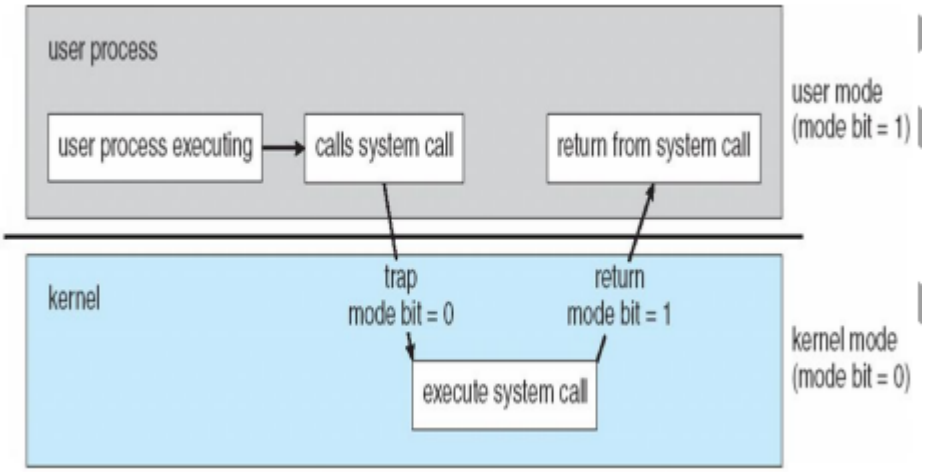
USN | | | | | | | | | | |

Internal Assessment Test I – Dec 2023

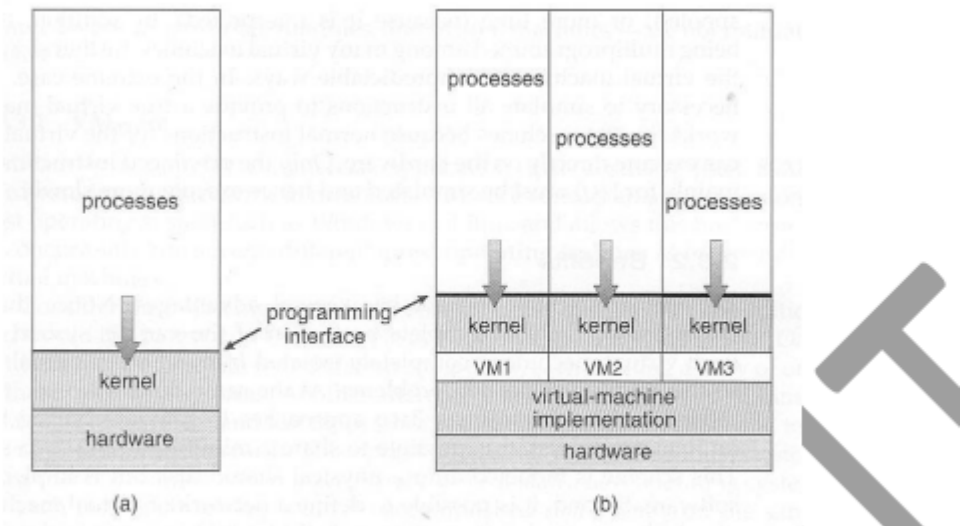
Operating Systems Scheme and Solution

Sub:	**Operating Systems**					Sub Code:	**BCS303**	Branch:	**CSE**	
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	a	**Explain the dual mode of operation of an operating system.(Diagram-2.5+Explaination 2.5)** Modern operating systems are interrupt-driven. If there are no processes to execute, no I/O devices to service, and no users to whom to respond, an operating system will sit quietly, waiting for something to happen. Events are signaled by the occurrence of an interrupt or a trap. A trap (or an exception) is a software-generated interrupt. For each type of interrupt, separate segments of code in the operating system determine what action should be taken. An interrupt service routine is provided that is responsible for dealing with the interrupt. a) Dual-Mode Operation Since the operating system and the user programs share the hardware and software resources of the computer system, it has to be made sure that an error in a user program cannot cause problems to other programs and the Operating System running in the system. The approach taken is to use a hardware support that allows us to differentiate among various modes of execution. The system can be assumed to work in two separate modes of operation: • user mode and • kernel mode (supervisor mode, system mode, or privileged mode). A hardware bit of the computer, called the mode bit, is used to indicate the current mode: kernel (0) or user (1). With the mode bit, we are able to distinguish between a task that is executed by the operating system and one that is executed by the user. When the computer system is executing a user application, the system is in user mode. When a user application requests a service from the operating system (via a system call), the transition from user to kernel mode takes place.	[5]	1	L1

