Computer Fundamentals
And
Operating System Concepts

➢Process Management

- When we say an OS does process management it means **an OS is responsible for process creation, to provide environment for an execution of a process, resource allocation, scheduling, resources management, inter process communication, process coordination, and terminate the process.**

Q. What is a Program?

❑User view:

- Program is a finite set of instructions written in any programming language given to the machine to do specific task.

❑System view:

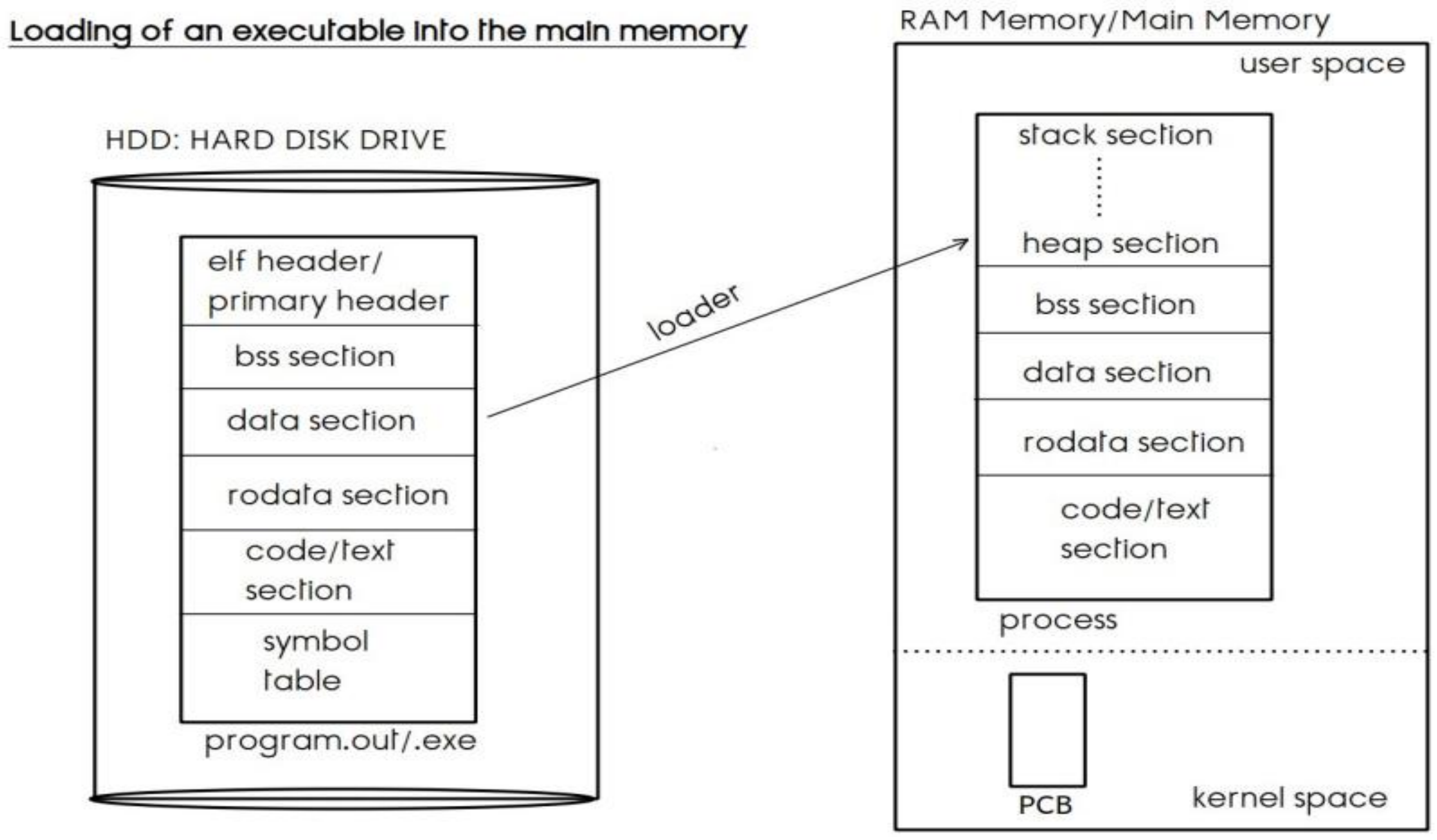
- Program is an executable file in HDD which divided logically into sections like exe header, bss section, data section, rodata section, code section, symbol table.

Q. What is a Process?

- **User view:**
  - Program in execution is called as **a process.**
  - Running program is called as **a process.**
  - When a program gets loaded into the main memory it is referred as **a process.**
  - Running instance of a program is referred as **a process**.
- **System view:**
  - Process is a program loaded into the main memory which has got PCB into the main memory inside kernel space and program itself into the main memory inside user space has **got bss section, rodata section, code section, and two new sections gets added for the process .**

- **stack section:** contains function activation records of called functions.
- **heap section:** dynamically allocated memory

Operating Systems and Computer Fundamentals



Operating Systems and Computer Fundamentals

- As a kernel, core program of an a OS runs continuously into the main memory, part of the main memory which is occupied by the kernel **referred as kernel space** and whichever part is left **is referred as an user space**, so **main memory is divided logically into two parts: kernel space & user space.**
- **User programs gets loaded into the user space only.**
- When we execute a program or upon submission of a process very first one structure gets created into the main memory inside kernel space by an OS in which all **the information which is required to control an execution of that process** can be kept, this structure is referred as a **PCB.**
- **PCB : Process Control Block, is also called as a Process Descriptor.**
- Per process one PCB gets created and PCB remains inside the main memory throughout an execution of a program, upon exit PCB gets destroyed from the main memory.

Operating Systems and Computer Fundamentals

❖PCB mainly contains:

- PID : Process ID
- PPID : Parent Processes ID
- PC : Program Counter
- Execution context
- Kernel stack
- Exit status

- CPU sched information, memory management information, information about resources allocated for that process, execution context etc...

1. **Resident monitor**
2. **Batch System**

   -The batch/group of similar programs is loaded in the computer, from which OS loads one program in the memory and execute it. The programs are executed one after another.

   - In this case, if any process is performing IO, CPU will wait for that process and hence not utilized efficiently.

