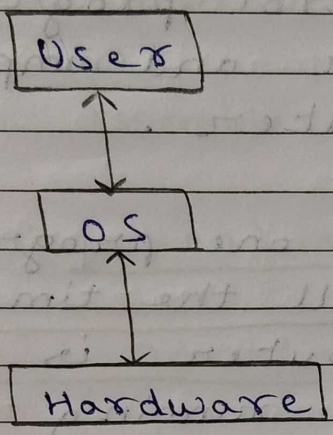
15/7/24

Miracle
Pg.No :
Date: / /20

Introduction

→ A program that acts as an intermediary between a user and computer hardware is known as operating system.



* Operating System goals :-

- Execute user programs
- Make computer system convinient to use.
- Use computer hardware in efficient manner.

* Components (IMP)

1. Hardware - CPU, memory, I/O devices.
2. Operating System - Windows, Ubuntu.
3. Application Program - MS Word, Web browser, VS code
4. User - people, machines, other computers.

- OS is a resource allocator.
→ Manage all resources
→ Efficient and fair resource use.

- OS is a control program.
→ prevent error and improper use of computer.

Kernel :- The one program running at all the times on the computer is Kernel.

→ It keeps on running behind operating system all the time.
→ Everything else is a
- system program or
- application program.

IMP

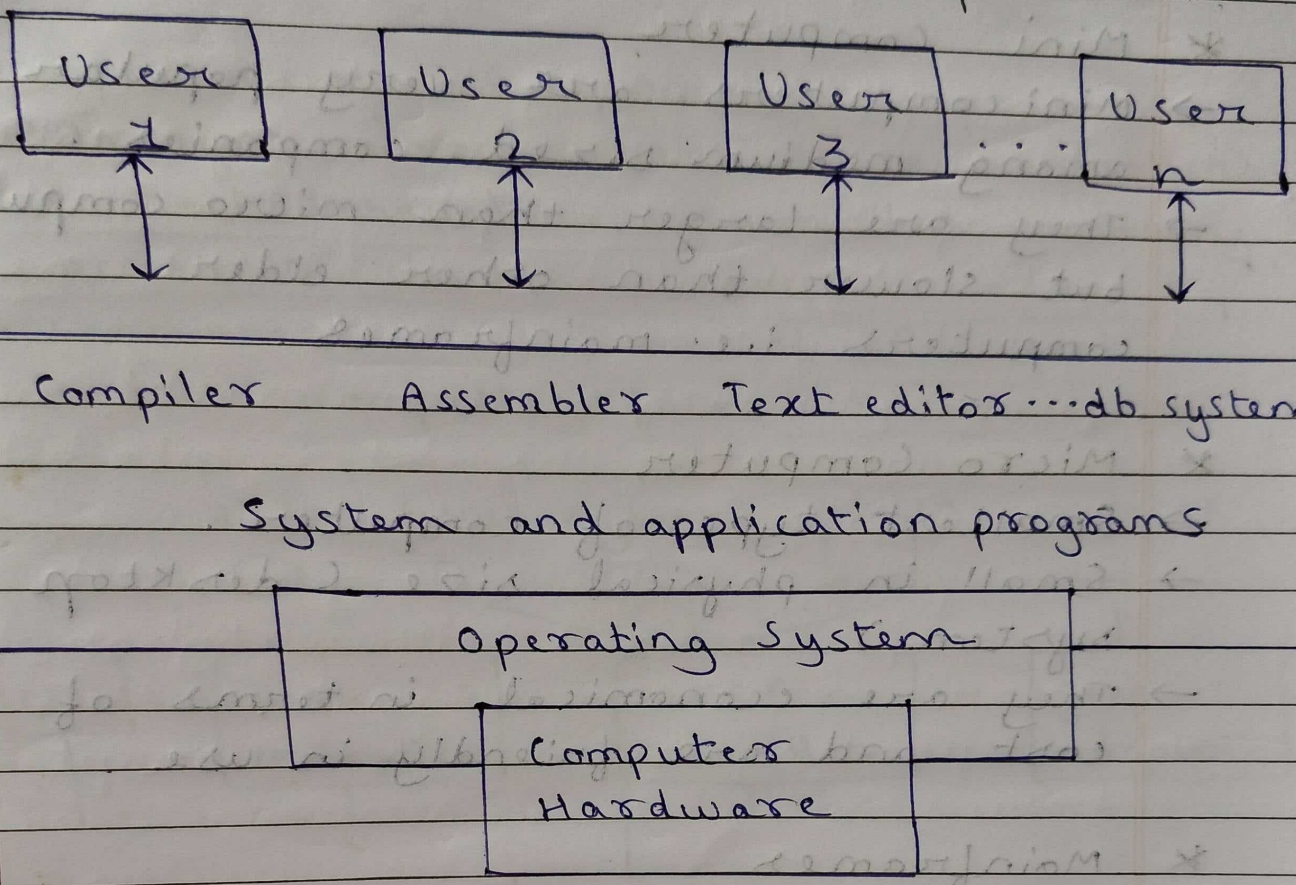
→ Bootstrap Program is loaded at power-up or reboot.
- Typically stored in ROM, or EPROM, generally known as firmware.
- initializes all aspects of system

* Static programs are always stored in EPROM.
* OS programs are stored in Bootstrap.

Miracle
Pg.No :
Date: / /20

Structure

* Flow of components of OS process.



* Difference between RAM and ROM.

RAM	ROM
→ Random Access Memory	→ Read only Memory
→ changable / Modify	→ cannot modified
→ Volatile	→ Non-volatile
→ Faster than ROM.	→ Slower than RAM.
→ Temperorary storage	→ Permanent storage.

Miracle
Pg.No :
Date: / /20

Types of Computer

* Mini computer
→ Mini computers are very popular among medium sized companies.
→ They are larger than micro computers but slower than other elder computers i.e. mainframes.

