Scheduling Algorithms :-

Preemptive scheduling

- Round Robin (RR)
- Shortest Remaining Time First (SRTF) (SJF Preemptive)
- Priority Scheduling (Preemptive)

Non-Preemptive scheduling

- FCFS
- Shortest Job first (FCFS)
- Priority Scheduli (Preemptive)

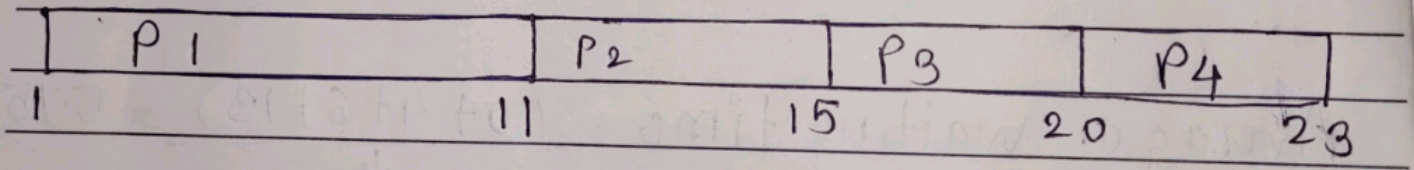
Q.8 Consider the following set of processes with their CPU burst time

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

Calculate average waiting time & Average turnaround time using FCFS

⇒ Turnaround Time (TAT) = CT - AT
waiting time = TAT - BT

Process	Arival Time	Burst time	CT	TAT	WT
P_1	1	10	11	11-1=10	10-10=0
P_2	2	4	15	15-2=13	13-4=9
P_3	3	5	20	20-3=17	17-5=12
P_4	4	3	23	23-4=19	19-3=16



Average Turnaround Time = $\frac{10+13+17+19}{4}$

$= 14.75$ ms

Average waiting time = $\frac{0+9+12+16}{4}$

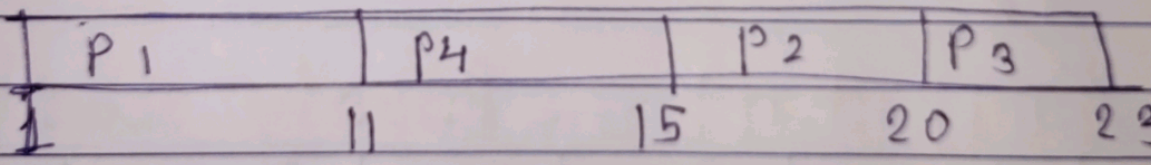
$= 9.25$ ms

* Shortest Job first (same Previous Q.)
(Non-preemptive)

=> Process Burst Time AT TAT WT CT

P_1	10	1			
P_2	4	2			
P_3	5	3			
P_4	3	4			

Gantt chart :-

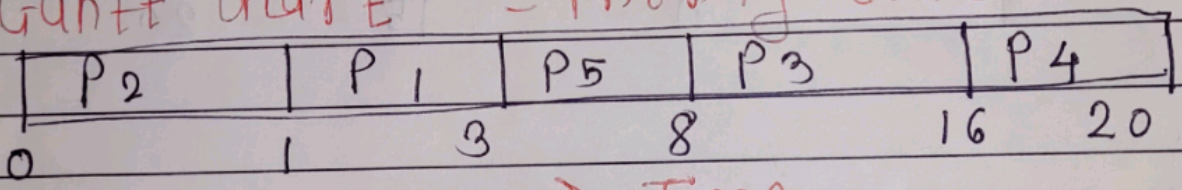


Priority Scheduling (Non-preemptive)

Q. Consider the following set of processes assuming all are arriving at time 0. Calculate Average waiting time and TAT using Priority (Non-Preemptive)

Process	Burst time	Priority
P_1	2	2
P_2	1	1 - Higher
P_3	8	4
P_4	4	5 - Lower
P_5	5	3

⇒ Gantt chart - Priority scheduling



TAT = CT - AT

WT = TAT - BT

mode = "Non-preemptive"

criteria = Priority

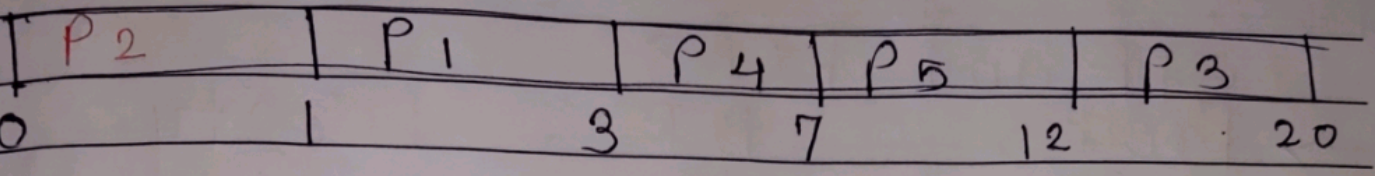
Process	BT	AT	TAT	CT	WT
P_1	2	0	3-0=3	3	3-2=1
P_2	1	0	1-0=1	1	1-1=0
P_3	8	0	16-0=16	16	16-8=8
P_4	4	0	20-0=20	20	20-4=16
P_5	5	0	8-0=8	8	8-5=3

Average TAT = $\frac{3+1+16+20+8}{5}$ = 9.6ms

Average WT = $\frac{1+0+8+16+3}{5}$ = 5.6ms

SJF (Non-preemptive)

