PAGE No.
DATE / /

Deadlock characterization.

1) Mutual exclusion :- only one process at a time can use resource
2) Hold n wait :- Process is holding one resource and waiting to hold another resource held by other process
3) No preemption :-
4) Circular wait :- if there a 3 process given P1, P2, P3 so, P1 is waiting for the resources held by P2.
P2 is waiting for the resources held by P3.
P3 is waiting for the resources held by P1.

* Example of Banker's Algorithm.

- 5 processes P_0 through P_4;
  3 processes types:
  A (10 instances), B (5 instances), and C (7 instances)

- Snapshot at time T_0:

	Allocation	Max	Available	Need
	A B C	A B C	A B C	
P_0	0 1 0	7 5 3	3 3 2	7 4 3
P_1	2 0 0	3 2 2		1 2 2
P_2	3 0 2	9 0 2		6 0 0
P_3	2 1 1	2 2 2		0 1 1
P_4	0 0 2	4 3 3		4 3 1

→ To prevent the deadlock in the system, we have to find the safety sequence of execution.

How will you calculate the need.

PAGE No.
DATE / /

o) we have to apply following two algorithm for Banker's

→ 1) Safety algorithm
2)

if need <= available
then
execute process
new_available = available + allocation
else.
do not execute go forward and execute next process.

need = max - allocation.

→ 2) Reques Resources - Request algorithm.

Condi^n

1. Request ≤ need
2. if Request ≤ Available.
if both Condi^n are true then

Available = Available - Request_i;
Allocation
Available = Allocation + Request_i;
Need_i = Need_i - Request_i;

o) Need of P0 <= available
743 <= 332 P0 will not execute.

o) Need of P1 <= 332 P1 will execute.
122 <= 332
new_available = 332 + 200 (allocation of P1) = 532

o) Need of P2 <= 532 P2 will not Execute.
600 <= 532

PAGE No
DATE / /

•) Need of P3 <= 5 3 2
0 1 1 <= 5 3 2 P3 will execute.
new-available = 5 3 2 + 2 1 1 (allocation of P3) = 7 4 3

•) Need of P4 <= 5 3 2 7 4 3
4 3 1 <= 7 4 3
new-available = 7 4 3 + 0 0 2 (allocation of P4) = 7 4 5
P4 will execute.

•) Need of P0 <= 7 4 5
7 4 3 <= 7 4 5
new-available = 7 4 5 + 0 1 0 (allocation of P0) = 7 5 5
P0 will execute.

•) Need of P2 <= 6 0 0
6 0 0 <= 7 5 5
new-available = 7 5 5 + 3 0 2 (allocation of P2) = 1 0 5 7
P2 will execute

Safety sequence for given problem is <P1, P3, P4, P0, P2>

Q. if Process P1 request 1 0 2 number of resources, can this sequence will be granted.

request ≤ need
1 0 2 ≤ 1 2 2 (outcome is true)

request ≤ available
1 0 2 ≤ 3 3 2 (outcome is true)

Available = Available - request;
= 3 3 2 - 1 0 2
= 2 3 0

Allocation = Allocation + request ;

= 200 + 102

= 302

Need = Need - request ;

= 122 - 102 = 020

if two Conditions are true then P_i request will be granted.

check the request ≤ Available (that is (102) ≤ (332)) true.

	Allocation	Max	Need	Available
	ABC	ABC	ABC	ABC
P_0	010	753	743	230
P_1	302	322	020	
P_2	302	902	600	
P_3	211	222	011	
P_4	002	433	431	

⇒ Executing safety algorithm show that sequence < P_1, P_3, P_4, P_0, P_2 > satisfy safety requirement.

- Can request for (3 3 0) by P_4 be granted ?

⇒ Can request for (0,2,0) by P_0 be granted ?

i) Request ≤ need

330 ≤ 431 - true

ii) Request ≤ need

020 ≤ 743 - true

example 2

Q.5

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

Q 1) How many

Total no of instances / resources.

A = 9, B = 13, C = 10, D = 11

Q 2) Calculate Need matrix

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

Q 3) Safety algorithm / Sequence

- need of P_0 <= Available.

  2011 <= 3211 — 4001 — P_0 will execute.

new-available = 3211 + 1750 (allocation of P_0) = 4961 7211

- need of P_1 <= Available.

  0650 <= 4961 7212 — not

new available = 7212 + 1100 = 8312 — P_1 will execute

Need of P2 <= available.

1102 <= 7212 P2 will execute.

new_available = 7212 + 1254 = 8466

Need of P3 <= available

1020 <= 8466 P3 will execute

new_available = 8466 + 0633 = 81099

Need of P4 <= available.

1444 <= 81099 P4 will execute

new_available = 81099 + 0212 = 8 12 10 12

Need of P1 <= available

0650

2011 <= 8 12 10 11 P1 will execute

new_available = 8 12 10 11 + 1 1 0 0 = 9 13 10 11

=> Safety sequence for given problem <P0, P2, P3, P4, P1>

— x — o x — x — x — x

	Allocation Max	Max Allocation	Available
→	A B C D	A B C D	A B C D
P0	0 2 1 2	0 3 2 2	2 5 3 2
P1	1 1 0 2	2 7 5 2	
P2	2 2 5 4	2 3 7 6	
P3	0 8 1 2	1 6 4 2	
P4	2 4 1 4	3 6 5 8	

→ i) Apply Banker's algorithm

ii) what is Contains of need matrix

iii) find the safety sequence

iv) if the request from process P1 arrives for 1321 Can the request be granted immediately.

→ 1) resources :-

A = 7 , B = 17 , C = 11 , D = 16

→ 2) Need of matrix :-

	Allocation				Max				Available				Need			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	
P_0	0	2	1	2	0	3	2	2	2	5	3	2	0	1	1	0
P_1	1	1	0	2	2	7	5	2					1	6	5	0
P_2	2	2	5	4	2	3	7	6					0	1	2	2
P_3	0	3	1	2	1	6	4	2					1	3	3	0
P_4	2	4	1	4	3	6	5	8					1	2	4	4

→ 3) Safety sequence.

Need of P0 <= available.

0110 <= 2532 P0 will execute

new_available = 2532 + 0212 = 2744

Need of P1 <= available

2 1650 <= 2744 P1 will not execute

Need of P2 <= available.

0122 <= 2744 P2 will execute

new_available = 2744 + 2254 = 4998

Need of P3 <= available

1330 <= 4998 P3 will execute

new_available = 4998 + 0312 = 5310

Need of P4 <= available

1244 = 4 12 10 10 P4 will execute.

new_available = 4 12 10 10 + 2414 = 6 16 11 14

Need of P1 <= available

1650 <= 8 16 11 14

P1 will execute

new_available = 8 16 11 14 + 2650 = 9 6 16 11 14 + 1102 = 8 7 17 11 16

→ Safety sequence for this problem is <P0, P2, P3, P4, P1>

→ iv) request from process P1, arrives for 1321

User - resources - request algorithm.

i) Request ≤ need

1321 ≤ 1650 (outcome is not true)

Request will not be granted by for P1

Dead-lock Prevention Algorithm

Ex

Process	Allocation	Max	Available
	A B C	A B C	A B C
P0	1 1 2	4 3 3	2 1 0
P1	2 1 2	8 2 2	
P2	0 2 0	4 4 2	
P3	0 6 3	2 6 3	
P4	1 1 2	2 2 3	

PAGE No.
DATE / /

Answer the following question:

i) Determine the total amount of resources of each type
ii) what is the content of need matrix?
iii) Determine the the system is in the safe state using safety algorithm
iv) if a request from Process P1 arrives for [1,1,0] Can the request be granted immediately.

---

* Example of Detection Algorithm

- five processes P_0 through P_4; three resources type A (7 instances), BC 2 instances), C (6 instances.

- Snapshot at time T_0:

	Allocation	Request	Available
	ABC	A BC	A B C
P_0	0 1 0	0 0 0	0 0 0
P_1	2 0 0	2 0 2	
P_2	3 0 3	0 0 0 → 0 0 1 for new table.	
P_3	2 1 1	1 0 0	
P_4	0 0 2	0 0 2	

→ work = available = 0 0 0
is request P_0 ≤ available P_0
000 ≤ 010 yes ✓ P_0 executed
work = work + allocation P_0 = 000 + 010 = 010

→ work = available = 0 1 0
is request P_1 ≤ available P_1
202 ≤ 0 0 0 No ✗ P_1 not executed

→ work = available = 0 1 0

is request $P_2 \leq$ available P_2

$0\,0\,0 \leq 0\,1\,0$ true P_2 will execute

→ work = work + allocation of P_2

$= 0\,1\,0 + 3\,0\,3$

$= 3\,1\,3$

→ work = available = 3 1 3

is request $P_3 \leq$ available P_3

$1\,0\,0 \leq 3\,1\,3$ true P_3 will execute

work = 3 1 3 + 2 1 1

$= \overset{5}{6}\,2\,4$

→ work = available = $\overset{5}{6}\,2\,4$

is request $P_4 \leq$ available P_4

$0\,0\,2 \leq \overset{5}{6}\,2\,4$ true P_4 will execute

work = $\overset{5}{6}\,2\,4 + 0\,0\,2$

$= \overset{5}{6}\,2\,6$

→ work = available = 6 2 6

is request $P_1 \leq$ available P_1

$2\,0\,2 \leq 6\,2\,6$ true P_1 will execute

work = $\overset{5}{6}\,2\,6 + 2\,0\,0$

$= 7\,2\,6$

→ for avoiding deadlock follow this sequence

$< P_0, P_2, P_3, P_4, P_1 >$

following previous table by changing (Request P2 = 001)

→ By changing this process, only one process will execute rest of the all processes will not execute. then it will create a deadlock.

* Recovery from Deadlock : Resources Preemption.

1) selecting a victim :-
Victim process from which we can take away resources and allocate to another process.

2) Rollback :-
return to some safe state, restart process for that state

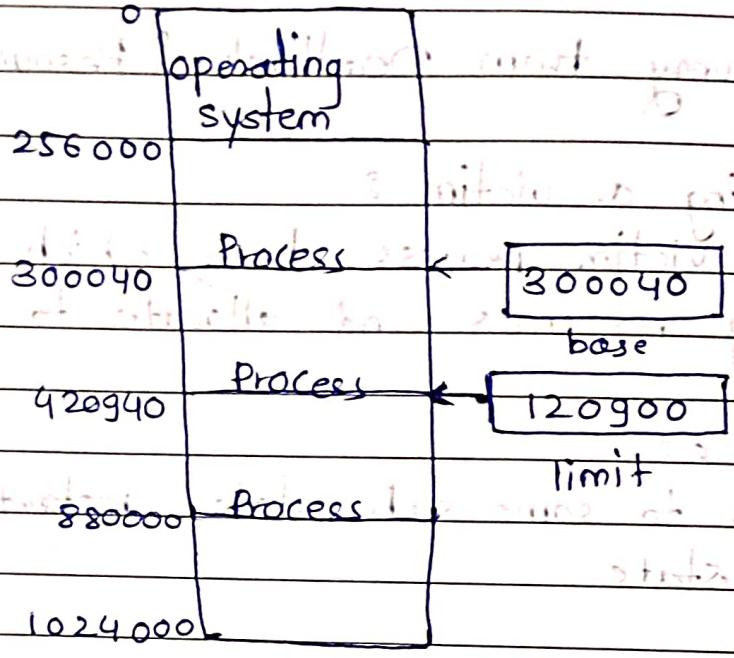
3) starvation :-
same process may always be picked as victim, include no of rollback in cost factor.

4. Memory Management

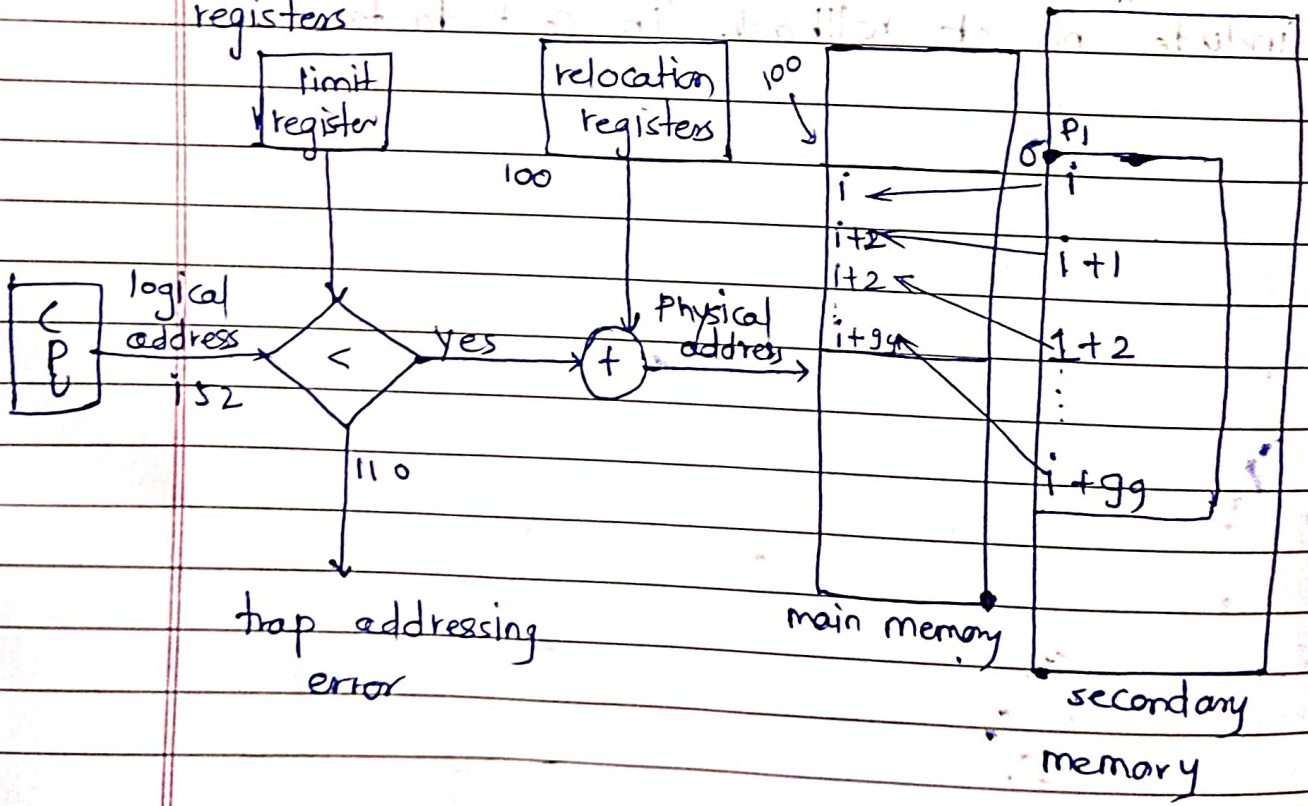
PAGE No.
DATE 24/2/23

* Base and limit Registers

-> A pair of base and limit registers define the logical address space.



