18/7/23 __/__/____

Unit - 1

Program under execution is process

multitasking - when more than one task is executed by OS.

OS provide GUI

Above hardware and below application layer, OS lies

→ How system software is diff from app. s/w

when all the programs are executed in ~~wrt~~ time it is RTW real time OS

20/7/23

Memory Hierarchy. Register

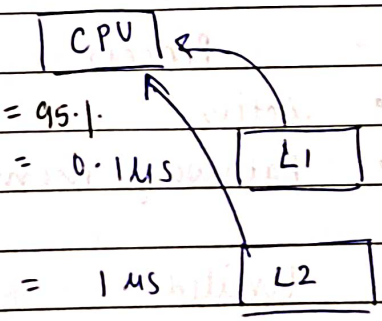
main memory

Hard disk

- The speed of processor is always high
- Cache memory is also responsible for high
- Cache and RAM both are volatile
- Access time of cache mem. is very low.
- RAM se phele cache me jaega or parallel CPU ~~cache~~ me jaega
- Hard disk → RAM → Cache → CPU → Register.

Function of flip-flop is to store 1 bit

Register is a flip flop



Hit ratio = 95%.

A.T = 0.1 ns

A.T = 1 ns

Hit ratio

1 - H = 0.5%.

Access time = A.T x Hit ratio

= 0.1 x 0.95

= 0.095

Total A.T = 0.95 x 0.1 + 0.5 (0.1 + 1)

= 0.095 + 0.5 x 1.1

= 0.095 + 0.55 = 0.545

__/__/____

Linux command-

1. top - display all running process.
2. PS - to display all current working process.
3. Kill pid - kill the process with given process id (Pid)
4. Kill all proc - kill all the process named proc.
5. date - show current date and time
6. cal - show month's calender
7. w - display who is online
8. finger user - display info. about user
9. uname - a → show Kernel information
10. cat / proc / cpuinfo → show CPU info.
11. df → show the disk usage
12. du → show directory space usage
13. free → show memory and swap usage

26/7/23

What is process?

It is a program in execution.

An instance of program is process.

→ class is a user-defined

→ An instance of program in a computer.

→ The entity that can be assigned to and execute on a processor (CPU)

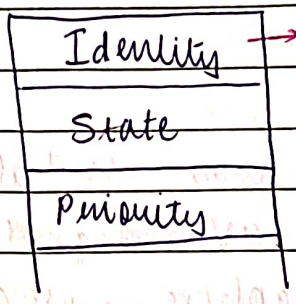
Diff b/w program and process

Program	Process
• It is an passive entity	• Active
• It is stored in sec. memory	• Pathway memory
• Time span is limited	• limited
• It is a set of sequence of instructions execution	• Set of instruction and data
• It is static object (generally stored in file form)	• It is a dynamic object (program in execution)

__/__/____

when a process runs in main memory. OS create a data structure called PCB (Program controlblock)

PCB.



→ each process in main memory

1/8/23.

Thread

1. A thread is a single sequential flow of execution of task of a process
2. There is a way of thread execution inside the process of any operating system
3. There can be more than one thread inside a proces
4. Each thread of the same process makes use of a seperate program center, stack of activation and control block.

A process can be split down into a many threads

Eg- In a browser, many tabs can be view as threads.

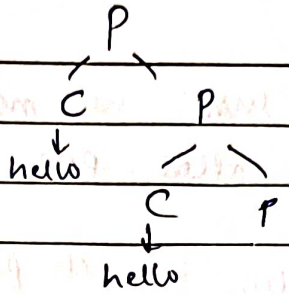
Need of thread -

1. It takes far less time to create a new thread in an existing process than create a process
2. Threads can share the common data, they do not need to use inter-process communication
3. Context switching is faster when working with threads.
4. It takes less time to terminate a thread than a process.

11/8/23 __/__/____

Fork()

$2^n - 1$



Q. A counting semaphore was initialized to 10 s = 9 +1/
then 6P (wait) and 4V (signal) operation were computed on this semaphore. what is the result?

Process Hierarchy -

Modern general process O/S permits a user to create and destroy process. A process may create several new processes during its time of execution. The creating process is called parent process. while the processes are child processes.

There are diff possibilities concerning creating new processes -

(1) Execution - The parent process continues to execute concurrently with its children process or it wait until all of its children have terminated

(2) Sharing - Either the parent & children process share all resources or the children processes share only a subset of their parent's resources or the parent and children process share no resources in common.

↳ Process termination points.

	Process	Thread
1.	System call involved	1. No system call involved
2.	Independent	2. Interdependent
3.	CS is slow	3. Context switching is fast
4.	Blocking a process will not block entire process	4. will block entire thread
5.	Program in execution	5. segment of process
6.	takes more time in creation	6. Less time in creation
7.	less efficient in term of communication	7. More efficient.

User Level thread	Integral part of OS. Kernel Level thread
• implemented by users	• Implemented by OS
• not recognized by OS	• recognized by OS
• context switch time is less	• C-s time is more
• if thread perform blocking then entire process gets blocked	• if one kernel level thread is blocked, no affect on another.
• typically fast	• KL thread are slower than UL thread

POSIX thread- execution model that jo independently run hota hai, jab same time pe kehet kaam hote hai usko overlap na hone de control karega
It is a IEEE pthreads, it is an execution model that exists independently from a lang. as well as parallel execution model. It allows a program to control multiple diff. flows of work that overlap in time.

__/__/____

Wait()

Wait op' is used to test if there is any other process using the resource or any other process is there in the critical section and if it is not there then it will decrement the s and enter the critical section or if some resource process is already using the resource this while cond^n will be true and requesting process will be stuck here.

Wait is actually used for testing

Signal()-

It just increments the value of A (semaphore) It will be called when the process that was making use of semaphore to enter into the critical section or was we using a resource completes its op^n and comes out of it. Hence denoting that it has now released the semaphore.

This used to signal other processes that it has completed using semaphore.

When one process modifies the semaphore value, no other process can simultaneously modify that same semaphore value.

Semph - integer

Types- mutex - is an object.

