USN | | | | | | | | | | |

Internal Assessment Test 1 – December 2023

Sub:	**Operating Systems**					Sub Code:	**BCS303**	Branch:	**AIML**
Date:	**20/12/23**	Duration:	**90 minutes**	Max Marks:	**50**	Sem/Sec:	**III -A,B,C**		**OBE**

		Answer any FIVE FULL Questions	**MARKS**	**CO**	**RBT**
1		Consider the following snapshot of processes. **JOBS** / **ARRIVAL TIME** / **BURST TIME** J1 / 0 / 9 J2 / 1 / 4 J3 / 2 / 9 J4 / 3 / 5 Calculate the Average Waiting Time and Average Turn Around Time if it is scheduled by (i) A person who counts the shortest remaining time (SRTF) among all the jobs. (ii) With time quantum of 2 Milliseconds for each job	[10]	2	L3
2		What is a thread? List out the benefits of Multithreading? Briefly explain the various multithreading models.	[10]	2	L1
3	a	Explain the implementation of inter-process communication using shared memory and message passing.	[5]	2	L2
	b	Distinguish between the following pairs of terms: User mode and kernel mode operations.	[5]	1	L3
4	a	Demonstrate the concept of virtual machine with an example.	[5]	1	L2
	b	Distinguish between the following terms: Multiprocessor systems and clustered systems.	[5]	1	L3
5		Is CPU scheduling necessary? Discuss the five different scheduling criteria's used in the computing scheduling mechanism.	[10]	2	L3
6		Explain process states with state transition diagram. Also explain PCB with a neat diagram.	[10]	2	L2

CI **CCI** **HoD**

----------All the Best----------

CO-PO Mapping																			
Course Outcomes		**Modules covered**	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**	**PSO1**	**PSO2**	**PSO3**	**PSO4**
CO1	Describe the Operating System Structure and Services.	**1**	3	-	-	-	-	-	-	-	-	-	-	3	-	2	-	-
CO2	Summarize the Process Management concepts like Processes, Threads, CPU Scheduling, Process Synchronization and Deadlocks	**1, 2**	3	2	2	-	-	-	-	-	-	-	-	3	-	2	-	-
CO3	Interpret the Memory Management concepts with respect to Main Memory and Virtual Memory.	**3, 4**	3	2	2	-	-	-	-	-	-	-	-	3	-	2	-	-
CO4	Discuss the Storage Management concepts like File-System Interface, File-System Implementation and Mass-Storage Structure	**4, 5**	3	2	2	-	-	-	-	-	-	-	-	3	-	2	-	-
CO5	Elucidate the Protection features in Operating System and case study in Linux OS.	**5**	3	2	2	-	-	-	-	-	-	-	-	3	-	2	-	-

SCHEME & SOLUTIONS

1) Consider the following snapshot of processes. (10) CO-2

JOBS	ARRIVAL TIME	BURST TIME
J1	0	9
J2	1	4
J3	2	9
J4	3	5

Calculate the Average Waiting Time and Average Turn Around Time if it is scheduled by
1.A person who counts the shortest remaining time (SRTF) among all the jobs.
2. With time quantum of 2 Milliseconds for each job

Explanation:
Gant chart SRTF : 3 Marks
Gant chart RR: 3 Marks
Results SRTF: 2 Marks
Results RR: 2 Marks

SRTF

J1	J2	J4	J1	J3
0	1	5	10	18 27

Process	Arrival Time	Burst Time	Completion Time	TAT	WT
J1	0	9	18	18	9
J2	1	4	5	4	0
J3	2	9	27	25	16
J4	3	5	10	7	2

AWT=(9+16+2+0)/4=6.75ms
ATAT=(18+4+25+7)/4=13.5ms

RR

J1	J2	J3	J4	J1	J2	J3	J4	J1	J3	J4	J1	J3	J1	J3
0	2	4	6	8	10	12	14	16	18	20	21	23	25	26 27

Waiting time J1=(0+8+16+21+25)/5=14
J2=(2+10)/2=6
J3=(4+12+18+23+26)/5=16.6
J4=(6+14+20)/3=13.3
AWT=12.475ms
Turn around Time J1=(2+10+18+23+26)/5=15.8
J2=(4+12)/2=6
J3=(6+14+20+25+27)/5=18.4
J4=(8+16+21)/3=15
ATAT= 13.8ms

