VIRTUAL MEMORY

VIRTUAL MEMORY

Virtual memory gives an illusion, to the programmer that, the programs of large size than actual physical memory can be executed.

Virtual memory is implemented by using demand paging or demand Segmentation.

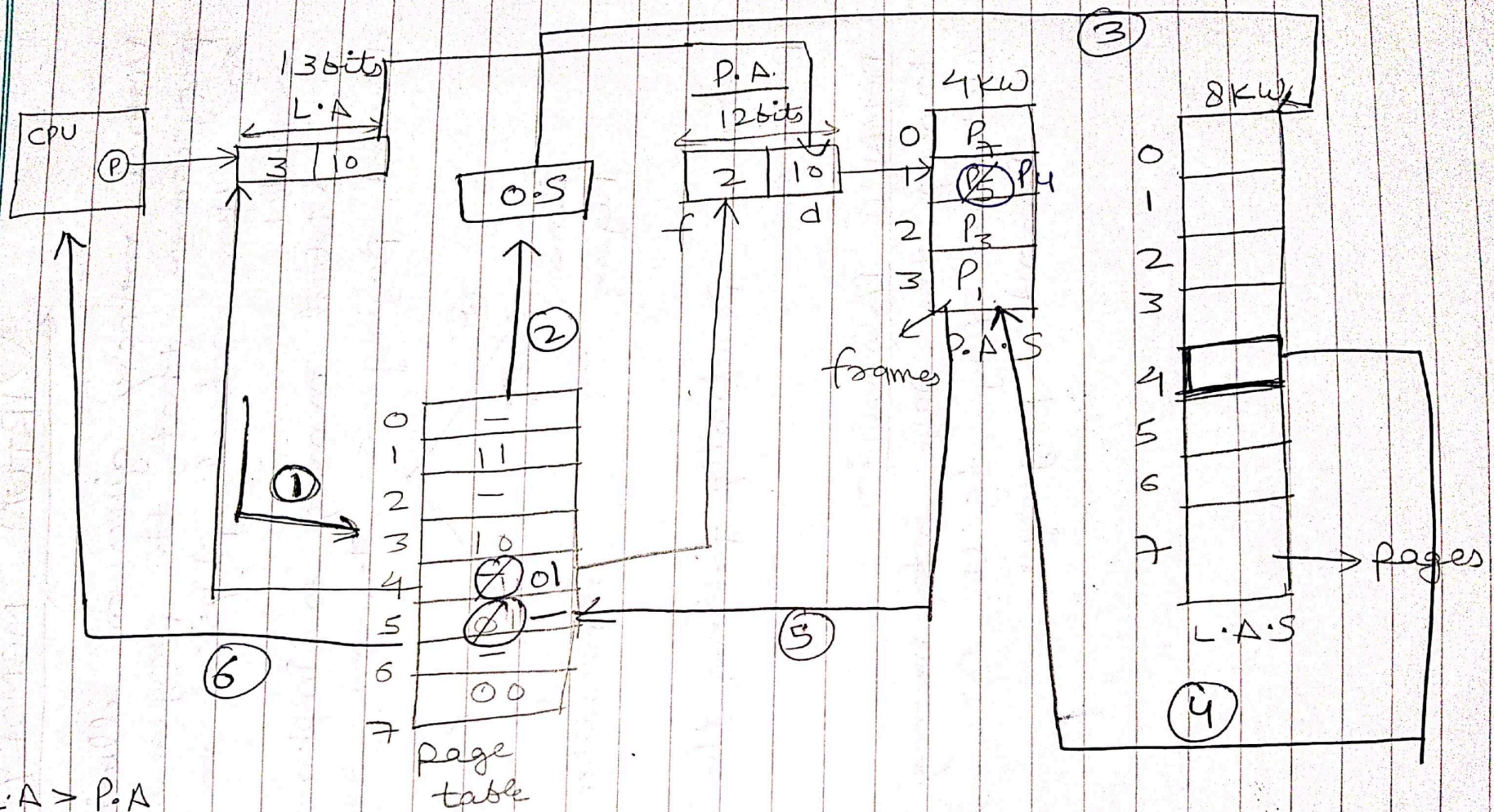
Demand Paging

Demand Segmentation

Windows practically follows the concept of demand paging.

DEMAND PAGING :-

Loading the page into memory on Demand (whenever page fault occurs) is called as a Demand paging.



1.) L.A > P.A

2.) CPU => L.A.

Step 1:- The CPU is trying to refer the page in the main memory, but the page is currently not available in the P.A.S. (main memory). This is called as a Page fault.

Step 2: The program execution will stop and the signal will be sent to the operating system regarding the page fault.

Step 3: O.S. will search for required page in the LAS ("Virtual memory").

Step 4:- The required page will be brought from L.A.S. to P.A.S.
→ The PAGE REPLACEMENT ALGORITHMS will be used for the decision making of selecting the page for the replacement.

Step 5:- The respective page table entry will be updated accordingly.

Step 6:- The signal will be send to the CPU to continue the program execution and the CPU will access the required page in the main memory and continue the program execution.

PAGE FAULT SERVICE TIME (P.F.S.T.):-

→ The time taken to service a page fault is called as page fault service time.

→ Page fault service time includes time to perform all the above 6 steps.

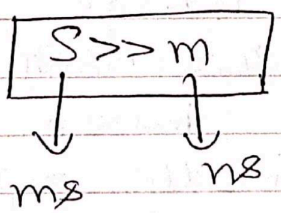
→ Page fault Service time = "S".

→ Main memory Access time = "M".

→ Page fault rate = 'P'.

Then the formula for effective memory access time.

$$EMAT = P \times S + (1-P) \times m$$



1.) Consider a system which has M.M.A.T of 35ns. P.F.S.T is 175ns. Page hit ratio is 75%. What is Effective memory access time?

Solve.) $EMAT = 0.25 \times 175 + (0.75) \times 35$

$= 43.75 + 26.25$

$= 70.00$ ns.

Workbook (P-111)

20)

$$EMAT = \frac{1}{k} \times (i+j) + i \times \left(1 - \frac{1}{k}\right)$$

$$= \frac{i+j}{k} + \frac{i-i}{k}$$

$$= \frac{i+j+ik-i}{k}$$

$$= \left(i + \frac{j}{k}\right) \quad (b)$$

gate P-113. 037)

options a.) 35ns
b.) 80ns
c.) 27ns
d.) 23ns

m = 20ns
S = 10ms
Page fault = $\frac{1}{10^6}$ (D)

1 second = 10^9 nano seconds

1 second = 10 ms

1 second = 10^6 micro

$= 10^{ms} \times \frac{1}{10^6} + \left(1 - \frac{1}{10^6}\right) \times 20ns$

$= \frac{1}{10^6} \times 10 \times 10^6 ns + \left(1 - \frac{1}{10^6}\right) \times 20ns$

$= 10 + 19.9999 \times 10$

$= 130$ ns.

Page-110

Q13.)

S = 8ms.	S = 20ms
Modified = 0	Modified = 1

M = 1ms, Ma = 70% of the time.

Effective = 2ms.

$$2ms = x(0.3\times 8 + 0.7\times 20) + (1-x)\times 1ms$$

$$2ms = x(2.4 + 14) + (1-x)$$

$$= x(16.4) + (1-x)$$

$$2 = 16.4x - x + 1$$

$$1 = 15.4x$$

$$\frac{1}{15.4} = x$$

$$x = 0.064$$

Workbook

Q21.)

S = 200ms.

m = 10ms.

TLB

E.M.A.T = ?

TLB = 80%	Not TLB = 20%
	2% cause page fault.

TLB hit, TLB time, MMBT

P.f.st

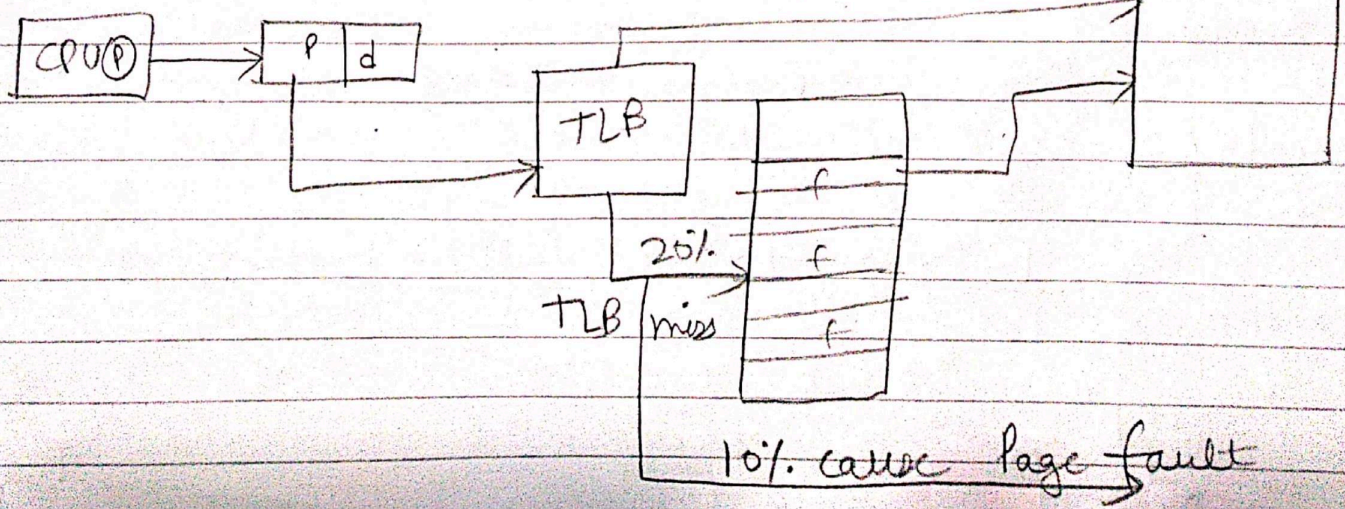
E.M.A.T = $0.80(0+10) + 0.20(0 + 0.10\times 200ms + 0.90\times 10)$

=

TLB miss, TLB time, P.f bat

T.L.B. hit 80%

= 132



TLB hit TLB Miss F.o.f.S.T

E.M.A.T = 0.80 (0 + 10) + 0.20 (0 + 0.10 × 200 ms

TLB time MMAT TLB Time P.f. rate

+ 0.90 × 10) = 13.8 ms.