					
	b	**What are system calls? Briefly point out its types. (System call-2.5+ Listing 2.5)** System calls is a means to access the services of the operating system. There are number of system calls used to finish this task. The first system call is to write a message on the screen (monitor). Then to accept the input filename. Then another system call to write message on the screen, then to accept the output filename. When the program tries to open the input file, it may find that there is no file of that name or that the file is protected against access. In these cases, the program should print a message on the console(another system call) and then terminate abnormally (another system call) and create a new one (another system call). Types: • 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 • 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	[5]	1	L1
2	a	**Explain in detail about the various services provided by the operating system.(Listing -2.5+ Explanation 2.5)** OS provide services for the users of the system, including:	[5]	1	L1

		• User Interfaces - Means by which users can issue commands to the system. Depending on the operating system these may be a command-line interface (e.g. sh, csh, ksh, tcsh, etc.), a Graphical User Interface (e.g. Windows, X-Windows, KDE, Gnome, etc.), or a batch command systems. In Command Line Interface(CLI)- commands are given to the system. In Batch interface – commands and directives to control these commands are put in a file and then the file is executed. In GUI systems- windows with pointing device to get inputs and keyboard to enter the text. • Program Execution - The OS must be able to load a program into RAM, run the program, and terminate the program, either normally or abnormally. • I/O Operations - The OS is responsible for transferring data to and from I/O devices, including keyboards, terminals, printers, and files. For specific devices, special functions are provided(device drivers) by OS. • File-System Manipulation – Programs need to read and write files or directories. The services required to create or delete files, search for a file, list the contents of a file and change the file permissions are provided by OS. • Communications - Inter-process communications, IPC, either between processes running on the same processor, or between processes running on separate processors or separate machines. May be implemented by using the service of OS- like shared memory or message passing. • Error Detection - Both hardware and software errors must be detected and handled appropriately by the OS. Errors may occur in the CPU and memory hardware (such as power failure and memory error), in I/O devices (such as a parity error on tape, a connection failure on a network, or lack of paper in the printer), and in the user program (such as an arithmetic overflow, an attempt to access an illegal memory location).			
	b	**Is it possible to have more than one OS in a computer system? If so, explain with the help of architecture.(Yes-2 Marks, Virtual Memory Diagram and explanation -3 Marks)** YES 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. Creates an illusion that a process has its own processor with its own memory. Host OS is the main OS installed in system and the other OS installed in the system are called guest OS. Benefits • Able to share the same hardware and run several different execution environments(OS). • Host system is protected from the virtual machines and the virtual machines are protected from one another. A virus in guest OS, will corrupt that OS but will not affect the other guest systems and host systems.	[5]	1	L2

		• Even though the virtual machines are separated from one another, software resources can be shared among them. Two ways of sharing s/w resource for communication are: a)To share a file system volume(part of memory). b)To develop a virtual communication network to communicate between the virtual machines. • The operating system runs on and controls the entire machine. Therefore, the current system must be stopped and taken out of use while changes are made and tested. This period is commonly called system development time. In virtual machines such problem is eliminated. User programs are executed in one virtual machine and system development is done in another environment. • Multiple OS can be running on the developer's system concurrently. This helps in rapid porting and testing of programmers code in different environments. • System consolidation – two or more systems are made to run in a single system.  **Figure 2.17 System modes. (A) Nonvirtual machine (b) Virtual machine**			
3	a	Elucidate the layered approach of OS structure with a supporting diagram **(Diagram-2.5+Explaination 2.5)**	[5]	1	L1

