CMR INSTITUTE OF TECHNOLOGY									
Internal Assesment Test – I Answer Key									
Sub:	Operating Systems							Code:	16MCA24
Date:	28.03.2017	Duration:	90 mins	Max Marks:	50	Sem:	II	Branch:	MCA

1(a)	Describe different types of processor registers.

A processor includes a set of registers that provide memory. But, this memory is faster and smaller than the main memory. These registers can be segregated into two types based on their functionalities as discussed in the following sections.

User – visible Registers
These registers enable the assembly language programmer to minimize the main memory references by optimizing register use. Higher level languages have an optimizing compiler which will make a choice between registers and main memory to store variables. Some languages like C allow the programmers to decide which variable has to be stored in register.

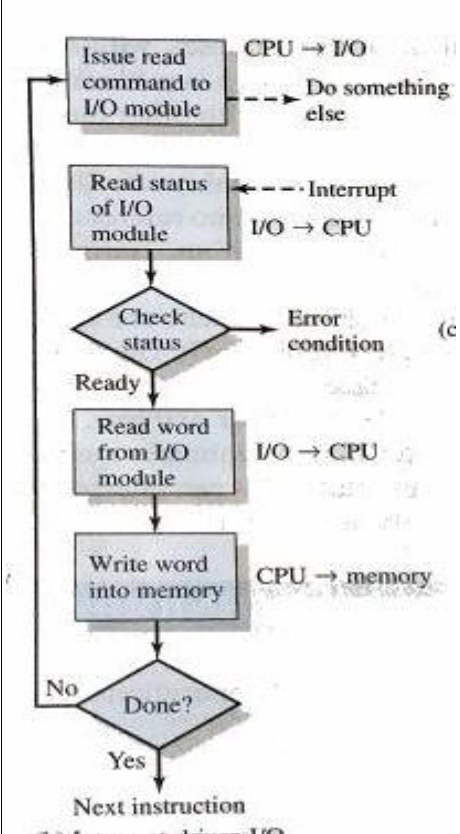
A user visible register is generally available to all programs. Types of registers that are available are: data, address and condition code registers.

Data Registers: They can be assigned to different types of functions by the programmer. Sometimes, these are general purpose and can be used with any machine instruction that performs operations on data. Still, there are some restrictions like – few registers are used for floating-point operations and few are only for integers.

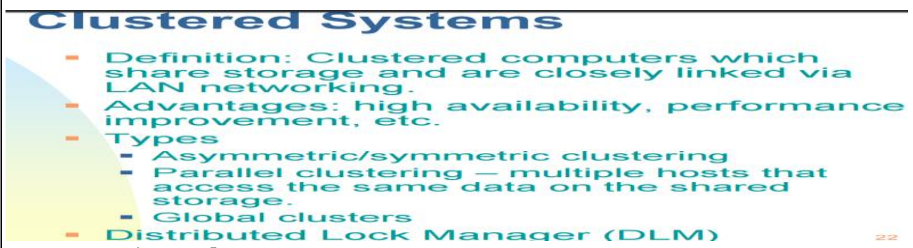
Address Registers: These registers contain main memory addresses of data and instructions. They may be of general purpose or may be used for a particular way of addressing memory. Few examples are as given below:
o **Index Registers:** Indexed addressing is a common mode of addressing which involves adding and index to a base value to get the effective address.
o **Segment Pointer:** In segmented addressing, a memory is divided into segments (a variable-length block of words). In this mode of addressing, a register is used to hold the base address of the segment.
o **Stack Pointer:** If there is a user-visible stack addressing, then there is a register pointing to the top of the stack. This allows push and pop operations on instructions stored in the stack.

Control and Status Registers
The registers used by the processor to control its operation are called as Control and Status registers. This registers are also used for controlling the execution of programs. Most of such registers are not visible to the user. Along with MAR, MBR, I/OAR and I/OBR discussed earlier, following registers are also needed for an instruction to execute:
 Program Counter: that contains the address of next instruction to be fetched.
 Instruction Register(IR): contains the instruction most recently fetched.

All processor designs also include a register or set of registers, known as ***program status word (PSW)***. It contains condition codes and status information like interrupt enable/disable bit and kernel/user mode bit.
Condition codes (also known as ***flags***) are bits set by the processor hardware as the result of operations. For example, an arithmetic operation may produce a positive, negative, zero or overflow result. Condition code bits are collected into one or more registers. And, they are the part of a control register. These bits only can be read to know the feedback of the instruction execution, but they can’t be altered

(b)	Explain Interrupt-driven I/O communication technique with a flowchart. The programmed I/O technique will degrade the performance of the processor. An alternative way is to provide interrupt-driven I/O. In this technique, the processor will issue an I/O command to the I/O module and then continue its regular instruction execution. When the I/O module is ready, it will interrupt the processor. Then the processor will execute the requested task and then resume its former processing.  Issue read command to I/O module CPU → I/O Do something else Read status of I/O module Interrupt I/O → CPU Check status Error condition Ready Read word from I/O module I/O → CPU Write word into memory CPU → memory Done? No Yes Next instruction (b) Interrupt-driven I/O

2(a) Discuss the following types of OS: i) Real Time ii) Cluster Systems iii) Handheld.



