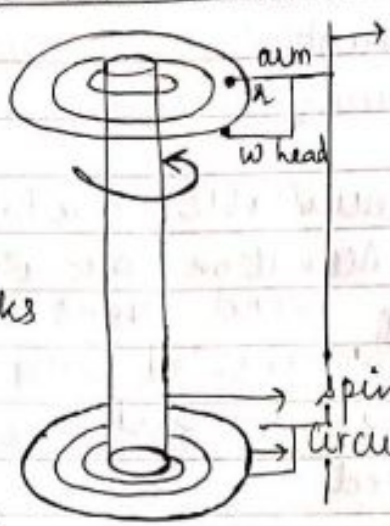
Memory Management, 26-27
I/O Manage, Disk, 1-2
File System. 3-3

IIIrd MODULE

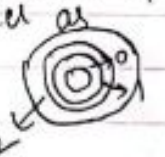
DISK MANAGEMENT.

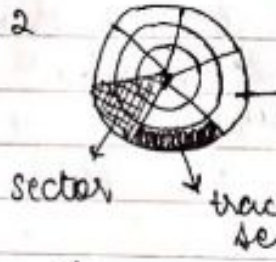


Each platter has 2 surfaces
Each surface has multiple tracks. and tracks divided into sectors
no of tracks and sectors on one surface = another
all platters have same structure
Data density increases as we move inside
Outer track sector has less data density as compared to innermost track sector
Each track sector stores same amount of data

* Each platter has got an arm and an arm has 2 r/w read write head.

5 platters → 10 surfaces
16 tracks.
↓
160 tracks
1 plate 4 sectors.
↓
Numbering from inner side of cylinder as





The arm has movement to and fro and two surfaces of same platter both have reader writer
disk rotates (platter circular movement)
7200 rotation / min
for data from track sector platter set, arm move, then read write

To read from one byte = platter rotation + seek time (arm) + transfer time (read write) + controller overhead

Scanned by CamScanner

Disk are secondary / tertiary storage system in memory hierarchy. It is an offline storage and data stored on disk are generally permanent. (temporary when used in virtual memory)

DISK STORAGE MECHANISM.

A disk contains multiple platters with two surface per platter (magnetic tape) each surface is divided into tracks and sectors and a read write head (R/W head). The read write head has to and fro motion to read the data from desired track sector. The platter rotates with the rotational speed measured in rpm (rotations per minute)

Disc access time = seek time + rotational latency + transfer time + controller overhead

(1.) The time taken to locate the disk arm to the specified track from ~~the~~ where the data is to be read or write is known as SEEK TIME. depends upon speed of arm assembly + ~~rotation of disk~~ no of tracks.

(2.) The time taken by desired sector of the disk to rotate into a position so that it can access read write head. is called rotational latency (minimum rotational latency desired)

depends upon:-

(a) revolutions per minute } (rotational bandwidth)
(b) bandwidth - }

** bandwidth → maximum rpms you can experience

(2) sector distance from read write head.

* sector distance ∝ rotational bandwidth.

(3) The time it takes to transfer the requested data is known as transfer time. depends upon majorly on amount of data (block ~~phase~~ size).

Scanned by CamScanner

* Note- Average Rotational latency for a disk is one half the amount of time it takes for the disk to make one revolution.

One Rotation = One revolution

A·R·L = $\frac{1}{2}$ (One revolution)

Ques. 7200 rpm (rotations/sec)

60 sec → 7200 rotations

1 sec = $\frac{7200}{60}$

1 sec = 120 rotation

1 rotation = $\frac{1}{120}$ sec = 1 revolution

Average R.L = $\frac{1}{2 \times 120}$ = $\frac{1}{240}$ sec.

Ques Consider a typical disk that rotates at 15,000 rotations per minute and has a transfer rate of 5×10^5 bytes/second If the average seek time of disk is twice the average rotational latency and controllers transfer time is 10 times the disk transfer time the average time to read or write 512 bytes sector will be ?

15000 rotation / min

60 sec → 15000

1 sec = $\frac{15000}{60}$ = 250 rotation

Average Rotational latency = $\frac{1}{250} \times \frac{1}{2}$

= $\frac{1}{500}$ sec = 2ms

Average seek time = $2 \times \frac{1}{500} = \frac{1}{250}$ sec = 4ms.

Scanned by CamScanner

Transfer rate (disk) = 5×10^6 bytes.

bytes : 512 = $\frac{5 \times 10^6}{5/2}$ $\frac{512}{5 \times 10^6}$

$= 10,24 \times 10^{-6}$ sec.

controller overhead = 1024 sec.

$= 1024 \times 10^{-6}$ sec

Average time = $0.002 + 0.004 + 0.001024 + 0.001024$

$= 71.264 \times 10^{-4}$ sec.

DISK SCHEDULING

FCFS
SSTF
SCAN
CSCAN
LOOK SCAN
C LOOK

Scan → bidirectional, one end

CSCAN → unidirectional both end.

LOOK - bidirectional no end.

CLOOK → unidirectional no end.

Need of disk scheduling →

① multiple I/O request by different processors, only one can be served at particular time ∴ scheduling required

② effective arm management

③ Hard disk are slower devices and need to be accessed in effective manner if not then whole effective speed will come to the slowest device and the slowest device will become dominant factor and will effect the effective speed overall speed is low.

