Operating System

Unit – 2
(Part-A)
Process

Mayank Mishra
School of Electronics Engineering
KIIT-Deemed to be University

Scheduling Criteria

Different Times in CPU Scheduling (Basic Terminologies)

Arrival Time (AT) :

- The time at which the process enters the Ready Queue or Ready State.
- It is a point of time not a time duration.

Burst Time (BT):

- Time required by a process to get executed on CPU.
- It is a time duration.
- It depends on the CPU that how much time CPU will take to execute that process.

Completion Time (CT) :

- The time at which process complete its execution.
- It is a point of time not a time duration.

Different Times in CPU Scheduling (Basic Terminologies)

Turn Around Time (TAT) :

- The time taken by the CPU since the Process has been ready to execute or since the process is in Ready Queue is known as Turn Around Time. The Turn Around Time can be calculated with the help of Completion Time and Arrival Time.
- TAT = CT - AT

Waiting Time (WT) :

- The time the Process has been waiting to complete its process since the assignment of process for completion is known as Waiting Time.
- WT = TAT - BT

Response Time (RT) :

- RT = (The time at which process gets CPU first time) – (AT)

Scheduling Criteria

- CPU scheduling is the process of determining which process or task is to be executed by the central processing unit (CPU) at any given time.

- It is an important component of modern operating systems that allows multiple processes to run simultaneously on a single processor.

- The CPU scheduler determines the order and priority in which processes are executed and allocates CPU time accordingly, based on various criteria such as CPU utilization, throughput, turnaround time, waiting time, and response time.

- Scheduling algorithm is a way of selecting a process from ready queue and put it in the CPU. Different CPU-scheduling algorithms have different properties, and the choice of a particular algorithm may favor one class of processes over another.

Importance of Scheduling Criteria

- **Efficient resource utilization** − By maximizing CPU utilization and throughput, CPU scheduling ensures that the processor is being used to its full potential. This leads to increased productivity and efficient use of system resources.

- **Fairness** − CPU scheduling algorithms that prioritize waiting time and response time help ensure that all processes have a fair chance to access the CPU. This is important in multi-user environments where multiple users are competing for the same resources.

- **Responsiveness** − CPU scheduling algorithms that prioritize response time ensure that processes that require immediate attention (such as user input or real-time systems) are executed quickly, improving the overall responsiveness of the system.

- **Predictability** − CPU scheduling algorithms that prioritize turnaround time provide a predictable execution time for processes, which is important for meeting deadlines and ensuring that critical tasks are completed on time.

Scheduling Criteria:

- **CPU utilization**– We want to keep the CPU as busy as possible. Conceptually, CPU utilization can range from 0 to 100 percent. In a real system, it should range from 40 percent (for a lightly loaded system) to 90 percent (for a heavily loaded system).(CPU utilization can be obtained by using the *top* command on Linux, macOS, and UNIX systems).

- **Throughput** – If the CPU is busy executing processes, then work is being done. One measure of work is the number of processes that are completed per time unit, called **throughput**. For long processes, this rate may be one process over several seconds; for short transactions, it may be tens of processes per second.

- **Turnaround time** - From the point of view of a particular process, the important criterion is how long it takes to execute that process. 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 in the ready queue, executing on the CPU, and doing I/O.

Scheduling Criteria:

- **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.

- **Response time** – In an interactive system, turnaround time may not be the best criterion. Often, a process can produce some output fairly early and can continue computing new results while previous results are being output to the user. Thus, another measure is the time from the submission of a request until the first response is produced. This measure, called response time, is the time it takes to start responding, not the time it takes to output the response.

Scheduling Algorithms

- CPU scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU's core.

- **Scheduling algorithms is a way of selecting a process from 'ready queue' and put it in the CPU.**

- Although most modern CPU architectures have multiple processing cores, we describe these scheduling algorithms in the context of only one processing core avail able. That is, a single CPU that has a single processing core, thus the system is capable of only running one process at a time.

- Scheduling Algorithms help out which process should be executed for how much time.

First-Come, First-Served (FCFS) Scheduling

- By far the simplest CPU-scheduling algorithm is the first-come first-serve (FCFS) scheduling algorithm.
- With this scheme, the process that requests the CPU first is allocated the CPU first.
- The implementation of the FCFS policy is easily managed with a FIFO queue.
- When a process enters the ready queue, its PCB is linked onto the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue. The running process is then removed from the queue.

First-Come, First-Served (FCFS) Scheduling

- On the negative side, the average waiting time under the FCFS policy is often quite long.