2) What is a thread? List out the benefits of Multithreading? Briefly explain the various multithreading models. (10) CO- 2

Explanation:

Thread : 3 Marks
Multithreading: **3 Marks**
Types: **2 Marks**
Diagram: 2 Marks

*)A **thread** is the smallest unit of **execution within a process**. It represents an independent flow of execution, **sharing the same resources** (such as memory space) with other threads belonging to the same process.
*) Threads within a process can run **concurrently**, allowing for parallelism and **improved performance.** Threads share the same address space and resources, making communication and data sharing between them more efficient compared to separate processes.

Benefits of Multithreading:

Concurrency:

Multithreading enables concurrent execution of tasks, allowing multiple threads to run simultaneously within a process. This can lead to improved performance and responsiveness.

Resource Sharing:

Threads within the same process share the same resources, such as memory space. This facilitates efficient communication and data sharing between threads.
Responsiveness:

Multithreading can enhance the responsiveness of applications, especially in user interfaces. While one thread is performing a time-consuming task, other threads can continue to respond to user input.
Parallelism:

Threads can execute in parallel on multicore processors, utilizing the available CPU resources more effectively and potentially speeding up the execution of tasks.
Simplified Programming:

In some cases, multithreading can simplify program design. Different threads can be designed to perform specific tasks, making the code modular and easier to understand.

Efficient Task Management:

Threads can be used to break down complex tasks into smaller, more manageable units. Each thread can handle a specific aspect of the task, leading to more efficient task management.
Improved Utilization of System Resources:

Multithreading allows better utilization of system resources by keeping the CPU busy with multiple threads, reducing idle time and improving overall system performance.

Multithreading Models:

Many-to-One Model (User-Level Threads):

In this model, multiple user-level threads are mapped to a single kernel-level thread. The operating system is not aware of the existence of user-level threads, and thread management is handled entirely by the application.
One-to-One Model (Kernel-Level Threads):

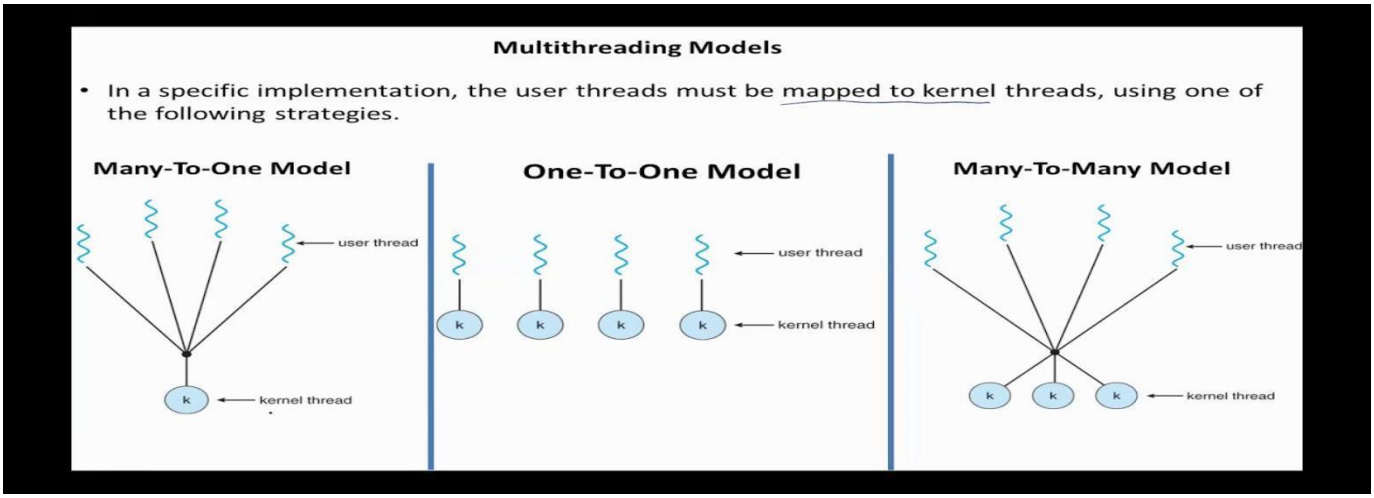
Each user-level thread corresponds to a separate kernel-level thread. This model provides more concurrency as multiple threads can execute in parallel. However, it may have higher overhead due to increased kernel involvement.