Scanned by CamScanner

Consider a disk Q with the request for IO to block all cylinder

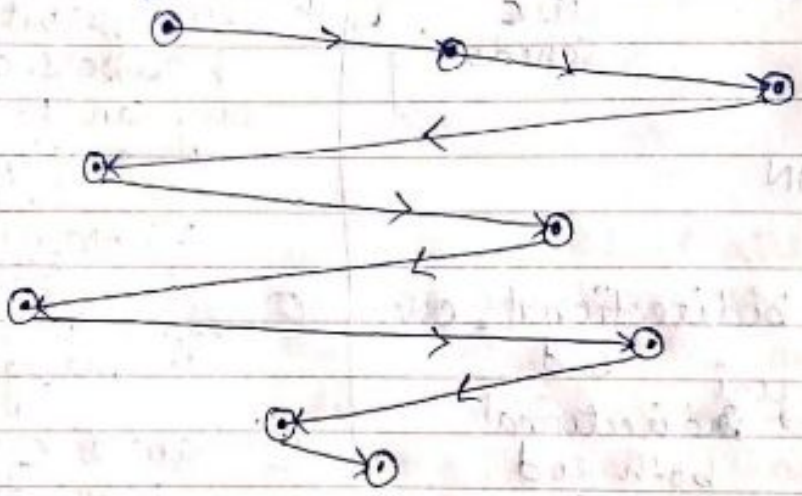
R → 98, 183, 37, 122, 14, 124, 65, 67

Currently the read write head is at cylinder 53. Calculate the total arm movement under following:- with 200 cylinders in total.

1. FCFS
2. SSTF
3. SCAN
4. C-SCAN
5. LOOK
6. C-LOOK

① FCFS. (First come First serve)

0 14 37 53 65 67 98 122 124 183 199



phele ascending m lekhlo. phir jaise jaisi request aarahi h waise jao.

Total arm movement

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

= 45 + 85 +

= 640 cylinders

② Shortest seek time first / Closest cylinder next algo.

Total arm movement:

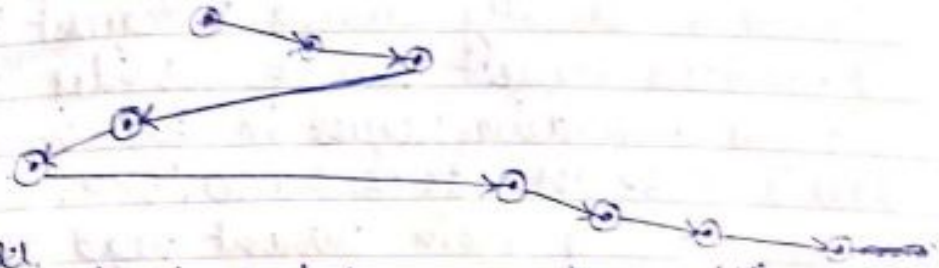
(65 − 53) + (67 − 65) + (67 − 37) + (37 − 14) + (98 − 14) + (122 − 98) + (124 − 22) + (183 − 124)

= 236 cylinders

Scanned by CamScanner

phele ascending order likho, phir dono side check karo jaha difference kam aaye waha chale jao

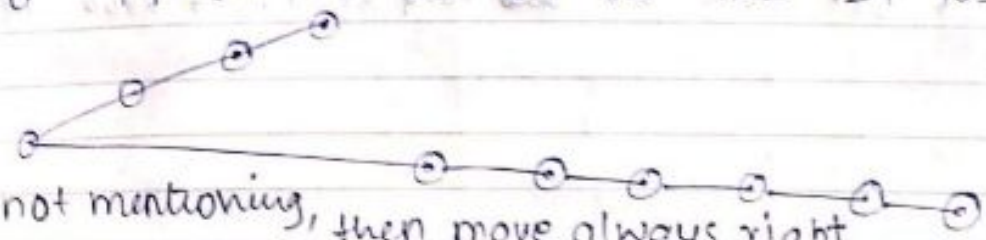
0 14 37 53 65 67 98 122 124 183 199



Total ead.

3) SCAN → bidirectional, aur end Elevator Algo
Jiski taraf pheli baar gaye the usthar ki taraf kaake end touch kaake aayenge.

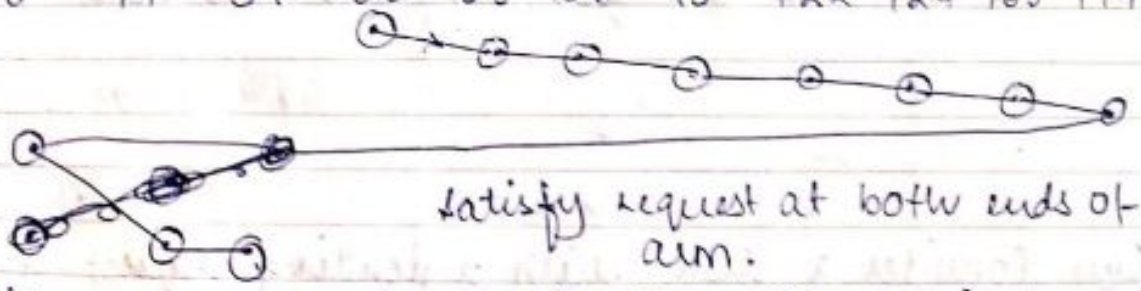
0 14 37 53 65 67 98 122 124 183 199.



if not mentioning, then move always right

4. CSAN. (unidirection and touch both end)

0 14 37 53 65 67 98 122 124 183 199

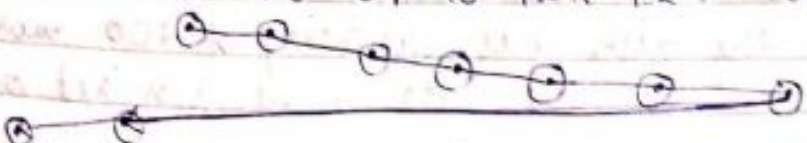


satisfy request at both ends of arm.