-> Hardware for support for Relocation and limit registers



The label for the diagram is Pagging (logical to physical

PAGE No.
DATE / /

- CPU will throw always logical address of secondary memory instruction
- for e.g 152 instruction will be called from given process
- instruction no should be lesser than the limit register value.
- if Condition is true then redocation register will Calculate the physical address of the process by adding base address into it.

e.g - base = 100

- so if the base register is having address 100 so the physical address is 152

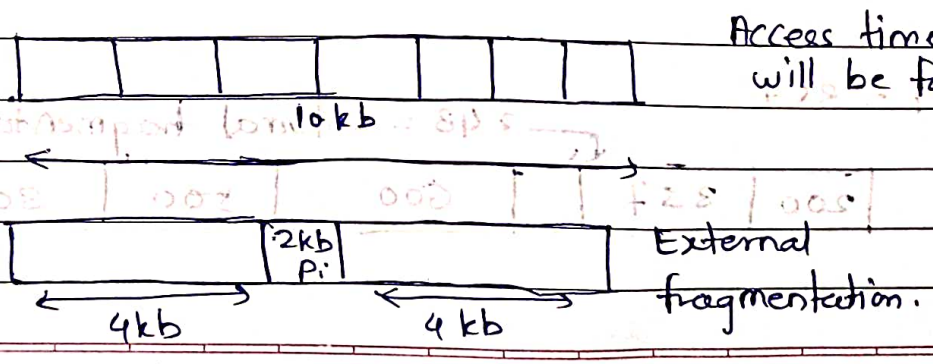
- This process is called as pagging.

- Program will divided into several no of parts are called as frames
- frames are located onto secondary memory
- frames are mapped onto pages of primary memory so, pages are located onto the primary memory.

→ Contigous Allocation - fragmentation.

There are two types.

- Internal fragmentation (Block size is greater than allocation)
- External fragmentation (Block size is less than allocation)

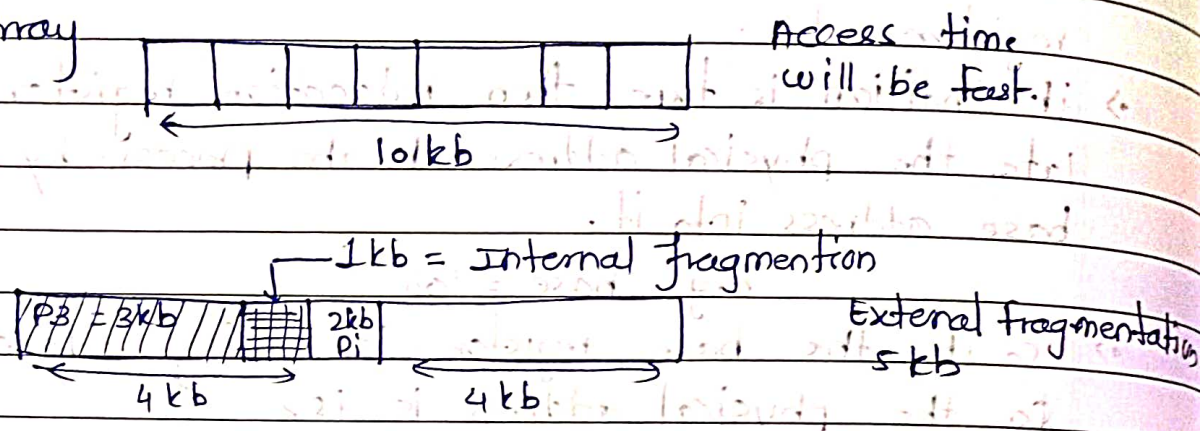


P2=5kb → X → X space not available.

PAGE No.
DATE / /

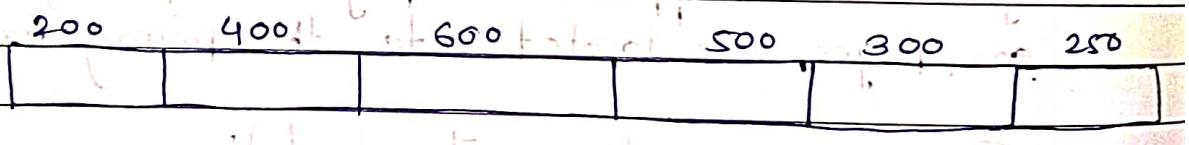
⇒ 5 kb is external fragmentation

internal fragmentation.



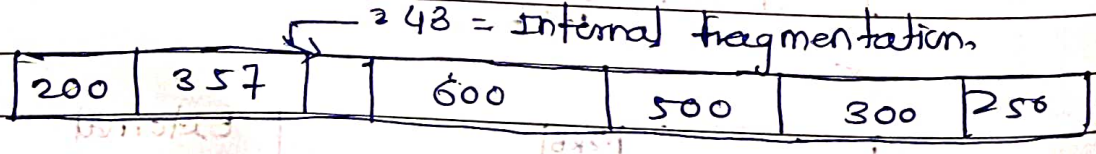
P2 = 5kb → X → X space not available.

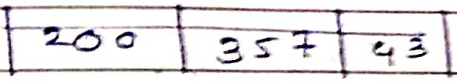
Que :- Apply first, best, worst fit on given memory partitions.

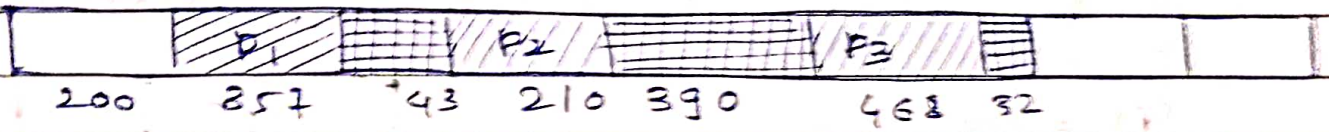


P1 = 357
P2 = 210
P3 = 468
P4 = 491

1) P1 = 357



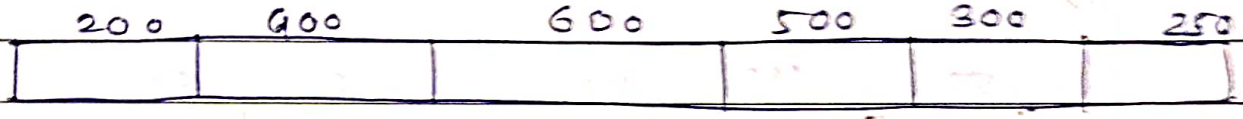




200 857 43 210 390 468 32

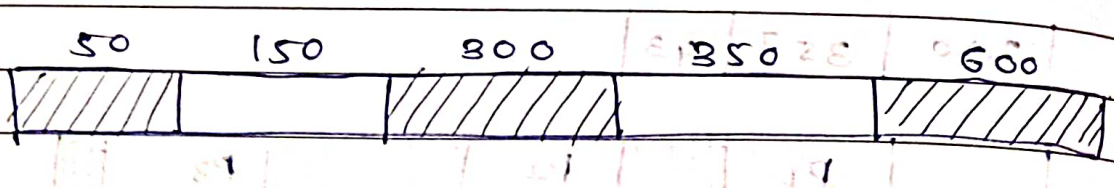
- internal fragmentation = ~~43 + 109~~ 43 + 390 + 32
- external fragmentation = 491

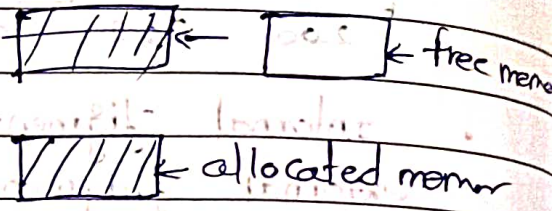
2) Best fit



- internal fragmentation =
- external fragmentation =

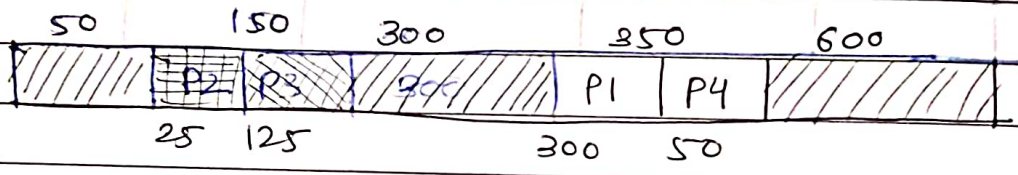
Q. Calculate Internal and External fragmentation





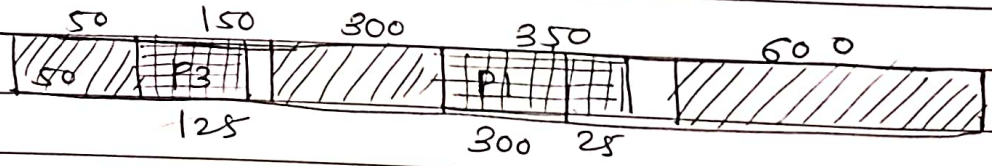
P1 = 300
P2 = 25
P3 = 125
P4 = 50

→ First fit.



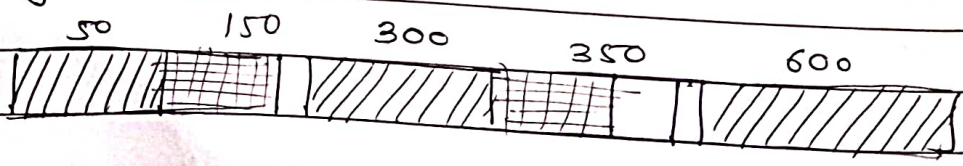
Internal fragmentation = 0
External fragmentation = 80

→ Best fit



Internal fragmentation = 0
External fragmentation = 50

→ Worst fit.



Internal fragmentation = 0
External fragmentation = 0

Q. paggin on logical to physical address.

PAGE No.
DATE / /

Logical address to Physical Address

1) CPU will call logical address of the page with offset id. → e.g

P2	P35

This is called as logical address.

2) There is process which has 3 Pages, page P1, P2, P3. These three pages are distributed on main memory at various frames

4) either the frames are Continous or non-Continous

5) The page table is created or design to locate the physical address of the page and each page will be mapped on to the respective frame number

6) Page is referred on secondary memory which is the logical address

7) Frame is referred on primary memory which is the physical address of the page.

This process is called as pagging and this is how the logical address to physical address takes place.