(1) Binary Semaphore- The value of binary semaphore can range only b/w 0 and 1. On some system b.s are k/a mutex locks, as they are locks that provide mutual exclusion.

__/__/____

Counting - Its value can range over an unrestricted domain it is used to control access to a resource that has multiple instances.

Monitors-

Fork() - to create child process.

child process is clone of parent process which has an id of its own

Fork 3 value return karegi

0 - child process

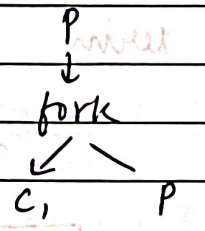
+1 - khud parent

-1 - agar child proces kisi vajah se nahi bana

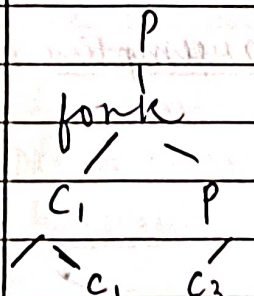
Eg. main()

fork(); fork();

printf("hello"); }



2 bar hello print hoga.



4 baar hello print hoga.

$[2^n - 1]$ total no. of child process.

Philosopher:

```
void philosopher (void)
{
  while (true)
{
  Thinking();
take_fork(i)   ~~left fork~~
"      (i+1)%N),  ~~Right~~ fork
```

```
EAT()
put fork(i);
put fork((i+1)%N);
}
}
```

((i+1)%N);

MT- II

unit - 3.

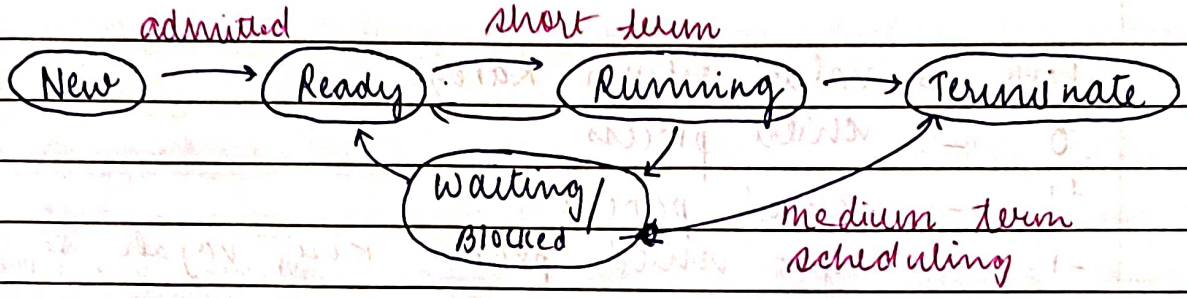
12/9/23

__/__/____

Reader / writer problem
do on your own

Process scheduling -
which process should be given priority / CPU.

1. Long term — state changes from new → ready
2. short term — ready → running



3. Medium term

On basis of process

Types of scheduling

Preemptive — when a process is running and some

Non-preemptive

On basis of type of OS

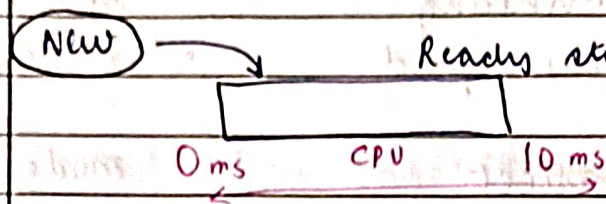
	Interactive OS	Batch OS	Real-Time OS
	LCD displays, devices, variable mobile devices	when particular task is divided into batches or diff devices / system	immediate response where deadline is not allowed
	Preemtive		
	→ Round Robin → Priority → Lottery scheduling → Fair share	→ FCFS → Shortest Job First → Shortest remaining time next	→ Earliest deadline first

→ Shortest fair next

__/__/__

Terms related with scheduling

1. Arrival Time when it is in new state and CPU is assigned to it. Time taken by process when it enters in a ready state



2. Burst Time - Time taken by process for the execution on the CPU

3. Turn around time - Total time spend by the process from coming in the ready state to its completion.

$$TAT = BT + \text{Waiting time}$$

$$TAT = \text{Completion Time} - \text{Arrival Time}$$

4. Response Time - It is the time spent by the process in a ready state and it gets the CPU for the execution.

if a process is not preemptive then its RT = Waiting time *

* For preemptive :

$$RT = \text{process gets the CPU for first time} - \text{Arrival Time}$$

5. Waiting time -

$$\text{Waiting Time} = TAT - BT$$

Time spend by the process in ready state

Algo's

1.	Shortest Remaining Time First	Preemptive
2	Round Robin	
3	Priority	
4	FCFS	Non - preemptive

__/__/____

Q.** Diff b/w all 3 schedulers long-term, short term, medium term

Ans.

	long term	short term	Medium term
State 1.	state new → ready	• ready → running	• wait → termination
speed •	slow speed	• fast speed	• moderate
context switching •	slow	fastest	• moderate
•	Job schedulers	• CPU scheduler	• Process swapping
•	Controls degree of multiprogramming	• less control over mp	• Reduces degree of mp.

15/9/23

Algorithms -

1. Shortest Remaining Time Next Algorithm -

Criteria = "Burst Time"

Mode = "Preemptive"

P2 complete hone ke bad ready state se remove ho jaega.

PID	AT	BT
P1	0	5 4
P2	1	3 2 1 0
P3	2	4
P4	3 4	10

Ready - P1 | P2 | P3 | P4

0 1

Running -

3 2 1

P1	P2	P2	P2	P4	P1	P3

0 1 2 3 4 5 9 13

jiska burst time kum hoga wo ready state me aega.

agar 2 ke burst time same hai to fir uska arrival time dekhnge

Parameters to be checked : CT, WT, TAT, RT

TAT − BT → CT − AT

__/__/____

PID	AT	BT	CT	WT	TAT	RT	
P1	0	5	9	4	9	0	— Jab kitni bar CPU assign hua minus AT.
P2	1	~~3~~	4	0	3	0	
P3	2	4	13	7	11	7	
P4	4	1	5	0	1	0	
			~~6.12~~ 7.75	2.75	6	1.75	

Round Robin Algorithm–

Criteria = "Time Quantum"

Mode = "Preemptive" TQ = 5 ms

PID	AT	BT	CT	WT	TAT	RT	
P1	0	~~7~~ 24	31	24	31	0	CT = 44.5
P2	1	~~4~~ 0	9	4	8	4	WT = 32.8
P3	2	~~15~~ 10	5 55	38	53	7	TAT = 43.8
P4	3	~~11~~ 6	1 56	42	53	11	RT = 9.5
P5	4	~~20~~ 15	10 66	42	62	15	
P6	4	~~9~~ 40	50	47	56	20	

Ready | P1 | P2 | P3 | P4 | P5 | P6 | — jiska bhi BT poora hota jaa usko yaha se remove kardenge.

