(Operating Systems - 22516)

Unit IV : CPU Scheduling Algorithm

(Weightage - 14 marks)

— Pranjal Save

CPU Scheduling

In multiprogramming system, the Operating Systems schedules the processes on the CPU to have maximum utilization of it and procedure is called CPU scheduling.

The Operating Systems uses various scheduling algorithm to schedule the processes.

Need Scheduling

1) Balanced Process mix: The scheduler selects a mix of CPU-bound (need more CPU) and I/O bound (need more I/O) processes to keep the CPU and I/O devices active.
2) Avoid Deadlocks: The OS ensures process don't get stuck waiting for resources by carefully managing resources allocation.
3) Dynamic ~~Adjustments~~ Adjustments: The OS changes priorites based on the current state of system to avoid overloading one resource (CPU).
4) Fairness: Processes are scheduled fairly so no process is stuck waiting forever, preventing deadlocks & starvation.
5) Resource Optimization: The OS works to make the best use of CPU, memory and I/O ensuring all resources are efficiently utilized.

Define CPU and I/O burst cycles (2m)

or

Describe I/O burst and CPU burst cycle with neat Diagram (4m)

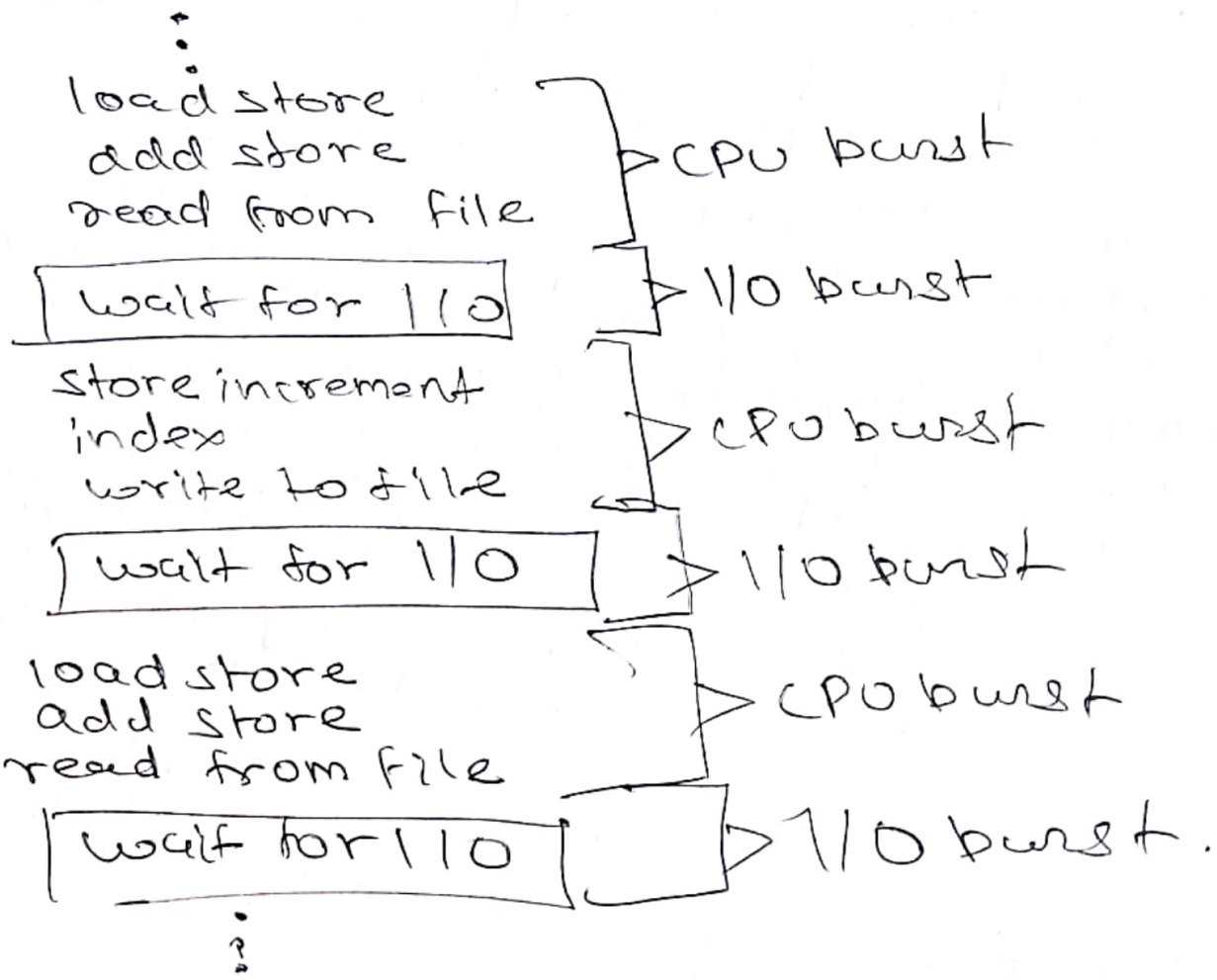
⇒ CPU burst cycle

It is time period when process is busy with CPU.

I/O burst cycle

It is time period when process is busy with I/O resources.

Diagram



iprocess runs in cycle, switching between CPU work & I/O operations.

- It starts with CPU work, where it uses the CPU to process tasks.
- Then it switches to I/O work, where it waits for input/output tasks to finish.
- The process keeps switching between CPU work and I/O work until it completes
- The final CPU work ends when process request to terminates.
- Example, printing.

Question: Define CPU bound program I/O bound program (2m)

CPU bound program

If execution of a program is highly dependent on CPU then it is known as CPU bound program.

or

If a task does a lot operations using CPU it called CPU-bound task.

I/O Bound program