Clustered Systems

- Definition: Clustered computers which share storage and are closely linked via LAN networking.
- Advantages: high availability, performance improvement, etc.
- Types
  - Asymmetric/symmetric clustering
  - Parallel clustering – multiple hosts that access the same data on the shared storage.
  - Global clusters
- Distributed Lock Manager (DLM)

22

Asymmetric mode

- one machine is in hot standby mode while the other is running the applications.
- The hot standby host (machine) just monitors the active server.
- If that server fails, the hot standby host becomes the active server.

Symmetric mode

- two or more hosts are running applications, and they are monitoring each other. This mode is obviously more efficient, as it uses all of the available hardware

REAL-TIME SYSTEMS

A real-time system is used when there is a time requirements on the operation of a processor or the flow of data. Thus, it is used as a control device in a dedicated application.
Sensors bring data to the computer. The computer must analyze the data and possibly adjust controls to modify the sensor inputs. Systems that control scientific experiments,
medical imaging systems, industrial control systems, and certain display systems are realtime systems. Some automobile-engine fuel-injection systems, home-appliance controllers, and weapon systems are also real-time systems.

A real-time system has well-defined, fixed time constraints. Processing must be done within the defined constraints, or the system will fail. For instance, a robot arm must be instructed to halt before it reaches a wall. A real-time system functions correctly only if it returns the correct result within its time constraints. So in real-time system, a quick response is expected; whereas in batch system time constraints are not at all there.

- Real-time systems may be hard or soft.
- A hard real-time system guarantees that critical tasks be completed on time.
- A soft real-time system is less restrictive, where a critical real-time task gets priority over
- other tasks, and retains that priority until it completes.
- But, soft real-time systems have limited utility than hard real-time systems. Given their lack of deadline support, they are risky to use for industrial control and robotics.
- They are useful in areas like multimedia, virtual reality, and advanced scientific projects-such as undersea exploration and planetary rovers.

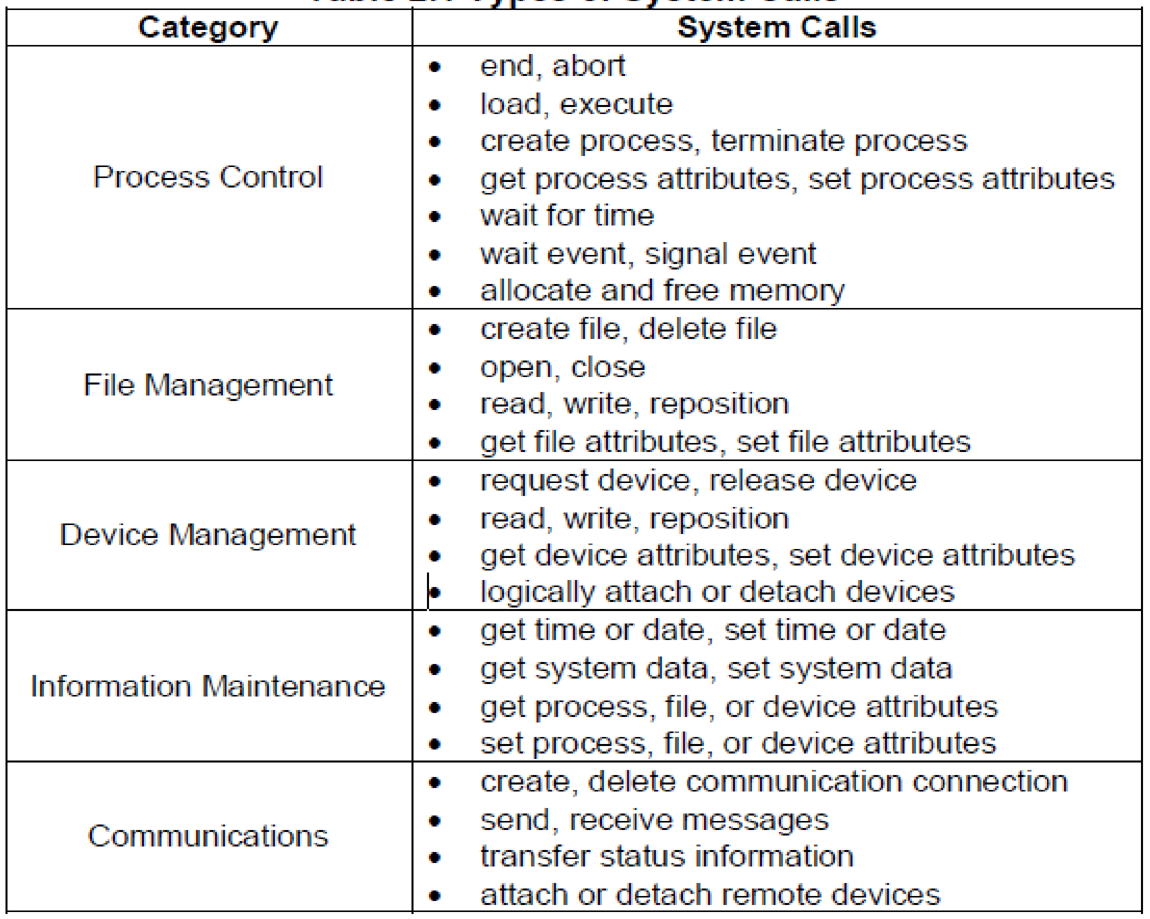
Hand-held systems: A portable computer that is small enough to be held in one's hand. Although extremely convenient to carry, handheld computers have not replaced notebook computers because of their small keyboards and screens. The most popular hand-held computers are those that are specifically designed to provide PIM (personal information manager) functions, such as a calendar and address book.