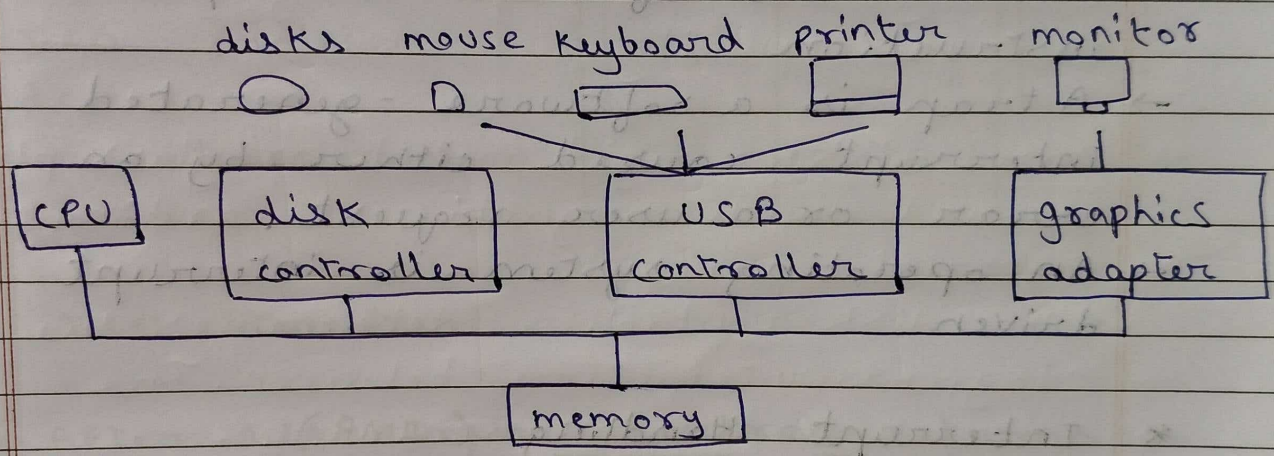
* Micro Computer
→ Smallest type of computer.
→ Small in physical size (desktop system)
→ They are economical in terms of cost and are friendly in use.

* Mainframes
→ Mainframes are bigger computers, capable of handling data processing.
→ They have larger storage and the speed of processing is also very high.

* Super Computer
→ Super computers have a speed of between 100 to 900 MIPS.
→ They are quite expensive and cost somewhere around 10 to 30 million dollars.
→ High levels of performance as compared to other computers.

* Computer System Organization

⇒ Computer system operation
- One or more CPUs, device controllers connect through common bus providing access to shared memory.
- Concurrent execution of CPUs and devices competing for memory cycles.



* Computer System Operation

→ I/O devices and the CPU can execute concurrently.
→ Each device controller is in charge of a particular device type.
→ Each device controller has a local buffer.
→ CPU moves data from/to main memory to/from local buffers.
→ I/O devices to local buffer of controller.
→ Device controller informs CPU that it has finished its operation by causing an interrupt.

* Common functions of Interrupt

→ Interrupt transfers control to the interrupt service routine generally through the interrupt vector, which containes the address of all the service routines.

→ Interrupt architecture must save the address of the interrupted instruction

→ A trap is a software-generated interrupt caused either by an error or a user request.

→ An operating system is interrupt driven.

* Interrupt Handling (IMP)

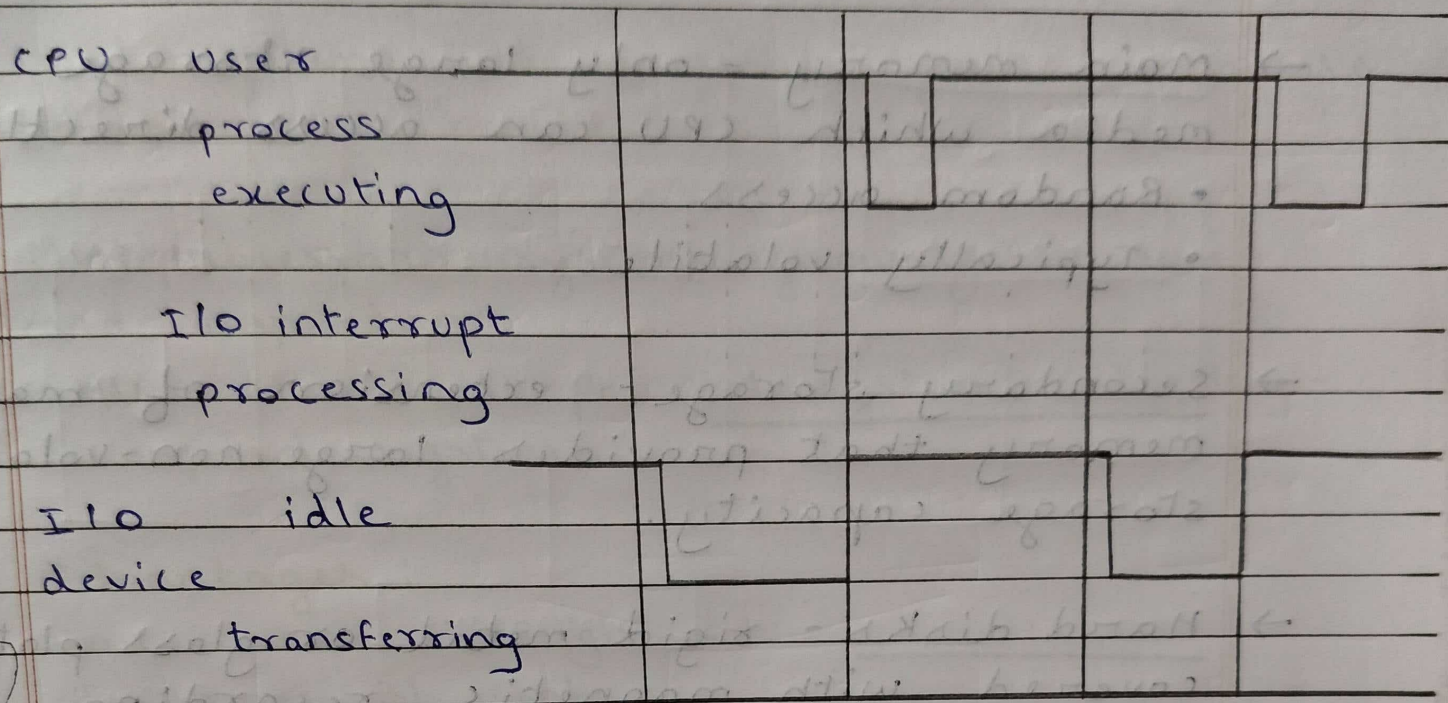
→ The operating system preserves the state of the CPU by storing registers and the program counter.

→ Determines which type of interrupt has occured :
- polling
- vectorred interrupt system.

→ Separate segments of code determine what action should be taken for each type of interrupt.

Miracle
Pg.No :
Date: / /20

Interrupt Timeline



* I/O Structure

→ After I/O starts, control returns to user program without waiting for I/O completion.

- System call - Request to the OS to allow user to wait for I/O completion.
- Device status table contains entry for each I/O device indicating its type, address and state.
- Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt.

Date: / /20

* storage structure

→ Main memory - only large storage media which CPU can access directly.
- Random access
- Typically volatile

→ Secondary storage - extension of main memory that provides large non-volatile storage capacity.

