Operating System

Unit – 2
(Part-B)
Process

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

Scheduling Algorithms

Round-Robin Scheduling

- The round-robin (RR) scheduling algorithm is similar to FCFS scheduling, but preemption is added to enable the system to switch between processes.

- A small unit of time, called a **time quantum** or **time slice**, is defined. A time quantum is generally from 10 to 100 milliseconds in length.

- The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum.

- To implement RR scheduling, we again treat the ready queue as a FIFO queue of processes. New processes are added to the tail of the ready queue.

- The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum, and dispatches the process.

Round-Robin Scheduling (Contd.)

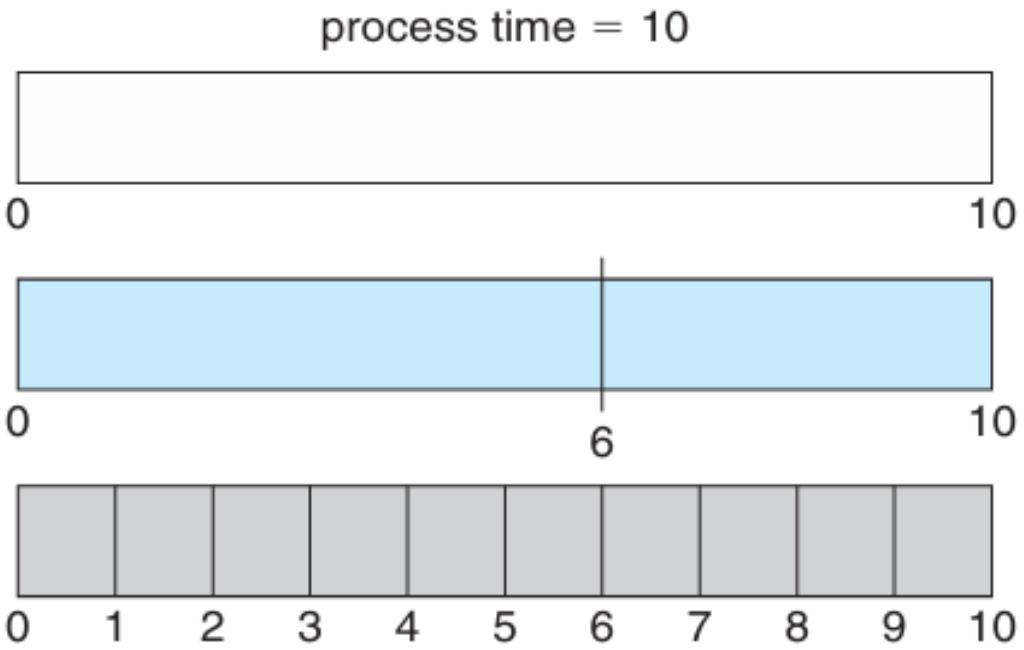
- One of two things will then happen. The process may have a CPU burst of less than 1 time quantum. In this case, the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue. If the CPU burst of the currently running process is longer than 1 time quantum, the timer will go off and will cause an interrupt to the operating system.

- A **context switch** will be executed, and the process will be put at the tail of the ready queue. The CPU scheduler will then select the next process in the ready queue.

- The performance of the RR algorithm depends heavily on the size of the time quantum. At one extreme, if the time quantum is extremely large, the RR policy is the same as the FCFS policy.

- In contrast, if the time quantum is extremely small (say, 1 millisecond), the RR approach can result in a large number of context switches.

- Assume, for example, that we have only one process of 10 time units. If the quantum is 12 time units, the process finishes in less than 1 time quantum, with no overhead. If the quantum is 6 time units, however, the process requires 2 quanta, resulting in a context switch. If the time quantum is 1 time unit, then nine context switches will occur, slowing the execution of the process accordingly



quantum	context switches
12	0
6	1
1	9

Question 1:

An operating system uses RR scheduling algorithm for pre-emptive scheduling of processes with time quantum = 2 units. Considering the set of 4 processes whose arrival time and burst time are given below:

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

Calculate TAT, WT, RT, and Number of time context switching happens.

Solution 1:

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

Ready Queue

P1	P2	P3	P1	P4	P2	P1

Gantt chart

Gantt chart

Process No.	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
P1	0	5	12	12	7	0
P2	1	4	11	10	6	1
P3	2	2	6	4	2	2
P4	4	1	9	5	4	4

Number of time context switching happens = ?