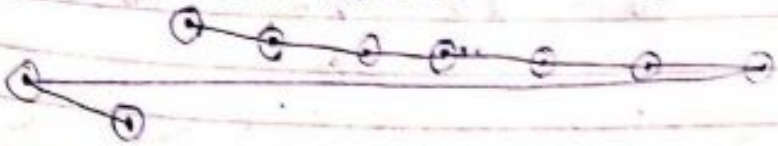
Arm movement - (65-53) + (67-65) + (98-67) + (122-98)
+ (124-122) + (183-124) + (199-183)
(199-0) + (14) + (37-14)

5. LOOK (bidirectional - touch no end)

0 14 37 53 65 67 98 122 124 183 199



CLOOK (unidirectional no end)



Scanned by CamScanner

Ans.

Suppose a disk drive has 5000 cylinders 0 to 4999. The drive is currently serving a request at 2150 and previous request was at cylinder 1805. The queue of pending request in fist in first out order is 2069, 1212, 2296, 2800, 544, 1618 356. Starting from current head position, what will be the total distance the disk arms moves to satisfy all pending request in scan and c look algo.

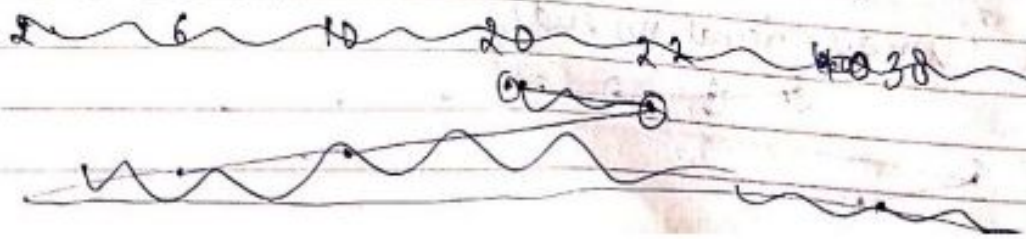
SCAN.

0 356 544 1212 1618 [illegible] 2069 2150 2296 2800 4999

CLOOK.

0 356 544 1212 1618 [illegible] 2069 2150 2296 2800 4999

Ques Consider a disk with a pending request of 10, 22, 20, 2, 40, 6, 38 Currently the read write head is at cylinder 20. Calculate the total seek time for the arm movement by applying FCFS scheduling and seek time per cylinder is 6 milli second and also the disk rotates 20 revolutions/ second, and has 100 words per sector. Each track of the disk has capacity of 400 words Then calculate the total time required to access one sector.



Scanned by CamScanner

FCFS m dobara serve karenge baki m nahi karenge

Arm movement = 10 + 12 + 2 + 18 + 38 + 134 + 32
= 146

Total seek time : 146 X 6 milliseconds
= 876 ms.

Rotational latency : $\frac{1}{2}$ x (time for one revolution)

1 sec → 20 rev
1 rev - $\frac{1}{20}$ sec

$\frac{1}{2} \times \frac{1}{20} = \frac{1}{40}$ s = $\frac{1000}{40}$
= 25 ms.

Transfer time. $\frac{400 \text{ words}}{100 \text{ words}}$ = 4 sectors on

1 revolution = 1 track = 400 words.
1 sec = 20 revolution
1 sec : 20 X 400
: 8000 words

1 sec = $\frac{100 \text{ words}}{8000}$
= 12.5 ms.

8000 words/sec → transfer rate.

