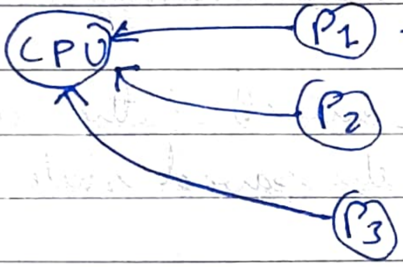
Goals of OS:-

-> Maximum CPU utilization
-> Less process startration
-> Higher priority job execution

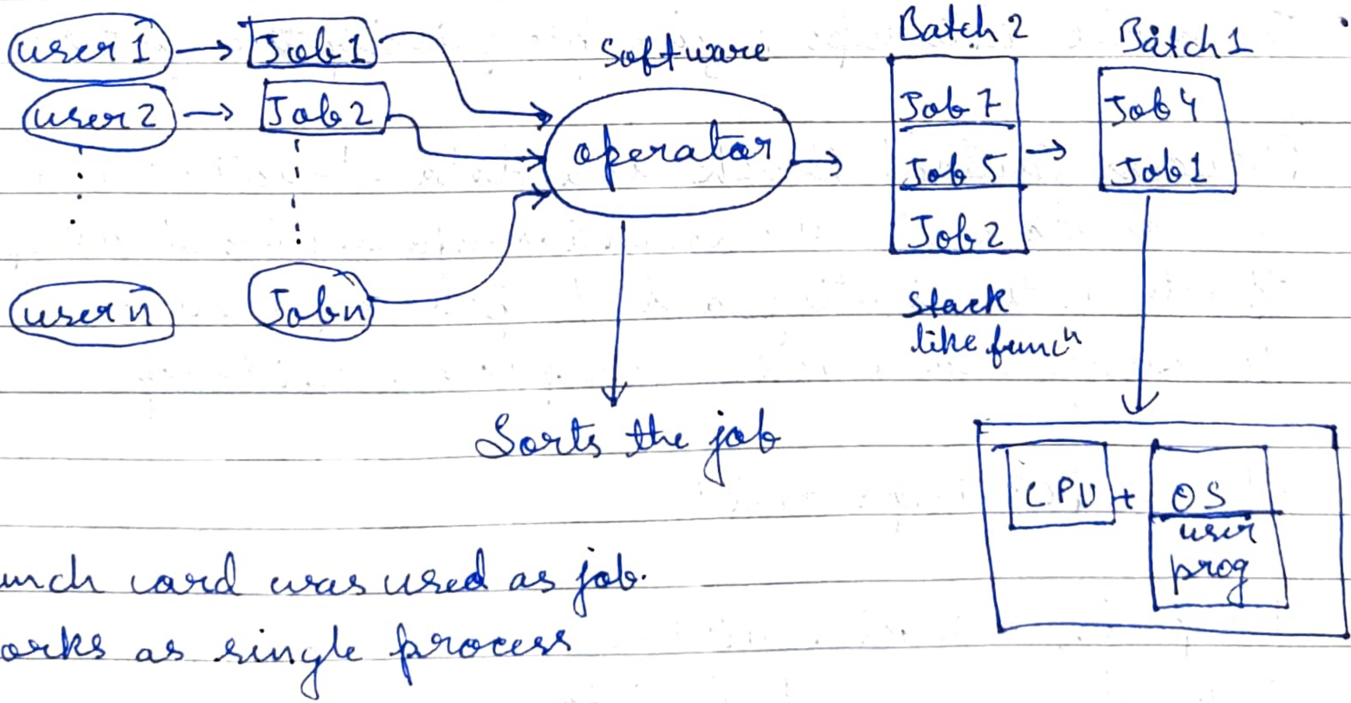
Types of operating system:-

1) Single process operating system [MS DOS 1981]



-> Goes to I/O
when it comes & executed
then P_2 will start
other wise no.
One process at one time

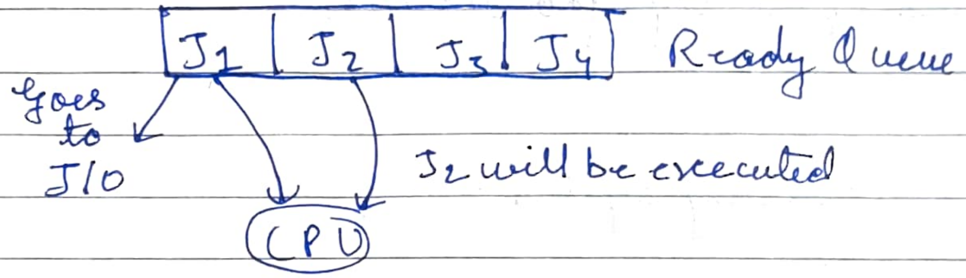
2) Batch Processing operating system [ATLAS, 1950-1960]



Punch card was used as job.
Works as single process

3) Multi Programming Operating System [THE, Dijkstra 1960s]

It is single CPU



Context Switching is the process by which a CPU stops executing one process or thread and starts executing another.
It involves saving the data (context) of the currently running process & loading the saved state of the next process to run.

why context switching needed?

- To multitask & share CPU time along multiple processes
- To switch from user mode to kernel mode or vice-versa
- To handle interrups & give ~~pio~~ priority to urgent tasks.

Process Control Block is a data structure used by the OS to store all the essential information about a process.
It acts as the identity card of a process, allowing the OS to manage and track each process efficiency

Why is PCB important?