Gantt chart

Process No.	Arrival Time	Burst Time	Completion Time	TAT	WT	RT
P1	0	5	12	12	7	0
P2	1	4	11	10	6	1
P3	2	2	6	4	2	2
P4	4	1	9	5	4	4

Number of time context switching happens = 6

Question 2:

An operating system uses RR scheduling algorithm for pre-emptive scheduling of processes with time quantum = 4 units. Considering the set of 4 processes whose arrival time and burst time are given below:

Process No.	Arrival Time	Burst Time
P1	0	8
P2	1	4
P3	2	9
P4	3	5

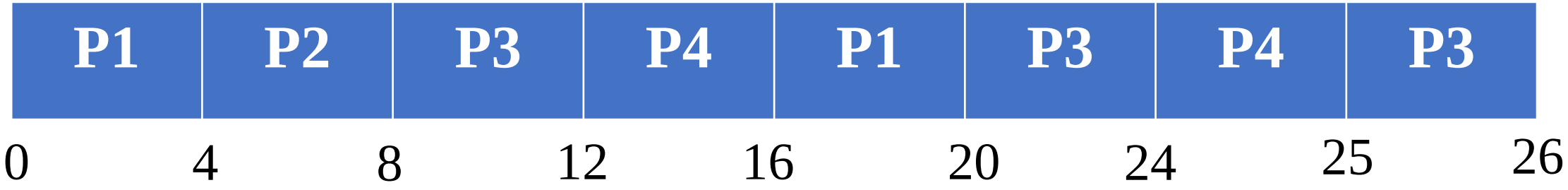
Calculate TAT, Average TAT, WT, and Average WT.

Solution 2:

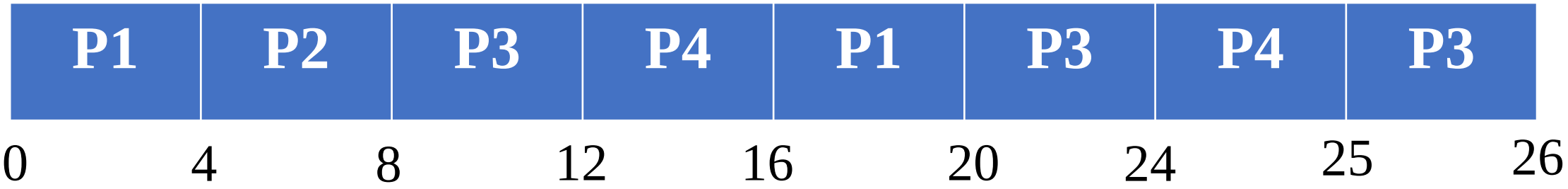
Process No.	Arrival Time	Burst Time (ms)
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Ready Queue

Gantt chart



Gantt chart



Process No.	Arrival Time	Burst Time	Completion Time	TAT	WT
P1	0	8	20	20	12
P2	1	4	8	7	3
P3	2	9	26	24	15
P4	3	5	25	22	17

Gantt chart

Process No.	Arrival Time	Burst Time	Completion Time	TAT	WT
P1	0	8	20	20	12
P2	1	4	8	7	3
P3	2	9	26	24	15
P4	3	5	25	22	17

Average TAT = 18.25 units

Average WT = 11.75 units

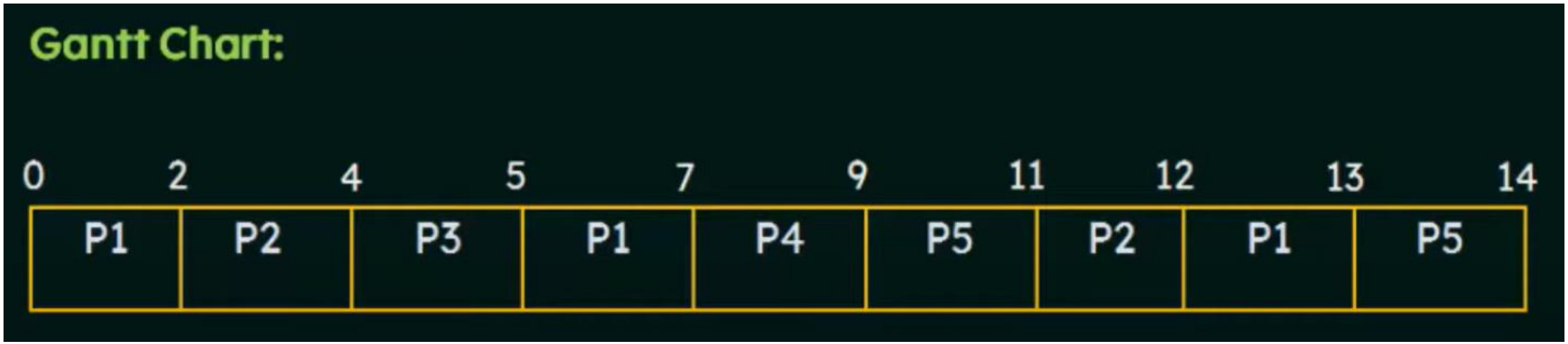
Question 3:

An operating system uses RR scheduling algorithm for pre-emptive scheduling of processes with time quantum = 2 units. Considering the set of 5 processes whose arrival time and burst time are given below:

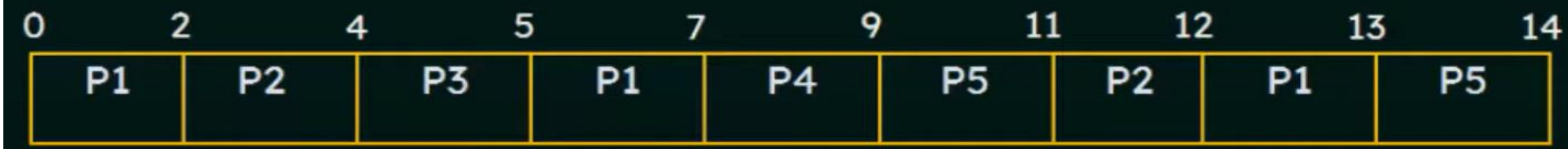
Process ID	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	1
P4	3	2
P5	4	3

Calculate Average TAT and Average WT.

Gantt Chart:



Gantt Chart:



Process ID	Completion Time	Turnaround Time	Waiting Time
P1	13	13 – 0 = 13	13 – 5 = 8
P2	12	12 – 1 = 11	11 – 3 = 8
P3	5	5 – 2 = 3	3 – 1 = 2
P4	9	9 – 3 = 6	6 – 2 = 4
P5	14	14 – 4 = 10	10 – 3 = 7

Process ID	Completion Time	Turnaround Time	Waiting Time
P1	13	13 – 0 = 13	13 – 5 = 8
P2	12	12 – 1 = 11	11 – 3 = 8
P3	5	5 – 2 = 3	3 – 1 = 2
P4	9	9 – 3 = 6	6 – 2 = 4
P5	14	14 – 4 = 10	10 – 3 = 7

Average Turn Around time

= (13 + 11 + 3 + 6 + 10) / 5

= 43 / 5 = **8.6 units**

Average waiting time

= (8 + 8 + 2 + 4 + 7) / 5

= 29 / 5 = **5.8 units**

Question 4:

An operating system uses RR scheduling algorithm for pre-emptive scheduling of processes with time quantum = 5 units. Considering the set of 6 processes whose arrival time and burst time are given below:

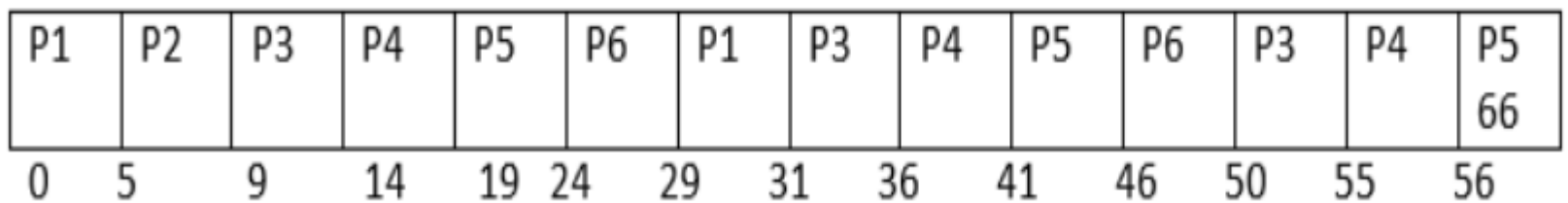
Process No.	Arrival Time	Burst Time
P1	0	7
P2	1	4
P3	2	15
P4	3	11
P5	4	20
P6	4	9