0

Running | P1 | P2 | P3 | P4 | P5 | P6 | P1 | P3 | P4 | 46 | P6 | P3 | P4 | P5 | P5 |

0 5 9 14 19 24 29 31 36 41 46 50 55 55 56 61 66

20/9/23 __/__/____

3. Priority scheduling -

mode - Criteria - Preemptive, non-preemptive
criteria - priority

non-preemptive

PID	AT	BT	Priority	CT	WT	TAT	RT
P_1	0	4 0	2	4	0	4	0
P_2	1	3 0	3	15	11	14	11
P_3	2	1 0	4	12	9	10	9
P_4	3	5 0	5	9	1	6	1
P_5	4	2 0	5	11	5	7	5

Ready: $|P_1|P_2|P_3|P_4|P_5| \rightarrow |\ |P_2|P_3|P_4|P_5|$ (process completed is removed from ready queue)

If priority is equal of the process we consider AT.

Running - $|P_1|P_4|P_5|P_3|P_2|$

0 4 9 11 12 15

8.6 4.6 7.6 0.8

Preemptive

PID	AT	BT	Priority	CT	WT	TAT	RT
P_1	0	4 3	2	15	11	15	0
P_2	1	3 2	3	12	8	11	0
P_3	2	1 0	4	3	0	1	0
P_4	3	5 4 0	5	8	0	5	0
P_5	4	2 0	5	10	4	6	4

Ready $|P_1|P_2|P_3|P_4|P_5| \rightarrow |P_1|P_2|\ \ |P_4|P_5|$

Running $|P_1|P_2|P_3|P_4|P_4|P_5|P_2|P_1|$

0 1 2 3 4 8 10 12 15

__/__/____

4. Shortest Job first (SJF)

criteria - Burst time
mode - Non-preemptive (Execution of the process completely at one go)

PID	AT	BT	CT 16.6	TAT 11.4	WT 5.4	RT 5.4
1	1	7	8	7	0	0
2	3	3	13	10	7	7
3	6	2	10	4	2	2
4	7	10	31	24	14	14
5	9	8	21	12	4	4

compare BT & apply non-preemptive

Ready

	P_1	P_2	P_3	P_4	P_5
0	1				

↳ freespace as no process

Running

	P_1	P_3	P_2	P_5	P_4
0	1 & 8	10	13	21	31

in b/w this P_2, P_3, P_4 arrives.

in b/w this P_5 arrives.

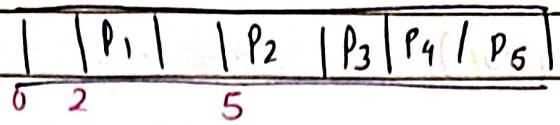
5. FCFS (First come first serve)

criteria - arrival time
mode - non-preemptive
* if arrival time is same then we check PID.

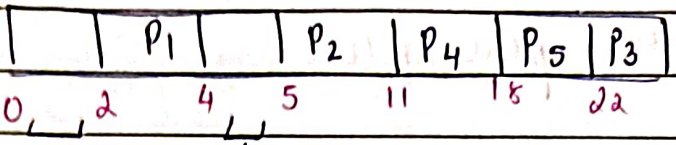
PID	AT	BT	CT	TAT 16.2	WT 10.6	RT 6	6
1	2	2	4	2	0	0	
2	5	6	11	6	0	0	
3	8	4	26	18	14	14	
4	6	7	18	12	5	5	
5	7	4	22	15	11	11	

__/__/____

Ready



Running



free space

(Here we consider time when there is no process.

25/9/23

Deadlock -

Necessary Conditions :-

1. Non-preemptive
2. Hold and wait
3. Circular wait
4. Mutual Exclusion

Definition of deadlock - A deadlock is a situation in which all the processe gets block because it is the holding resource and also requires some requires

1. Non-preemptive - The resources can't be released by a particular process before its complete execution.

2. Hold and wait - A process is holding atleast one resource at a time & is waiting to acquire other resource hold by some other process.

3. Circular Wait - A set of processes are waiting for each other in circular mode.

4. Mutual Exclusion - Only one process can use the resource

__/__/____

at a given time i.e resources are non-shareable.

Deadlock Handling Strategies :-

1. Deadlock Prevention
2. Deadlock Avoidance (Banker's algo.) → safe state
   → unsafe state.
3. Deadlock Detection and Recovery
4. Deadlock Ignorance. → Also Ostrich's algo.

1. Deadlock Prevention - Since the deadlock occurs when all the 4 necessary conditions are fulfilled so we try to prevent any one of them Thus preventing the deadlock.

2. Deadlock Avoidance - When a process request for a resource that is Banker's algo. check for resource-state allocation. If allocating the resources sends the system into unsafe, so the request is not generated.

3. Deadlock detection & Recovery - We let the system fall into deadlock & when it happens then following steps are taken to recover.

   i) Abort all the process due to which deadlock occurs.

   ii) Abort the process due to which deadlock has occured one by one & check for recovery.

   iii) Resource preemption - resources are taken one by one from a process and are assigned to a higher priority process.

4. Deadlock Ignorance - In this we follow ostrich algo. which states that if the deadlock occurs simply reboot the

__/__/____

system & act like deadlock has never occured.

Banker's algo.
- ↳ Allocation
- ↳ Max. need
- ↳ Available
- ↳ Remaining Need.

