Introduction to CPU Management

Process Management

In modern computers, multiple programs often run at the same time. Process management ensures these running programs (called processes) are tracked, managed, and executed without interference. The OS uses process states and structures like the PCB to organize this behavior.

Computers must organize multiple running programs to work properly.

Definition:

A process is a program in execution. The Process Control Block (PCB) stores all process details.

Explanation:

- Manages process states: New, Ready, Running, Waiting, Terminated.

- PCB holds process-specific information (state, memory, CPU registers).

CPU Scheduling Algorithms

With many processes competing for CPU time, the operating system must decide which process to run next. This is done through CPU scheduling, which helps maintain system responsiveness, efficiency, and fairness in multitasking environments.

The OS needs rules to decide which process gets the CPU first.

Definition:

CPU Scheduling is how the OS selects the next process to run.

Explanation:

- FCFS: First come, first served — simple but can cause long waiting.

- SJF: Shortest job first — faster jobs prioritized.

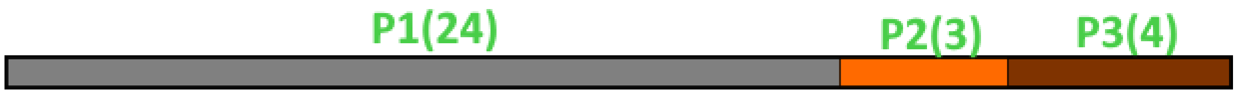
- Round Robin: Each process gets a short turn (time slice), then cycles.

First Come First Serve

FCFS (Example)

Process	Duration	Oder	Arrival Time
P1	24	1	0
P2	3	2	0
P3	4	3	0

Gantt Chart :



P1 waiting time : 0

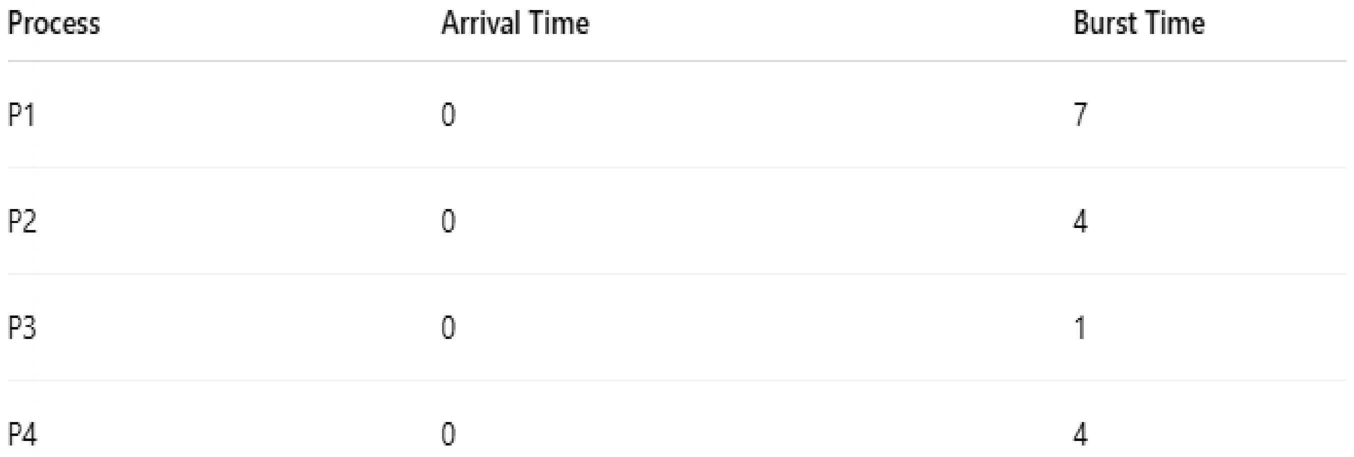
P2 waiting time : 24

P3 waiting time : 27

The Average waiting time :

