# OS NUMERICALS

# BASIC TERMINOLOGIES

**AT (Arrival Time)**:

- The time at which a particular process arrives in the ready queue.
- It indicates when a process is ready to be executed by the CPU.

**BT (Burst Time)**:

- The total time required by a process for its execution on the CPU.
- It represents the amount of time a process needs to run in the CPU without interruption.

**TAT (Turnaround Time)**:

- The total time taken by a process from its arrival to its completion.
- Formula:
  **TAT = Completion Time - Arrival Time**
- It includes both the waiting time and the execution time (burst time) of the process.

# BASIC TERMINOLOGIES

**Completion time (CT)** – Completion time is the time at which the process has been terminated.

**WT (Waiting Time)**:

- The amount of time a process spends waiting in the ready queue before it gets the CPU for execution.
- Formula:
  **WT = Turnaround Time - Burst Time**
- It's the difference between the total turnaround time and the actual time the process spends on the CPU.

# FCFS [ CPU SCHEDULING]

Gantt Chart: A **Gantt chart** is a visual representation used to display the schedule of tasks or processes over time.

> TAT: Finished Time - Arrival Time

> WT: TAT - BT

| P.No | AT | BT | CT | TAT | WT |
|---|---|---|---|---|---|
| P0 | 0 | 4 | 4 | | |
| P1 | 1 | 5 | 9 | | |
| P2 | 2 | 2 | 11 | | |
| P3 | 3 | 3 | 14 | | |
| P4 | 4 | 6 | 20 | | |

| | | | | |
|---|---|---|---|---|
| P0 (0-4) | P1 (4-9) | P2 (9-11) | P3 (11-14) | P4 (14-20) |

# FCFS [ CPU SCHEDULING]

QUESTION 1:

> TAT: Finished Time - Arrival Time

> WT: TAT - BT

| P.No | AT | BT | CT | TAT | WT |
|---|---|---|---|---|---|
| P0 | 0 | 8 | | | |
| P1 | 1 | 3 | | | |
| P2 | 2 | 6 | | | |
| P3 | 3 | 2 | | | |
| P4 | 4 | 2 | | | |

| | | | | |
|---|---|---|---|---|
| P0 | P1 | P2 | P3 | P4 |

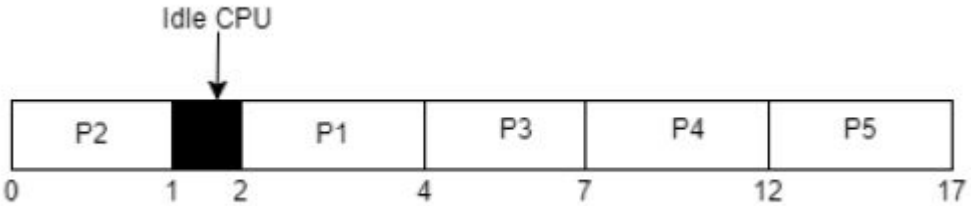
# FCFS [ CPU SCHEDULING]

Find Idle CPU Time, CT, TAT, WT

| Process ID | Arrival time | Burst time | CT | TAT | WT |
|---|---|---|---|---|---|
| P0 | 2 | 2 | | | |
| P1 | 0 | 1 | | | |
| P2 | 2 | 3 | | | |
| P3 | 3 | 5 | | | |
| P4 | 4 | 5 | | | |

# SOLUTION

| Process ID | Arrival time | Burst time | CT | TAT=CT-AT | WT=TAT-BT | RT |
|---|---|---|---|---|---|---|
| P1 | 2 | 2 | 4 | 4-2= 2 | 2-2= 0 | 0 |
| P2 | 0 | 1 | 1 | 1-0= 1 | 1-1= 0 | 0 |
| P3 | 2 | 3 | 7 | 7-2= 5 | 5-3= 2 | 2 |
| P4 | 3 | 5 | 12 | 12-3= 9 | 9-5= 4 | 4 |
| P5 | 4 | 5 | 17 | 17-4= 13 | 13-5= 8 | 8 |