$8_A\ 4_B\ 6_C$

Q.

agar ye given na ho to available line ! de

Process	Allocation			Max. Need			Available			Remaining Need		
	A	B	C	A	B	C	A	B	C	A	B	C
P_0	1	0	1	4	3	1	3	3	0	3	3	0
P_1	1	1	2	2	1	4	+ 1 / 4	3	1	1	0	2
P_2	1	0	3	1	3	3	4	3	1	0	3	0
P_3	2	0	0	5	4	1	6	4	6	3	4	1
Sums	5	1	6									

given

[18 4 6] — if not given
↳ Last available + last allocation

8-5 4-1 6-6

Total resources = Total allocation + available

so, here total resources =

for A : 5+3 = 8
B : 1+3 = 4
C : 6+0 = 6

Need <= Available

Remaining Need = Max. Need - Allocation.

for A : 4-1 = 3
B : 3-0 = 3
C : 1-1 = 0

P_0 =

$P_0 - P_2 - P_1 - P_3$
allocations.

~~$P_0 - P_1 - P_2 - P_3$~~

26/9/23

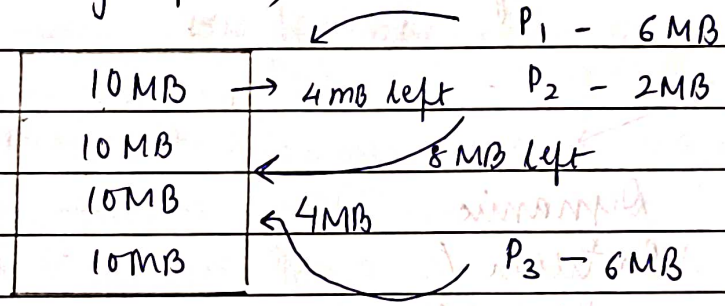
__/__/____

Unit 4 Memory Management

Subdividing the memory for different processes.

Why memory management is required -

1. To allocate & deallocate memory
2. For efficient and proper utilization of memory space
3. To maintain the integrity of process residing in the memory.
4. To resolve the fragmentation issue (free or unused memory space)



To overcome this problem we use memory mngmt

Job device change krne pr add. bhi change hoga.

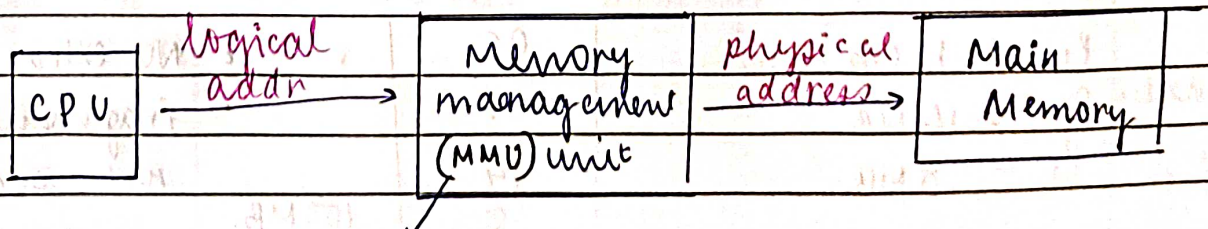
Address space

- Logical Address space
- Virtual Address
- Generated by CPU
- Changes from device to device

Physical address space

- Real address
- Generated by residing
- Remains constant.

1 word = 16 bits = 2 bytes.
Double word = 2 word.



mapping of logical address to physical address.

collection of addresses.

__/__/__

Physical Address space	Logical Address space
• generated by residing space	• generated by CPU
• Not visible to user	• visible to user.
PA - $\log_2 N$ bits	PA - $\log_2 L$ bits.

Memory management Techniques -

Contiguous Memory allocation

- Fixed / static Partition
- Dynamic Partition / Variable

Non-contagious allocation

- ↳ Paging
- ↳ Inverted paging
- ↳ segmentation
- ↳ segmentation with paging
- ↳ Multilevel paging.

Fixed

first part will be given to OS

P1 — 12 MB

P2 — 14 MB

	Memory	Size
	OS	20MB
12, 8	P1 (12)	20
14, 6	P2 (14)	20
		20
		20

equal partition → fixed amount of space is distributed.

disadvantage - It creates fragments

Dono me space continous allocate ho raha hai

Variable / Dynamic

fixed slot nahi hunge, jitna required hai utna hi milega

P1 - 12MB

P2 - 14MB

P3 - 8MB

OS
12
14
8

100MB.

- No extra space
- Fragmentation is not occured.

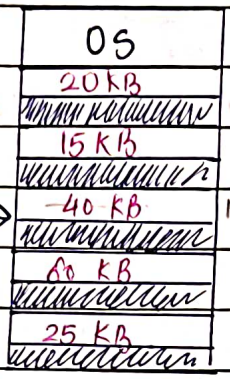
* increases the degree of multiprogramming.

__/__/____

Fixed partition method -

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

(1) First fit



P_1
25 KB
40 - 25
= 15

pehli jo space mili usko utilize kar liya

disadvantage - Jab isne memory space use kari, isko need 25 ki thi but 40 vali use kari to 15 KB fir bhi bach gaya.

advantage - . it doesn't need to traverse the complete block
. less time is required.

(2) Best fit — suppose $\boxed{P_1}$ (25 KB) will traverse whole same diagram. block and find best segment which will create small or no fragment and here will utilize 25 Kb (complete utilize). If suppose 30 KB was present rather than 25, then out of 40, 60, 3 it will choose the one with less wastage.

less storage waste

(3) Worst fit - suppose $\boxed{P_1}$ (25Kb) will traverse and will go to 60Kb because wastage of memory.
[max. no. of fragments created]

2/10/23

sabko diagram pora bana ho hai

__/__/____

Q. 600 k (600 − 212 = 388 k) 300 k 200 k 500 k 100 k

Process → P_1 212 k, P_2 417 k, P_3 112 k, P_4 426 k

① First fit 2) Best fit 3) worst fit

First fit

600 k → 600 − 212 = 388 k

388 − 112 = 276

300 k

200 k

500 k — 500 − 417 = 83 k

100 k

P4 → will go in the waiting state.