→ Hard disks - rigid metal or glass platter covered with magnetic recording material.
- The disk controller determines the logical interaction between the device and the computer.

→ Solid-state disks - faster than hard disks, non-volatile.
- Various technologies
- Becoming more popular.

* Computer System Architecture

→ Most systems use a single general purpose processor.

→ Multiprocessors systems growing in use and importance
- Also known as parallel systems, tightly-coupled systems.

Advantages :
1. Increased throughout
2. Economy of scale
3. Increased reliability

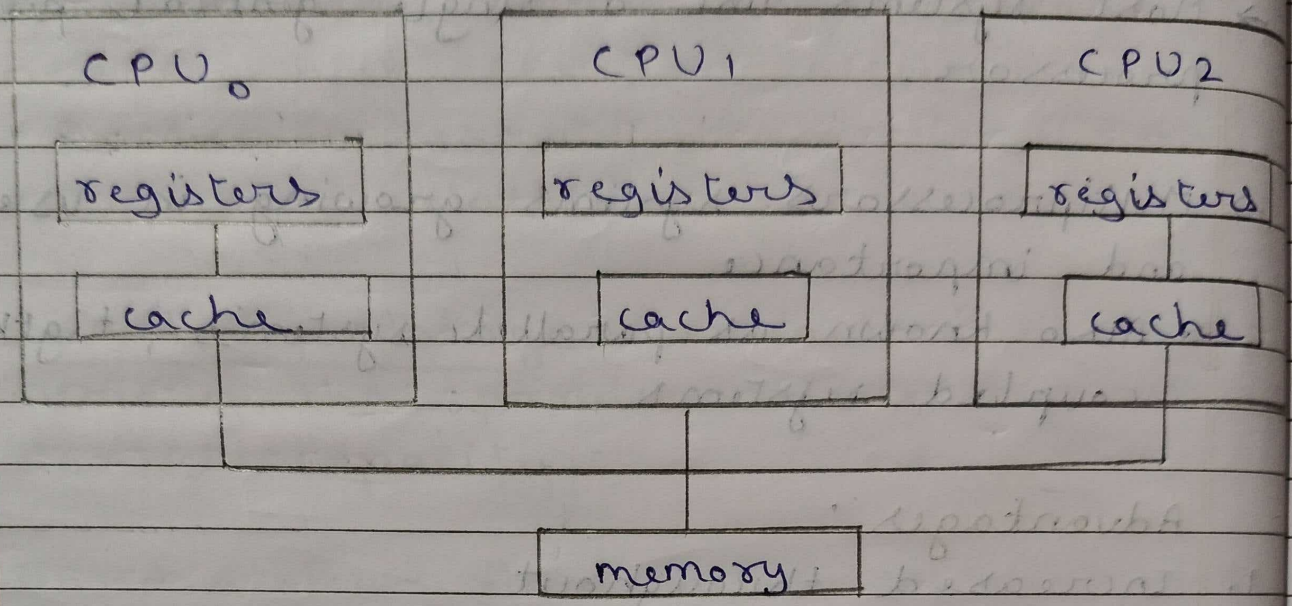
* Two types :