(b) Define system call. Classify the types of systems calls.

System calls provide the interface between a process and OS. These calls are normally assembly-language instructions and used by the assembly-language programmers. Some systems allow system calls to be made directly from a higher level language program. In such cases, the calls resemble predefined function or subroutine calls. They may generate a call to a special run-time routine that makes the system call or the system call may be generated directly in-line.

System calls can be grouped roughly into five major categories: process control, file management, device management, information maintenance, and communications. These are discussed in the following sections. Table 2.1 summarizes the types of system calls provided by OS.

Table 2.1 Types of System Calls



Category	System Calls
Process Control	• end, abort • load, execute • create process, terminate process • get process attributes, set process attributes • wait for time • wait event, signal event • allocate and free memory
File Management	• create file, delete file • open, close • read, write, reposition • get file attributes, set file attributes
Device Management	• request device, release device • read, write, reposition • get device attributes, set device attributes • logically attach or detach devices
Information Maintenance	• get time or date, set time or date • get system data, set system data • get process, file, or device attributes • set process, file, or device attributes
Communications	• create, delete communication connection • send, receive messages • transfer status information • attach or detach remote devices

3(a) Describe in detail about the components of operating system and its responsibilities.

1. Process Management
2. Main Memory Management
3. File Management
4. I/O System Management
5. Secondary Storage Management
6. Networking
7. Protection System
8. Command-Interpreter System

Process Management

A *process* is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.
• Processes can create sub-processes to execute concurrently.
• A program by itself is not a process; a program is a passive entity, whereas a process is an active entity.
• The execution of a process must progress in a sequential fashion. The CPU executes one instruction of the process after another until the process completes.
• Operating System processes: Those execute system code. • User processes: Those that execute user code. The operating system is responsible for the following activities in connection with process management.
– Process creation and deletion.
– Process suspension and resumption.
– Provision of mechanisms for:
1) process synchronization
2) process communication
– Deadlock handling

Main Memory Management

Memory is a large array of words or bytes, each with its own address. It is a repository (storage) of quickly accessible data shared by the CPU and I/O devices.
• Main memory is a volatile storage device. It loses its contents in the case of system failure.
• The operating system is responsible for the following activities in connections with memory management:
– Keep track of which parts of memory are currently being used and by whom.
– Decide which processes to load when memory space becomes available.
– Allocate and deallocate memory space as needed.

File Management

A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data.
• A file consists of a sequence of bits, bytes, lines, or records whose meanings are defined by their creators.
• The operating system is responsible for the following activities in connections with file management:
– File creation and deletion.
– Directory creation and deletion.
– Support of primitives for manipulating files and directories.
– Mapping files onto secondary storage.
– File backup on stable (nonvolatile) storage media.
• I/O System Management
The I/O system consists of:
– A buffer-caching system
– A general device-driver interface
– Drivers for specific hardware devices
• The O.S. hides the peculiarities of specific hardware devices from the user.

Secondary Storage Management
Since main memory (*primary storage*) is volatile and too small to accommodate all data and programs permanently, the computer system must provide *secondary storage* to back up main memory.
• Most modern computer systems use disks as the principle on-line storage medium, for both programs and data.
• The operating system is responsible for the following activities in connection with disk management:
– Free space management
– Storage allocation
– Disk scheduling

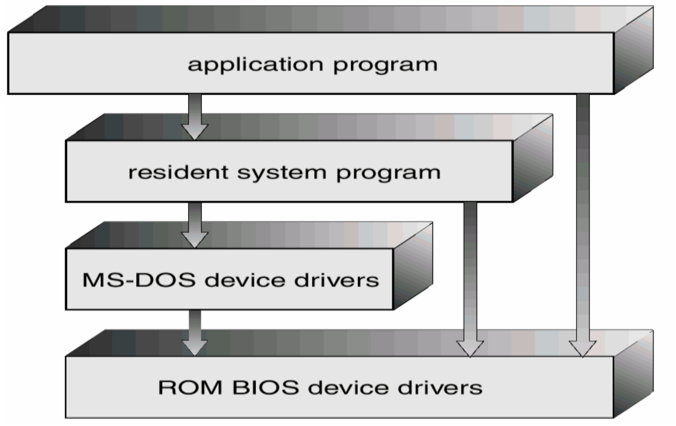
Networking
A *distributed* system is a collection of processors that do not share memory or a clock. Each processor has its own local memory and clock.
• The processors in the system are connected through a communication network.
• A distributed system provides user access to various system resources.
• Access to a shared resource allows:
– Computation speed-up
– Increased data availability
– Enhanced reliability

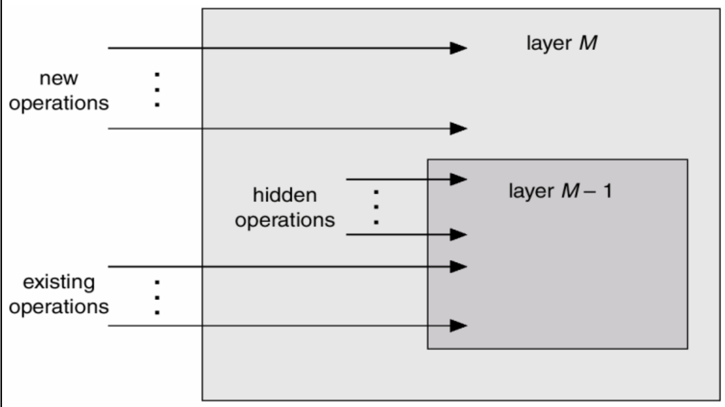
Protection System
• *Protection* refers to a mechanism for controlling access by programs, processes, or users to both system and user resources.
• The protection mechanism must:
– distinguish between authorized and unauthorized usage.
– specify the controls to be imposed.
– provide a means of enforcement.