Many-to-Many Model (Hybrid Model):

This model allows for a many-to-many relationship between user-level threads and kernel-level threads. Multiple user-level threads are mapped to a smaller or equal number of kernel-level threads. It aims to combine the advantages of both user-level and kernel-level threads.

Two-Level Model:

In this model, a combination of user-level and kernel-level threads is used. The user-level threads are managed by a thread library, while a smaller number of kernel-level threads are managed by the operating system. The model aims to provide efficiency and flexibility.



3 a)Explain the implementation of inter-process communication using shared memory and message passing. (5) CO-2

IPC Definition: 1 Mark
Direct communication: 2 Marks
Indirect communication: 2 Marks

IPC:

Inter-process communication (IPC) is a mechanism that allows processes to communicate with each other and synchronize their actions. The communication between these processes can be seen as a method of co-operation between them. Processes can communicate with each other through both:

1.**Shared Memory**
2. Message passing

Direct and indirect communication :

Direct communication:The process which wants to communicate must explicitly name the recipient or sender of the communication.
e.g. **send(p1, message)** means send the message to p1.

Similarly, **receive(p2, message)** means to receive the message from p2.
*) In this method of communication, the **communication link gets established automatically**,
*) which can be either unidirectional or bidirectional,

*)but **one link** can be used between one pair of the sender and receiver and one pair of sender and receiver should not possess more than one pair of links.

*) Symmetry and asymmetry between sending and receiving can also be implemented i.e. either **both processes will name each other** for sending and receiving the messages or **only the sender will name the receiver** for sending the message and there is no need for the receiver for naming the sender for receiving the message.

*)The problem with this method of communication is that if the name of one process changes, this method will not work.

Indirect communication: processes **use mailboxes** (also referred to as ports) for sending and receiving messages. Each mailbox has a unique id and processes can communicate only if they share a mailbox.

*) Link established only if processes **share a common mailbox** and a single link can be associated with many processes.

*) Each pair of processes can share several communication links and these links may be **unidirectional or bi-directional.** Suppose two processes want to communicate through Indirect message passing, the required operations are:

*) **create** a mailbox, **use** this mailbox for sending and receiving messages, then **destroy** the mailbox.

*) The standard primitives used are: **send(A, message)** which means send the message to mailbox A.

*) The primitive for the receiving the message also works in the same way
e.g. **received (A, message)**.

Diagrams

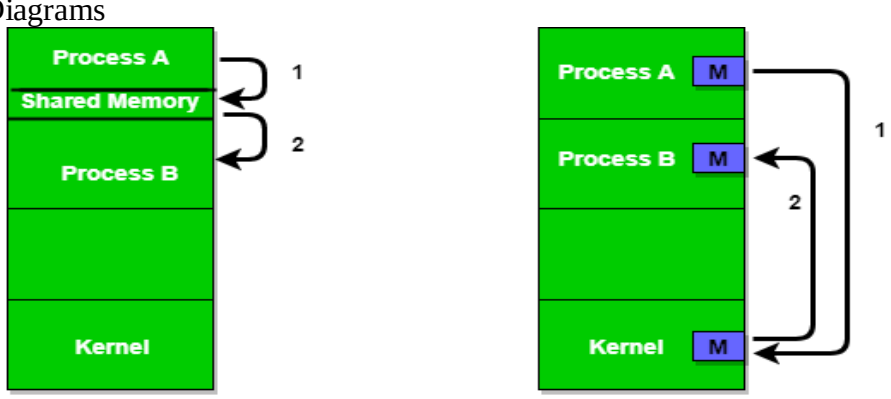


Figure 1 - Shared Memory and Message Passing

b)Distinguish between the following pairs of terms User mode and kernel mode operations.(5) CO-1

Explanation: 3 Marks
Diagram: 2 Marks

*) **User mode** is a restricted mode of operation in a computer system where user-level applications and processes run. In user mode, processes have limited access to system resources and cannot directly execute privileged instructions.