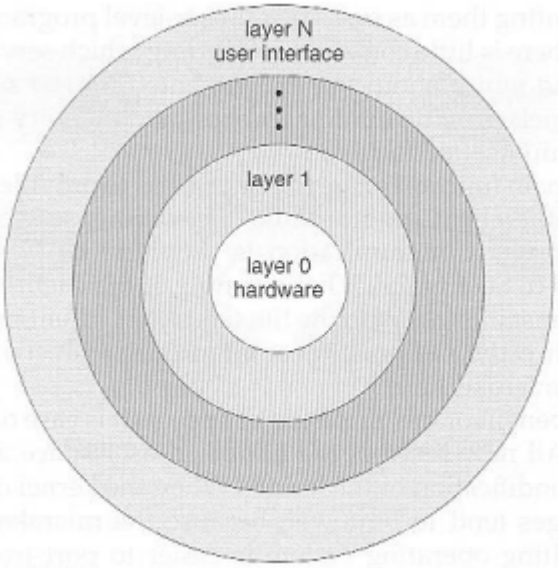


Figure 2.14 A layered Operating System The OS is broken into number of layers (levels). Each layer rests on the layer below it,

and relies on the services provided by the next lower layer.

• Bottom layer(layer 0) is the hardware and the topmost layer is the user interface.

• A typical layer, consists of data structure and routines that can be invoked by higher-level

layer.

Advantage of layered approach is simplicity of construction and debugging.

The layers are selected so that each uses functions and services of only lower-level layers. So

simplifies debugging and system verification. The layers are debugged one by one from the

lowest and if any layer doesn't work, then error is due to that layer only, as the lower layers are

already debugged. Thus the design and implementation is simplified.

A layer need not know how its lower level layers are implemented. Thus hides the operations

from higher layers.

Disadvantages of layered approach:

• The various layers must be appropriately defined, as a layer can use only lower level

layers.

• Less efficient than other types, because any interaction with layer 0 required from top

layer. The system call should pass through all the layers and finally to layer 0. This is an

overhead.

	b	**What is the difference between Program and Process? Explain how the Processes transit between different states with a diagram. (Diagram-1+Explaination 2.5+Difference 1.5)** Program: A program is a set of instructions or code written in a programming language that tells a computer how to perform a specific task or function	[5]	2	L2

Process:

A process, on the other hand, is an instance of a program that is being executed or run by the computer's operating system.

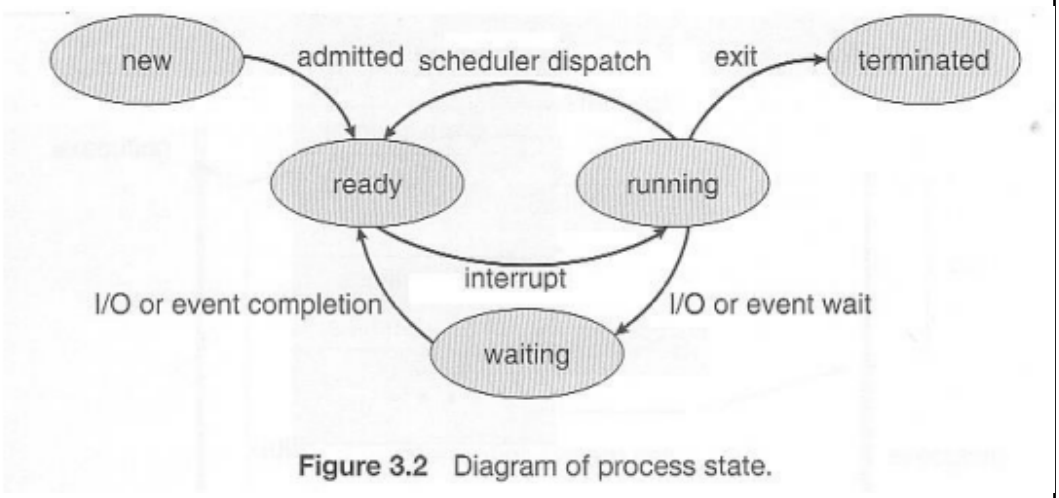


Figure 3.2 Diagram of process state.