Command-Interpreter System
Command-Interpreter system is a system program, which is the interface between the user and the operating system.
• Command-Interpreter system is known as the shell.
• Some operating systems provide a user-friendly interface (mouse-based window) such as, Macintosh and Microsoft Windows.
• Some operating systems provide text interface (commands are typed on keyboard) such as MS-DOS and Unix shells.
Many commands are given to the operating system by control statements which deal with:
– process creation and management
– I/O handling
– secondary-storage management
– main-memory management
– file-system access
– protection
– networking
• The program that reads and interprets control statements is called variously:
– control-card interpreter
– command-line interpreter
– shell (in UNIX)
• Its function is to get and execute the next command statement.

4(a) Summarize MS DOS layer structure with a neat diagram.

Many commercial systems do not have a well-defined structure. Frequently, such operating systems started as small, simple, and limited systems, and then grew into wider range. MSDOS is an example for one such OS. It was written to provide the most functionality in the least space, so it was not divided into modules carefully. Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated





In the layered approach, the OS is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.

The main advantage of layered approach is modularity. The layers are selected such that each uses functions (operations) and services of only lower-level layers. This approach simplifies debugging and system verification.

(b) Write a short note on symmetric multiprocessing (SMP) and its benefits.

- One of the popular approaches for providing parallelism is symmetric multiprocessors (SMPs), where processors are replicated.

Based on the communication among processors, MIMD can be further divided. If every processor has a dedicated memory, then each processing element is a self-contained computer. Communication among the computers is either via fixed path or via some network. Such a system is known as a ***cluster***. If the processors share a common memory, it is known as ***shared-memory multiprocessor.*** This again can be further divided into ***master/slave*** architecture and ***SMP.***

The master/slave architecture has disadvantages:

 A failure of the master brings down the whole system

 As master has to do all scheduling and process management, the performance may slow down.

But, in SMP, the kernel can execute on any processor and it allows portions of kernel to execute in parallel. Here, each processor does self-scheduling from the pool of available

- process or threads.

5(a) What is a process? With a neat diagram explain the different states of a process with single and multiple queue model.

PROCESS

A process can be defined in several ways:

 A program in execution
 An instance of a program running on a computer
 The entity that can be assigned to and executed on a processor
 A unit of activity characterized by the execution of a sequence of instructions, a current state, and an associated set of system resources.

Two essential elements of a process are:

 program code: which may be shared with other processes that are executing the same program
 Set of data: associated with that code

The various States of the Process are as Followings:-

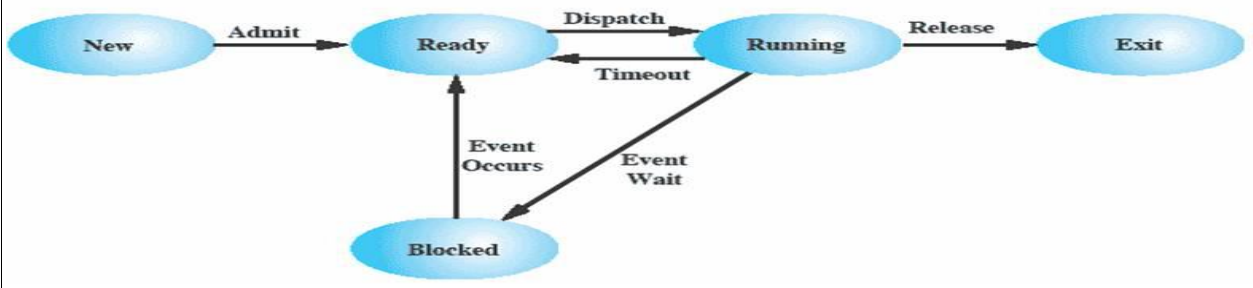
1) **New State:** When a user request for a Service from the System , then the System will first initialize the process or the System will call it an initial Process . So Every new Operation which is Requested to the System is known as the New Born Process.

2) **Running State:** When the Process is Running under the CPU, or When the Program is Executed by the CPU , then this is called as the Running process and when a process is Running then this will also provides us Some Outputs on the Screen.