*) User mode is designed to provide a secure and isolated environment for user applications, protecting the core system components from unintended interference.

*) Processes in user mode have restricted access to system resources. They can only access their allocated memory space and have limited control over hardware devices.

*) In user mode, processes execute user-level instructions and have no direct access to the operating system's core functions or sensitive hardware instructions.

*) User mode processes do not handle exceptions and interrupts directly. When an exception or interrupt occurs, control is transferred to the operating system in kernel mode to handle the event.

*) User mode provides a layer of security and stability by isolating user-level processes from the critical functions of the operating system. This isolation helps prevent unauthorized access to sensitive resources.

*) **Kernel Mode**: (also called supervisor or privileged mode) is a mode of operation where the operating system's core functions, including system-level tasks and critical operations, are executed.

*) In kernel mode, the operating system has unrestricted access to the hardware and can execute privileged instructions.

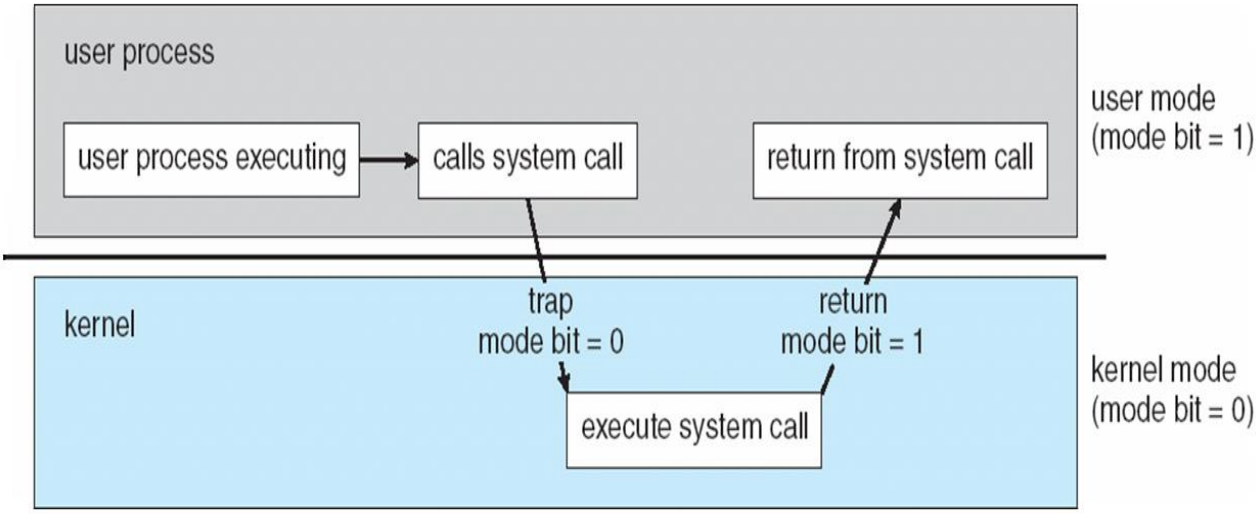
*) Kernel mode allows the operating system to perform tasks that require direct access to hardware resources, such as managing memory, handling interrupts, and controlling I/O devices.

*) Kernel mode allows unrestricted access to all system resources. The operating system, running in kernel mode, has the highest level of privilege and can execute privileged instructions.

*) Kernel mode provides full execution privileges, allowing the operating system to execute privileged instructions and perform critical tasks that are necessary for system management.

*) Kernel mode allows the operating system to handle exceptions and interrupts directly, enabling it to respond to events such as hardware interrupts, system calls, or page faults.

*) Kernel mode is less secure due to its elevated privileges. Incorrect or malicious operations in kernel mode can have a more profound impact on system stability and security.