3. **Multi-programming**

   - Better utilization of CPU

   - Loading multiple Programs in memory

   - Mixed program(CPU bound + IO bound)

4. Time-sharing/Multitasking

- Sharing CPU time among multiple process/task present in main memory and ready for execution
- Any process should have response time should be less then 1sec
- Multi-tasking is divided into two types
    - Process based multitasking
    - Thread based multitasking
        - Thread is a light weight process .
        - When new thread is created a new stack and new TCB is created.
        - Thread Share text, data, heap sections with the parent process

▪ Process and thread

-Process is a container for resources.

-Thread is unit of execution/ scheduler.

- For each process one thread is created by default it is called as main thread

5. **Multi-user system**

   - Multiple users runs multiple programs concurrently.

6. **Multi-processor/ Mutli-core system**

   - System can run on a machine in which more than one CPU's are connected in a closed circuit.
   - Multiprocessing Advantage is it increased throughput (amount of work done in unit time)
   - There are two types of multiprocessor systems:

❖ **Asymmetric Multi-processing**

❖ **Symmetric Multi-processing**

- **Asymmetric Multi-processing**

- OS treats one of the processor as master processor and schedule task for it. The task is in turn divided into smaller tasks and get them done from other processors.

- **Symmetric Multi-processing**

- -OS considers all processors at same level and schedule tasks on each processor individually. All modern desktop systems are SMP.

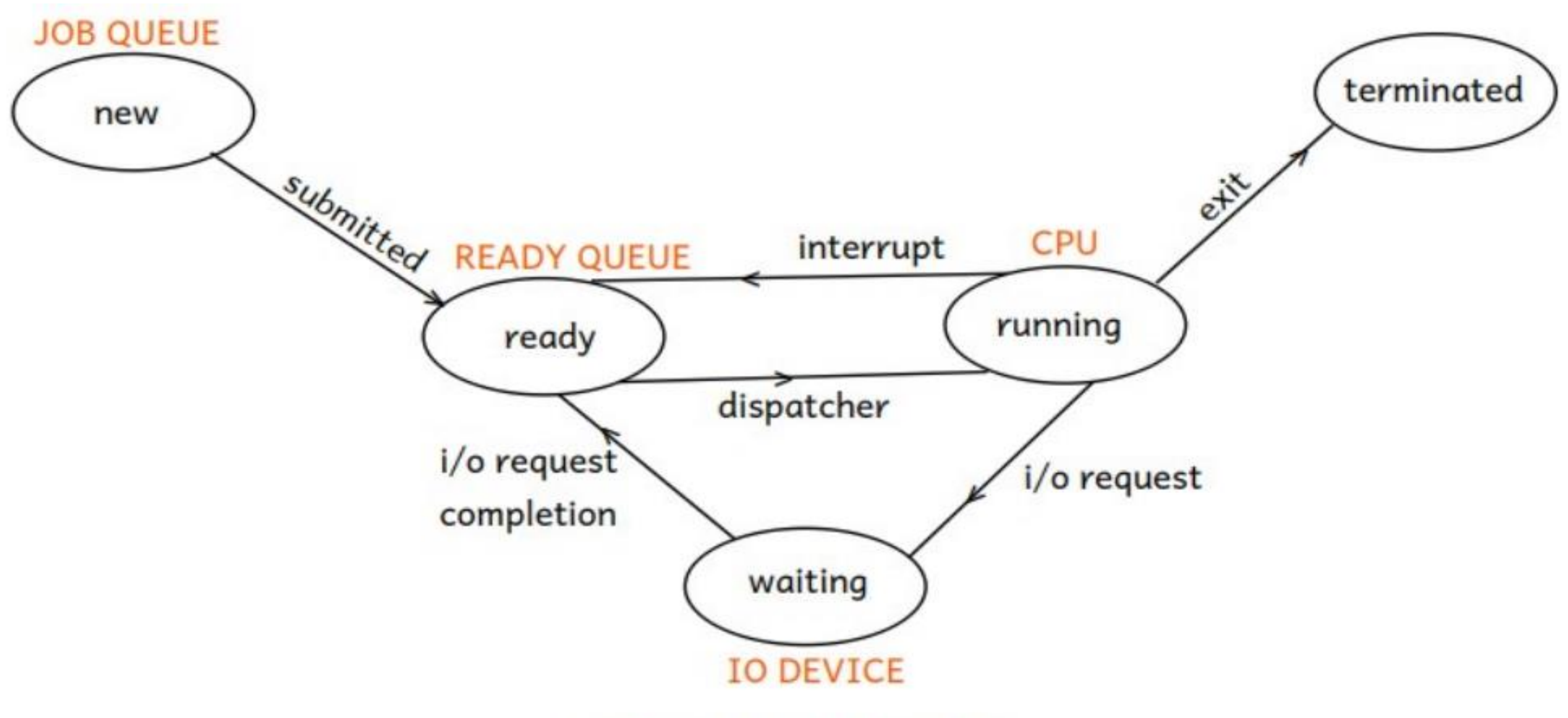
➢Process States:

- Throughout execution, process goes through different states out of which at a time it can be only in a one state.

- States of the process:

1. **New state:** upon submission or when **a PCB** for a process gets created into the main memory process is in **a new state.**
2. **Ready state:** after submission, if process is in the main memory and waiting for the CPU time, it is in **a ready state**.
3. **Running state:** if currently the CPU is executing any process then state of that process is considered **as a running state**.
4. **Waiting state:** if a process is requesting for any i/o device then state of that process is considered as **a waiting state**.
5. **Terminated state:** upon exit, process goes into terminated state and its PCB gets destroyed from the main memory.

Operating Systems and Computer Fundamentals



PROCESS STATE DIAGRAM

➢Process States:

- To keep track on all running programs, an OS maintains few data structures referred **as kernel data structures:**

1. **Job queue:** it contains list of PCB's of all submitted processes.
2. **Ready queue:** it contains list of PCB's of processes which are in the main memory and waiting for the CPU time.
3. **3. Waiting queue:** it contains list of PCB's of processes which are requesting for that particular device.