# Shortest Job First (SJF) (Non-Preemptive) / SRTF (Preemptive)

**Non-Preemptive**: We cannot stop the process, once it is started, it will definitely gonna end.

| P.No | AT | BT | CT | TAT | WT |
|---|---|---|---|---|---|
| P0 | 3 | 1 | 7 | | |
| P1 | 1 | 4 | 16 | | |
| P2 | 4 | 2 | 9 | | |
| P3 | 0 | 6 | 6 | | |
| P4 | 2 | 3 | 12 | | |

| | | | | |
|---|---|---|---|---|
| P3 (0-6) | P0 (6-7) | P2 (7-9) | P4 (9-12) | P1 (12-16) |

# Shortest Job First (SJF) (Non-Preemptive)

Question: Find CT, TAT, WT

| P.No | AT | BT | CT | TAT | WT |
|---|---|---|---|---|---|
| P0 | 0 | 3 | | | |
| P1 | 3 | 5 | | | |
| P2 | 2 | 7 | | | |
| P3 | 4 | 9 | | | |
| P4 | 1 | 4 | | | |

| | | | | |
|---|---|---|---|---|
| | | | | |

# SRTF (Preemptive)

**Preemptive**: We can stop the process if the process have less burst time.

NOTE: Always first make Gantt Chart then you can better analyse what will be the finish time.

| P.No | AT | BT | CT | TAT | WT |
|---|---|---|---|---|---|
| P0 | 3 | 1 | 7 | | |
| P1 | 1 | 4 | 16 | | |
| P2 | 4 | 2 | 9 | | |
| P3 | 0 | 6 | 6 | | |
| P4 | 2 | 3 | 12 | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| P3(0-1) | P1(1-2) | P1(2-3) | P0(3-4) | P1(4-6) | P2(6-8) | P4(8-11) | P3(11-16) |

# SRTF (Preemptive)

Question:

1. Make Gantt Chart.

2. Find CT, TAT, WT.

| P.No | AT | BT | CT | TAT | WT |
|---|---|---|---|---|---|
| P0 | 0 | 1 | | | |
| P1 | 2 | 4 | | | |
| P2 | 1 | 2 | | | |
| P3 | 4 | 6 | | | |
| P4 | 3 | 3 | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

# Round Robin [CPU Scheduling] {Preemptive}

In **Round Robin (RR) scheduling**, the **time slice** (or **quantum**) is the fixed duration of CPU time allocated to each process before it is preempted and moved to the back of the queue.

Time Slice/quantum =2

NOTE: JO PROCESS EXECUTE HO RHA H VO LINE ME LAGEGA

| Process | AT | BT | CT | TAT | WT |
|---|---|---|---|---|---|
| P0 | 0 | 5 | 13-0 | | |
| P1 | 1 | 3 | 12-1 | | |
| P2 | 2 | 1 | 5-2 | | |
| P3 | 3 | 2 | 9-3 | | |
| P4 | 4 | 3 | 14-4 | | |

Maintain Queue

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| P0 | P1 | P2 | P0 | P3 | P4 | P1 | P0 | P4 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| P0(0-2) | P1(2-4) | P2(4-5) | P0(5-7) | P3(7-9) | P4(9-11) | P1(11-12) | P0(12-13) | P4(13-14) |

# Round Robin [CPU Scheduling] {Preemptive}

In **Round Robin (RR) scheduling**, the **time slice** (or **quantum**) is the fixed duration of CPU time allocated to each process before it is preempted and moved to the back of the queue.

Time Slice/quantum =2

| Process | AT | BT | CT | TAT | WT |
|---|---|---|---|---|---|
| P0 | 4 | 5 | | | |
| P1 | 3 | 3 | | | |
| P2 | 2 | 1 | | | |
| P3 | 1 | 2 | | | |
| P4 | 0 | 3 | | | |

Maintain Queue

# Round Robin [CPU Scheduling] {Preemptive}

Time Slice/ Quantum = 3

| Process ID | Arrival time | Burst time | CT | TAT | WT |
|---|---|---|---|---|---|
| P0 | 2 | 4 | | | |
| P1 | 0 | 1 | | | |
| P2 | 2 | 3 | | | |
| P3 | 3 | 5 | | | |
| P4 | 4 | 5 | | | |

# Priority Algorithm

1. NON-PREEMPTIVE APPROACH:

   JAB TAK KHATAM NA HOJAYE PROCESS

| Process | AT | BT | Priority | CT | TAT | WT |
|---|---|---|---|---|---|---|
| P0 | 0 | 4 | 2 | | | |
| P1 | 1 | 3 | 3 | | | |
| P2 | 2 | 1 | 4 | | | |
| P3 | 3 | 5 | 5 | | | |
| P4 | 4 | 2 | 5 | | | |