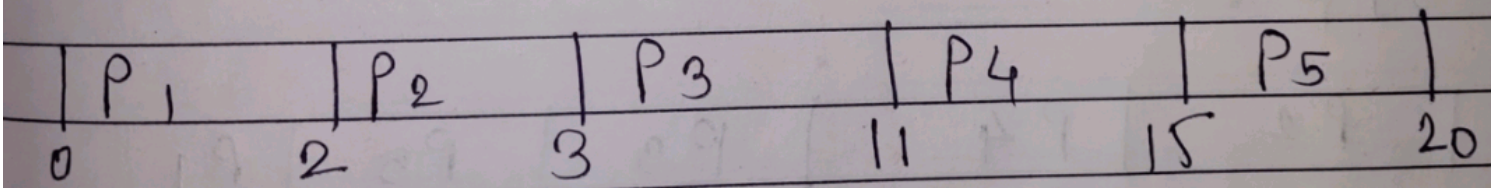
Gantt chart



Avg. TAT

Avg. WT

FCFS

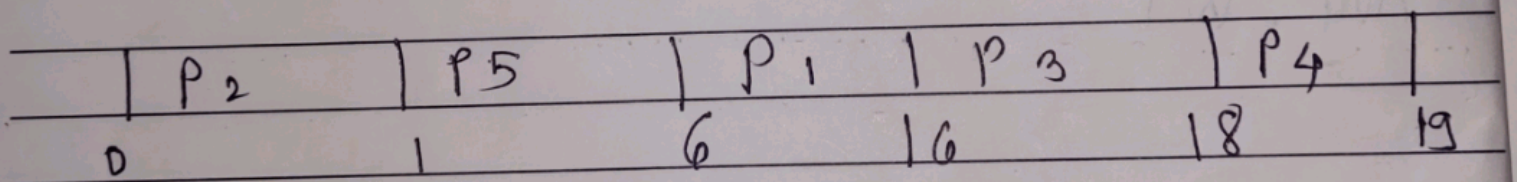


Q. Arrival time for all processes is 0 Solve using Non-preemptive priority SJF and FCFS.

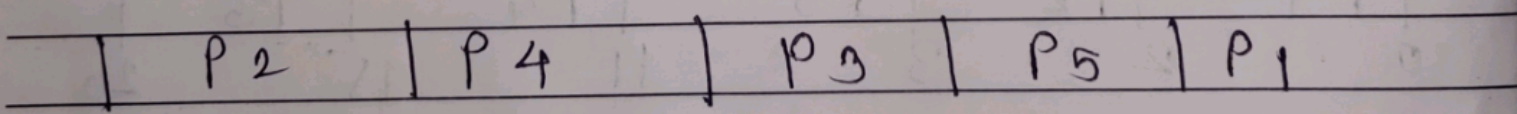
⇒

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

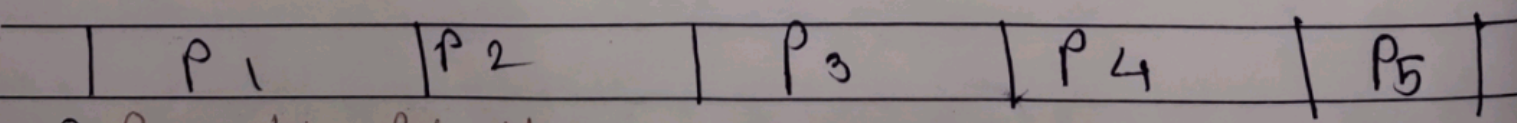
⇒ Priority scheduling :-



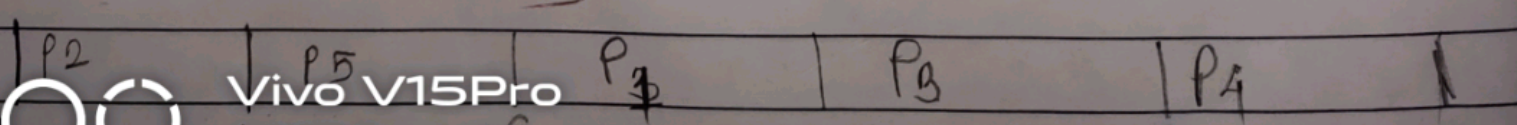
SJF :-



FCFS :-



Preemptive Priority



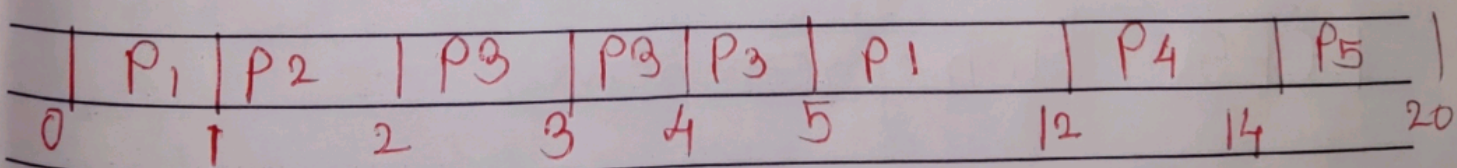
Non-preemptive Priority Scheduling

Q. Use following scheduling algo. to calculate Average TAT and avg. WT for following processes
① FCFS(Non) ② SJF (Non) ③ Preemptive Priority