A Process has 5 states. Each process may be in one of the following states –

• New - The process is in the stage of being created.

• Ready - The process has all the resources it needs to run. It is waiting to be assigned to the

processor.

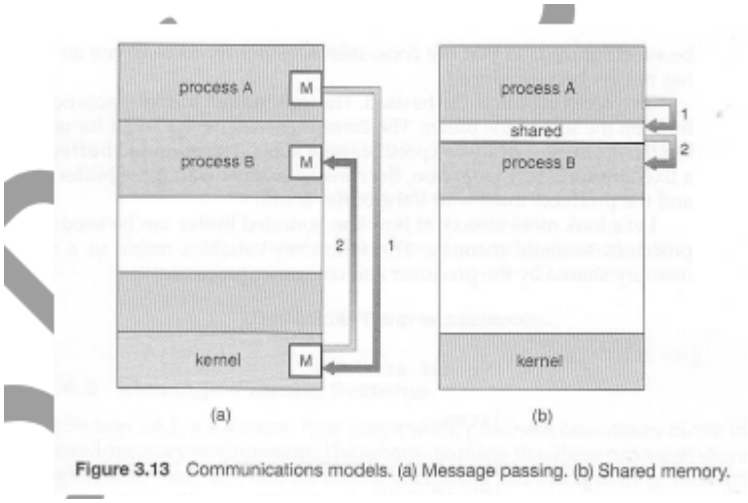
• Running – Instructions are being executed..

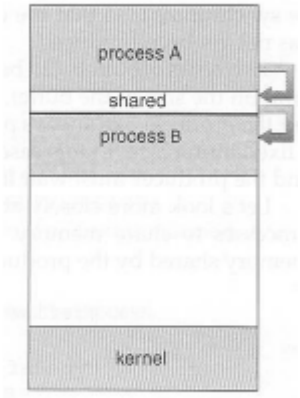
• Waiting - The process is waiting for some event to occur. For example the process may be

waiting for keyboard input, disk access request, inter-process messages, a timer to go off, or a

child process to finish.

• Terminated - The process has completed its execution.

4	a	**Describe the implementation of inter process communication using Shared Memory and Message Passing(Shared Memory-2.5+Message passing 2.5)** Cooperating processes require some type of inter-process communication. This is allowed by two models : 1) Shared Memory systems 2)Message Passing systems.  **Figure 3.13** Communications models. (a) Message passing. (b) Shared memory.	[5]	2	L1

		• Shared Memory is faster once it is set up, because no system calls are required and access occurs at normal memory speeds. Shared memory is generally preferable when large amounts of information must be shared quickly on the same computer. • Message Passing requires system calls for every message transfer, and is therefore slower, but it is simpler to set up and works well across multiple computers. Message passing is generally preferable when the amount and/or frequency of data transfers is small.  Message exchange is done among the processes by using objects. System call is used during every read and write operation. Message is communicated slowly.			
b		**Define the following(EACH -1 Mark)** 1. Booting Booting is basically the process of starting the computer. When the CPU is first switched on it has nothing inside the Memory. In order to start the Computer, load the Operating System into the Main Memory and then Computer is ready to take commands from the User 2. Multi Programming and Multi-Tasking Multiprogramming in an operating system as the name suggests multi means more than one and programming means the execution of the program. when more than one program can execute in an operating system then this is termed a multiprogramming operating system. Multi tasking operating systems allow multiple users to perform multiple tasks at the same time. The allocation of system resources such as input/output devices, CPU and memory among processes can be easily managed by multi-tasking operating system. 3. Scheduler and Dispatcher The CPU scheduler is a core component of an operating system responsible for managing the allocation of the CPU (Central Processing Unit) among multiple processes. Its primary objective is to maximize CPU utilization, enhance system performance, ensure fairness, and provide responsiveness to different tasks or processes running on the system.	[5]	2	L1