1. Asymmetric Multiprocessing - each processor is assigned a special task.
→ No hierarchy is followed.

2. Symmetric Multiprocessing - each processor performs all tasks.
→ Hierarchy is followed.

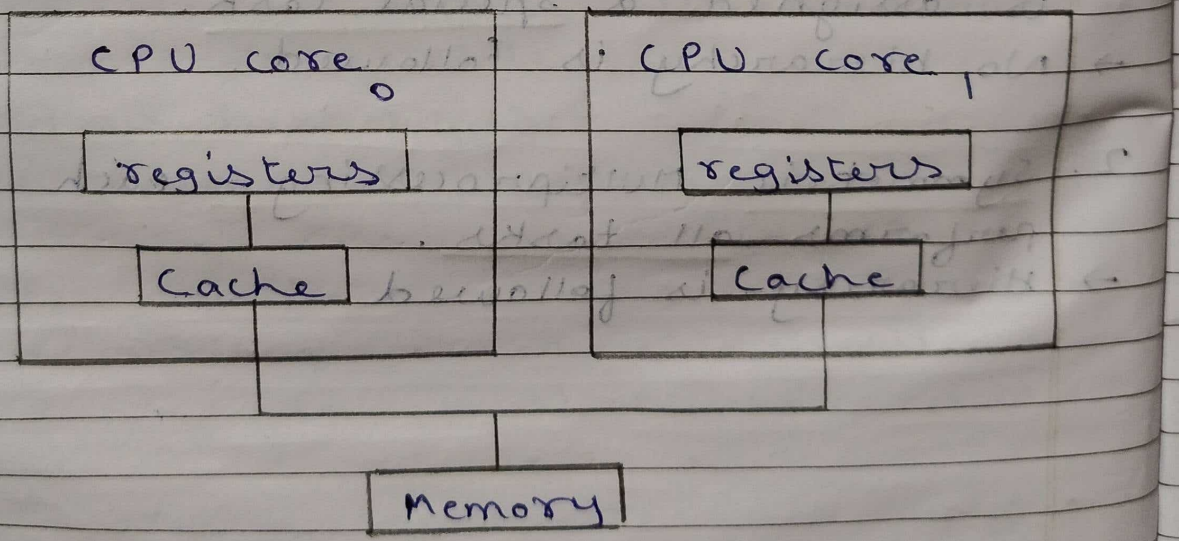
Date: / /20

* Symmetric Multiprocessing Architecture

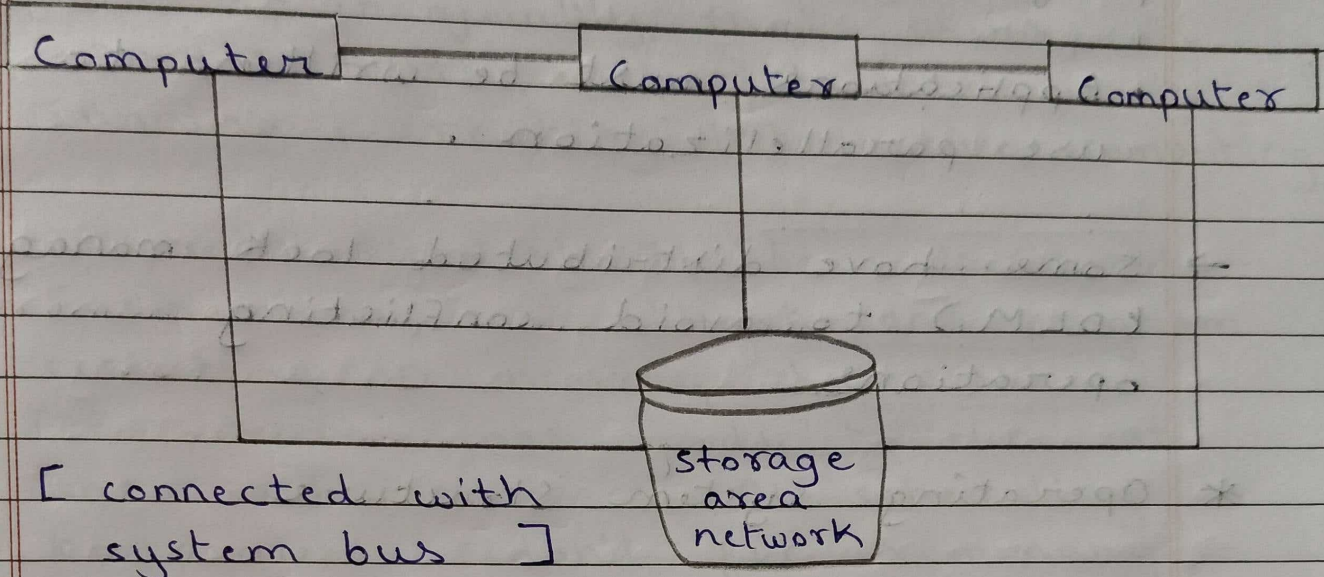


* A dual-core design

→ Multi-chip and multi core.
→ Systems containing all chip.



* Clustered systems



[connected with system bus]

→ The Multiprocessor systems, but multiple systems working together:

- Usually sharing storage via a storage area network (SAN)
- Provides a high availability service which survives failures

→ Asymmetric clustering has one machine hot-standby mode

→ Symmetric clustering has multiple nodes running applications, monitoring each other.

→ Some clusters are for high-performance computing (HPC).

- Applications must be written to use parallelization.

→ Some have distributed lock manager (DLM) to avoid conflicting operations.

* Operating system structure

→ Multiprogramming (Batch system) needed for efficiency
- Single user cannot keep CPU and I/O devices busy at all times.
- Multiprogramming organizes jobs (code and data) so CPU always has one to execute.
- A subset of total jobs in system is kept in memory.
- One job selected and run via job scheduling
- When it has to wait (for I/O for example) OS switches to another job.

→ Timesharing (multitasking) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating interactive computing.

- Response time should be < 1 second.
- Each user has atleast one program executing in memory ⇒ process.
- If several jobs ready to run at the same time ⇒ CPU scheduling.
- If processes don't fit in memory, swapping moves them in and out to run.
- Virtual memory allows execution of processes not completely in memory

Storage Hierarchy

→ storage systems organized in hierarchy
- Speed
- Cost
- Volatile

→ Caching - copying information into faster storage system; main memory can be viewed as a cache for secondary storage.

→ Device Driver: for each device controller to manage I/O
- Provides uniform interface between controller and kernel

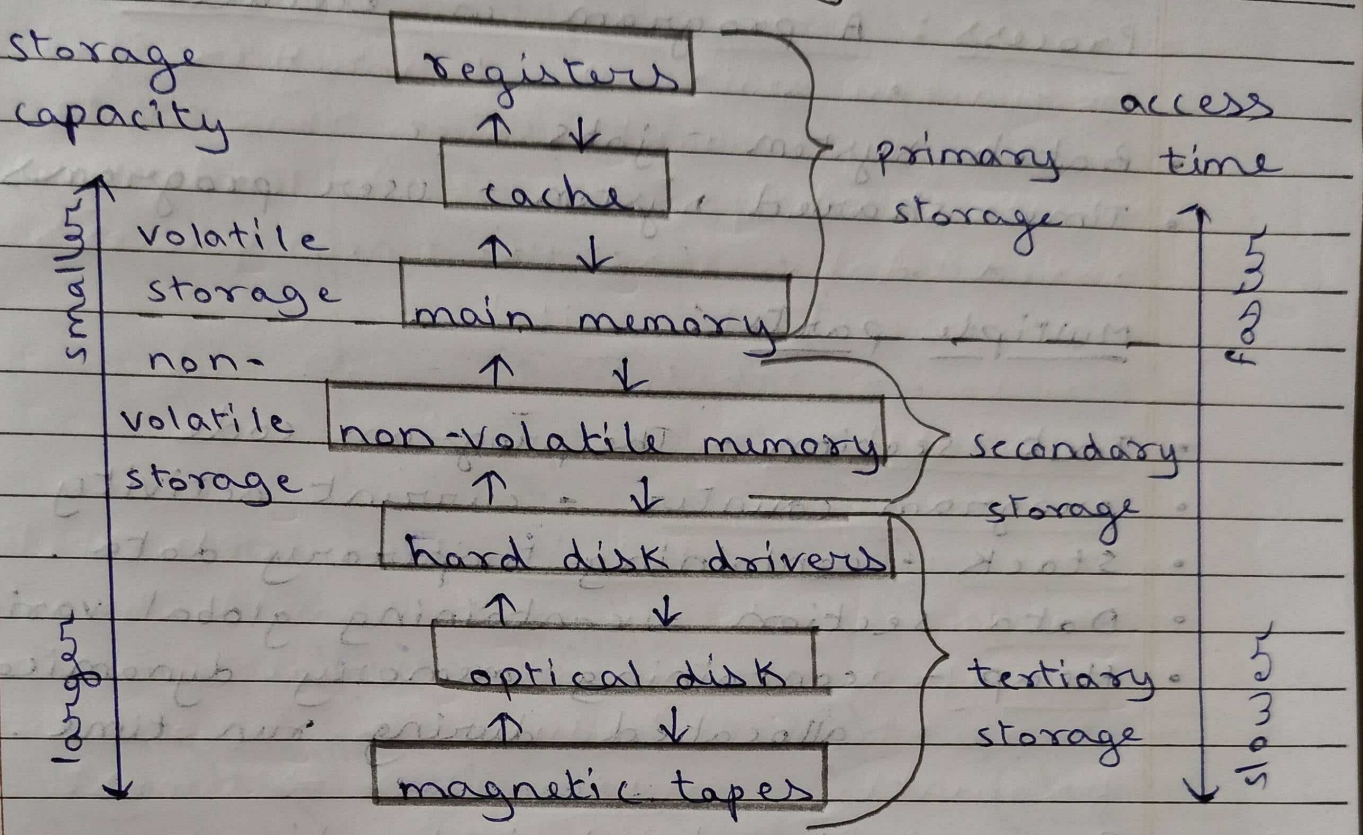
* Caching

→ Information in use copied from slower to faster storage tempurarily.

