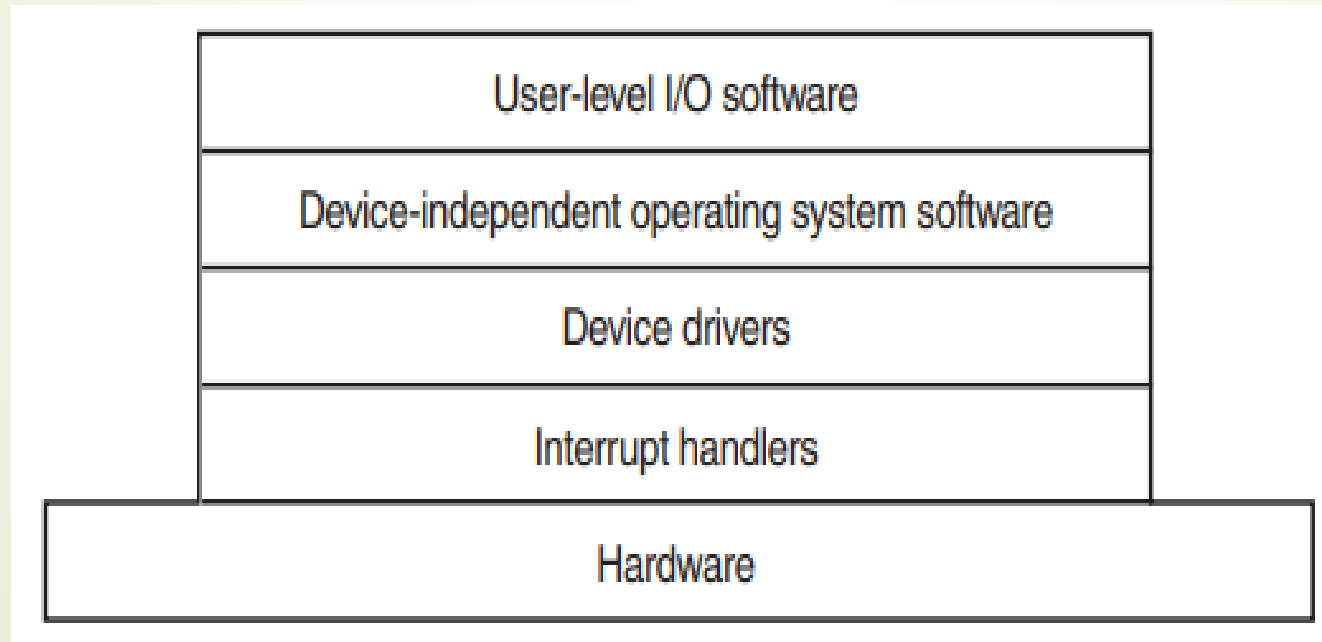
- 1) It is faster than Programmed I/O
- 2) Efficient to use

I/O using DMA

- Direct Memory Access is a technique for transferring data within main memory and external device without passing it through the CPU.
- This technique overcomes the drawbacks of other two I/O techniques which are the time consuming process when issuing a command for data transfer and tie-up the processor in data transfer.
- It is more efficient to use DMA method when large volume of data has to be transferred.
- For DMA to be implemented, processor has to share its' system bus with the DMA module.

I/O Software Layers

- I/o software is typically organized in four layers.
- Each layers has a well-defined function to perform and well-defined interface to the adjacent layers.



I/O Software Layers

1) Interrupt Handlers:

- Also known as interrupt service routine or ISR
- Piece of software or callback function in an OS or more specially in a device driver.
- Execution is triggered by the reception of an interrupt.
- When the interrupt happens , the interrupt procedure does whatever it has to in order to handle the interrupt and wakes up process that was waiting an interrupt to happen.
- Interrupt mechanism accepts an address-a number that selects a specific interrupt handling routine/function from a small set.
- Address is stored in table called interrupt vector table.
- This vector contains the memory addresses of specialized interrupt handlers.

I/O Software Layers

2) Device-Independent I/O Software:

- Basic function is to perform the I/O functions that are common to all devices and uniform interface to the user-level software.
- Difficult to write completely device independent software but we can have some modules which are common among all the devices.
 - Some functions of device-independent I/O software are:
 - Uniform interfacing for device drivers
 - Device naming
 - Device protection
 - Buffering
 - Error Reporting

I/O Software Layers

3) Device-Driver:

- software modules that can be plugged into an OS to handle a particular device.
- Operating System takes help from device drivers to handle all I/O devices.
- Device drivers encapsulate device-dependent code and implement a standard interface in such a way that code contains device-specific register reads/writes.
 - Device drivers performs:
 - To accept request from the device independent software
 - Interact with the device controller to take and give I/O and perform required error handling.
 - Making sure that the request is executed successfully