⇒ Preemptive Priority

criteria = "Priority"
Mode = Non-preemptive

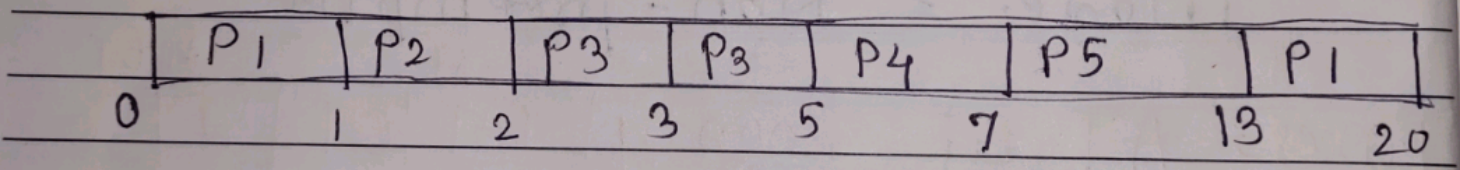
Process	AT	BT	Priority	CT-AT TAT	CT	TAT-BT WT
P1	0	7	3			
P2	1	1	1 - H			
P3	2	3	2 - H			
P4	3	2	3			
P5	4	6	4 - L			



(2) Shortest Job first (Preemptive)

Process	AT	BT	Priority	TAT	CT	CT-AT (WT)	TAT-BT
P1	0	~~8~~ 7	3	20	20	12	
P2	1	~~1~~ 0	1	1	2	0	
P3	2	~~3~~ 2	2	3	5	0	
P4	3	2	3	4	7	~~3~~ 2	
P5	4	6	4	9	13	~~7~~ 3	

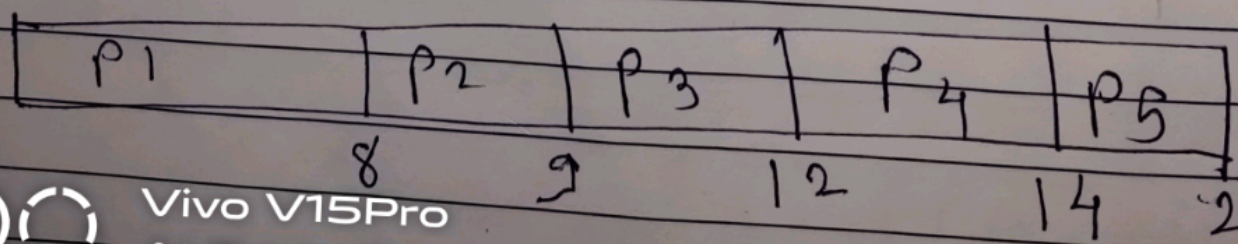
⇒ Gantt chart:



Average TAT = $\frac{20+1+3+4+9}{5} = \frac{37}{5} = 7.4$ ms

Average WT = $\frac{12+0+0+2+3}{5} = \frac{17}{5} = 3.4$ ms

FCFS (Non-preemptive)

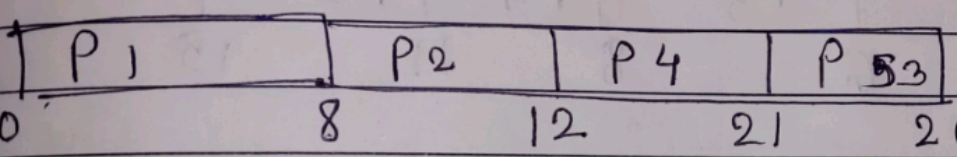


(3) SJF (Preemptive and Non-preemptive)

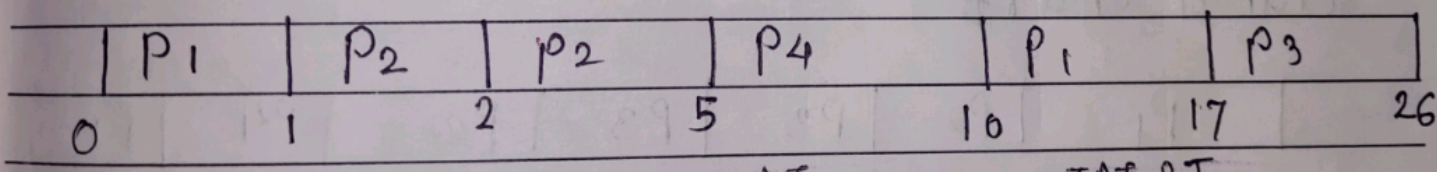
⇒

Process	Arrival Time	Burst Time
P_1	0	~~8~~ 7
P_2	1	~~4~~ 3 2
P_3	2	9
P_4	3	5

⇒ Gantt chart (SJF- Non-preemptive).