- ***Example 1:***

Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:

Process	Burst Time
P_1	24
P_2	3
P_3	3

- If the processes arrive in the order **P1, P2, P3**, and are served in **FCFS order**, we get the result shown in the following **Gantt chart** (which is a bar chart that illustrates a particular schedule, including the start and finish times of each of the participating processes):

What will be the waiting time for process P1, P2, and P3?

- The waiting time is 0 milliseconds for process P1, 24 milliseconds for process P2, and 27 milliseconds for process P3.

- Thus, the average waiting time is (0 + 24 + 27)/3 = 17 milliseconds.

What will be the waiting time for process P1, P2, and P3 if the processes arrive in the order P2, P3, P1?

- If the processes arrive in the order P2, P3, P1, however, the results will be as shown in the following Gantt chart:

What will be the waiting time for process P1, P2, and P3?

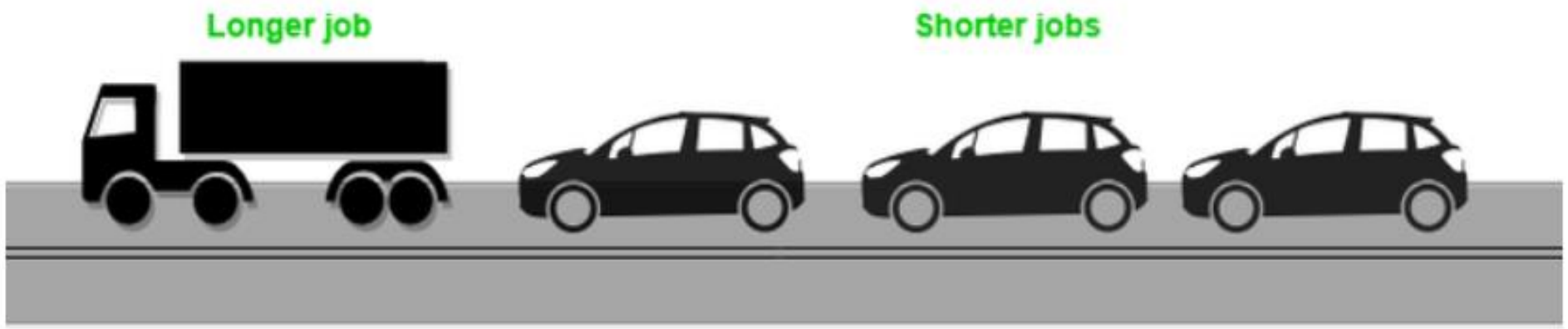
- Following are the waiting time for respective processes :
  - P1 = 6 ms
  - P2 = 0 ms
  - P3 = 3 ms

  - The average waiting time is now (6 + 0 + 3)/3 = 3 milliseconds

- **This reduction is substantial. Thus, the average waiting time under an FCFS policy is generally not minimal and may vary substantially if the processes CPU burst times vary greatly.**

First-Come, First-Served (FCFS) Scheduling (Contd.)

- If the processes with higher burst time arrived before the processes with smaller burst time, then, smaller processes have to wait for a long time for longer processes to release the CPU. This is called **Convoy Effect.**



Convoy Effect

First-Come, First-Served (FCFS) Scheduling (Contd.)

- Note also that the **FCFS scheduling algorithm is nonpreemptive**. Once the CPU has been allocated to a process, that process keeps the CPU until it releases the CPU, either by terminating or by requesting I/O.

- The FCFS algorithm is thus particularly troublesome for interactive systems, where it is important that each process get a share of the CPU at regular intervals.

- It would be disastrous to allow one process to keep the CPU for an extended period.
-

Question 1:

Considering the set of 4 processes whose arrival time and burst time are given below:

Process No.	Arrival Time	Burst Time
P1	0	2
P2	1	2
P3	5	3
P4	6	4

Calculate TAT, WT, RT, Average WT, and Average TAT.

Solution 1:

Process No.	Arrival Time	Burst Time (ms)
P1	0	2
P2	1	2
P3	5	3
P4	6	4

Gantt chart

Solution 1:

Gantt chart

Process No.	Arrival Time	Burst Time (ms)	Completion Time	TAT (ms)	WT (ms)	RT (ms)
P1	0	2	2	2	0	0
P2	1	2	4	3	1	1
P3	5	3	8	3	0	0
P4	6	4	12	6	2	2

Average TAT = (2+3+3+6)/4 = 3.5 ms

Average WT= (0+1+0+6)/4 = 0.75 ms

Question 2:

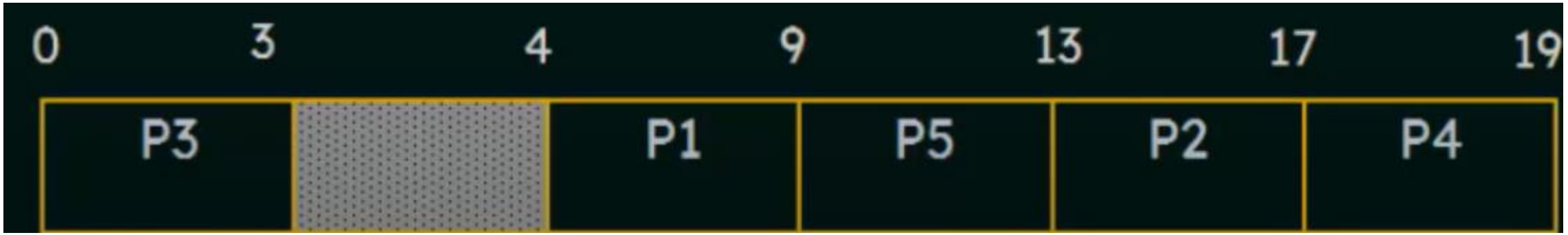
Considering the set of 5 processes whose arrival time and burst time are given below:

Process ID	Arrival Time	Burst Time
P1	4	5
P2	6	4
P3	0	3
P4	6	2
P5	5	4

Calculate TAT, WT, Average WT, and Average TAT.

Solution 2:

Gantt chart



Process ID	Completion Time	Turnaround Time	Waiting Time
P1	9	9 - 4 = 5	5 - 5 = 0
P2	17	17 - 6 = 11	11 - 4 = 7
P3	3	3 - 0 = 3	3 - 3 = 0
P4	19	19 - 6 = 13	13 - 2 = 11
P5	13	13 - 5 = 8	8 - 4 = 4

Solution 2:

Process ID	Completion Time	Turnaround Time	Waiting Time
P1	9	9 – 4 = 5	5 – 5 = 0
P2	17	17 – 6 = 11	11 – 4 = 7
P3	3	3 – 0 = 3	3 – 3 = 0
P4	19	19 – 6 = 13	13 – 2 = 11
P5	13	13 – 5 = 8	8 – 4 = 4

Average Turn Around time = (5 + 11 + 3 + 13 + 8) / 5
= 40 / 5
= 8 units

Average waiting time = (0 + 7 + 0 + 11 + 4) / 5
= 22 / 5
= 4.4 units

Shortest-Job-First Scheduling (with non-preemptive criteria)

- Till now, we were scheduling the processes according to their arrival time (in FCFS scheduling). However, SJF scheduling algorithm, schedules the processes according to their burst time.

- When the CPU is available, it is assigned to the process that has the smallest next CPU burst.

- If the next CPU bursts of two processes are the same, FCFS scheduling is used to break the tie.

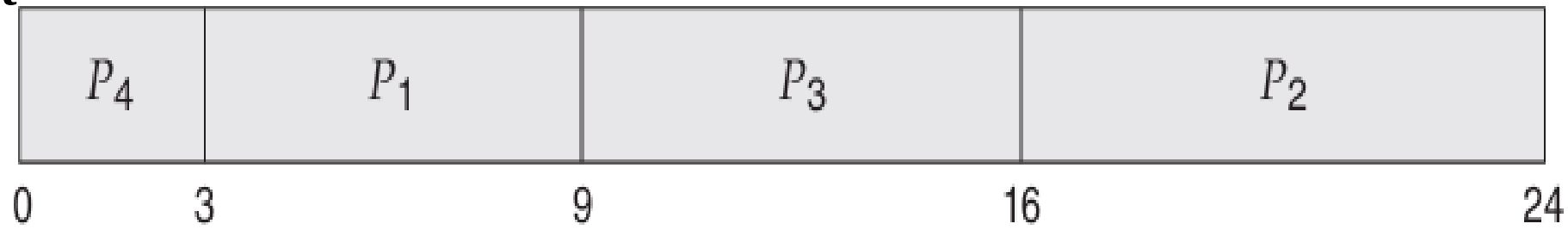
- ***Example 1***: Consider the following set of processes *(Assuming all the process has arrived at time 0)* , with the length of the CPU burst given in milliseconds:

Calculate the average waiting time?

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

Shortest-Job-First Scheduling (Contd.)

Gantt chart



- The waiting time is 3 milliseconds for process P1, 16 milliseconds for process P2, 9 milliseconds for process P3, and 0 milliseconds for process P4.

