Final Q. Solve (18-19)

1) (a) We have stressed the need for an operating system to make efficient use of the computing hardware. When is it appropriate for the OS to forsake this principle and to "waste" resources? Why is such a system not really wasteful? [3]

Ans: An operating system usually tries to use hardware resources effecienlty. This means CPU, memory and I/O devices should not be idle. But in some case, it is better to not use all recources fully.

For example, the OS may keep some memory free. This helps to respond quickly when a new program starts. Also, sometimes extra processes run for safety. They help the system recover if somethings fails. These look like a waste, but they make the system more reliable and fast.

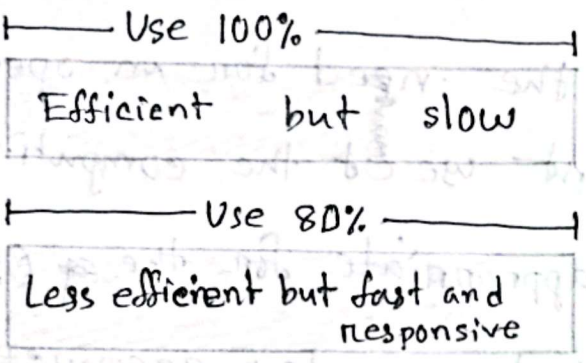


Fig: Resource Usage Vs. User Experience

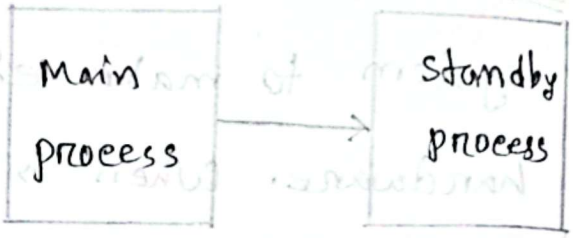


Fig: Redundancy for reliability

So, this "waste" is not bad always. Some times, it improves user experience and system stability.

(b) Why do some system store the operating system in firmware, while others store it on disk? [2]

Ans: ~~Some system store the OS in firmware, like ROM or EPROM. These use a bootstrap program~~

Some system store bootstrap program in firmware like ROM or EPROM. This is because it needs to run immediately when the system powers-on or reboots. The bootstrap program initializes the system and

loads the operating system from disk.

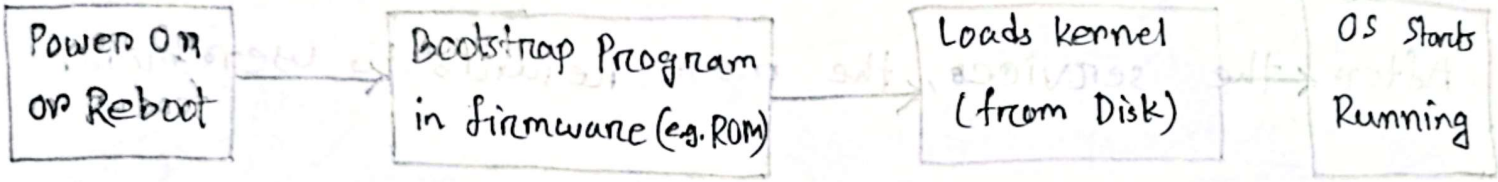
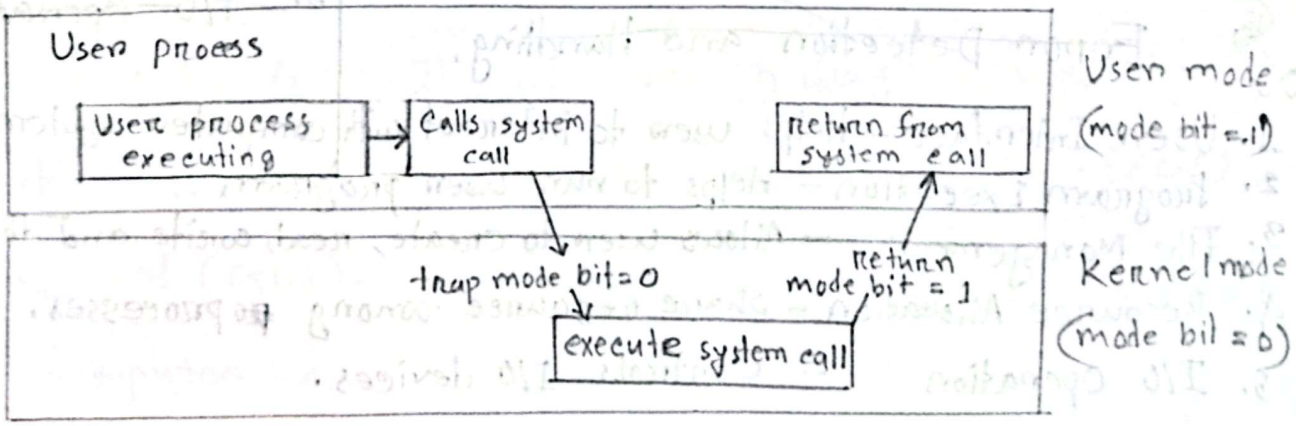


Fig: Boot Sequence.

How does the distinction between kernel mode and user mode function as a rudimentary form of protection (security) system? [3]

Ans: Operating system use dual-mode operation: User mode and Kernel mode. The hardware provides a "Mode bit" to track which mode the system is in. This helps to protect the OS and system resources from being changed by user programs.

Fig: User vs Kernel mode: Basic Protection



which are privileged and only allowed by kernel.

When a user program needs a service, it makes a system call, and it switches the mode to kernel mode. After the services, the mode returns to user mode.

(d) List five services provided by an operating system and explain how each creates convenience for user. [4]

Ans: There are several services provided by an operating system. Five services of them are given below:

~~1. Program Execution~~
~~2. File Management~~
~~3. Input / Output Devices Operations~~
~~4. Memory Management~~
~~5. Error Detection and Handling.~~

1. User Interface
2. Program Execution
3. File Management
4. Resources Allocation
5. ~~I/O Operation.~~

1. User Interface - Helps user to interact with computer system.
2. Program Execution - Helps to run user program.
3. File Management - Allows user to create, read, write and delete file
4. Resource Allocation - Share resource among processes.
5. I/O Operation - Controls I/O devices.