No PF MMAT

Q) The amount of memory on a system is inversely proportional to page fault Rate. Each time when memory doubles then, the page fault rate is reduced by third ('p' is mate as p/3). Currently system has 32MB of memory and Page fault Rate is 2%. The main memory access time is 500 ns, and E.M.A.T is 300 μs. If the memory is increased to 128MB, then what would be the EMAT ____ in μs?

Ans.) EMAT = p × s + (1 − p) × m

300×10^3 ns = 0.02 × s + (1 − 0.02) × 500 ns

s = 14975500 ns

Memory	Page fault	EMAT
32 MB	0.02	300 μs
64 MB	0.02/3	
128 MB	0.02/9	

$P \times S + (1-P) \times M$

$$EMAT = \frac{0.02}{9} \times 14975500 + \left(1 - \frac{0.02}{9}\right) \times 500 \text{ ns}$$

$$= 33777.7777 \text{ ns}$$

$$= 33.777 \text{ µs}$$

33.77 — 33.76, 33.78

CCL + Speed

PAGE REPLACEMENT ALGORITHMS:-

Reference String:- page no.s referred by the process in the Logical Address.

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

NOTE:- The No. of frames allocated to the process is decided by Instruction set architecture.

FIRST IN FIRST OUT:-

⇒ 4 frames allocated to the process

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
			2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1
		1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	4	4	4	4	4	4	4	4	4	4	7	7	7
7	7	7	7	7	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2
F	F	F	F	F		F				f			F	F		f			

Total page faults = 10

⇒ 3 frames allocated to the process

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
		1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1
	0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1	1	0	0
7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
F	F	F	F		F	F	F	F	F	F			F	F		f		f	f

Total page faults = 15

Reference string:- 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
solve using 3 frames
& 4 frames

① Using 3 frames

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

Page fault = 9

② Using 4 frames

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

Page fault = 10.

3 frames Page fault = 9 } Belady's anamoly.
4 frames " " = 10 }

Belady's Anamoly : By increasing no. of frames to the process, the no. of page fault should decrease, but instead they are increasing. This problem is called as Belady's anamoly.

OPTIMAL PAGE REPLACEMENT :-

In the event of page fault, replace the place which is not used for longest duration of time in future.

Reference string :- 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

4 frames.

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

Total faults = 8

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

total faults = 9.

LEAST RECENTLY USED :- (LRU page replacement):-
In the event of page fault, Replace the page which is least recently used.

Reference string :- 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7 0 1

3 frames

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

total Page faults = 12

4 frames

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

total Page faults = 8

MRU Page replacement :- (Most recently used) :-
In the event of page fault, replace the page which is most recently used.

4 frames :-

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
		2	2	2	2	2	2	3	0	3	3	2	2	0	0	0	0	0	
			1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	0	0	0	0	3	0	4	4	4	4	4	4	4	4	4	4	4	4	
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	
F	f	f	f	f	F	f	F		F	f	F	F		F					

total page fault = 12

3 frames

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
		1	2	2	2	2	2	3	0	3	2	1	2	0	1	1	1	1	
	0	0	0	0	3	0	4	4	4	4	4	4	4	4	4	4	4	4	
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	0	0	
F	f	f	f	f	F	f	F		F	f	F	f		F			F		

total page faults = 16

Workbook

P-111

Q17) 0100, 0200, 0400, 0499, 0510, 0530, 0560, 0120, 0220, 0240, 0260, 0320, 0370.

Page size = 100 records.

Record No	Page No
0-99	0
100-199	1
200-299	2
300-399	3
400-499	4
500-599	5

1, 2, 4, 4, 5, 1, 2, 2, 3.

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

total 7 page faults.

NOTE:→ If there is only 1 frame allocated to the process, then every page referred will be a page fault.

→ The referencing will never contain continuous same page.

Eg:- 1, 2, 3, 4, 4, 3, 2, 1 ✗

1, 2, 3, 4, 3, 2, 1 ✓

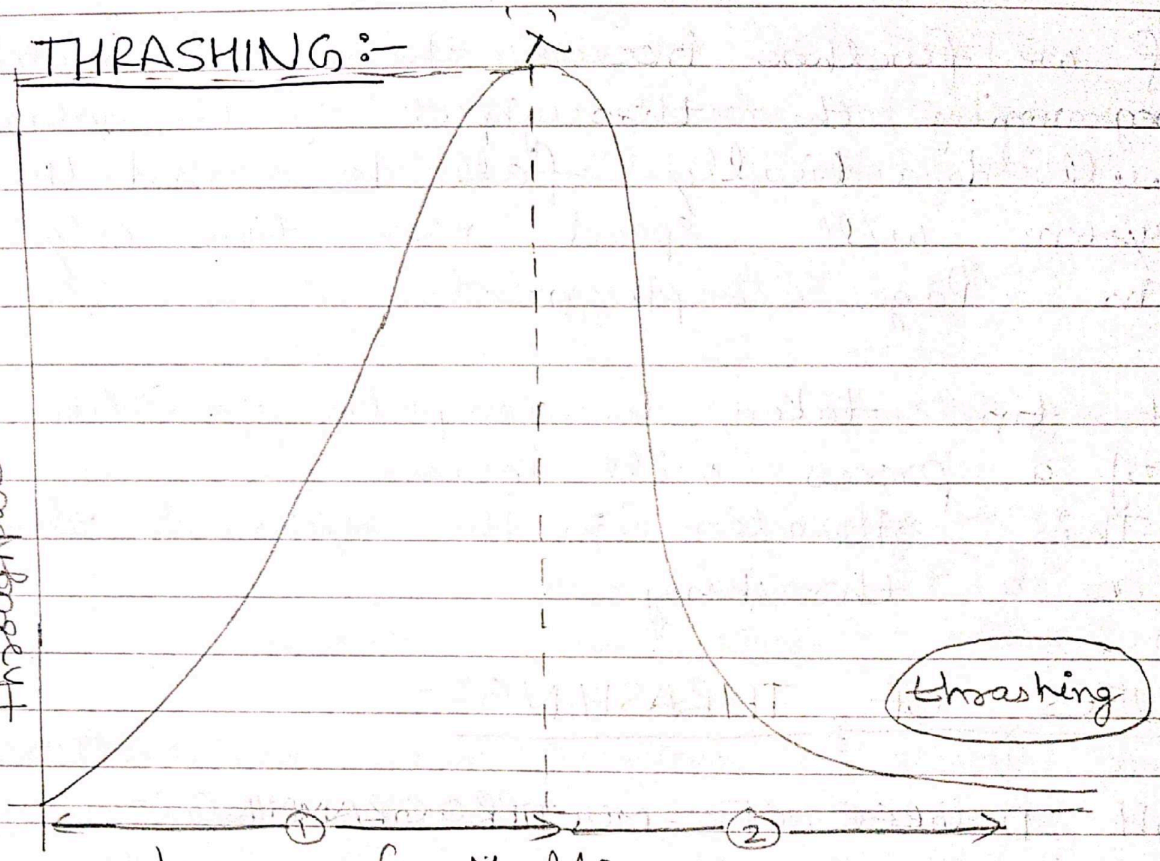
P-110 Q12.) 1 → n pages → 5, (C) m, n.

m

min – m.

max – n.

THRASHING :-



Eg:-

Free Frames = 400

① [No of processes ⇒ 100
Each process gets = 4 frames]

② [No of processes ⇒ 400
Each process gets = 1 frame]

NOTE :- → In a initial degree of multiprogramming upto the point of λ, the CPU utilization & throughput of the system are very high and system resource are utilized perfectly.

→ If we further increase the degree of multiprogramming, the throughput of the system will drastically fall down and the system will spend more time only in Page Replacement.

→ The time taken to completion execution of a process will increase.
This situation in the system is called as a thrashing.

CAUSES OF THRASHING :-

1.) HIGH DEGREE OF MULTIPROGRAMMING :-

2.) LACK OF FRAMES (MAIN MEMORY)

RECOVERY OF THRASHING :-

1.) Do not allow the system to go into thrashing and instruct the long term scheduler, not to bring the processes into the memory, after the point of λ.