Total time: seek time + Rotational latency + transfer time

876 ms + 25 ms + 12.5 ms

=

Scanned by CamScanner

20-11-17

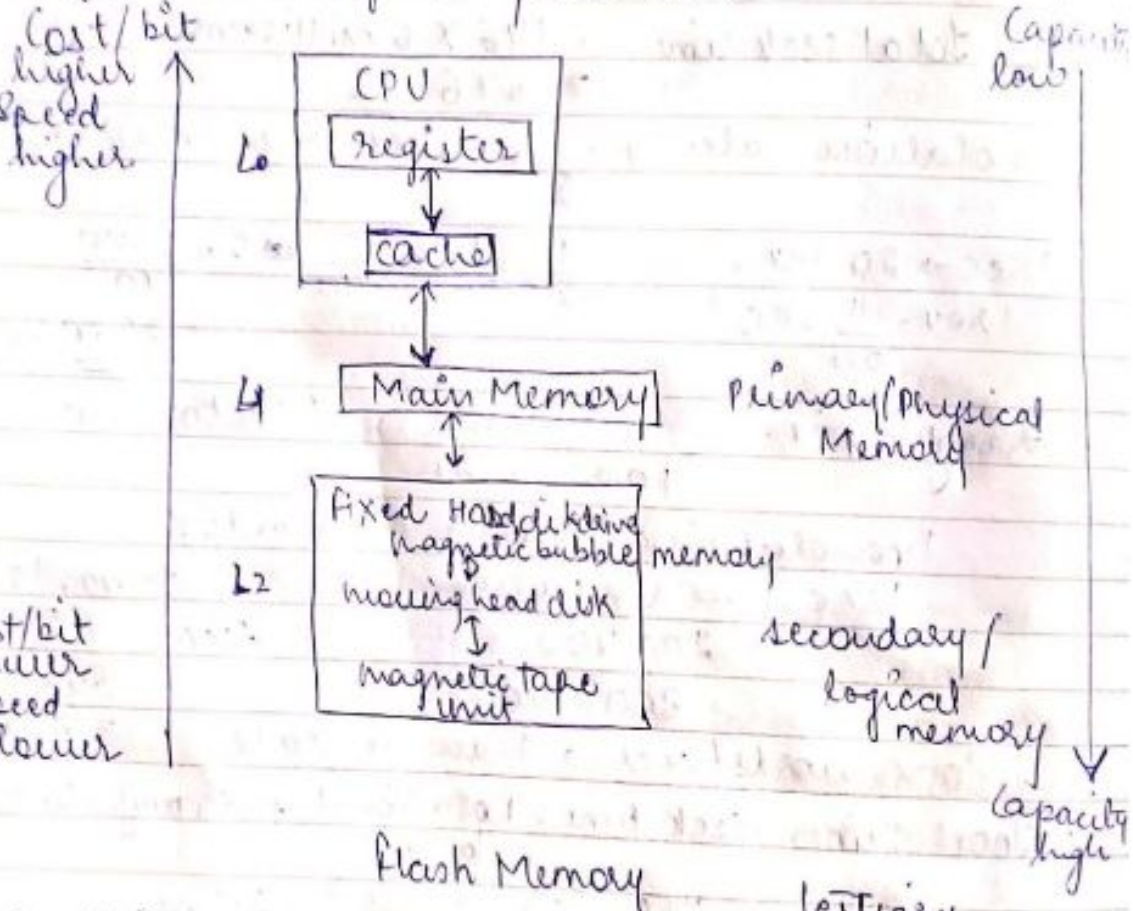
Memory Management

Memory Hierarchy

CPU
Register
Cache
Main Memory
Secondary Memory

Any content fixed property has 100% probability in secondary storage

Input & output from secondary but for processing of data, data flow up and down.



Flash Memory — tertiary

Cost/bit and speed increases as we move up and capacity decreases classified as configuration of hard disk and evolution

* At any level of the memory, the memory can interact only with its immediate neighbour Mi (memory at ith level) can interact with M_{i-1} and M_{i+1} only. Implies that secondary can never interact with cache.

Scanned by CamScanner

In OS, CPU directly interacts with Main Memory.

Performance of CPU always measured in terms of miss ratio.
Access time (t_i), Memory size(S_i), Cost (C_i), cost/bit
Bandwidth (b_i), Block size of transfer (X_i)
(rate of transfer)
Bit size m data transfer karega
Unit of transfer (X_i)

Sabse bada data transfer secondary memory se hoga (size badi h)
mexpensive (access time bad jayegi)

cache
$t_{i-1} < t_i$ access time- memory ka size kam, to access time kam
$S_{i-1} < S_i$ size neeche badega
$C_{i-1} > C_i$ cost upar
$b_{i-1} > b_i$
$X_{i-1} < X_i$

memory

A memory is a storage device, used for storing words (data and instructions), each on unique addresses (locations of memory). Memory is collection of memory locations.

Memory m bhut sari memory location hoti h or har location pe memory word hota h

Memory consists of large array or words or bytes each with its own address and CPU fetches instructions from memory according to the value of program counter.
After the instruction has been executed on the operands results are stored back into the memory.

For successful execution of a process the Operating system needs to ensure that each process needs

Scanned by CamScanner

to have seperate memory spaces. To seperate memory spaces, the range of legal addresses that a process may access must be determined and must be ensured that the process can access only these legal addresses.

If addres is not legal (not in range) then not successful execution and we have to trap it

address generated by CPU → logical address
address seen by memory corresponding to logical address is called physical address.

27-11-2019

for a

Contiguous.

- assign the consecutive block of memory to a process requesting for memory.
- Consecutive blocks of memory are allocated.
- do not have the overhead of address translation while execution.
- Process executes faster

Non contiguous.

- Assigns seperate memory blocks at different location in the memory space in a non-consecutive manner to a process requesting for memory.

Contiguous.	Non contiguous.
Memory space must be divided into fixed size partition (equal or variable) and each partition is allocated to a sigle process only. (memory space is divided)	Seperate block of memory is allocated overhead of address translation present process executes slower divide the process into several blocks and place them in diff

Scanned by CamScanner

part of memory according to the availability of memory space (process is divided)

Hole

The strip or chunk of available memory is known as whole. There may be multiple hole formation in memory

Fixed size Partition Allocation V/S Dynamic Allocation.

Fixed size - Memory is divided into fixed partition of equal or varying sizes. Each partition contains exactly one process. The partitions cannot be easily expanded or shrunged

Dynamic - Initially all memory is available and is considered as one large block of memory or hole. All memory is available for user process

User process has to return the allocated memory once its finishes its execution. Memory can be returned in any order without any relation to the order in which it was allocated. Hence multiple holes will be formed.

Memory Allocation Strategies :-

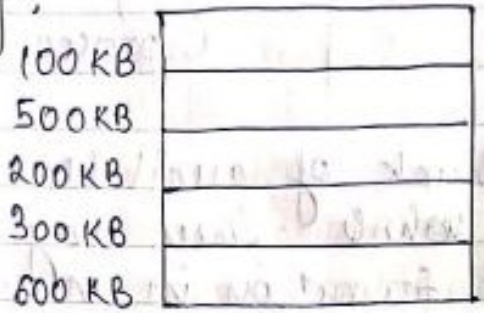
1. First fit
2. Best fit
3. Worst fit
4. Next fit

Ques Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, & 600 KB in order. How would the first fit, Best fit, worst fit & Next fit algorithm places the process of

Scanned by CamScanner

jisme memory wastage ↓.
most efficient storage

size 212, 417, 112, 426 KB in order? which algo makes the most efficient use of memory?



First fit - Inspite of any memory wastage, the first partition which could accomodate the process (as whole) give it to that.

