Internal Assessment Test Answer Key 1 – Nov. 2025
Operating Systems(BCS303)

<table>
<tr><th></th><th>Questions</th><th>Marks</th></tr>
<tr><td>1</td><td>

a) Write the difference between symmetric and asymmetric multiprocessing.

Symmetric Multiprocessing (SMP)	Asymmetric Multiprocessing (AMP)
All processors are treated equally and share the same memory and I/O.	One processor (master) controls the system; others (slaves) perform assigned tasks.
Each processor can perform any task, including OS functions.	Only the master processor handles OS functions; slave processors handle user programs.
Load is evenly distributed among all processors.	Load distribution is fixed or controlled by the master processor.
More reliable — failure of one CPU doesn't stop the system.	Less reliable — failure of the master CPU can halt the system.
More complex to design and manage due to shared memory and synchronization.	Simpler to design since one CPU manages control.

</td><td>4</td></tr>
<tr><td></td><td>

b) Explain the services of the operating system that are helpful for the user and the system

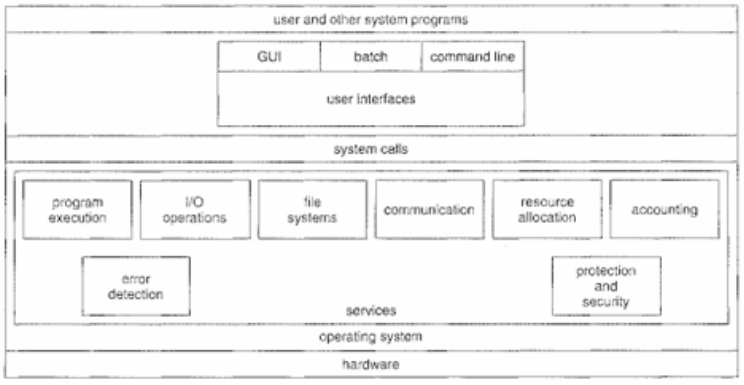


Figure 2.1 A view of operating system services.

2 marks

</td><td>4</td></tr>
</table>

1. User-Oriented Services

These services make the system convenient and easy for users to use:

Service	Description
User Interface (UI)	The OS provides different types of interfaces — Command-Line Interface (CLI), Graphical User Interface (GUI), or Touch Interface — to interact with the system easily.
Program Execution	The OS loads a program into memory and executes it. It handles the program's execution, including halting when finished or due to errors.
I/O Operations	OS manages input and output operations, providing a uniform interface for I/O devices (keyboard, printer, disk, etc.).
File System Manipulation	OS allows programs and users to read, write, create, delete, and organize files on storage devices.
Communication	OS allows processes to exchange information, either within the same system (interprocess communication) or over a network (client-server communication).
Error Detection	OS constantly monitors for possible errors in CPU, memory, I/O devices, or programs, and takes corrective actions.

---

2. System-Oriented Services

These ensure efficient and secure system operation:

Service	Description
Resource Allocation	The OS allocates system resources (CPU time, memory, I/O devices) among multiple users or processes efficiently.
Accounting	The OS keeps track of resource usage for each user or process, which helps in billing or performance analysis.
Protection and Security	The OS ensures that all system resources are accessed only by authorized users and processes, protecting against

<table>
<tr><td></td><td>unauthorized access and failures.
2 marks</td><td></td></tr>
<tr><td></td><td>c) Define Operating System with an example.
An Operating System (OS) is a program that acts as an intermediary between the user and the computer hardware.
Its main purpose is to make the computer system convenient to use and to use the hardware efficiently.
Windows (Microsoft)

Linux (open-source)

macOS (Apple)

Android (mobile devices</td><td>2</td></tr>
<tr><td>2</td><td>a) Compare batch, time-sharing, and distributed operating systems in terms of control, user interaction, and resource sharing.