		A dispatcher is responsible for the Switching of Context, Switching to the user mode, Jumping into the user program's proper location for restarting that given program. 4. CPU Bound Process and IO Bound Process Processes can be described as either: I/O-bound process – spends more time doing I/O than computations, CPU-bound process – spends more time doing computations and few I/O operations. An efficient scheduling system will select a good mix of CPU-bound processes and I/O bound processes. • If the scheduler selects more I/O bound process, then I/O queue will be full and ready queue will be empty. If the scheduler selects more CPU bound process, then ready queue will be full and I/O queue will be empty.			
5	a	**Briefly discuss about the contents of PCB and explain how it helps OS in the process of Context Switch. (PCB- 2.5+ Context Switching 2.5)** For each process there is a Process Control Block (PCB), which stores the process-specific information as shown below – **Process State** – The state of the process may be new, ready, running, waiting, and so on. **Program counter** – The counter indicates the address of the next instruction to be executed for this process. **CPU registers** - The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward. **CPU scheduling information**- This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters. **Memory-management information** – This includes information such as the value of the base and limit registers, the page tables, or the segment tables. Accounting information – This information includes the amount of CPU and real-time used, time limits, account numbers, job or process numbers, and so on. **I/O status information** – This information includes the list of I/O devices allocated to the process, a list of open files, and so on. The PCB simply serves as the repository for any information that may vary from process to process.	[5]	2	L2

		The Process Control Block (PCB) plays a vital role in context switching within an operating system. The PCB is a data structure that the operating system uses to manage information about each individual process in the system. It contains crucial information necessary for the operating system to control and manage processes effectively. Here's how the PCB aids in context switching: Process State Information: PCB stores the current state of a process, including its program counter, CPU registers, scheduling information (like priority, process ID), and other relevant details. When a context switch occurs, the contents of these registers are saved into the PCB of the current process. Memory Management Information: PCB holds data about the memory management of the process, such as the memory allocation and page tables associated with that process. This information is crucial for ensuring that the process can resume execution correctly when it gets CPU time again. I/O Status Information: PCB maintains details about the process's I/O operations, such as open files, pending I/O requests, or device pointers. Saving this information is crucial during a context switch to ensure that ongoing I/O operations are not lost or interrupted. Process Identification: PCB contains unique identifiers for each process, like Process ID (PID), which helps the operating system keep track of and manage all active processes.			
	b	**What is a thread? List out the benefits of Multithreading? Briefly explain the various multithreading models. (Definition -1+Benefits-2+Types-2)** A thread is a basic unit of CPU utilization. It consists of a thread ID, program counter, a stack, and a set of registers. The four major benefits of multi-threading are: 1. Responsiveness - One thread may provide rapid response while other threads are blocked or slowed down doing intensive calculations. Multi threading allows a program to continue running even if part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user. 2. Resource sharing - By default threads share common code, data, and other resources, which allows multiple tasks to be performed simultaneously in a single address space. 3. Economy - Creating and managing threads is much faster than performing the same tasks for processes. Context switching between threads takes less time. 4. Scalability, i.e. Utilization of multiprocessor architectures – Multithreading can be greatly utilized in a multiprocessor architecture. A single threaded process can make use of only one CPU, whereas the execution of a multi-threaded application may be split among the available processors. Multithreading on a multi-CPU machine increases concurrency. In a single processor architecture, the CPU generally moves between each thread so quickly as to create an illusion of parallelism, but in reality only one thread is running at a time. The four major benefits of multi-threading are:	[5]	2	L1

1. Responsiveness - One thread may provide rapid response while other threads are blocked or slowed down doing intensive calculations.
Multi threading allows a program to continue running even if part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user.
2. Resource sharing - By default threads share common code, data, and other resources, which allows multiple tasks to be performed simultaneously in a single address space.
3. Economy - Creating and managing threads is much faster than performing the same tasks for processes. Context switching between threads takes less time.
4. Scalability, i.e. Utilization of multiprocessor architectures – Multithreading can be greatly utilized in a multiprocessor architecture. A single-threaded process can make use of only one CPU, whereas the execution of a multi-threaded application may be split among the available processors.