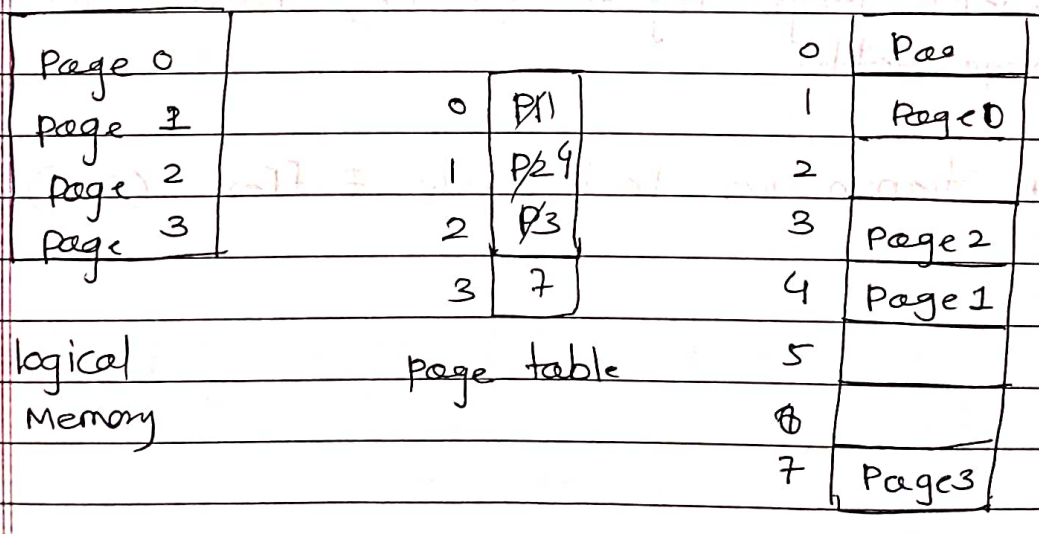
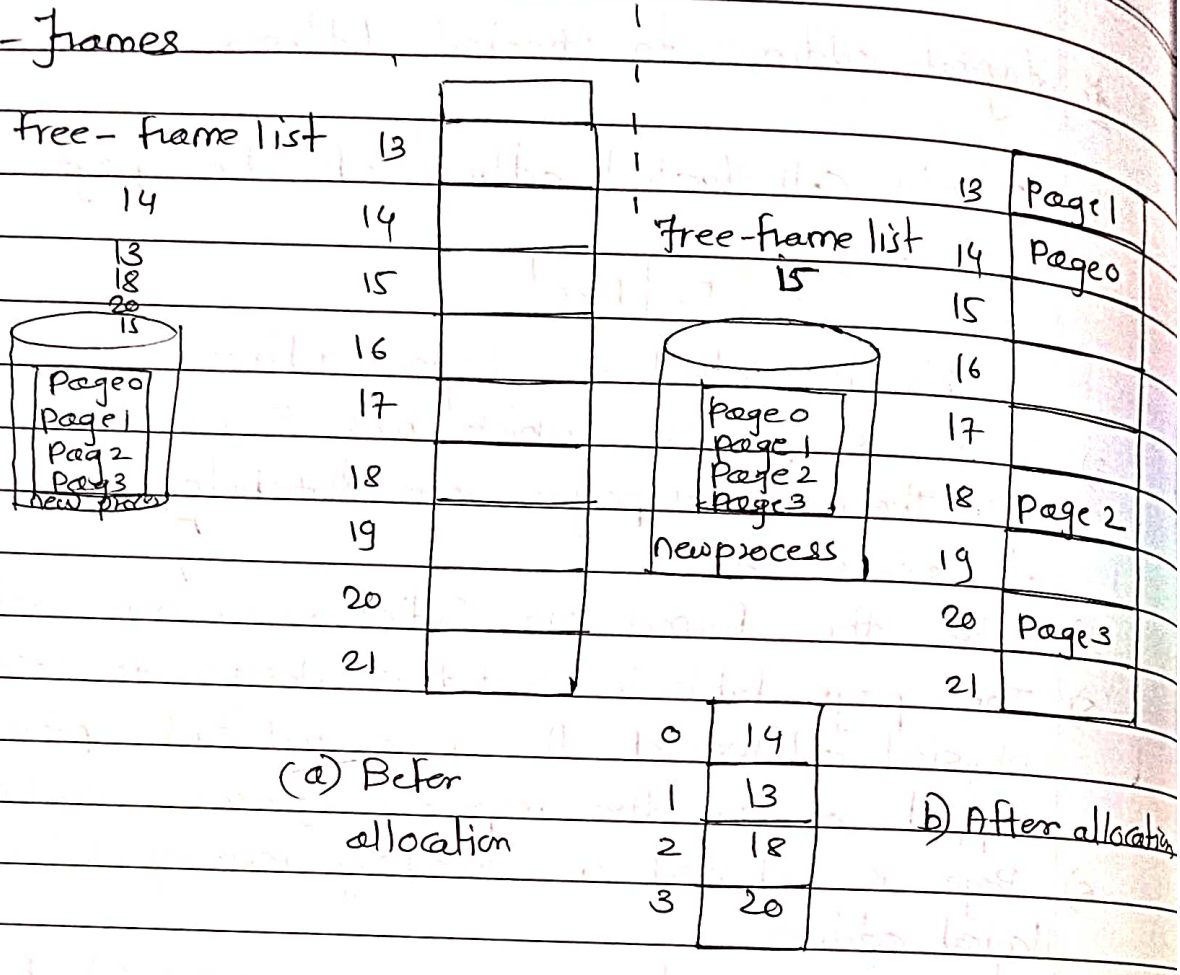


fig - Pagging model of logical to physical Memory

PAGE No.
DATE / /

•) Free - frames

→

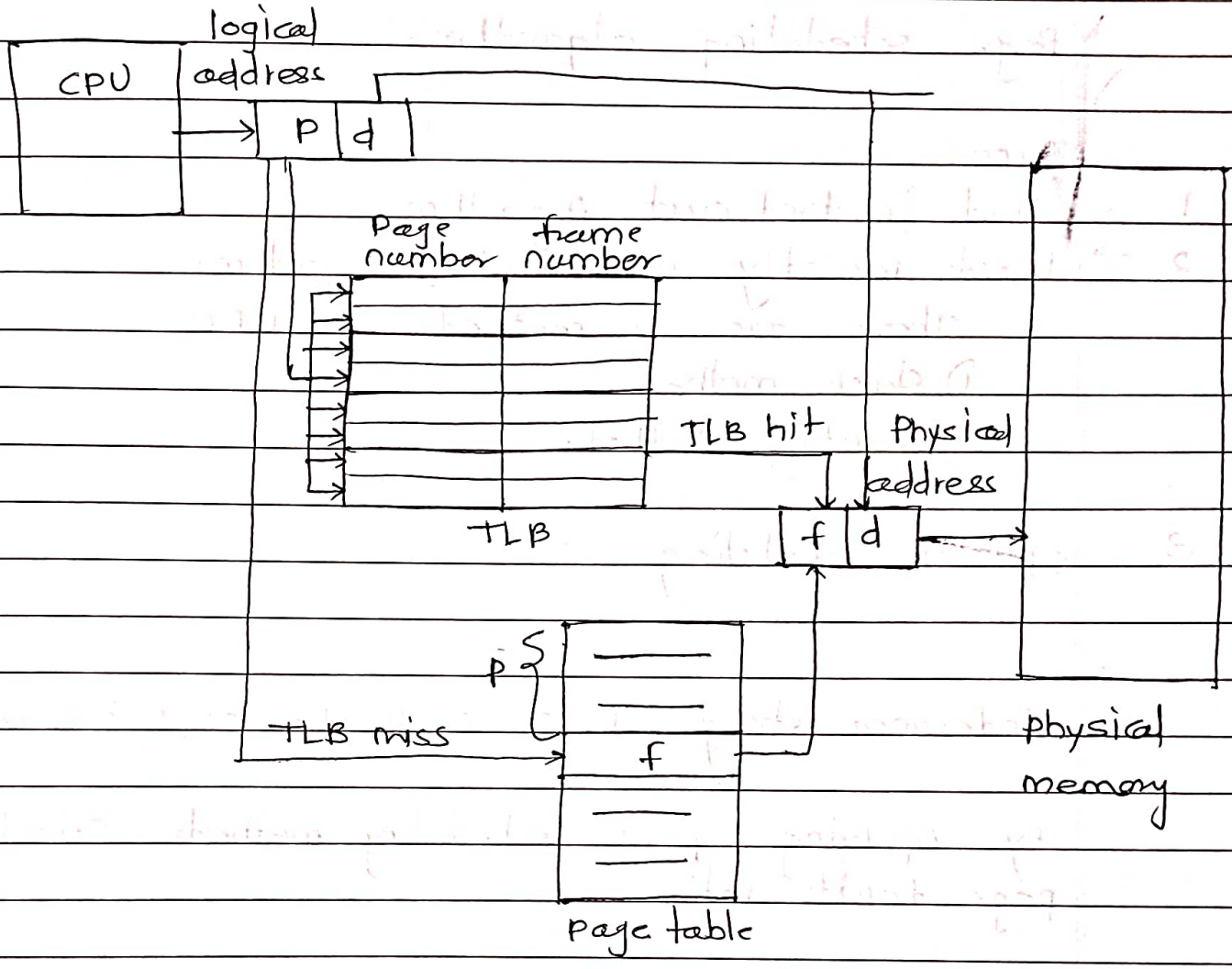


•) page-table base register (PTBR) points to page table
•) page-table length register (PRLR) indicates the size of the page table.

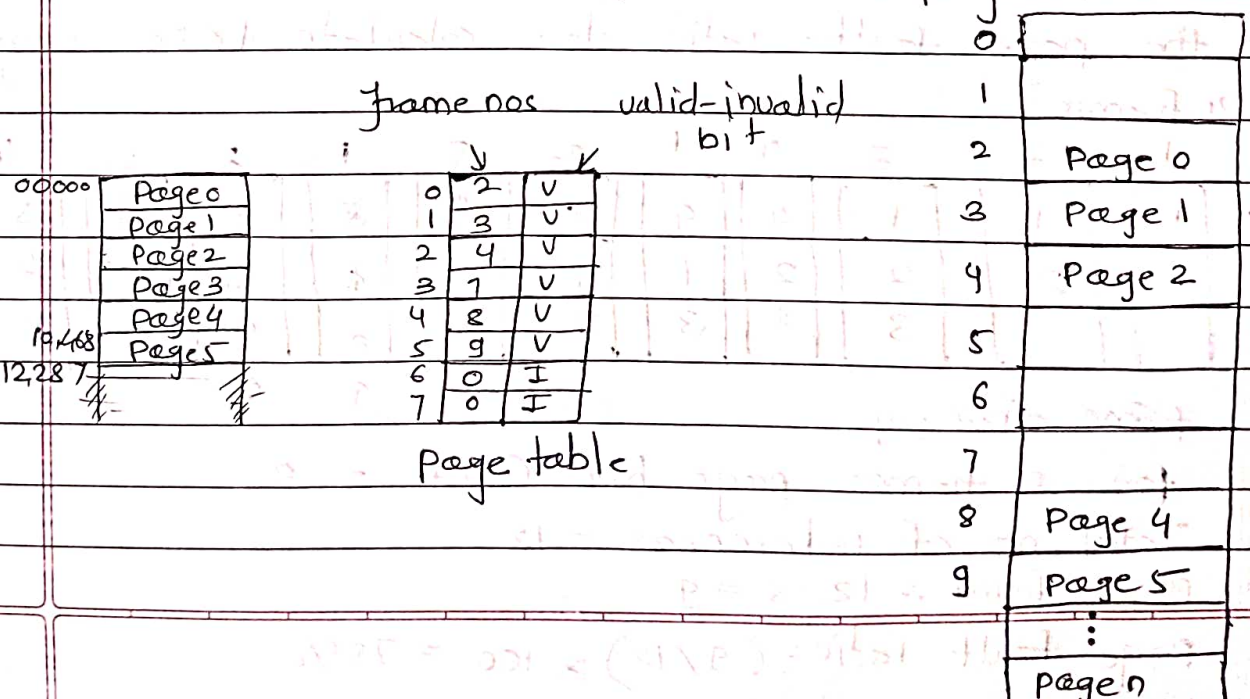
Imp What is transmission Look aside Buffer (TLB) & (10m)

PAGE No.
DATE / /

• Pagging Hardware with TLB



⇒ valid (v) or Invalid (I) Bit in A page table.



PAGE No.
DATE / /

virtual memory

3-marks

Page scheduling algorithm's

Types

1 → first in first out Algorithm

2 → Least recently Used (LRU) Algorithm.

There are 2 methods in LRU

1) Stack method

2) Counting method.

3 optimal scheduling

→

Reference string 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

sy Applying page scheduling methods Calculate page fault ratio

→ FCFS method

Assume 3 frames for execution. and find out the page fault. ratio also calculate FCFS using 4 frames