Gantt chart (SJF - Preemptive)



CT-AT TAT-BT

Process	AT	BT	TAT	CT	WT
P_1	0	8	17	17	9
P_2	1	4	4	5	0
P_3	2	9	24	26	~~15~~ 15
P_4	3	5	7	10	~~2~~ 2

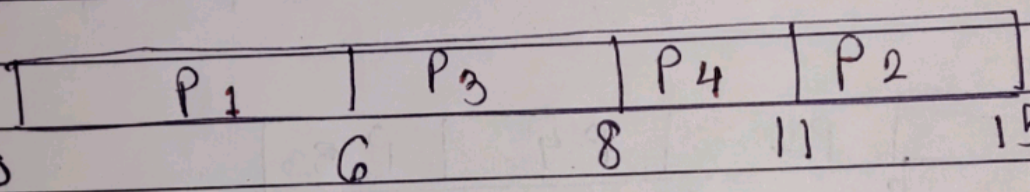
Average WT = $\frac{26}{4}$ = 6.5 ms

(4)

Process	AT	Burst Time
P1	0	~~6~~ 5
~~P2~~	1	~~4~~ 3
~~P3~~	2	~~2~~ 1
P4	3	3

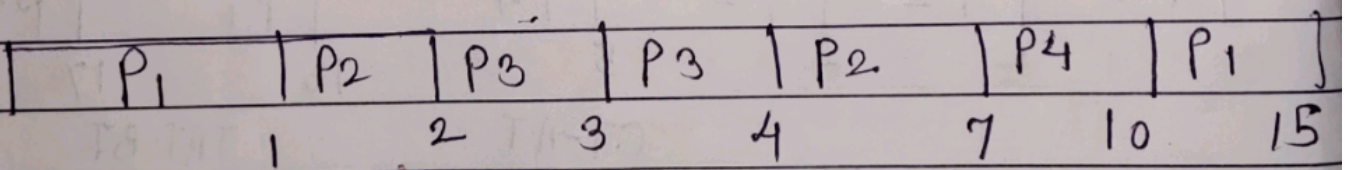
⇒ SJF (Non-preemptive)

Gantt chart



SJF (Preemptive)

Gantt chart



time

Process	AT	BT	CT	TAT → WT
P1	0	6	15	9
P2	1	4	7	6 2
P3	2	2	4	2 0
P4	3	3	10	7 4

Round Robin Scheduling algorithm (Preemptive)

Response time = { CPU first time used − AT }

Q. Process | Burst Time | Arrival Time

P_1	~~24~~ ~~20~~ 16	0
P_2	~~3~~ 0	0
P_3	~~3~~ 0	0

Time Quantum = 4ms. Find average Response Time.

⇒ Round Robin Non-preemptive Algorithm

Criteria = "Time Quantum", mode = "Preemptive"

Ready Queue: P_1 | ~~P_2~~ | ~~P_3~~ | P_1 | P_1 | P_1 (0)

Running Queue: | P1 | P2 | P3 | P1 | P1 | P1 | P1 | P1 |

0 4 7 10 14 18 22 ~~26~~ 26 30

CT − AT TAT − BT

Process	AT	BT	CT	TAT	WT	RT
P_1	0	24	30	30	6	0 − 0 = 0
P_2	0	3	7	7	4	4 − 0 = 4
P_3	0	3	10	10	7	7 − 0 = 7

$\frac{0+4+7}{3} = \frac{11}{3}$

$\frac{6+4+7}{3} = 3.66ms$

~~= 12.33ms~~

= 5.66ms

Q.2.

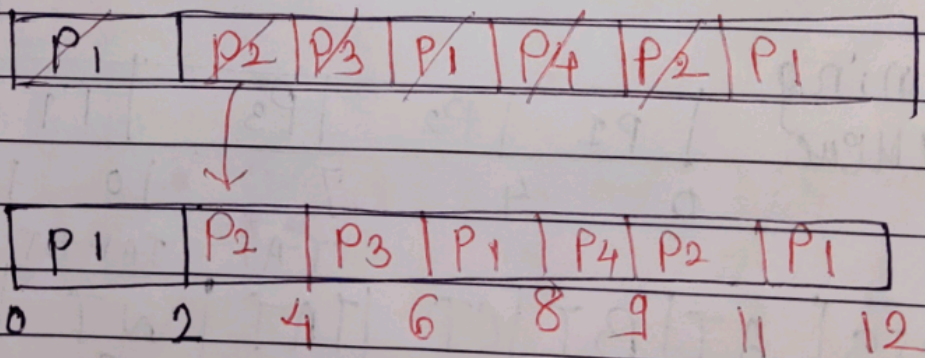
Process	AT	BT	CT	TAT	WT	RT
P1	0	5 3 1	12	12	7	0-0 = 0
P2	1	4 2 0	11	10	6	2-1 = 1
P3	2	2 0	6	4	2	4-2 = 2
P4	4	1 0	9	5	4	8-4 = 4
					4.75ms	1.75 ms

find out average waiting time and average response time. Time Quantum = 2ms

⇒

Given Time Quantum = 2ms.

Mode = "Preemptive"
Criteria = "Time Quantum"



① Chek which process is arrived first and allocate CPU to that process for given time quantum. if two processes arrives at same time then follow FCFS order

(2) Next step check running queue time and see whether in that time interval any other process is arrived. if yes add them in Ready queue in sequence.

(3) Then take, check whether first process has completed its burst time or not if no, then at that process at end in ready queue.

(3) Take one one process from Ready to Running @ for given time quantum and solve.

Response time = {CPU first time used − AT}

Average Response time = $\frac{0+1+2+4}{4}$

= 1.75ms.

Average WT = $\frac{7+6+2+4}{4}$ = 4.75ms.