| | | | | |
|---|---|---|---|---|
| P0(0-4) | P3(4-9) | P4(9-11) | P2(11-12) | P1(12-15) |

# Priority Algorithm

1. NON-PREEMPTIVE APPROACH:

   JAB TAK KHATAM NA HOJAYE PROCESS

| Process | AT | BT | Priority | CT | TAT | WT |
|---|---|---|---|---|---|---|
| P0 | 4 | 1 | 4 | | | |
| P1 | 3 | 7 | 4 | | | |
| P2 | 2 | 9 | 2 | | | |
| P3 | 1 | 6 | 6 | | | |
| P4 | 0 | 3 | 8 | | | |

| | | | | |
|---|---|---|---|---|
| | | | | |

# Banker Algorithm: /Deadlock Avoidance Algorithm

The **Banker's Algorithm** is a resource allocation and deadlock avoidance algorithm used in operating systems. It helps manage multiple processes by ensuring that resource allocation does not lead to deadlock situations.
Basically, We have to provide all the information to the operating system which processes is coming, which process will request which resource and for how much time, then OS will perform accordingly.

**NOTE: This is used for Deadlock Detection that deadlock will occur in future or not.**

# Banker Algorithm: /Deadlock Avoidance Algorithm

**Question)** You have given Process, Allocation, Max Need, Current/Available, Need Matrix/ Remaining Need, Now

1. Find Need Matrix and Remaining Need?
2. Is the current allocation in safe state? If not answer the sequence.
3. Would (1,1,0) request be granted to P1.

# Banker's Algorithm

**Allocation**: This matrix represents the number of resources currently allocated to each process. Each row corresponds to a process, and each column corresponds to a resource type. It indicates how many instances of each resource are currently being used by that process.

**Max**: This matrix represents the maximum number of resources that each process may need to complete its task. It defines the upper limit of resource requests for each process.

**Available**: This vector represents the number of available instances of each resource type. It indicates how many resources are free and can be allocated to processes.

**Need**: This matrix is derived from the Max and Allocation matrices. It indicates the remaining resources needed by each process to complete its task.

# Banker Algorithm: /Deadlock Avoidance Algorithm

| Total Available Time | 12 | 11 | 20 |
|---|---|---|---|

| PROCESS | ALLOCATION | | | MAX | | | AVAILABLE | | | | NEED | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | R1 | R2 | R3 | R1 | R2 | R3 | R1 | R2 | R3 | | R1 | R2 | R3 |
| P1 | 2 | 2 | 3 | 3 | 6 | 8 | | | | | | | |
| P2 | 2 | 0 | 3 | 4 | 3 | 3 | | | | | | | |
| P3 | 1 | 2 | 4 | 3 | 4 | 4 | | | | | | | |
| Total Available | | | | | | | | | | | | | |

STEP1: Calculate Need Matrix : NEED= MAX- ALLOCATION
STEP2: Calculate Available : SUM OF ALLOCATIONS and Subtract it from Total Given Available Time
STEP2: SAFE STATE or NOT : Means Deadlock is occurring or not, If AVAILABLE is more than NEED
STEP3: Execute P1 and then add P1(Available) + P1(Allocation)

# Banker Algorithm: /Deadlock Avoidance Algorithm

1. Would (1,1,0) request be granted to P1.

Solve this by changing in Available/Current , by Subtracting and Addition in Allocation.

| Total Available Time | 12 | 11 | 20 |
|---|---|---|---|

| PROCESS | ALLOCATION | | | MAX | | | AVAILABLE | | | | NEED | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | R1 | R2 | R3 | R1 | R2 | R3 | R1 | R2 | R3 | | R1 | R2 | R3 |
| P1 | 2 | 2 | 3 | 3 | 6 | 8 | | | | | | | |
| P2 | 2 | 0 | 3 | 4 | 3 | 3 | | | | | | | |
| P3 | 1 | 2 | 4 | 3 | 4 | 4 | | | | | | | |
| Total Available | | | | | | | | | | | | | |

# Banker Algorithm: /Deadlock Avoidance Algorithm

| Total Available Time | 10 | 5 | 7 |
|---|---|---|---|

| PROCESS | ALLOCATION | | | MAX | | | AVAILABLE | | | | NEED | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | R1 | R2 | R3 | R1 | R2 | R3 | R1 | R2 | R3 | | R1 | R2 | R3 |
| P1 | 0 | 1 | 0 | 7 | 5 | 3 | | | | | | | |
| P2 | 2 | 0 | 0 | 3 | 2 | 2 | | | | | | | |
| P3 | 3 | 0 | 2 | 9 | 0 | 2 | | | | | | | |
| P4 | 2 | 1 | 1 | 4 | 2 | 2 | | | | | | | |
| P3 | 0 | 0 | 2 | 5 | 3 | 3 | | | | | | | |
| Total Available | | | | | | | | | | | | | |