I/O Software Layers

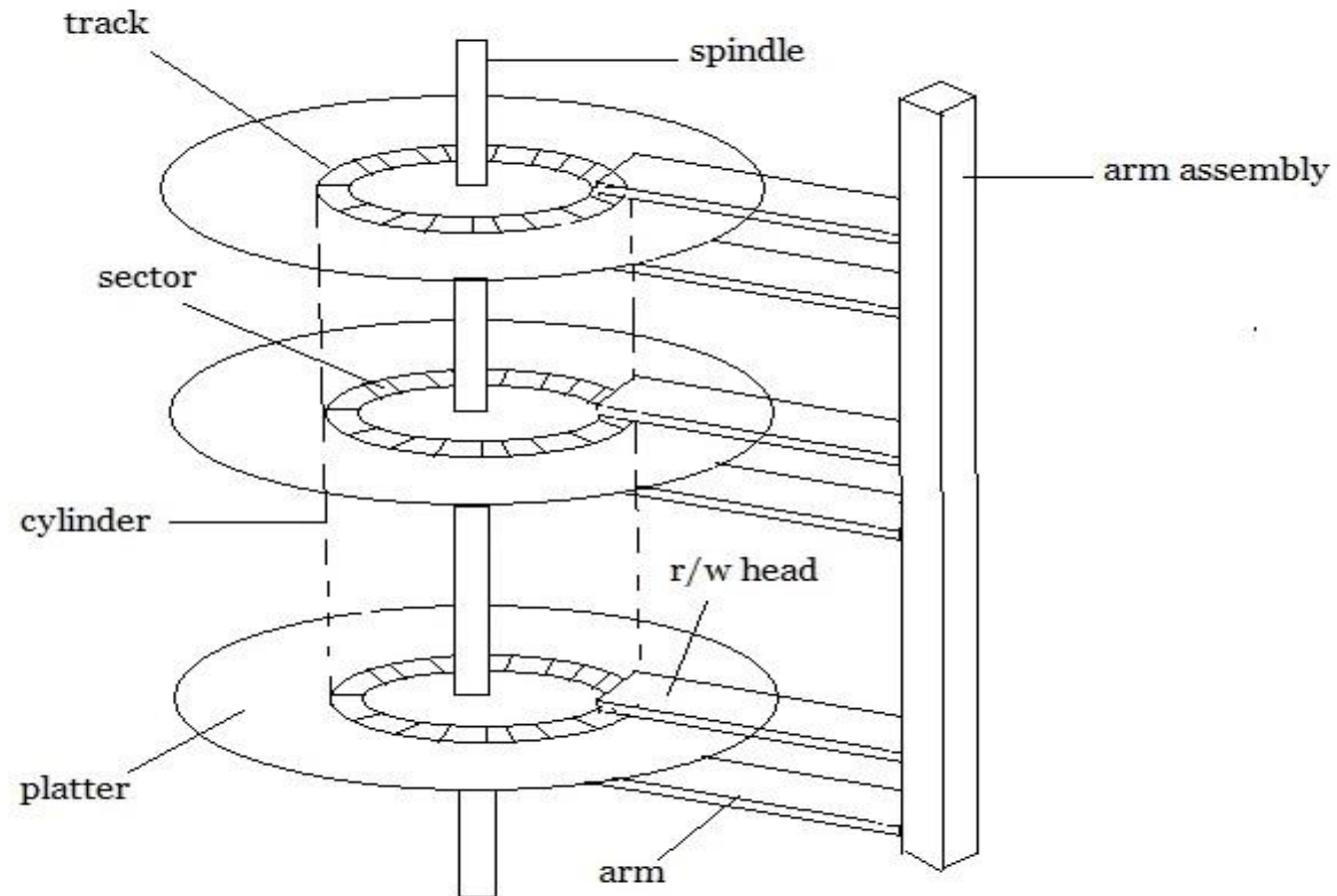
4) User Level I/O software:

- most of the I/O software is within the OS, a small portion of it consists of libraries linked together with user programs, and even whole programs running outside the kernel.
- These are the libraries which provide richer and simplified interface to access the functionality of the kernel or ultimately interactive with the device drivers.
- For example `putchar()`, `getchar()`, `printf()` and `scanf()` are example of user level I/O library `stdio` available in C programming.