Comparison of Batch, Time-Sharing, and Distributed Operating Systems
<table><tr><th>Feature</th><th>Batch Operating System</th><th>Time-Sharing Operating System</th><th>Distributed Operating System</th></tr><tr><td>Control</td><td>Centralized control — the operating system controls job sequencing and execution in batches.</td><td>Centralized control — CPU time is shared among multiple users; each gets a small time slice.</td><td>Decentralized control — multiple systems coordinate and share control through communication networks.</td></tr><tr><td>User Interaction</td><td>No direct interaction with users during execution. Jobs are submitted and output is received later.</td><td>Direct interaction with users through terminals; users can input commands and get immediate responses.</td><td>Users interact with the system as if it were a single unified system, even though resources are on different machines.</td></tr></table></td><td>6</td></tr>
</table>

	Resource Sharing	Limited; resources are allocated to one job at a time.	Shared among multiple users simultaneously using scheduling.	Resources (CPU, memory, files, printers, etc.) are shared among interconnected computers through the network.	
	Objective	Maximize system utilization and throughput.	Provide quick response time and support multiple users interactively.	Improve performance, reliability, and resource availability through cooperation between systems.	
	Example	Early mainframe systems like IBM 1401.	UNIX, Windows multi-user systems.	Amoeba, LOCUS, or modern networked Linux clusters.	

b) Explain the layered approach of operating system structure with a supporting diagram. 4

The layered approach is a method of designing an operating system where the system is divided into a number of layers (levels), each built on top of lower layers.

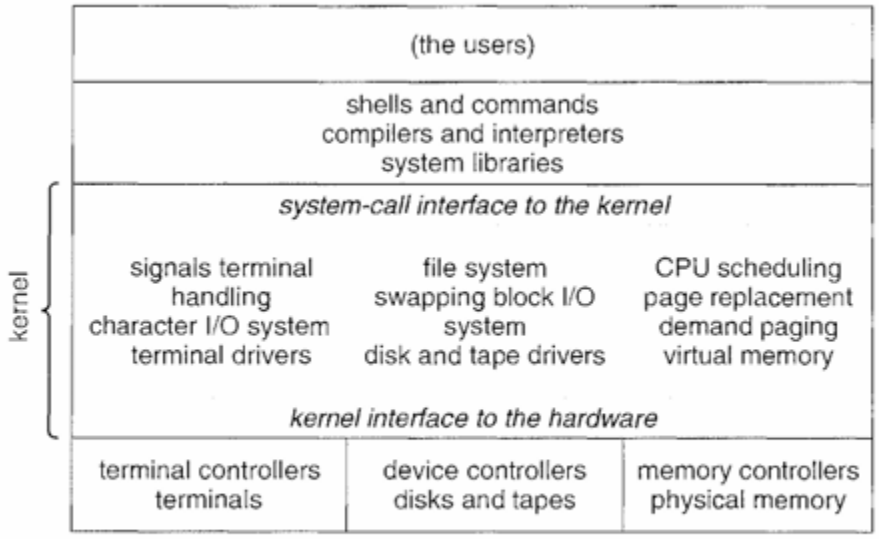


Figure 2.13 Traditional UNIX system structure.

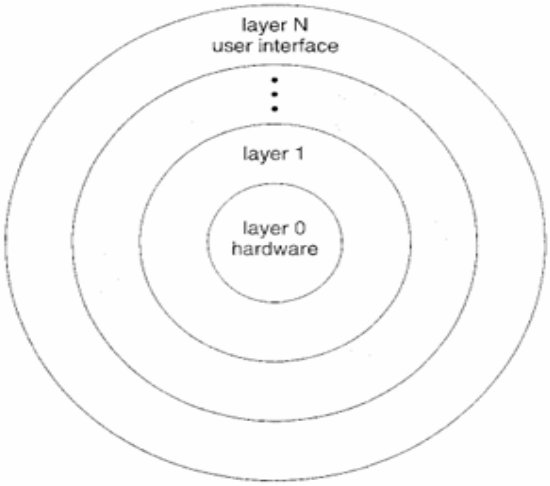


Figure 2.14 A layered operating system.

2 marks

The layered approach divides the operating system into a number of levels (layers), where each layer is built upon the one below it.

