1. PROCESS MANAGEMENT :-

1. Introduction to OS :- OS is an interface b/w user and Hardware.

Types of operating system :-

(a) Batch OS :-
- Starvation
- less throughput
- No pre-emption.

(b) Multiprogramming :-
- No starvation
- More throughput
- No pre-emption.

(c) MultiTasking :-
- preemption is present.

(d) Multiprocessing :-
- Throughput is more.

2. various Times related to process :-

$$TAT = CT - AT \qquad WT = TAT - BT$$

3. CPU Scheduling :-

(a) Introduction to FCFS :-

criteria :- Arrival Time

mode :- Non-preemptive

$$\text{Response time} = \text{waiting Time}$$

$$\text{Efficiency } \eta = \left(1 - \frac{\text{Process No}}{\text{completion time}}\right) \times 100$$

(b) Introduction to SJF :-

criteria :- Burst time

mode :- Non-preemptive

$$\text{Throughput} = \frac{\text{No. of process}}{\text{Total Schedular time}}$$

- Simple Averaging :-

$$\tau_{n+1} = \frac{1}{n} \sum_{i=1}^{n} t_i$$

- Exponential Average / Aging :-

$$\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n \qquad 0 \le \alpha \le 1$$

(c) Introduction to SRTF :-

criteria :- Burst time

mode :- pre-emtive.

(d) Round Robin Algorithm :-
- Not depending on BT.
- Each process is executed for TQ amount of time. (TQ → Time Quantum).
- less starvation problem.
- criteria :- TQ + AT (First come first serve)
- Round Robin :- pre-emptive.
- As the time quantum [↑] then no. of context switches [↓].
- If the TQ is too large then starvation occurs.

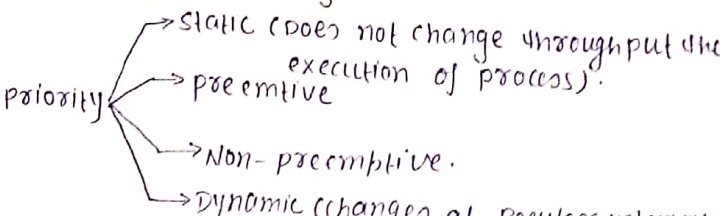
(e) longest Job first :-
- process having longest BT gets scheduled first
- criteria :- Burst time
- mode :- Non-preemptive.

(f) longest Remaining Time first :-
- mode :- preemptive

(g) Highest Response ratio next (HRRN) :-
- criteria :- Response ratio (RR) $= \frac{W+S}{S}$.

where, W → waiting time for a process so far.
S → service time of a process or BT.
- mode :- Non-preemptive.

(h) priority Scheduling :-

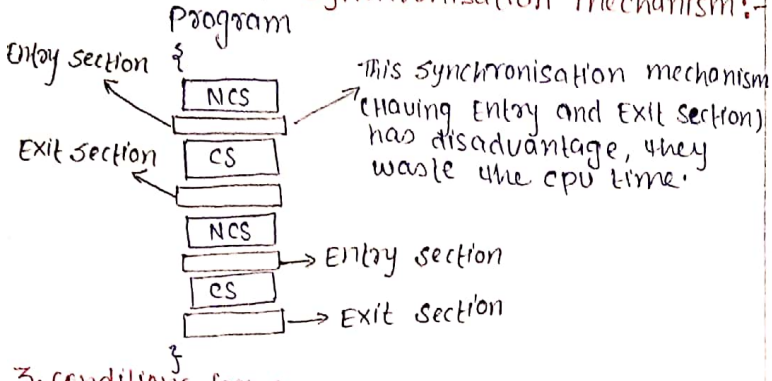


2. PROCESS SYNCHRONISATION :-

1. Need for synchronisation :-

- critical section :- It is the part of the program where shared resources are accessed by processor.
- Race condition :- The order of execution of instructions defines the results produced.

2. Introduction to synchronisation mechanism :-



3. Conditions for synchronisation mechanisms :-

Requirements for synchronisation mechanisms :-

- primary :- Mutual exclusion, program.
- Secondary :- Bounded waiting, portability (or) Architectural Neutrality.

4. lock variables :-

- Software mechanism implemented in user mode.
- Busy waiting solution.
- Can be used even for more than 2 process.

```
while (lock != 0)   → Entry Section
lock = 1

critical section ← CS

lock = 0            → Exit Section
```

5. TSL (Test set lock) :-

```
1. load lock R0     → TSL lock, R0
2. Store #1, lock
3. CMP R0, #0
4. JNZ Step 1.
```

- In TSL Never occure deadlock condition.
- TSL provides mutual exclusion, progress, Bounded waiting depends.
- TSL is dependent on Architecture.