DISK

- A disk is a hard or floppy round ,flat and magnetic platter capable of having information read from and write into it.
- Each modern disk contains concentric tracks and each track is divided into multiple sectors.
- Disks are usually arranged as a one dimensional array of blocks , where blocks are smallest storage unit. Blocks can be also as sectors.
- For each sector of disk , there is a read/write desk available.
- Track of the same distance from center form a cylinder.
- The same tracks on all the surface is known as a cylinder.
- Read write head is used to read data from a sector of the magnetic disk.
 - Transfer rate: rate to move data from disk to computer
 - Random access time: seek time + rotational latency

DISK Structure



DISK Structure

- **Seek time** : Time taken in locating the disk arm to a specified track where the read/write request will be satisfied.
- **Rotational latency**: Time is taken by the sector to come under the R-W head
- **Data transfer time** : Time is taken to transfer the required amount of data. It depends upon the rotational speed.
- **Disk access time**: Rotational time + seek time + Transfer time
- **Disk response time**: average of time spent by a request waiting to perform its I/O operation.

DISK Scheduling Algorithm

- An important process in operating systems that determines the order in which disk access requests are serviced.
- Objective of disc scheduling is to minimize the time it takes to access data on the disk and to minimize the time it takes to complete a disk access request.
- Disk access time is determined by two factors: seek time and rotational latency.
- Disk scheduling algorithms are an essential component of modern operating systems and are responsible for determining the order in which disk access requests are serviced.
- The primary goal of these algorithms is to minimize disk access time and improve overall system performance.