1. **Job Scheduler/Long Term Scheduler:** it is a system program which selects/schedules jobs/processes from job queue to **load them onto the ready queue**.
2. **CPU Scheduler/Short Term Scheduler:** it is a system program which selects/schedules job/process **from ready queue to load it onto the CPU.**
3. **Dispatcher:** it is a system program which loads a process onto the CPU which is scheduled by the CPU scheduler, and **the time required for the dispatcher to stops an execution of one process and to starts an execution of another process is referred as dispatcher latency.**

➢Context Switch:

- As during context-switch, the CPU gets switched from an execution context of one process onto an execution context of another process, and hence it is referred as **"context-switch".**
- **context-switch = state-save + state-restore**
- **state-save** of suspended process can be done i.e. an execution context of suspended process gets saved into its PCB.
- **state-restore** of a process which is scheduled by the CPU scheduler can be done by the dispatcher, dispatcher copies an execution context of process scheduled by the cpu scheduler from its PCB and restore it onto the CPU registers.
- When a high priority process arrived into the ready queue, low priority process gets suspended by means of sending an interrupt, and control of the CPU gets allocated to the high priority process, and its execution gets completed first, then low priority process can be resumed back, i.e. the CPU starts executing suspended process from the point at which it was suspended and onwards.

- CPU Scheduler gets called in the following four cases:

Case-1: Running -> Terminated

Case-2: Running -> Waiting

Case 3: Running -> Ready

Case-4: Waiting -> Ready

- There are two types of CPU scheduling:

1. **Non-preemptive:** under non-preemptive cpu scheduling, **process releases the control of the CPU by its own i.e. voluntarily**.

   - e.g. in above case 1 & case 2

2. **Preemptive:** under preemptive cpu scheduling, **control of the CPU taken away forcefully from the process.**

   - e.g. in above case 3 & 4.

Following algorithms used for CPU Scheduling:

1. **FCFS (First Come First Served) CPU Scheduling**
2. **SJF (Shortest Job First) CPU Scheduling**
3. **Round Robin CPU Scheduling**
4. **Priority CPU Scheduling**

- Multiple algorithms are there for CPU scheduling, so there is need to decide which algorithm is best suited at specific situation and which algorithm is an efficient one, to decide this there are certain criteria's **called as scheduling criteria's: cpu utilization, throughput, waiting time, response time and turn-around-time.**

➢CPU Scheduling Criteria's:

1. **CPU Utilization:** one need to select such an algorithm in which utilization of the CPU must be as **maximum as a possible**.
2. **Throughput**: total work done per unit time. One need to select such an algorithm in which throughput must be **as maximum possible.**
3. **Waiting Time:** it is the total amount of time spent by the process into the ready queue for waiting to get control of the CPU from its time of submission. One need to select such an algorithm in which waiting time must be as **minimum as possible.**
4. **Response Time:** it is a time required for the process to get first response from the CPU from its time of submission. One need to select such an algorithm in which response time must be as **minimum as possible**

5. **Turn-Around -Time:** it is the total amount of time required for the process to complete its execution from its time of submission.

- One need to select such an algorithm in which turn-around-time must be as **minimum as possible.**
- **Execution Time:** it is the **total amount of time spent by the process onto the CPU to complete its execution**.

  - **OR CPU Burst Time:** total no. of CPU cycles required for the process to complete its execution.
- **Turn-Around-Time = Waiting Time + Execution Time.**
- **Turn-around-time is the sum of periods spent by the process into ready queue for waiting and onto the CPU for execution from its time of submission.**

➢CPU Scheduling

1. **FCFS (First Come First Served) CPU Scheduling**

- In this algorithm, process which is arrived first into the ready queue gets the control of the CPU first i.e. control of the CPU gets allocated for processes as per their order of an arrival into the ready queue.
- This algorithm is simple to impalement and can be implemented by using fifo queue.
- It is a non-preemptive scheduling algorithm.

❑**Convoy effect:** in fcfs, due to an arrival of longer process before shorter processes, shorter processes has to wait for longer duration and due to which average waiting time gets increases, which results into an increase in an average turn-around-time and hence overall system performance gets down.

➢FCFS Scheduling

Process	Arrival Time	CPU Burst	Wait Time	Turn Around Time
P1	0	24		
P2	0	3		
P3	0	3		

2. SJF(Shortest Job First) CPU Scheduling:

- In this algorithm, process which is having minimum CPU burst time gets the control of the CPU first, and whenever tie is there it can be resolved by using fcfs. - SJF algorithm ensures minimum waiting time.
- Under non-preemptive SJF, algorithm fails if the submission time of processes are not same, and hence it can be implemented as preemptive as well.
- Non-preemptive SJF is also called as SNTF(Shortest-Next-Time-First).
- Preemptive SJF is also called as SRTF(Shortest-Remaining-Time-First).

- **Starvation:** in this algorithm, as shorter processes has got higher priority, process which is having larger CPU burst time may gets blocked i.e. control of the CPU will never gets allocated for it, such situation is called as starvation/indefinite blocking.

➢ SJF/SNTF Scheduling

Process	Arrival Time	CPU Burst	Wait Time	Turn Around Time
P1	0	7		
P2	2	4		
P3	4	1		
P4	5	4		

➢ SRTF Scheduling

Process	Arrival Time	CPU Burst	Remaining Time	Wait Time	Turn Around Time
P1	0	7			
P2	2	4			
P3	4	1			
P4	5	4			

3. Round Robin Scheduling Algorithm

- In this algorithm, before allocating the CPU for processes, **some fixed time slice or time quantum** gets decided in advanced, and at any given time control of the CPU may remains allocated with any process maximum for that decided time-slice, once the given time slice is finished of that process, it gets suspended and control of the CPU will be allocated to the next process again for maximum that decided time slice and so on..., each process gets control of the CPU in a round robin manner i.e. cpu gets shared among processes equally.
- If any process completes its execution before allocated time slice then control of the CPU will be released by that process and CPU gets allocated to the next process as soon as it is completed for effective utilization of the CPU.
- There **is no starvation in RR Scheduling algorithm**.
- This algorithm is **purely preemptive.**
- This algorithm **ensures minimum response time**.
- **If time slice is minimum then there will be extra overhead onto the CPU due to frequent context-switch.**

➢ Round Robin Scheduling

Process	CPU Burst	Remaining Time	Wait Time	Response Time
P1	53			
P2	17			
P3	68			
P4	24			

4. Priority Scheduling

- In this algorithm, process which is having highest priority gets control of the CPU first**, each process is having priority in its PCB**.
- priority for a process can be decided by two ways:
    1. **internally** – priority for process can be decided by an OS depends on no. of resources required for it.
    2. **externally** – priority for process can be decided by the user depends on requirement.
- **Minimum priority value indicates highest priority.**
- This algorithm **is purely preemptive.**
- **Due to the very low priority process may gets blocked into the ready queue and control of the CPU will never gets allocated for such a process, this situation is referred as a starvation or indefinite blocking**.
- **Ageing:** it is a technique in which, an OS gradually increments priority of blocked process, i.e. priority of blocked process gets incremented after some fixed time interval by an OS, so that priority of blocked process becomes sufficient enough to get control of the CPU, and starvation can be avoided.