Q. Process Synchronization

Race Condition Example :-

Example ① shared = 5

P1	P2
int x = shared;	int y = shared
x++;	y--;
sleep(1);	sleep(1);
shared = x;	shared = y;

Example ②

shared

```
void bankAccount (double money)
{
}
shared = shared + money
```

*

*Peterson's Solution for Critical Sect. Problem

Structure of process Pi

```
do {
    flag[i] = true;
    turn = j;
    while (flag[j] && turn == j);
        critical section
    flag[i] = false;
        remainder section
} while (true);
```

```
do {
    flag[1] = true;
    turn = 2;
    while (flag[2] && turn = = 2);
        critical section
    flag[1] = false;
        remainder section
} while (true);
```

Three conditions that Peterson's solution satisfies :-

1. Mutual Exclusion
2. Progress
3. Bounded waiting

Semaphores :

A semaphore S is an integer variable that apart from initialization is accessed only through two standard atomic operations; wait() & Signal().

P() - Down, wait
V() - up, signal

Types :- Counting & Binary

Entry section code -

Down (Semaphore S) 3 or 1

```
{
   S.value = S.value - 1
   if (S.value < 0)
{
   Put Process (PCB) in Suspended list,
                sleep();
   }
else
   return;
}
```

Exit section code

```
up(Semaphore S)
{
    S.value = S.value + 1;
    if (S.value <= 0)
    {
        select a process from suspended list
        & wake up();
    }
}
```

Example of lock

@what is semaphore and its types?
In a certain appln, a value of counting semaphore is 17. The following operations were completed on the semaphores in the given order 2P, 2OP, 5V, 10V, 10P, 2P. What would be the new value of counting semaphore?

⇒ Given counting semaphore value = 17.

The sequence of wait and signal operations is given as 2P, 2oP, 5V, 10V, 10P, 2P.

The new value of counting semaphore

= ~~17 - 2P - 20P + 5~~

= 17 - 2 - 20 + 5 + 10 - 10 - 2

= ~~17 - 27 - 2~~ 17 - 22 + 3

= ~~17 - 29~~ 20 - 22

= -2.

∴ The new value of counting semaphore is -2. So there are two processes which are waiting to enter into critical section.

Memory Management

Logical address space	Physical Address space.
• Generated by CPU	• Generated by memory unit.
• Just an illusion • Also referred as virtual address	
• Set of all logical addresses generated by Program	• set of all physical addres generated by Program.

★ Logical address & Physical are same in compile time load time address binding scheme.

★ Logical & Physical address are differ in execution time add-binding scheme.

★ Degree of Multiprogramming

Memory Hierarchy

Typical access time		Capacity
1 nsec	Registers	(v. fast) < 1 KB
2 nsec	Cache	(slow fast) 4 MB
10 nsec	RAM	(fast slow) 2 - 64 GB
10 msec	Hard Disk	(v. f. Very slow) 1 - 2 TB
	Offline Storage (Tape Drives etc)	

Bootstrap loader will load OS ~~programs from secondary storage to main memory~~

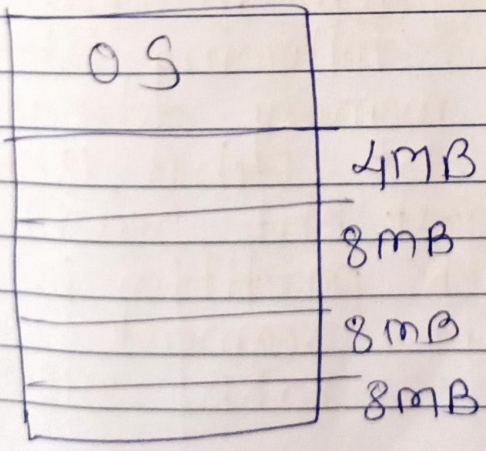
from secondary Storage. Programs will be loaded into main memory by using paging.

$2^1 = 2$
$2^2 = 4$
$2^3 = 4$
$2^4 = 16$
$2^5 = 32$
$2^6 = 64$
$2^7 = 128$
$2^8 = 256$
$2^9 = 512$
$2^{10} = 1024 - 1K$

$2^{20} = 1M$
$2^{30} = 1G$
$1T = 2^{40}$.

Memory Management

- Contiguous
  - Static
    - fixed Partition
      - No. of Partitions are fixed
  - Dynamic
    - Variable Partition
- Non-Contiguous
  - Paging
  - Multilevel Paging
  - ~~Hierarchical~~ Inverted Paging
  - Segmentation
  - Segmentation Paging



Memory Allocation Techniques

First Fit | Best Fit | Next Fit | Worst Fit

First Fit :- Chooses the first available block that is large enough

Best fit :- Chooses the block that is closest in size to the request

Next fit :- Searches for Loc^n of last placement & chooses next available block which is large enough

Worst Fit :- It allocates largest hole. Searches for largest partition & allocates.

* Consider swapping system in which memory consists of the following hole size or partition sizes in memory order as 20KB, 8KB, 40KB, 36KB, 14KB, 18KB, 24KB, 30KB. Using First Fit, Best fit & worst fit see which partition is selected for successive segment request of 1) 24KB 2) 10KB 3) 18KB

⇒ Given

20KB	8KB	40KB	36KB	14KB	18KB	24KB	30KB

⇒ First fit for 24KB.

		Used cell	Unused cell				
20KB	8KB	40KB		36KB	14KB	18KB	24KB
		24KB	16KB				

First fit for 10KB.