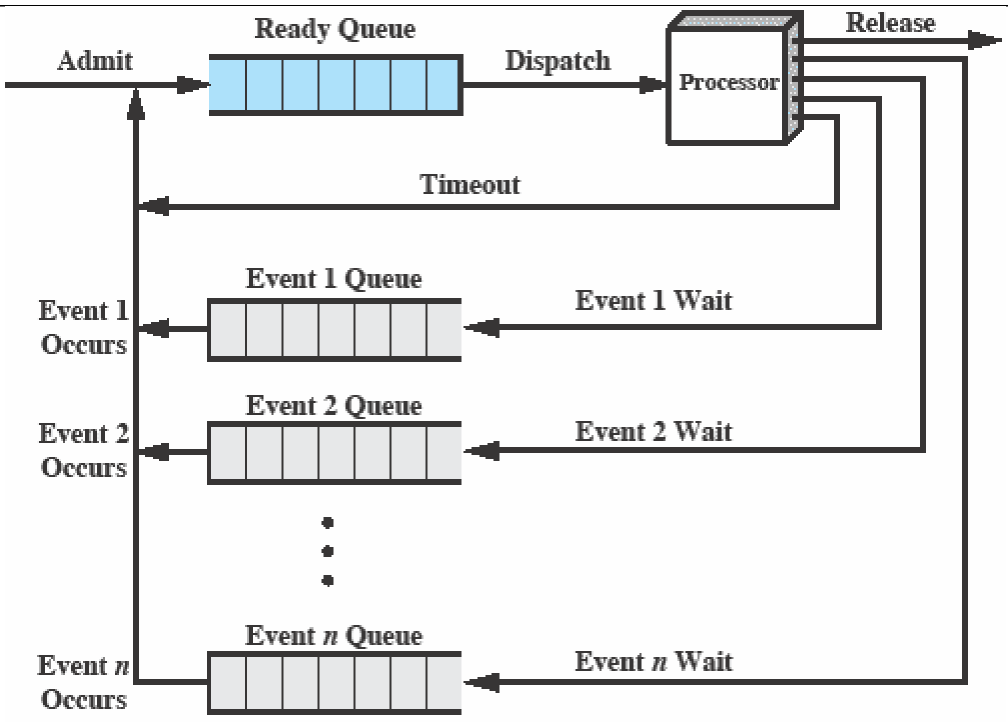
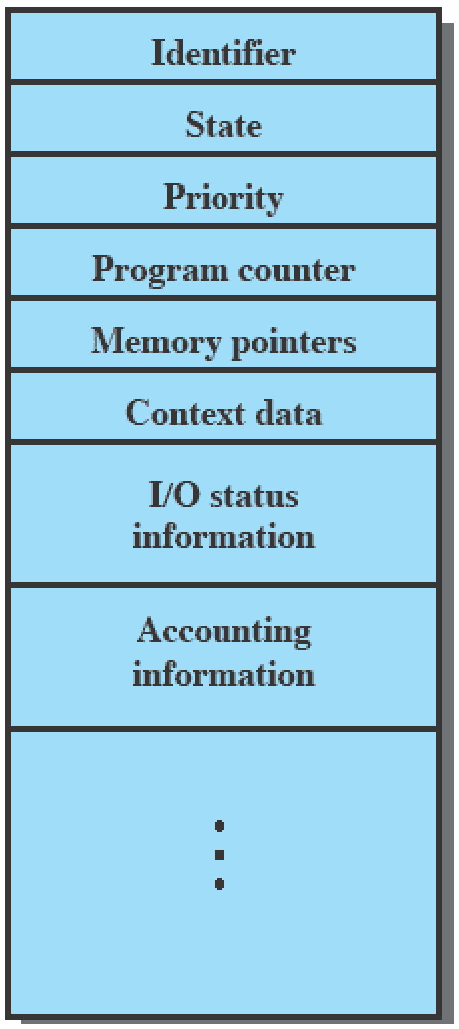
3) **Waiting:** When a Process is Waiting for Some Input and Output Operations then this is called as the Waiting State. And in this process is not under the Execution instead the Process is Stored out of Memory and when the user will provide the input then this will Again be on ready State.

4) **Ready State:** When the Process is Ready to Execute but he is waiting for the CPU to Execute then this is called as the Ready State. After the Completion of the Input and outputs the Process will be on Ready State means the Process will Wait for the Processor to Execute.

Terminated State: After the Completion of the Process , the Process will be Automatically terminated by the CPU . So this is also called as the Terminated State of the Process. After executing the whole Process the Processor will also de-allocate the Memory which is allocated to the Process. So this is called as the Terminated Process.



Five-state Process Model

	 Admit Ready Queue Dispatch Processor Release Timeout Event 1 Queue Event 1 Wait Event 1 Occurs Event 2 Queue Event 2 Wait Event 2 Occurs Event *n* Queue Event *n* Wait Event *n* Occurs Multiple Blocked queues
(b)	What is a PCB? What are its contents?  Identifier State Priority Program counter Memory pointers Context data I/O status information Accounting information At any given point in time, *while the program is executing*, this process can be uniquely characterized by a number of elements, including the following: **Identifier:** A unique identifier associated with this process, to distinguish it from all

other processes.
 State: If the process is currently executing, it is in the running state.
 Priority: Priority level relative to other processes.
 Program counter: The address of the next instruction in the program to be executed. **Memory pointers:** Includes pointers to the program code and data associated with this process, plus any memory blocks shared with other processes.
 Context data: These are data that are present in registers in the processor while the process is executing.
 I/O status information: Includes outstanding I/O requests, I/O devices (e.g., disk drives) assigned to this process, a list of files in use by the process, and so on.
 Accounting information: May include the amount of processor time and clock time used, time limits, account numbers, and so on.
The above information is stored in a data structure known as ***process control block*** as shown in Figure 3.1. It is created and managed by OS.

6(a) Define waiting time, throughput, and turnaround time?