212 KB (500 m ayega) ⇒ dedo jo waste horahi h hone do. uske next ke liye starting se dobara karo.

417 KB → (600 m ayega)

112 KB → (200 m ayega)

426 KB → cannot be accomodated now

100 KB	
500 KB	212 KB.
200 KB	112 KB.
300 KB	
600 KB	417 KB.

Total memory wastage : 100 KB + 288 KB + 88 KB + 300 KB + 183 KB

Best fit - allocate in that partition which has minimum memory wastage.

100 KB.	
500 KB	417 KB.
200 KB.	112 KB.
300	212
600	426 KB.

Scanned by CamScanner

Memory = 100 + 83 + 88 + 88 + 174 KB
= 533 KB

Worst fit fit process such that memory wastage is maximised.

100 KB	
500	417 KB
200	
300	112 KB
600	212 KB

and 426 KB cannot be accomodate.
Memory wastage: 100 + 83 + 200 + 188 + 388 =

Next fit After allocation, uske neeche se and memory in cycle is used.

	100
212	500
112	200
	300
417	600

and 426 KB cannot be accomodated.
100 + 288 + 88 + 300 + 183.

So Best fit algorithm makes the best use of memory.

Ques Given 6 memory partitions 200 KB, 400 KB, 600 KB, 500 KB, 300 KB, 250 KB in order. Size of process 357 KB, 210 KB, 468 KB, 491 KB in order. Apply first, best, worst and next fit and determine which algo makes most efficient use of memory.

Total M. Wastage				
200				
400	357	357		357
600	210	491	357	210
500	468	468	210	468
300				
250		210		

Best fit

Scanned by CamScanner

It is not mandatory that the size of process will be equal to that of partitions. First we divide the memory space into partitions and then allocate memory to process and then some space in partitions are always left (in fixed partitions) contiguous. So space left in each partition after allocation of process is called internal fragmentation But when dynamically allocated (non contiguous) holes (chunks of memory) are left out. That is called external fragmentation internal fragmentation can be solved by external fragmentation and external fragmentation can be solved by compaction.

INTERNAL FRAG. V/S EXTERNAL.

Internal - When a process is allocated more memory than required, few space is left unused and it occurs when memory is divided into fixed size partitions. It can be cured by allocating memory dynamically or having partitions of different sizes. EXTERNAL

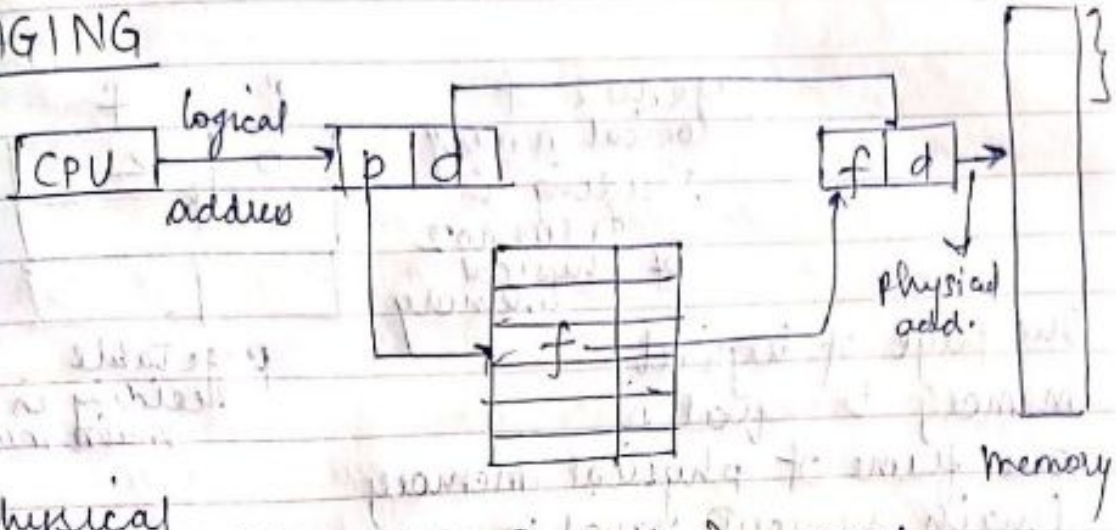
Many small non contiguous blocks of unused spaces are formed which can serve a new request if all of them are joined together. But as they are not adjacent to each other, a new request cannot be served. It can cured by compaction, paging & segmentation.

COMPACTION: External frag can be reduced by compaction or shuffling the memory content and to place all the free memory together into a block.

Scanned by CamScanner

Compaction requires large amount of relocation and hence it is not feasable.

PAGING



physical address = (frame no x frame size) + offset

p: page no.
f: frame no.
d: offset / displacement

Paging is a memory management technique which divides the memory into fixed sizes partitions called frames and divided logical memory (secondary) into same size partitions (fixed) called page.

The size of partition is usually in the power of 2

Reason:-

A page table is a special data structure maintained by operating system to represent page - frame mapping and resides in main memory. [memory]

Each program has its page table.

* The size of frame is kept equal to size of page to have optimum utilization of main memory and to avoid external fragmentation

Page address is called logical address and is represented by page number (p) and page

Scanned by CamScanner

offset(d): Similarly frame address is represented by logical address.

page no 2 of logical memory residing in frame no 2 of physical memory

p	f
0	2

page table residing in main memory

One page of logical memory is equal to one frame of physical memory (main memory) equal to one block of cache memory.

Page table resides in main memory and each process has page table

process is divided into pages in logical memory (secondary memory)

Ques 4 bytes in each page.

Page 0	0-3
Page 1	0-3
Page 2	0-3

Page no | offset

frame no | offset

Logical Address → Page No | Frame No → Physical Address.

Page table

Physical address: frame no | offset remains same.