Used	Unused		Used	Unused					
20KB		8KB	40KB		36KB	14KB	18KB	24KB	30KB
10KB	10KB		24KB	16KB					

First fit for 18 KB

Used	Unused		Used	Unused	Used	Unused				
20KB		8KB	40KB		36KB		14KB	18KB	24KB	30KB
10KB	10KB		24KB	16KB	18KB	18KB				

Best fit for 24KB

						Used	
20KB	8KB	40KB	36KB	14KB	18KB	24KB	30KB

Best fit for 10KB

20KB	8KB	40KB	36KB	Used 14KB / 10KB	Unused 4KB	18 KB	24KB	30KB

Best fit for 18 KB

20KB	8KB	40KB	36KB	Used 14KB / 10KB	Unused 4KB	Used 18 KB	24KB	30KB

Similarly solve using worst fit

Allocate largest hole for request

Worst fit for 24 KB

20KB	8KB	Used 40KB / 24KB	Unused 16KB	36KB	14KB	18KB	24KB	30KB

for next request 36KB block will be given

University question

Q. A variable partition memory system has at some point in time, the following hole sizes in the given order:
20K, 15K, 40K, 60K, 10K, 25K. A new process of 25K is to be loaded. which hole size would be filled using best fit, first fit and worst fit respectively.

⇒

Best fit for 25K will prefer ~~25~~ hole of 25K.

Because, best fit deals with allocating the smallest free partition which meets given request.

20K	15K	40K	60K	10K	Used 25K

first fit for 25K will prefer first block which satisfies the request i.e. 40K
It will claim first available memory

20K	15K	Used 40K / ~~25K~~	Unused 15K	60K	10K	25K

Worst fit for 25K

It will prefer largest hole which is big enough

20K	15K	40K	Used Space 60K 25K	35K	10K	25K

Paging :- (Non-Contiguous Allocation)

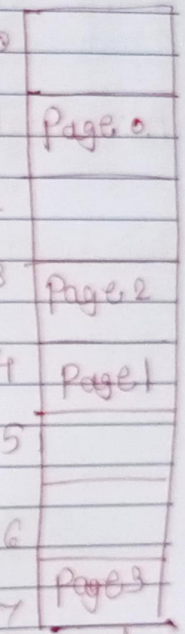
- No. of pages = no. of frames in Physical memory
- Each process will have its separate page table

Page 0
Page 1
Page 2
Page 3

Logical Memory

0	1
1	4
2	3
3	7

Page table



* Process is divided into equal size pages and keep it inside main memory

Note :- we need page tables to map from logical address to physical address of a byte.

Swapping :-

Is a mechanism in which a process can be swapped temporarily out of main memory to secondary storage (disk) and make that memory available to other processes. At some later time, the system swaps back the process from secondary storage to main memory

Advantage :- Multiprogramming

Disadvantage : Context switching

Consider a logical address space of eight pages of 1024 words each, mapped into a physical memory of 32 frames. (a) How many bits are there in the logical address? (b) How many bits are there in the physical address?

⇒ (a) Logical address

Addressing within a 1024-word page requires 10 bits because $1024 = 2^{10}$

Since the logical address space consists of 8 pages $= 2^3$ pages.

Size of logical address space = No. of pages × Page size

$$2^m = 8 \times 1024$$
$$= 2^3 \times 2^{10}$$
$$= 2^{3+10}$$
$$= 2^{13}$$

Therefore the no. of regd. bits in the logical address = 13 bits.

(b) Physical address

Size of physical address space $= 2^n$

Size of physical address space = No. of frames × frame size

Size of physical address space, = 32 X 1024

$$2^K = 2^5 \times 2^{10}$$

$$2^K = 2^{5+10}$$

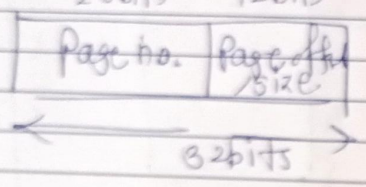
Therefore Total no. of reqd. bits in physical address = 15 bit.

Logical address space & Physical address space

Given LAS = 4GB , Page size = 4KB
PAS = 64MB

find out no of pages ? , no of frames , no of entries in Page Table ?

⇒ Memory is byte addressable
LAS = 4GB $= 2^2 \times 2^{30} = 2^{32}$
Logical address = 32 bits are required



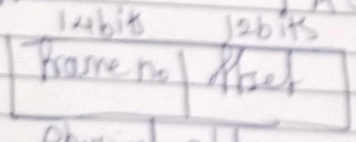
Page Size = 4 KB $= 2^2 \times 2^{10}$
$= 2^{12}$
= 12 bits

20 no of bits are used to represent page no

no of pages $= 2^{20}$

No of frames ?

PAS = 64 MB $= 2^6 \times 2^{20}$ B
$= 2^{26}$.



no of frames $= 2^{14}$
$= 16384$

No of page table entries = no of pages in Page Table
$= 2^{20}$.

$2^1 = 2$
$2^2 = 4$
$2^3 = 4$
$2^4 = 16$
$2^5 = 32$
$2^6 = 64$
$2^7 = 128$

$2^{10} = 1K = 1024$
$2^{20} = 1M$
$2^{30} = 1G$
$2^{40} = 1T$

Numerical based on Segmentation (Module 3)

Q. Consider the following segtable Table

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

What are the physical address for the following addresses ?

a. 0,430
b. 1,10
c. 2,500
d. 3,400
e 4,112.

