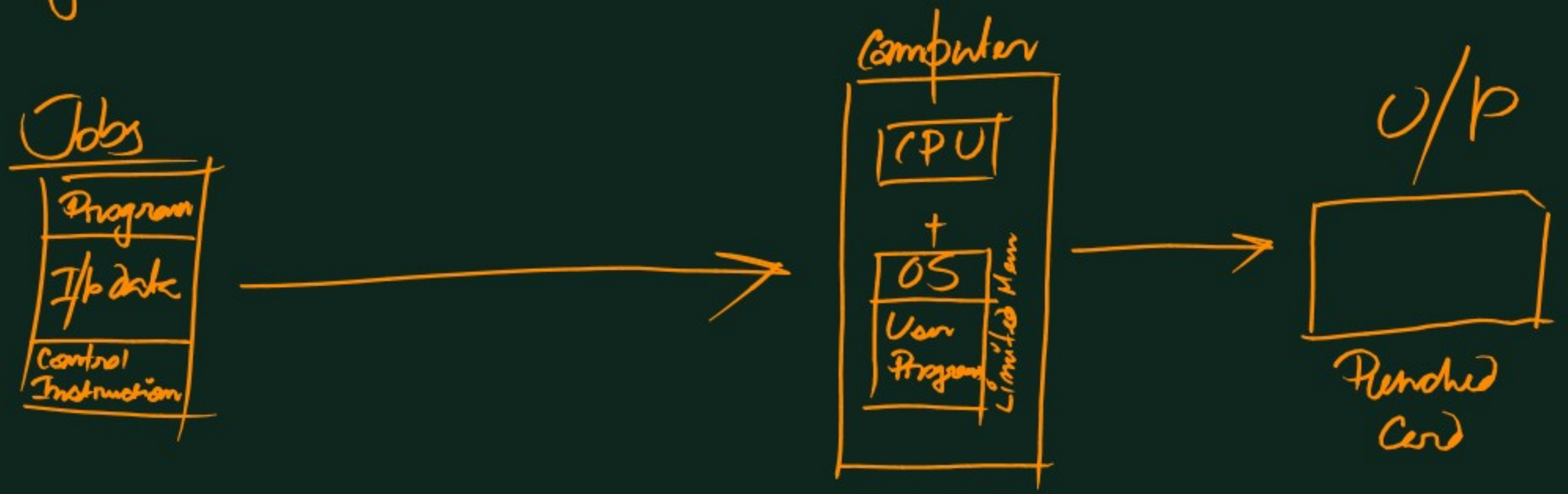
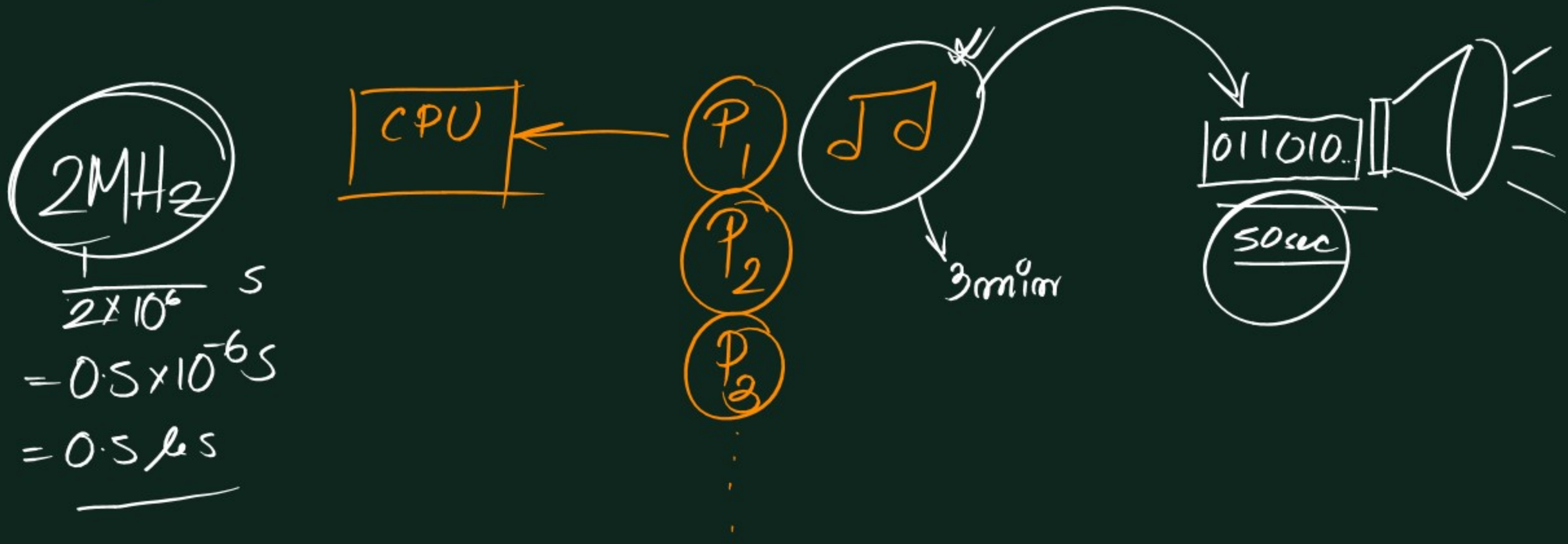
1) Single Processor OS