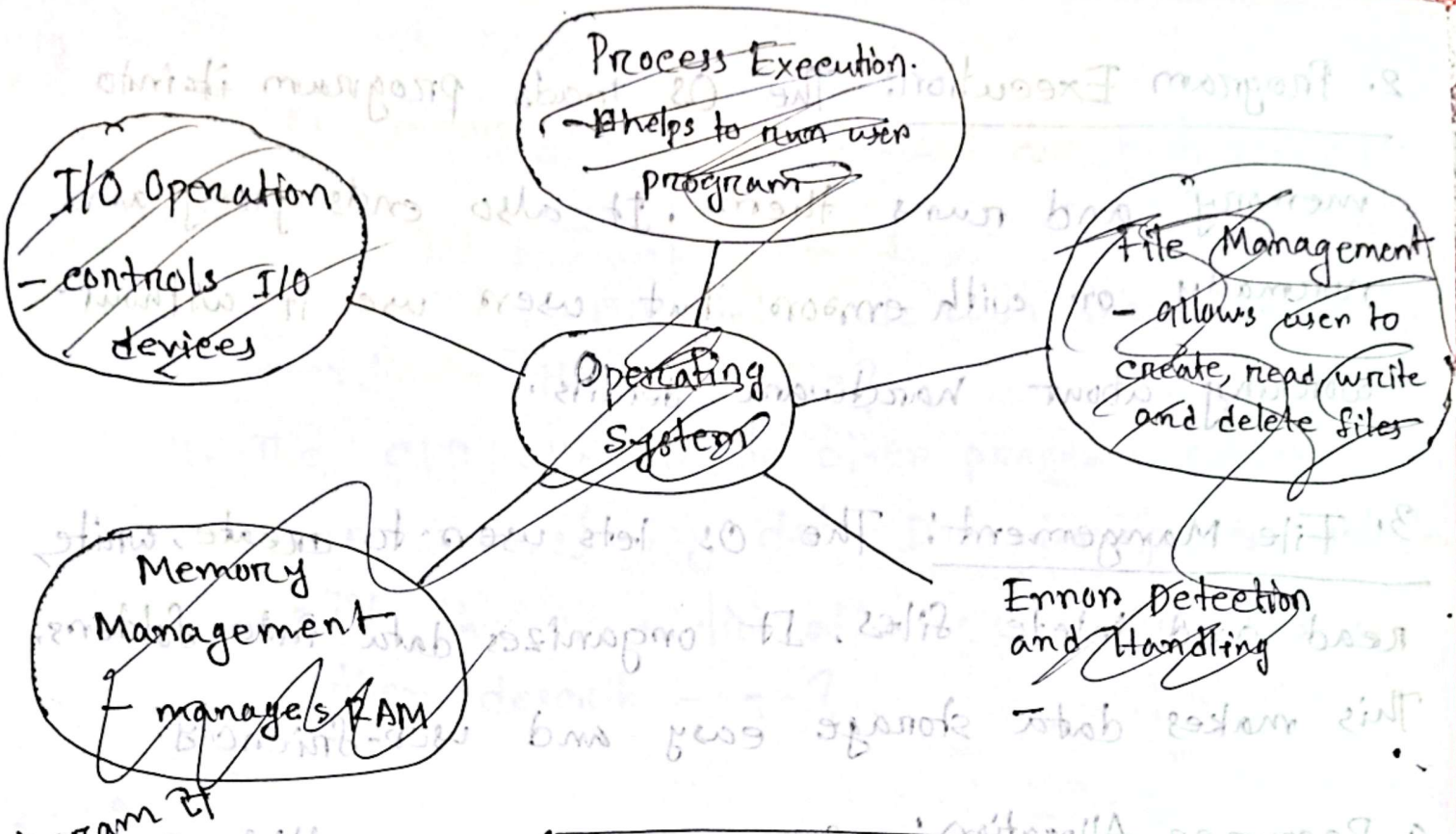
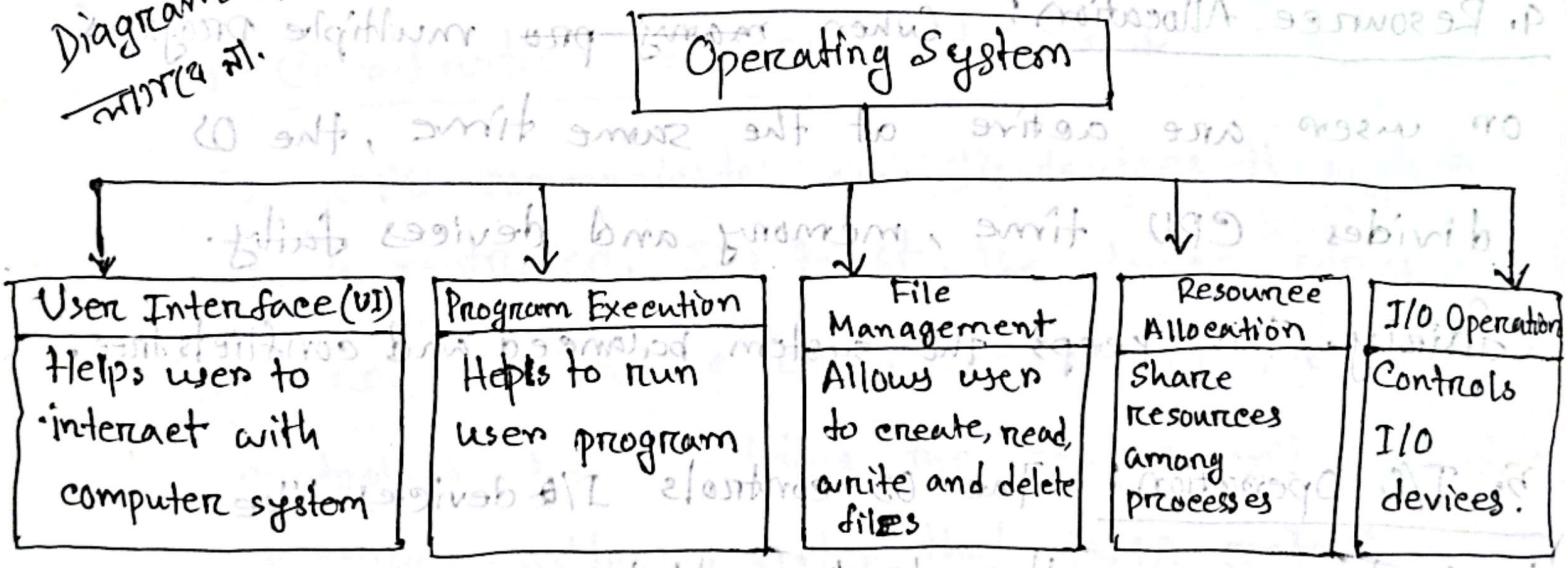


Diagram of OS.



Explain:

1. User Interface (UI): The OS gives a way for user to interact with the system by command line (CLI) or graphical (GUI). This helps user to control and manage the computer easily.

2. Program Execution: The OS loads program ~~it~~ into memory and runs them. It also ends programs normally or with errors but user use it without worrying about hardware details.

3. File Management: The OS lets users to create, write, read and delete files. It organizes data into folders. This makes data storage easy and user-friendly.

4. Resource Allocation: When ~~many pro~~ multiple programs or users are active at the same time, the OS divides CPU time, memory and devices ~~fairly~~. fairly. This keeps the system balanced and conflicts free.

5. I/O Operation: The OS controls I/O devices like keyboard, mouse, monitor ~~etc.~~ ~~by which user can give their inputs on and see their resu required results~~. helps users to give input and gets result.

(2)

(a) Direct memory access is used for high speed I/O devices ---------- [3]

i. How does the CPU interface with the device to coordinate the transfer?

ii. The CPU allowed to other program when DMA transfering data. Does the process interfere with the execution of user program? If so, then describ ----?

Ans:

i. Coordination between CPU and Devices.

The CPU communicates with I/O devices through a DMA controller. ~~Fist~~ First, the device sends an I/O request to the CPU. Then sets up the DMA controller by providing the source and destination memory address. After that, DMA controller directly transfer data from I/O devices to memory without CPU's involment.

ii. Yes, DMA can cause minor ~~interface~~ interference with user program execution, because it temporarily takes controll of system bus to

transfer data. During this time CPU slightly delaying its operation but it is better than fully blocking the CPU and user program execution.

