Name : Udoy Saha

ID : 23341134

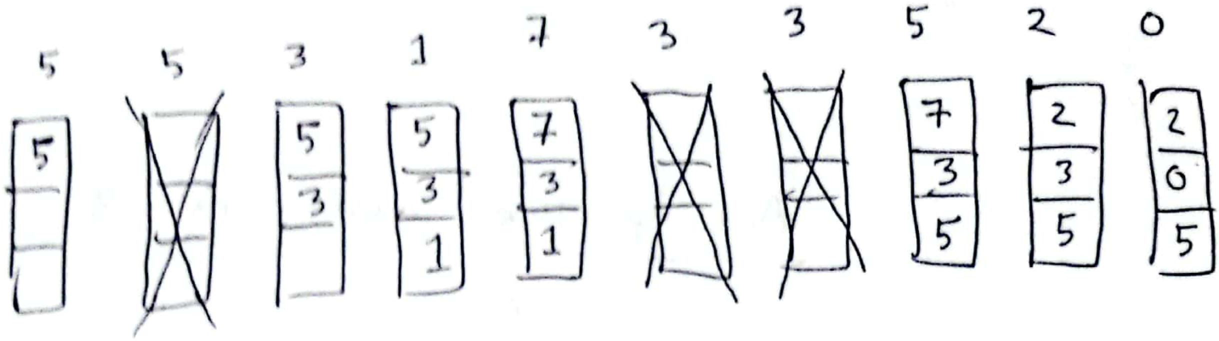
Section : 03

Ans to the ques no:- 4

a

Here, frames available = 3.

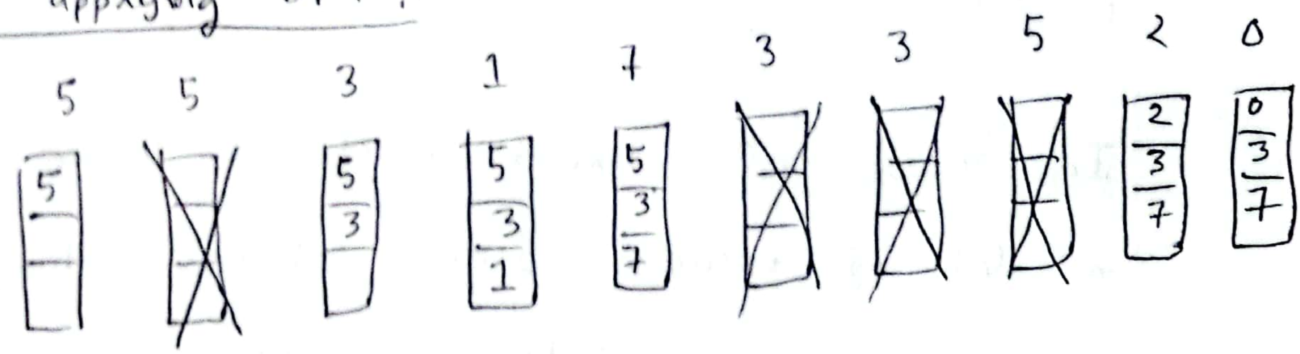
Firstly applying LRU:



$\therefore$ Hit ratio = $\frac{3}{10} \times 100\% = 30\%$

~~Miss~~ Page fault = $\frac{7}{10} \times 100\% = 70\%$

Now applying OPT :-



$\therefore$ Hit ratio = $\frac{4}{10} \times 100\% = 40\%$

Page fault = $\frac{6}{10\cancel{0}} \times 100\% = 60\%$

In this scenario, OPT performs better. Because, it has a higher hit ratio and lower page faults.

—∘——✕——∘—

Ans to the ques no:- 3

a

If we want to apply contiguous memory allocation, we have 2 choices : either choose fixed partition or choose variable partition.

In case of fixed partition, there are a lot of issues like internal fragmentation, external fragmentation, multiprocessing degree limitation and process size limitation. These cause processes to go in a starvation state.

The variable or dynamic memory allocation reduces some issues, but it still has external fragmentation.

Due to these issues, it is not recommended to use contiguous memory allocation.