4 a)Demonstrate the concept of virtual machine with an example. (5) CO-1

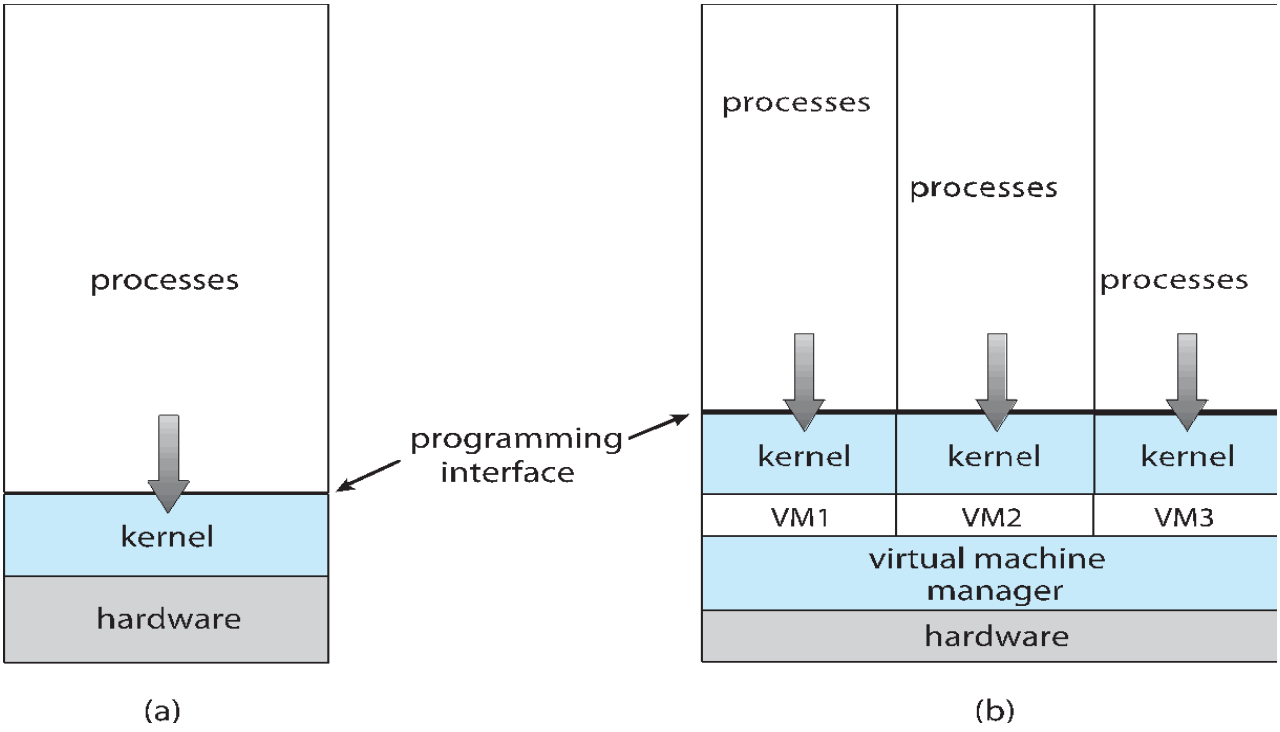
Explanation: 3 Marks
Diagram: 2 Marks

*) The fundamental idea behind a virtual machine is to abstract the hardware of a single computer (the CPU, memory, disk drives, network interface cards, and so forth) into several different execution environments. Thereby creating the illusion that each separate execution environment is running its own private computer.

*) Virtualization – OS natively compiled for CPU, running guest OSes also Natively compiled

*) Consider VMware running WinXP guests, each running applications, all on native WinXP host OS
VMM (virtual machine Manager) provides virtualization services

*) Use cases involve laptops and desktops running multiple OSes for exploration or compatibility
Apple laptop running Mac OS X host, Windows as a guest
Developing apps for multiple OSes without having multiple systems
QA testing applications without having multiple systems
Executing and managing compute environments within data centers
VMM can run natively, in which case they are also the host
There is no general purpose host then (VMware ESX and Citrix XenServer)



b)Distinguish between the following terms Multiprocessor systems and clustered systems.(5) CO-1