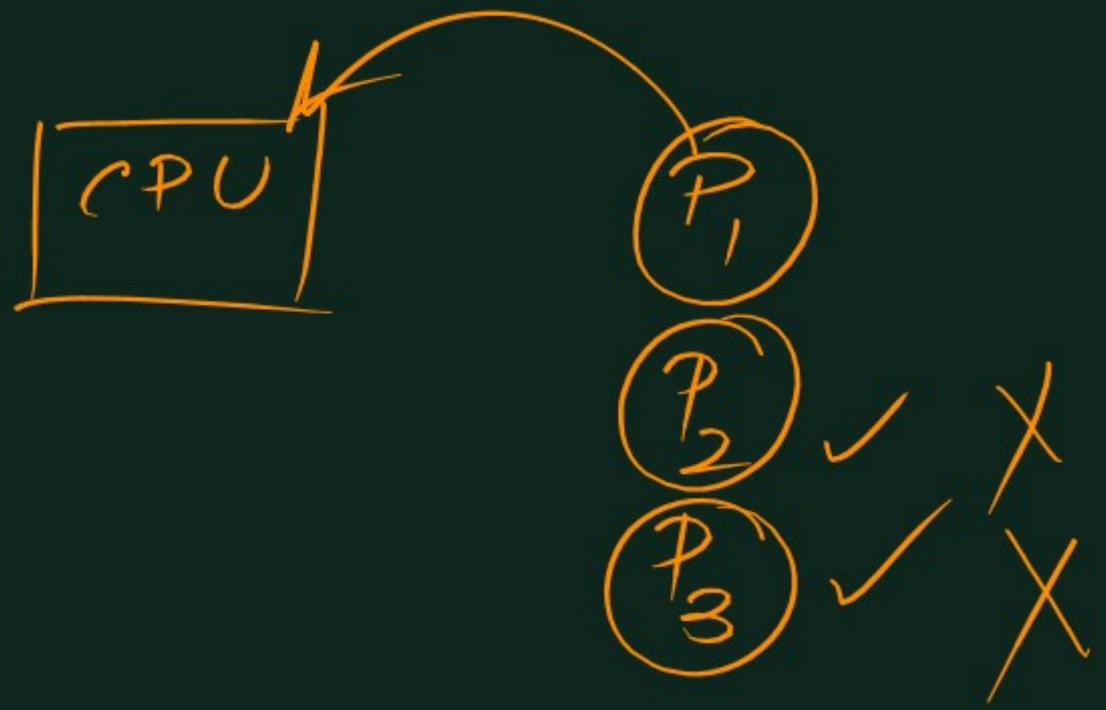


Goals:

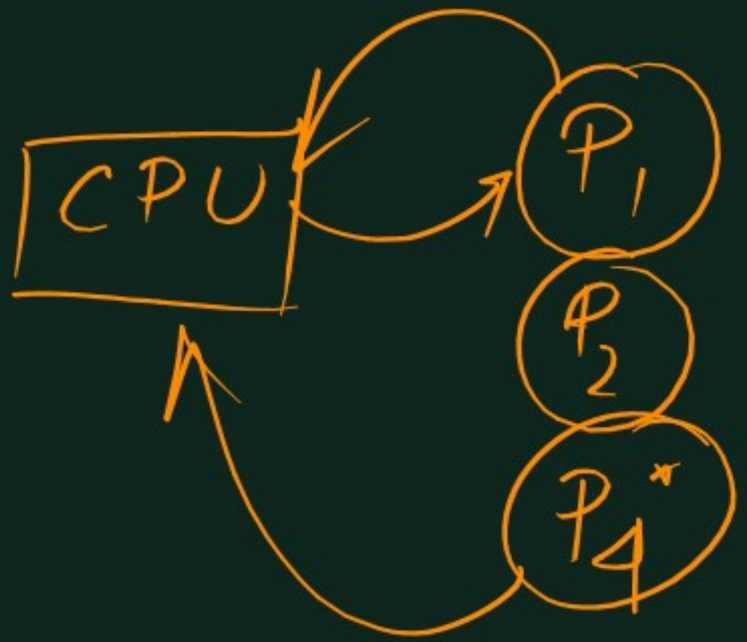
① Maximum CPU Utilization



② Minimal Process starvation:

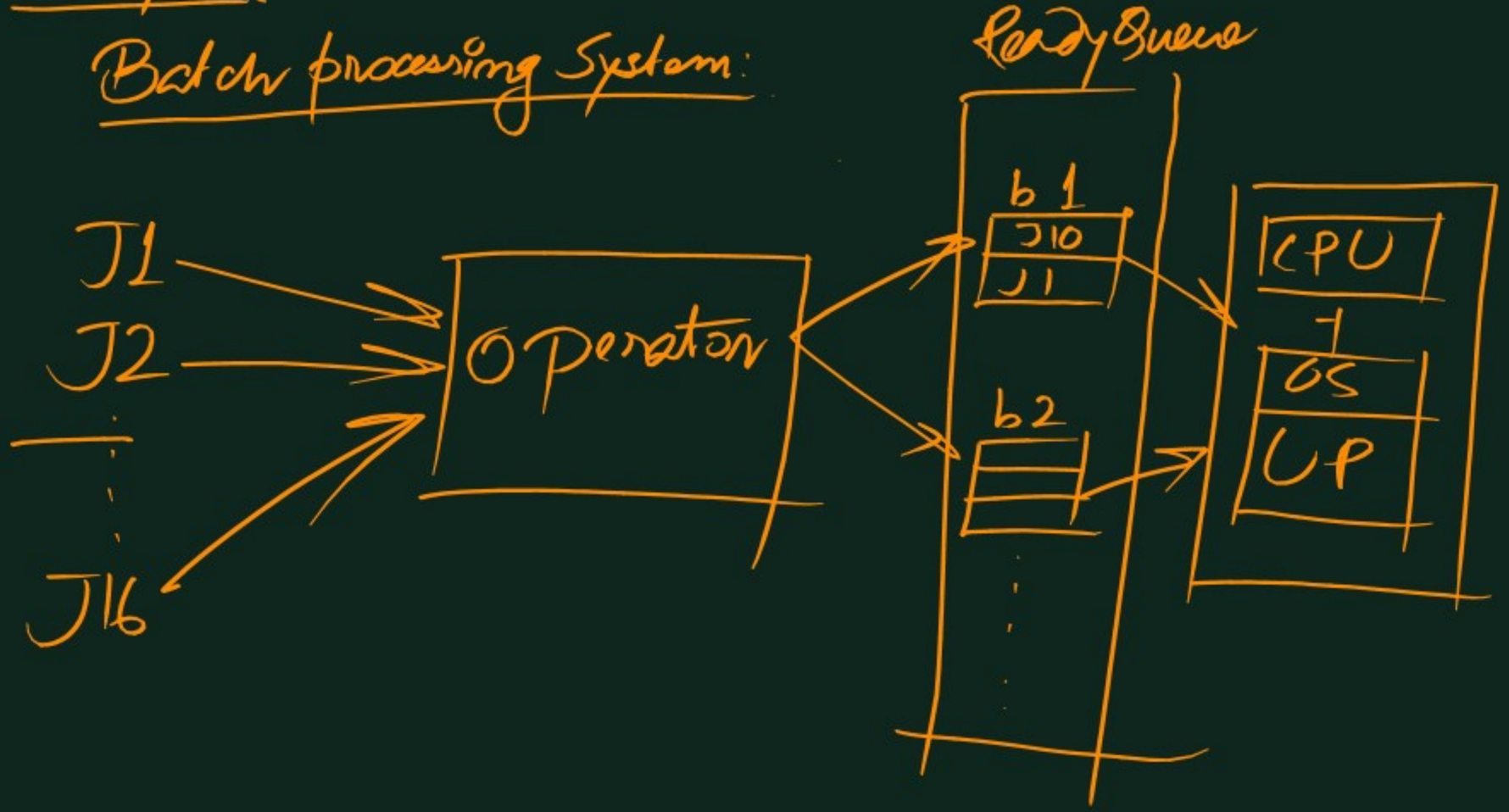


③ Timely execution of high priority Processes:



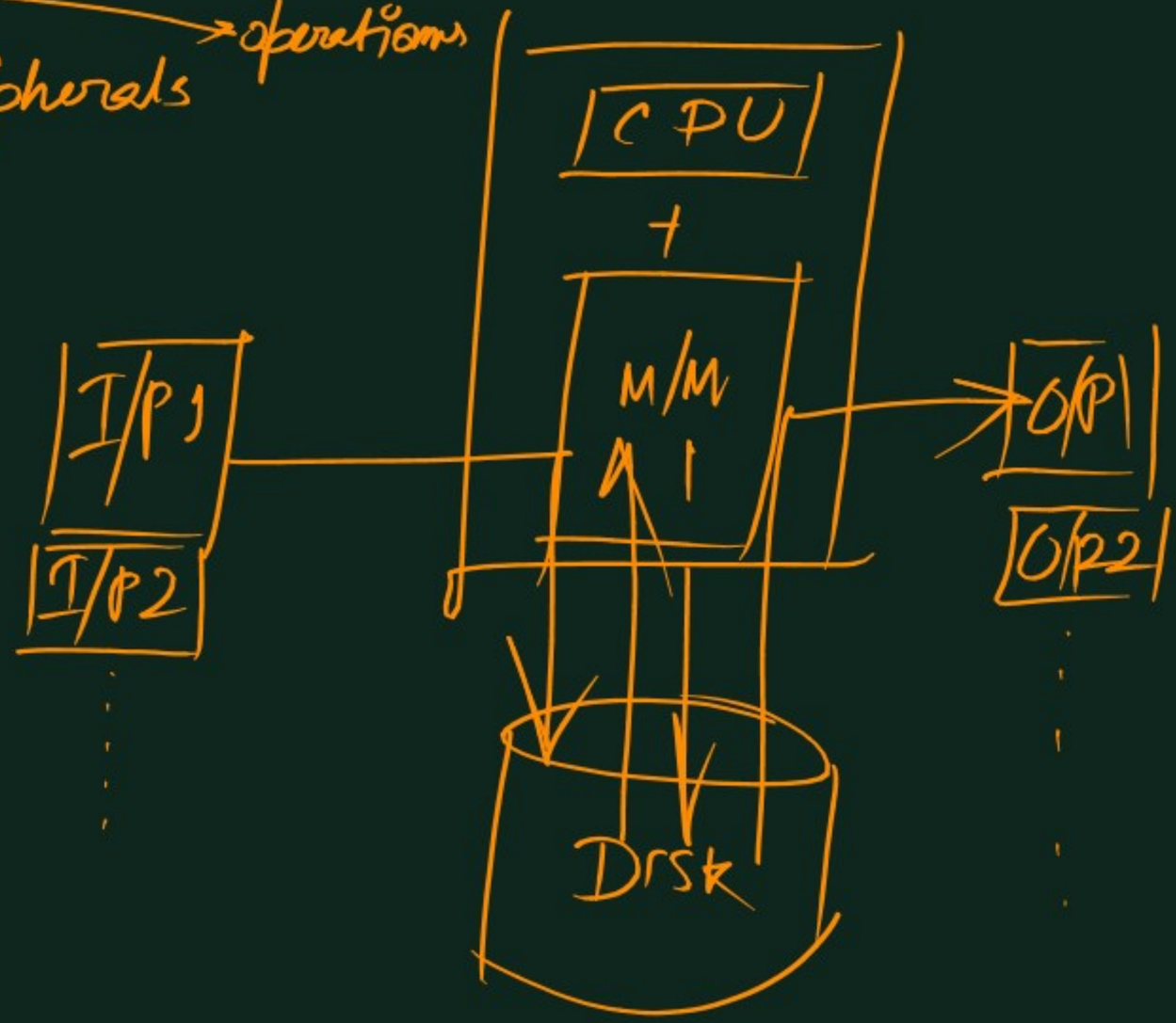
2nd Type

Batch processing System:

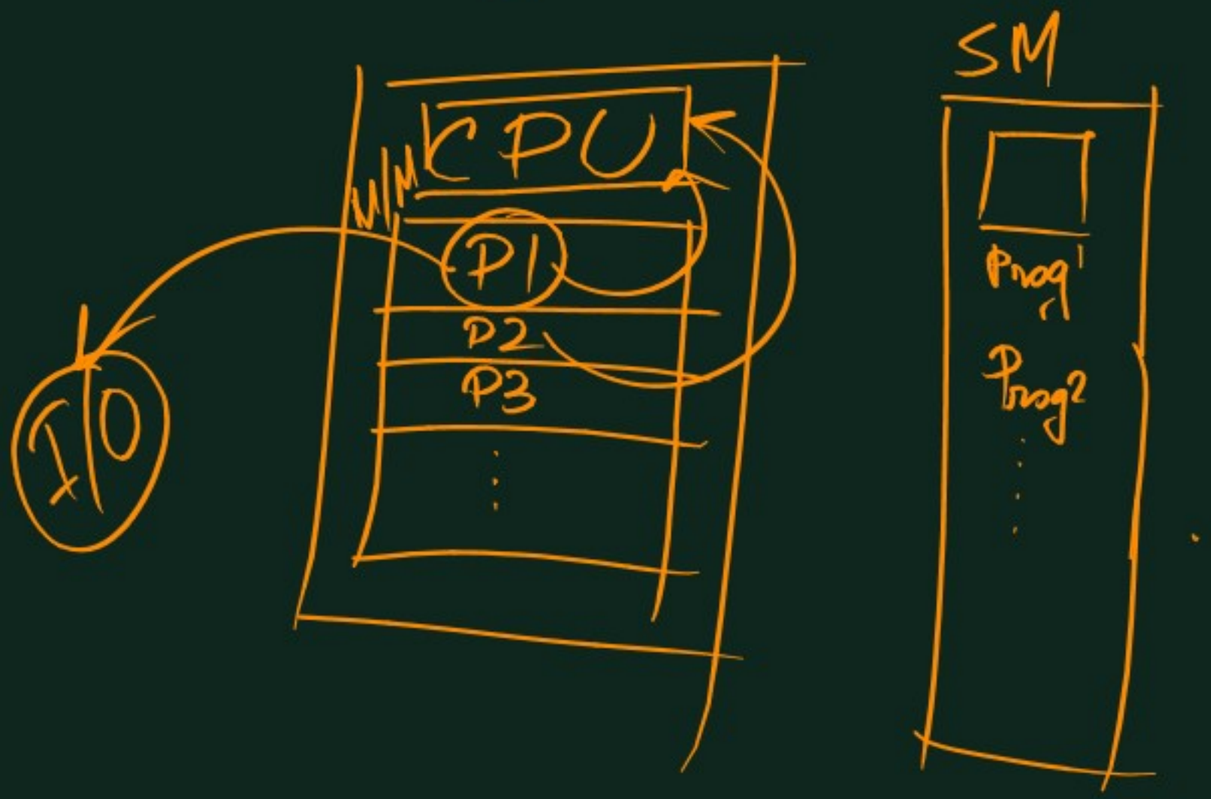


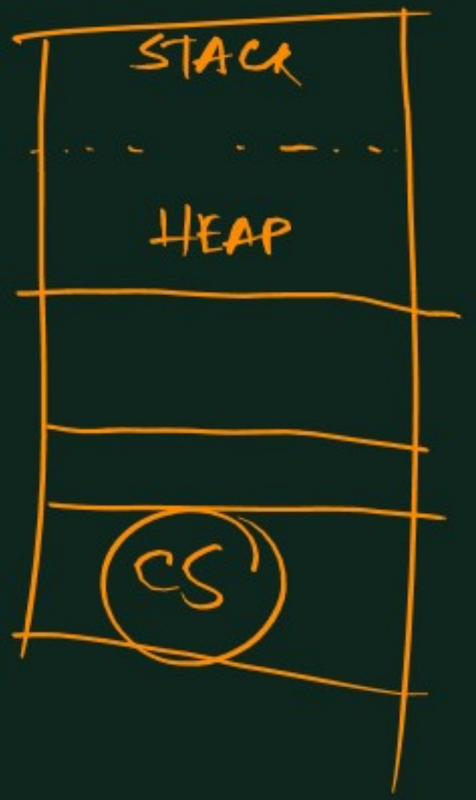
3rd Type: SPOOLING → Online operations

→ Simultaneous Peripherals



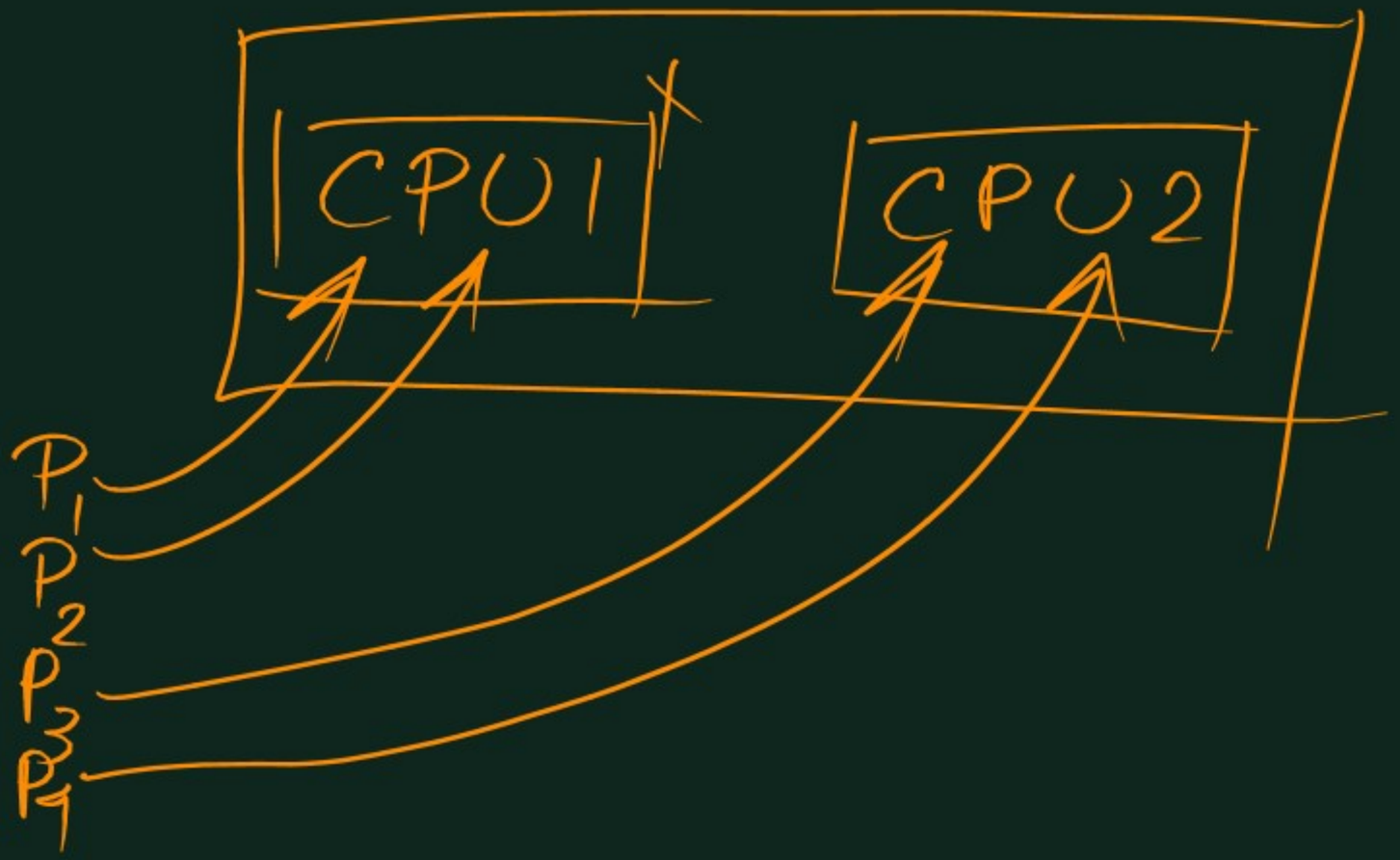
9th Multiprogramming



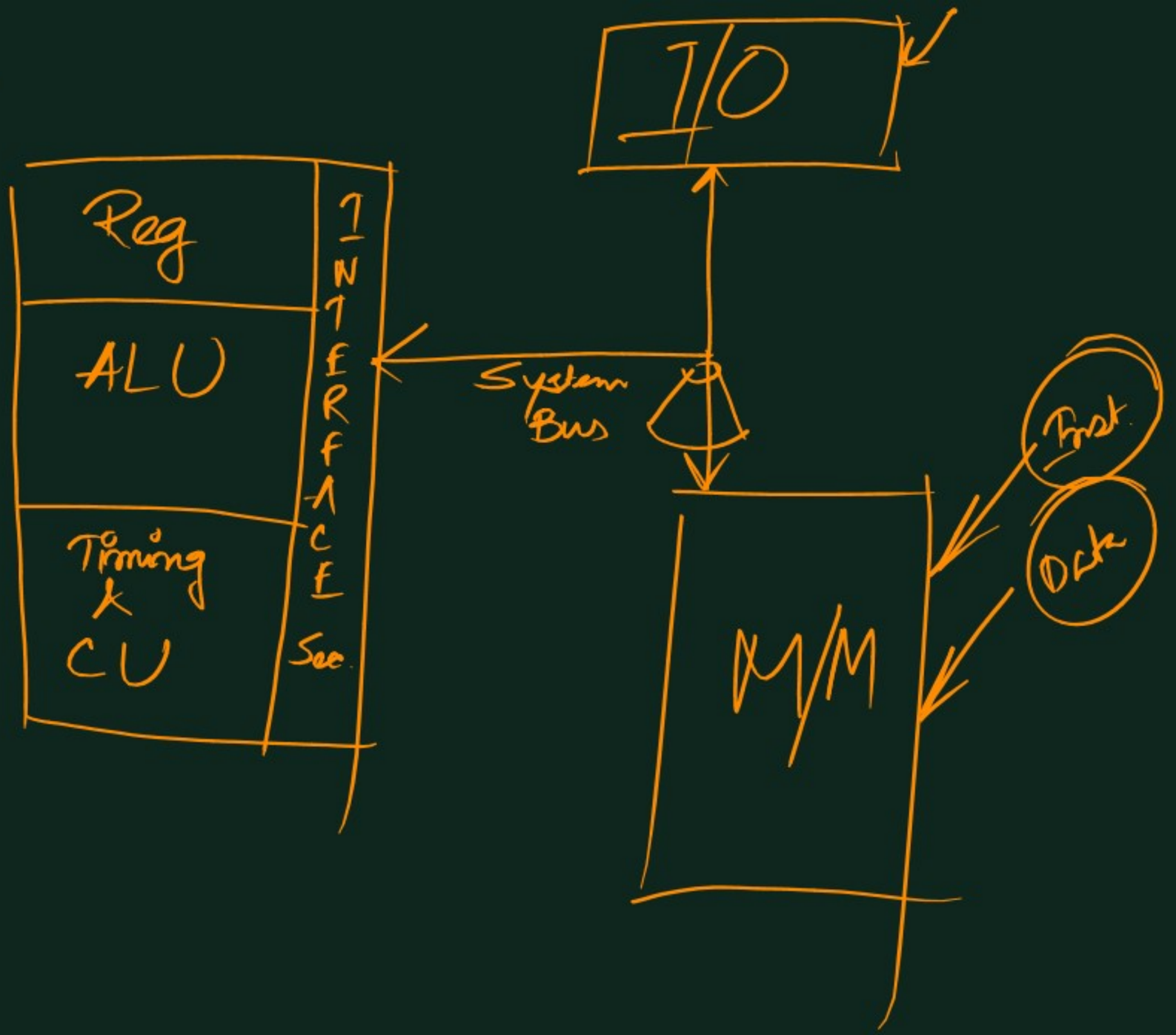


5th Multi Tasking OS (Time sharing / Fair share / Multiprogramming with Round Robin)

6th Multiprocessing OS



"Von Neumann Architecture"



- **Evolution of OS:**

At first, Mainframe computers were introduced with OS.