→ Faster storage (cache) checked first to determine if information is there
- If it is, information used directly from the cache (fast)
- If not, data copied to cache and used there.

Miracle
Pg.No :
Date: / /20

* Storage Device Hierarchy



Unit 2 : Processes

Process : A program in execution

- Batch system - jobs
- Time - shared system - user programs

Multiple parts :

- Text section - The program code
- program counter - current activity
- Stack - containing tempirary data.
- Data section - containing global variable
- Heap - containing memory dynamically allocated during run time.

Note: Program : set of instructions.

Process : set of instructions in execution which use CPU.

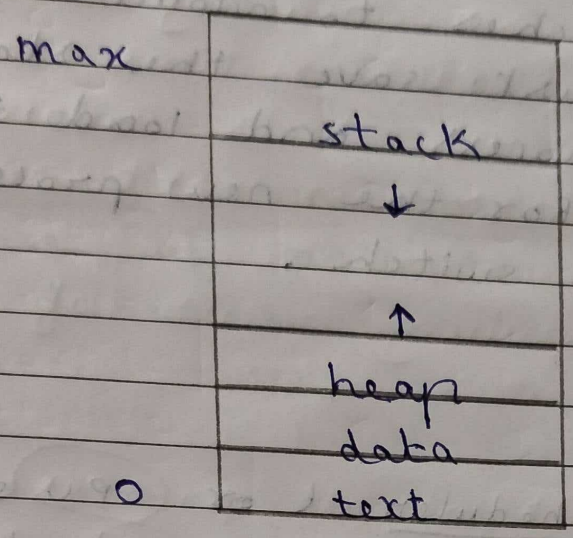
* Process Concept

→ Program is a passive entity stored on disk (executable file); Process is active.

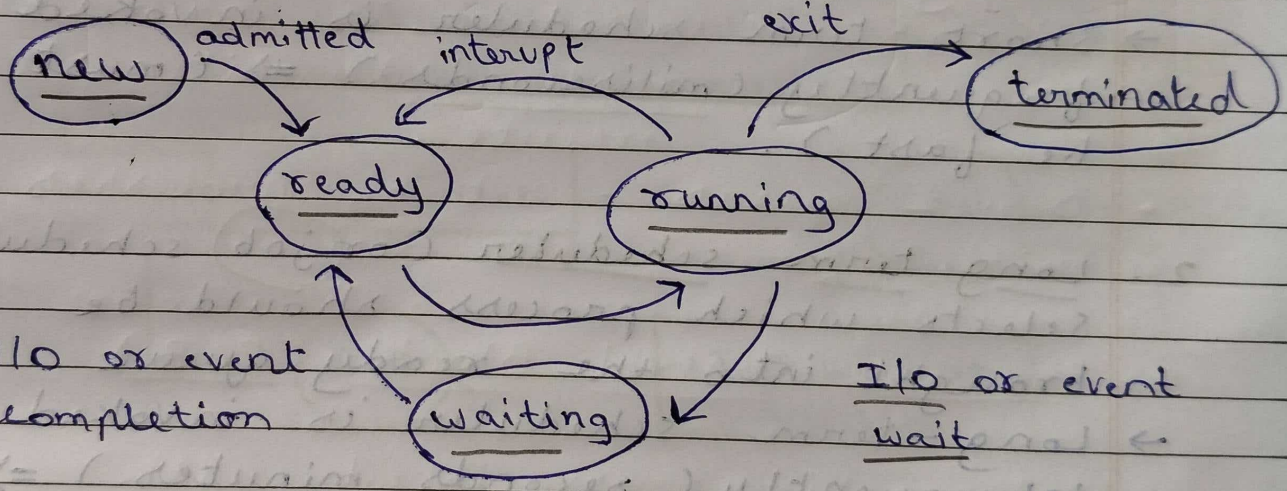
→ Execution of program started via GUI mouse clicks, command line entry, etc.

- Program becomes process when executed file in memory.

* Process in Memory



* Process state



Note: Scheduler is responsible to send program from one state to another.

* Types of scheduler :

(i) Long - term (moves data from secondary to main memory)
(ii) Short - term (ready to running)
(iii) Midium - term (Main memory / primary to secondary or secondary to primary).

* Context Switching

→ When CPU switches to another process the system must save the state of the old process and load the saved data for the new process via a context switch.

* Schedulers

1. Short term scheduler (or CPU scheduler) selects which process should be executed next and allocates CPU.
   → Short-term scheduler is invoked frequently (milliseconds) ⇒ (must be fast).

2. Long term scheduler (or job scheduler) selects which process should be brought into the ready queue.
   → Long-term scheduler is invoked infrequently (seconds, minutes) ⇒ (may be slow).
   → It controls the degree of multiprogramming.

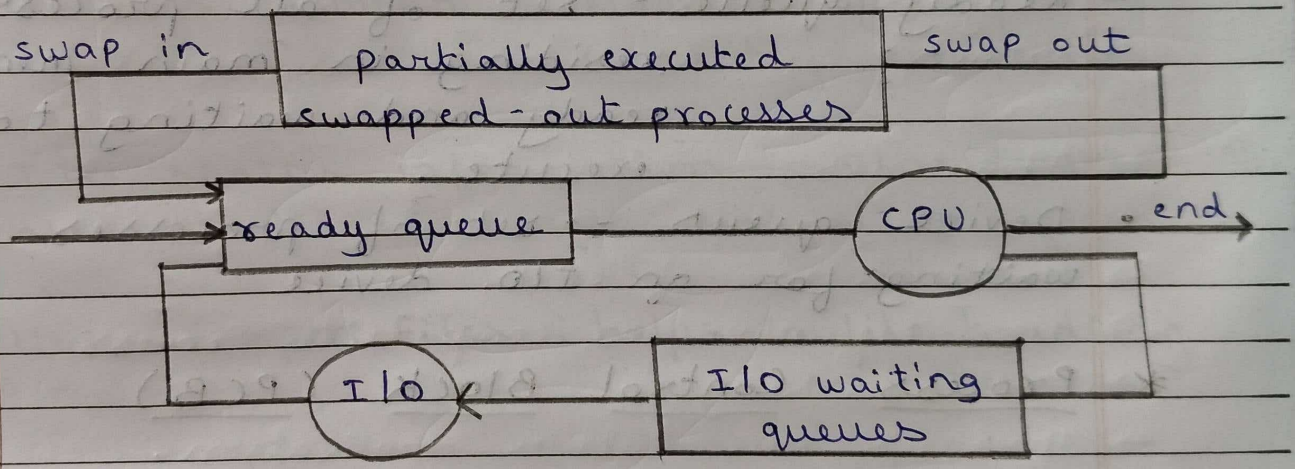
Processes can be described as either:

- I/O bound process :- spends more time doing I/O than computations, many short CPU bursts.