	1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	4	4	4	5	5	5	5	5	
2		2	2	2	1	1	1	1	2	3	3	
3			3	3	3	2	2	2	2	2	4	
								↑	↑			

frame size = 4

for 3 frames page hit Count = 3

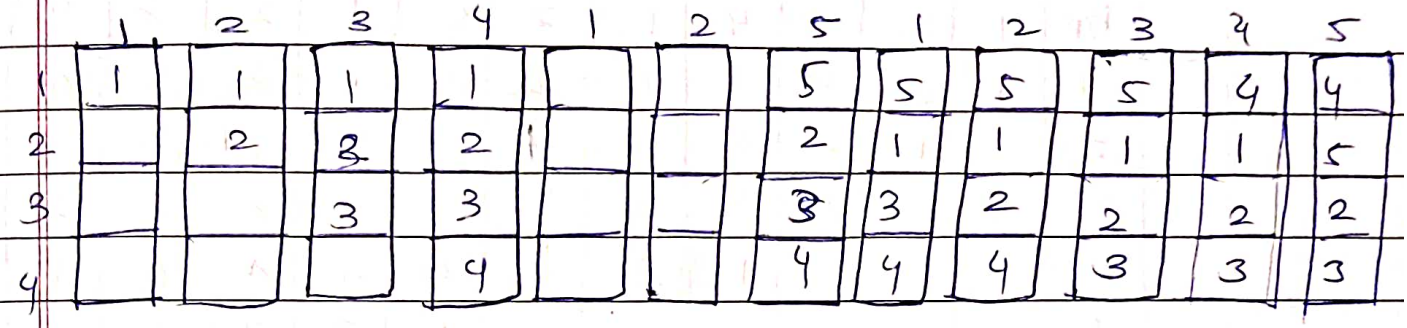
Total no of references = 12

Page fault = 12 - 3 = 9

Page fault ratio = (9/12) × 100 = 75%

PAGE No.
DATE / /

frame size = 4



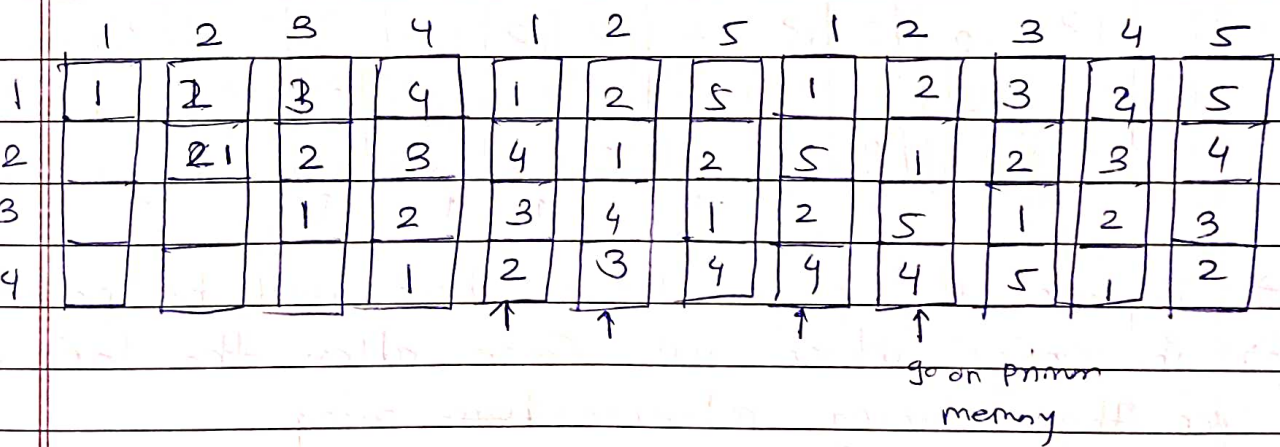
for 4 frames page hit count = ~~4~~ ~~2~~ 4 2

total no of references = 12

page fault = 12 - ~~4~~ 2 = ~~8~~ 10

page fault ratio (10/12) × 100 = 83.33%

LRU (stack method)



frame size = 4

for 4 frames page hit count = 4

total no of references = 12

page fault = 12 - 4 = 8.

page fault ratio = (8/12) × 100 = 0.66 × 100

= 66.66%

PAGE No.
DATE 1 1

LRU (counting method)

P/R	1	2	3	4	1	2	5	1	2	3	4	5
1	1	2	3	4	1	2	3	1	2	3	4	0
2		1	2	3	4	1	2	3	4	2	3	4
3			1	2	3	4	0	0	0	1	2	3
4				1	2	3	4	4	4	0	1	2
5							1	2	3	4	0	1

↑ ↑

Page hit Page hit.

Optimal Page Replacement.

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	1	1	1	1	4	4
	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3	3
			4	4	4	5	5	5	5	5	5

↑ ↑ ↑ ↑ ↑ ↑

→ Replace the page when there will be page fault
→ The page which will come after the long run on the remaining reference ~~from~~ string

Frame size = 4
page hit = 6 total no of references = 12
page fault = ~~6 - 4~~ = 2 12 - 6 = 6
page fault ratio = (6/12) × 100 = ~~33.33%~~ 50%.

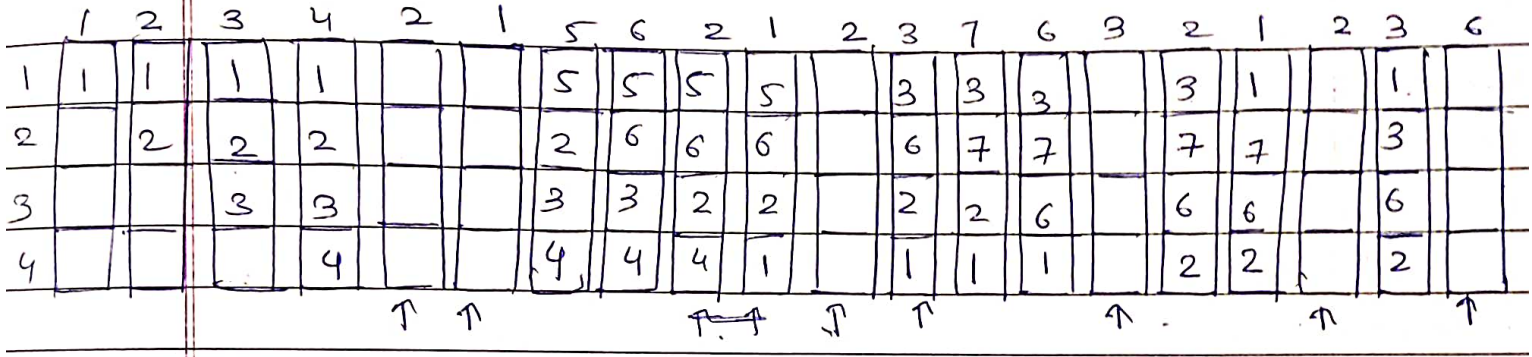
PAGE No.
DATE / /

Example There are 7 pages and 4 frames. Calculate number of pages fault / pages miss and page fault ratio.

Reference string

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

1) LRU FCFS Method (first in first out)



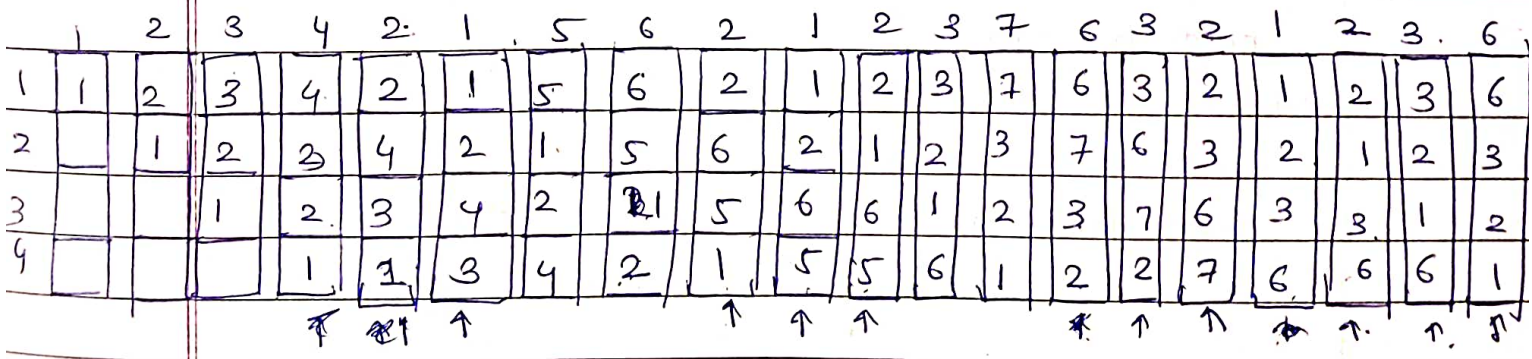
frame size = 4

Page hit = 6 Total no of refere = 20

page fault = ~~6-4=2~~ 20 - 6 = 14

Page fault ratio = (14/6) × 100 = ~~33.33%~~ 70%

2) LRU (stack method)



frame size = 4

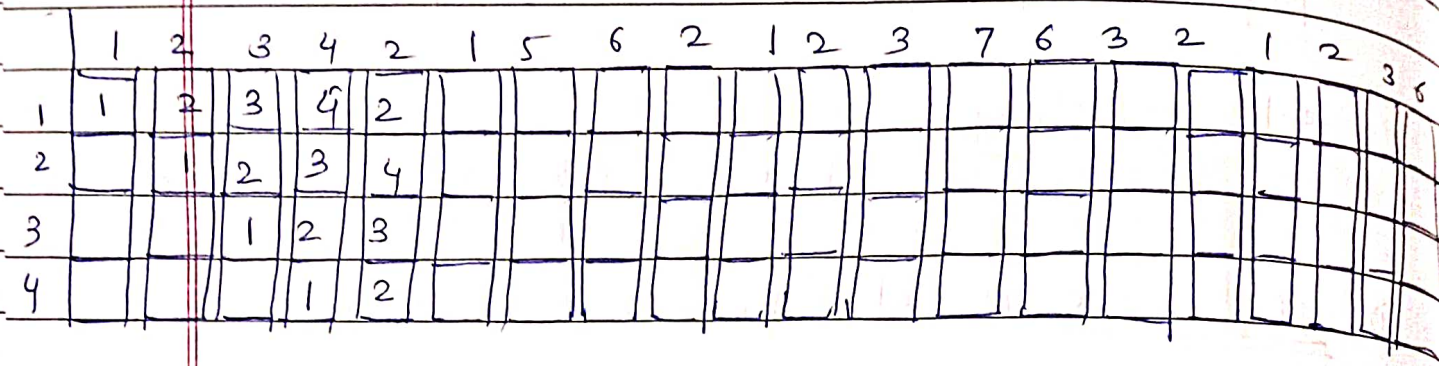
Page hit = ~~8~~ 10

total no of references = 20

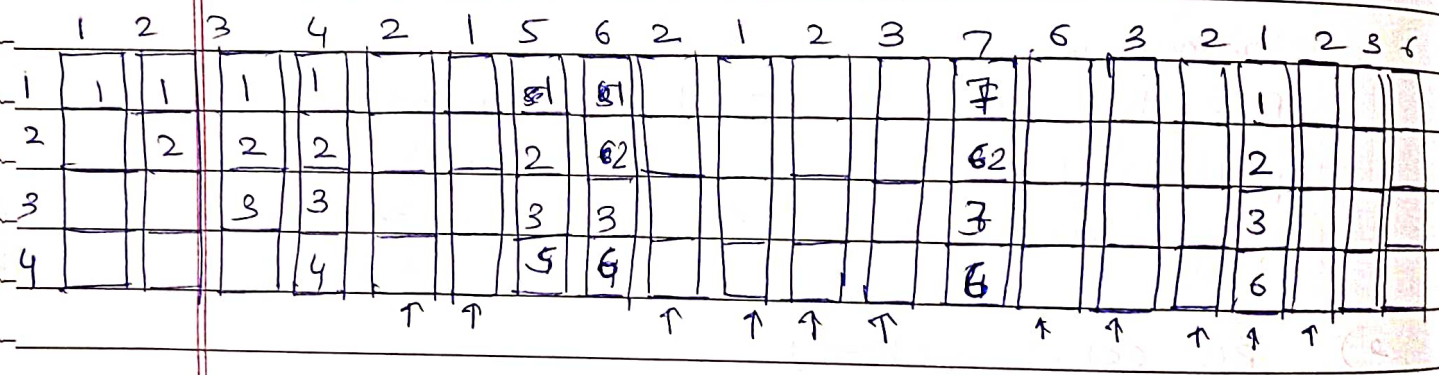
page fault = 20 - ~~8~~ 10 = 10

Page fault ratio = ~~(13/7) × 100~~ = ~~70/2~~ (10/20) × 100 = 50%

3) LRU (counting method)



4) optimal



page hit = 12

Total Refer = 20

page fault = 20 - 12 = 8

page fault ratio = $\frac{8}{20} \times 100 = 40\%$

PAGE No.
DATE / /

3) LRU Counting method.

P/R	1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
1	1	2	3	4	4	1	2	3	4	1	2	3	4	0	0	0	1	2	3	4
2		1	2	3	1	2	3	4	1	2	1	2	3	4	0	1	2	1	2	3
3			1	2	3	4	0	0	0	0	0	1	2	3	1	2	3	4	3	1
4				1	2	3	4	0	0	0	0	0	0	0	0	0	0	0	0	6
5							1	2	3	4	4 3	0	0	0	0	0	0	0	0	0
6								1	2	3	3 4	4	0	1	2	3	4	4	4	5
7													1	2	3	4	0	0	0	0

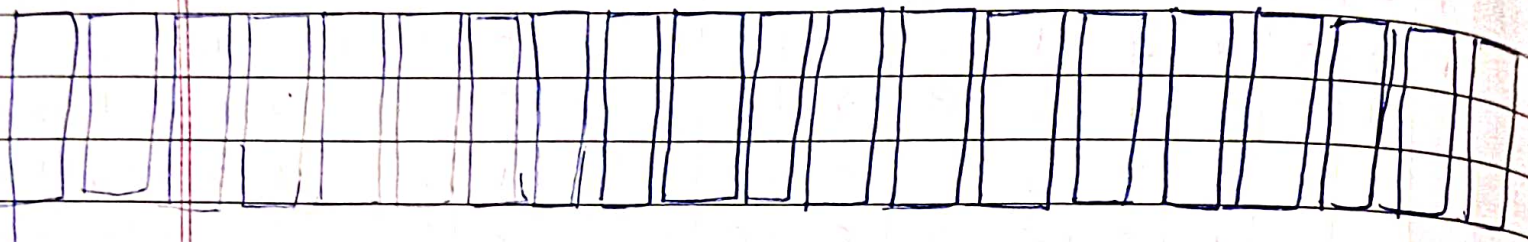
Example-3 Calculate page fault/miss and page hit for FIFO, LRU, OPTIMAL, Consider frame size = 3