Page Table → page & frame mapped

Each process has got seperate page table not kept in PCB because extra memory is required as process might be very large

a lot of process in logical space therefore lot of process

Scanned by CamScanner

page table

CPU → logical address (p, d) → (2, 2) content k word.

physical address (f, d) → (1, 2)

physical address = frame no X frame size + offset

= 1 X 4 + 2

⑥ location get k word.

Advantages and Disadvantages of Paging

Advantage:- ① Paging reduces external fragmentation and suffers from internal fragmentation

② Since the page size and frame size are equal, swapping is very easy.

③ Since paging requires special data structure called page table residing in main memory, hence paging is not suitable for small memory sizes.

PTBR → Base register for page table (indicate first index) to search for page table in main memory

and PTLR → Length of page table for each process

for both (PTBR) and (PTLR) are two values stored in PCB of the process.

There are a lot of process in logical space and hence lot of page tables in main memory so find that we have PTBR value in PCB of the process and PTLR as length of PageTable and hence two memory reference of memory

Scanned by CamScanner

no of entries in page table is equal to no of pages in process.

PAGE TABLE

(1) no of entries in page table = no of pages in program/ logical memory.
(2) each process have its individual page table
(3) each page table resides in main memory (additional overhead of memory)
(4) each page table has PTBR value and PTLR value stored in PCB.
(5) page table is a metadata
(6) to access any instruction two memory cycles are required, one for page table and 1 for actual content hence access time is double & speed is half.

K, M, G, T, P.
2^{10}, 2^{20}, 2^{30}, 2^{40}, 2^{50}.

Memory is a byte addressable - each byte has unique location.

1 word accesible 1 word : 4 bytes
then 4 bytes pe ek word store ho raha h

Que We have main memory addressable by 10 bit what will be size of memory if it is byte addressable.

Sol locations = 2^{10}.
byte addressable, $2^{10} \times 1$ byte
1KB.

Que Consider a physical & logical memory which is byte addressable.

Sol. logical address = 31 bit
space = 2^{31}
2GB.
2 GigaBytes.

Scanned by CamScanner

① Logical address space = 128 MBytes
Logical address bit = 2^7 MByte
$= 2^{27}$
27 bit.

② Physical address = 22 bit
Physical address space = $2^{22} = 2^2 \times 2^{20}$
= 4 MBytes.

14 bit address (1 Byte) addressable = $2^{14} \times 1$ Byte
$= 2^4 \times 2^{10}$
= 16 KB.

22 bit address (2 Byte) = $2^{22} \times 2$ Byte $= 2^{23}$
$= 2^3 \times 2^{20}$
8 MB.

34 bit address (4 byte) = $2^{34} \times 4 = 2^{36}$
64 GB.

64 KB Memory size (1 byte) = $2^6 \times 2^{10}$ Byte
(16 bit)

32 KB Memory size (1 byte) = $2^5 \times 2^{10}$ Byte
(15 bit)

256 KB (2 byte) = $\frac{2^8 \times 2^{10}}{2}$ = (17 bit)

16 GB (4 byte) = $\frac{16 \times 2^{30}}{4} = 2^{32}$
(32 bit)

Ques Consider a memory scenario in which logical addresses bit are of 24 bits, physical addresses bit are of 16 bits and the page size is of 1 KB
Calculate bit require [p | d] and [f | d]
Calculate total no of pages and total no of frames in memory.

Scanned by CamScanner

logical addres space = 2^{24}

$= 2^4 \times 2^{20}$

$= 16\ MB.$

physical addres space : 2^{16}

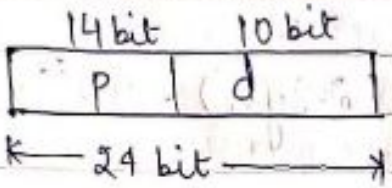
$= 2^6 \times 2^{10}$

$= 64\ KB.$

page size = 1KB.

total No of pages: $\frac{16 \times 2^{10} \times 2^{10}}{2^{10}} = \frac{16MB}{1KB}$.

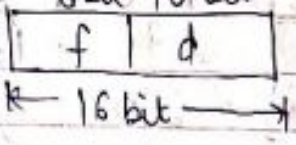
$= 2^4 \cdot 2^{10}$



frame size = 1KB.

total no of frames: $\frac{64 \times KB}{1KB} = 64 = 2^6$.

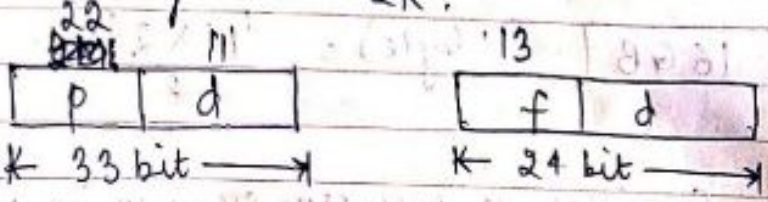
$= 6$ bit.



Qus logical addres = 33 bit

physical addres = 24 bit

Page size = 2K.



logical addres space

$= \frac{2^{33}}{2K} \cdot \frac{8GB}{2K}$

$= 4MB$

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

$= 2^{22} B$

Scanned by CamScanner

Consider a logical memory of 256 pages and page size of 4KB and a physical memory of 64 frames. How many bits are required for logical & physical address.