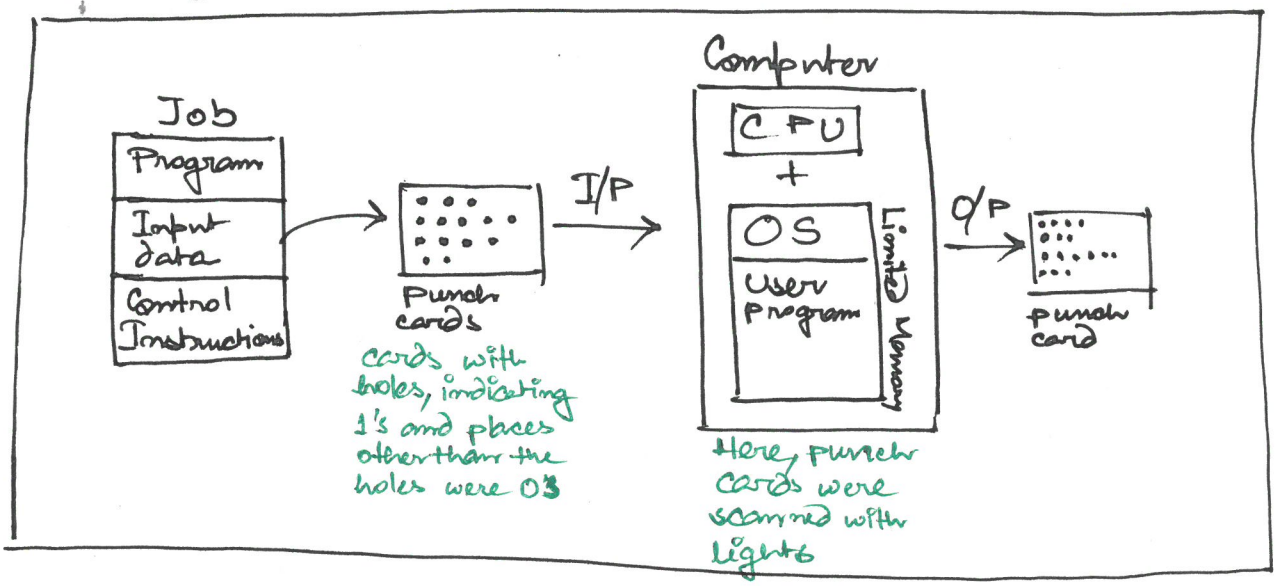


fig. Old fashioned computation

- Common I/O devices were card readers (for punch cards) & tape drives.
- Users would prepare a job consisting,
  - (i) Program,
  - (ii) I/P data
  - (iii) Control Instruction

  This job was given (fed into) to computers in the form of punch cards. finally, the output was also produced in the form of punch cards after processing.
- Reading the cards and producing the output onto the cards used to consume a lot of time in comparison to the time taken by the computer to process the job. Thus, CPU used to remain idle most of the time.

So, OS was very simple, always present in memory. The only major task was to transfer control from one job to another.

⊙ Batch Processing Systems (first Remarkable advancement in OS):

- Jobs with similar needs were batched together & executed through the Processor as a group.

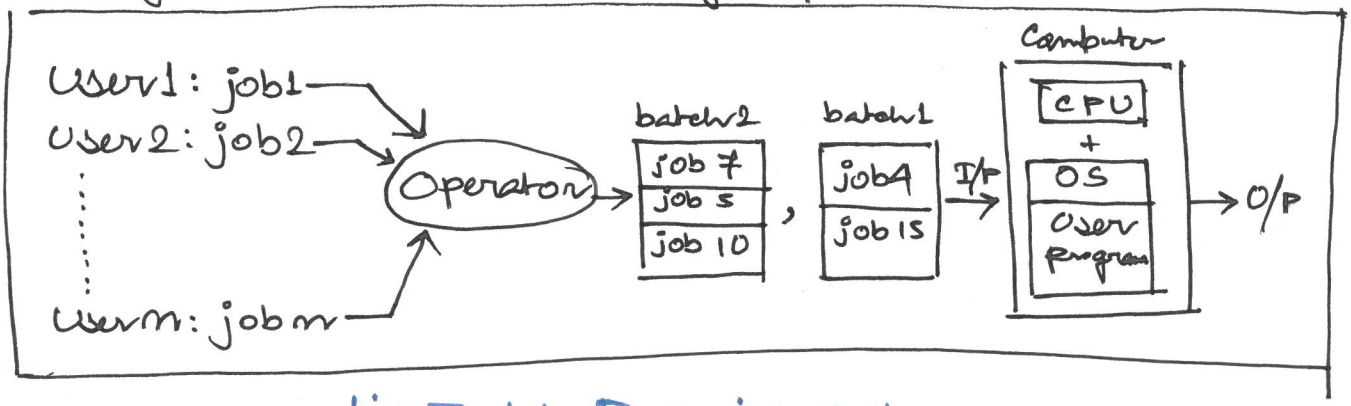


fig: Batch Processing systems

- Operator would sort jobs as a deck of punch cards into batch with similar needs. e.g. FORTRAN batch, COBOL batch etc.

Pros:
- In a batch, jobs were executed one after another saving time in activities like, loading compilers.
- During batch execution, no manual intervention was needed.

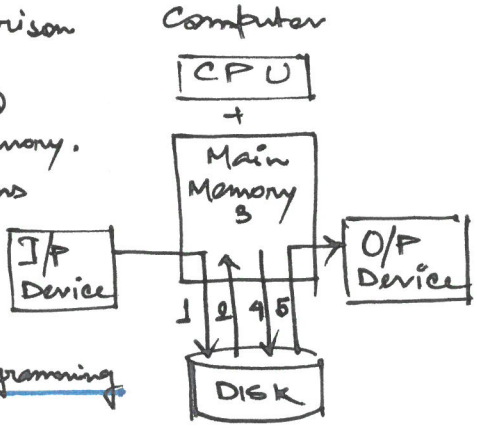
Cons:
- limited memory
- Interaction of I/O devices were direct with CPU.
- Suffers from starvation.

⊙ Spooling (Next milestone in the evolution of OS):

SPOOLing

- Simultaneous
- Peripheral
- Operations
- Online (during the CPU is performing executions)

- I/O devices are relatively slow in comparison to CPU (digital)
- In spooling, data is stored into disk and later CPU interacts with disk via Main Memory.
- It's capable of overlapping I/O operations for one jobs with CPU operations of other jobs.

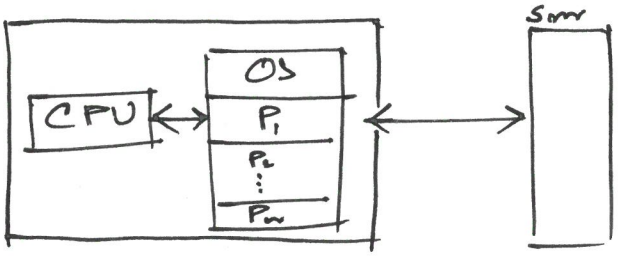


Pros:
- No interaction of I/O devices with CPU.
- CPU utilization was more as it remained busy.