IA-2

6 0 5 2 0 3 0 4 2 3 0 3 2 5 2 0 5 6 0 5

LRU

6



Op

optimal

	6	0	5	2	0	3	0	4	2	3	0	3	2	5	2	0	5	6	0	5
1	6	6	6	2		2		2		2	2			2				6		
2		0	0	0		0		4			0			0				0		
3			5	5		3		3			3			5				5		
			↑			↑		↑		↑		↑			↑		↑		↑	

page hit = 11

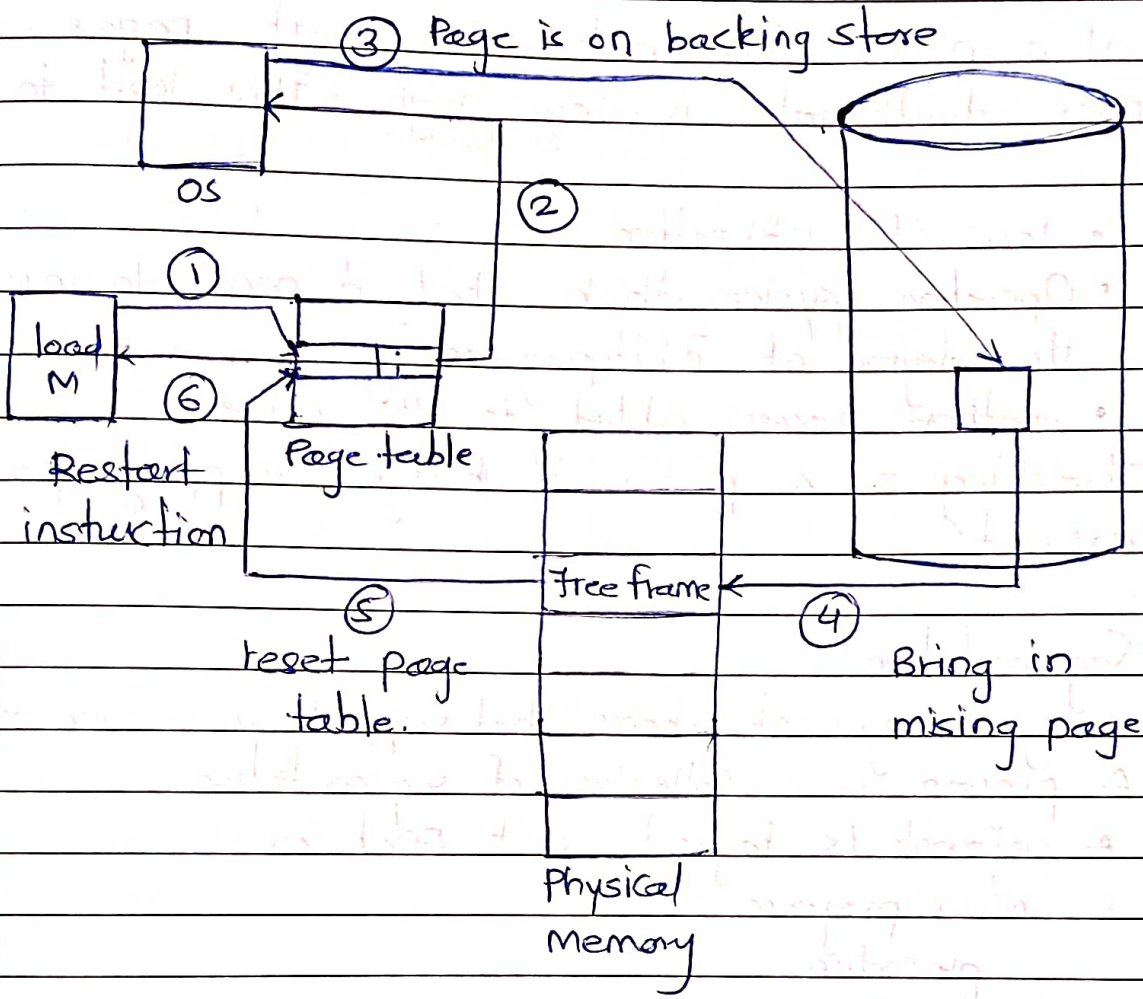
total refer = 20

page fault = 20 - 11 = 9

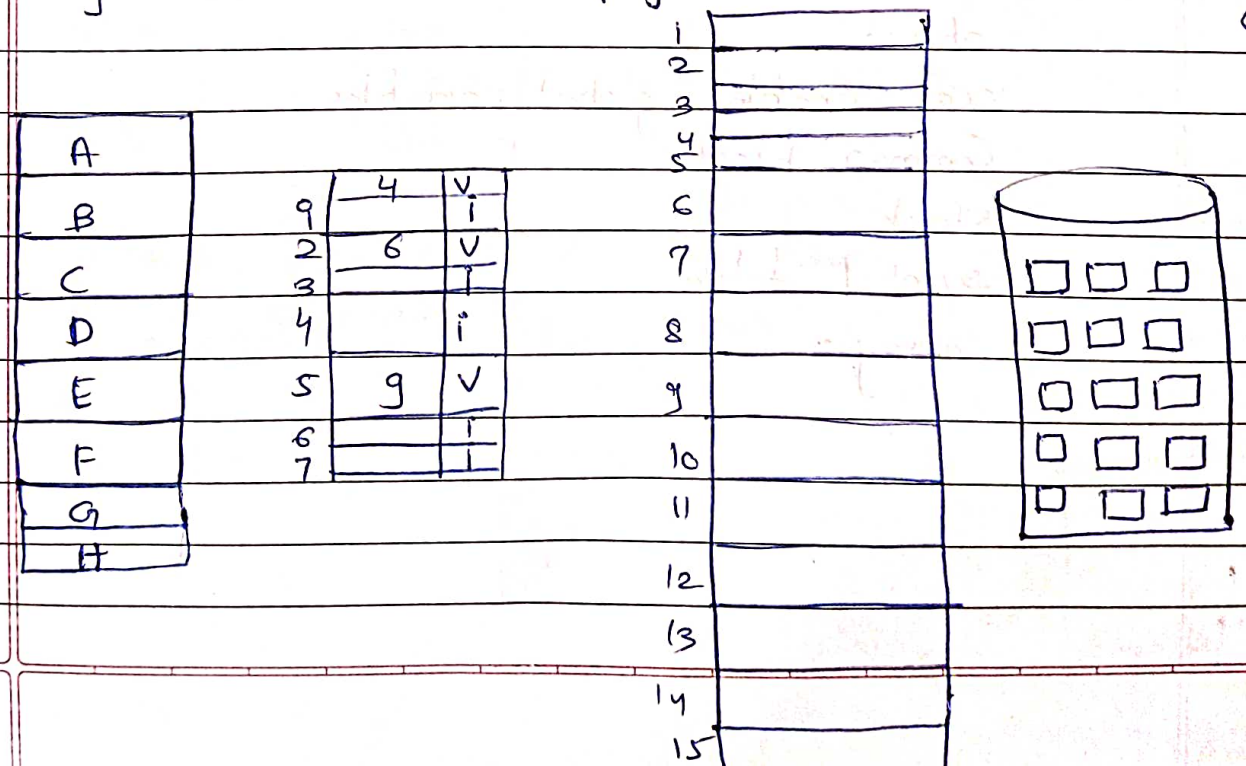
page fault ratio = $\frac{9}{20} \times 100 =$

PAGE No.
DATE / /

steps in handling a page fault.



→ Page table when some pages are not in main memory.



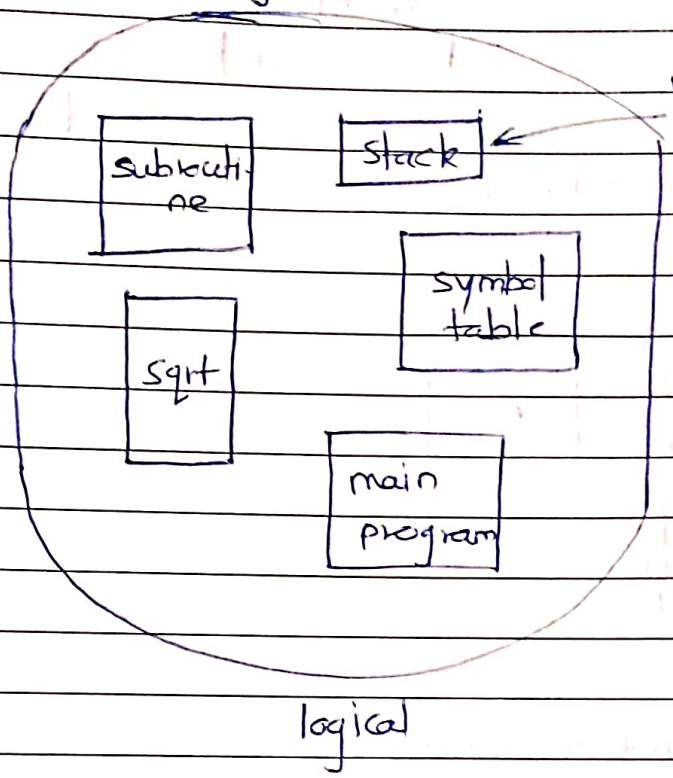
⇒ Page fault rate ... increasing.

Thrashing:

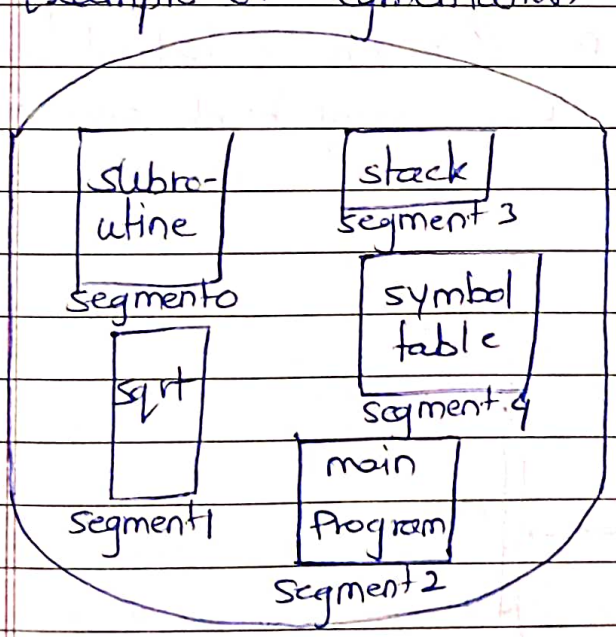
- If a process does not have "enough" pages, the page fault rate is very high. This lead to:
  - low CPU utilization
  - Operating system thinks that it needs to increase the degree of multiprogramming.
  - another process added to the system.
- Thrashing = a process is busy swapping pages in and out.

⇒ Segmentation
→ memory management scheme that supports user view of memory
→ A program is a collection of segments.
→ A segment is logical unit such as.
  main program
  procedure
  function
  method
  object
  local variables, global variables
  common block
  stack
  symbol table
  arrays.

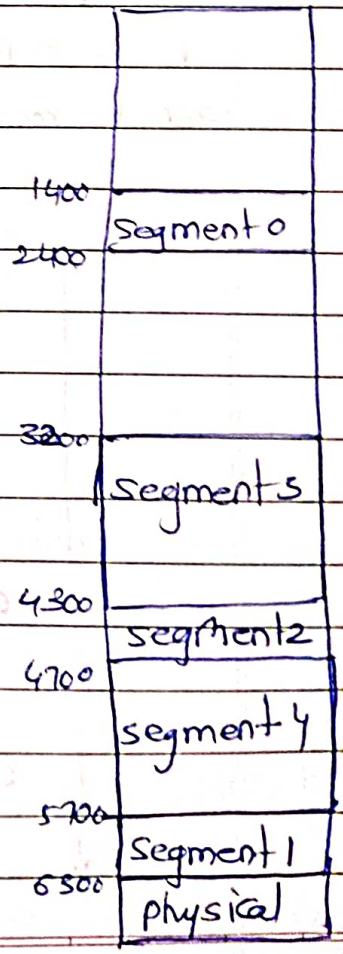
o/ User's view of a program.