Ans: physical = 18 bit
logical = 20 bit

Total logical space
$= 256 \times 4$
$= 2^8 \times 2^{12}$ Bytes
$= 2^{20}$ Bytes
$= 1$ MB

physical = 64×4 KB
$= 2^6 \times 2^{12}$
$= 2^{18}$ Bytes

① After paging, we require now 2 memory references to memory for executing a particular instruction and increase time complexity to reference instruction (TIME)

② A lot of space is wasted as each of the process has seperate page table (SPACE).

logical memory is a collection of process. So the unit in which logical memory is divided and its same unit other memories are divided

page table — Demanding Paging

valid invalid bit

TLB, Heirarchical page table, Hashed Table Indexed Table

A special data structure called MMU is maintained to indicate free and available frames in main memory. To protect the memory from illegal address generation one additional valid / invalid is attached to the pagetable. (each entry) before calculating the physical address of corresponding logical address. The valid / invalid bit is checked. If

Scanned by CamScanner

the reference is set to valid physical address will be generated else the reference will be trapped because of wrong address generation

TRANSLATION LOOK ASIDE BUFFER

to reduce time complexity.
If a instruction from a process is being excecuted, then most likely next instruction will also be excecuted So now each time controller goes to page-table then frames table for each offset and hence again and again referencing memory so to reduce the time and memory reference to main memory (as page table is stored in main memory) we introduce a hardware special TLB (to store frequently used ~~address~~ pages) along with frame no.
So now as soon logical address reference generated check TLB if found page then TLB hit

∴ Access time =
(TLB hit * TLB access time)
+ (Memory access time)

and if not found in TLB
then
Access time = (TLB miss * TLB access time) + page table access + memory access time

Initially TLB hit low and TLB hit increases as per process continues.
TLB works as per process.
As soon as context switching happens. Whole of TLB is flushed and as the new process starts

Scanned by CamScanner

Only one TLB present in the system, TLB cannot be used in case of high context switching.

In paging system, two memory cycles are required to access any instruction one for page table and other for memory access.
With this approach the access time of instruction increases drastically to twice which causes major delay and this delay is intolerable under certain circumstances.
To reduce this access time, by reducing the effective memory cycles, a small fast lookup hardware cache called TLB is used. TLB is Associated high speed memory and each entry of TLB consists of two values page no and frame no.
If a page is referenced by CPU under TLB implemented paging scheme

1. If the page frame mapping is found TLB. TLB hit - no need to access page table only one memory cycle required. ∴ TLB access time + memory access time	If the page is not found in TLB - TLB miss. need to access page table to obtain page frame mapping two memory cycles required (TLB access time + 2* memory access time)

Effective Memory access time (whenever case of hit or miss)

→ = hit of TLB*(TLB access time + Memory Access time) + miss of TLB * (TLB access + Memory access time + page table access time)

Ques Consider a paging system with page table stored in memory
If a memory reference takes 50ns, then

Scanned by CamScanner

how long does a paged memory reference takes?

ii) If we add TLB and 75% of all page table reference are found in TLB. What will be effective memory reference time if finding an entry in TLB requires 2 ns.

i) two memory cycle, 2X50, 100ns.

ii) $0.75*(2+50)+0.25\times(2\times10^{-9}+2\times50)$
$= 0.75\times52+0.25\times102$
$=$

Note:- Unlike pagetable there exist only one TLB in the system so whenever context switch occurs, the entire content of TLB is flushed and deleted.
. TLB is then again updated with the currently running process
When a new process gets scheduled, TLB will be empty initially. So TLB misses are frequent.
With every access from the page-table the TLB is updated, and after sometime TLB hits increases & TLB miss decrease

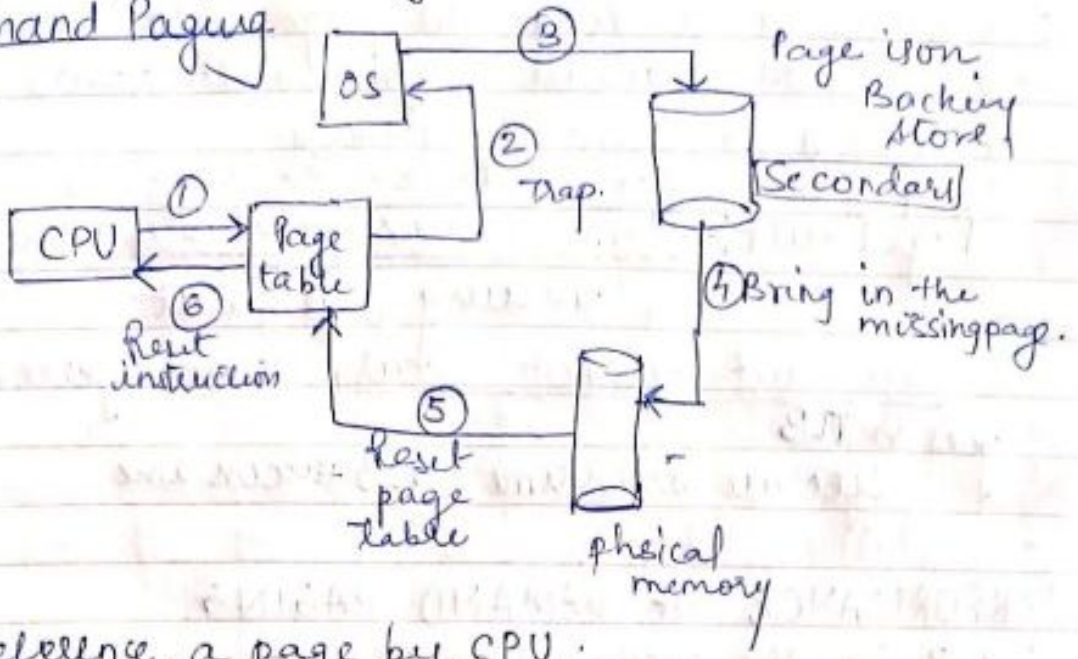
effective access time = 160 ns
memory access = 100 ns
TLB hit = 0.8

$160 = 0.8*(x+100)+0.2(x+200)$
$160 = 0.8x+80+0.2x+40$
$160-120 = x$
$x = 40$ ns

Scanned by CamScanner

Page Replacement Algorithm:

Demand Paging



(1) Reference a page by CPU.
(2) Check the entry in page table along with the valid/invalid bit.