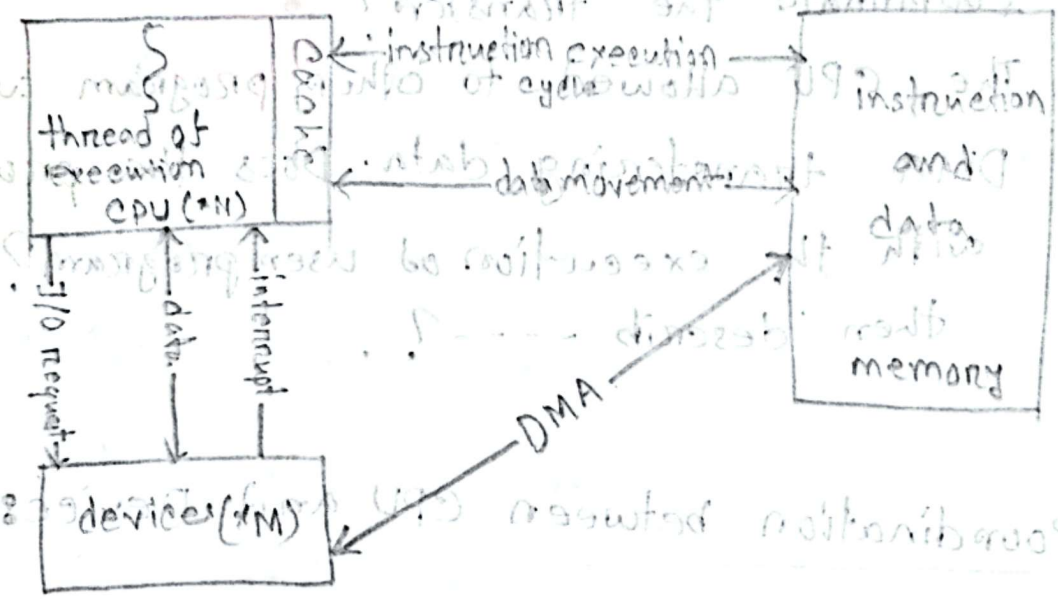


Fig: DMA Data Transfer Process

(b) What are the two essential parts of process? How is a process differ from a program? [3]

Ans: The two essential parts of process are,

i. Program code.
ii. Process context.

i. Program code: This part contains the actual code or instructions and it is called as "text section" of the process. CPU fetches and executes these

instruction step by step. Without this the process has no logic to perform.

ii. Process Context: This includes program counter, CPU register and other information that tracks execution. It is essential for multitasking and process switching. When a process is resumed, this context helps to store it as it was.

~~A program is a passive set of instruction stored on enti~~

A program is a passive entity stored on disk such as an excutable file. It doesn't perform any action on its own. A program becomes a process when it is loaded into memory and begins execution.

A process is an active instance of a program. It has its own memory, CPU state and resource. One program can create multiple processes for example several user can runs same program at the same time.

(c) প্রশ্ন অনেক বড় তাই সংক্ষেপে সমাধান:

Process	Burst time	Priority
P1	2	2
P2	1	1
P3	8	4
P4	4	2
P5	5	3

All process arrived at time = 0.

i) Draw four Gantt chart for, FCFS, SJF, non-preemtive, priority (Higher number = Higher priority), RR (quantum = 2)

ii) find a turnaround time for each process..

iii) " waiting " " " "

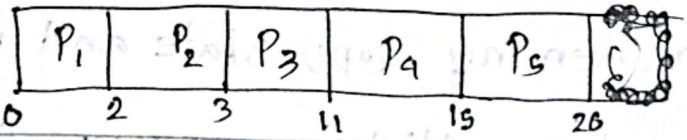
iv) " avg. " " " " and which one is best.

Ans:

(i) AT = Arrival time, BT = Burst time, CT = completion time, TAT = Turnaround time, WT = waiting Time.

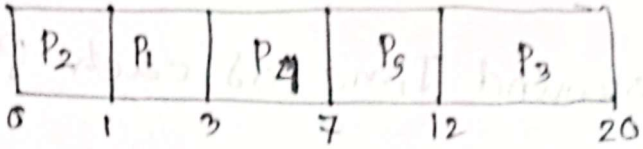
FCFS:

Gantt chart:



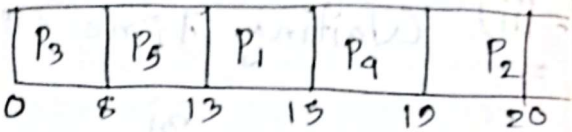
Process	AT	BT	CT	TAT	WT
P1	0	2	2	2	0
P2	0	1	3	3	2
P3	0	8	11	11	3
P4	0	4	15	15	11
P5	0	5	20	20	15
				Σ(TAT) = 51	Σ(WT) = 31

SJF: Gantt chart:



Process	AT	BT	CT	TAT	WT
P1	0	2	3	3	1
P2	0	1	1	1	0
P3	0	8	20	20	12
P4	0	4	7	7	3
P5	0	5	12	12	7
				Σ(TAT) = 33	Σ(WT) = 23

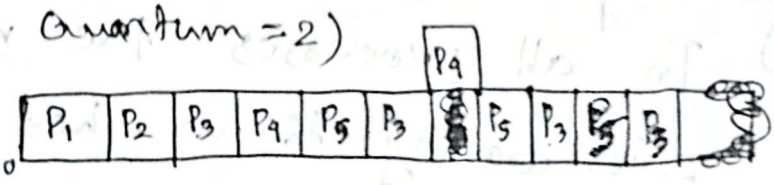
Non-preemptive Priority: Gantt chart:



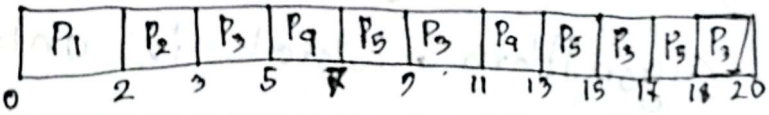
Process	Priority	AT	BT	CT	TAT	WT
P1	2	0	2	15	15	13
P2	1	0	1	20	20	19
P3	4	0	8	8	8	0
P4	2	0	4	19	19	15
P5	3	0	5	13	13	8
					Σ(TAT) = 75	Σ(WT) = 55

Round Robin (RR): (Time Quantum = 2)

Ready Queue:



Gantt Chart:



Process	AT	BT	CT	TAT	WT
P1	0	2	2	2	0
P2	0	1	3	3	2
P3	0	8	20	20	12
P4	0	4	13	13	9
P5	0	5	18	18	13
				Σ(TAT) = 56	Σ(WT) = 36.

ii) Turn around Time of each Processes given below:

$$FCFS = \frac{51}{5} = 10.20 \text{ milisec}$$

$$SJF = \frac{33}{5} = 6.6 \text{ ms}$$

$$\text{non-preemptive priority} = \frac{75}{5} = 15 \text{ ms}$$

$$RR = \frac{56}{5} = 11.2 \text{ ms.}$$

iii) Waiting Time of each process given below:

$$FCFS = \frac{31}{5} = 6.20 \text{ ms}$$

$$SJF = \frac{23}{5} = 4.6 \text{ ms}$$

$$\text{non-preemptive priority} = \frac{55}{5} = 11 \text{ ms}$$

$$RR = \frac{36}{5} = 7.2 \text{ ms.}$$

iv) In all processes the minimum average waiting time has Shortest Job First(SJF) algorithm, which is only 4.6 ms.