Helps in context switching
Supports process scheduling
Enables process tracking, suspension and resumption.

Ex:- When a process is paused, the OS saves its state in the PCB. Later, when the process resumes, the OS retrieves info from the PCB & continues execution seamlessly.

1) Multitasking Operating System [CTSS, MIT early 1960s]

Single CPU

An OS that allows a computer to run two or more programs simultaneously by managing CPU time and resources among them efficiently

How it works?

- The CPU executes part of one program; then context switches to another.
- By switching rapidly between process, it creates the illusion that multiple tasks are running at the same time

Types of Multitasking :-

1) Preemtive Multitasking

- The OS decides how long each process runs
- Ensures fairness & responsiveness

2) Cooperative Multitasking

- Each process voluntarily yields control to others
- Less reliable; one ~~hang~~ misbehaviour process whole system gets hang

Types of preemtive multitasking :-

1) Robin Round (RR)

- Each process get a fixed time slice
- If it doesn't finish, it's preemted and moved to the back of the queue
- Fair & simple, but not ideal for high-priority tasks.

2) Priority based preemtive scheduling

- Each process is assigned a priority
- The CPU always picks the highest-priority ready process
- Can cause starvation of low-priority processes.

3) Shortest remaining time first (SRTF)

- Extension of shortest job first, but preemtive
- If a new process arrives with shorter remaining time than the current one, the current one is preemted
- Can be complex to implement due to need for accurate time predictions.

4) Multilevel Queue Scheduling

- Processes are grouped into different queue based on priority or type.
- Each queue has its own scheduling algorithm.
- Higher-priority queues can preemt lower ones.

5) Multiple level Feedback Queue

- Avanced version of multi level queue
- Allows proccesses to move b/w queues based on age

and behaviour
- Preemtive by design to prioritize interactive and short tasks.

Types of Non-Preemtive

1> First come first serve (FCFS)

- Processes are executed in the order they arrive.
- Simple queue-based scheduling
- No prioritization, long job delay short ones (convoy effects)

2> Shortest job first

- The process with the smallest burst time is selected next
- Optimal in terms of average waiting time
- Difficult to know exact burst time in advance.

3> Priority Scheduling

- Each process is assigned a priority
- If two have the same priority, use FCFS
- Starvation possible for low-priority tasks.

4> Multilevel Queue

- Processes are permanently assignment to a specific queue
- Each queue has a scheduling algorithm (like FCFS)
- No preemption b/w queues.

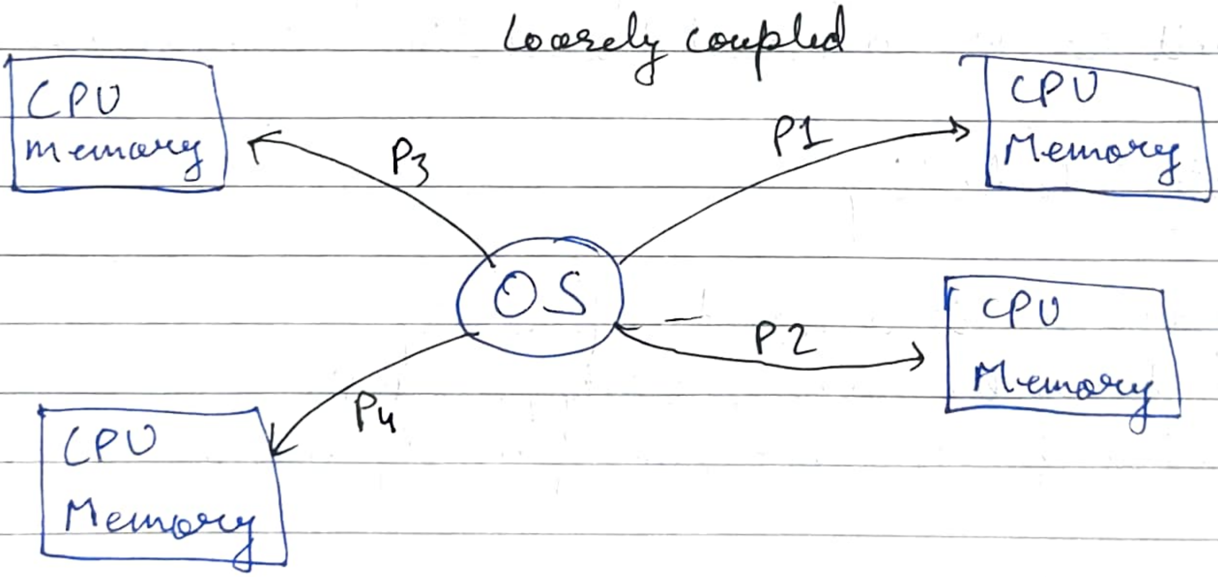
5) Multi-processing operating system [Windows NT]

- More than 1 CPU in a single computer
- Increases reliability, 1 CPU fails other can work
- ~~Bettern~~ Better throughput
- Lesser process starvation

6) Distributed Operating System [LOCUS]

- OS manages many bunches of resources,
  >=1 CPUs, >=1 memory >=1 GPUs etc
- Loosely connected ~~ato~~ autonomus, interconnected computer nodes
- Collection of independent, networked, communicating and physically separate computational nodes

Loosely coupled



P1, P2, P3, P4

7) Real time operating system [ATCS]

- Real time-error free, computations with in tight time boundaries.
- Air traffic control system, ROBOTS etc.