6. Disabling Interrupts :-

Disabling the Interrupts
CS
Enables the interrupts.

- If we disable interrupt preemption is Guaranteed.
- Both ME and progress is Guaranteed.
- Bounded waiting is not guaranteed unless you maintain the Queue.
- Archilure dependent.

7. Turn variable or Strict alteration method :-

- Software mechanism implemented at user mode.
- Busy waiting solution.
- 2-process solution (P_0, P_1) or (P_1, P_j).
- Mutual exclusion is guaranteed but progress is not guaranteed.
- Bounded wait is present. Busy waiting is Absent.
- cannot be implemented for more than 2 process.

8. Peterson solution :-

- Software mechanism at user mode.
- Busy waiting solution.
- 2-process solution (P_0, P_1)
- It uses (Turn + Intrested) variable.
- Platform Independent.
- No overhead to the operating System (Every thing executes in user mode).

```
Entry
Critical Section
Exit
```

9. Synchronisation mechanism (without Busy waiting) :-

- Sleep and wake

10. Introduction to Semaphores :-

- Implementing semaphors at the user level will be dangerous and inconsistent.
- Variables on which Read, modify and update happen atomically in kernal mode. (No preemption).

Types of semaphore :-

(a) Counting semaphore :-

Decrement	Increment
Down	up
P	V
wait	Signal

(b) Binary semaphore :- (or mutexes) :-

- can have only 2 values 0 and 1.

3. DEADLOCKS :-

1. Introduction to deadlocks :-

- Starvation is long waiting.
- Deadlock is infinite waiting.

Necessary conditions for Deadlock :-

→ Mutual Exclusion
→ Hold and wait
→ No preemption
→ circular wait.

2. Deadlock handling mechanisms :-

Strategies for handling deadlock :-

(a) Deadlock ignorance :- Both window and linux uses Deadlock ignorance.

(b) Deadlock prevention :- Disable one of the four necessary condition.

(c) Deadlock avoidance.

(d) Deadlock Detection and Recovery :- most practical method.

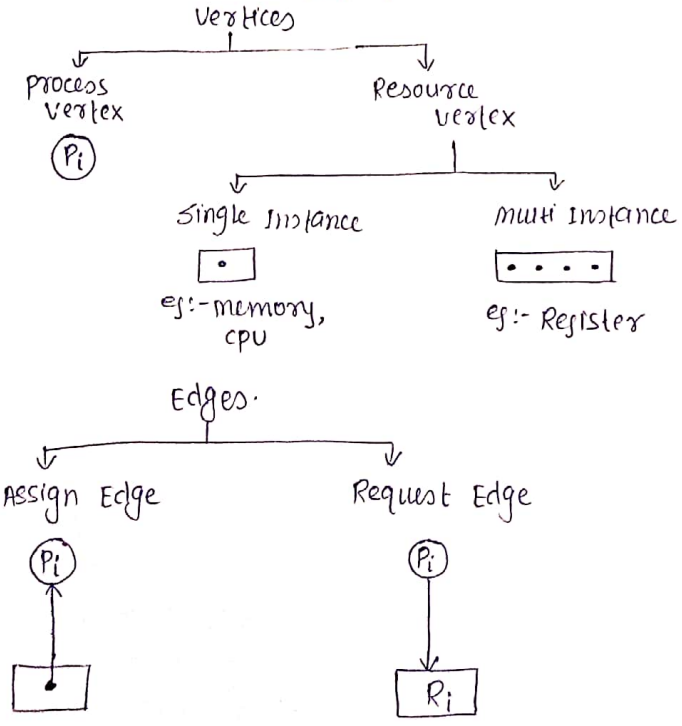
3. Deadlock prevention :-

Condition	Approach.
→ Mutual exclusion	Spool Everything
→ Hold and wait	Request all resources initially.
→ No preemption	Take resources away
→ circular wait	Order resources numerically