3(a) Define demand paging and segmentation with example. [4]

Demand Paging: Demand paging is a memory management technique where pages of a process are loaded into memory only when they are needed. If a required page is not in memory, a page fault occurs and the OS brings it from secondary storage (e.g. disk or swap space) into RAM.

Example: When we open a large PDF file, only the first few pages are loaded into memory. The rest of the file remains on disk.

As we scroll down and try to read page 20, the system sees that is not in memory and a page fault occurs. The OS then loads that specific page into RAM from disk.

Segmentation: Segmentation is a memory management scheme where a program is divided into logical units called segments (e.g. main program, functions,

array, stack). Each segment has a base and limit. Logical address has two parts: <segment number, offset>

Example: ~~Let~~ Think of a C++ program, ~~where~~ that has; code (instructions), function, stack, variable, heap, and queue etc. In segmentation each of these parts are stored in seperate memory segments. So when the entire program runs, the OS manage these segments ~~seperately~~ independently.

3(b) Draw the storage device hierarchy. [4]

Ans:

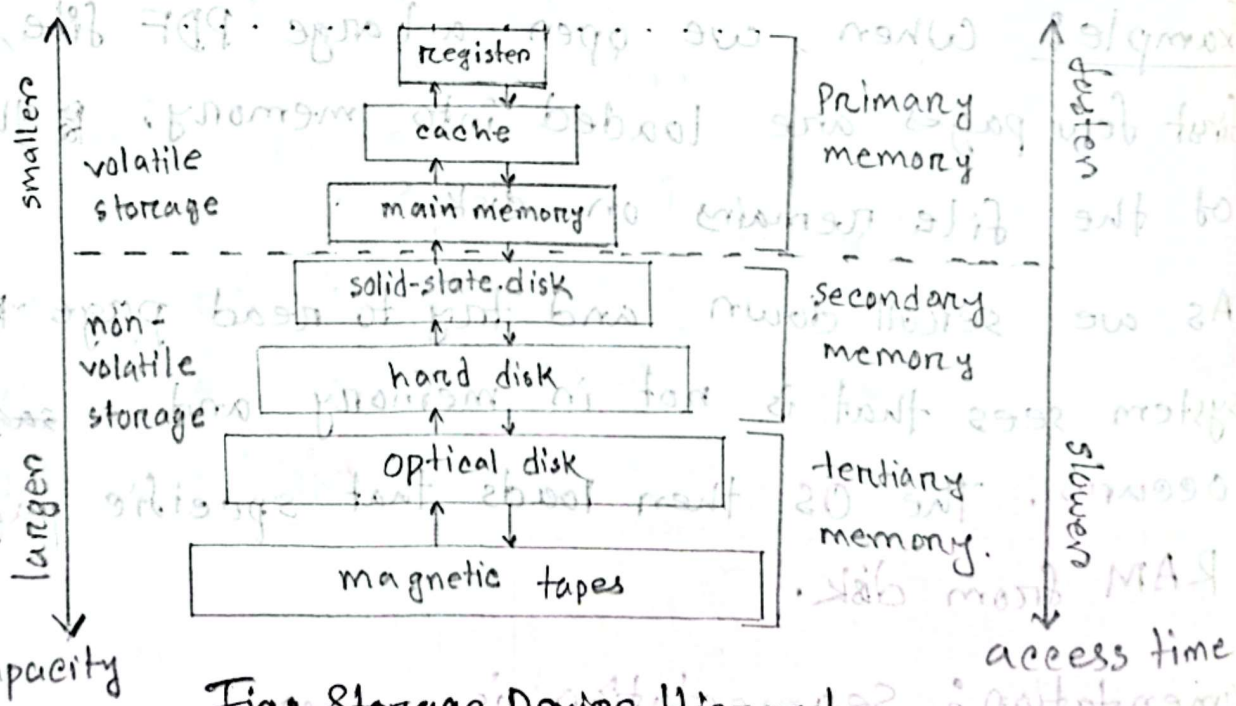


Fig: Storage Device Hierarchy

3(c) Explain different threading models with sufficient diagram? [4]

Ans: There are three different type of models exists in threading, such as,

1. Many-to-One Model:

Many user-level threads mapped to single kernel thread. ~~If one thread~~ Only one thread can access the kernel at a time. If one thread blocks, all threads are blocked. It doesn't take advantage of multiple cores.

paragraph আকারে না লিখে point আকারে লিখব।

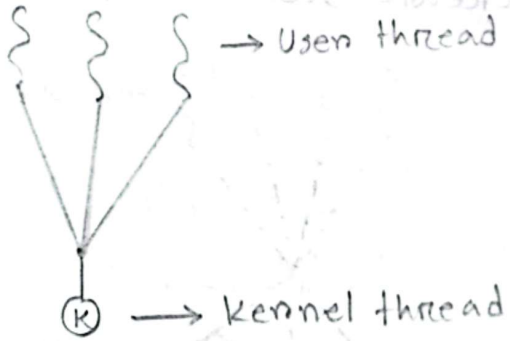


Fig: Many-to-One Model.

2. One-to-One Model:

- Each user-thread maps to a kernel thread.
- Creating a user thread also creates a kernel thread.
- Thread count may be limited due to system overhead.
- More concurrency than many-to-one.

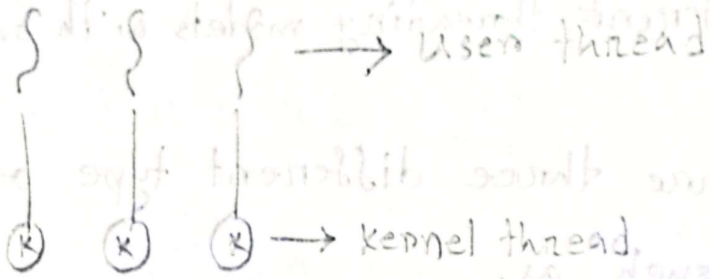


Fig: One-to-One Model.

3. Many-to-Many Model:

- Allows many user-level thread mapped to many kernel thread.
- Threads can run in parallel on multiple processor.
- Allows OS to create sufficient number of kernel threads.

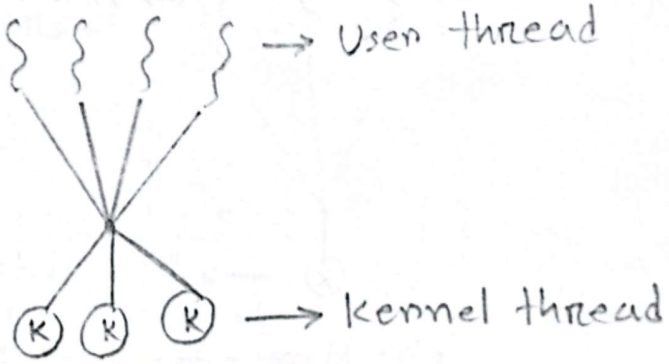
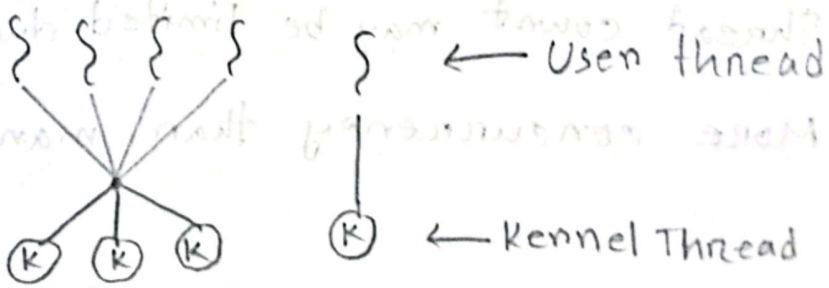


Fig: Many-to-Many Model.