- The bottom layer (Layer 0) is the hardware, and the top layer is the user interface.
- Each layer uses the functions and services of lower layers and provides services to the higher layers.

The main advantage of this approach is simplicity of design, construction, and debugging.

- Each layer can be tested and verified independently.
- If an error occurs in one layer, only that layer needs to be checked since lower layers are already debugged.

This structure supports modularity, information hiding, and ease of maintenance, making the operating system easier to develop and verify. **2 marks**

3	**a) Explain the transition between user mode and kernel mode with the help of a diagram.**	

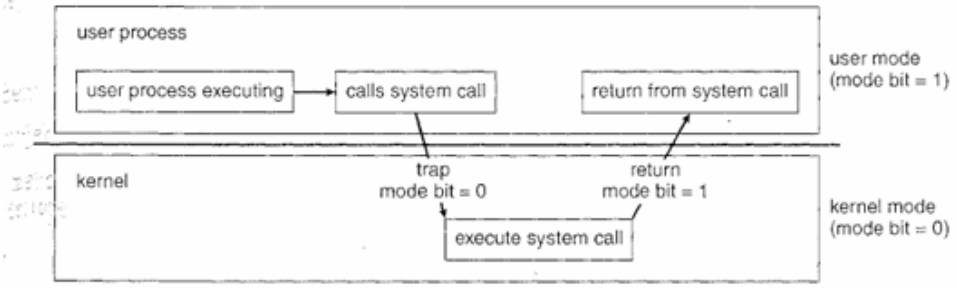


Figure 1.10 Transition from user to kernel mode.

2 marks

Modern operating systems use **dual mode of operation** to protect the system and distinguish between **user programs** and **operating system code**.

There are two modes:

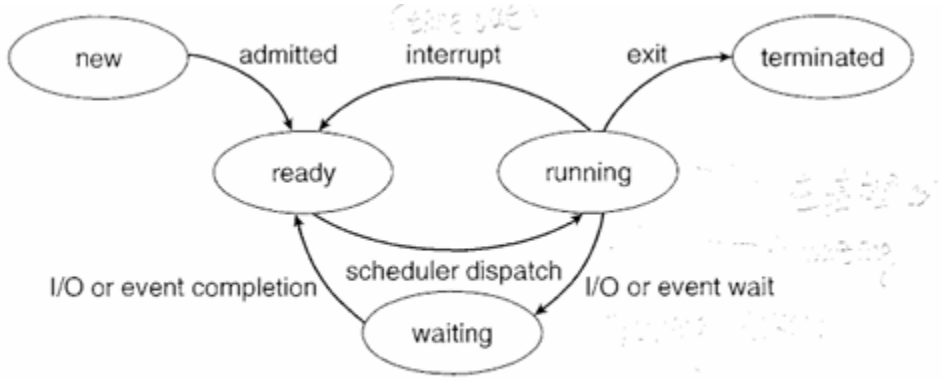
1. **User Mode** – for executing user applications.
2. **Kernel Mode** – for executing operating system (privileged) instructions.

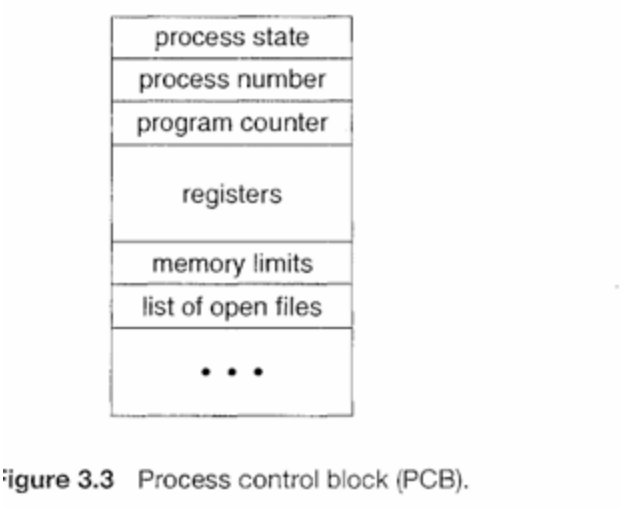
Explanation

- The CPU uses a **mode bit** (0 or 1) to indicate the current mode:
    - **0 → Kernel Mode**
    - **1 → User Mode**
- When a user program needs to perform an operation that requires OS privileges (like I/O or memory access), it makes a **system call**.
- The system call causes a **mode switch** from **user mode → kernel mode**.
- After completing the requested service, control returns to the user program and the mode switches **back to user mode**.