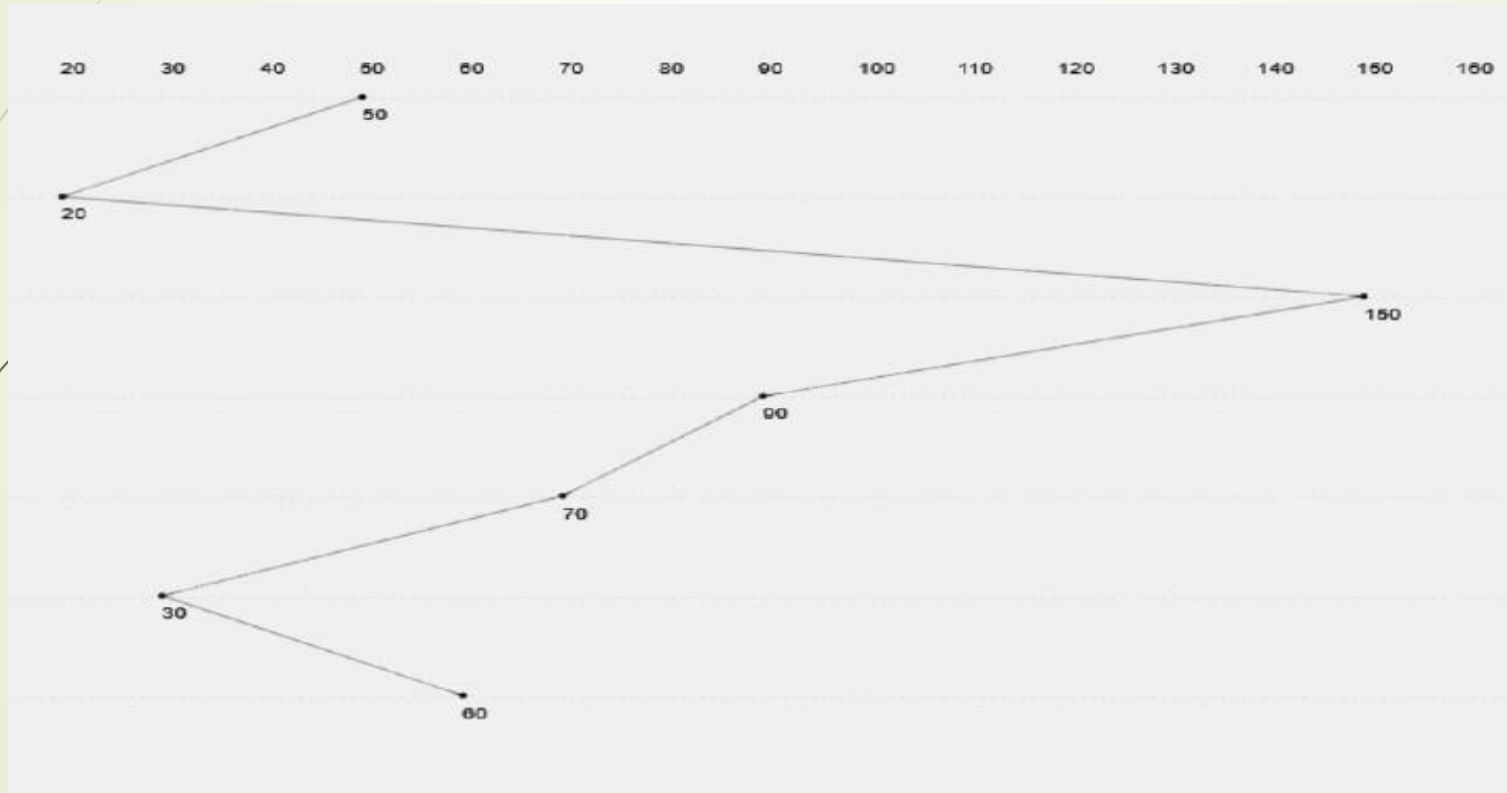
DISK Scheduling Algorithm

1) First-Come-First-Serve

- simplest and most straightforward disk scheduling algorithms
- operates on the principle of servicing disk access requests in the order in which they are received
- the disk head is positioned at the first request in the queue and the request is serviced.
- The disk head then moves to the next request in the queue and services that request.
- This process continues until all requests have been serviced.
- Advantage:
 - Every request gets a fair chance
 - No indefinite postponement
- Disadvantage
 - Does not try to optimized seek time.
 - May not provide the best possible service.

DISK Scheduling Algorithm

Example: Suppose we have an order of disk access requests: 20 150 90 70 30 60. and current head position is 50. A disk contain 200 track



The total seek time = $(50-20) + (150-20) + (150-90) + (90-70) + (70-30) + (60-30) = 310$

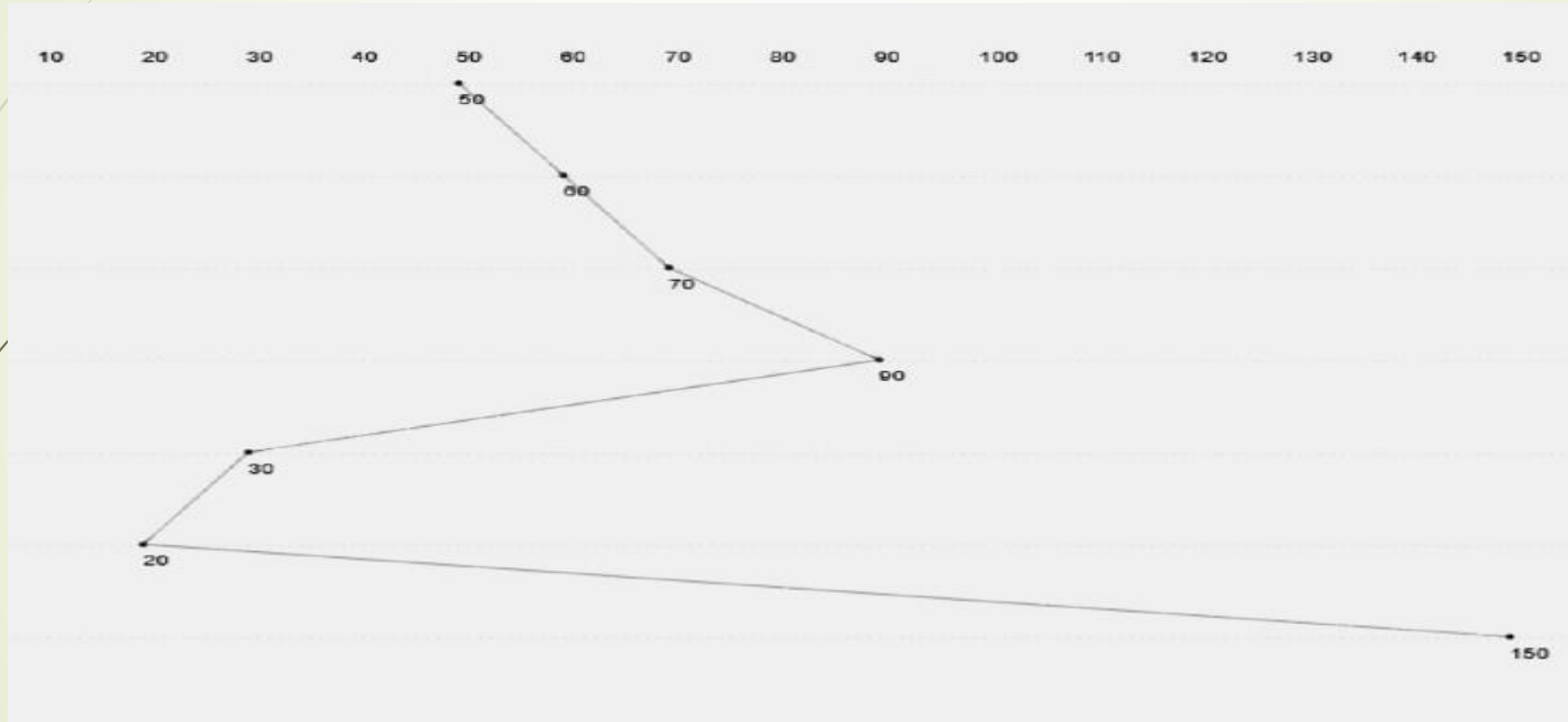
DISK Scheduling Algorithm

2) Shortest-Seek-Time-First

- used in operating systems to efficiently manage disk I/O operations.
- The goal of SSTF is to minimize the total seek time required to service all the disk access requests.
- The disk head moves to the request with the shortest seek time from its current position, services it, and then repeats this process until all requests have been serviced.
- Decrease the response time and increase throughput of the system.
- Advantage:
 - Average response decrease
 - Throughput increase
- Disadvantage
 - Can cause starvation for a request if it has higher seek time as compared to incoming request.