4. Two-level Model:

- Similar to M:M model except that it allows a user thread to be bound to kernel thread.



4(a) Provide two programming example in which multithreading provides better performance than a single - thread solution. [3]

Ans: Two examples where multithreading perform better than single-threading given below:

1. Web Server Handling Multiple Clients:

In a single-threaded web server, client request are served one after another. this cause delays and user waits for other completion.

But in a multithreaded web server can handle each client in a seperate thread simultaneously. This reduce delays and make faster service.

2. Compiler Design:

In a single-threaded compiler, steps like reading the code, checking the structure and creating machine code are done one by one.

But a multithreaded compiler divide those work among threads, seperate → at the same time.

~~result it makes the program execution faster.~~
This parallel execution makes the compilation process faster.

9(b) Illustrate how a binary semaphore can be used to implement mutual exclution among n processes. [3]

Ans: A binary semaphore can take values 0 or 1 and is commonly used like a mutex lock. It ensure that only one process can enter the critical section at a time.

The n processes share a semaphore, mutex, initialized to 1. Each process P_i is organized as follows:

```
semaphore mutex = 1;
Process Pi;
do {
    wait (mutex);
        /* critical section */
    signal (mutex);
        /* remainder section */
} while (true);
```

4(c) ঐ প্রশ্ন ও অনেক বড় তাই চুরিনি। Race condition related, husband-wife এর টাকা তুলা ~~নিয়া~~ নিয়া গোছোর মাপের। [4]

Ans: A race condition occurs when multiple processes access and changed shared data at the same time.

Let's assume,

Initial account balance: 1000

Husband calls withdraw (500)

~~with~~ wife calls deposite (200)

Both function access and shared same variable: balance.

Race Condition possible:

Both action runs at the same time,

Husband reads balance → 1000

wife " " → 1000

Husband withdraw (500) and want to set balance = 500

wife deposite (200) " " " " = 1200

Final value is either 500 or 1200 but correct value is 700. Due to no control over execution order, ~~these~~ this scenario goes to the race condition.

To prevent this race condition we use ~~me~~ mutual exclusion (mutex) or a binary semaphore to allow

only one operation at a time can access the balance.

4(d) Question 4 এর (d) and 1 এর (b) same Question.

5(a) প্রশ্ন অনেক বড় তাই তুলি নাই। Deadlock related.
চাকরি-লেতায় আছে দুই বার লেকচার লাগে। [4]

1. What are the arguments for installing dead-lock-avoidance algorithm
11. What are the arguments against installing ....

Ans:

i. Arguments for installing deadlock-avoidance Algorithm:

Arguments	Description
Eliminates Deadlock Loss	Avoid 2 deadlock/month x10 jobs = 20j/m saved.
Saved CPU cost	\$1 loss/job x20 jobs = \$20/month saved
Utilizes Idle system capacity	Despite 10% longer execution, system has 30% idle time, so it can still run all 5000 jobs.

11. Arguments against installing deadlock-avoidance algorithm:

Argument	Description
Deadlock are infrequent	Only two deadlock/month in 5000 jobs (0.04% rate), affecting just 20 jobs. The impact is very low.
Increase Turn-around Time	Each job will take ~ 20% longer to finish on average.
Minimizing cost may not justify	Saving ~~no~~ only $20/month doesn't justify the complexity and effort of implementing ~~Bankers Algorithm~~. deadlock-avoidance Algorithm.

5(c) Here are three real life example of deadlock outside the computer system environment:

1. Traffic Deadlock: Four cars stop at a four way intersection, each awaiting for the next to move — no one can go.
2. Dining Table Deadlock: Four people share five forks. If each person picks up one fork and waits for second ~~one~~ — no one can eat.

3. Elevator Deadlock: Two people hold elevator doors on different floor, each waiting for other to release it - elevator can't move.

5(b) প্রশ্ন অনেক বড়। Banker's algorithm এর প্রশ্ন।

Five processes P0 through P4 and resource instance A(10), B(5), C(7).

i) content of Need?

ii) define safe and unsafe? Is this system safe state if so, show a safe order - - - -

iii) If P0 arrives for (0, 2, 0), can granted immediately

Ans:

Process	Max			Allocation			Need			Available		
	A	B	C	A	B	C	A	B	C	A	B	C
P0	7	5	3	0	1	0	7	4	3	3	3	2
P1	3	2	2	2	0	0	1	2	2	5	3	2
P2	9	0	2	3	0	2	6	0	0	7	4	3
P3	2	2	2	2	1	1	0	1	1	7	4	5
P4	4	3	3	0	0	2	4	3	1	10	4	7
				Sum= 7	2	5				10	5	7

< $\leq$ P1, P3, P4, P2, P0 >

i)

The content of the matrix Need is defined.

Need = Max − Allocation

∴

Process	Need		
	A	B	C
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

ii) The system is in a safe state since the sequence $\langle P_1, P_3, P_4, P_2, P_0 \rangle$ satisfies safety criteria.

iii) Request is will be granted or not checked by

= Request ≤ Available,

P_0 arrives for $(0, 2, 0) \leq (3, 3, 2)$ = true.

So P_0 request granted immediately.

6(a)

Given that, [Mark-5]

Total processes = 4

Total resources = 3

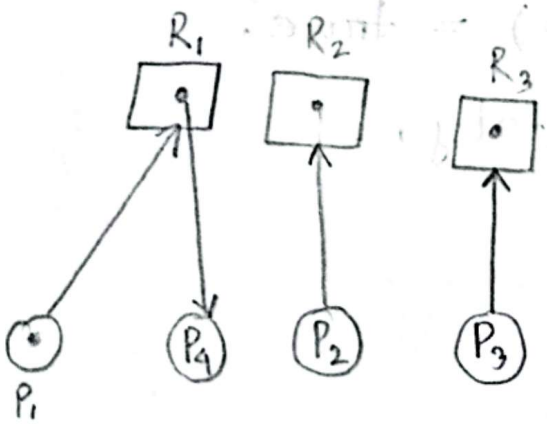
Three process has each one Request Edge to one of the resources (you decide which one).

i. Draw the resource allocation graph.

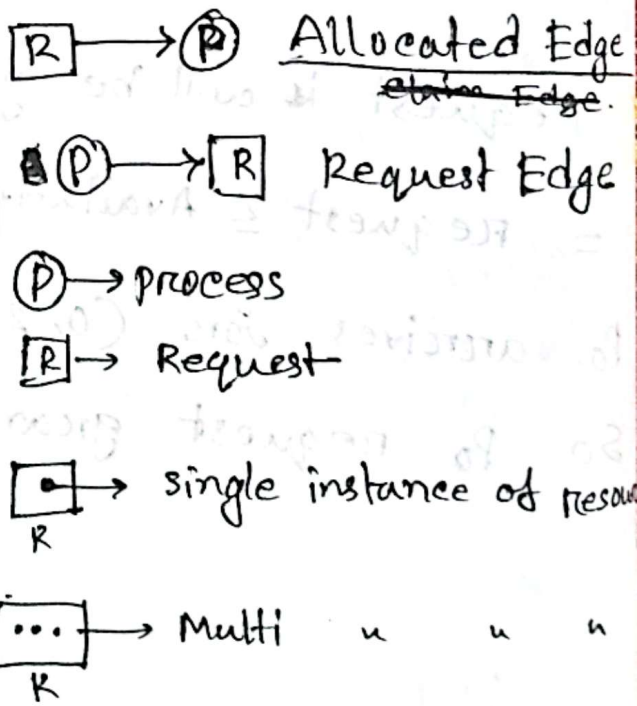
ii. Is the system you draw in safe/unsafe/deadlock.