# Banker Algorithm: /Deadlock Avoidance Algorithm

| Total Available Time | 20 | 10 | 14 |
|---|---|---|---|

| PROCESS | ALLOCATION | | | MAX | | | AVAILABLE | | | | NEED | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | R1 | R2 | R3 | R1 | R2 | R3 | R1 | R2 | R3 | | R1 | R2 | R3 |
| P1 | 2 | 4 | 5 | 8 | 6 | 9 | | | | | | | |
| P2 | 5 | 3 | 1 | 7 | 8 | 10 | | | | | | | |
| P3 | 2 | 1 | 4 | 12 | 5 | 9 | | | | | | | |
| P4 | 2 | 1 | 1 | 4 | 2 | 2 | | | | | | | |
| P3 | 0 | 0 | 2 | 5 | 3 | 3 | | | | | | | |
| Total Available | | | | | | | | | | | | | |

# PAGE REPLACEMENT ALGORITHMS (FIFO)

**Page replacement algorithms** are used in operating systems to manage memory, specifically when physical memory (RAM) is full and a new page needs to be loaded from disk.

Q) Consider the following page references string.
1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6

(a) Calculate the no. of page fault & page hit for (a) FIFO (b) LRU (c) Optimal

| Frame | 1 | 2 | 3 | 4 | 2 | 1 | 5 | 6 | 2 | 1 | 2 | 3 | 7 | 6 | 3 | 2 | 1 | 2 | 3 | 6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | |
| Page Fault/Page Hit | | | | | | | | | | | | | | | | | | | | |

# Page Replacement Algorithm (Using FIFO)

Q) Consider the following page references string.
6,3,2,1,2,3,6,7,3,2,1,2,6,5,1,2,4,3,2,1

(a) Calculate the no. of page fault & page hit for (a) FIFO (b) LRU (c) Optimal

| **Frame** | **6** | **3** | **2** | **1** | **2** | **3** | **6** | **7** | **3** | **2** | **1** | **2** | **6** | **5** | **1** | **2** | **4** | **3** | **2** | **1** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | |
| Page Fault/Page Hit | | | | | | | | | | | | | | | | | | | | |

# Page Replacement Algorithm (Using LRU)

Q) Consider the following page references string.
1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6

(a) Calculate the no. of page fault & page hit for (a) FIFO (b) LRU (c) Optimal

| **Frame** | **1** | **2** | **3** | **4** | **2** | **1** | **5** | **6** | **2** | **1** | **2** | **3** | **7** | **6** | **3** | **2** | **1** | **2** | **3** | **6** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | |
| Page Fault/Page Hit | | | | | | | | | | | | | | | | | | | | |

**NOTE**: CHECK FROM LEFT HAND SIDE

# Page Replacement Algorithm (Using LRU)

Q) Consider the following page references string.
6,3,2,1,2,3,6,7,3,2,1,2,6,5,1,2,4,3,2,1

(a) Calculate the no. of page fault & page hit for (a) FIFO (b) LRU (c) Optimal

| **Frame** | **6** | **3** | **2** | **1** | **2** | **3** | **6** | **7** | **3** | **2** | **1** | **2** | **6** | **5** | **1** | **2** | **4** | **3** | **2** | **1** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | |
| Page Fault/Page Hit | | | | | | | | | | | | | | | | | | | | |

# Page Replacement Algorithm (Using Optimal)

Q) Consider the following page references string.
1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6

(a) Calculate the no. of page fault & page hit for (a) FIFO (b) LRU (c) Optimal

| **Frame** | **1** | **2** | **3** | **4** | **2** | **1** | **5** | **6** | **2** | **1** | **2** | **3** | **7** | **6** | **3** | **2** | **1** | **2** | **3** | **6** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | |
| Page Fault/Page Hit | | | | | | | | | | | | | | | | | | | | |

**NOTE**: CHECK FROM RIGHT HAND SIDE

# Page Replacement Algorithm (Using Optimal)

Q) Consider the following page references string.
6,3,2,1,2,3,6,7,3,2,1,2,6,5,1,2,4,3,2,1

(a) Calculate the no. of page fault & page hit for (a) FIFO (b) LRU (c) Optimal

| **Frame** | **6** | **3** | **2** | **1** | **2** | **3** | **6** | **7** | **3** | **2** | **1** | **2** | **6** | **5** | **1** | **2** | **4** | **3** | **2** | **1** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | |
| Page Fault/Page Hit | | | | | | | | | | | | | | | | | | | | |

# THANK YOU