➢ Priority Scheduling

Process	Arrival Time	CPU Burst	Priority	Wait Time
P1	0	10	3	
P2	0	1	1	
P3	0	2	4	
P4	0	5	2	

Operating Systems and Computer Fundamentals

➢Multi-level queue :

- In modern OS, the ready queue can be divided into multiple sub-queues and processes are arranged in them depending on their scheduling requirements. This structure is called as "Multi-level queue".
- If a process is starving in some sub-queue due to scheduling algorithm, it may be shifted into another sub-queue. This modification is referred as "Multi-level feedback queue".
- The division of processes into sub - queues may differ from OS to OS.

1.	Important Services	:Priority Scheduling
2.	Background Task	:SJF
3.	GUI Tasks	:RR
4.	Other Tasks	:FCFS

➢Inter Process Communication

- Processes running into the system can be divided into two categories:

1. **Independent Processes:** - Process which do not shares data (i.e. resources) with any other process referred as an independent process. OR Process which do not affects or not gets affected by any other process referred as an independent process.
2. **Co-operative Processes:** - Process which shares data (i.e. resources) with any other process referred as co - operative process. OR Process which affects or gets affected by any other process referred as co-operative process.

❖Reasons for cooperating processes:

- Information sharing
- Computation speedup
- Modularity
- Convenience
- Cooperating processes need inter process communication (IPC)

Operating Systems and Computer Fundamentals

Q. Why there is need of an IPC?

As concurrently executing co-operative processes shares common resources, so there are quite chances to occur conflictions between them and to avoid this conflictions there is a need of communication takes place between them.

Q. What is an Inter Process Communication?

An IPC is one of the important **service made available by the kernel, by using which co-operative processes can communicates with each other.**

- Inter process communication takes place only **between co-operative processes.**
- Any process cannot directly communicates with any other process, hence there is a need of some medium, and to provide this medium is the job of an OS/Kernel.

Operating Systems and Computer Fundamentals

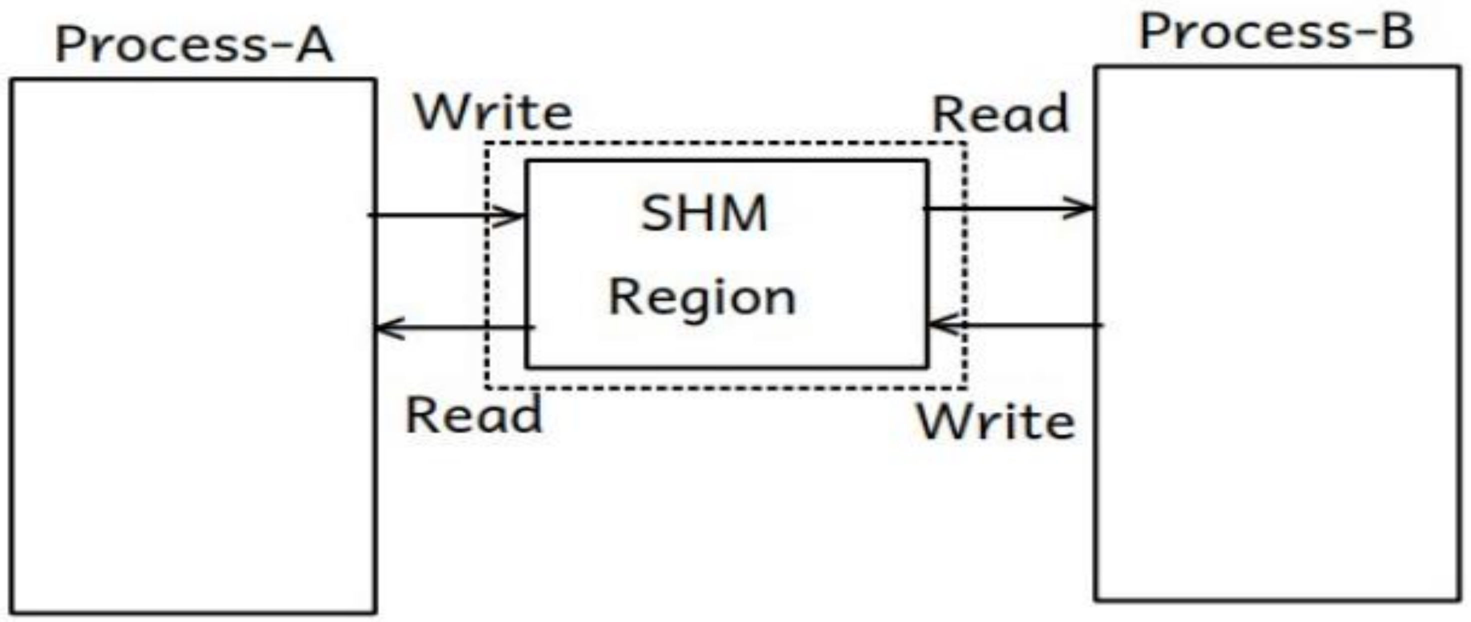
❖There are **two techniques** by which IPC can be done/there are two IPC Models:

1. **Shared Memory Model:** under this technique, processes can communicates with each other by means of reading and writing data into the shared memory region (i.e. it is a region/portion of the main memory) which is provided by an OS temporarily on request of processes want to communicates.
2. **Message Passing Model:** under this technique, processes can communicates with each other by means of sending messages.

- Any process cannot directly send message to any other process.
- Shared Memory Model is faster than Message Passing Model

Operating Systems and Computer Fundamentals

-



SHARED MEMORY MODEL

2. Message Passing Model: there are further different IPC techniques under message passing model.

I. **Pipe: -** By using Pipe mechanism one process can send message to another process, vice versa is not possible and hence **it is a unidirectional communication technique**.