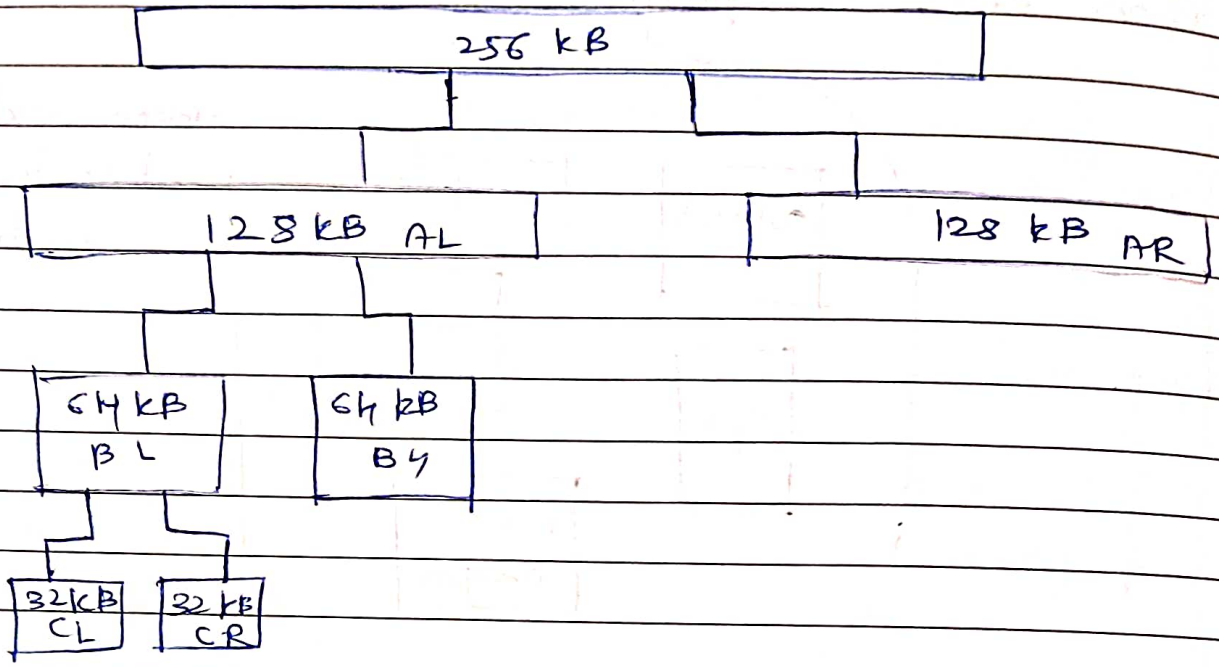
o/ Example of segmentation



	limit	Base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

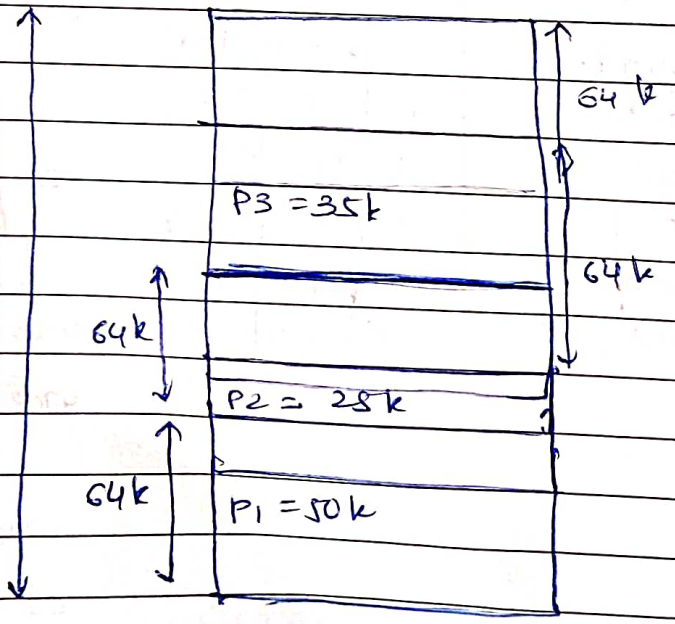


→ Buddy system Allocator



•> Buddy system example.

In Buddy system of memory management request of $P_1 = 50k$, $P_2 = 25k$, $P_3 = 35k$ are satisfied with 256 k available memory. how many block size are left?



read - 4 $\frac{4+2+1}{7}$

PAGE No.
DATE / /

Module 5 : Storage and file Management.

File

=> File is Continous logical entity from secondary storage two major types are - data
program

=> contents defined by file a creator are of many types
e.g text file
source file
executable file

=> following are the file attributes.

1. Name of file
2. file identifier - i) which will be unique tag numbe
ii) used for indexing.
3. Type of file :
4. location of the file : where the file located on device ster
5. size of the file
6. process protection : which will be read write and executio
7. time, date and user identification :

File Operations
file is an abstract data type
1. create 2. write 3. read 4. Reposition within the file.

4. Reposition within file : This will Calculate the seek time depends upon ~~drag~~ track sector and cylinder from the

5. delete.
6. Truncate : delete the Contents of the file without deleting file.
7. open (fi) : search the directory structure on disk for entity fi and move the Content of entity to memory.

PAGE No.
DATE / /

7. close(fi) :- move the Contents of

* Open files.

several places of data are needed to manage open files.

- open-file table
- file-open count.

→ open-file table :- tracks open files.
→ file pointer :-

- open file locking.

↳ shared lock - similar to reader lock (reader enter than writer lock but multiple reader acce
↳ Exclusive lock - similar to writer lock (when writer enter both reader & writer lock outsid

→ Mandatory or advisory lock

→ mandatory :- access to desired denied depends on locks held and requested
→ advisory lock :- Processes can find status of locks and decide what to do.

- Access lists and groups.

→ mode of access :- read, write, execute.
→ Three classes of user on unix / Linux.

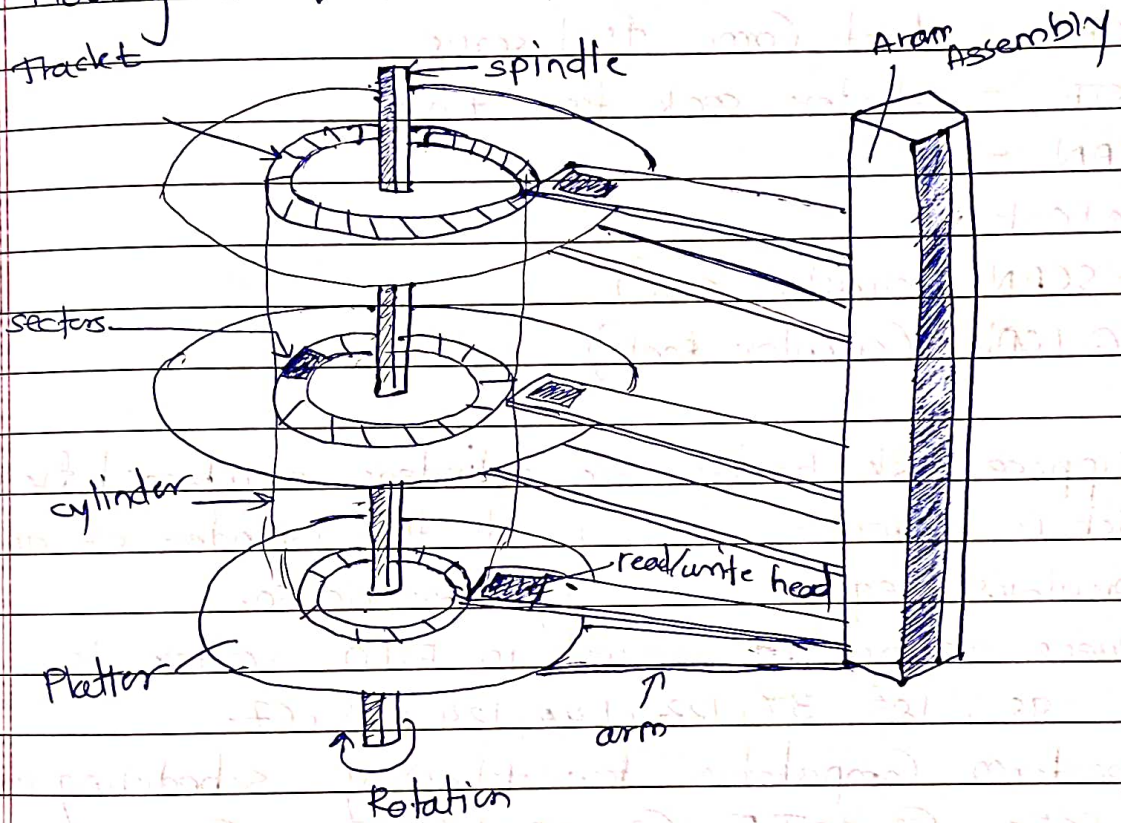
		RWX
a) owner access	7	1 1 1
b) group access	6	RWX 1 1 0
c) Public access	1	RWX 0 0 1

owner group public

chmod 761 game.

→ Attach a group to file
chgroup and game.

* Moving-head Disk Mechanism



P.P (physical addr - 1° storage)
(logical addr - 2° storage)

- Data transfer will takes place between disk drive and processor
- platter diameter will be of 1.8 to 3.5 inch
- platter Covers with magnetic materials from both sides.
- magnetic pattern from the platter is used for read and write purpose.
- rotates, write, rotates 60 to 50 per second which is called as rotation per minute.

PAGE No.
DATE / /

- Common drive spins :-
- rotation speed relates to transfer speed

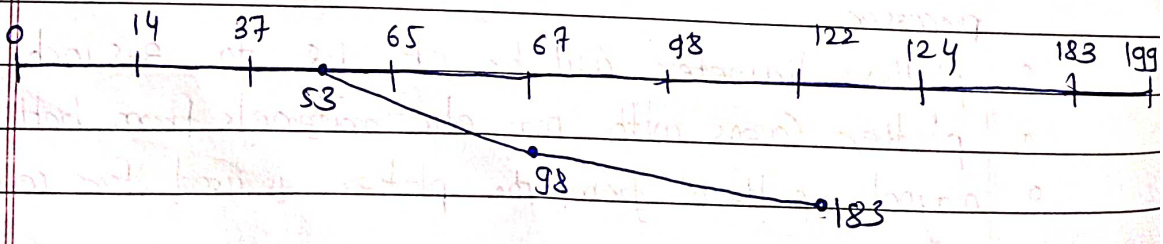
→ ⇒ Disk scheduling

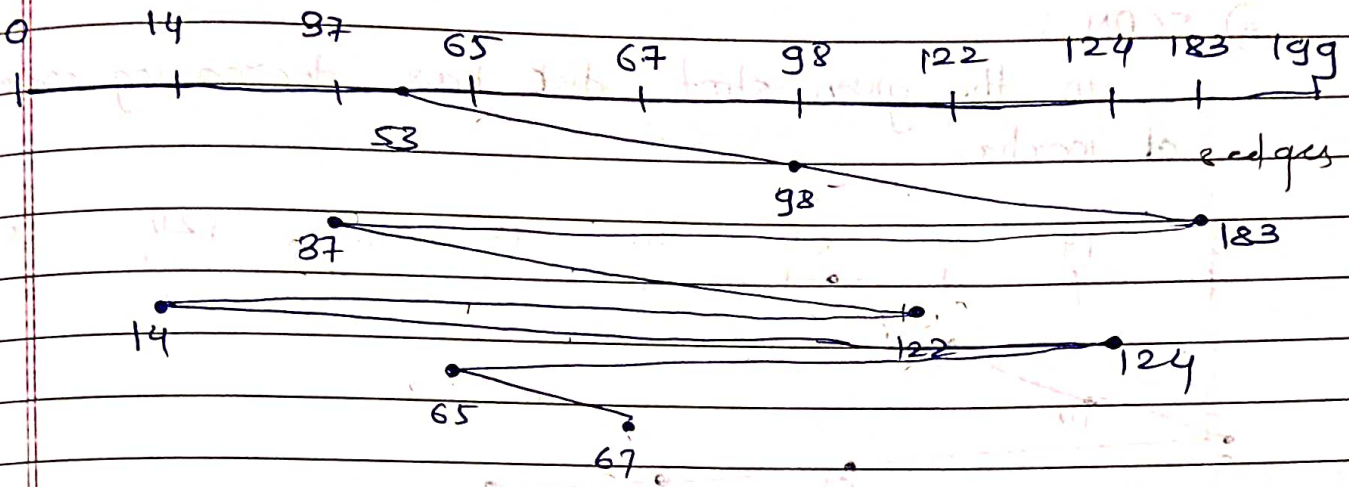
1) FCFS - first Come first serve
2) SSTF - shorter seek time first
3) SCAN -
4. LOOK
5. C-SCAN (circular scan)
6. C-LOOK (circular look)

example Suppose disk having 200 cylinders numbered from 0-199. disk is currently serving at the cylinder 53 and previous request is at cylinder no 60.
Queue of pending request in FIFO order is -
98, 183, 37, 122, 146, 124, 65, 67.
perform Computation for following scheduling algorithm
① FCFS ② SSTF ③ SCAN ④ LOOK ⑤ C-SCAN ⑥ C-LOOK

Solution

① FCFS
→ Queue of pending request in FIFO order is 98, 183, 37, 122, 146, 124, 65, 67

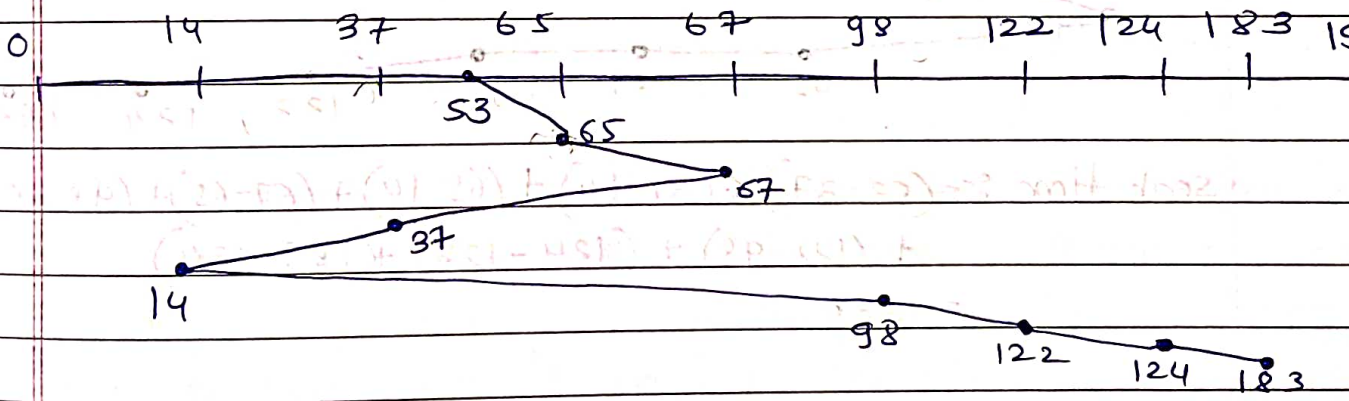




seek time :- (98-53) + (183-98) + (37-122) + (122-14) + (183-37) + (122-37) + (122-14) + (124-14) + (124-65) + (67-65)