2.) If the System is already in the thrashing, then instruct the mid-term scheduler to suspend some of the processes, so that we can recover the system from

Thrashing:

Q.) Consider a system with the following parameters.

CPU Utilization → 10%.
Paging Disk → 90%. → Time spent on Page replacement.

which of the following factors will improve the CPU Utilization.

1.) Install bigger Disk. — NO
2.) Install faster CPU. — NO
3.) Increase degree of Multiprogramming. — NO
4.) Decrease " " " — YES
5.) Install More Main memory. — YES
6.) Decrease Page size. — NO
7.) Increase Page size. — ~~YES~~ LIKELY to increase.

NOTE:→ The main purpose of virtual memory is by allocating less memory to the process, we are trying to execute large size process and more number of processes. Indirectly we are trying to increase Degree of Multiprogramming.

→ If the virtual memory is removed, then efficient implementation of Multiprogramming and multiuser no longer possible.

FILE SYSTEMS:-

File:- Collection of logically related entities (records).
File will have various attributes.

FILE SYSTEM

Attributes:-

1.) Name
2.) Type
3.) Size
4.) location
5.) Creation Date
6.) last modified date
7.) permissions
8.) owner
9.) password.

File Context. → Store → F.C.B (File Control Block)

File Context will be store in F.C.B.

TYPES:-

1.) .doc
2.) .Txt
3.) .obj
4.) .exe
5.) .dll
6.) .JPg
7.) .png
8.) .c/.cpp/.Java/.HTML/.php/.Javascript
9.) .Xlsx
10.) .mp3/.mpy
11.) .avi/.FLV/.mkv/.3gp

OPERATIONS PERFORMED ON THE FILE:-

1.) Creation
2.) open
3.) write
4.) read
5.) modified
6.) hide
7.) truncate
8.) Rename
9.) Save
10.) Save as

11.) Delete
12.) Copy
13.) paste
14.) Cut
15.) Undo
16.) Redo

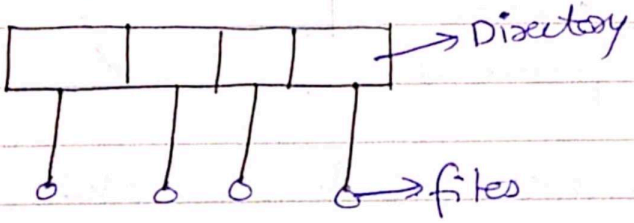
ACCESS METHODS:-

1.) Sequential Access.
2.) Random Access.

NOTE:- for the better classification of files, files will be stored in the directory.

DIRECTORY STRUCTURE:-

1.) Single Level Directory:-

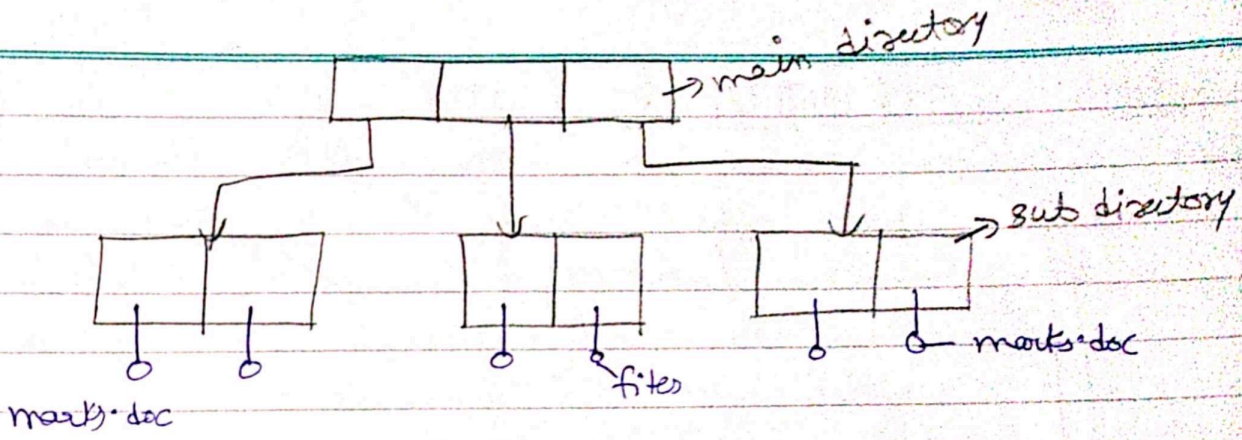


→ Implementation of the directory structure is easy or simple.

→ Two files cannot have the same name.
→ Searching time for the files will be more.

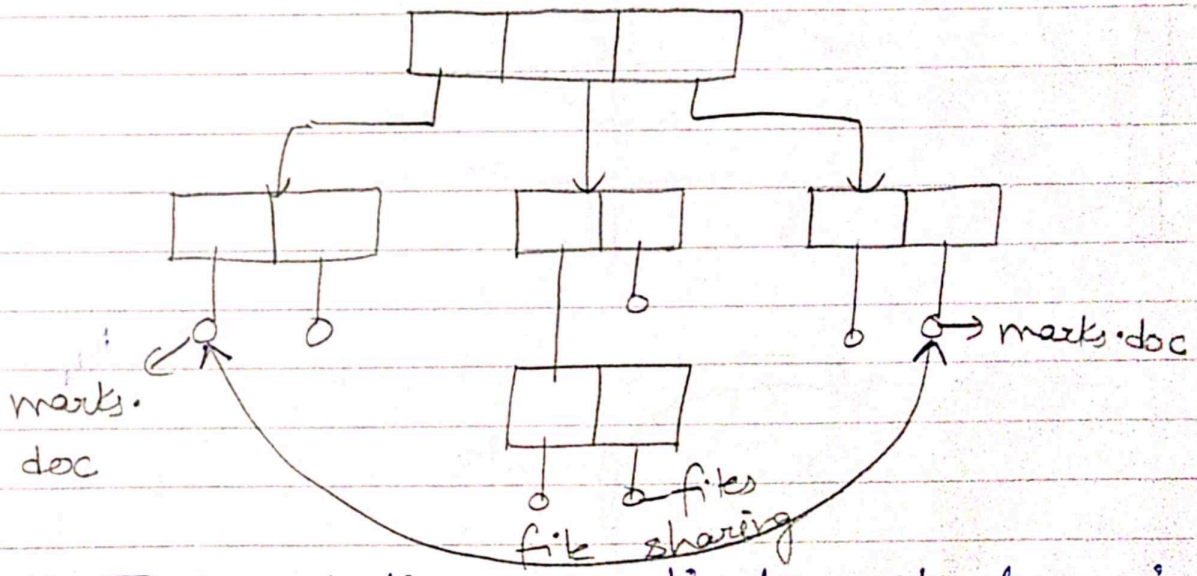
2.) Multi Level Directory (Tree level directory):-

→ The implementation of the directory structure is difficult or complicated.
→ Better classification of the files as per the criteria.
→ Searching Time for the files will be less.
→ If the same file exist in the 2 different directories, if one file is updated then the



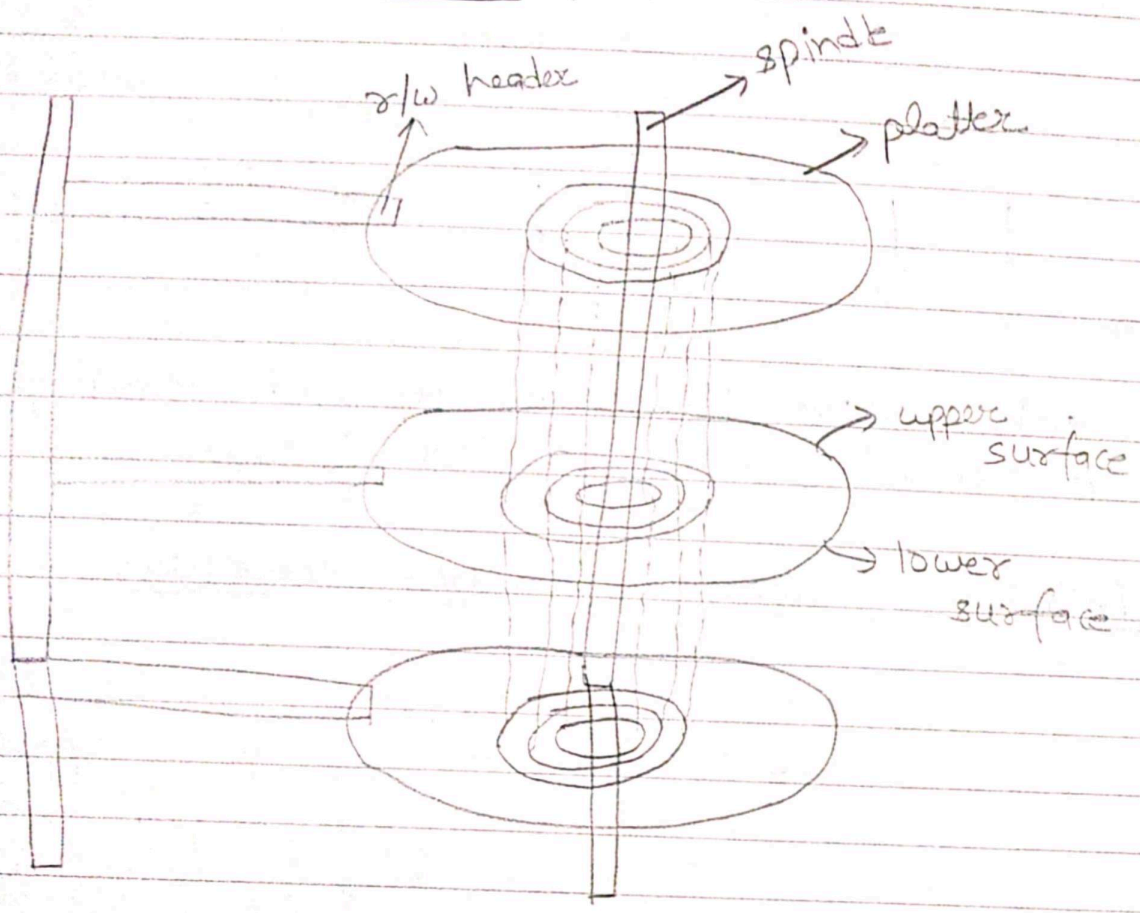
other file has to be updated accordingly, otherwise there will be a inconsistency.