If execution of program is dependent on input-output system and its resources such disk drives and peripheral devices then it is known as I/O bound program.

VIMP Question : Explain Scheduling (any four).

1) CPU utilization

In multiprogramming the main objective is keep CPU as busy as possible.

CPU utilization can range from 0 to 100 percent.

2) Throughput

It is number of processes that completed per unit time.

It measures of work done in the system.

Throughput depends on execution time required for any processes.

Simple words, its performance.

3) Turnaround time

Turnaround time is total time taken from the submission of process to its completion.

It includes all waiting, processing & I/O times.

Turnaround time = Completion Time − Arrived Time

4) Waiting Time

waiting time is total process spends in ready queue for CPU allocate.

It does not include the time spent executing or performing I/O operations.

waiting time = Turnaround Time − Burst Time

5) Response Time

It is also known as completion time. It is time between making a request & getting first reply. It doesn't mean that process is finished, but that it has started responding. Some process ...

Preemptive Scheduling

The CPU can be taken away from a running processes.

- Allows higher priority processes to interrupted lower priority ones.
- Mostly preemptive scheduling used in Real time system where quick response is needed.
- Examples. Round Robin
  Shortest Remaining Time First (SRTF)

Non-Preemptive Scheduling

- Once the process starts, it cannot be interrupted until it finishes or voluntarily gives up to CPU.
- Algoriths are - SJF (Shortest Job First)
  FCFS (First come First serve)
  Priority scheduling.

Question: How premptive scheduling is better than non-premptive scheduling.

⇒ 1) Pre-emptive scheduling allows shorter or more urgent process to interrupt long-running ones, leads to improving Response Time.

2) It reduces the time processes spend waiting in Queue by priority shortest & highest.

3) It also make CPU utilization effectently, Better Real time systems,

4) Prevents Process monopolization.