2 marks

b) Define a process and explain different states of a process with a neat state transition diagram.

	A process is a program in execution. It is an active entity with its own program counter, stack, data section, and set of resources. **1 mark**	
	 **Figure 3.2** Diagram of process state. **2 marks** **New → Ready:** Process is admitted to the ready queue after creation. **Ready → Running:** The CPU scheduler selects the process for execution. **Running → Waiting:** The process waits for I/O or an event to complete. **Waiting → Ready:** The event occurs; process becomes ready to execute again. **Running → Terminated:** Process finishes execution and is removed from memory. **Running → Ready:** Process is preempted by the scheduler (time slice expired). **2 marks**	
4	**a) Illustrate how Process Control Block (PCB) is used during context switching** Process Control Block (PCB) is a data structure maintained by the operating system for every process. It contains all the information required to manage and resume a process.	



igure 3.3 Process control block (PCB).

1 mark

Context Switching is the process of storing the state of a running process so that it can be resumed later, and loading the state of another process to continue its execution.

It occurs when:

- A process is preempted by the scheduler.
- A process waits for I/O.
- An interrupt or system call occurs.

Role of PCB During Context Switch

1. Save Current Process State:
    - The CPU's registers, program counter, and other information of the running process are saved into its PCB.
2. Update Process State:
    - The OS changes the process state from Running → Ready or Running → Waiting, depending on the reason for the switch.
3. Select New Process:
    - The scheduler selects a new process from the ready queue.

4. Load the New Process State:
    - The CPU registers and program counter are loaded from the new process's PCB.
5. Resume Execution:
    - The new process continues from the exact point it was previously stopped. **2 marks**