⇒ a. Given 0,430 (logical address)

Segment no - 0 → offset 430

Entry corresponding to segment no 0 in Segment table is,

Base = 219, size (limit) = 600

As offset < size (limit) i.e 430 < 600

Physical address = Base + offset

= 219 + 430

= 649.

b. Given 1,10

Base = 2300 size = 14

As offset < size

10 < 14

Physical address = Base + offset

= 2300 + 10

= 2310

c. 2,500

Base = 90 size = 100

As offset < size ∝

500 ≥ 100

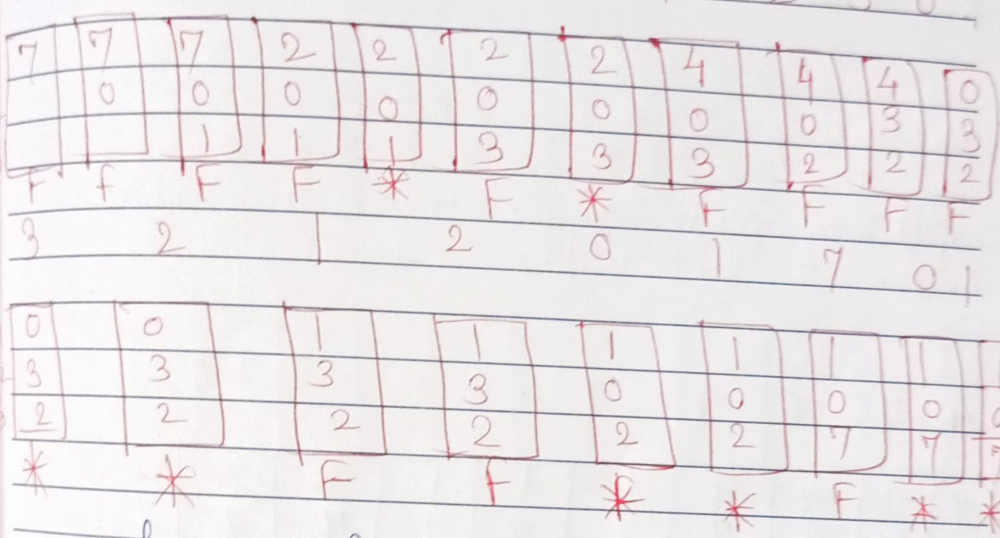
∴ Here Reference to physical address of logical address (2,500) will result in an error.

last

LRU (Least Recently Used) :-

Replaces the page frame that is not used recently.

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

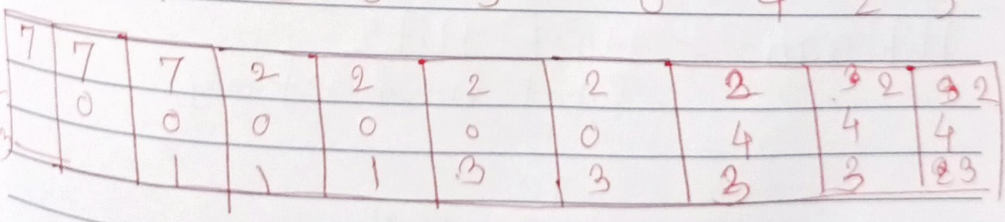


No. of Page Faults = 12
Number of Page Hits = 8

Optimal Page Replacement Algorithm

Replaces the page that is not in demand in future

7 0 1 2 0 3 0 4 2 3



0 3 2 1 2 0 1 7 0 1

2	3	2	2	2	2	2	7	7	7
0	0	0	0	0	0	0	0	0	0
3	3	3	1	1	1	1	1	1	1

Q. Reference string given as,
4, 7, 6, 1, 7, 6, 1, 2, 7, 2. Solve using optimal page replacement algorithm. Find out hit ratio and miss ratio.

⇒

4 7 6 1 7 6 1 2 7 2

f1	4	4	4	1	1	1	1	2	2	2
f2		7	7	7	7	7	7	7	7	7
f3			6	6	6	6	6	6	6	6
	F	F	F	F	*	*	*	F	*	*

No. of Page faults (F) = 5
No. of Page Hits (*) = 5

Hit ratio = $\frac{\text{No. of Hits}}{\text{Total no. of reference}}$
$= \frac{5}{10}$
$= 0.5 = 50\%$

Miss ratio = $\frac{\text{No. of Miss}}{\text{Total no. of references}}$
$= \frac{5}{10}$
$= 0.5$
$= 50\%$.

LRU (Least Recently Used)

4 7 6 1 7 6 1 2 7 2

f1	4	4	4	1	1	1	1	1	1	1
f2		7	7	7	7	7	7	2	2	2
f3			6	6	6	6	6	6	7	7
	F	F	F	F	* Hit	* Hit	Hit	F	F	*

No. of Hits = 4
No. of Page faults = 6

Hit ratio = $\frac{\text{No. of Hits}}{\text{Total no. of references}}$
$= \frac{4}{10} = 0.4 = 40\%$

Miss ratio = $\frac{\text{No. of Page faults}}{\text{Total no. of references}}$
$= \frac{6}{10} = 0.6 = 60\%$

- Consider the page reference String
  1, 2, 3, 5, 2, 4, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.

Calculate the page Fault using
1. Optimal 2. LRU 3. FIFO algorithm for a memory with three frames.