[b]

Here, EAT = 95 ns

Associative lookup time, ε = 2ns

Memory access time, t_m = 72 ns

Hit ratio, α = ?

We know,

$$EAT = \alpha \times (\varepsilon + t_m) + (1-\alpha)\cdot(2 \times t_m)$$

$$\Rightarrow 95 = \alpha\,(2+72) + (1-\alpha)\cdot(2\times 72)$$

$$\Rightarrow 95 = 74\alpha + 144(1-\alpha)$$

$$\Rightarrow 95 = 74\alpha + 144 - 144\alpha$$

$$\Rightarrow 95 = -70\alpha + 144$$

$$\Rightarrow 70\alpha = 144 - 95$$

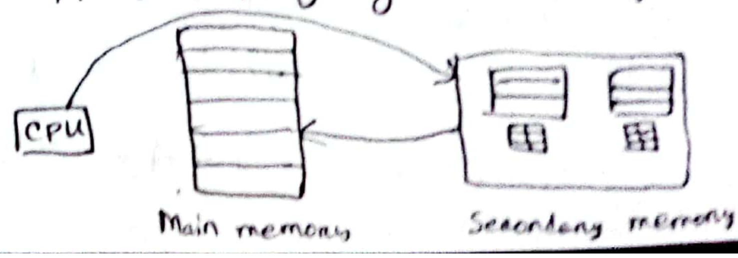
$$\therefore \alpha = \frac{144 - 95}{70}$$

$$= 0.7$$

$$= 70\% \quad \text{(Ans)}$$

(e)

(i) The users view of the memory is logical memory address. We pass the logical address to Memory Management Unit (MMU), and MMU takes the address and adds with the relocation register to find out the Physical memory address. This Physical memory address is called the main memory address. We also apply paging technique along with it.



(ii) Here,

Page size = Frame size = 8 bytes.

Main memory = 72 bytes.

For 25(11001),

page number = $25 \div 8 = 3$

offset = $25 \% 8 = 1$

$\therefore$ Physical address = $(f \times P) + \text{offset}$

$= (3 \times 8) + 1$

$= 25$ (Ans)

For 37(160101),

page number = $37 \div 8 = 4$

offset = $37 \% 8 = 5$

$\therefore$ Physical address = $(f \times P) + \text{offset}$

$= (6 \times 8) + 5$

$= 53$ (Ans)

For 23(10111), page number = $23 \div 8 = 2$

offset = $23 \% 8 = 7$

$\therefore$ Physical address = $(f \times P) + \text{offset}$

$= (7 \times 8) + 7$

$= 63$ (Ans)

$\boxed{d}$

Here, page size = 10 kB

Process size = 31110 B

So, frames required = $\left\lceil \frac{\text{Process size}}{\text{Page size}} \right\rceil$

$$= \left\lceil \frac{31110}{10 \times 1024} \right\rceil$$

$$= \lceil 3.0381 \rceil$$

$$= 4$$

As the actual frame size is fractional, there will be internal fragmentation.

$\therefore$ Internal fragmentation = $(10 \times 1024) - (10 \times 1024 \times 0.0381)$