DISK Scheduling Algorithm

Example: Suppose we have an order of disk access requests: 20 150 90 70 30 60. and current head position is 50



The total seek time = $(60-50) + (70-60) + (90-70) + (90-30) + (30-20) + (150-20) = 240$

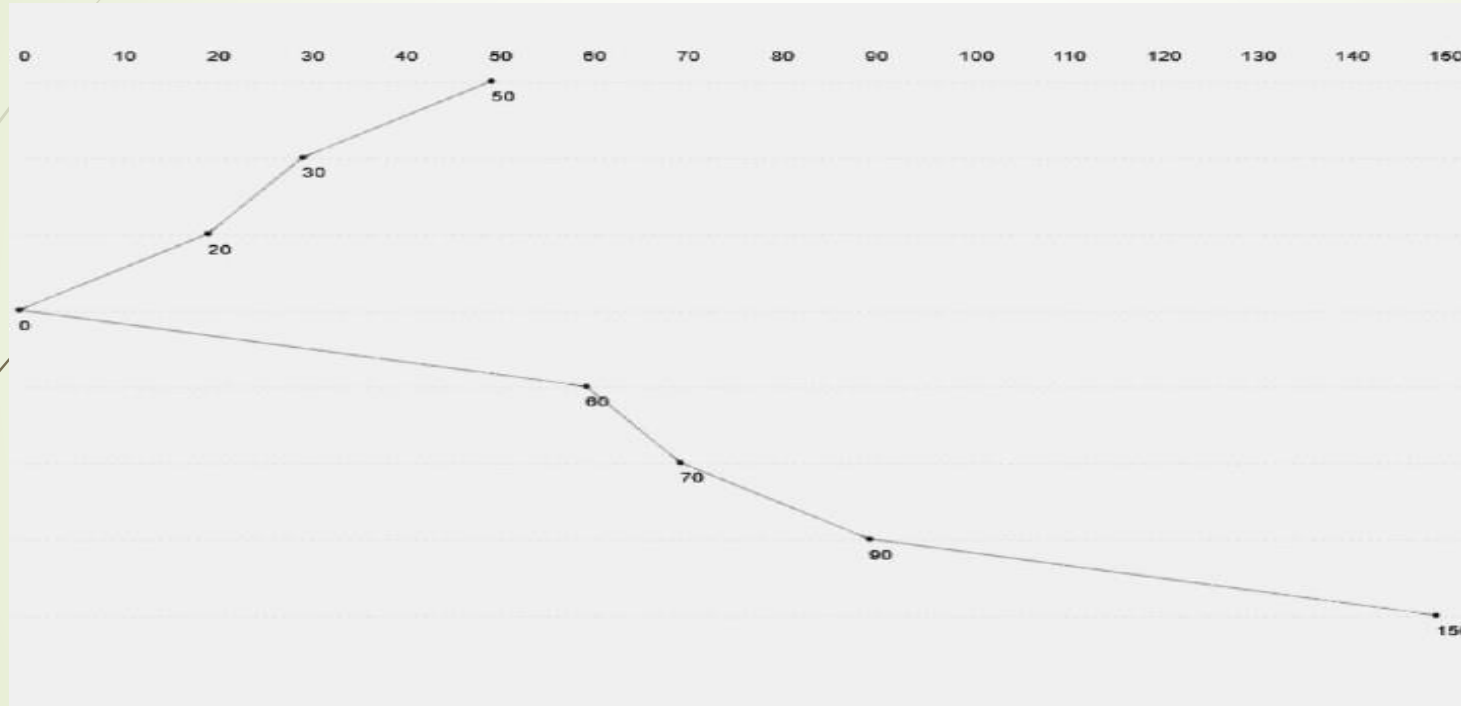
DISK Scheduling Algorithm

3) SCAN:

- SCAN algorithm moves the disk head in a single direction and services all requests until it reaches the end of the disk, and then it reverses direction and services all the remaining requests.
- The disk head starts at one end of the disk, moves toward the other end, and services all requests that lie in its path.
- SO this algorithm works like elevator and hence also known as elevator algorithm.
- Advantage:
 - High throughput
 - Low variance of response time
 - Average response time
- Disadvantages
 - Long waiting time for requests for locations just visited by disk arm.