→

FIFO

1	2	3	5	2	4	5	6	2	1	2	3	7
1	1	1	5	5	5	5	5	2	2	2	2	7
	2	2	2	2	4	4	4	4	1	1	1	1
		3	3	3	3	3	6	6	6	6	3	3
F	F	F	F	H	F	H	F	F	F	H	F	F

6	3	2	1	2	3	6
7	7	7	1	1	1	1
6	6	6	6	6	3	3
3	3	2	2	2	2	6
F	H	F	F	H	F	F

Optimal Page Replacement (Replace page which is not require in future, require very late)

1	2	3	5	2	4	5	6	2	1	2	3
1	1	1	1	1	4	4	6	6	6	6	6
	2	2	2	2	2	2	2	2	2	2	2
		3	5	5	5	5	5	5	1	1	3
F	F	F	F	*	F	H	F	H	F	F	F

7	6	3	2	1	2	3	6
6	6	6	6	1	1	1	6
7	7	7	2	2	2	2	2
3	3	3	3	3	3	3	3
F	H	H	F	F	H	H	F

LRU (Least recently Used)

1	2	3	5	2	4	5	6	2	1	2
1	1	1	5	5	5	5	5	5	1	1
	2	2	2	2	2	2	6	6	6	6
		3	3	3	4	4	4	2	2	2
F	F	F	F	H	F	H	F	F	F	H

3	7	6	3	2	1	2	3	6
1	7	7	7	2	2	2	2	2
3	3	3	3	3	3	3	3	3
2	2	6	6	6	1	1	1	6
F	F	F	H	F	F	H	H	F

*Disk Scheduling Algorithms

- First Come First Serve (FCFS)
- Shortest Seek Time First (SSTF)
- Elevator (SCAN)
- Circular SCAN (C-SCAN)
- LOOK
- C-LOOK

Note:- No of head posⁿ. movements = Disk access time

Q. Given the following

SJF

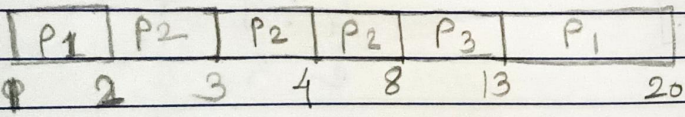
①

Processes	Burst Time	Arrival Time
P_1	8	0
P_2	3	0
P_3	4	0
P_4	11	0

②

Processes	Arrival Time	Burst Time
P_1	1	~~8~~ 7
P_2	2	~~6~~ 4
P_3	3	5

① How many resources are there of type A, B & C.
② Find need Matrix
③ If a request from process P_1 arrives for (0, 4, 2, 0), can the request be granted immediately.



Banker's Algorithm :-

①

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	5	2	0
P_1	1	0	0	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

⇒ Step 1 :- Find need matrix
Need = Max - Allocation

Need	A	B	C	D
P_0	0	0	0	0
P_1	0	7	5	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

Step 2 : To decide whether system is in safe state or not.

Process P_0 — A B C D — A B C D
Need <= Avail. i.e (0,0,0,0) <= (1,5,2,0)
is true. Process P_0 is executed.

New Avail. = Availability + Alloc.
= (1,5,2,0) + (0,0,1,2) = (1,5,3,2)

Process P_1
Need <= Avail. i.e. (0,7,5,0) <= (1,5,3,2)
is false, Resource B & C need is more then avail., P_1 is not executed.

Process P_2

Need <= Avail. i.e. (1,0,0,2) <= (1,5,3,2)
is True. P_2 is executed.

New Avail. = Avail. + Alloc = (1,5,3,2) + (1,3,5,4)
New Avail. = (2,8,8,6)

Process P_3
(0,0,2,0) <= (2,8,8,6) is True.
P_3 is executed.

New Avail = (2,8,8,6) + (0,6,3,2)
= (2,14,11,8)

Process P_4 (0,6,4,2) <= (2,14,11,8) is True.
Process P_4 is executed. = (2,14,11,8) + (0,0,1,4)

Process P_1, $(0,6,4,2) <= (2,14,12,12)$ is True
Process P_1 is executed.

New. Avail = $(2,14,12,12) + (1,0,0,0)$
$= (3,14,12,12)$

System is in Safe state, because Safe seq. exists such as $(P_0, P_2, P_3, P_4, P_1)$

Step 3:- If a request from Process P_1 arrives for $(0,4,2,0)$, can the request be granted immediately?

→ Request $<=$ Need i.e $(0,4,2,0) <= (0,7,5,0)$ is True

→ Request $<=$ Avail. i.e $(0,4,2,0) <= (1,5,2,0)$ is True. Avail. is $(1,5,2,0)$.

Update the snapshot with new Avail. allocation and need.

Avail = Avail − Request $= (1,5,2,0) - (0,4,2,0)$
$= (1,1,0,0)$

Alloc. = Alloc. + Request $= (1,0,0,0) + (0,4,2,0)$
$= (1,4,2,0)$

Need = Need - Request = (0,7,5,0) - (0,4,2,0)
= (0,3,3,0)

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	1	0	0
P_1	1	4	2	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

	Need			
	A	B	C	D
P_0	0	0	0	0
P_1	0	3	3	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

P_0 P_2 P_3 P_4 P_1

Process P_0

New avail = (3,14,12)