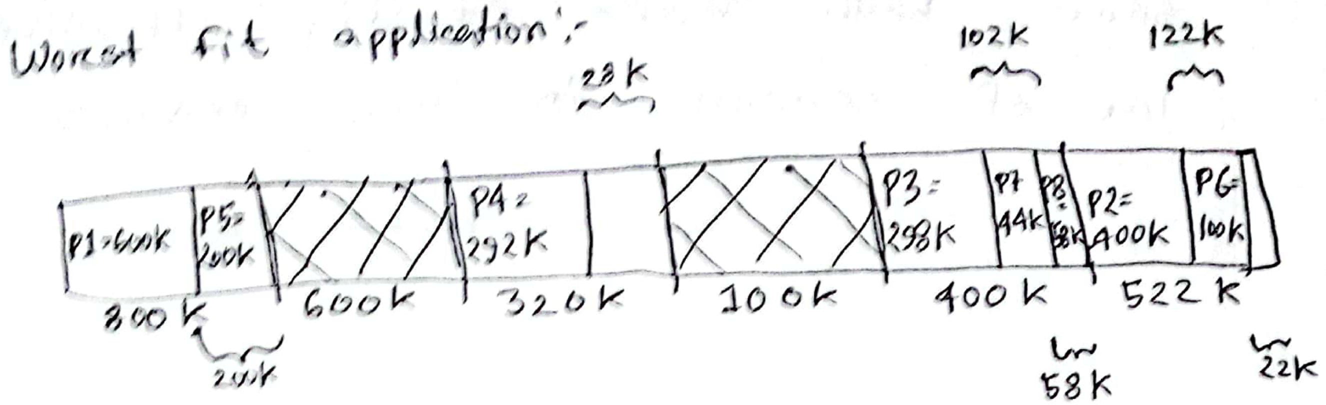
$$= 10240 - 390$$

$$= 9850 \text{ B}$$

(Ans)

(e)

Worst fit application :-

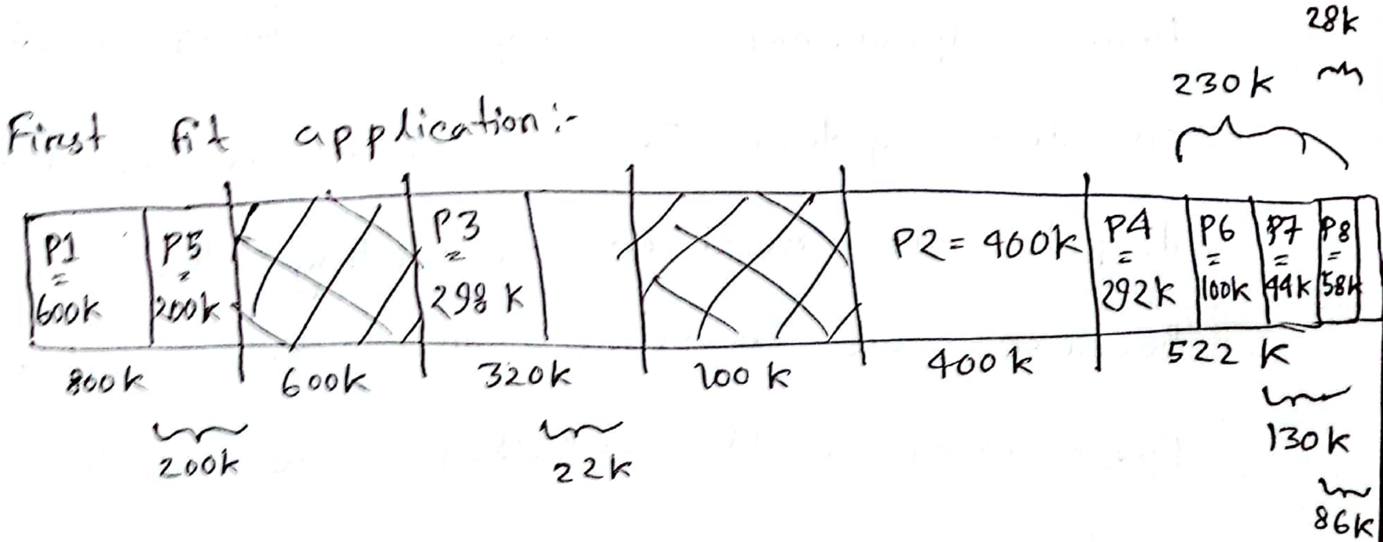


Here, Internal fragmentation = 0 B

External fragmentation = (28 KB + 22 KB)

= 50 KB

First fit application :-



Here, Internal fragmentation = 0 B

External fragmentation = (22 KB + 28 KB)

= 50 kB

As both of the algorithms can occupy all the processes and their performance is same, both make equally effective use of memory in this scenario.