Best fit

600 k — 600 − 426 = 174

300 k — 300 − 212 = 88

200 k — 200 − 112 = 88

500 k — 500 − 417 = 83

100 k

174 − 112 = 62

174
426
88
212
88
112
83
417
100 k

Worst fit

600 k — 600 − 212 = 388

300 k — 300 − 112 = 188

200 k

500 k — 500 − 417 = 83

100 k

388 − 112 = 276

388
212
188
112
200 k
83
417 k
100 k

Non-contiguous Memory Allocation. __/__/____

Fragmentation

Internal ← → External (requires compaction)

1. A fragmentation is defined as the process of loading & removing the process from the memory block, it creates a small hole or fragments
2. These holes cannot be assigned to a new process because they do not fulfil the requirement of the upcoming process
Kum fragment

3. Fragmentation decreases the degree of multiprogramming which needs to be dissolved.

Internal fragmentation - occurs when memory blocks are allocated to the process more than their requested size due to this some unused memory space is left

Eg: In 600K memory 200 is assigned due to which large fragmentation is occured.

External fragmentation - In this we have a free memory block but it cannot be assigned to a process because it is less than requested or non-continuous

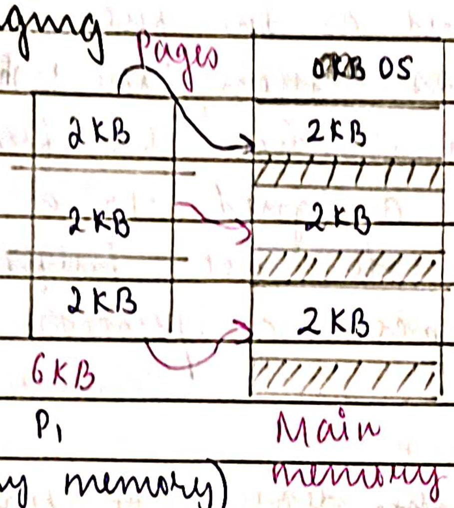
Eg: If 212 is required but we have only 200K available agar memory available bhi hai to bhi vo continuous nahi hai

Compaction - In which all free memory space are combined to make one large block so this large memory block should be assigned to the

3/10/23 upcoming process.

Non- contiguous Memory allocation

1. Paging



memory management scheme that eliminates the need of contiguous memory.

Example :

Process size = 4B Frame size

Process size = 4B
Page size = 2B

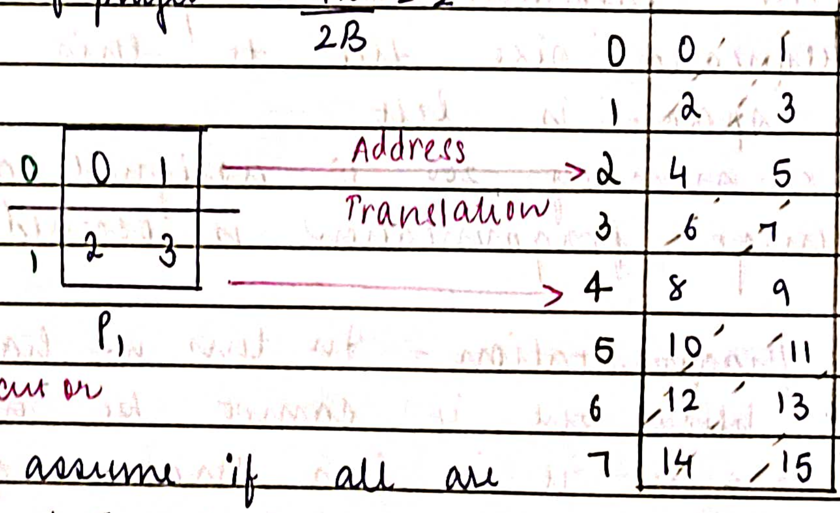
Main memory size = 16B
Frame size = 2B

No. of pages = $\frac{4B}{2B} = 2$

Total no. of frame = 8

$\frac{16}{2} = 8$.

case ①



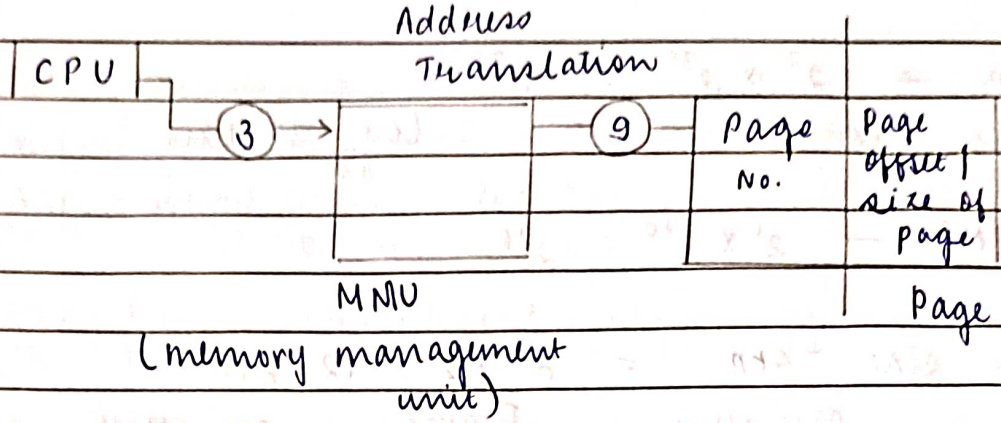
isno diagram base ne hai kiya cut or cut.