DISK Scheduling Algorithm

Example: Suppose we have an order of disk access requests: 20 150 90 70 30 60. and current head position is 50 and head direction is left .



The total seek time = $(50-30) + (30-20) + (20-0) + (60-0) + (60-70) + (90-70) + (90-150) = 200$

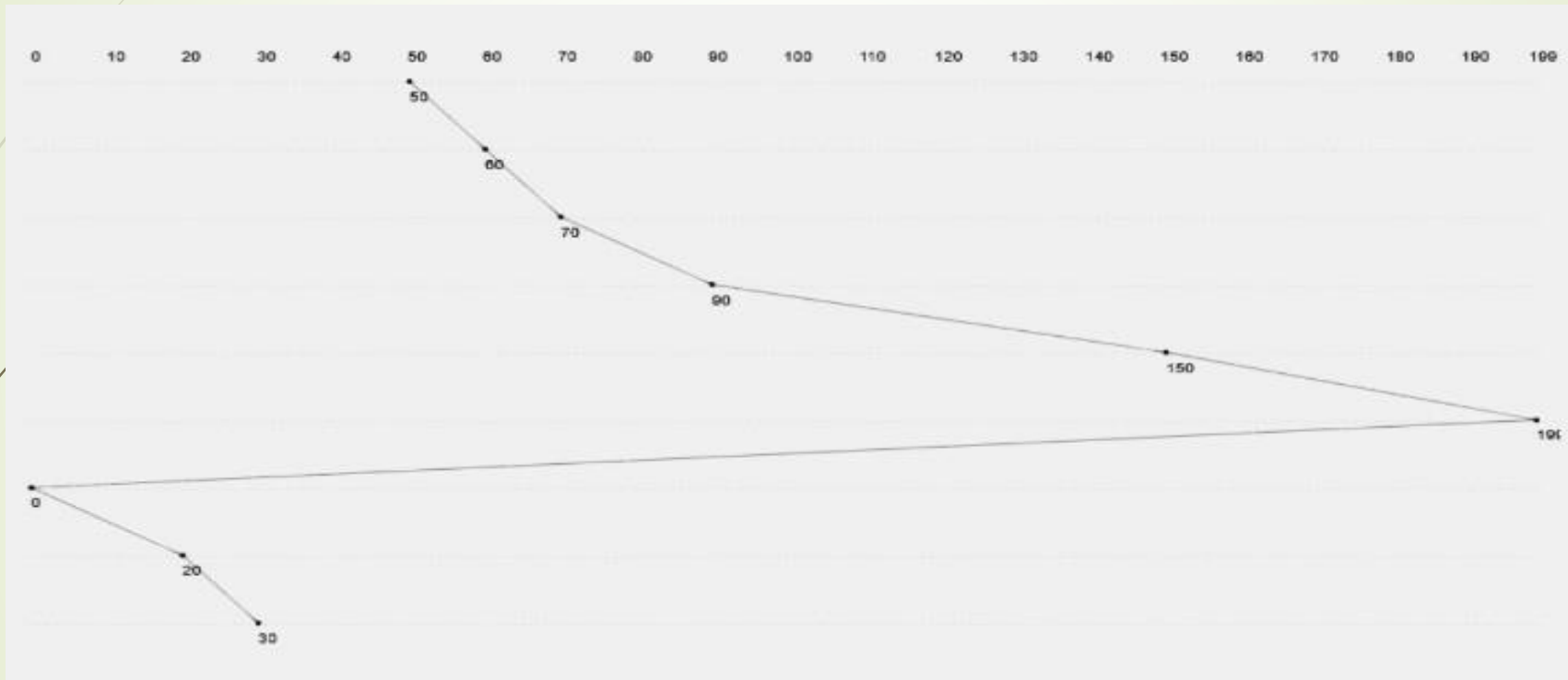
DISK Scheduling Algorithm

4) C- SCAN:

- The C-SCAN (Circular SCAN) algorithm operates similarly to the SCAN algorithm, but it does not reverse direction at the end of the disk.
- Instead, the disk head wraps around to the other end of the disk and continues to service requests.
- Can reduce the total distance the disk head must travel, improving disk access time.
- Advantage:
 - It provides uniform waiting time
 - Provide better performance
 - Average response time
- Disadvantages
 - It cause more seek movements as compare to SCAN Algorithm.
 - It cause the head to move till the end of disk even if there are no requests to be serviced.

DISK Scheduling Algorithm

Example: Suppose we have an order of disk access requests: 20 150 90 70 30 60. and current head position is 50 and head direction is right.