Ans:

i)



Note:



ii)

My graph that I drew is currently unsafe state but don't have a deadlock.

Deadlock occurs when the following conditions are fulfilled:

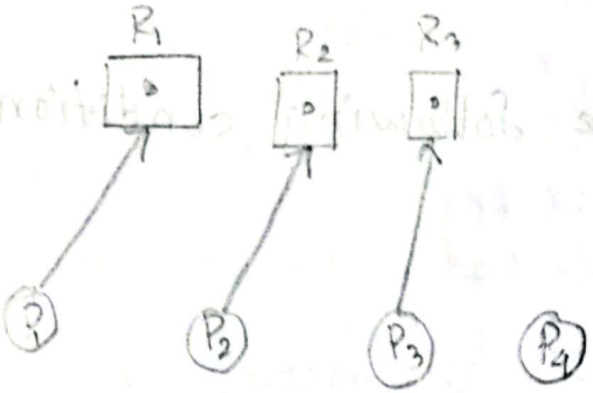
1. Mutual Exclution
2. Hold and wait
3. No preemption
4. Circular wait.

In my graph, $P_1 \rightarrow R_1 \rightarrow P_4$ implies a hold-and wait situation. But there is no circular wait. So, my graph is not a deadlock.

However R_2 and R_3 are not Allocated yet. So no process can complete and release to help others.

So, the conclusion is the graph that drew is at a unsafe state but not deadlock.

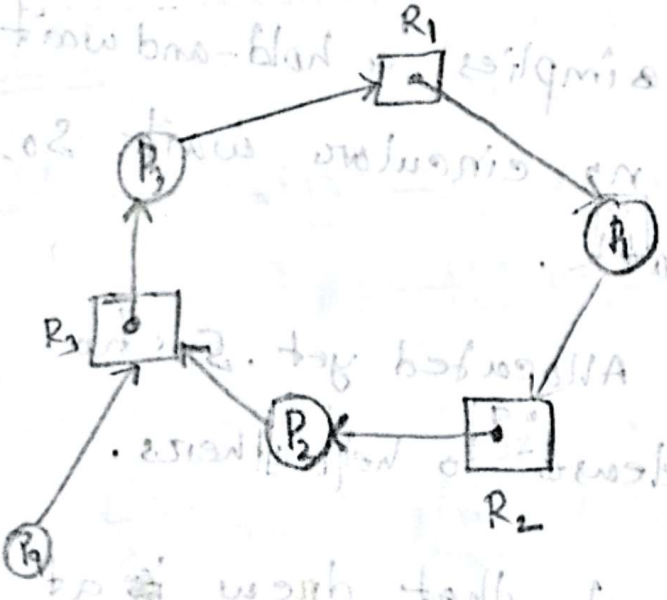
প্রশ্ন (a) টা যদি এভাবে করলাম তাহলে safe state আসবে,



Safe state.

~~Every Resource are requested~~
No cycle, all resources are ~~currently free~~ requested. and can b allocate easily. Then execuit and release. so other can proceed.

Deadlock:



Deadlock state.

Each resource has one instance → Mutual Exclution

Each process hold one resource and wait for another one.

Resource can't take forcefully and a circular wait detected,

$P_1 \rightarrow R_2 \rightarrow P_2 \rightarrow R_3 \rightarrow P_3 \rightarrow R_1 \rightarrow P_1$

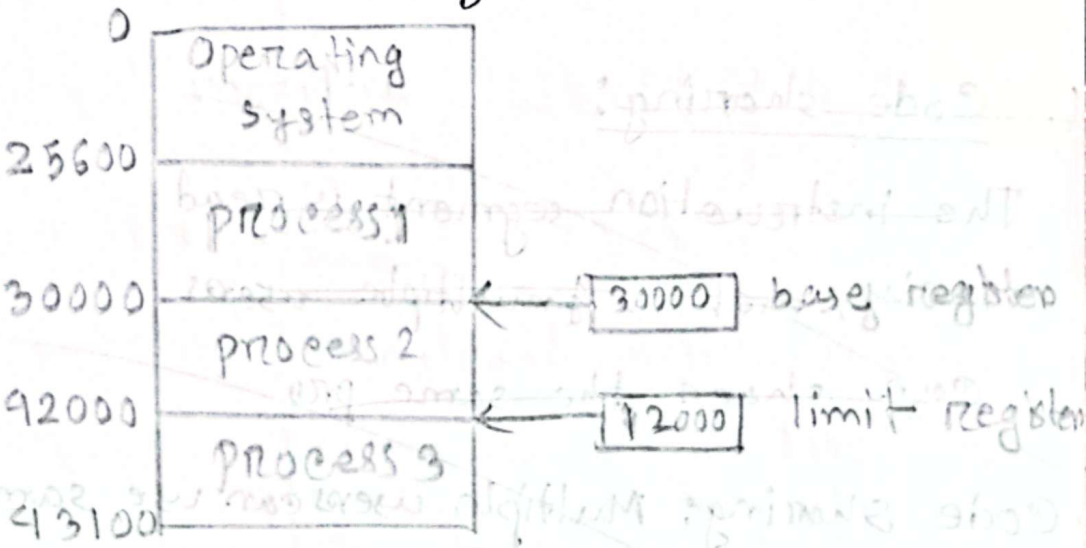
So, it fullfill all condition and deadlock

6(b)

[Mark - 3]

Ans:

Seperate base-limit register architecture:



Advantage of this scheme:

1. Code can be sheared between users (save memory)
2. Protects code from being changed (read-only)
3. Keeps code and data seperate, reducing errors.

Disadvantage of this scheme:

1. Needs two base-limit register pairs (complex hardware)
2. Code can't modify its own code, even if needed.
3. Memory mangement becomes harder.

6(a) What is paging? Consider a logical address space 64 pages of 1024 words each, mapped into a physical memory 32 frames. [4]

i. How many bit in logical address

ii. " " " " physical "

Ans: Paging is a memory management technique where the logical address is divided into fixed size blocks called pages and the physical address also divided into same size blocks called frames.

Given that,

Logical address space = 64 pages

Each pages = 1024 words.

physical memory = 32 frames.

i) We know,

Logical address = page number + offset.

Number of pages = $\log_2(64)$ = 6 bits

page offset = $\log_2(1024)$ = 10 bits.

∴ Logical address = (6+10) bits = 16 bits.

11) We know,

physical address = frame number + ~~frame~~ offset

Number of frames = $\log_2(32)$ = 5 bits

offset = $\log_2(1024)$ = 10 bits,

∴ Physical address = (5+10) bits = 15 bits.

7(a) প্রশ্ন বড়। তাই ~~ছবি~~ ছবিটি সংক্ষেপে নিচে ↓

six partition = 300KB, ~~250KB, 200~~ 600KB, 350KB, 200KB, 750KB, 125 KB (in order).