Multithreading on a multi-CPU machine increases concurrency. In a single processor architecture, the CPU generally moves between each thread so quickly as to create an illusion of parallelism, but in reality only one thread is running at a time.

a) Many-To-One Model

In the many-to-one model, many user-level threads are all mapped onto a single kernel thread.

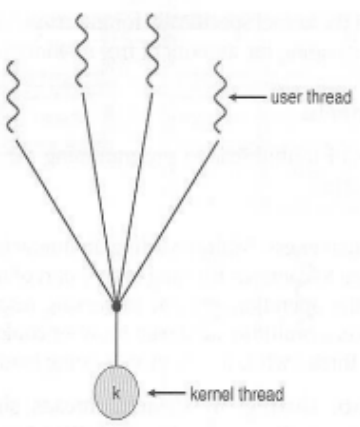


Figure 4.2 Many-to-one model.

- Thread management is handled by the thread library in user space, which is very efficient.
- If a blocking system call is made by one of the threads, then the entire process blocks. Thus blocking the other user threads from continuing the execution.
- Only one user thread can access the kernel at a time, as there is only one kernel thread. Thus the threads are unable to run in parallel on multiprocessors.
- Green threads of Solaris and GNU Portable Threads implement the many-to-one model.

b) One-To-One Model

- The one-to-one model creates a separate kernel thread to handle each user thread.
- One-to-one model overcomes the problems listed above involving blocking system calls and the splitting of processes across multiple CPUs.
- However the overhead of managing the one-to-one model is more significant, involving more overhead and slowing down the system.
- This model places a limit on the number of threads created.
- Linux and Windows from 95 to XP implement the one-to-one model for threads.

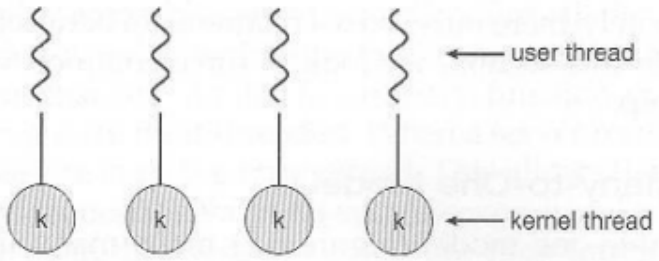


Figure 4.3 One-to-one model.

c) Many-To-Many Model

The many-to-many model multiplexes any number of user threads onto an equal or smaller number of kernel threads, combining the best features of the one-to-one and many-to-one models.

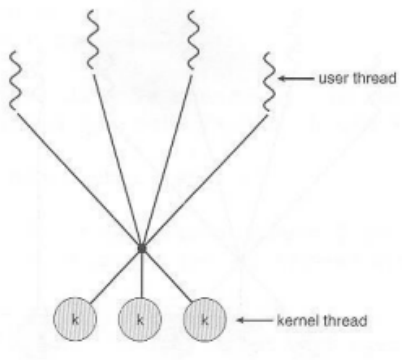


Figure 4.4 Many-to-many model.

- Users have no restrictions on the number of threads created.
- Blocking kernel system calls do not block the entire process.
- Processes can be split across multiple processors.
- Individual processes may be allocated variable numbers of kernel threads, depending on the number of CPUs present and other factors.
- This model is also called as two-tier model.
- It is supported by operating system such as IRIX, HP-UX, and Tru64 UNIX.