- CPU bound process :- spends more time doing computations; few very long CPU bursts.

3. Medium-term scheduler can be added if degree of multiple programming needs to decrease.

→ Remove process from memory, store on disk, bring back from disk to continue execution : swapping.



* Process State

As process executes, it changes state

(i) new : The process is being created.
(ii) running : Instructions are being executed.
(iii) waiting : The process is waiting for some event to occur.
(iv) ready : The process is waiting to be assigned to a processor.
(v) terminated : The process has finished execution.

* Process Scheduling

→ Maximize CPU use, quickly switch processes onto CPU for time sharing.
→ Process scheduler selects among available processes for next execution on CPU.
→ Maintains scheduling queues of processes

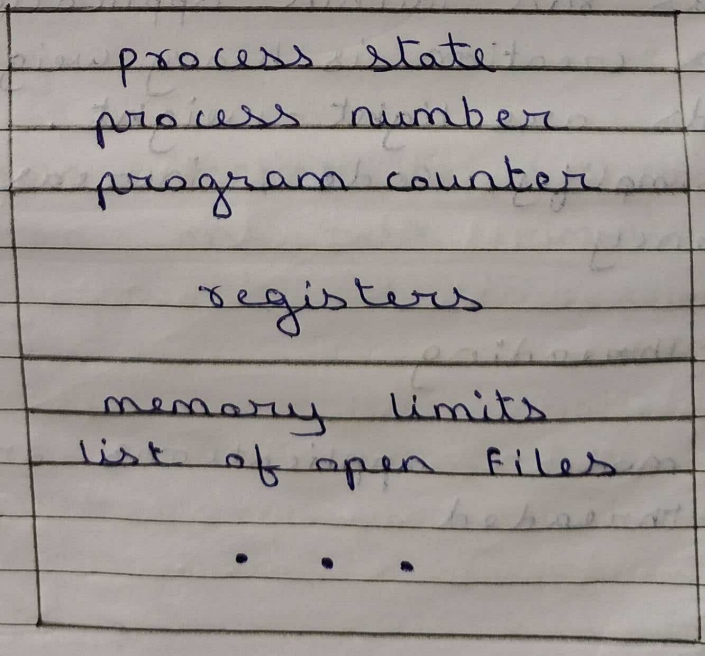
- Job queue - set of all processes in the system
- Ready queue - set of all processes residing in main memory, ready and waiting to execute.
- Device queues - set of processes waiting for an I/O device

* Process Control Block (PCB)

Information associated with each process (as called task control block)

- Process state : running, waiting, etc.
- Program counter : location of instruction to next execute.
- CPU registers : smallest unit of storage.

- CPU scheduling information : priorities, scheduling queue pointers.
- Memory - management information : memory allocated to the process.
- Accounting information : CPU used, clock time elapsed since start, time limits.
- Pid - unique id of process.
- Process Number : Number of a process currently executing.
- I/O status information : I/O devices allocated to process, list of open files.
- list of open files : helps in file handling.



Threads

→ A thread is a fundamental unit of CPU utilization that forms the basis of multi-threaded computer systems.

→ Thread is a light weight process.

→ It also compriises of :

- Thread id
- Program counter
- Register
- Stack

* Advantages :

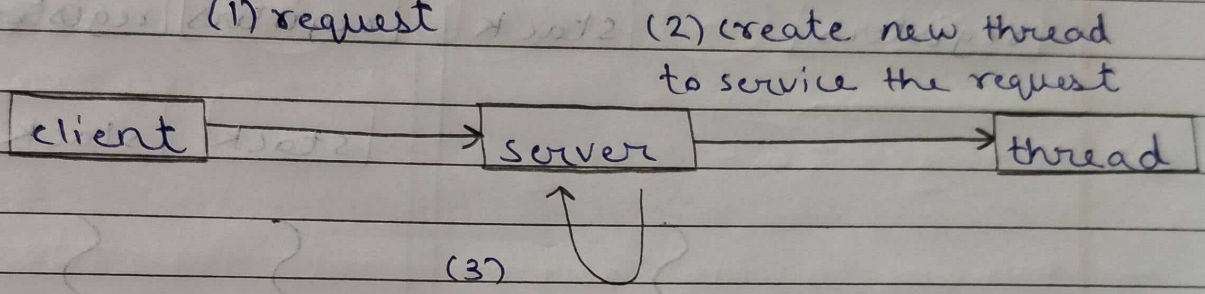
→ Maximum use of CPU.
→ less expensive as compare to Process
→ Multi-tasking
→ Threads run within application.
→ Process creation is heavy weight whereas threads are light-weight.
→ can simplify code, increase efficiency

* Multi-Threading

→ Most modern applications are multi-threaded.

→ Threads run within application.
→ Multiple tasks with the application can be implemented by separate threads

* Multi-threaded server architecture



* Benefits :

- responsiveness
- resource sharing
- Economy
- Scalability

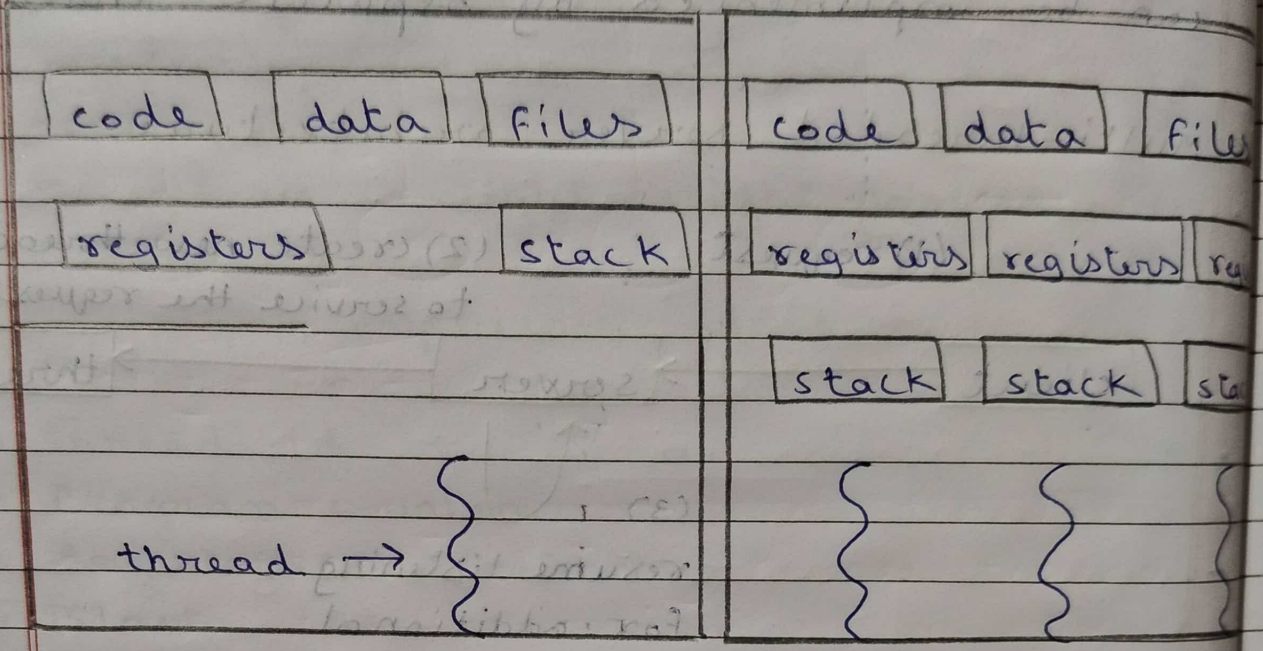
• Parallelism implies a system can perform more than one task simultaneously.