- In this IPC mechanism, from one end **i.e. from write end one process can writes data into the pipe, whereas from another end i.e. from read end, another process can read data from it, and communication takes place.**

- There are two types of pipes:

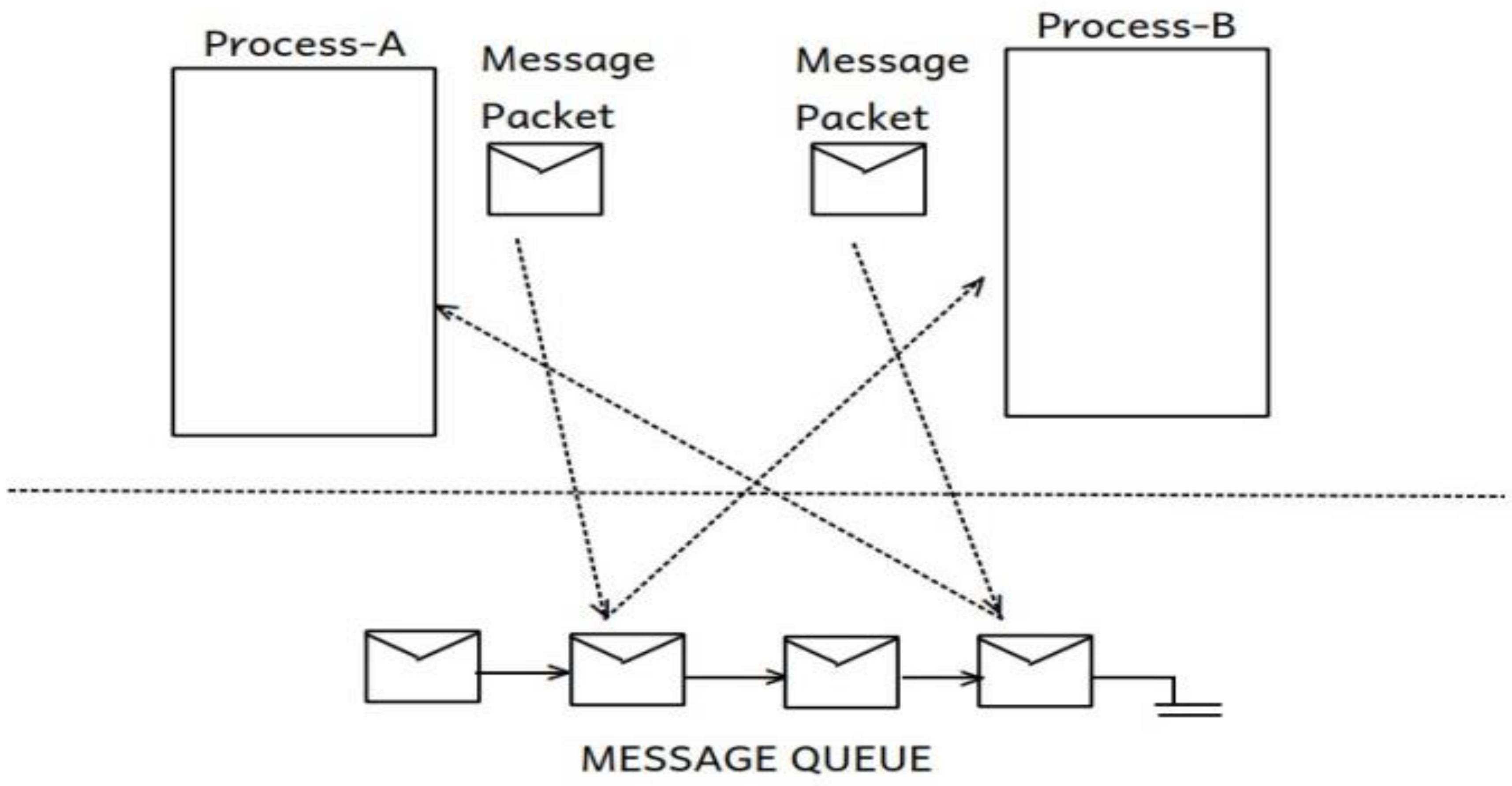
1. **unnamed pipe:** in this type of pipe mechanism, only related processes can communicates by **using pipe (|) command.**
2. **named pipe:** in this type of pipe mechanism, related as well as non-related processes can communicates by **using pipe() system call.**

- By using Pipe only processes which are running in the same system can communicates,

ii. Message Queue:

- By using message queue technique, processes can communicates by means of sending as well as receiving **message packets** to each other via message queue provided by the kernel as a medium, and hence it **is a bidirectional communication.**
- **Message Packet: Message Header(Information about the message) + Actual Message.**
- Internally an OS maintains message queue in which message packets sent by one process are submitted and can be sent to receiver process **and vice-versa**.
- By using message queue technique, only processes which are running in the same system can communicates.

Inter Process Communication



iii. Signals:

- Processes communicates by means of sending signals as well.
- One process can send signal to another process through an OS.
- An OS sends signal to any process but any another process cannot sends signal to an OS.

- **Example**: When we shutdown the system, an OS sends **SIGTERM** signal to all processes, due to which processes gets terminated normally, but few processes can handle **SIGTERM** i.e. even after receiving this signal from an OS they continues execution, to such processes an OS sends **SIGKILL** signal due to which processes gets **terminated forcefully**.
- **e.g. SIGSTOP, SIGCONT, SIGSEGV etc...**

❑Important Signals

- **SIGINT (2):** When CTRL+C is pressed, INT signal is sent to the foreground process.
- **SIGKILL (9)**: During system shutdown, OS send this signal to all processes to forcefully kill them. Process cannot handle this signal.
- **SIGSTOP (19**): Pressing CTRL+S, generate this signal which suspend the foreground process. Process cannot handle this signal.
- **SIGCONT (18):** Pressing CTRL+Q, generate this signal which resume suspended the process
- **SIGSEGV (11):** If process access invalid memory address (dangling pointer), OS send this signal to process causing process to get terminated. It prints error message "Segmentation Fault".

- By using signal ipc technique, only processes which are running in the same system can communicates.

iv. Socket

- Limitation of above **all IPC techniques** is, **only processes which are running on the same system can communicates**, so to overcome this limitation **Socket IPC mechanism** has been designed.
- By using socket IPC mechanism, **process which is running on one machine can communicates with process running on another machine, whereas both machines are at remote distance from each other and provided they connected in a network (either LAN / WAN/Internet).**
- **Socket = IP Address + Port Number.**

- e.g. chatting application.