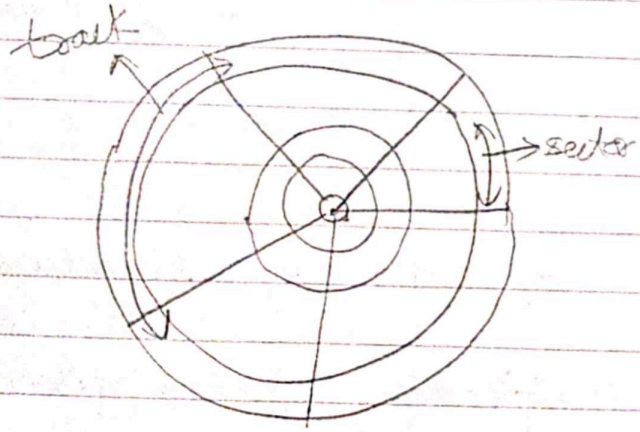
ACYCLIC GRAPH DIRECTORY STRUCTURE:-



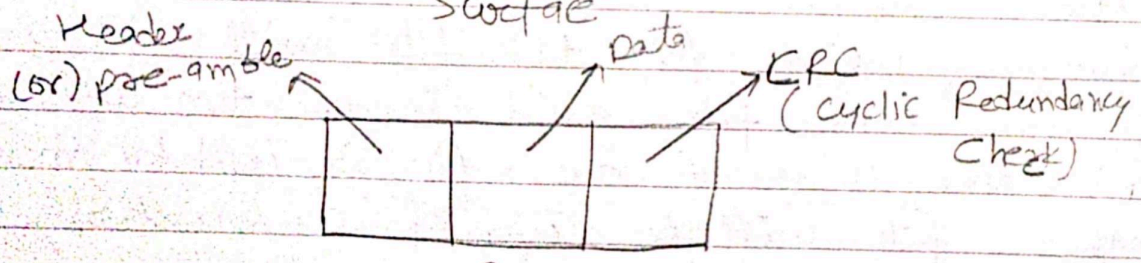
→ Implementation of directory structure is difficult.
→ The better classification of the file as per the criteria.
→ Searching time for the files will be less.
→ If the same file exist in the two different directories, then if one file is updated then the other file will be updated automatically by using file sharing concept.
→

DISK STRUCTURE:-





Surface



Sector
(or)
Disk Block.

Q.) Consider a disk which has 16 platters and each platter is having 2 surfaces and each surface is divided into 1K tracks. And Each track is further divided into 512 sectors & each sector can store 2 KB data. Then Calculate

i.) Capacity of the disk.

ii.) How many bits are required to identify the specific sectors of a disc.

Ans.) (i) = $2^4 \times 2^1 \times 2^{10} \times 2^9 \times 2^{11}$

$= 2^{35}$ B

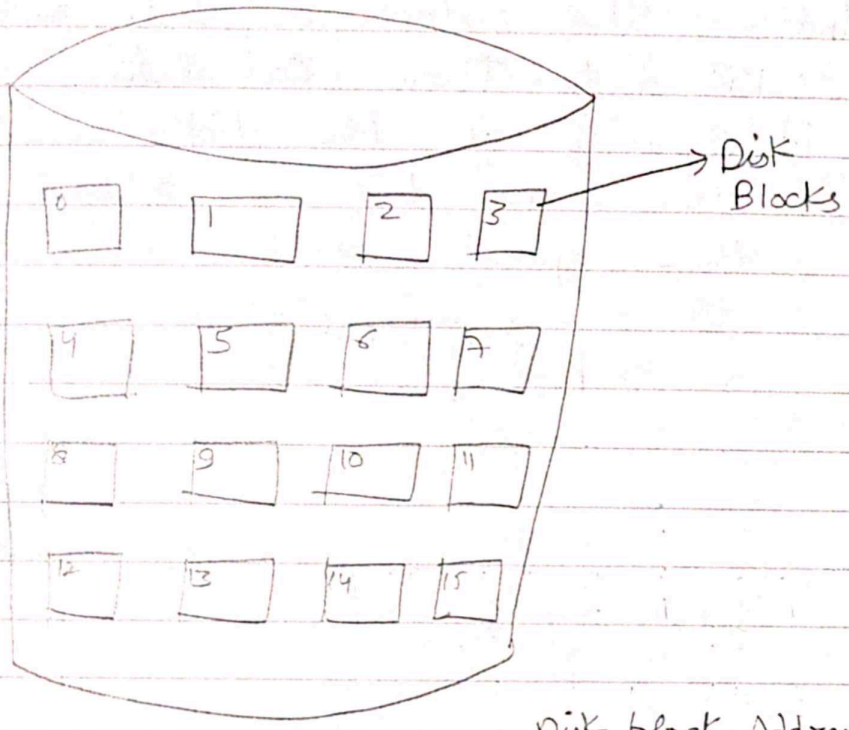
$= 32$ GB.

(ii.) $4 + 1 + 10 + 9$

$= 24$ bits.

DISK SPACE ALLOCATION METHODS :-

1.) Contigous Allocation :-



Disk block Address

File	Starting D.B.A	Size → Cort No of Disk Blocks.
abc.doc	2	4
xyz.doc	9	5

→ In the Contiguous Allocation, whenever the file is created, the disk blocks are allocated in a continuous manner.

→ Every file is associated with the 2 parameters :-
1.) Starting DBA
2.) Size (Cort No of Disk Blocks).

→ Increasing the file size may not possible always.

→ It is suffering from external fragmentation.

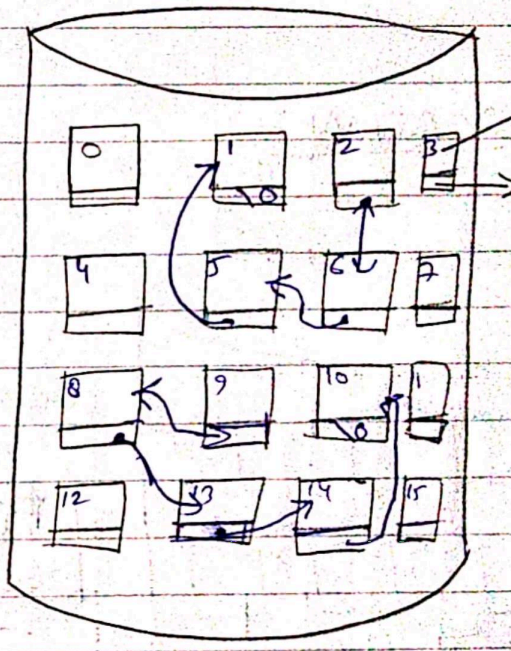
→ Internal fragmentation may exist in the last disk block of the file.

→ It supports both sequential and Random access of the file.