• Concurrency

Date: / /20

* Single and Multi-threaded processes

Diagram



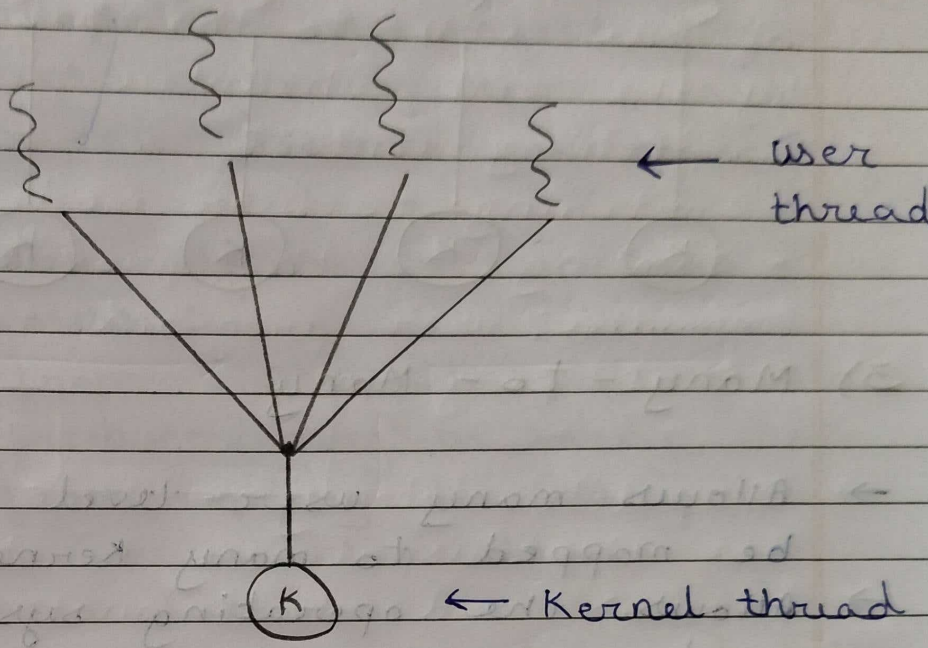
* Multi-threading models

1) many to one
2) one to one
3) many to many

1) Many-to-one

→ Many user-level threads mapped to single Kernel thread
→ One thread blocking causes all to block.
→ Multiple threads may not run in parallel on multicore system because only one proceed may be in kernel at a time.

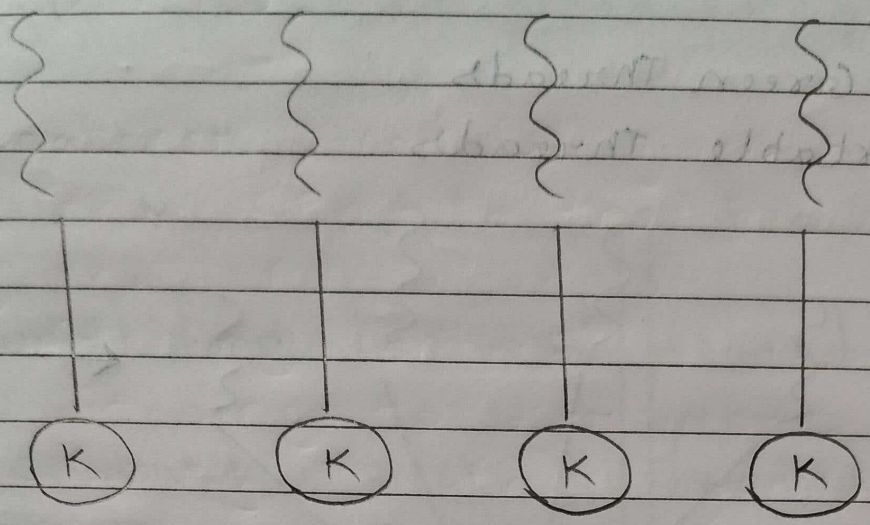
Examples:
- Solaris Green Threads
- GNU Portable Threads



2) One-to-one

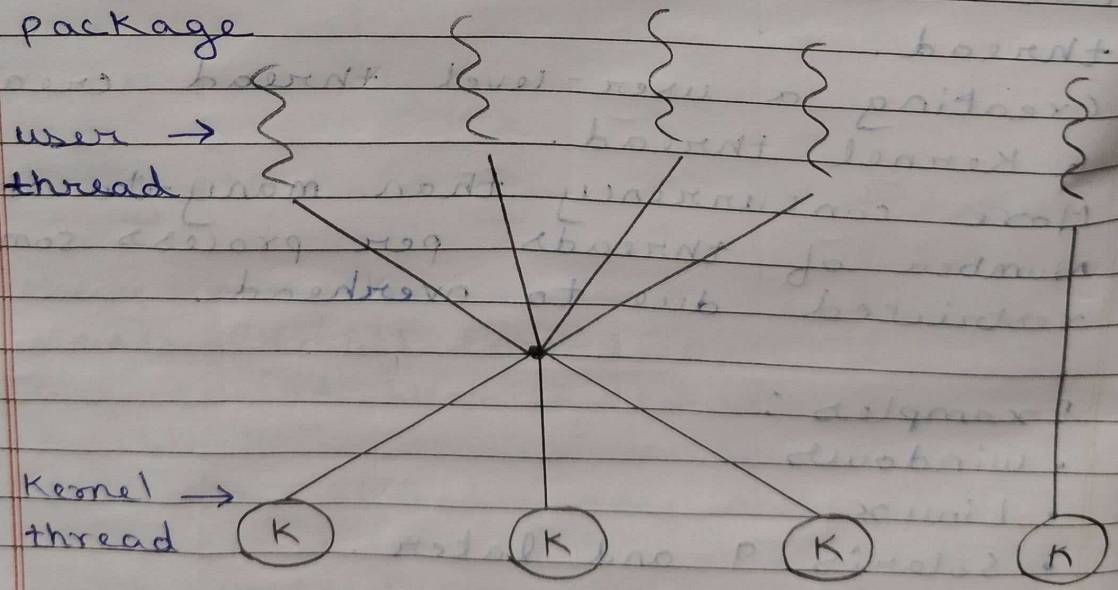
→ Each user-level thread maps to Kernel thread.
→ Creating a user-level thread creates a Kernel thread.
→ More concurrency than many-to-one.
→ Number of threads per process sometimes restricted due to overhead.