Explanation: 3 Marks
Diagrams-2 Marks

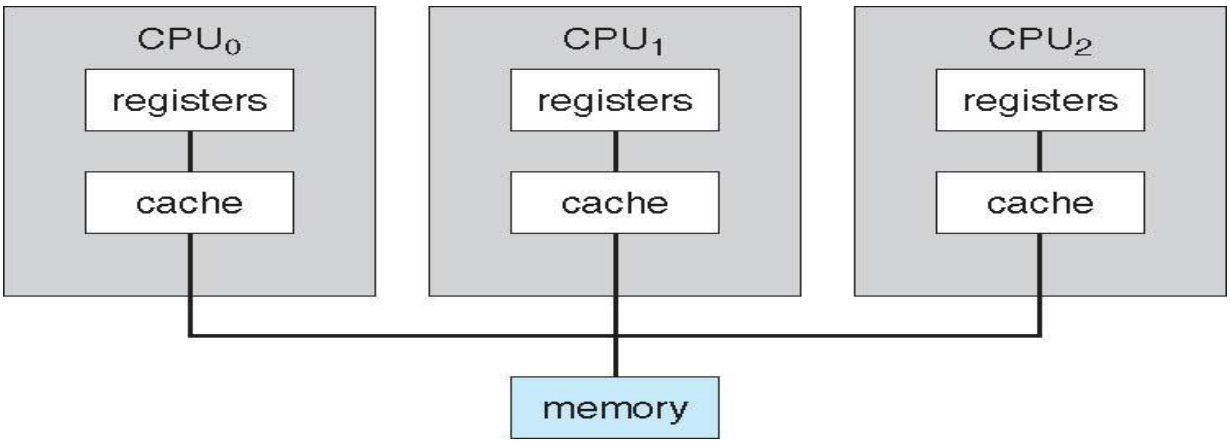
***)Multiprocessor** systems refer to computer systems that have multiple processors (or central processing units, CPUs) connected to a common memory. These processors share access to the same memory and I/O devices.

*) In multiprocessor systems, processors are typically connected through a shared bus or a network, enabling them to communicate and coordinate their activities.

*) Processors in a multiprocessor system communicate with each other by accessing shared data in the common memory. Inter-process communication is achieved through shared variables and messages.

*) Multiprocessor systems work together to execute a set of tasks or processes concurrently. They share the overall workload, and tasks can be divided among processors for parallel processing.

*)Eg: A symmetric multiprocessor (SMP) system is a common type of multiprocessor system where each processor has equal access to the memory and I/O devices.



Clustered Systems:

*) Clustered systems are composed of multiple independent computer systems (nodes or servers) that are interconnected to work together. Each node in a cluster is a separate computer system with its own

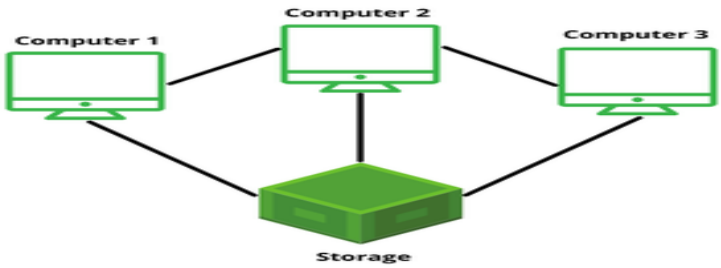
memory and resources.

*) In clustered systems, nodes are typically connected through a network. Unlike multiprocessor systems, each node in a cluster has its own local memory and may have its own local storage.

*) Communication between nodes in a clustered system occurs through the network. Processes on different nodes can communicate using message-passing mechanisms.

*) Nodes in a clustered system operate independently but can collaborate on specific tasks. Load balancing and task distribution may be managed by a cluster controller or a load balancer.

*)Eg: High-Performance Computing (HPC) clusters are examples of clustered systems. Each node in The cluster is a standalone computer, and tasks can be distributed across multiple nodes for parallel processing.



Differences:

*) In a multiprocessor system, processors share a common memory space, while in a clustered system, each node has its own local memory.

*) Multiprocessor systems are more tightly integrated, often sharing memory and I/O, whereas clustered systems are more loosely coupled, communicating over a network.

*) Communication in multiprocessor systems often involves shared memory mechanisms, while clustered systems rely on message passing or other inter-node communication mechanisms.

*) Multiprocessor systems aim for high performance through parallel processing, sharing a common workload, while clustered systems provide high availability and scalability by distributing tasks across independent nodes.