case ②

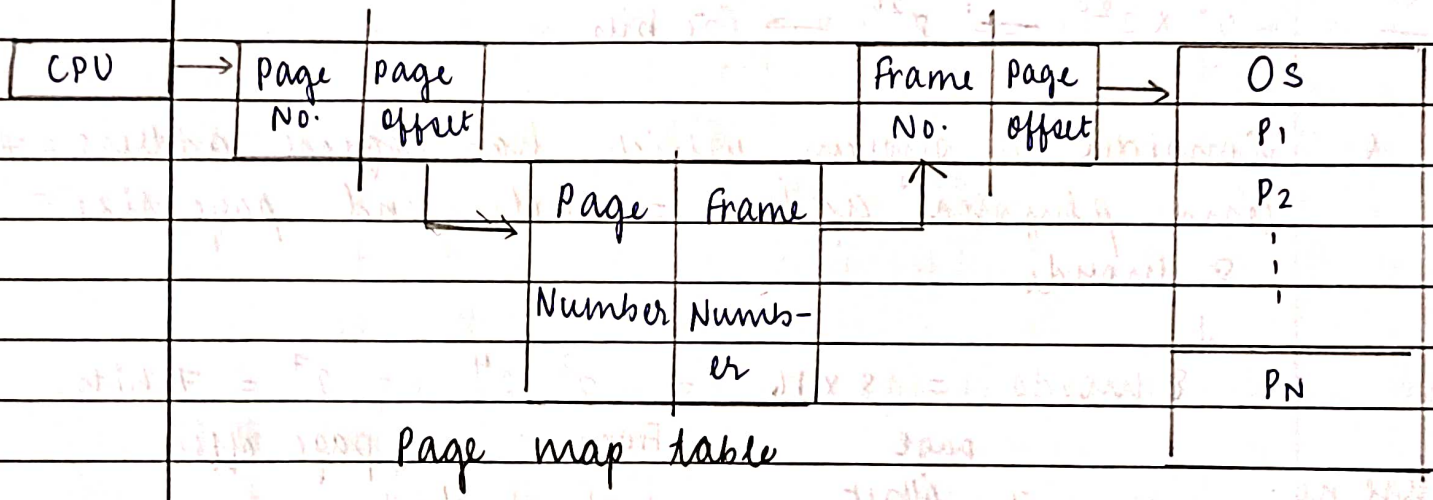
Now, let us assume if all are used except 4 5 and 8 9 then 0 1 will be assigned to 4 5 and 2 3 → 8 9.

memory is byte addressable / addressable

__/__/____



Page map table is a ds that keeps track of a rel^n b/w page and a frame in physical memory



logical address = Page No. + Page offset

Physical address = Frame no. + page offset.
frame

4/10/23

Q. LAS (space) - 4 GB PAS - 64 GMB

Page size = 4KB

Find no. of pages and no. of frames.

2^{20} = 1MB
2^{30} = 1GB
2^{40} = 1 TB
2^{10} = 1KB
2^{9} = 512
2^{8} = 256
2^{7} = 128

Sol^n:

logical address / Page address = Page Number + Page offset

Physical / Frame address = Frame no. + page offset

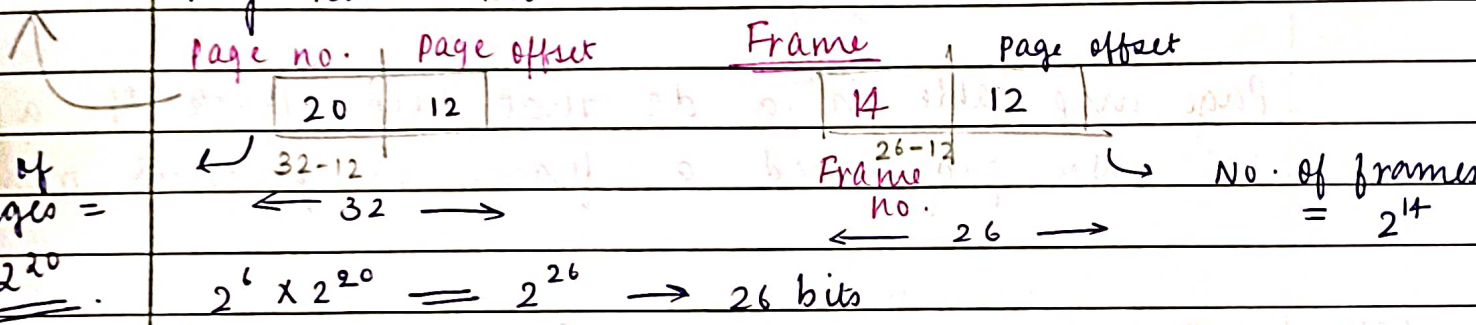
Page size = Frame size

4 GB → $2^2 \times 2^{30} = 2^{32}$

these convert to bits → $\log_2^{32} = 32$ bits

64 MB → $2^6 \times 2^{20} = 2^{26}$ = 26 bits.

Page size = 4KB = $2^2 \times 2^{10}$ = 12 bits

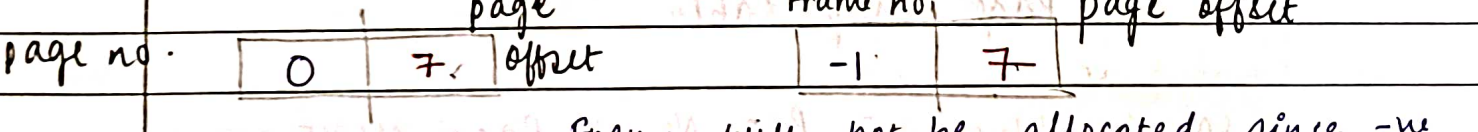


$2^6 \times 2^{20} = 2^{26}$ → 26 bits

Q. Consider a system which has logical address = 7 bits and physical address = 6 bits and page size = 8 words (128 bits).

↓

8 words = $8 \times 16 = 2^3 \cdot 2^4 = 2^7$ = 7 bits.



Frame will not be allocated since -ve value.

Demand Paging

↙ Frame allocation

↘ Page replacement algo
1. FIFO
2. Optimal
3. LRU
4. MRU

The process of calling the pages to main memory from secondary memory upon demand is k/s demand paging.

__/__/____

Page replacement algo. are used to decide which page needs to be replaced in a frame when a new page comes for execution.