(0+24+27)/3 = 17

Shortest Job First



Process	Arrival Time	Burst Time
P1	0	7
P2	0	4
P3	0	1
P4	0	4

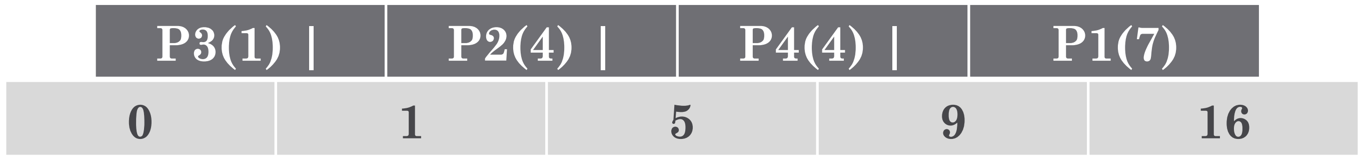
Shortest Job First

Since all processes arrive at time 0, we sort by

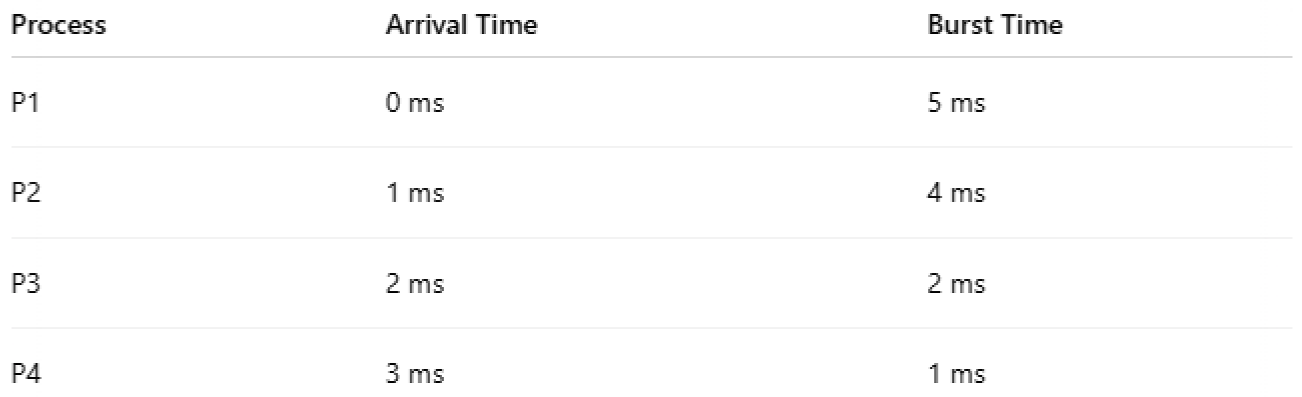
Burst Time:

- P3 (1)
- P2 (4)
- P4 (4)
- P1 (7)

Gantt Chart



Round Robin (RR)



Process	Arrival Time	Burst Time
P1	0 ms	5 ms
P2	1 ms	4 ms
P3	2 ms	2 ms
P4	3 ms	1 ms

Time Quantum = 2ms

Simulation Flow

Time 0:
P1 arrives → Runs for 2 ms → Remaining: 3 → Queue: P2, P3, P4, P1

Time 2:
P2 runs for 2 ms → Remaining: 2 → Queue: P3, P4, P1, P2

Time 4:
P3 runs for 2 ms → Done → Queue: P4, P1, P2

Time 6:
P4 runs for 1 ms → Done → Queue: P1, P2

Time 7:
P1 runs for 2 ms → Remaining: 1 → Queue: P2, P1

Time 9:
P2 runs for 2 ms → Done → Queue: P1

Time 11:
P1 runs for 1 ms → Done

Gantt Chart

P1 (2)	P2(2)	P3(2)	P4(1)	P1(2)	P2(2)	P1(1)

0	2	4	6	7	9	11	12

Memory Management Concepts

Every process needs memory to function, and the operating system is responsible for assigning, tracking, and protecting that memory. Memory management techniques like paging and segmentation help optimize memory use and prevent conflicts.

Programs need carefully managed memory to avoid crashes.

Definition:

Memory management handles the allocation of memory spaces for processes.

Explanation:

- Paging: Fixed-size memory blocks (pages).

- Segmentation: Logical memory division based on program parts.

Memory Hierarchy

Memory in a computer is organized in layers, each with different speed and size. Understanding the memory hierarchy helps explain why some operations are faster than others and how data moves within the system.

Memory types vary by speed, size, and cost.

Definition:

Memory Hierarchy organizes storage types based on speed: Registers, Cache, RAM, Storage.

Explanation:

- Faster memory is smaller and costlier.

- Slower memory stores more but is cheaper.

Paging and Segmentation

Operating systems divide memory to improve efficiency and program organization. Paging splits memory into equal parts, while segmentation aligns memory blocks with program components for better structure and protection.

The OS slices up memory to maximize efficiency.

Definition:

- Paging: Fixed-size division of memory.

- Segmentation: Logical, meaningful division.

Explanation:

- Paging avoids wasted space but ignores program structure.

- Segmentation protects and organizes program components.

Virtual Memory

When physical RAM is full, virtual memory enables systems to run more applications by using part of the hard drive as extended memory. This allows larger programs to run but may lead to performance trade-offs.

When RAM is not enough, virtual memory helps.

Definition:

Virtual Memory uses part of the storage drive as additional RAM.

Explanation:

- Enables multitasking even with limited RAM.

- Can slow down performance (thrashing risk).

History of I/O Systems

I/O systems allow communication between a computer and external devices. From early mechanical input methods to today’s high-speed interfaces, I/O systems have evolved dramatically to meet user and application demands.

Early computers struggled to interact with users and devices.

Definition:

I/O Systems manage communication between the CPU and peripherals.

Explanation:

- From punch cards to keyboards and networks.

- Enabled real-time interaction.

Device Drivers

Every hardware device needs a way to talk to the operating system. Device drivers act as translators, ensuring smooth communication between software and hardware and enabling devices to function properly.

Different devices need translators to talk to the OS.

Definition:

Device drivers allow the OS to communicate with specific hardware.

Explanation:

- Translate OS commands into device-specific actions.

- Required for hardware to work.

I/O Scheduling Techniques

Just like processes are scheduled for the CPU, I/O requests must be organized efficiently. I/O scheduling strategies reduce waiting time and improve system performance by managing how read and write operations are prioritized.

I/O operations must be handled smartly to avoid delays.

Definition:

I/O Scheduling manages the order of read/write requests.

Explanation:

- FIFO: Serve requests in arrival order.

- SSTF: Serve the nearest request.

- SCAN: Sweep in one direction, then reverse.

Assembly Language Structure

Assembly language allows programmers to write instructions that directly control the CPU. It’s close to machine language and offers powerful control, but requires a strong understanding of computer architecture.

Programming at the lowest hardware level gives full control.

Definition:

Assembly Language is a low-level programming language close to machine instructions.

Explanation:

- Directly controls the CPU and memory.

- Requires detailed coding.

MASM Syntax and Commands

MASM is a tool that helps programmers write Assembly code more easily. It simplifies syntax, adds macros, and compiles low-level instructions for efficient execution on Intel-based processors.

MASM simplifies writing complex Assembly programs.

Definition:

MASM (Microsoft Macro Assembler) helps write Assembly programs efficiently.

Explanation:

- Macros simplify repeated instructions.

- Generates machine-level code.