6 Consider the following snapshot of processes. [10] 2 L3

PROCESS	ARRIVAL TIME	BURST TIME	PRIORITY
P1	0	4	2
P2	1	3	3
P3	2	1	4
P4	3	5	5
P5	4	2	5

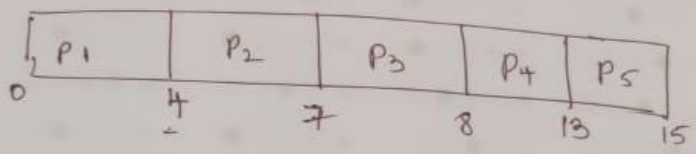
Calculate the Average Waiting Time and Average Turn Around Time if it is scheduled by

(i) Printing Machine in your Lab

		(ii) A person who counts the shortest remaining time among all the jobs. **(iii) Non Preemptive Priority Scheduler** **(iv) Preemptive Priority Scheduler(Answer is same as III) (Each-2 Marks)**			

Process	A.T	B.T	Priority
P_1	0	4	2
P_2	1	3	3
P_3	2	1	4
P_4	3	5	5
P_5	4	2	5

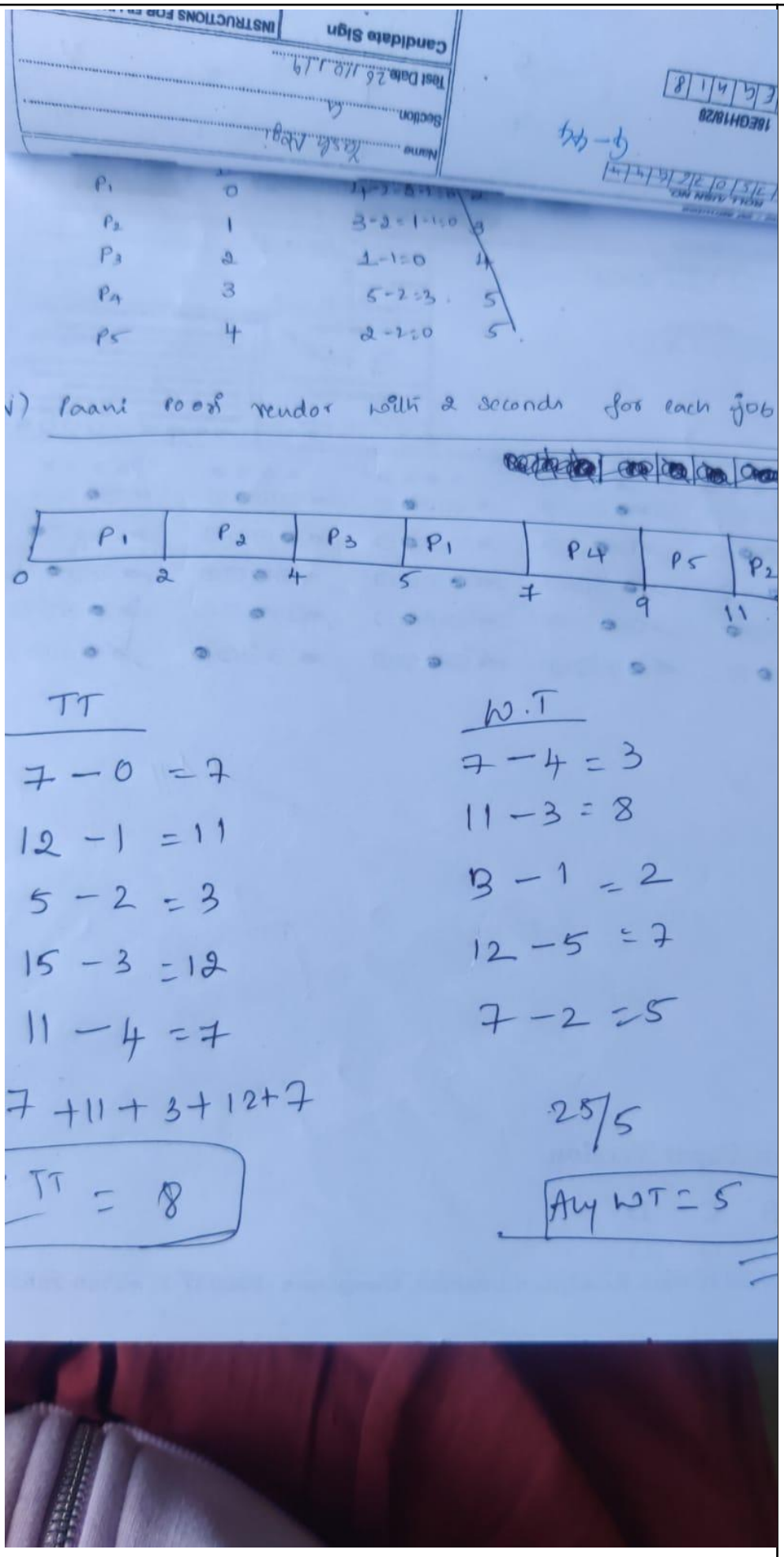
(i) Printing Machine in your lab (FCFS)