1. FIFO Page replacement Algorithm

Reference page : 6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 2, 0

frame size = 3

page hit – If found in frame.

page miss / fault – If not found in frame

	i	2	3	4	5	6	7	8	9	10	11	12	13	14	15
frame size f3				2	2	2	4	4	4	2	2	2	2	2	2
frame size f2		1	1	1	1	3	3	3	0	0	0	0	0	0	0
frame size f1	6	6	6	6	0	0	0	6	6	6	1	1	1	1	1
	F	F	H	F	F	F	F	F	F	F	F	H	H	H	H

	16	17	18	19	20
	2	2	2	2	0
	3	3	3	3	3
	1	1	1	1	1
	F	H	H	H	F.

pehle 6 gaya since usko jaga koi nahi tha isly fault, next 1, fir 2, 0, 3 but agar 0 bhi nahi hai to pehle 6 hat jaega, jo pehle aya wo pehle remove hoga

$$\text{Hit \%} = \frac{\text{Total target Hit}}{\text{Total Reference page}} \times 100$$

No. of hits = 8

No. of faults = 12

Hit : Fault

8 : 12

2 : 3

$$\%\ \text{Hit} = \frac{8}{20} \times 100 = 40\%$$

$$\%\ \text{Fault} = 60\%$$

frame size = 4

6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 2, 0

f4					0	0	0	0	0	2	2	2	2	2
f3				2	2	2	2	6	6	6	6	6	6	6
f2		1	1	1	1	1	4	4	4	4	4	4	4	4
f1	6	6	6	6	6	3	3	3	3	3	1	1	1	1
	F	F	H	F	F	F	F	F	H	F	F	H	H	H

Percentage nikal lena

2	2	2	2	2	2	
6	3	3	3	3	3	
0	0	0	0	0	0	
1	1	1	1	1	1	
F	F	H	H	H	H	

9/10/23.

Belady's Anomaly -

Eg: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

Frame size = 3 Page faults = 9

" = 4 " = 10

→ Agar frame size badega to page faults bhi badega. (only for some). (This is only applicable for FIFO).

It is a phenomenon where increasing the no. of page frames, increases (results) in increase of page faults

2. Optimal Page Replacement Algo (Farthest future set)

6, 1, 1, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 2, 0

frame size = 3

F3				2	2	3	4	4	4	4	1	1	1	1	1
F2		1	1	1	0	0	0	0	0	0	0	0	0	0	0
F1	6	6	6	6	6	6	6	6	6	2	2	2	2	2	2
	F	F	H	F	F	F	F	H	H	F	F	H	H	H	H

agar teeno me se jo sabse door hai use karega.

__/__/____

1	1	1	1	1	11 fault
3	3	3	3	0	9 hit.
2	2	2	04	4	
F	H	H	F	F	

Hit : fault %. $\frac{9}{20} \times 100 = 45\%$ hit

9 : 11 %. $\frac{11}{20} \times 100 = 55\%$ fault.

Jo left Hand me sabse door hai voi replace hoga.

3. Least Recently used. (Farthest ref from past)

6, X, X, 2, 0, 3, 4, 6, 0, 2, 1, 2, 1, 2, 0, 3, 2, 1, 2, 0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
F_3				2	2	2	4	4	4	2	2	2	2	2	2
F_2		1	1	1	1	3	3	3	0	0	0	0	0	0	0
F_1	6	6	6	6	0	0	0	6	6	6	1	1	1	1	1
	F	F	H	F	F	F	F	F	F	F	F	H	H	H	H

	16	17	18	19	20
F_3	2	2	2	2	2
F_2	0	0	1	1	1
F_1	3	3	3	3	0
	F	H	F	H	F

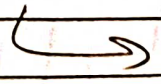
F = 12

H = 8

4. Most recently used (MRU). Just left hand vala replace hoga.

6, X, X, X, 0, 3, 4, 6, 0, 2, 1, X, X, 2, 0, 3, 2, 1, 2, 0

F_3				2	0	3	4	4	4	4	4	4	4	4	
F_2		1	1	1	1	1	1	1	1	1	1	1	1	1	
F_1	6	6	6	6	6	6	6	6	0	2	2	2	2	2	
	F	F	H	F	F	F	F	H	F	F	H	H	H	H	



__/__/____

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

F_3			1	2	2	2	2	2	2	3	0	3	2	1
F_2		0	0	0	0	3	0	4	4	4	4	4	4	4
F_1	7	7	7	7	7	7	7	7	7	7	7	7	7	7
	F	F	F	F	H	F	F	F	H	F	F	F	F	F

2	0	1	1	1	1	Hit = 4
4	4	4	4	4	4	fault = 16
7	7	7	7	0	7	
F	F	F	H	F	H	

1). FIFO.

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

F_3			1	1	1	1	0	0	0	3	3	3	3	3	2
F_2		0	0	0	0	3	3	3	2	2	2	2	2	1	1
F_1	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0
	F	F	F	F	H	F	F	F	F	F	F	H	H	F	F

2	2	2	2	1	hit = 5
1	1	1	0	0	fault = 15.
0	0	7	7	7	
H	H	F	F	F	

2. optimal.

F_3			1	1	1	3	3	3	3	3	3	3	3	1	1
F_2		0	0	0	0	0	0	4	4	4	0	0	0	0	0
F_1	7	7	7	2	2	2	2	2	2	2	2	2	2	2	2
	F	F	F	F	H	F	H	F	H	H	F	H	H	F	H

1	1	1	1	1	fault = 9
0	0	0	0	0	hit = 11
2	2	7	7	7	

__/__/____

Least recently

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

F_3			1	1	1	3	3	3	2	2	2	2	2	2	2
F_2		0	0	0	0	0	0	0	0	3	3	3	3	3	3
F_1	7	7	7	2	2	2	2	4	4	4	0	0	0	1	1
	F	F	F	F	H	F	H	F	F	F	F	H	H	F	H

2	2	7	7	7
0	0	0	0	0
1	1	1	1	1
F	H	F	H	H

fault = 12
hit = 8

10/10/23.

Segmentation (Non-contagious)

(Related code)

```
SUB 1
{ —
}
SUB 2
{ =
}
MAIN
{ =
}
```

500	SUB 1	400	SUB 2	600	MAIN
→	{ =		{ =		{ =
2000	}		}		}
		3500		5000	
	segment 1		segment 2		segment 3

segment map table

Process (secondary memory)

Execution time will be faster.