= 640

2] SSTF [shortest seek time first]



seek time :- (65-53) + (67-65) + (67-37) + (37-14) + (98-14) + (122-98) + (124-122) + (183-124)

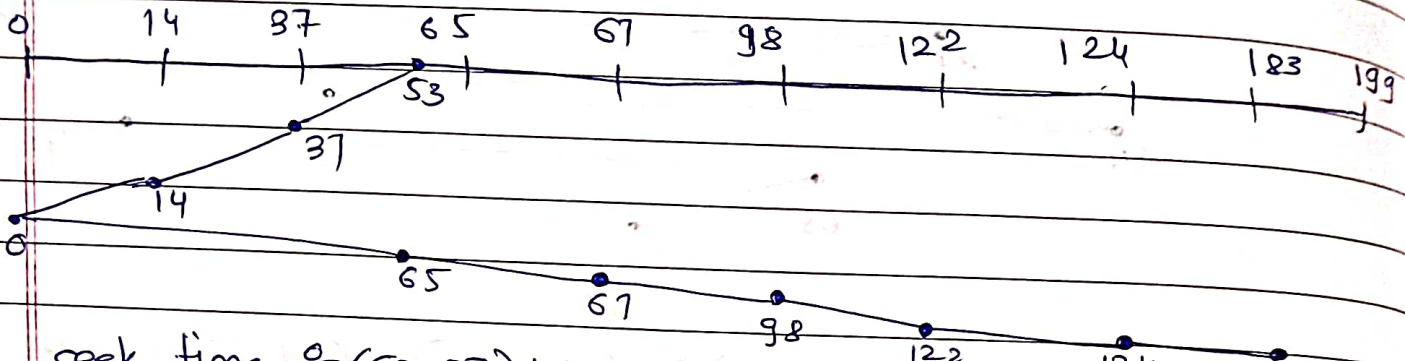
= 12 + 2 + 30 + 23 + 84 + 24 + 2 + 59

= 236.

PAGE No.
DATE / /

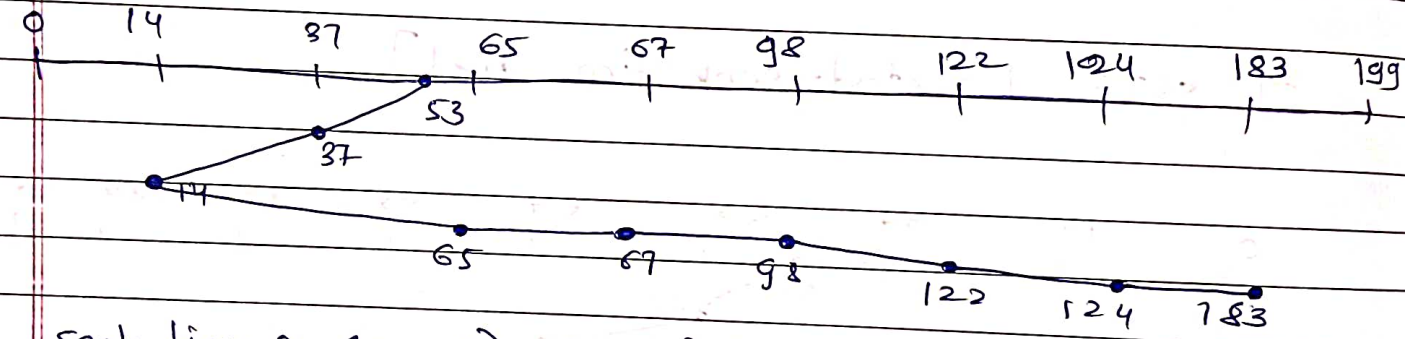
2) SCAN :-

In the given strat disk has decreasing order of inertia.



seek time :- (53-37) + (37-14) + (14-0) + (65-0) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) = 236.

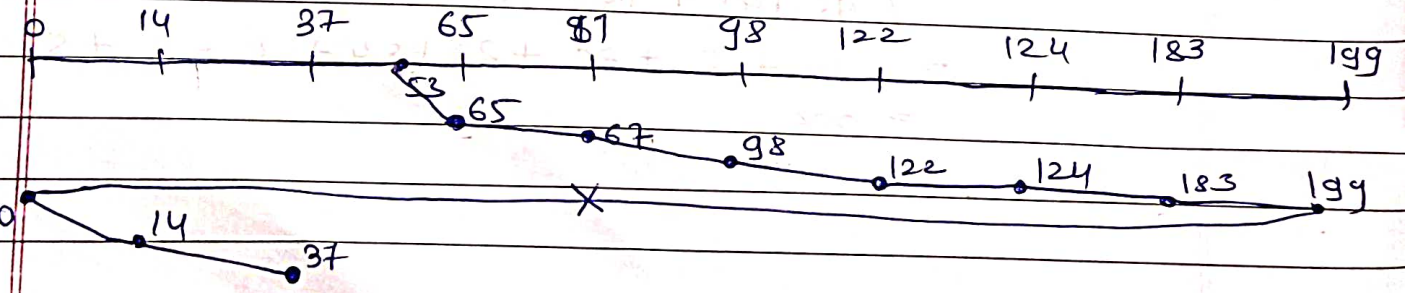
4) LOOK :-



seek time :- (53-37) + (37-14) + (65-14) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124)

= 208

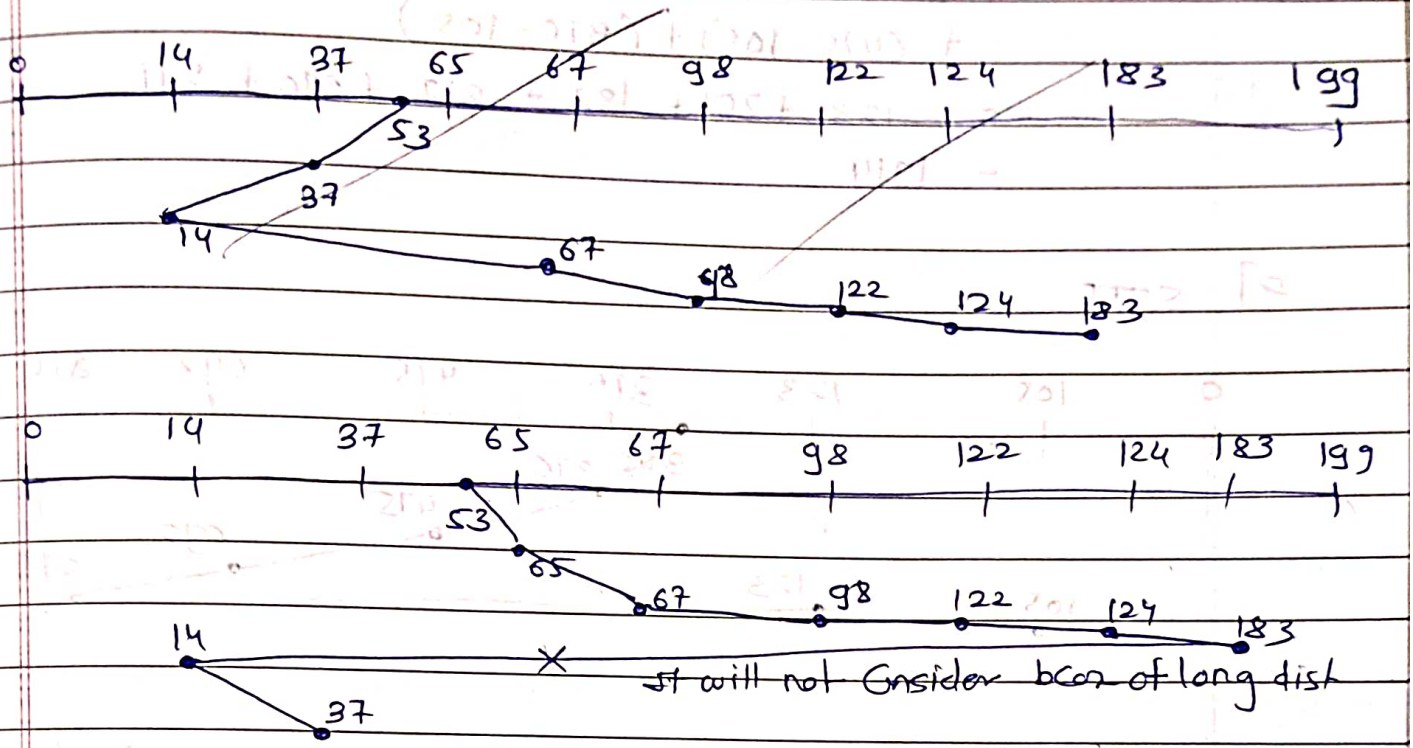
5) C-SCAN



seek time :- (65-53) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) + (199-183) + ~~(199-0)~~ + (14-0) + (37-14)

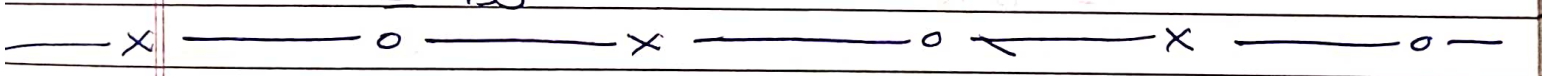
= 183

6) C-LOOK :-



seek time :- (65-53) + (67-65) + (98-67) + (122-98) + (124-122) + (183-124) + (37-14)

= 153.

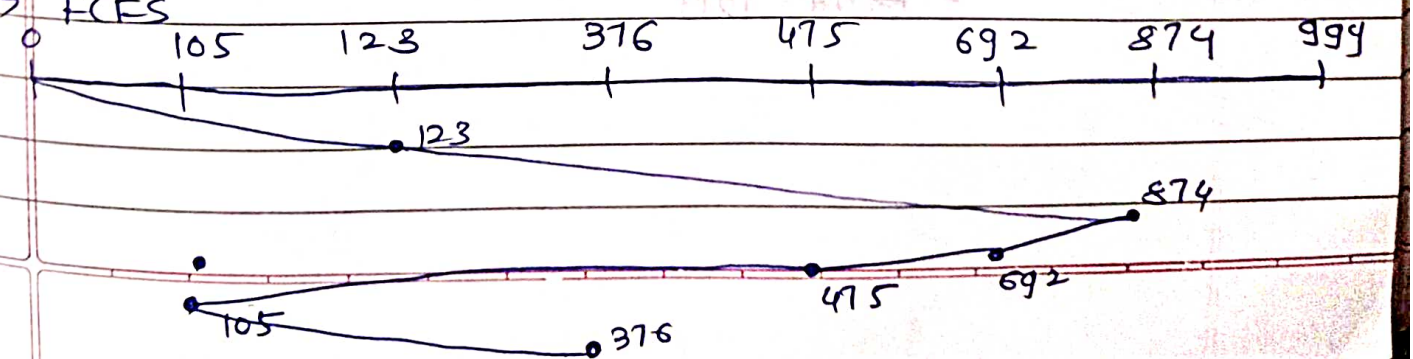


Example-2 on disk of 1000 cylinders, number 0-999, Consider number of tracks that disk arm must move to satisfy all request in disk queue. Assume the last request received was at track 345 & head is moving toward track 0. The queue in FIFO order Contains request for following track

123, 874, 692, 475, 105, 376

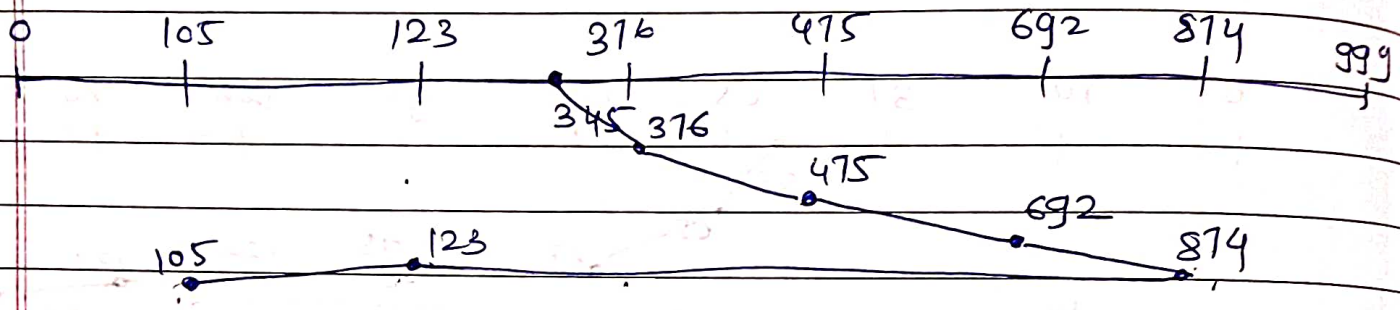
perform (a) FCFS (b) SSTF (c) SCAN (d) Look