The total seek time = $(60-50) + (70-60) + (90-70) + (150-90) + (199-150) + (199-0) + (20-0) + (30-20) = 378$

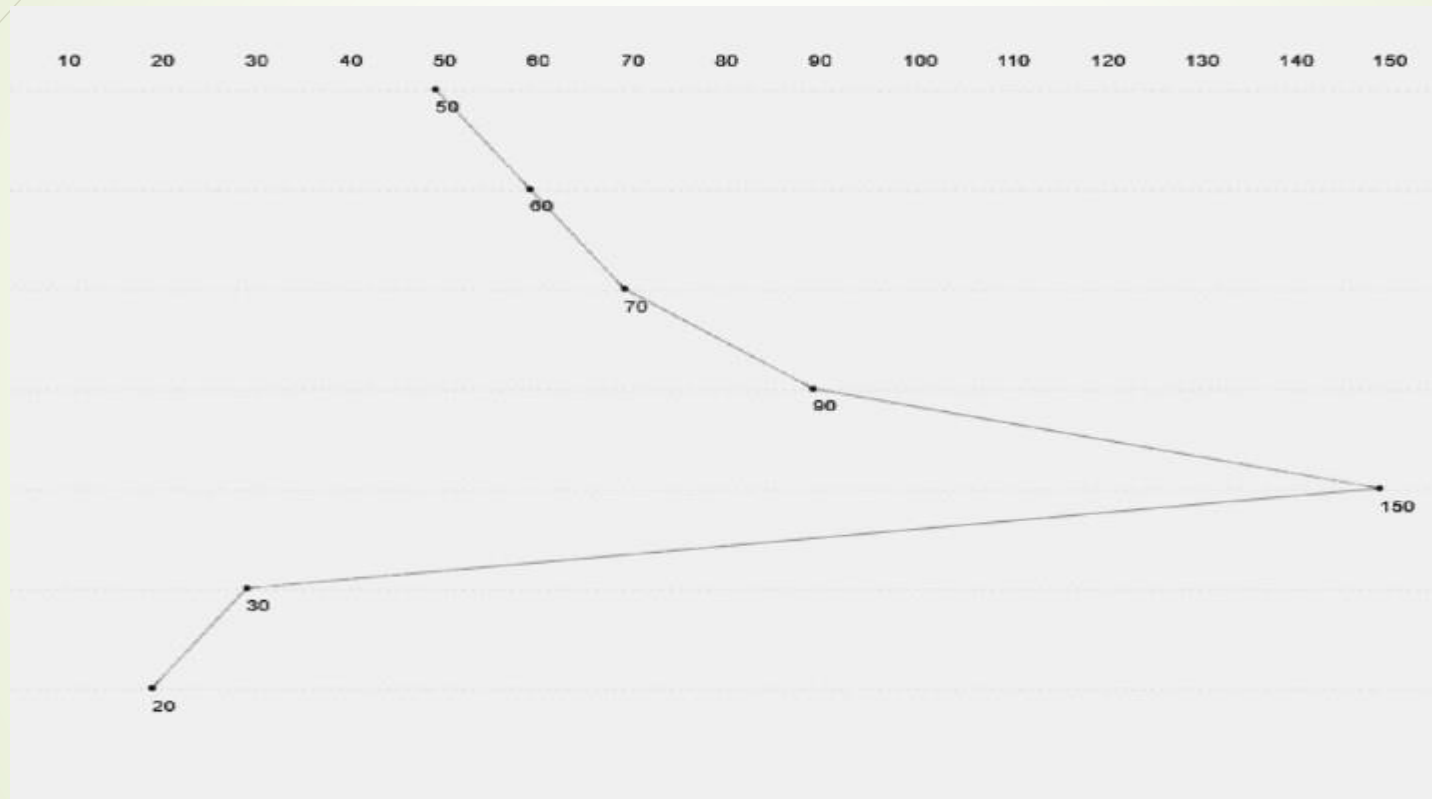
DISK Scheduling Algorithm

5) LOOK:

- LOOK algorithm is similar to the SCAN algorithm but disk arm goes only to the last request to serviced in front of the head and then reverses its direction from there only.
- can reduce the total distance the disk head must travel, improving disk access time.
- Advantage:
 - It does not cause the head to move till the ends of the disks when there are no requests to be serviced
 - Provides better performance as compare with SCAN
 - Provides low variance in response time and waiting time.
- Disadvantages
 - There is a overhead of finding end request.
 - Cause long waiting time for the cylinders just visited by the head.

DISK Scheduling Algorithm

Example: Suppose we have an order of disk access requests: 20 150 90 70 30 60. and current head position is 50 and head direction is right .