Throughput: The number of processes completed per time unit is called as throughput.
 Turnaround time: The interval from the time of submission of a process to the time of completion is the turnaround time. Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.
 Waiting Time: The CPU-scheduling algorithm does not affect the amount of time during which a process executes or does I/O; it affects only the amount of time that a process spends waiting in the ready queue. Waiting time is the sum of the periods spent waiting in the ready queue.

(b) Consider the following set of processes with given length of CPU burst:

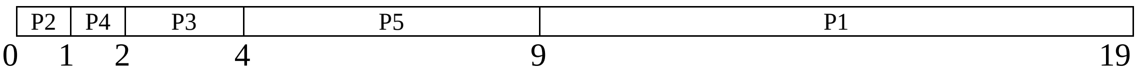
Processes	P1	P2	P3	P4	P5
Burst Time	10	1	2	1	5
Priority	3	1	3	4	2

All processes arrived at time 0 in the given order.

i) Draw Gantt chart using SJF (non-preemptive), Priority (Non-preemptive) [Smallest number implies highest priority], and Round Robin [Quantum-2 ms] scheduling policies.

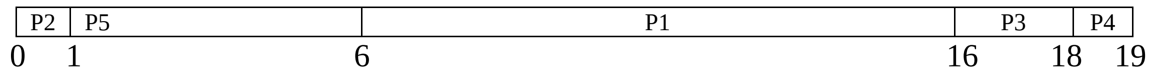
ii) Find the average waiting time for each scheduling policy.

SJF (Non preemptive)



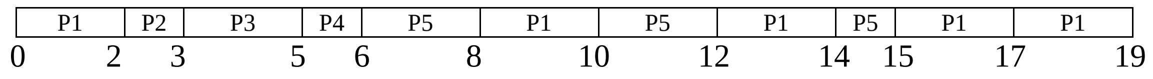
AWT = 9+0+2+1+4 = 16/5 = 3.2 ms

Priority (Non-preemptive)



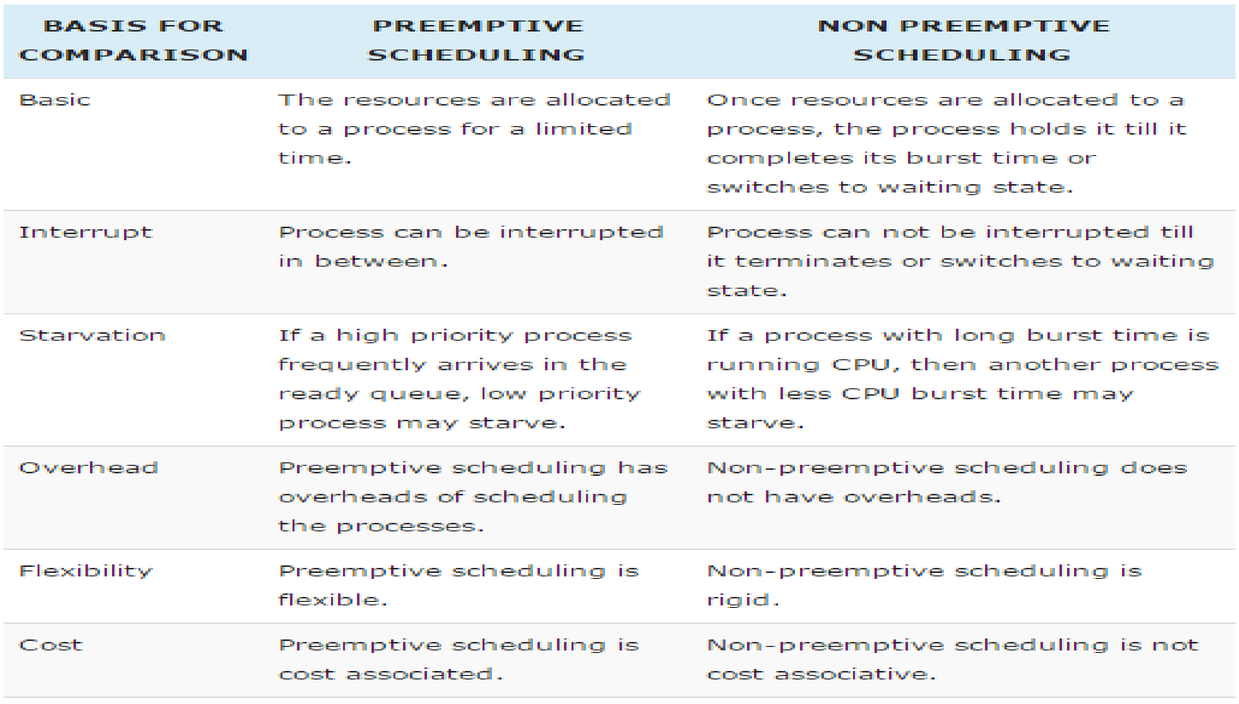
AWT = 6+0+16+18+1 = 41/5 = 8.2 ms

Round Robin



Waiting Time for P1 = 0+(8-2)+(12-10)+(15-14) = 0+6+2+1=9
Waiting Time for P2 = 2
Waiting Time for P3 = 3
Waiting Time for P4 = 5
Waiting Time for P5 = 6+(10-8)+(14-12) = 6+2+2=10
AWT = 9+2+3+5+10 = 29/5 = 5.8 ms

7(a) Compare preemptive with non-preemptive scheduling.