→ FCFS



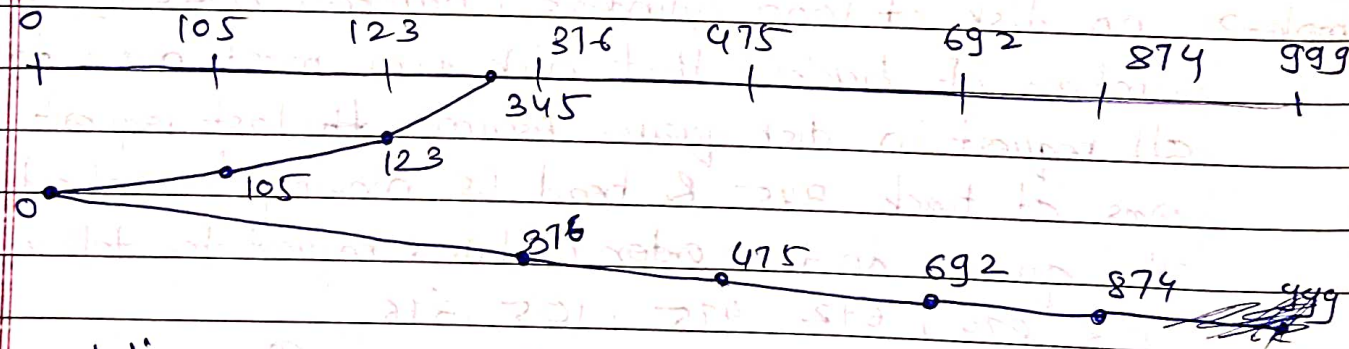
seek time = (123-0) + (874-123) + (874-692) + (692-475)
+ (475-105) + (376-105)
= 123 + 751 + 182 + 217 + 370 + 271
= 1914

2] SSTF



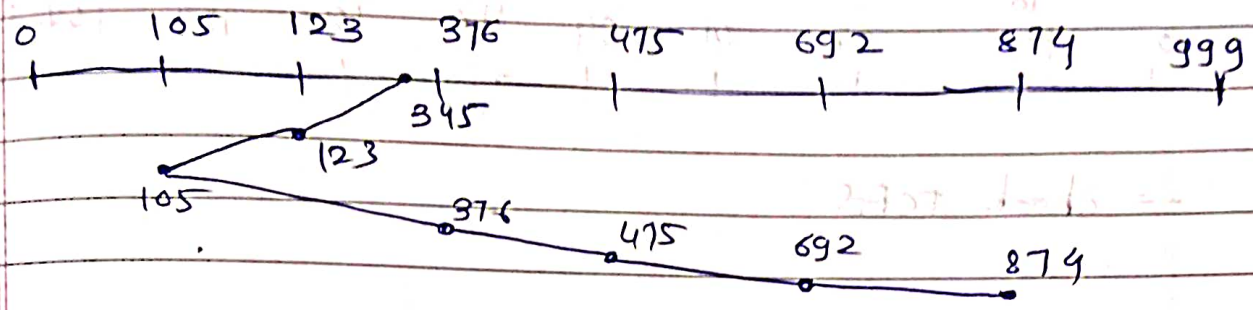
seek time = (376-345) + (475-376) + (692-475) + (874-692)
+ (874-123) + (123-105)
= 31 + 99 + 217 + 182 + 751 + 18
= 1298

3] SCAN



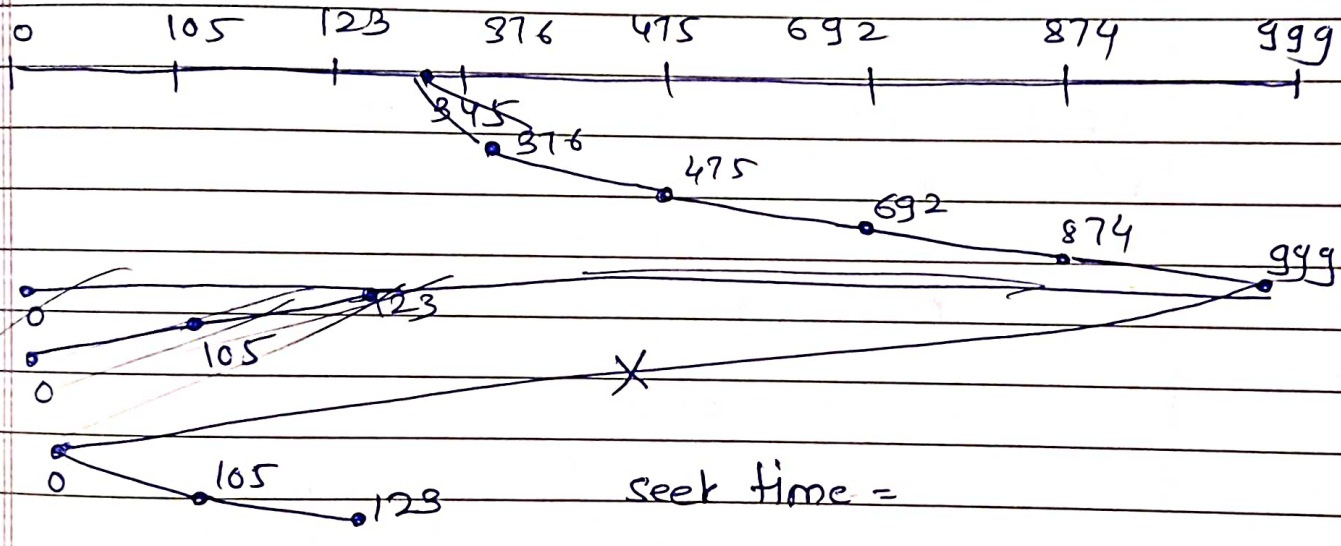
seek time = 222 + 18 + 105 + 376 + 99 + 217 + 182 + ~~185~~
= ~~1384~~ 1219

4] LOOK :-



Seek time = 222 + 18 + 271 + 99 + 217 + 182
= 1009.

5] C-SCAN :-



Seek time =

6] C-LOOK :-

Q write benefit/advan... raid... single... parity on data file in Raid-5 or raid 6

PAGE No.
DATE / /

RAID Structure

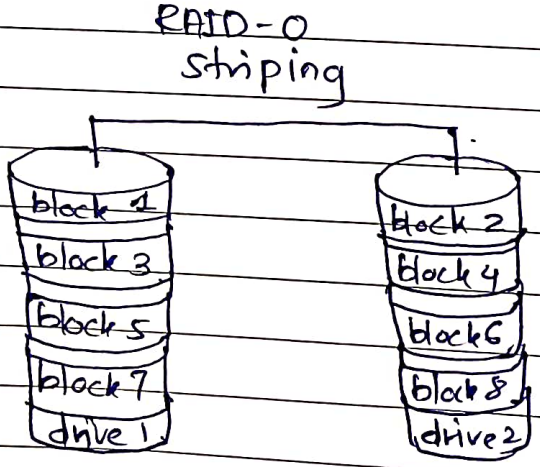
It is redundand array of inexpensive / independent di... following types of RAID

1. → ~~RAID~~ RAID-0 Block striping
2. RAID-1 disk mirroring
   RAID-2 Bit striped
   RAID-3 Bit striped with parity
   RAID-4 Block striped
   RAID-5 Block striped with parity.
   RAID-6 Dichibuted block with 2 parity.
   RAID-01
   RAID-10

1] RAID-0 [Block striping] : Data is re written in the form of blocks

RAID-0

→ block is nothing but multiple bytes together



→ disadvantages : if 1 drive fail then entire structure will be fail (drive failure)
→ In drive failur won't be recover data

Compare & differentiate Raid-0 & Raid-1

advantages & disadvan of raid-0 & raid-1

PAGE No.
DATE / /

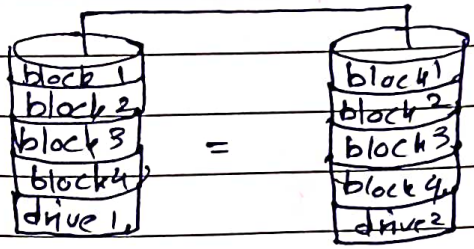
→ Compare raid 2 & raid 3

→ raid: 0 & raid 3

2] RAID-1 : [Mirroring]

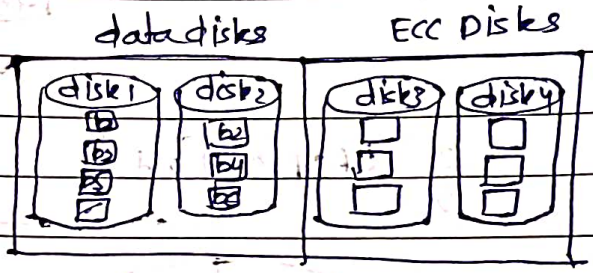
data is written in terms of blocks

It is bit's



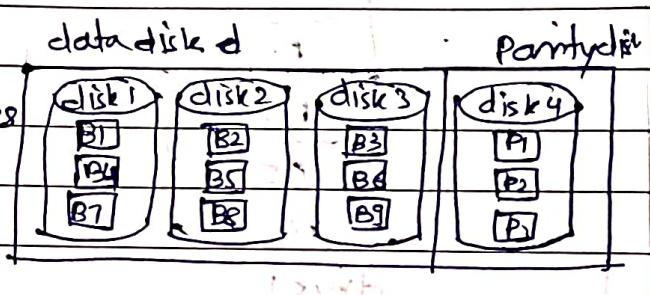
3] RAID-2 [Bit stripping]

- data is written in term of bits
- store error correction code



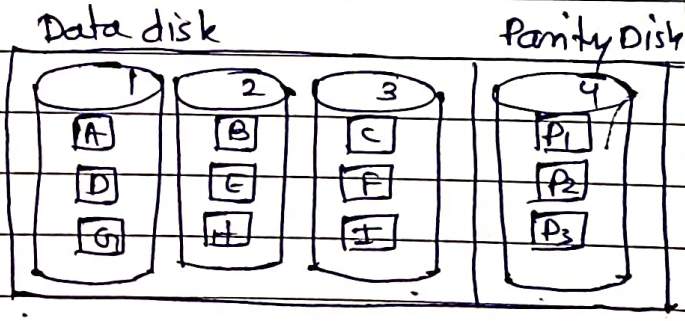
4] RAID-3 - [Byte stripping]

- data is written in terms of bytes



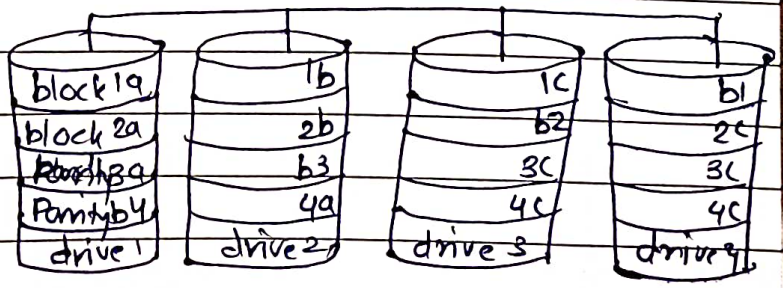
5] RAID-4 [Block stripping]

(Parity disk)



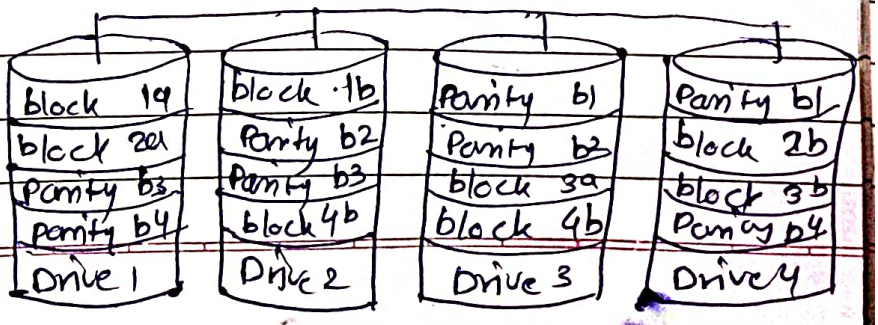
6] RAID-5 [striping with parity across drives)

— Block stripping with one distributed parity



7] RAID-6 :

- Distributed block with 2 parity

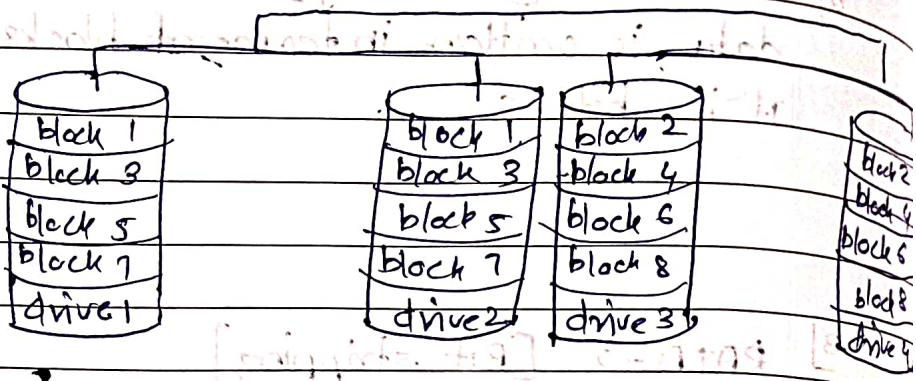


PAGE No.
DATE / /

Raid 10 (1+0)

8] RAID 10 (1+0)

mirroring + stripping



9] RAID-01 [0+1]