Examples:
- Windows
- Linux
- Solaris 9 and later.



3) Many - to - Many

→ Allows many user level threads to be mapped to many Kernel threads
→ Allows the operating system to create a sufficient number of Kernel threads
→ Solaris prior to version 9
→ Windows with the ThreadFiber package



CPU Scheduling

* CPU scheduler

→ The CPU scheduler selects from among the processes in ready queue, and allocates a CPU core to one of them.

→ CPU scheduling decisions may take place when a process:

1. switches from running to waiting state.
2. Switches from running to ready state.
3. Switches from waiting to ready.
4. Terminates.

* Preemptive and Non-preemtive scheduling

→ When scheduling takes place only under circumstances 1 and 4, the scheduling scheme is nonpreemtive.

→ Otherwise, it is preemtive.

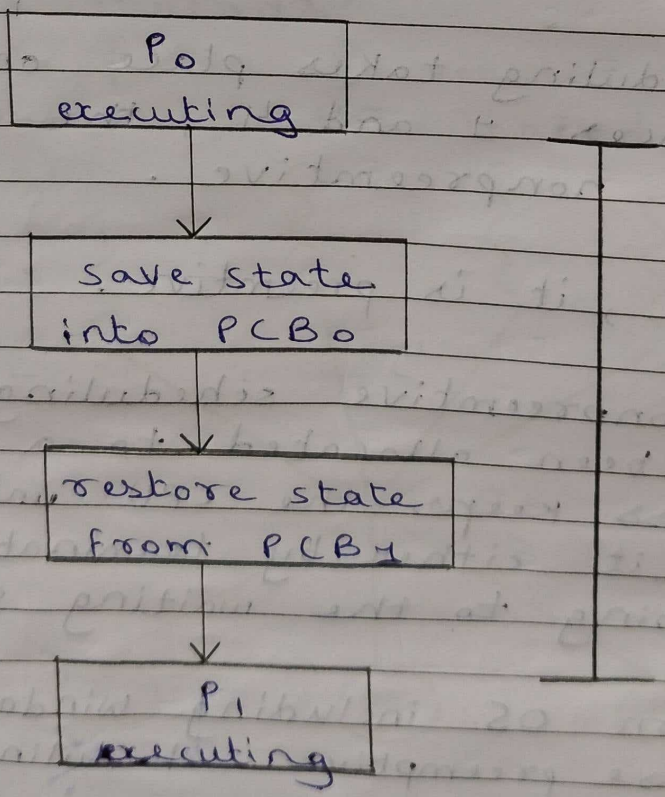
→ Under Nonpreemtive scheduling, once the CPU has been allocated to a process, the process keeps the CPU until it releases it either by terminating or by switching to the waiting state.

→ All modern OS including Windows, Linux, MacOS use preemptive scheduling algorithms.

* Dispatcher

→ Dispatcher module gives control of the CPU to the process selected by the CPU scheduler; this involves:
- switching context
- switching to user mode.
- Jumping to the proper location in the user program to restart that program.

→ Dispatch latency - time it takes for the dispatcher to stop one process and start another running.



* Scheduling Criteria

(i) CPU utilization - Keep the CPU as busy as possible.

(ii) Throughput - # of processes that complete their execution per time unit

(iii) Turnaround time - amount of time to execute a particular process.

Formula :- Completion time - Arrival time.

∴ TAT = CT - AT

(iv) Waiting time - Amount of time a process has been waiting in the ready queue.

Formula :- Turn around time - Burst time

∴ WT = TAT - BT

(v) Response time - amount of time it takes from when a request was submitted until the first response is produced.

Date: / /20

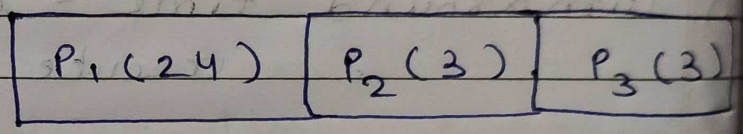
Formula :- CPU allocation time − Arrival time

∴ RT = CAT − AT [BT − AT]

* Burst time - time a program takes to execute.

Example :

Gantt chart -

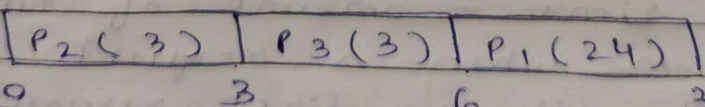


(FCFS)

	AT	BT	Turnaround	Waiting	Response Time
P_1	0	24	24	24	0
P_2	0	3	27	27	24
P_3	0	3	30	30	27
Avg:			27	17	

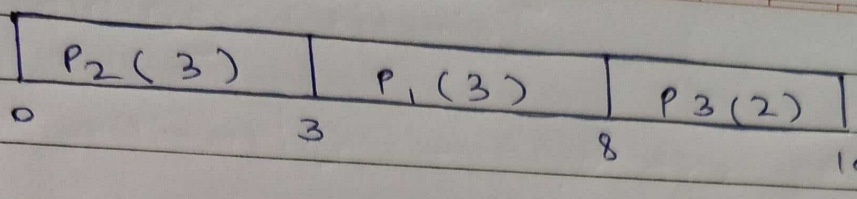
Q- (SJF)

	BT	AT	TAT	TAT	WT	RT
P_2	3	0	27	3	0	0
P_3	3	0	27	6	3	3
P_1	24	0	6	30	6	6
Avg:				13	3	3



Note : SJF : shortest job first
FCFS : First come First serve.

Q-



	AT	BT	TAT	WT	RT
P_1	1	5	7	2	4
P_2	0	3	3	0	3
P3	2	2	8	6	6
Avg			6	2.67	2.33