Question: Define Deadlock (2m)

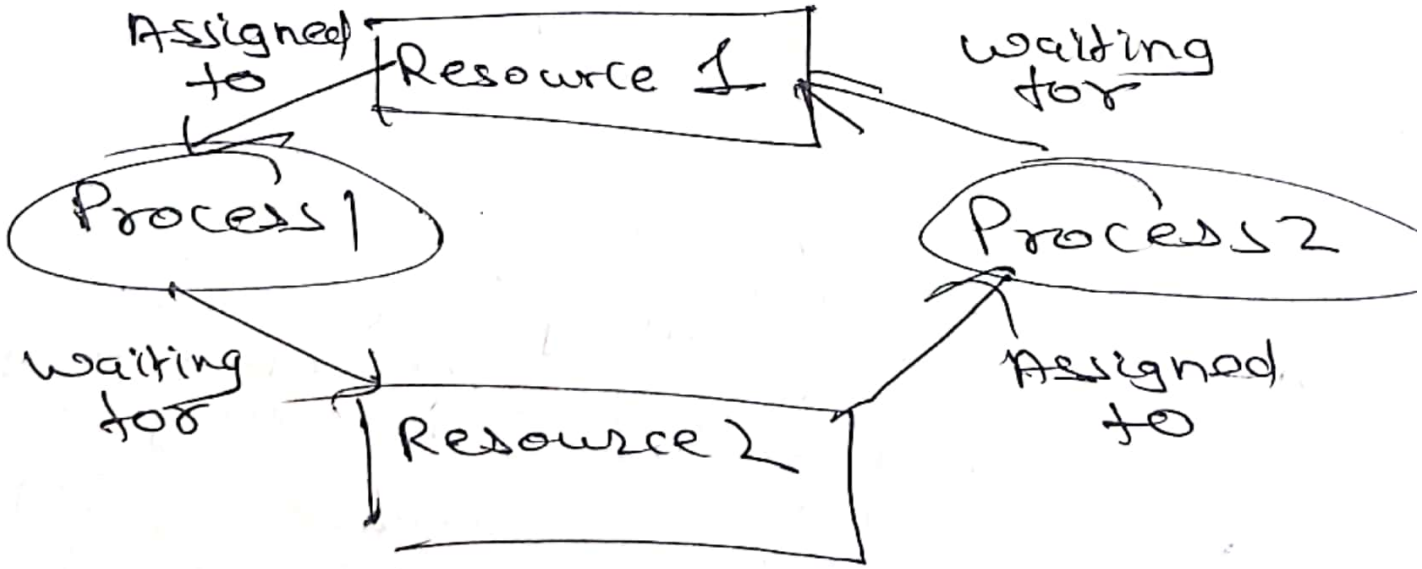
A Deadlock is situation in operating system where two or more processes are unable to proceed because each one is waiting for resource that other holds.

A situation where a set of processes are blocked because each process is holding a resource and waiting for anothe resource that is held by another process.

This Result in all involved processes stuck in infinite waiting state.

Example

Process P_1 holds Resource R_1
Process P_2 holds Resource R_2
Process P_1 waiting for Resource R_2
& Process P_2 waiting for Resource R_1



What are necessary conditions of deadlock.

A deadlock can only occur if the following conditions met simultaneously –

1. Mutual Exclusion.

⇒ 1) At least one resource must be held in non-sharable mode.
2) Only one process can use the resource at a time
3) If another process requests the resource, it must wait until resources is released.

2. Hold and wait.

⇒ 1) A process is currently holding one or more resources and is waiting for Additional resources that are held by other processes.
2) Process does not release its current resources while waiting.

3. No Pre-emption.

⇒ 1) Resources cannot be forcibly taken from a process
2) Resources must only be released voluntarily by process holding them.

4. Circular wait

This means that there is a circular chain of processes, where each process is waiting for resources that next process chain holds.

P_0 is waiting for resources hold by P_1.
P_1 is waiting for resources hold by P_2
P_2 is waiting for resources hold by P_3
P_3 is waiting for resources hold by P_1.