→

Starting D.B.A	+ offset ⇒	Random Access

LINKED ALLOCATION:- (Non-contiguous)



File	Starting DBA	Ending D.B.A
abc.doc	2	1
xyz.doc	9	10

→ In the linked allocation, whenever the file is created, disk blocks are allocated in a non-contiguous manner.
→ Every file is associated with the two parameters.
  i) Starting block address.
  ii) Ending block address.
→ Increasing the file size is always possible, if the free disk block is available.
→ There is no external fragmentation.
→ Internal fragmentation may exist in the last

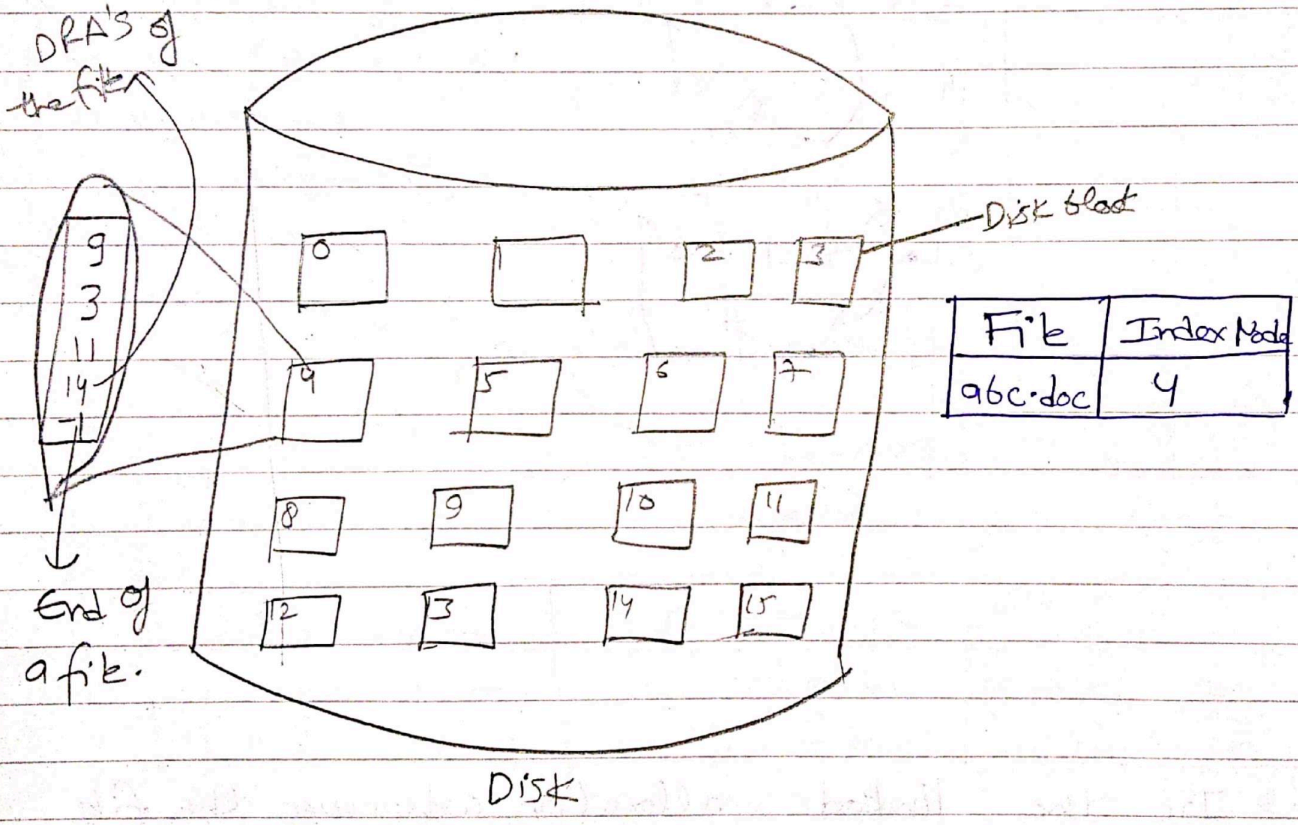
disk block of the file.

→ There is a overhead of maintaining the pointer in every disk block.

→ If the pointer of any disk block is lost, then the file will be truncated.

→ It supports only sequential access of the file.

INDEXED ALLOCATION :-



→ In the Indexed allocation, whenever the file is created, every file is associated with its own indexed node.

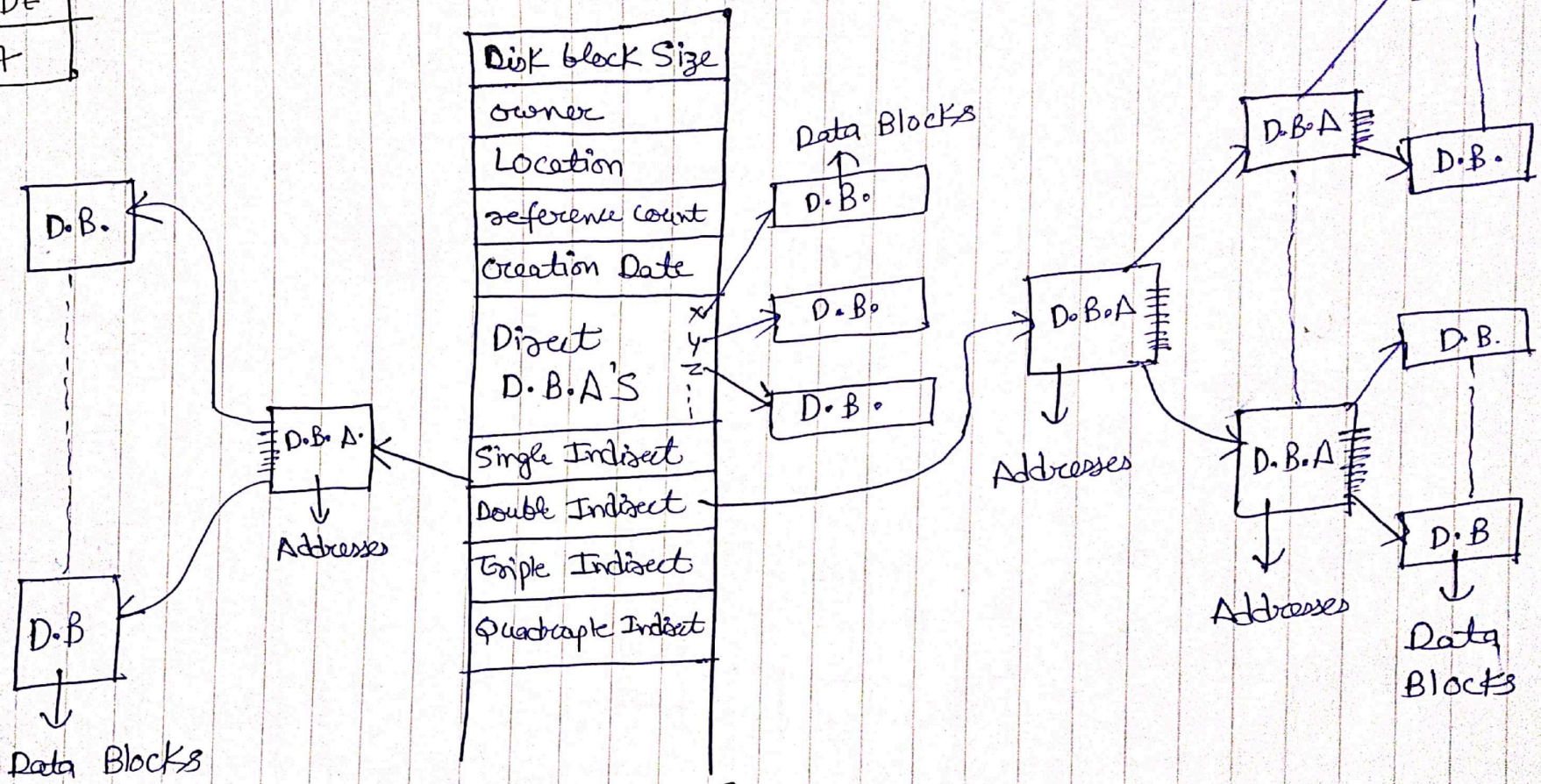
→ Indexed node contains all the disk block addresses of the file.

→ If the file is very very large, then one disk block may not be sufficient to store all the disk block addresses.

→ If the file is very very small, it is waste of using entire one disk block, just to store the addresses.
→ There is no External fragmentation.
→ Internal fragmentation may exist in the last disk block of the file.

File	I-NODE
abc.doc	27

UNIX-I-NODE ← Implementation:- (An Extension of indexed allocation →)



I-NODE #27

$$\left[\text{No. of Direct D.B.A'S} + \left(\frac{\text{D.B. Size}}{\text{D.B.A}}\right) + \left(\frac{\text{D.B. Size}}{\text{D.B.A}}\right)^2 + \left(\frac{\text{D.B. Size}}{\text{D.B.A}}\right)^3 + \cdots\right] \times \text{DB Size}$$

No. of single indirect D.B.A's

No of Double Indirect D.B.A's

No of triple indirect DBA's

→ Unix follows i-Node way of implementation in order to allocate disk space to the files.

→ Every file is associated with its own i-node.

$$\frac{\text{D.B. Size}}{\text{D.B. A.}} = \text{No. of Disk Block's one can contain}$$

Q1.) Consider a system which maintains Unix i-node which have 10 direct DBA, 1 single indirect, 1 double indirect, 1 triple indirect DBA.

D.B size is 1KB.

D.B. Address requires = 32 bits, then what is a maximum file size possible with the triple indirect.

a.) 2 GB
b.) 4 GB
c.) 8 GB
d.) 16 GB

$$= \left(\frac{\text{D.B.Size}}{\text{DBA}}\right)^3 \times \text{DBsize}$$

$$= \left(\frac{2^{10}}{4}\right)^3 \times 2^{10}\ \text{Byte}$$

$$= \left(\frac{2^{10}}{2^2}\right)^3 \times 2^{10}$$

$$= (2^8)^3 \times 2^{10}$$

$$= 2^{24} \times 2^{10}$$

$$= 2^{34}$$

$$= \boxed{16\text{GB}}.$$

2 marks
Gate 2019

Q2.) The Unix -i-node have 12 direct, 1 single, 1 double indirect pointers (D.B.A). The Disk block size is 4KB. The D.B.A is 32 bits. What is maximum file size possible (Rounded off to 1 Decimal) 4.0 in GB.