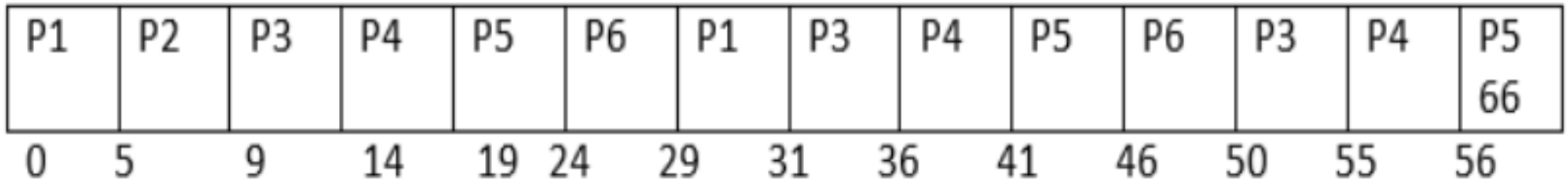
Calculate Average TAT and Average WT.

Ready Queue:

> P1, P2, P3, P4, P5, P6, P1, P3, P4, P5, P6, P3, P4, P5

Gantt chart:





Process ID	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time
P1	0	7	31	31	24
P2	1	4	9	8	4
P3	2	15	55	53	38
P4	3	11	56	53	42
P5	4	20	66	62	42
P6	4	9	50	46	37

Average Waiting Time

> Average Waiting Time = (5 + 26 + 5 + 42 + 42 + 37) / 6
> Average Waiting Time = 157 / 6
> Average Waiting Time = 26.16667

Average Turn Around Time

> Average Turn Around Time = (31 + 8 + 53 + 53 + 62 + 46) / 6
> Average Turn Around Time = 253 / 6
> Average Turn Around Time = 42.16667

Scheduling Algorithms

Priority Scheduling

- The SJF algorithm is a special case of the general priority-scheduling algorithm.
- A priority is associated with each process, and the CPU is allocated to the process with the highest priority. **Equal-priority processes are scheduled in FCFS order**.
- An SJF algorithm is simply a priority algorithm where the priority (p) is the inverse of the (predicted) next CPU burst. The larger the CPU burst, the lower the priority, and vice versa.
- Note that we discuss scheduling in terms of high priority and low priority. Priorities are generally indicated by some fixed range of numbers, such as 0 to 7 or 0 to 4,095.
- However, there is no general agreement on whether 0 is the highest or lowest priority. Some systems use low numbers to represent low priority; others use low numbers for high priority.

Scheduling Algorithms

Priority Scheduling (Contd.)

- Priority scheduling can be either **preemptive** or **non-preemptive**.

- When a process arrives at the ready queue, its priority is compared with the priority of the currently running process. A preemptive priority scheduling algorithm will preempt the CPU if the priority of the newly arrived process is higher than the priority of the currently running process.

- A non preemptive priority scheduling algorithm will simply put the new process at the head of the ready queue.

- A major problem with priority scheduling algorithms is **indefinite blocking**, or **starvation**.

- A process that is ready to run but waiting for the CPU can be considered blocked. A priority scheduling algorithm can leave some low priority processes waiting indefinitely.

Scheduling Algorithms

Priority Scheduling (Contd.)

- In a heavily loaded computer system, a steady stream of higher-priority processes can prevent a low-priority process from ever getting the CPU.

- A solution to the problem of indefinite blockage of low-priority processes is **aging**.

- **Aging involves gradually increasing the priority of processes that wait in the system for a long time.**
- For example, if priorities range from 127 (low) to 0 (high), we could periodically (say, every second) increase the priority of a waiting process by 1. Eventually, even a process with an initial priority of 127 would have the highest priority in the system and would be executed. In fact, it would take a little over 2 minutes for a priority-127 process to age to a priority-0 process.

Priority Scheduling (With non-preemptive)

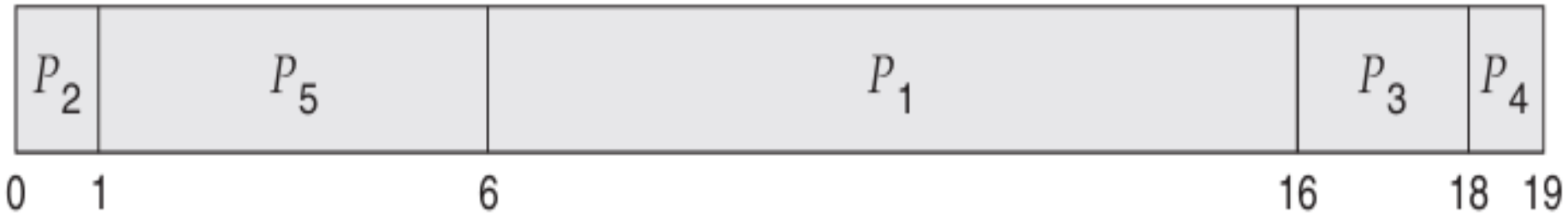
- ***Example 1***: Consider the following set of processes, assumed to have arrived at time 0 in the order P1, P2, ⋯, P5, with the length of the CPU burst given in milliseconds:

(Priority: Lower the priority value, higher is the priority)

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

Calculate the average waiting time?

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2



Average waiting time = 8.2 milliseconds

Question 1:

An operating system uses Priority scheduling algorithm for pre-emptive scheduling of processes. Considering the set of 4 processes whose arrival time, burst time, and priority value are given below: (Priority- Higher the priority value, higher the priority)

Process No.	Arrival Time	Burst Time	Priority
P1	0	5	10
P2	1	4	20
P3	2	2	30
P4	4	1	40

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

Solution 1:

Process No.	Arrival Time	Burst Time	Priority
P1	0	5	10
P2	1	4	20
P3	2	2	30
P4	4	1	40

Gantt chart

Gantt chart

Process No.	Arrival Time	Burst Time	Priority	Completion Time	TAT	WT
P1	0	5	10	12	12	7
P2	1	4	20	8	7	3
P3	2	2	30	4	2	0
P4	4	1	40	5	1	0

Process No.	Arrival Time	Burst Time	Priority	Completion Time	TAT	WT
P1	0	5	10	12	12	7
P2	1	4	20	8	7	3
P3	2	2	30	4	2	0
P4	4	1	40	5	1	0

Average TAT = 5.5 units

Average WT = 2.5 units

Question 2:

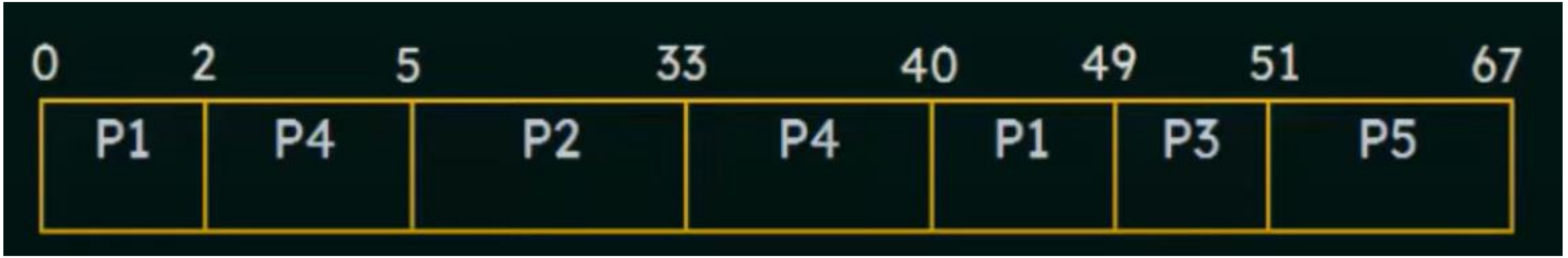
Considering the set of 5 processes whose arrival time (in millisecond), and burst time (in millisecond), and priority (0 is the highest priority) are given below:

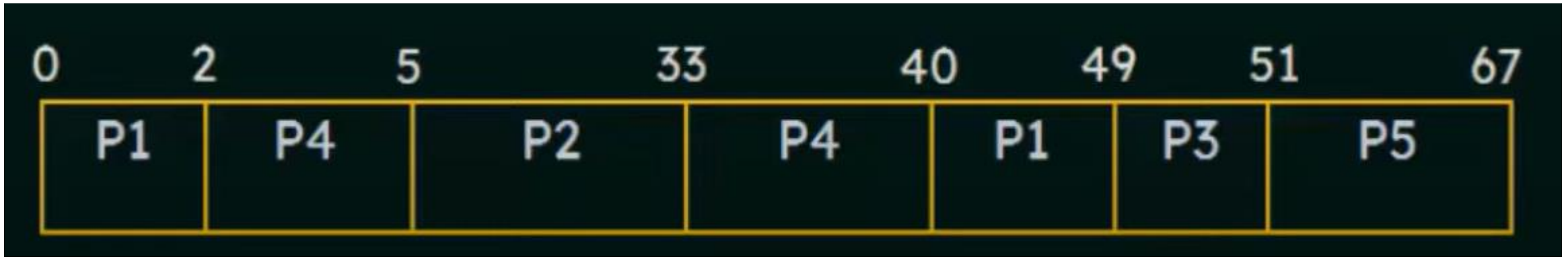
Process ID	Arrival Time	Burst Time	Priority
P1	0	11	2
P2	5	28	0
P3	12	2	3
P4	2	10	1
P5	9	16	4