T.T. (Exit - AT)	W.T (TT - BT)
4 − 0 = 4	4 − 4 = 0
7 − 1 = 6	6 − 3 = 3
8 − 2 = 6	6 − 1 = 5
13 − 3 = 10	10 − 5 = 5
15 − 4 = 11	11 − 2 = 9

$\frac{4+6+6+10+11}{5}$ | $\frac{0+3+5+5+9}{5}$

Avg TT = 7.4 | Avg W.T 22/5 = 4.4

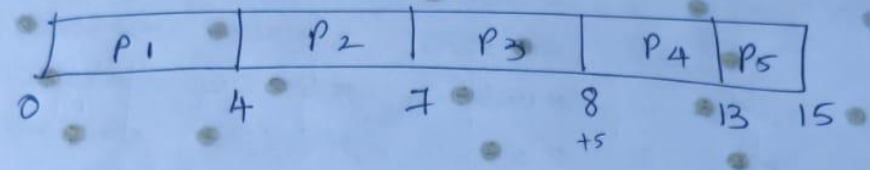


(v)

(vi) Paani Poori Vendor with 2 seconds for each job

	A.T	B.T	Priority
P_1	0	4	2
P_2	1	3	3
P_3	2	1	4
P_4	3	5	5
P_5	4	2	5

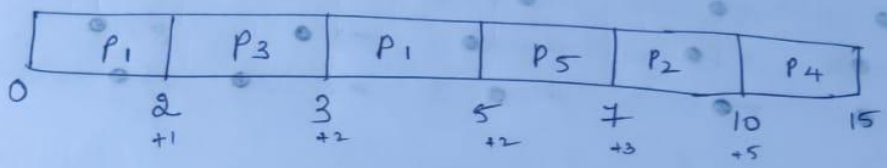
(ii) Non-Preemptive priority Scheduler



T.T	W.T
4 − 0 = 4	4 − 4 = 0
7 − 1 = 6	6 − 3 = 3
8 − 2 = 6	6 − 1 = 5
13 − 3 = 10	10 − 5 = 5
15 − 4 = 11	11 − 2 = 9
Avg T.T = 37/5	Avg W.T = 22/5
Avg TT = 7.4	Avg W.T = 4.4

	A.T	B.T
P_1	0	4-1 = 3-1 = 2-2 = 0
P_2	1	3-3 = 0
P_3	2	1-1 = 0
P_4	3	5
P_5	4	2-2 = 0

(i:) A person who counts the Shortest Remaining time arrory all the jobs.



T.T (exit - AT)

5 - 0 = 5

10 - 1 = 9

3 - 2 = 1

15 - 3 = 12

7 - 4 = 3.

(5 + 9 + 1 + 12 + 3) / 5

TT Avg: 6

W.T (TT - B.T)

5 - 4 = 1

9 - 3 = 6

1 - 1 = 0

12 - 5 = 7

2 - 2 = 1.

(1 + 6 + 0 + 7

Avg W.T = 3

CI CCI HoD