BASIS FOR COMPARISON	PREEMPTIVE SCHEDULING	NON PREEMPTIVE SCHEDULING
Basic	The resources are allocated to a process for a limited time.	Once resources are allocated to a process, the process holds it till it completes its burst time or switches to waiting state.
Interrupt	Process can be interrupted in between.	Process can not be interrupted till it terminates or switches to waiting state.
Starvation	If a high priority process frequently arrives in the ready queue, low priority process may starve.	If a process with long burst time is running CPU, then another process with less CPU burst time may starve.
Overhead	Preemptive scheduling has overheads of scheduling the processes.	Non-preemptive scheduling does not have overheads.
Flexibility	Preemptive scheduling is flexible.	Non-preemptive scheduling is rigid.
Cost	Preemptive scheduling is cost associated.	Non-preemptive scheduling is not cost associative.

(b) Explain in detail about OS design and implementation issues.

Design Goals

- The very first challenge in a system design is specifying the goals of the system.
- At the highest level, the design of the system will be affected by the choice of hardware and type of system: batch, time shared, single user, multiuser, distributed, real time, or general purpose.
- Beyond this highest level, the requirements can be divided into two types:
  - **User goals: operating system should be convenient to use, easy to learn, reliable,** safe, and fast.
  - **System goals: operating system should be easy to** design, implement, and maintain, as well as flexible, reliable, error-free, and efficient.

(c) List down the steps involved in the change of process state.

The steps involved in the change of process state are as below:

- Save context of processor including program counter and other registers
- Update the process control block of the process that is currently in the Running state
- Move process control block to appropriate queue – ready; blocked; ready/suspend
- Select another process for execution
- Update the process control block of the process selected
- Update memory-management data structures
- Restore context of the selected process

8(a) Define a thread. Explain the benefits of a multithreaded programming.

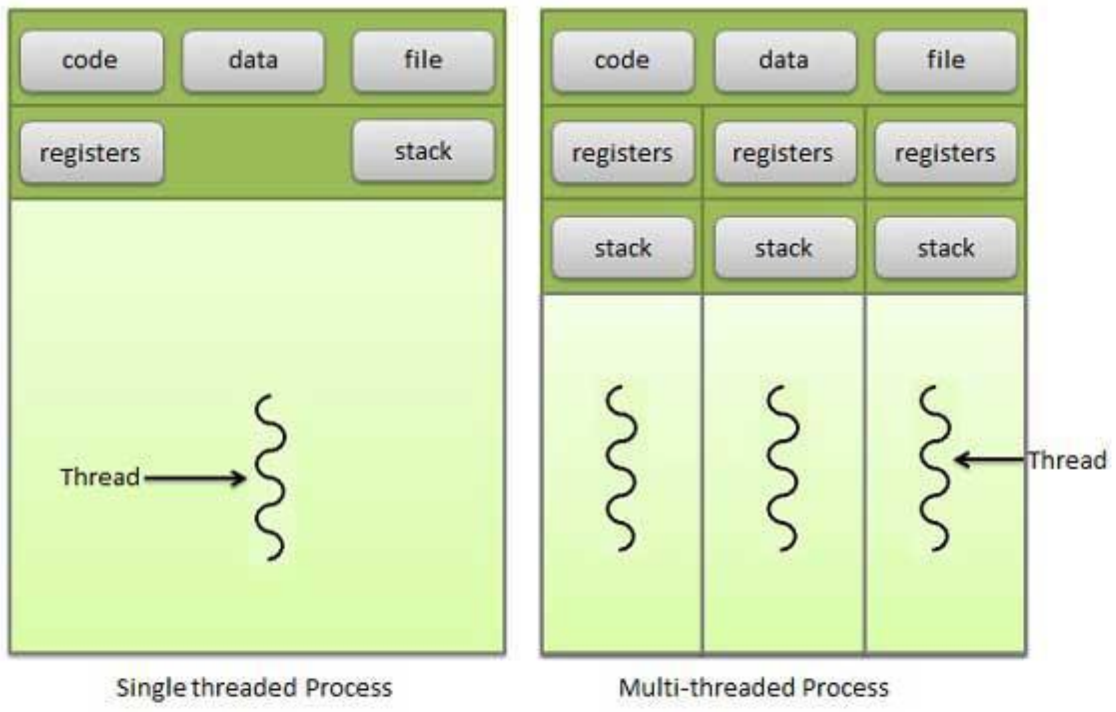
A thread is a flow of execution through the process code, with its own program counter, system registers and stack. A thread is also called a light weight process. Threads provide a way to improve application performance through parallelism. Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.

Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control. Threads have been successfully used in implementing network servers and web server. They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors. Following figure shows the working of the single and multithreaded processes.

The benefits of threads are:

- Thread takes less time to create compared to a process
- It takes less time to terminate compared to a process
- Switching between two threads takes less time than switching processes
- **Threads can communicate with each other without invoking the kernel**

(b) Explain three multi-threading models with a diagram.