Deadlock occurs when all four necessary conditions (Mutual Exclusion, Hold and wait, No Pre-emption and circular wait) happen together.

To prevent deadlock, we can break any one of these condition.

1. Eliminate Mutual Exclusion.
   1) Mutual Exclusion means only one process can use resource at a time.
   2) Make resources shareable where possible, so multiple processes can use them simultaneously.
   3) Example, Read-only files can be accessed by many processes at same time.

2. Eliminate Hold and wait.
   1) This occurs when a process hold some resources and waits for others.
   2) Ensure processes request all resources at the start. Alternatively, release held resources before requesting new ones.
   3) Need scanner & printer both.

3. Eliminate No-Preemption
   1) A process can be forced to release resource using it.
   2) Allow system to preempt resources from processes if necessary.
   3) Priority-wise (Example).

4. Eliminate circular wait.

Assign priority numbers to resources. Processes can only request resources in increasing order of priority.
Example P1 has priority 1, & P2 has priority 2. P1 holding & P2 can't request P1.

Steps required for Banker's algorithm to avoid Deadlock.

The Banker's Algorithm is a resource allocation and deadlock avoidance algorithm.

It is designed to simulate allocation of resources to multiple processes in way that ensures the system can avoid deadlocks.

The algorithm is named so because it analgous to banking system where a bank never allocates its available cash in such a way that it cannot satisfy needs of all its clients.

Steps

1) Calculate Need matrix.

Sub - Allocated resources from maximum requriement for process.

2) Check Safe Sequence

- Start with available Resources.
- Look for process whose Need is less than equal to Available resources.
- If such a process is found.
  - Assume it runs and releases its allocated resources
  - Add these resources to Available resources
  - mark process as finished.
- Repeat this process for all process

3) Safe State: If all process can finish in some order, the system is in safe state. If not, unsafe state, deadlock can occur.

Given

Available resources (3,3,2)

Process	Maximum	Allocation	(M-A) Need
P_0	(7,5,3)	(0,1,0)	(7,4,3)
P_1	(3,2,2)	(2,0,0)	(1,2,2)
P_2	(9,0,2)	(3,0,2)	(6,0,0)

Calculate Need matrix

Safe sequence

P_1's Need : = (1,2,2) is $\leq$ Available (3,3,2)

. P_1 can finish an release resources.

Now Available = (3,3,2) + (2,0,0) = (5,3,2)

P_0's Need : (7,4,3) is $\leq$ (3,3,2)

It cant be executed.

P_2 is Need = (6,0,0) is $\leq$ (5,3,2)

P_2 can finish the resources

(5,3,2) + (3,0,2) = (8,3,4)

P_0 is Need = (7,4,3) $\leq$ (8,3,4)

can finish

(8,3,4) + (0,1,0) = (8,4,4)

P_1's Need : (1,2,2) is $\leq$ Available (8,4,4)

P_1 can finish

Safe Sequence

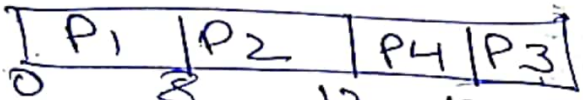
$P_1 \rightarrow P_2 \rightarrow P_0$

Question

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

SJF non-premptive

⇒ GANNT chart



Completion time

$P_1 = 8$
$P_2 = 12$
$P_4 = 17$
$P_3 = 26$

Turnaround time

$P_1 = 8 - 0 = 8$
$P_2 = 12 - 1 = 11$
$P_3 = 17 - 2 = 15$
$P_4 = 26 - 3 = 23$

$Avg = \frac{8+11+15+23}{4}$

$= 14.25$ ms

Waiting Time

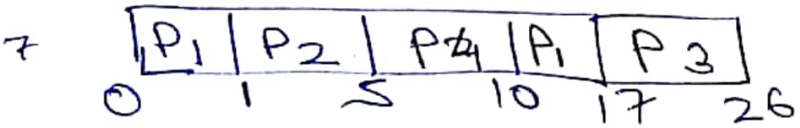
$P_1 = 8 - 8 = 0$
$P_2 = 11 - 4 = 7$
$P_3 = 15 - 9 = 6$
$P_4 = 23 - 5 = 18$