Ans.) $\left[12 + \left(\frac{4KB}{4B}\right) + \left(\frac{4KB}{4B}\right)^2\right] \times 4KB$

$= \left[12 + 2^{10} + 2^{20}\right] \times 2^{12} b$

$= \left(12 + 2^{10} + 2^{20}\right) \times 2^{12} b$

~~$= 12 * 2^{12}$ bytes $+ 2^{42}$ bytes~~

~~$= (12 \times 4)$ KB $+ 2^{12}$ GB~~

~~$= 48KB + 4 \times 1024$ G~~

$= [12 + 1024 + 1048576] \times 2^{12} B$

$= [1049612] \times 2^{12} B$

$= \left[\frac{1049612}{2^{20}}\right] \times 2^{12} B$

$= 2^{20} \times 2^{12} B$

$= 2^{32} B$

$= \boxed{4.0 GB.}$

Workbook P-117

Q7.)

10 – direct DBA
1 – double Indirect
1 – triple
1 – quadruple.

D.B. size → 1KB = 2^{10} Byte.
1TB = 2^{40}
4TB = 2^{42}.

$$\left(\frac{\text{D.B. Size}}{\text{DBA}}\right)^4 \times \text{DB size} = 2^{42}$$

$$\left(\frac{2^{10}}{\text{DBA}}\right)^4 \times 2^{10} = 2^{42}$$

$$\frac{2^{40} \times 2^{10}}{2^{42}} = (\text{DBA})^4$$

$$\frac{2^{50}}{2^{42}} = (\text{DBA})^4$$

$$2^8 = (\text{DBA})^4$$

$$(2^2)^4 = (\text{DBA})^4$$

DBA = 4bytes

DBA = 4bytes
= 32bits.

Disk Free Space Management:-

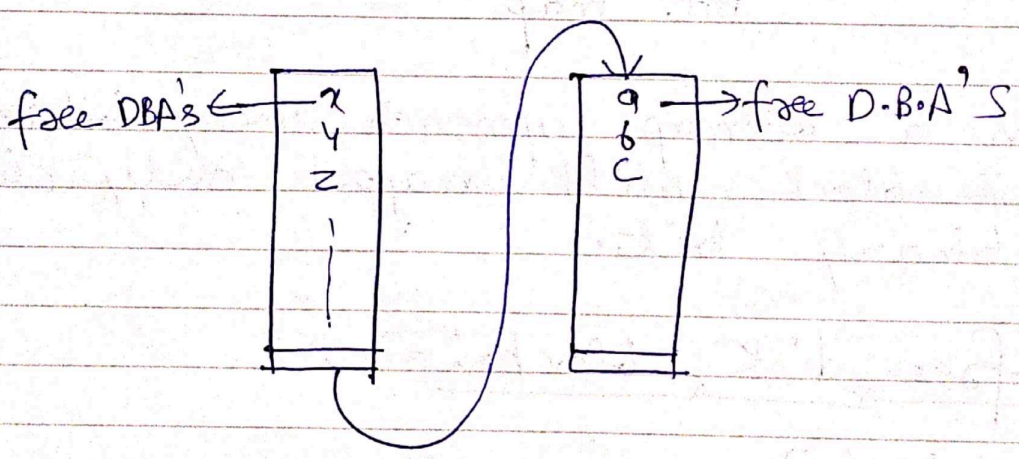
Size of Disk = 20MB
Disk Block Size = 1KB
D.B.A = 16 bits.

No. of disk blocks available on the given Disk:-

$$= \frac{\text{Size of Disk}}{\text{Disk block Size}} = \frac{20MB}{1KB} = \frac{20 \times 2^{20}}{2^{10}} = 20 \times 2^{10}$$

$$= \boxed{20K}$$

1.) Free list Approach:-



→ In the free list approach, some disk block are used to store the free disk block addresses.

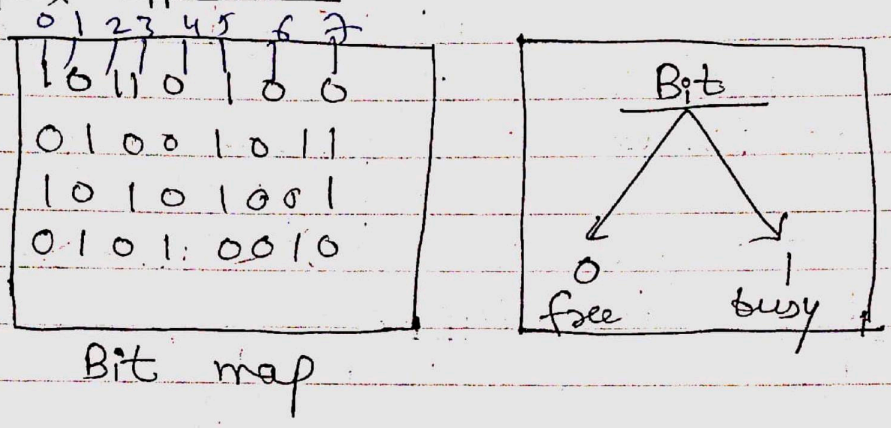
→ No. of disk block addresses possible to store in '1' Disk block.

$$\frac{\text{DB Size}}{\text{D.B.A}} = \frac{1KB}{2B} = \frac{2^{10}}{2^1} = 2^9 = 512.$$

2.) 512 D.B.A's ⟷ we can store in '1' D.B.
to store 20K D.B.A's ——— ?

$$\frac{20K}{512} \times 1 = \frac{20 \times 2^{10}}{2^9} = 20 \times 2 = 40$$

2.) Bit Map Approach:-



Bit map

→ In a bitmap approach, every ~~bitma~~ disk block will mapped only with 1 binary bit.

free disk blocks:-

1, 4, 6, 7

Busy disk blocks:-

0, 2, 3, 5

1.) No. of disk blocks, we can map in '1' Disk Block.
Disk block size = 1KB = 1K × 8 bits
= 8K bits.

2.) 8K disk blocks ⟷ we can map in '1' Disk block.
to map 20K disk blocks ⟶ ?

$$\frac{20K}{8K} = 2.5 = 3 \text{ blocks.}$$

17.)

Size of Disk = 40 MB.
Disk block Size = 2 KB.
D.B.A = 3 bytes = 24 bits.

1.) No of D.B's we can map in 1 Disk Block.

D.B.S = 2KB = 2K × 8 bits = 16 K bits.