5) Is CPU scheduling necessary? Discuss the five different scheduling criteria used in the computing scheduling mechanism. (10) CO-2

Explanation: 7 Marks
Examples-3

*) Yes, CPU scheduling is necessary in modern computer systems because multiple processes often compete for the CPU's attention. The CPU scheduler, a component of the operating system, determines which process should be executed next. Efficient CPU scheduling is crucial for optimizing system performance, improving responsiveness, and ensuring fair resource allocation among competing processes.

Here are five different scheduling criteria used in CPU scheduling mechanisms:

CPU Utilization:

Definition: It measures the percentage of time the CPU is actively executing processes.
Objective: The goal is to keep the CPU as busy as possible to maximize system throughput.
Consideration: A high CPU utilization rate indicates effective use of system resources.

Example: An operating system aims to keep the CPU busy as much as possible. If the CPU is idle,

it is not contributing to the overall throughput of the system.

Throughput:

Definition: Throughput is the number of processes that are completed within a given time period.
Objective: The objective is to maximize the number of processes completed in a unit of time, contributing to overall system efficiency.
Consideration: High throughput implies efficient utilization of resources and a responsive system.

Example: In a batch processing system, the throughput is measured by the number of jobs completed per unit of time. A scheduling algorithm that maximizes throughput would be preferred in such a scenario.

Turnaround Time:

Definition: Turnaround time is the total time taken to execute a process, including both waiting time and execution time.
Objective: Minimizing turnaround time ensures that processes are completed and results are obtained As quickly as possible.
Consideration: Short turnaround time is essential for interactive systems and overall system responsiveness.
Example: For a user submitting a task to a system, turnaround time is the total time taken from the submission of the task to the completion of its execution. Short turnaround time is crucial for tasks that require quick results.

Waiting Time:

Definition: Waiting time is the total time a process spends waiting in the ready queue before getting CPU time.
Objective: Reducing waiting time enhances system responsiveness and overall user satisfaction.
Consideration: Efficient scheduling algorithms aim to minimize the waiting time of processes.
Example: Waiting time is the total time a process spends waiting in the ready queue before getting CPU time. A scheduling algorithm that minimizes waiting time is desirable to reduce the overall response time of the system.

Response Time:

Definition: Response time is the time elapsed between submitting a request (e.g., initiating a process) and receiving the first output.
Objective: Minimizing response time is crucial for interactive systems to provide users with quick feedback.
Consideration: Low response time contributes to a more interactive and user-friendly computing experience.
Example: Response time is the time elapsed between submitting a request (e.g., initiating a process) and receiving the first output. In an interactive system, minimizing response time is essential for providing a quick and interactive user experience.

Different scheduling algorithms prioritize these criteria differently based on the specific goals and characteristics of the system.

*) For example, real-time systems may prioritize response time, while batch processing systems may prioritize throughput and CPU utilization. The choice of a scheduling algorithm depends on the system's requirements and objectives.

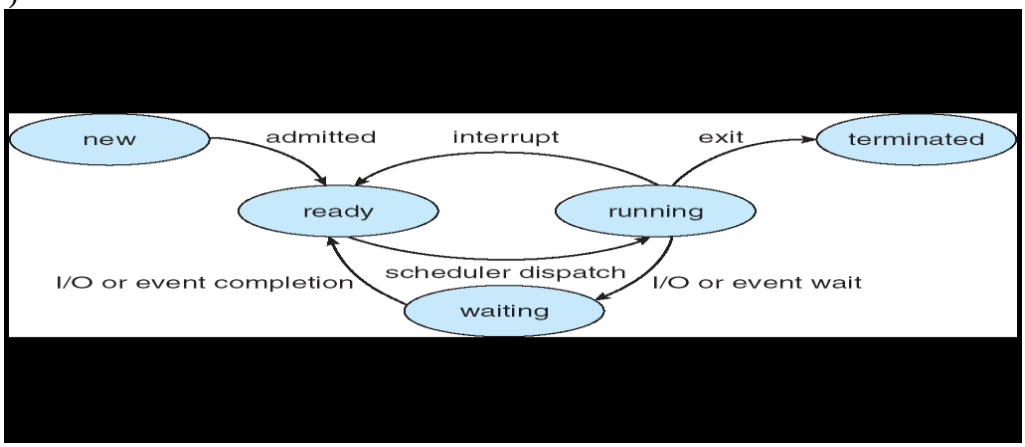
6) Explain process states with state transition diagram.Also explain PCB with a neat diagram.(10) CO-2