$Avg = \frac{0+7+6+18}{4}$

$= 7.75$ ms

SJF premptive

Gantt chart



Completion time

$P_1 = 17$ $P_3 = 26$
$P_2 = 5$ $P_4 = 10$

T.A.T

$P_1 = 17 - 0 = 17$
$P_2 = 5 - 1 = 4$
$P_3 = 26 - 2 = 24$
$P_4 = 10 - 3 = 7$

Avg 13 ms

W.T

$P_1 = 17 - 8 = 9$
$P_2 = 4 - 4 = 0$
$P_3 = 24 - 9 = 15$
$P_4 = 7 - 5 = 2$

$= 6.5$ ms

Types of Scheduling Algorithms

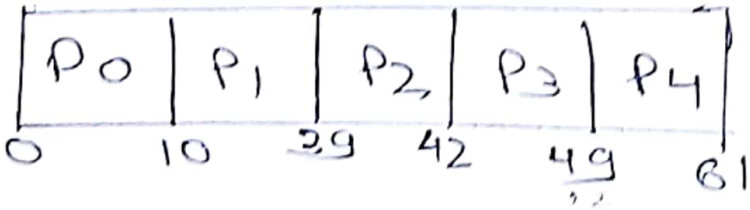
1) First-come, First served (FCFS)
2) Shortest Job First (SJF)
3) Priority scheduling.
4) Round Robin (RR)
5) Multilevel Queue scheduling.

First Come First Serve:

Q// Calculate Average waiting time & turn around time.

Process	Arrival time	Burst time
P_0	0	10
P_1	1	29
P_2	2	3
P_3	3	7
P_4	4	12

⇒ GANTT chart



Completion Time

$P_0 = 10$
$P_1 = 39$
$P_2 = 42$
$P_3 = 49$
$P_4 = 61$

Turnaround time

(C.T - A.T)

Completion time - Arrival time

$P_0 = 10 - 0 = 10$
$P_1 = 39 - 1 = 38$
$P_2 = 42 - 2 = 40$
$P_3 = 49 - 3 = 46$
$P_4 = 61 - 4 = 57$

Waiting Time

(T.A.T − B.T)