2.) ~~$\frac{\quad}{16K} =$~~

No. of Disk Block available on given disk

$$= \frac{40MB}{2KB} = \frac{40 \times 2^{20}}{2 \times 2^{10}}$$

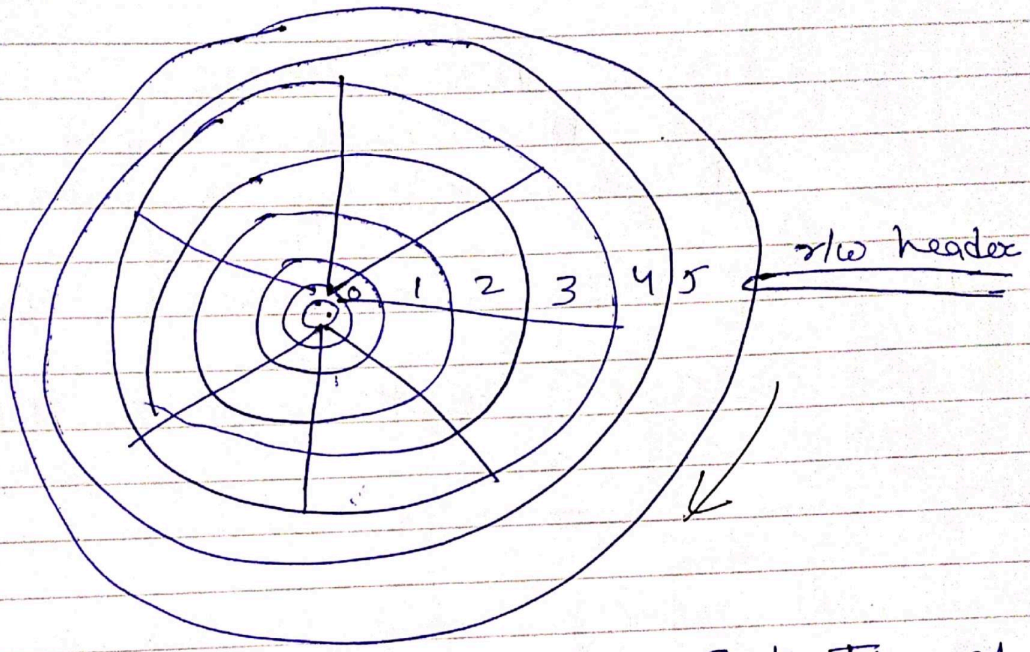
$$= 20 \times 2^{16}$$

2.) 16 K D.B ⟶ we can map 1 D.B. = 20K

$$\frac{20K}{16K} = 1.2 = \textcircled{2} \text{ blocks.}$$

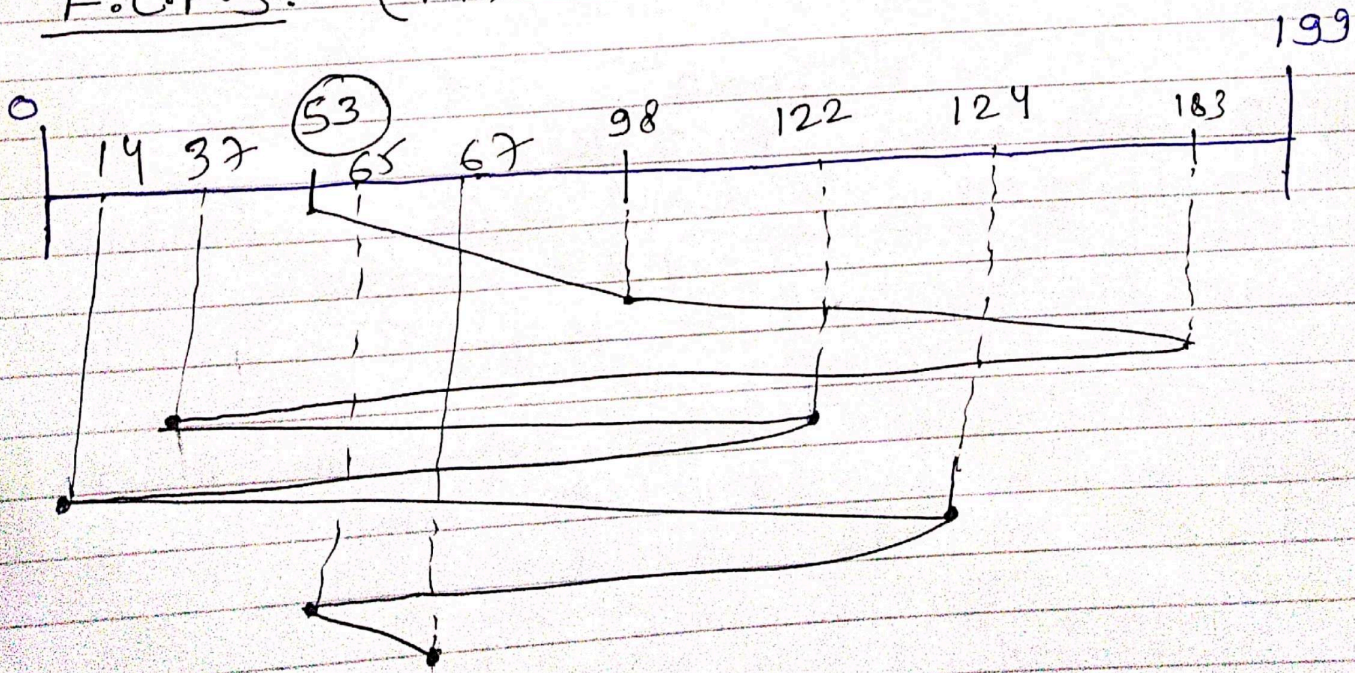
DISK SCHEDULING:-

Track Requests:-
98, 183, 37, 122, 14, 124, 65, 67.



GOAL:- To minimize the avg Seek Time of the Disk.

F.C.F.S:- (First come First Serve):



Total track movements made by r/w header (seek time):-

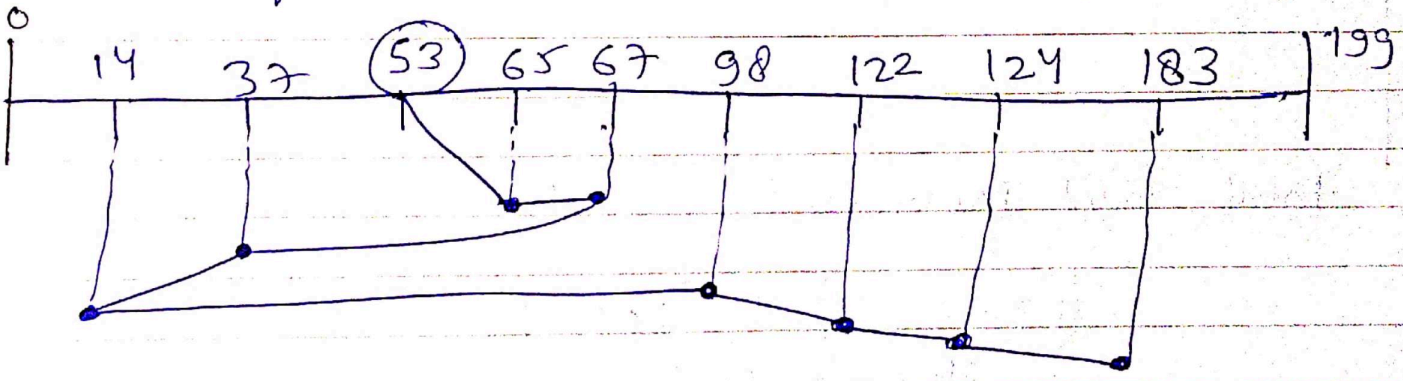
$= (98-53) + (183-98) + (183-37) + (122-37)$
$+ (122-14) + (124-14) + (124-65) + (67-65)$