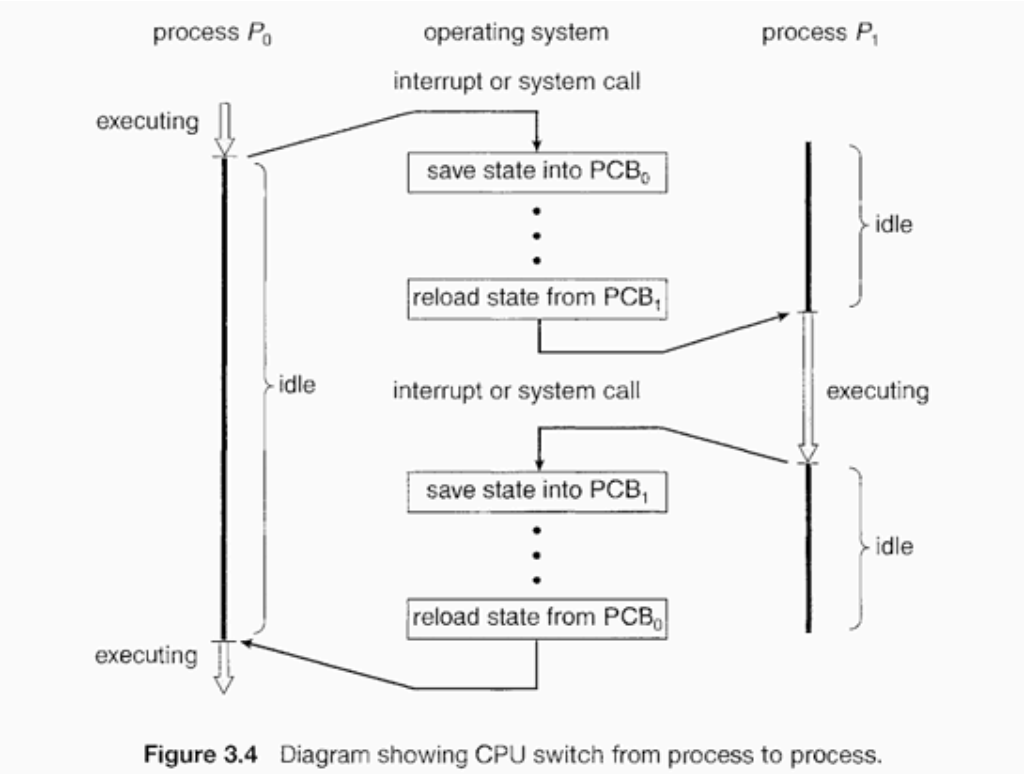


Figure 3.4 Diagram showing CPU switch from process to process.

2 marks

b) Differentiate between long-term, short-term, and medium-term schedulers with examples. 5

Schedulers are special system programs that decide which processes are to be admitted, executed, or swapped.
They help manage CPU utilization and process flow efficiently.

Feature	Long-Term Scheduler (Job Scheduler)	Short-Term Scheduler (CPU Scheduler)	Medium-Term Scheduler (Swapper)

	Function	Determines which processes are admitted into the system for processing.	Determines which ready process will be executed next by the CPU.	Temporarily removes (suspends) processes from main memory and later reintroduces them.	
	Frequency of Execution	Executes rarely (when a new process is created).	Executes very frequently (milliseconds).	Executes occasionally (based on memory load).	
	Speed	Slow – decisions made infrequently.	Fastest – must make quick decisions for CPU allocation.	Moderate – depends on system load and swapping needs.	
	Goal	Controls the degree of multiprogramming (number of processes in memory).	Provides CPU scheduling among ready processes.	Improves process mix and reduces memory load.	
	Process State Transition	Moves process from New → Ready state.	Moves process from Ready → Running → Waiting/Terminated.	Moves process from Ready/Waiting → Suspended and back.	
	Example	Decides which batch job to load from disk.	Chooses between processes in ready queue for CPU.	Swaps out background processes to free memory (used in UNIX).	

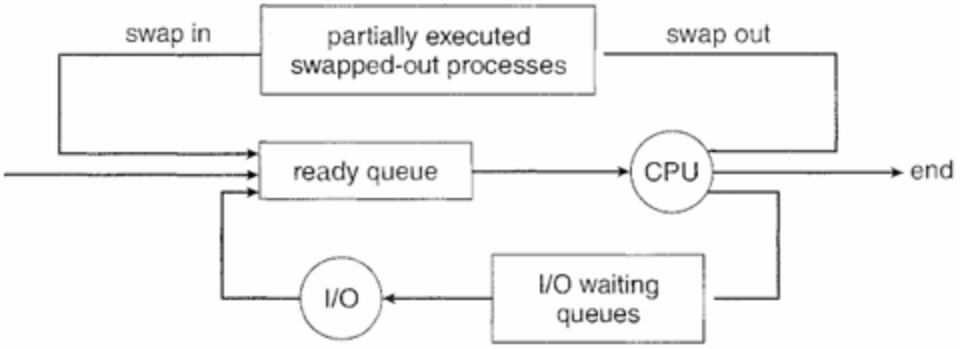


Figure 3.8 Addition of medium-term scheduling to the queueing diagram.

5 **a) Explain interprocess communication (IPC) using shared memory and message passing.**