Turnaround time − Burst Time

$P_0 = 10 - 10 = 0$
$P_1 = 38 - 29 = 9$
$P_2 = 40 - 3 = 37$
$P_3 = 46 - 7 = 39$
$P_4 = 57 - 12 = 45$

Average Turnaround time

$$\frac{P_0 + P_1 + P_2 + P_3 + P_4}{5} = \frac{10 + 38 + 40 + 46 + 57}{5}$$

$= 38.2$ ms/unit

Avg waiting time

$$\frac{0 + 9 + 37 + 39 + 45}{5}$$

$= 26$ ms

Question

Process	A.T	B.T
P_1	0	7
P_2	1	4
P_3	2	10
P_4	3	6
P_5	4	8

i) SJF
ii) FCFS

⇒ 1) FCFS

Gantt chart

P_1	P_2	P_3	P_4	P_5
0	7	11	21	27	35

C.T
$P_1 = 7$
$P_2 = 11$
$P_3 = 21$
$P_4 = 27$
$P_5 = 35$

Turnaround time

$P_1 = 7 - 0 = 7$
$P_2 = 11 - 1 = 10$
$P_3 = 21 - 2 = 19$
$P_4 = 27 - 3 = 24$
$P_5 = 35 - 4 = 31$

Avg = 18.2 ms

Waiting time

$P_1 = 7 - 7 = 0$
$P_2 = 10 - 4 = 6$
$P_3 = 19 - 10 = 9$
$P_4 = 24 - 6 = 18$
$P_5 = 31 - 8 = 23$

Avg = 11.2 ms

2) SJF

Non-preemptive

GANTT chart

P_1	P_2	P_4	P_5	P_3
0	7	11	17	25	35

C.T

Turnaround time

$P_1 = 7 - 0 = 7$
$P_2 = 11 - 1 = 10$
$P_3 = 35 - 2 = 33$
$P_4 = 17 - 3 = 14$
$P_5 = 25 - 4 = 21$

Avg 17 ms

Waiting time

$P_1 = 7 - 7 = 0$
$P_2 = 10 - 4 = 6$
$P_3 = 33 - 10 = 23$
$P_4 = 14 - 6 = 8$
$P_5 = 21 - 8 = 13$

10 ms

Premptive

GC | P1 | P2 | P5 | P4 | P5 | P3 |
0 1 5 11 17 25 35

TAT
P1 = 11 - 0 = 11
P2 = 5 - 1 = 4
P3 = 35 - 2 = 33
P4 = 17 - 3 = 14
P5 = 25 - 4 = 21

Avg TAT 16.6 ms

WT
P1 = 11 - 7 = 4
P2 = 4 - 4 = 0
P3 = 33 - 10 = 23
P4 = 14 - 6 = 8
P5 = 21 - 8 = 13

Avg 9.6 ms

Question: Explain Round Robin algorithm with suitable example.

⇒ Each process is assigned a small, fixed amount of CPU time called time Quantum or time slice.

The ready processes are stored in circular Queue, and CPU goes through one by one.

If a process doesn't finish in its time quantum, it gets preempted (stopped) and placed at the back of Queue to wait for another turn.

Every progess gets an equal chance to use CPU, making it fair and responsive.

It follows concept of context switching.

Process	A-T	B-T
P11	0	8
P12	1	4
P13	2	9
P14	3	5

(Time Quantum = 3ms)

| P11 | P12 | P13 | P14 |
0 3 6 9 12

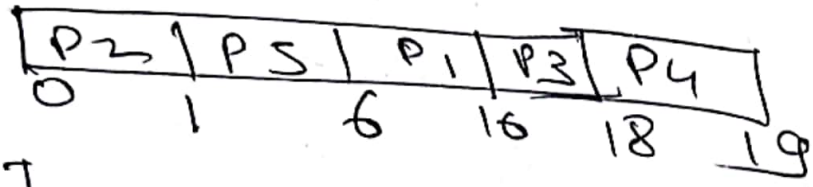
| P11 | P12 | P13 | P14 | P11 |
12 15 18 19 21 23

P11 = 23 P13 = 26
P12 = 16 P14 = 22

P13 26

riority scheduling

Process	B.T	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2



C.T

P1=16
P2=1
P3=18
P4=19
P5=6

W.T

P1=16-10=6
P2=1-1=0
P3=18-2=16
P4=19-1=18
P5=6-5=1

Avg=8.2 ms

Question: Explain multilevel Queue scheduling with suitable Example. (4m).

⇒ multilevel Queue scheduling in CPU scheduling method where processes are grouped into seperate queues based on their priority.

Each Queue uses it own scheduling method

- Foreground Queue: Uses Round Robin
- Background Queue: Uses FCFS

Higher priority Queues run before lower ones.

Process stay in one queue and do not move between them.

b) Lower priority processes may face starvation if higher priority queues are always busy.

Example

- Queue 1 (High Priority): For system tasks, uses Round Robin.
- Queue 2 (Medium Priority): For system tasks, uses Round Robin
- Queue 3 (Low Priority): For batch tasks, uses FCFS.

If batch process is running in queue 3 and system process arrives in queue 1 the system process will prempt batch process.

State two features of Non-premptive Scheduling

⇒ - No interuption
- Less computation resources are used
- It is very easy procedure.
- It has minimal scheduling Burden
- It has high throughput rate