$= 45 + 85 + 146 + 85 + 108 + 108 + 59$
$+ 2$

$= \boxed{670}$

SSTF (Shortest Seek Time first):-
(Nearest Track Next):-

Track Request:- 98, 183, 37, 122, 14, 124, 65, 67



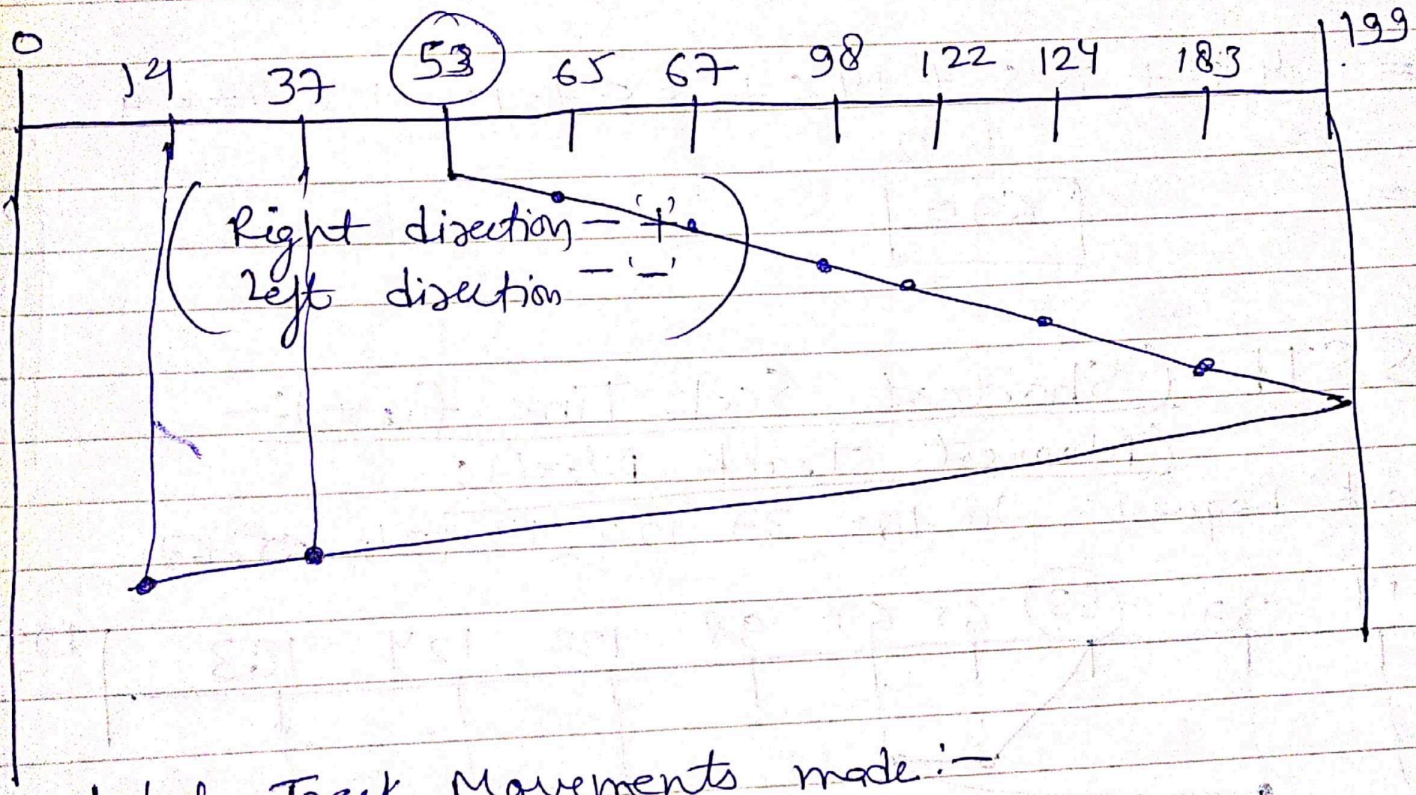
total tracks movements made:-

$= (67-53) + (67-14) + (183-14)$

$= 236.$

SCAN (Elevator):-

Track Request :- 98, 183, 37, 122, 14, 124, 65, 67

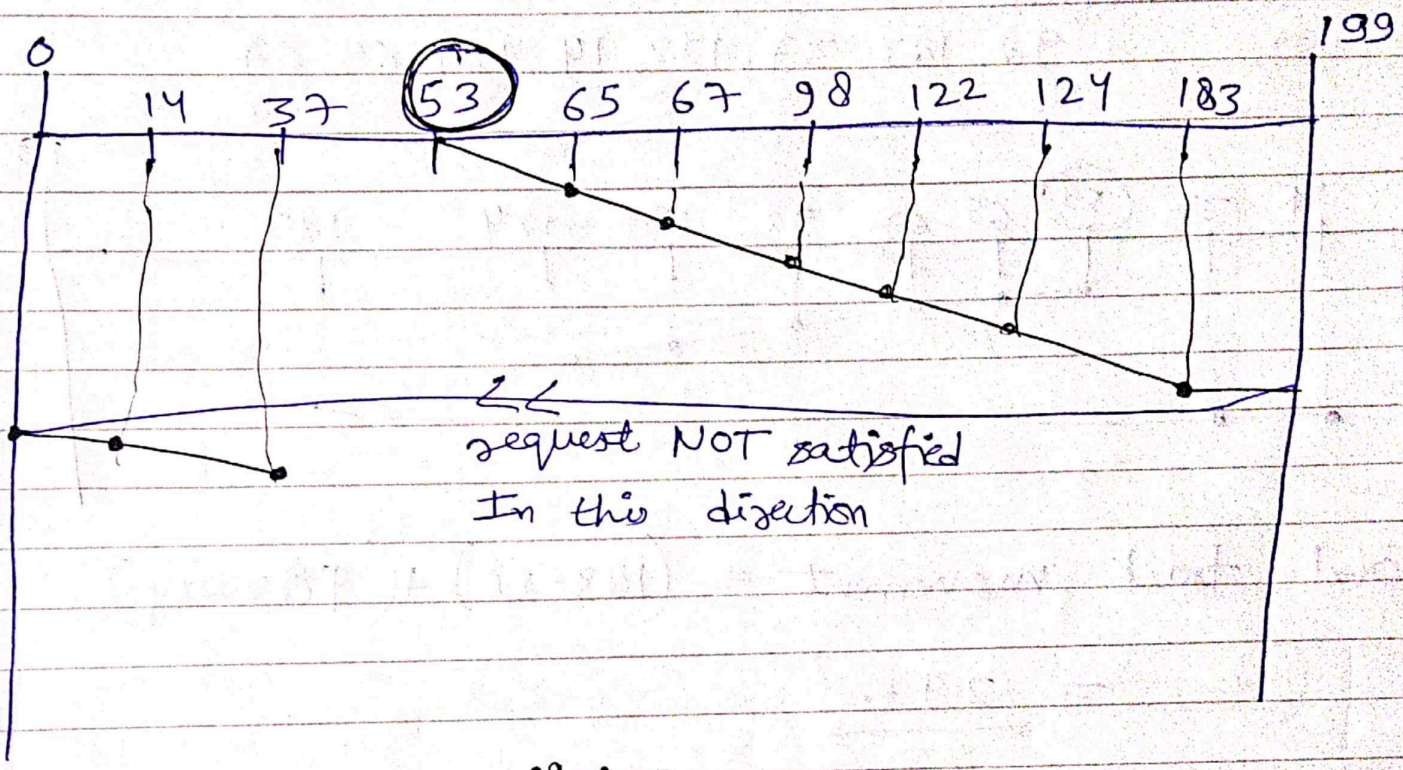


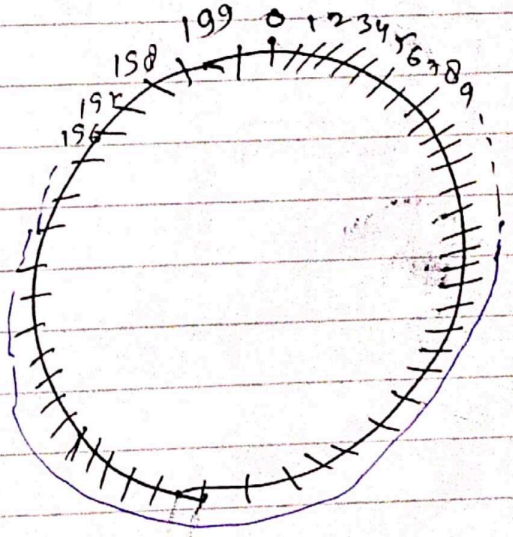
total Track Movements made :-

$$= (199-53) + (199-14)$$
$$= 146 + 185$$
$$= \underline{331}$$

C- Scan (Circular) :-

Track Request :- 98, 183, 37, 122, 14, 124, 65, 67.





total track Movements made :-

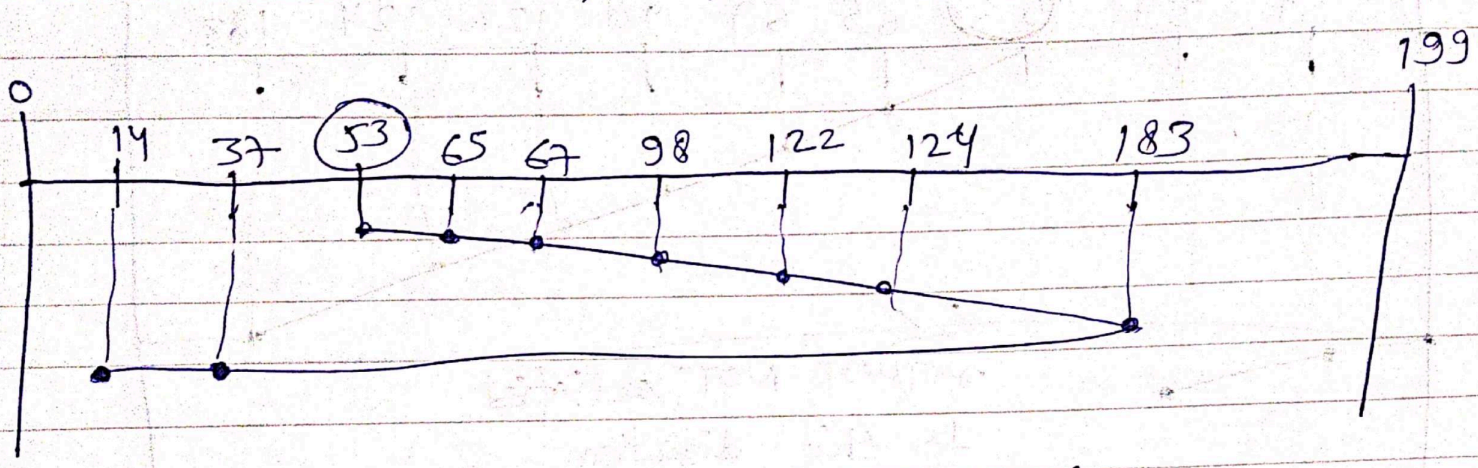
$= (199-53) + (199-0) + (37-0)$

$= 382.$

LOOK (Look for last pending request in the direction of r/w header):-

Track Requests:-

98, 183, 37, 122, 14, 124, 65, 67

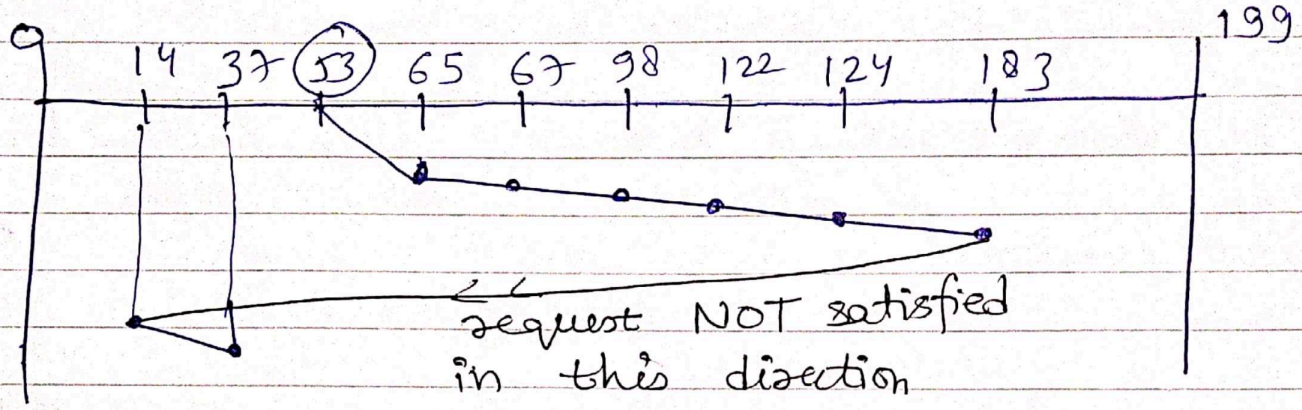


Total track movement = (183-53) + (183-14)

= 299.

Circular

C-LOOK :-



$$= (183-53) + (183-14) + (37-14)$$
$$= 130 + 169 + 23$$
$$= \boxed{322}$$

Kapil Yadav

M.Tech AI DTU || Computer Science Engineer || Ex -GeeksforGeeks || Influenced by few, Influencing few

Talks about #dsa, #coding, #datastructures, #problemsolving, and #softwareengineer

Delhi Technological University (Formerly DCE) • Delhi Technological University (Formerly DCE)

South Delhi, Delhi, India