Process Coordination / Process Synchronization

Why Process Co-ordination/Synchronization?

- If concurrently executing co-operative processes are accessing common resources, then conflictions may takes place, which may results into the problem of **data inconsistency**, and hence to avoid this problem coordination / synchronization between these processes is required.
- **Race Condition:** if two or more processes are trying to access same resource at a time, race condition may occurs, and data inconsistency problem may takes place due to race condition.

- **Race condition** can be avoided by an OS by

1. deciding order of allocation of resource for processes .
2. whichever changes did by the last accessed process onto the resource remains final changes.

➢Synchronization Tools:

1. **Semaphore:**
2. **Mutex :**

➢Semaphore:

- Semaphore was suggested by Dijkstra scientist (dutch math)
- Semaphore is a counter

❑On semaphore two operations are supported:

➢wait operation: decrement op: P operation:

1. Semaphore count is decremented by 1.
2. If cnt < 0, then calling process is blocked(block the current process).
3. Typically wait operation is performed before accessing the resource.

Operating Systems and Computer Fundamentals

➢signal operation: increment op: V operation:

1. semaphore count is incremented by 1.
2. if one or more processes are blocked on the semaphore, then wake up one of the process.
3. typically signal operation is performed after releasing the resource.

Q. If sema count = -n, how many processes are waiting on that semaphore?

Answer: "n" processes waiting

• There are two types of semaphore

i. **Binary semaphore :** can be used when at a time resource can be acquired by only one process.

  - It is an integer variable having either value is 0 or 1.

ii. **Counting / Classic semaphore :** can be used when at a time resource can be acquired by more than one processes

Operating Systems and Computer Fundamentals

2. **Mutex Object:**

- Can be used when at a time resource can be acquired by only one process.
- Mutex object has two states : locked & unlocked, and at a time it can be only in a one state either locked or unlocked.
- Semaphore uses signaling mechanism, whereas mutex object uses locking and unlocking mechanism

➢ **Semaphore vs Mutex**

S: Semaphore can be decremented by one process and incremented by same or another process.

M: The process locking the mutex is owner of it. Only owner can unlock that mutex.

S: Semaphore can be counting or binary.

M: Mutex is like binary semaphore. Only two states: locked and unlocked.

S: Semaphore can be used for counting, mutual exclusion or as a flag.

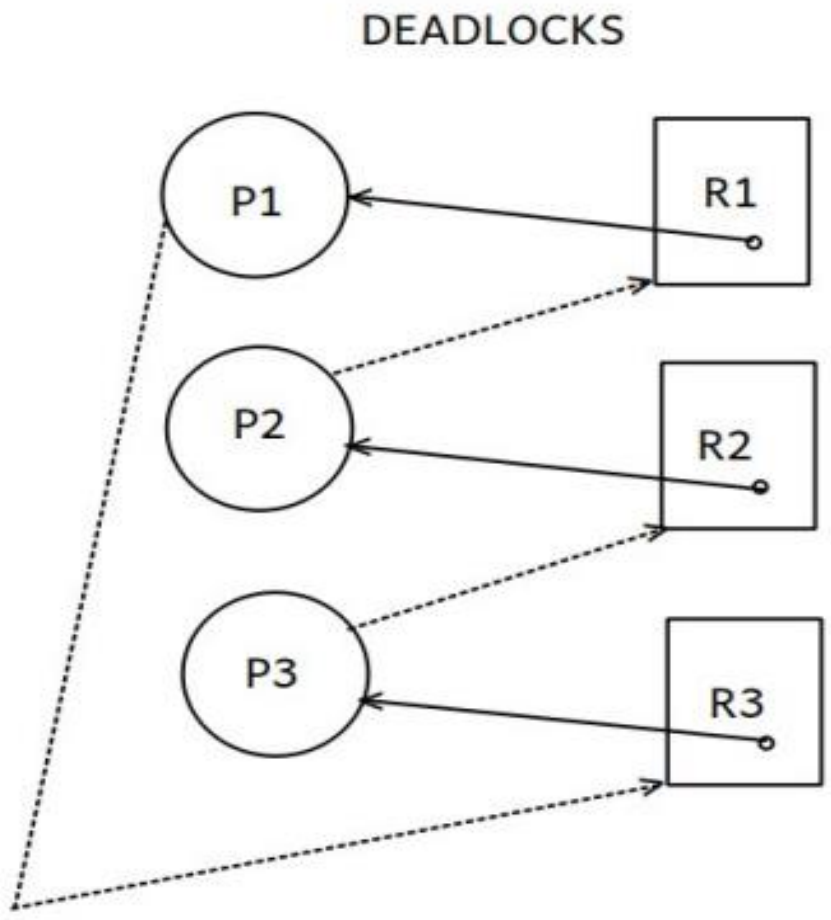
M: Mutex can be used only for mutual exclusion.

➢Deadlock:

- **There are four necessary and sufficient conditions to occur deadlock / characteristics of deadlock:**

1. **Mutual Exclusion:** at a time resource can be acquired by only one process.
2. **No Preemption:** control of the resource cannot be taken away forcefully from any process.
3. **Hold & Wait:** every process is holding one resource and waiting for the resource which is held by another process.
4. **Circular Wait:** if process P1 is holding resource and waiting for the resource held by another process P2, and process P2 is also holding one resource and waiting for the resource held by process P1.

Deadlock: Resource Allocation Graph



Three deadlock handling methods are there:

1. **Deadlock Prevention:** deadlock can be prevented by discarding any one condition out of four necessary and sufficient conditions.
2. **Deadlock Detection & Avoidance:** before allocating resources for processes all input can be given to deadlock detection algorithm in advanced and if there are chances to occur deadlock then it can be avoided by doing necessary changes.

❖There are two deadlock detection & avoidance algorithms:

1. **Resource Allocation Graph Algorithm**
2. **Banker’s Algorithm**

3. Deadlock Recovery:

- System can be recovered from the deadlock by two ways:

1. **Process termination:** in this method randomly any one process out of processes causes deadlock gets selected and terminated forcefully to recover system from deadlock.

   - Process which gets terminated forcefully in this method is referred as **a victim process**.

2. **Resource preemption:** in this method **control of the resource taken away forcefully from a process** to recover system from deadlock.