The average waiting time (in millisecond) of all the processes using **preemptive priority scheduling algorithm** is _____.

Process ID	Arrival Time	Burst Time	Priority
P1	0	11	2
P2	5	28	0
P3	12	2	3
P4	2	10	1
P5	9	16	4

Gantt chart





Average Waiting Time

= (38 + 0 + 37 + 28 + 42) / 5

= 145/5 ms

= 29 ms

Question 3:

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

Process Id	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

If the CPU scheduling policy is priority preemptive, calculate the average waiting time and average turn around time. (Higher number represents higher priority)

Process Id	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

Gantt Chart

Process Id	Exit time	Turn Around time	Waiting time
P1	15	15 – 0 = 15	15 – 4 = 11
P2	12	12 – 1 = 11	11 – 3 = 8
P3	3	3 – 2 = 1	1 – 1 = 0
P4	8	8 – 3 = 5	5 – 5 = 0
P5	10	10 – 4 = 6	6 – 2 = 4

- Average Turn Around time = (15 + 11 + 1 + 5 + 6) / 5 = 38 / 5 = 7.6 unit
- Average waiting time = (11 + 8 + 0 + 0 + 4) / 5 = 23 / 5 = 4.6 unit

Question 4:

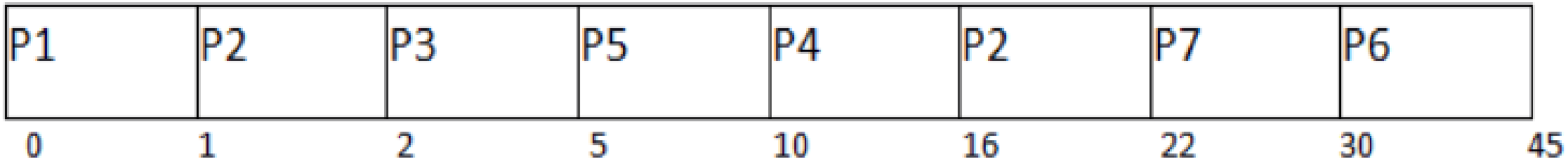
There are 7 processes P1, P2, P3, P4, P5, P6 and P7 given. Their respective priorities, Arrival Times and Burst times are given in the table below.

Process Id	Priority	Arrival Time	Burst Time
1	2(L)	0	1
2	6	1	7
3	3	2	3
4	5	3	6
5	4	4	5
6	10(H)	5	15
7	9	15	8

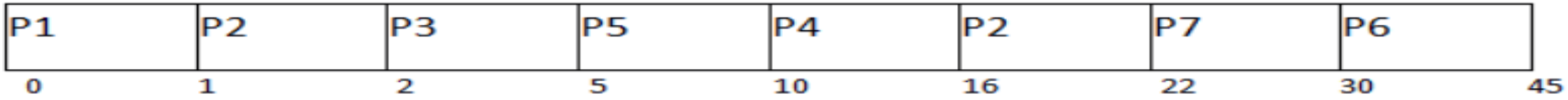
If the CPU scheduling policy is priority preemptive, calculate the average waiting time. (Higher number represents higher priority)

Process Id	Priority	Arrival Time	Burst Time
1	2(L)	0	1
2	6	1	7
3	3	2	3
4	5	3	6
5	4	4	5
6	10(H)	5	15
7	9	15	8

Gantt chart



Gantt chart



Process Id	Priority	Arrival Time	Burst Time	Completion Time	Turn around Time	Waiting Time
1	2	0	1	1	1	0
2	6	1	7	22	21	14
3	3	2	3	5	3	0
4	5	3	6	16	13	7
5	4	4	5	10	6	1
6	10	5	15	45	40	25
7	9	6	8	30	24	16

Avg Waiting Time = (0+14+0+7+1+25+16)/7 = 63/7 = 9 units

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/.