- The average waiting time is (3 + 16 + 9 + 0)/4 = 7 milliseconds.

- **What will be the average waiting time of the same scenario with FCFS algorithm?**

Shortest-Job-First Scheduling (Contd.)

- The waiting time is 3 milliseconds for process P1, 16 milliseconds for process P2, 9 milliseconds for process P3, and 0 milliseconds for process P4.

- The average waiting time is (3 + 16 + 9 + 0)/4 = 7 milliseconds.

- **What will be the average waiting time of the same scenario with FCFS algorithm?**
  - **Average waiting time = 10.25**

- The SJF scheduling algorithm is provably optimal, in that it gives the minimum average waiting time for a given set of processes.

Shortest-Job-First Scheduling (with preemptive criteria)

- The SJF algorithm can be either preemptive or non-preemptive. The choice arises when a new process arrives at the ready queue while a previous process is still executing.

- The next CPU burst of the newly arrived process may be shorter than what is left of the currently executing process.

- A preemptive SJF algorithm will preempt the currently executing process, whereas a non-preemptive SJF algorithm will allow the currently running process to finish its CPU burst.

- Preemptive SJF scheduling is sometimes called **Shortest-Remaining Time-First (SRTF) Scheduling.**

Shortest-Job-First Scheduling (with preemptive criteria)

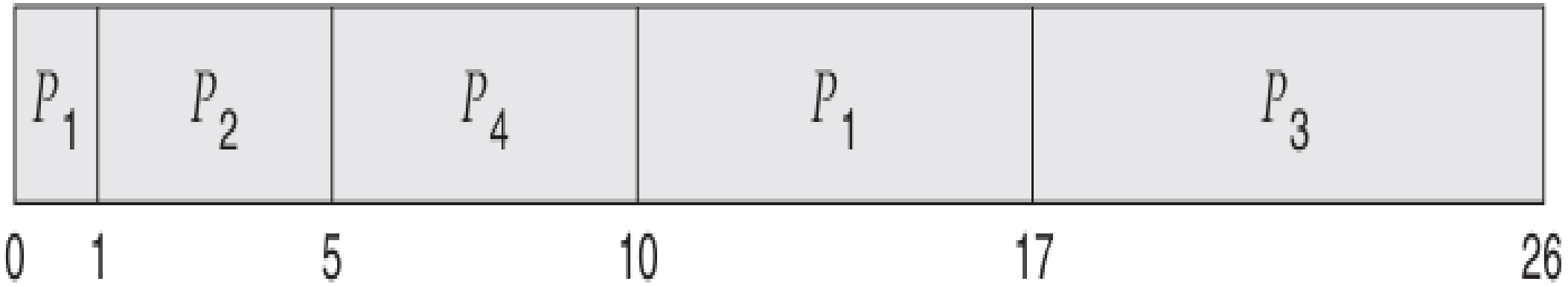
- ***Example 2***: consider the following four processes, with the length of the CPU burst given in milliseconds:

Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

Calculate the average waiting time?

Shortest-Job-First Scheduling (with preemptive criteria)

Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5



The average waiting time for this example is **6.5 milliseconds**.

Shortest-Job-First Scheduling (Contd.)

- Although the SJF algorithm is optimal, it cannot be implemented at the level of CPU scheduling, as there is no way to know the length of the next CPU burst.
- One approach to this problem is to try to approximate SJF scheduling. We may not know the length of the next CPU burst, but we may be able to predict its value.
- We expect that the next CPU burst will be similar in length to the previous ones. By computing an approximation of the length of the next CPU burst, we can pick the process with the shortest predicted CPU burst.
- The next CPU burst is generally predicted as an exponential average of the measured lengths of previous CPU bursts.

Question 1: (SJF with Non-Preemptive Criteria)

Considering the set of 4 processes whose arrival time and burst time are given below:

Process No.	Arrival Time	Burst Time (ms)
P1	1	3
P2	2	4
P3	1	2
P4	4	4

Calculate CT, TAT, WT, RT, Average WT, and Average TAT.