❑Starvation:

- The process not getting enough CPU time for its execution.
- Process is in ready state/queue. Reason: Lower priority (CPU is busy in executing high priority process).

❑Deadlock:

- The process not getting the resource for its execution.
- Process is in waiting state/queue indefinitely. Reason: Resource is blocked by another process (and there is circular wait).

Memory Management

❖Memory holds (digital) data or information.

-Bit = Binary Digit (0 or 1) => Internally it is an electronic circuit i.e. FlipFlop

-1 Byte = 8 Bits

-B, KB (2^10), MB (2^20), GB (2^30), TB (2^40), PB (2^50), XB (2^60), ZB (2^70)

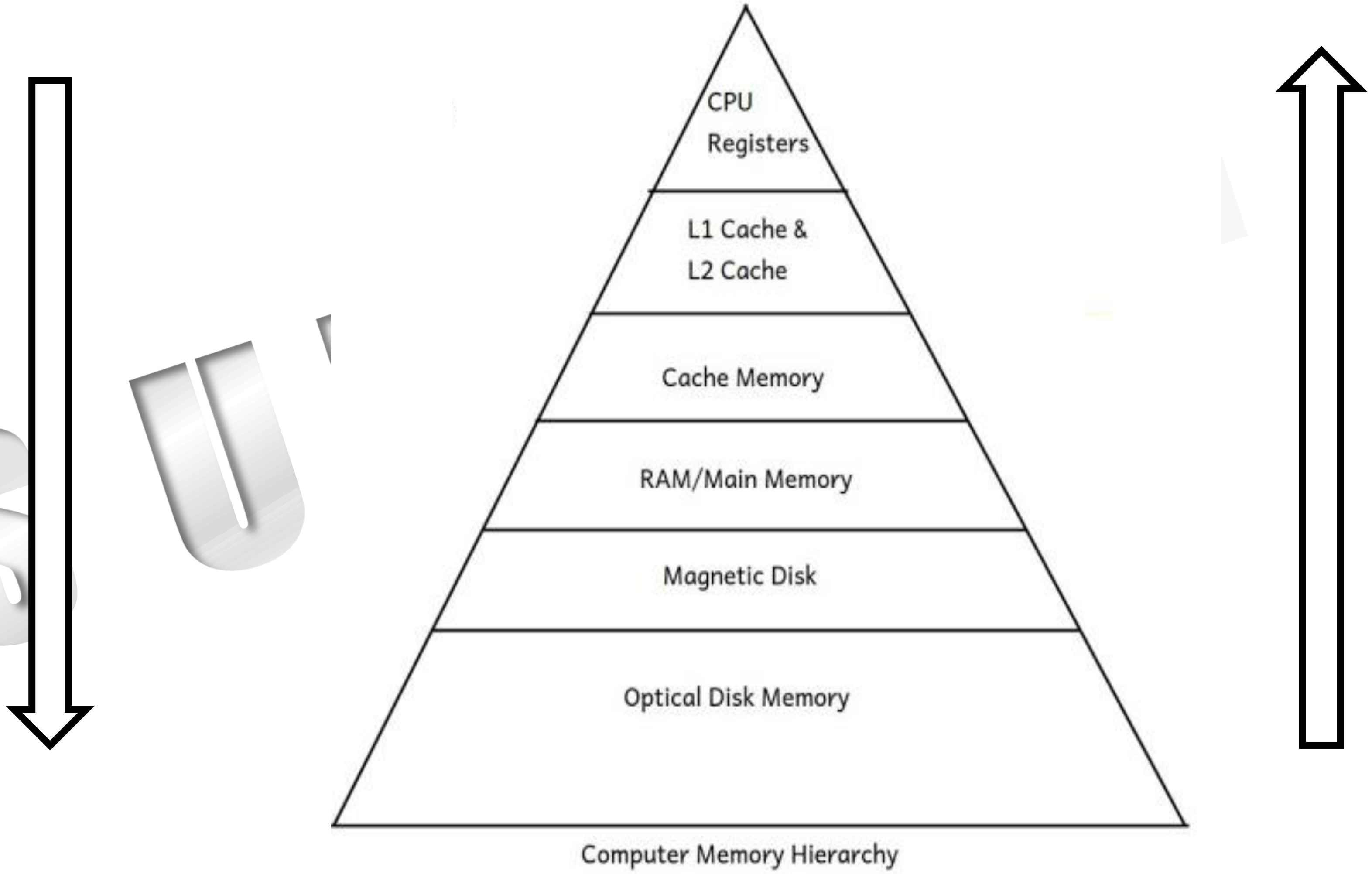
❖Volatile vs Non-volatile memory

-Volatile memory: The contents of memory are lost when power is OFF.

- Non-volatile memory: The contents of memory are retained even after power is OFF.

➢Computer Memory Technologies

As we go Top to Bottom
1.Acses speed decrease
2.Coast also decrease
3.Capacity increase



Computer Memory Hierarchy

Computer Fundamentals and Operating Systems

➢Computer Memory Technologies:

- **CPU Registers:** memory which is very close to the CPU are registers which is at the top in a computer memory hierarchy.
- Instructions and data currently executing by the CPU can be kept temporarily into the CPU registers.
- MAR, MBR, IOAR, IOBR, PC, SP, Accumulator etc...

➢Computer memory can be categorized into two categories as per its location:

- **Internal Memory & External Memory.** - Internal Memory: memory which is internal to the motherboard is referred as an internal memory.

 - e.g. CPU registers, L1 & L2 cache Cache memory, RAM.

- **External Memory:** memory which is external to the motherboard is referred as an external memory.

 - e.g. magnetic disk, optical disk, magnetic tape etc...

➢Computer Memory Technologies:

- Computer memory can also categorized into two categories**: Primary Memory & Secondary Memory**.

- **Primary Memory:** memory which can be accessible directly by the CPU is referred as primary memory, i.e. memory which can accessible by the CPU with the help of instruction set having with the CPU.

- e.g. CPU registers, L1 & L2 Cache, Cache Memory, RAM

- **Secondary Memory:** memory which cannot be accessed directly by the CPU is referred as secondary memory.

- e.g. Magnetic Disk, CD/DVD, PD etc..
- If the CPU want to access disk contents, first it gets fetched into the RAM and then it can be accessed by the CPU from RAM.