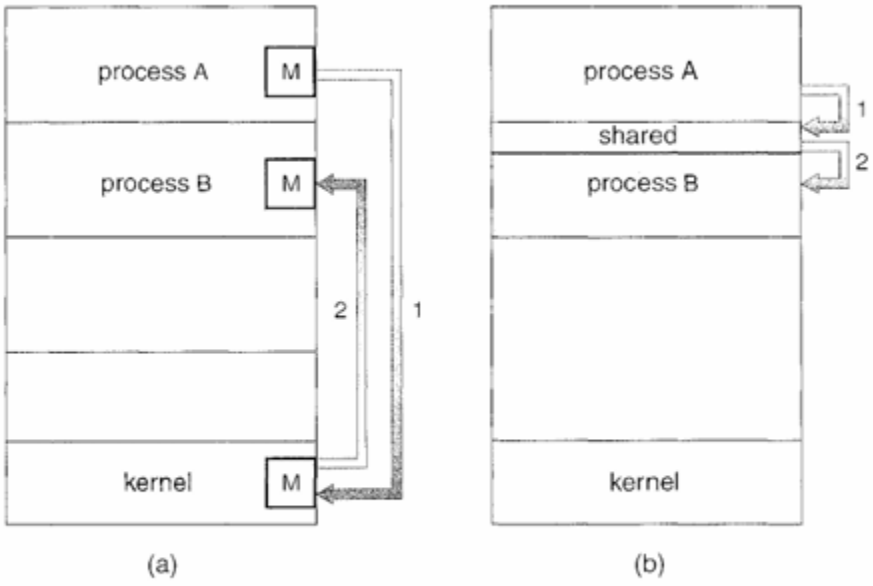


Figure 3.13 Communications models. (a) Message passing. (b) Shared memory.

2 marks

Interprocess Communication (IPC) allows processes to exchange data and information with each other.

Types of IPC Mechanisms

There are two main models of interprocess communication:

1. Shared Memory

2. Message Passing

Shared Memory System

Concept:

- A region of memory is shared by cooperating processes.
- Each process can read and write to this memory area.
- It is the fastest form of IPC because data is exchanged directly through memory.

How it Works:

1. Two processes establish a shared memory region using system calls.
2. Once created, the OS no longer needs to be involved.
3. Processes use synchronization (like semaphores) to avoid data inconsistency.

Concept:

- Processes communicate by sending and receiving messages through the OS.
- No shared memory is used; the kernel handles message transfer.

How it Works:

1. Processes use system calls like `send(message)` and `receive(message)`.
2. Messages can be direct (process-to-process) or indirect (through mailboxes).
3. It is slower than shared memory but safer and easier to implement in distributed systems.

3 Marks

b) Explain the concept of multithreading. Differentiate between user-level and kernel-level threads with one diagram illustrating their interaction with the operating system.

Multithreading is the ability of an operating system to execute multiple threads within a single process simultaneously.

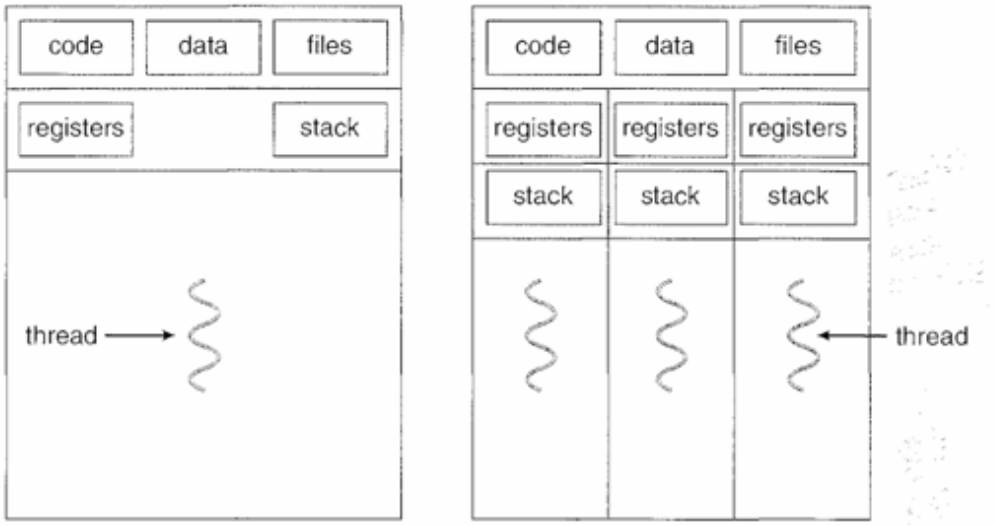


Figure 4.1 Single-threaded and multithreaded processes.