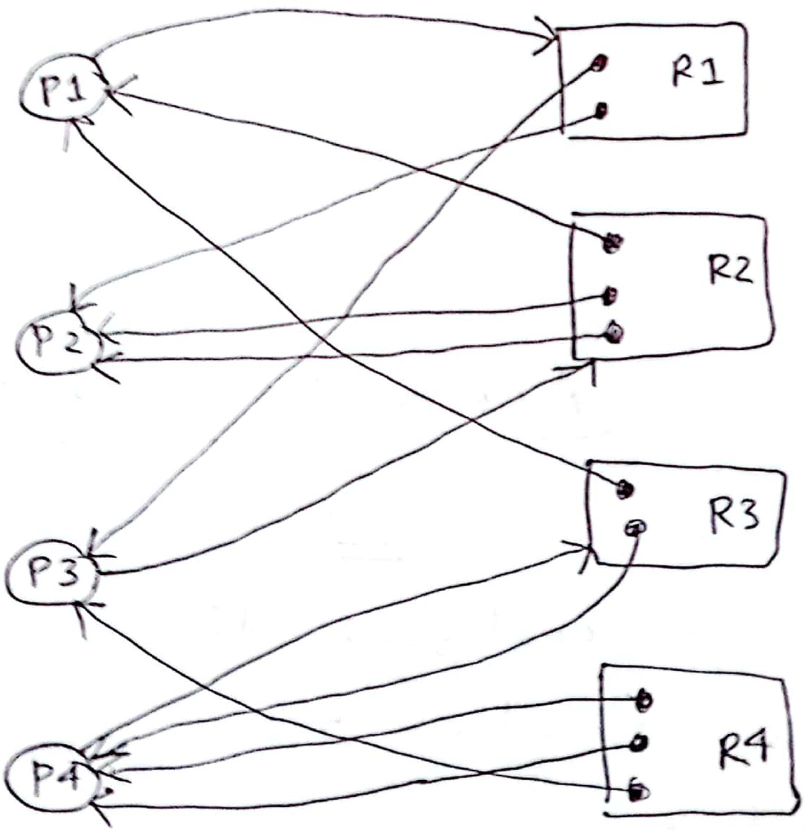
Ans to the ques no:- 2

a

Here, ignorance strategy is being applied on the system. This system is popular, despite the need of restarting the system. Because, we assume that the application programmer will develop the system in such a way that it never comes accross any deadlock situation. So, it is ensured by the application developer to design the software in a way that it never occurs any deadlock. By any chance if deadlock occurs, a simple restart can solve the problem.

Many modern day OS like UNIX use this method nowadays.

b



There are a few cycles like

P1 – R1 – P3 – R2 – P1

As, the resources have multiple instances,

so, there might be a deadlock.

The system will avoid deadlock if it processes

in the following sequence →

P2 → P1 → P3 → P4.

Ans to the ques no:- 1

$\boxed{a}$

Number of employees = 10

Maximum allocation = 5

Occupied = 2

Required = 6

Free = 3

Here, 3 processes are available, but 6 processes are waiting to be executed. So, here is a memory allocation problem to solve.

To solve this issue, we should employ process synchronization with Critical Section, so that one employee can access the internet when required, satisfying the critical section.

$\boxed{b}$

The issue is called starvation. When memory management is done without employing bounded waiting criteria, problem like this happen. To solve this issue, we should keep a bounded waiting time so that no process goes to starvation state.

c

Queue = [P2, P3, P1]

Semaphore, S = 2

RR time quantum = 9 ms

Each statement = 3 ms

CS = 2

RS = 3

Process 1	Process 2	Process 3
	wait(2) while (s<=0) → F S = 1	
		wait (1) while (s<=0) → F s = 0
wait (0) while (s<=0) → T while (s<=0) → T		
	CS 1 CS 2 signal (0)	
		CS 1 CS 2 signal (0)
while (s<=0) → T (3 times)		
	s = 1 RS 1 RS 2	
		s = 2 RS 1 RS 2
while (s<=0) → F s = 1 CS 1		
	RS 3 wait (1) while (s<=0) → TF	
		RS 3 wait (1) while (s<=0) → TF

CS 2 signal (1) S = 2		
	while $(s<=0) \Rightarrow F$ S = 1 CS 1	
		S = 0 cs 1 cs 2
RS 1 RS 2 RS 3		

(And repeat)