Cons:
Spooling was Uniprogramming back then.

Multiprogramming OS:

- Maximized CPU utilization as more than one process in Main memory which are ready to be executed.
- Processes generally require both CPU & I/O time. So, if a running process perform I/O or some other event which does not require CPU then, instead of, ~~context switching~~ sitting idle CPU makes a context switch and picks some other process and this process continues.



- CPU never remains idle unless, there is no process ready to be executed & at the time of context switch.

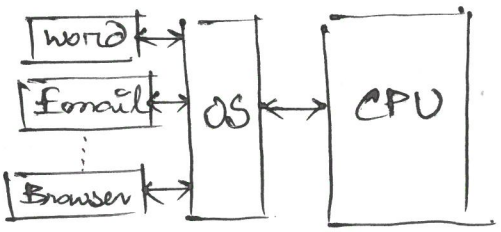
Pros:
- High CPU Utilization
- Less waiting time, response time etc.
- May be extended to multiple users.
- Very useful for today's Computations where load is more.

Cons:
- Difficult scheduling.
- Main Memory management is required.
- Memory fragmentation.
- Paging (non-contiguous memory allocation).

⊙ Multitasking Operating System:

(AKA. Time sharing / fair share / Multiprogramming with Round-Robin)

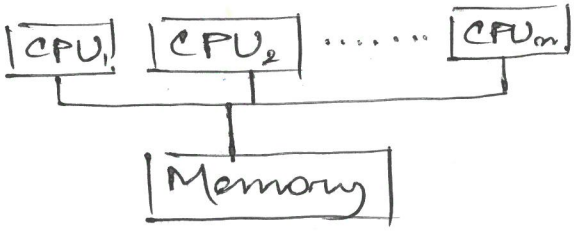
- Multitasking is multiprogramming with timesharing.
- Only one CPU but it switches between processes so quickly that gives an illusion of all are being executed at the same time.
- The task in multitasking may refer to multiple threads of the same program.
- Main idea:
  - Better response time.
  - Executing multiple process together.



- It's an extension of multiprogramming.

⊙ Multiprocessing OS:

- Two or more CPUs within a single computer, in close communication, sharing the system bus, memory & other I/O devices.
- Different process may run on different CPU, TRUE, Parallel execution.



2 TYPES:

- Symmetric: One OS controls all CPUs, each CPU has equal rights.
- Asymmetric: Master-slave architecture; System task on one processor & application on others, or, 1 CPU will handle all the H/W interrupt or I/O devices (POPULAR)
  - Easy to design
  - Less efficient.

Pros:

- Increased throughput.
- Increased Reliability.
- Cost saving.
- Battery efficient.
- True parallel processing.

Graceful Degradation!!

Cons:

- More complex.
- Overhead on coupling reduce throughput
- Large main memory.

Dependency of processes on one another!!

Class 2

Topic: Types of Operating Systems

I. Goals of an Operating System

- **Maximum CPU Utilization:** Keep the CPU busy; if a process waits for I/O, the CPU immediately works on another job.
- **Minimal Process Starvation:** Prevent indefinite blocking so every process eventually executes.
- **Timely Execution of High-Priority Jobs:** Critical tasks (e.g., antivirus scans) must run promptly.

II. Types of Operating Systems

1. **Single-Process (Single-User, Single-Tasking) OS**
    - Runs only one job at a time.
    - Drawbacks: low CPU utilization, possible starvation, and high-priority jobs must wait.
    - *Example:* Early MS-DOS.
2. **Batch Processing OS**
    - Users submitted jobs on punch cards; an operator grouped similar jobs into batches.
    - CPU executes batch by batch.
    - Drawback: if one job runs long, later jobs wait.
    - *Historical Example:* Atlas computer's batch system.
3. **Multiprogramming OS**
    - Multiple jobs kept in a ready queue on a **single CPU**.
    - When one job waits for I/O, the OS performs **context switching** to another job.
    - Goal: maximize CPU utilization.
    - *Example:* IBM OS/360, early UNIX.
4. **Multitasking / Time-Sharing OS**
    - Logical extension of multiprogramming.
    - CPU time is divided into fixed **time quanta** (e.g., 100 ms).
    - Pre-emptive scheduling improves responsiveness and fairness.
    - *Example:* CTSS (Compatible Time-Sharing System).
5. **Multiprocessing OS**
    - Uses **multiple CPUs**.

- Provides higher throughput and reliability—failure of one CPU doesn't halt the system.
- *Example:* Modern Windows, Linux on multi-core processors.

6. **Distributed OS**
   - Loosely coupled computers connected via a network share tasks and resources.
   - Nodes may have different configurations.
   - *Example:* Amoeba, modern cloud clusters.
7. **Real-Time OS (RTOS)**
   - Guarantees completion of tasks within strict time limits.
   - Used in flight control, industrial automation, nuclear systems.

III. Context Switching

- **Definition:** The act of saving the state of a running process and loading the state of another so multiple processes share a single CPU.
- **Steps:**
  1. Save current process state (program counter, CPU registers, stack pointer) to its **Process Control Block (PCB)**.
  2. Load the next process's saved state from its PCB.
  3. Resume execution of the next process.
- This rapid save-and-restore creates the illusion of simultaneous execution.