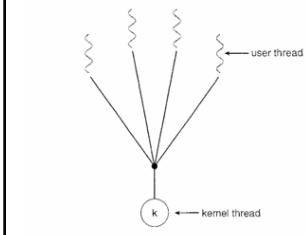


Figure 4.5 Many-to-one model.

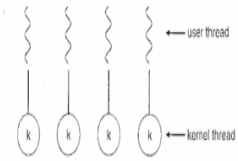


Figure 4.6 One-to-one model.

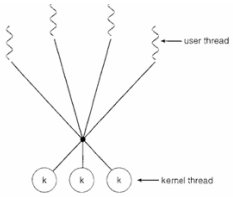


Figure 4.7 Many-to-many model.

2 marks

Many-to-One Model (M:1)
Description:

- Many user-level threads are mapped to a single kernel thread.
- Thread management is done by the user-level thread library, so the kernel is unaware of the existence of multiple threads.

Advantages:

- Thread creation and switching are very fast (no kernel involvement).

Disadvantages:

- If one thread makes a blocking system call → entire process blocks.
- No true parallelism on multiprocessor systems.

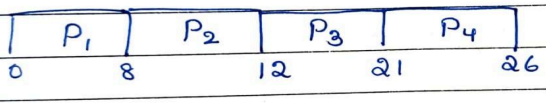
	Examples: ● Older thread systems like Green Threads (early Java). One-to-One Model (1:1) Description: ● Each user thread maps to one kernel thread. ● The OS handles scheduling and management of all threads. Advantages: ● If one thread blocks, others continue execution. ● True concurrency on multiprocessor systems. Disadvantages: ● More overhead due to kernel involvement. ● Creating many threads may burden the system. Examples: ● Windows, Linux, Solaris 9 and later. Many-to-Many Model (M:N) Description: ● Multiple user threads are mapped to a smaller or equal number of kernel threads. ● The OS can schedule kernel threads on available processors, while the thread library maps user threads to them dynamically. Advantages: ● Combines flexibility and concurrency. ● If one thread blocks, others can still run. ● Supports parallelism and better resource use. Examples: ● Solaris, Windows with ThreadFiber package, Modern UNIX systems**.** **3 Marks**	
6	**Consider the following set of processes with their respective burst times and arrival times:** **a) Draw Gantt charts and calculate the average waiting time and average turnaround time using the FCFS, SRTF, and Round Robin (quantum = 3 ms) scheduling algorithms.**	10

Process	Arrival Time (ms)	Burst Time (ms)
P1	0	8
P2	1	4
P3	2	9
P4	3	5

FCFS.

Process	AT	BT	CT	TAT	WT
P_1	0	8	8	8	0
P_2	1	4	12	11	7
P_3	2	9	21	19	10
P_4	3	5	26	23	18
				61	35

Gantt chart:

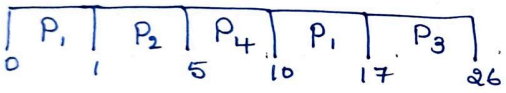


Average TAT = $\frac{61}{4}$ = 15.25 ms

Average WT = $\frac{35}{4}$ = 8.75 ms — 2 marks

SRTF

Process	AT	BT	CT	TAT	WT
P_1	0	8 7	17	17	9
P_2	1	4	5	4	0
P_3	2	9	26	24	15
P_4	3	5	10	7	2
				52	26

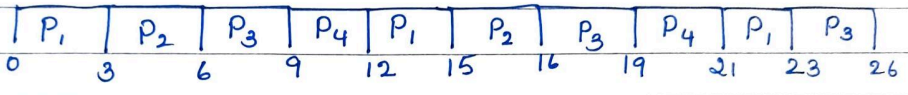


Average WT = 26/4 = 6.5 ms

Average TAT = 52/4 = 13 ms. — 3 marks

Round Robin (3 quantum time)

Process	AT	BT	CT	TAT	WT
P_1	0	8 5 2	23	23	15
P_2	1	4 1 0	16	15	11
P_3	2	9 6 3	26	24	15
P_4	3	5 2	21	18	13
				80	54



Average WT = 54/4 = 13.5 ms

Average TAT = 80/4 = 20.0 ms — 5 marks