(1a)
- if bit set to valid bit
- hit
- Calculate the physical address
- fetch the instruction
- Execute

(1b)
- if bit is set to invalid bit
- miss
- Check from table for free frame

(2a) if frame available allocate page to free frame
- update frame table
- update page table and set bit to valid
- follow (1a)

(2b)

(2b) if free frame is not available
- select the victim page by applying page replacement algo
- send victim page to backing store
- update page table & set corresponding bit to invalid
- update frame table as one frame free now
- follow 2a

Why do invalid bits occur?

- when page has once come in physical memory and its account is there in page table then if due to some circumstances page was swapped out but entry in page table remains, hence invalid bit occur.

Scanned by CamScanner

pure demand paging - empty memory se fully filled ka duration.

The process of loading the page into memory on demand whenever page fault occurs. is known as demand paging.

Only after checking pagetable we know if page fault not

Page Fault :- time to access pagetable + then removing page fault.

If page fault does not occur, then reference goes to TLB

Effective access time = TLB access time

PERFORMANCE OF DEMAND PAGING.

Let P be the probability of page fault.

$0 \leq P \leq 1$

* Effective access time = [(1 - p) * memory access time] + [p * page fault service time]

* No page fault = 1 - p

Ques Let the page fault service time is 10 ms in a computer with average memory access time being 20 ns. If one page fault is generated every 10^6 memory access what will be the effective access time for the memory?

~~1 ms = 10 seconds~~.

Sol. 1 ms = 10^6 ns.

20 ns.

page fault memory acce

10^6 memory → 1

memory 10^{-6}.

$1 s = 10^{-3} ms \times 10$

$1 s = 20 \times 10^{-1} s$

ms = -11

(10^{-6}) [20 ns + 10 ms]

$+ (1 - 10^{-6}) * 20$

= 30 ms.

Scanned by CamScanner

PAGE REPLACEMENT ALGO

Page replacement is a process of swapping out an existing page from the frame of main memory and replacing it with the required page.
It occurs when all the frames of the main memory are already occupied. A good page replacement algo. minimises the page fault.

1. FIFO
2. LIFO (dont use till all frames are completely filled) ✗ worst
3. LRU
4. MRU
5. LFU (Least Frequently Used)
6. MFU (Most ...)
7. Random
8. Optimal (99%) = MRU.

(1.) FIFO

Oldest page in the memory will be replaced first.
Drawback:- Irrespective of its frequent use no matter.

Que Consider the following reference string and Calculate the no of page fault / hit ratio / miss ratio on a physical memory of three frames.
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

FIFO

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
L0	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
L1		0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1	1	0	0
L2			1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1
					H							H	H			H	H			

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

Scanned by CamScanner

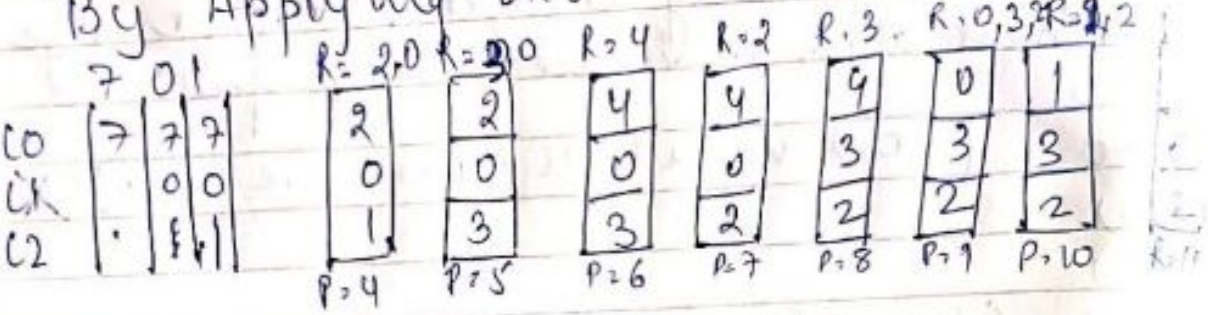
Page fault : 15.

Hit Ratio : 5/20

Miss Ratio - 15/20

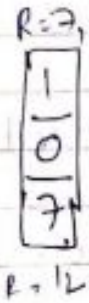
(11)

By Applying LRU.



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

① ② ③ ⑤ ⑥ ④



Page fault = 12

Hit Ratio = 8/20.

Miss Ratio - 12/20.

Note: In FIFO, on increasing number of frames, no of page faults will increase this is called BELADY'S ANAMOLY. (unexpected result occurs)

In rest of cases on increase number of frames no of page fault ↓.

→ It is generally seen, by ↑ the no of frames in main memory no of page fault must be reduced but in certain references of FIFO page replacement algo this condition violates and the no of page faults ↑ with the number of frames in main memory

Scanned by CamScanner

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

3 frames / 4 frams.

Ques Addres sequence genealed by tracing a particular program executing in a pure demand paging system with 100 records per page with one free main memory frame is recorded as follows: What will be the no of page fault.

0 1 00, 0200, 0430, 0499, 0510, 0530, 0560, 0120, 0220, 0240, 0260, 0320, 0370.

page 0 → 100 records (0-99)
page 1 → 100 records (100-199)
(200-299)

1, 2, 4, 4, 5, 5, 5, 1, 2, 2, 3, 3. → Reprence.

Initially one frame

| 1 | 2 |

Page fault : 7

Scanned by CamScanner