Process State:

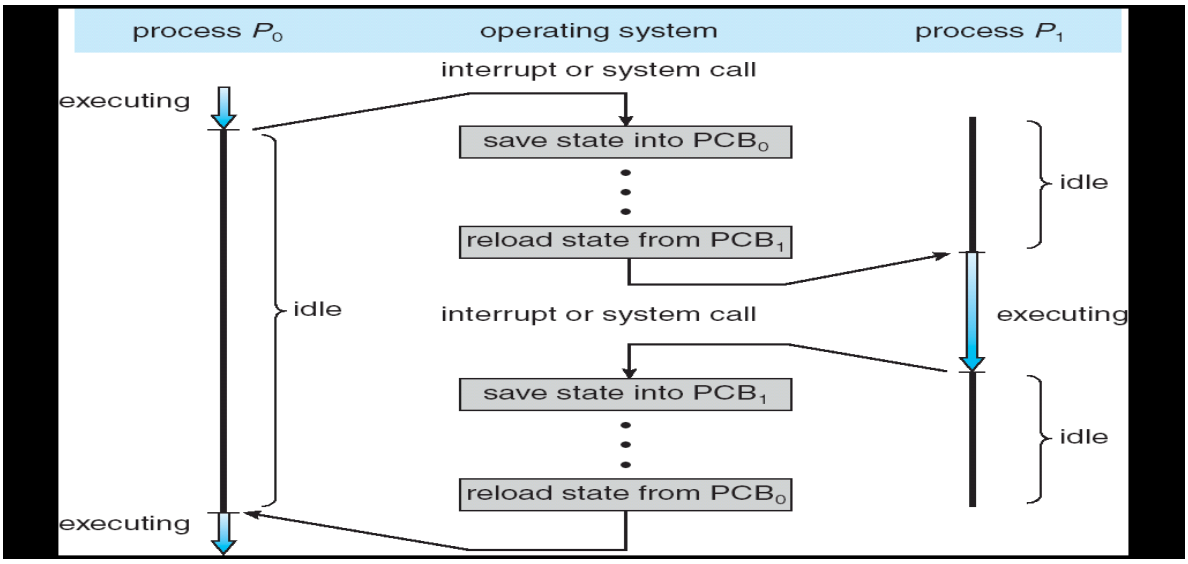
Explanation: 3 Marks
Diagram: 2 Marks

*) Process – a program in execution; process execution must progress in sequential fashion
A process includes:
program counter - processor's registers
Stack – temporary data (function parameters, local variable……
data section –global variable

*) As a process executes, it changes state
new: The process is being created
running: Instructions are being executed
waiting: The process is waiting for some event to occur(I/O completion or reception of a signal)
ready: The process is waiting to be assigned to a process
terminated: The process has finished execution
*)



PCB
Explanation: 3 Marks
Diagram: 2 Marks
PCB:
*) A PCB (Process Control Block) is a data structure used in the operating system to store all data related information to the process.
*) For example, when a process is created in the operating system, updated information of the process, switching information of the process, terminated process in the PCB.



when switching Process P1 to Process 2:

1.First, the context switching needs to save the state of process P1 in the form of the program counter and the registers to the PCB (Program Counter Block), which is in the running state.

2. Now update PCB1 to process P1 and moves the process to the appropriate queue, such as the ready queue, I/O queue and waiting queue.

3. After that, another process gets into the running state, or we can select a new process from the ready state, which is to be executed, or the process has a high priority to execute its task.

4. Now, we have to update the PCB (Process Control Block) for the selected process P2. It includes switching the process state from ready to running state or from another state like blocked, exit, or suspend.

5. If the CPU already executes process P2, we need to get the status of process P2 to resume its execution at the same time point where the system interrupt occurs.

6. calculate the context switch time between two running processes.:

If all the processes' total execution time was T, then the context switch time = T – (SUM for all processes (waiting time + execution time)).