The total seek time $(60-50) + (70-60) + (90-70) + (150-90) + (150-30) + (30-20) = 230$

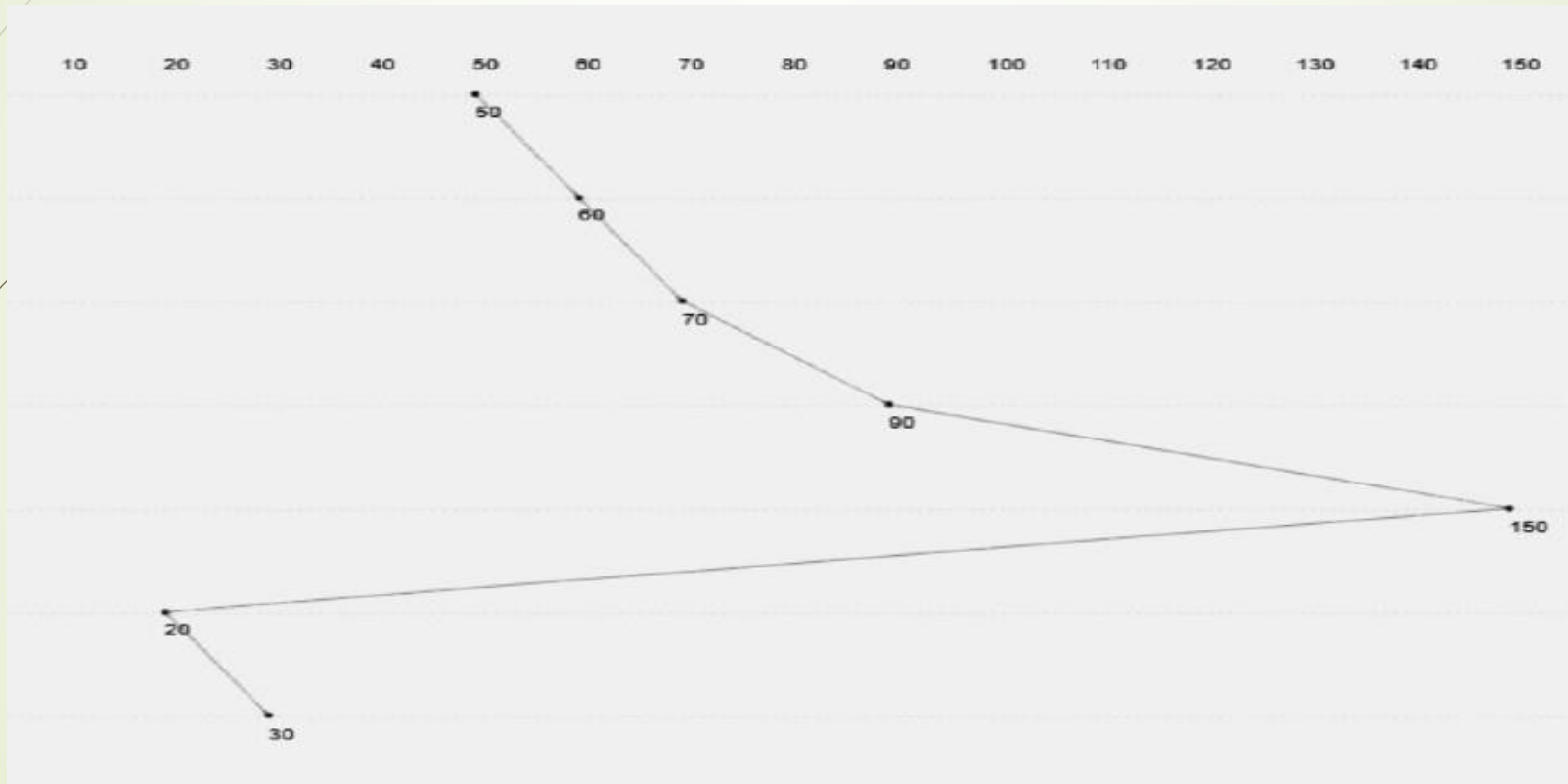
DISK Scheduling Algorithm

6) C-LOOK:

- C-LOOK is similar to the C-SCAN disk scheduling algorithm.
- In this head goes only to the last request to be serviced.
- Head starts from the first request at one end of the disk and moves towards the last request at the other end servicing all the requests in between.
- After reaching the last request at the other end, head reverse its direction. It then returns to the first request at the starting end without servicing any request in between.
- Advantage:
 - It does not cause the head to move till the ends of the disks when there are no requests to be serviced
 - It reduce the waiting time for the cylinders just visited by the head.
 - Provides better performance then Look
- Disadvantages
 - There is a overhead of finding end request.

DISK Scheduling Algorithm

Example: Suppose we have an order of disk access requests: 20 150 90 70 30 60. and current head position is 50 and head direction is right .



The total seek time= $(60-50) + (70-60) + (90-70) + (150-90) + (150-20) + (30-20) = 240$