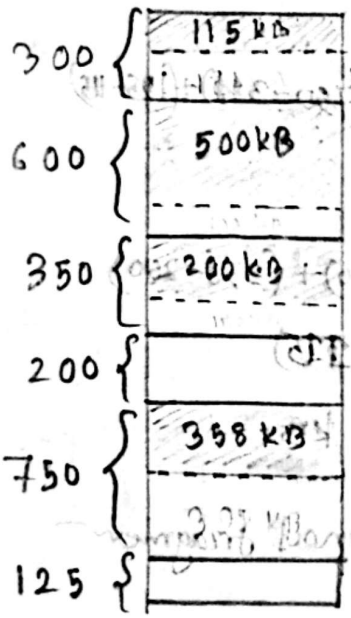
processes = 115KB, 500KB, 358KB, 200KB and 375 KB (in order)

~~Rank~~ fit the process using first fit, best fit and worst fit and Rank the algorithms.

Ans: Given, Processes,

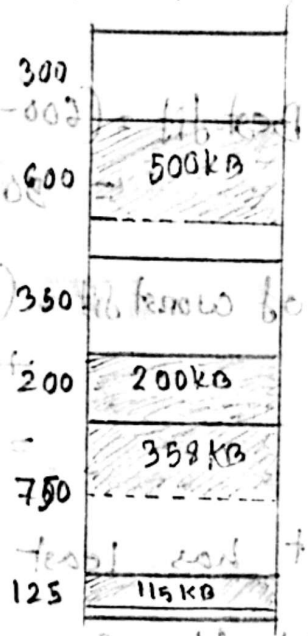
$P_1 = 115$ KB $P_4 = 200$ KB
$P_2 = 500$ KB $P_5 = 375$ KB
$P_3 = 358$ KB

First Fit



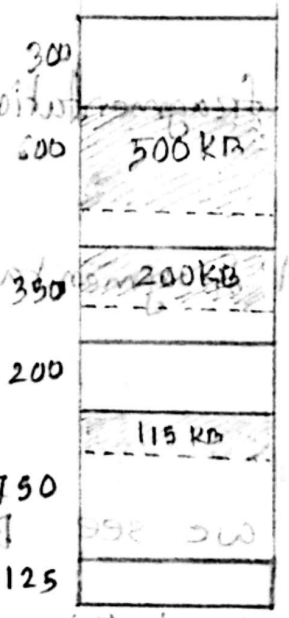
$P_5 = 375$ KB can't allocate.

Best Fit



$P_5 = 375$ KB can't allocate

Worst Fit



$P_3 = 358$ and P_5 can't allocate.

~~Memory waste of First Fit = (300 − 115) + (600 − 500) + (350 − 200) + 200 + (750 − 358) + 125~~

~~= 1152 KB~~

~~Memory waste of Best Fit = 300 + (600 − 500) + 350 + (750 − 358) + (125 − 115)~~

~~= 1152 KB~~

~~Worst~~ Memory waste of worst fit = ~~(300 + (600 − 500) + (350 − 200) + 200 + (750 − 115) + 125~~

~~= 1510 KB~~

Internal fragmentation of First Fit = (300-115) + (600-500)
+ (750-356) + (350-200)
= 827 KB.

Internal fragmentation of Best fit = (600-500) + (750-356) + (125-115)
= 502 KB

Internal fragmentation of worst fit = (600-500) + (950-200)
+ (750-115)
= 885 KB.

Since we see Best Fit has least internal fragmen
then First Fit and lastly Worst fit, so
the ranking of algorithm would be:

Best Fit > First Fit > Worst Fit.

প্রশ্নে বলা নেই 7(a) Fixed partition নাকি
variable partition তাই দুই রকম ভাবেই উত্তর
করছি। যাতে যেটাই বোক লিখে আসতে পারি।

$P_1 = 115$

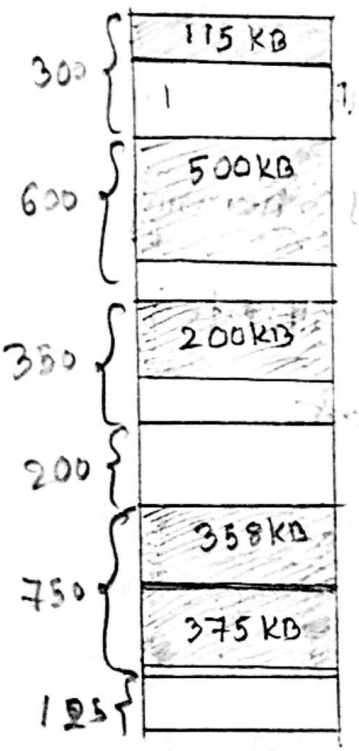
$P_2 = 500$

$P_3 = 358$

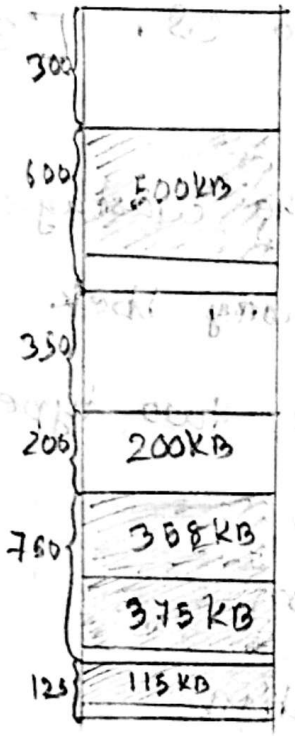
$P_4 = 200$

$P_5 = 375$

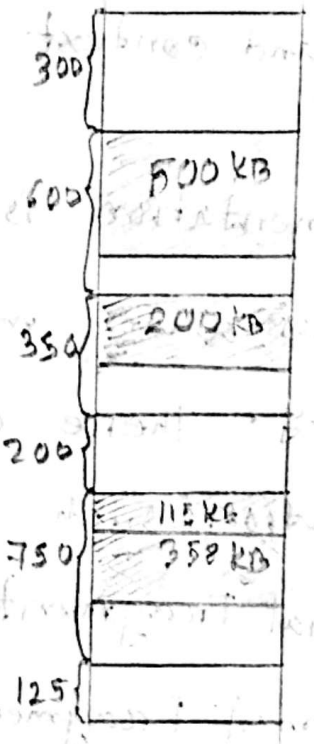
First Fit



Best Fit



Worst Fit



P_5 = 375KB can't allocate.

~~Internal fragmentation of First Fit~~ waste of memory of First fit $= (300-115)+(600-500)+(350-200)+(750-(358+375))+125+200$

$= 777$ KB

~~If of Best Fit~~ " " " Best Fit $= (600-500)+(750-(358+375))+(125-115)+350+300 = 777$ KB

~~If of worst Fit~~ " " " worst Fit $= (600-500)+(350-200)+(750-(115+358))+125+200+300 = 1152$

So Rank of algo --- ~~Best Fit > First Fit > worst Fit~~

First Fit = Best Fit > worst Fit.

7(b) What is Fragmentation? explain different types and context into OS. [3]

Ans: Fragmentation is the wastage of memory that occurs when memory block are allocated inefficiently. there are two types of fragmentation, such as,

1. Internal Fragmentation
2. External Fragmentation.

Internal Fragmentation:

It occurs when a fixed-size partition is allocated to a process and some space remain unused.