	OS
0	
1000	
2000	SUB 1
2500	
3500	SUB 2
3900	
5000	MAIN
5600	

Primary memory.

	Base address	limit
0	2000	500
1	3500	400
← 2	5000	600

variable size

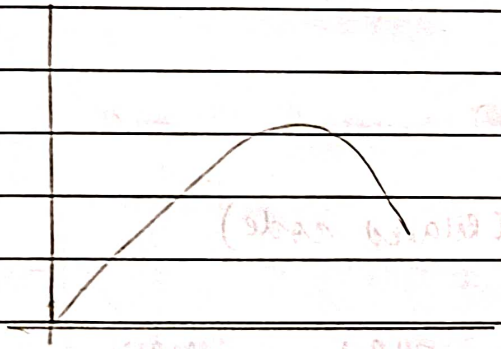
__/__/____

Q. Why segmentation is preferred over paging?

In the case of paging technique a fun or a piece of code is divided into pages without considering that the relative part of a particular code gets divided, so to overcome this in segmentation techniques code is divided into modules so that related code can be combine into one single block.

Thrashing -

- swapping is more page faults more



when a page fault & swapping occur frequently then the OS spends more time in swapping these pages rather than executing the process this decreases the CPU utilisation and degree of utilization

__/__/____

Assignment - 1.

Q. 3, 8, 2, 3, 9, 1, 6, 3, 8, 9, 3, 6, 2, 1, 3

FIFO.

F5						1	1	1	1	1	1	1	1	1	1
F4					9	9	9	9	9	9	9	9	2	2	2
F3			2	2	2	2	2	2	8	8	8	8	8	8	8
F2		8	8	8	8	8	8	3	3	3	3	3	3	3	3
F1	3	3	3	3	3	3	6	6	6	6	6	6	6	6	6
	F	F	F	H	F	F	F	F	F	H	H	H	F	H	H

hit = 6 fault = 9.

LRU

F5						1	1	1	1	1	1	1	2	2	2
F4					9	9	9	9	9	9	9	9	9	9	9
F3			2	2	2	2	2	2	8	8	8	8	8	1	1
F2		8	8	8	8	8	6	6	6	6	6	6	6	6	6
F1	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	F	F	F	H	F	F	F	H	F	H	H	H	F	F	H

fault = 9 hit = 6

11/10/23

__/__/____

Unit - 5

Input - Output Devices

Block Devices

- complete block is read at once

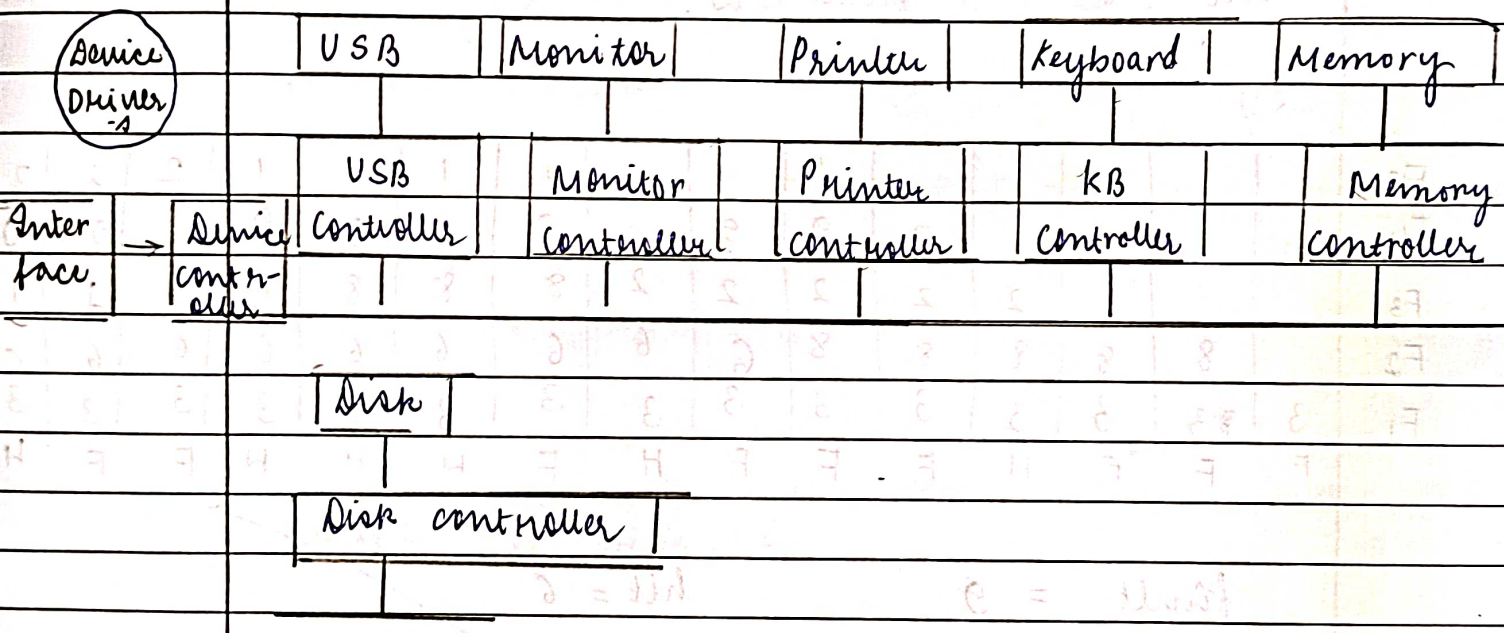
Eg. USB Cameras

Character Devices

→ Read byte by byte

Eg. Serial Ports, Parallel Ports.

Device Controller



Communication to I/O Devices

1. Spicial Instr^{n} I/O
2. Memory mapped I/O
3. DMA (Direct Memory Access)

Memory mapped I/O -

→ In memory - mapped I/O the same address page is shared by memory and I/O devices

→ The device is directly connected to memory location so that I/O devices can transfer data

__/__/____

to | from memory without going through the CPU