Multithreading Models

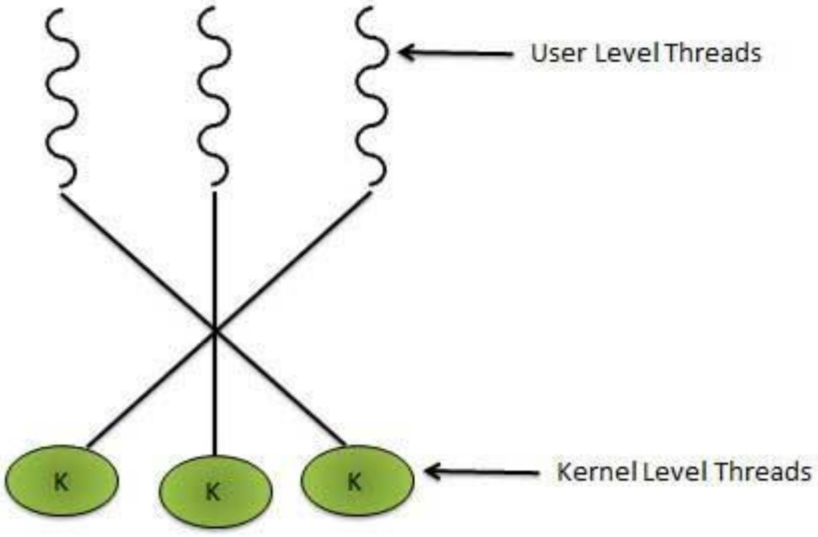
Some operating system provides a combined user level thread and Kernel level thread facility. Solaris is a good example of this combined approach. In a combined system, multiple threads within the same application can run in parallel on multiple processors and a blocking system call need not block the entire process. Multithreading models are three types

- Many to many relationship.
- Many to one relationship.
- One to one relationship.

Many to Many Model

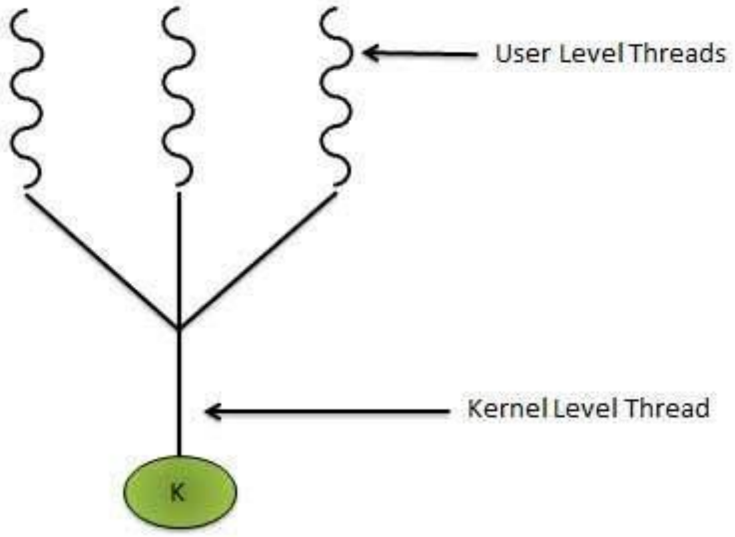
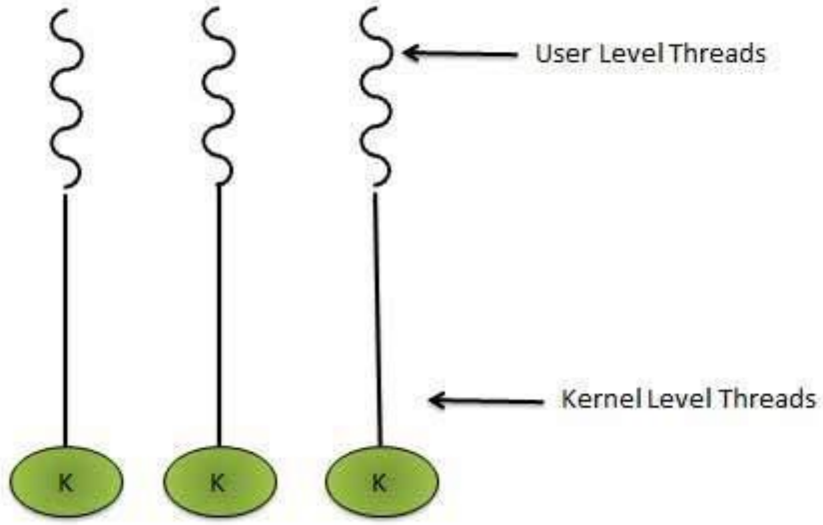
In this model, many user level thread multiplexes to the Kernel thread of smaller or equal numbers. The number of Kernel threads may be specific to either a particular application or a particular machine.

Following diagram shows the many to many model. In this model, developers can create as many user threads as necessary and the corresponding Kernel threads can run in parallels on a multiprocessor.



Many to One Model

Many to one model maps many user level threads to one Kernel level thread. Thread management is done in

	user space. When thread makes a blocking system call, the entire process will be blocked. Only one thread can access the Kernel at a time,so multiple threads are unable to run in parallel on multiprocessors. If the user level thread libraries are implemented in the operating system in such a way that system does not support them then Kernel threads use the many to one relationship modes.  **One to One Model** There is one to one relationship of user level thread to the kernel level thread. This model provides more concurrency than the many to one model. It also helps another thread to run when a thread makes a blocking system call. It support multiple thread to execute in parallel on microprocessors. Disadvantage of this model is that creating user thread requires the corresponding Kernel thread. OS/2, windows NT and windows 2000 use one to one relationship model.  User Level Threads Kernel Level Threads K K K