Solution 1:

Gantt chart

Process No.	Arrival Time	Burst Time (ms)	Completion Time	TAT (ms)	WT (ms)	RT (ms)
P1	1	3	6	5	2	2
P2	2	4	10	8	4	4
P3	1	2	3	2	0	0
P4	4	4	14	10	6	6

Average TAT = (5+8+2+10)/4 = 6.25 ms

Average WT= (2+4+0+6)/4 = 3 ms

Question 2: (SJF with Preemptive Criteria)

Considering the set of 4 processes whose arrival time and burst time are given below:

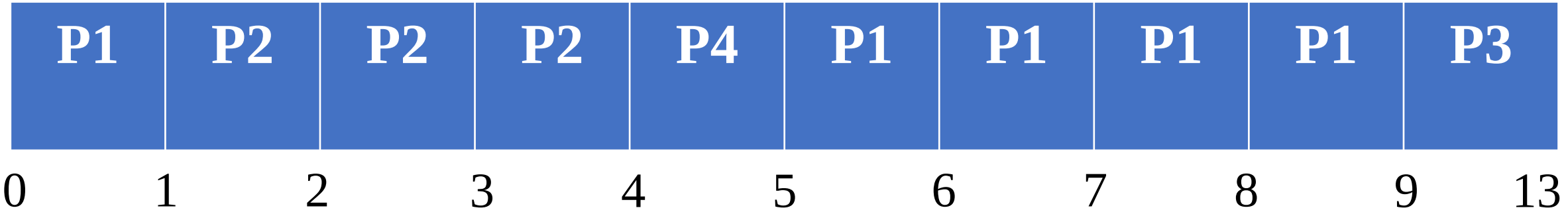
Process No.	Arrival Time	Burst Time (ms)
P1	0	5
P2	1	3
P3	2	4
P4	4	1

Calculate CT, TAT, WT, RT, Average WT, and Average TAT.

Process No.	Arrival Time	Burst Time (ms)
P1	0	5
P2	1	3
P3	2	4
P4	4	1

Gantt chart

(Detailed Elaboration of Gantt chart)



Gantt chart

Process No.	Arrival Time	Burst Time (ms)	Completion Time	TAT (ms)	WT (ms)	RT (ms)
P1	0	5	9	9	4	0
P2	1	3	4	3	0	0
P3	2	4	13	11	7	7
P4	4	1	5	1	0	0

Process No.	Arrival Time	Burst Time (ms)	Completion Time	TAT (ms)	WT (ms)	RT (ms)
P1	0	5	9	9	4	0
P2	1	3	4	3	0	0
P3	2	4	13	11	7	7
P4	4	1	5	1	0	0

Average TAT = 6 ms

Average WT= 2.75 ms

Average RT= 1.75 ms

Question 3:

An operating system uses SJSF scheduling algorithm for pre-emptive scheduling of processes. Considering the following set of processes whose arrival time and CPU burst times (in millisecond) are given below:

Process ID	Arrival Time	Burst Time
P1	0	12
P2	2	4
P3	3	6
P4	8	5

Calculate Average WT.

Process ID	Arrival Time	Burst Time
P1	0	12
P2	2	4
P3	3	6
P4	8	5

Solution:

Gantt Chart:

Average Waiting Time

= (15 + 0 + 3 + 4)/4 = 5.5 ms

Question 4:

An operating system uses SJSF scheduling algorithm for pre-emptive scheduling of processes. Considering the following set of processes whose arrival time and CPU burst times (in millisecond) are given below:

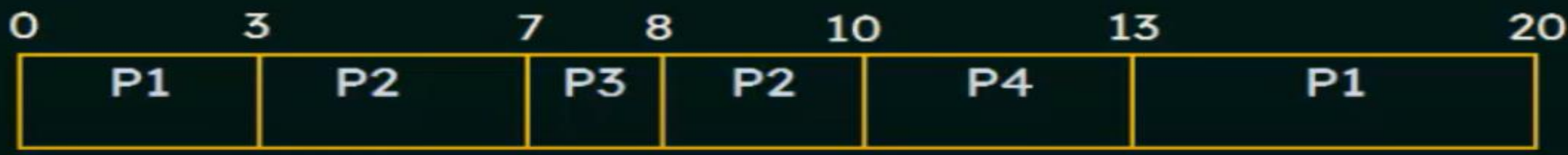
Process ID	Arrival Time	Burst Time
P1	0	10
P2	3	6
P3	7	1
P4	8	3

Calculate Average TAT.

Process ID	Arrival Time	Burst Time
P1	0	10
P2	3	6
P3	7	1
P4	8	3

Solution:

Gantt Chart:



Average Turnaround Time

= (20 + 7 + 1 + 5)/4

= 33/4

= 8.25 ms

References

1. Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, “Operating System Concepts,” Eleventh Edition (Willey).

2. Andrew S. Tanenbaum, “Modern Operating Systems”, Fourth Edition (Pearson Publications), 2014.

3. https://www.geeksforgeeks.org/.

4. https://www.javatpoint.com/.

5. https://www.tutorialspoint.com/.

6. https://www.nesoacademy.org/.