- As for an execution of every program RAM memory is must and hence **RAM is also called as Main memory.**

➢ Why there is a need of cache memory?

- As the rate at which the CPU can execute instructions is faster than the rate at which data can be accessed from the main memory, so even the CPU is very fast, with the same speed data do not gets fetched from the main memory for execution, hence due to this speed mismatch overall system performance gets down.
- To reduce speed match between the CPU and the main memory Cache memory (hardware) can be added between them and system performance can be increases by means **reducing speed mismatch.**

➢What is Cache Memory ?

- Cache memory is faster memory, which is a type RAM i.e. SRAM, in which most recently accessed main memory contents can be kept/stored in an associative manner i.e. in a key-value pairs.
- There are two types of RAM:

1. **DRAM (Dynamic RAM):** memory cells are made up of capacitors and transistor.

   - Main memory is as example of DRAM.

2. **SRAM (Static RAM):** memory cells are made up of transistor.

   - Cache Memory is an example of SRAM

Computer Fundamentals and Operating Systems

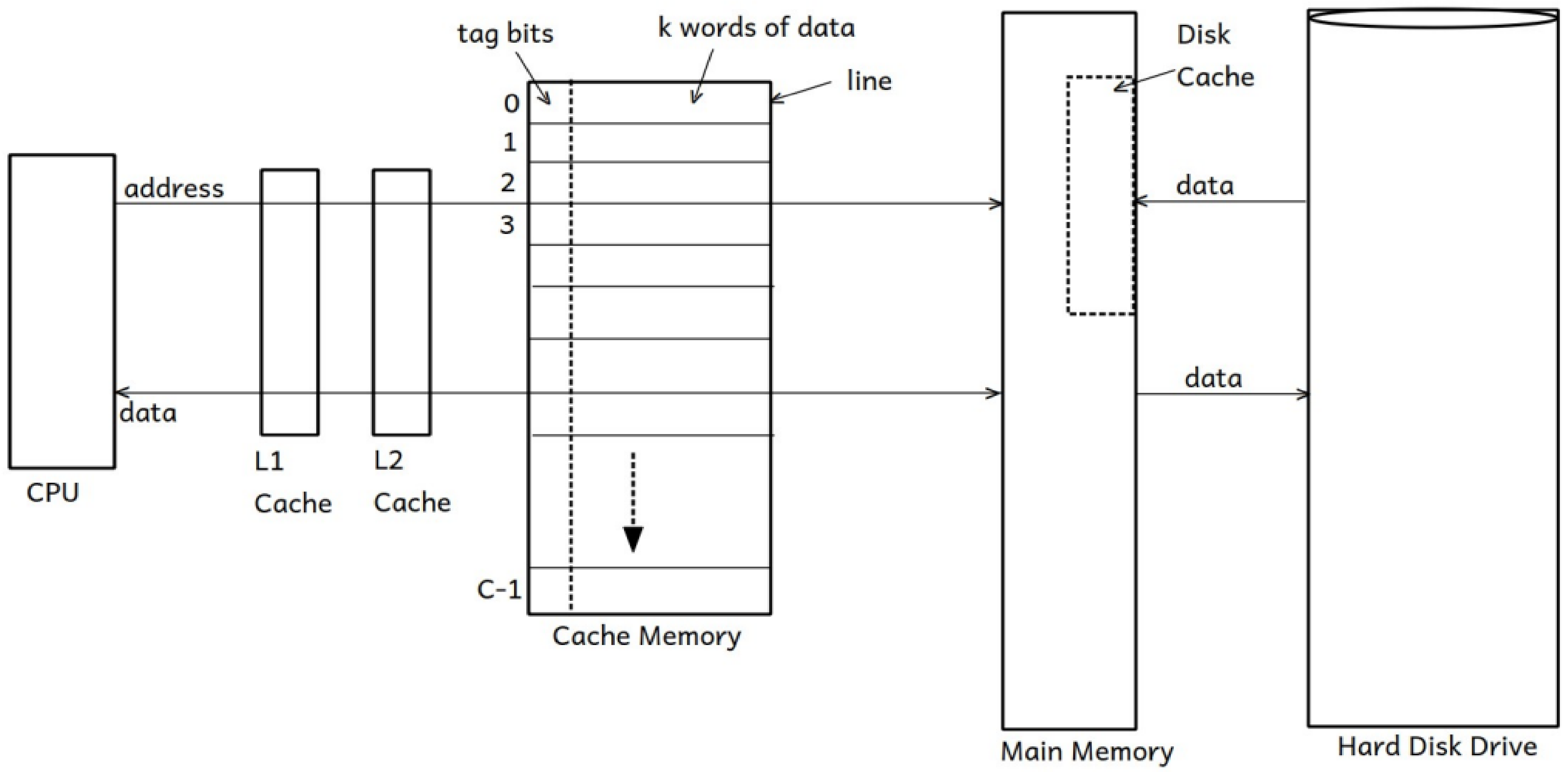
- **Cache Memory** has C no. of lines, whereas each line is divided into two parts, each line contains k words of data (recently accessed main memory contents) and its main memory addresses can be kept in few tag bits.

1. **First part of a line** : few tag bits contains main memory addresses of k words of data in that line
2. **Second part :** of a line contains k words of data.

- When the CPU want to fetch data from the main memory it requests for its address, and this requested address gets searched into the cache memory first, if requested addr is found in the cache memory then data also found in a cache memory, it is referred as **cache hit**, whereas if the requested address and hence data is not found in a cache memory then it is referred as a **cache miss,** in that data gets fetched from main memory and gets transferred to the CPU via cache memory only.

- Even after adding cache memory between the CPU and main memory, the rate at which the CPU can execute instructions is faster than the rate at which at which data can be accessed from cache memory, and hence to reduce speed mismatch between the CPU and cache memory one or more levels of cache memory i.e. L1 cache & L2 cache can be added between them.
- **Disk Cache:** it is purely a software technique in which portion of the main memory can be used as a cache memory in which most recently accessed disk contents can be kept in an associative manner, so whenever the CPU want to access data from hard disk drive it first gets searched into the disk cache.
- Disk cache technique is used to reduce speed mismatch between the CPU and Secondary memory

Computer Fundamentals and Operating Systems



SUNBEAM

Thank you!

Kiran Jaybhave

email – kiran.jaybhave@sunbeaminfo.com