Here, P_2 use only 500KB of 800KB and rest of it is unused, which is internal fragmentation.

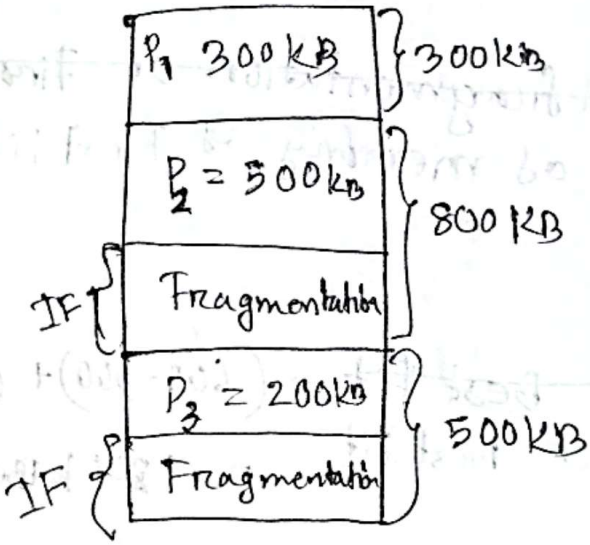
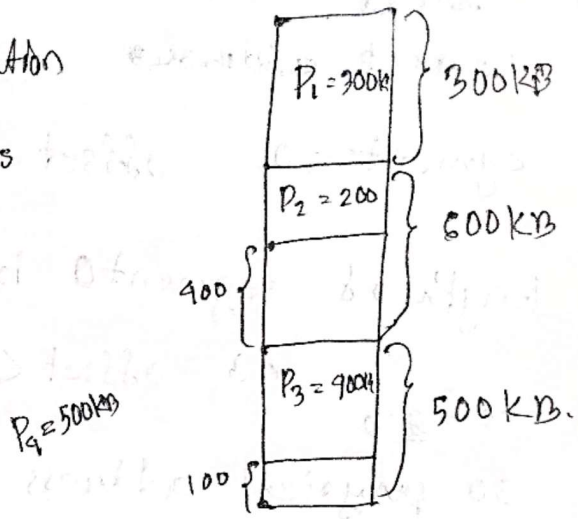


Fig: Internal Fragmentation

External Fragmentation:

It happens in variable size allocation when free memory space exists to satisfy a request but they are not contiguous.



Here, P_4 needs 500KB. we see that 500KB space is ~~avai~~ ~~exis~~ exists but at different partition which is called external fragmentation.

Fig: External Fragmentation

7(c) what is fragmentation? consider the segment table — — — — — what are the physical address for the following logical address — — — — — ? [5]

Ans: Fragmentation ছি 7(b) তে বিস্তারিত আছে।

We know, physical address = Base + offset

if offset < length.

(i) 0, 430

~~Logical addresses~~

Segment = 0, offset = 430

length of segment 0 is 600

and offset < length

Base = 210

so physical address = 210 + 430 = 640

(ii) 1, 10

segment = 1, offset = 10

length of segment 1 = 14 and offset < length

Base = 2300

So, physical address = 2300 + 10 = 2310

(iii) 2, 500

segment = 2, offset = 500

~~length of~~ ~~but~~ length of segment 2 = 100

∴ offset ≯ length

So, Address out of bound.

(iv) 3. 400

segment 3, offset = 400

length of segment 3 = 158096

and offset < length

Base = 1327

physical Address = 1327 + 400 = 1727.

8 (a) [Marks - 6]

(i) Least Recently Used (LRU) Replacement:

Given page reference string: check for frame 3 and 4.

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

string	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
F_1	1	1	1	4	4	4	5	5	5	1	1	1	7	7	7	2	2	2	2	2
F_2		2	2	2	2	2	2	6	6	6	6	3	3	3	3	3	3	3	3	3
F_3			3	3	3	1	1	1	2	2	2	2	2	6	6	6	1	1	1	6
	M	M	M	M	H	M	M	M	M	M	H	M	M	M	H	M	M	H	H	M

Number of fault = 15

string	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
F_1	1	1	1	1	1	1	1	1	1	1	1	1	1	6	6	6	6	6	6	6
F_2		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
F_3			3	3	3	3	5	5	5	5	5	3	3	3	3	3	3	3	3	3
F_4				4	4	4	4	6	6	6	6	6	7	7	7	7	1	1	1	1

Number of fault = 10

Optimal Replacement:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

check for Frame 3 and 4:

string	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
F_1	1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3	3	3	3	6
F_2		2	2	2	2	2	2	2	2	2	2	2	7	7	7	2	2	2	2	2
F_3			3	4	4	4	5	6	6	6	6	6	6	6	6	6	1	1	1	1
H/M	M	M	M	M	H	H	M	M	H	H	H	M	M	H	H	M	M	H	H	M

Number of Fault = 11

string	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
F_1	1	1	1	1	1	1	1	1	1	1	1	1	7	7	7	7	1	1	1	1
F_2		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
F_3			3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
F_4				4	4	4	5	6	6	6	6	6	6	6	6	6	6	6	6	6
H/M	M	M	M	M	H	H	M	M	H	H	H	H	M	H	H	H	M	H	H	H

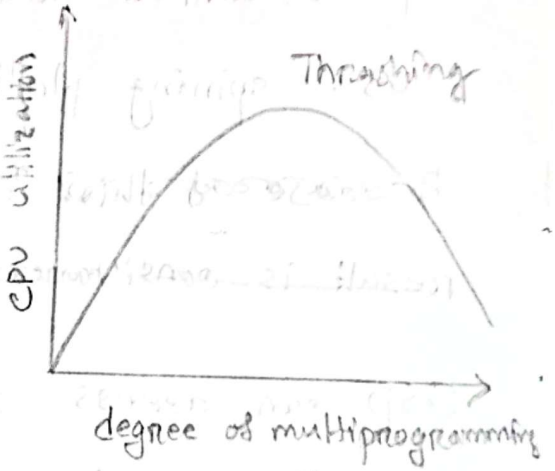
Number of fault = 8

8(b) [Marks - 3]

Thrashing: Thrashing is a condition when the system is spending a major portion of its time servicing the page fault, but the actual processing done in very short time.

The main causes of thrashing are,

1. High degree of multiprogramming
2. lack of frames
3. Page replacement policy.



RAID: RAID is a data storage tecnology that combines multiple physical disk drives into one logical unit, ~~This combination improves data performance,~~ ~~It provides~~ This combination provides,

1. Data Redundancy (Backup)
2. Increased read/write speed.
3. ~~Data~~ Increased data performance.

8(c) Explain why SSDs often use an FCFS disk scheduling algorithm. [3]

Ans: often use the FCFS disk scheduling algorithm because they SSDs (Solid-State Disk) don't have no moving parts, unlike traditional hard disk (HDDs) which use spining platter and read/write heads. ~~Because of this. That For this reason, it's result is consistance and fast access time.~~

SSD can access any memory cell instantly. So, seek time and rotational delay don't exists. In SSDs, there aren't using any complex algorithm like SSTF or SCAN which are reduce the ~~speed of read/write~~. movement of head but that is don't exist in SSDs.

SSDs use ~~the~~ FCFS (First Come-First Served) type algorithm which is simple and fair. It handles request in the order they come with low overhead.