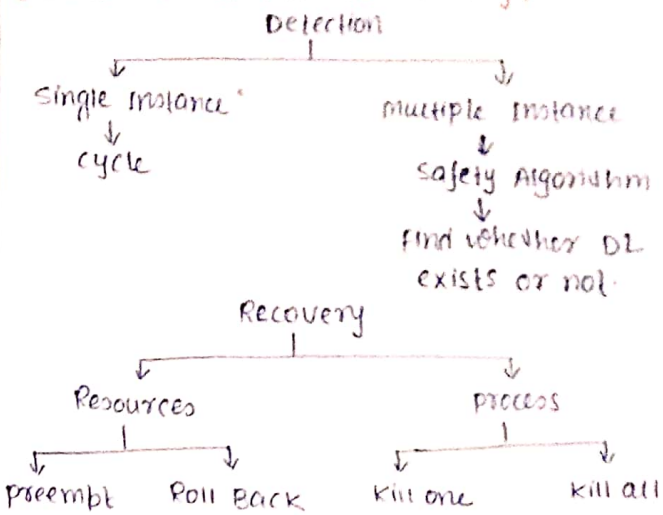
4. Safe, unsafe, Deadlock avoidance and Bankers Algorithm :-

- Safe state :- If you could satisfy the needs of all the process in some sequence then it is called safe state.

5. Resource allocation Graph :-



6 Deadlock Detection and Recovery :-



4. MEMORY MANAGEMENT :-

1. Need for multiprogramming and memory management :-

→ CPU can directly access, Registers, main memory, cache (if available).

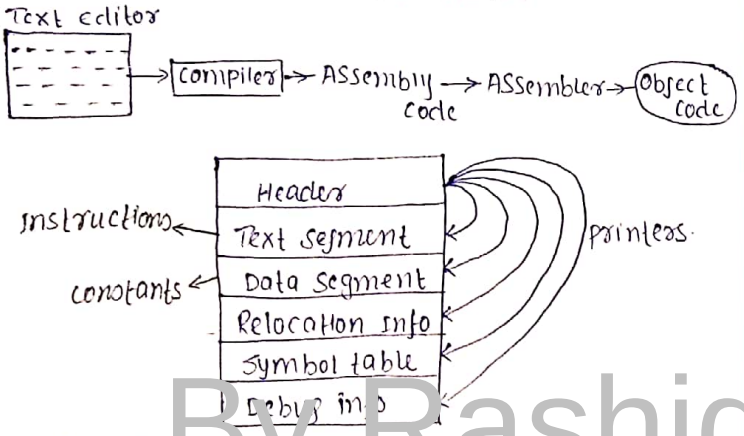
→ CPU utilization = The amount of time that you are using CPU running the program.

$$\text{CPU utilization} = 1 - p^n$$

where, n → no. of process in main memory.

→ Degree of multiprogramming :-
The no. of process that can be present in main memory at any given instant of time.

2. Object code, Relocation and linker :-



- The "Relocation info" field in the object file contains what are the lines that are changed to (Relocatable address → absolute address).
- "Symbol table" contains every symbol that is present in the program.

→ The main purpose of linker is Relocation and symbol Resolution.

3. Loader :-

- linker :-
  Symbols Resolution + Relocation.
- loader :-
  program loading + Relocation.

→ loading is nothing but copying something and loading it into main memory.

→ loading time ∝ process size.

→ various times that are present while the program and process creation are :-

(a) Compile time :- what ever you do during the compilation is called compile time.
Symbolic names → Relocatable Address.

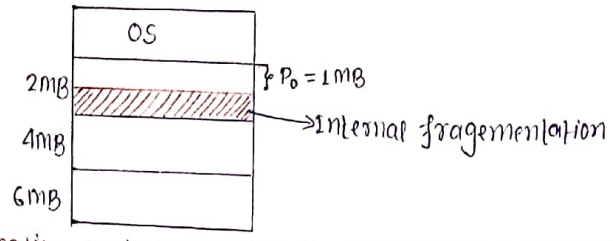
(b) link time :- If linker knows the address where it is loaded then linker converts one form of.

(c) load time :-
Relocatable address → Absolute address.

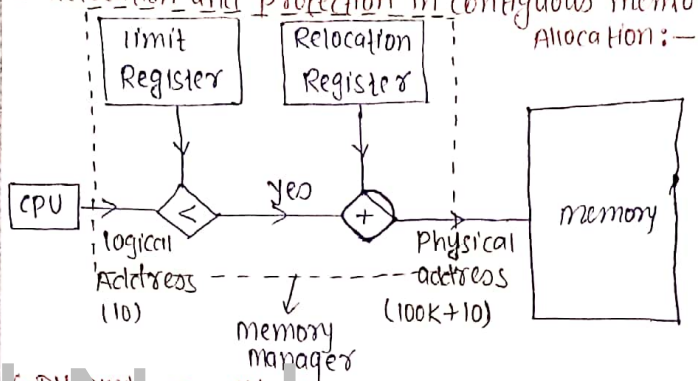
(d) Run time :-
Dynamic linking happens.
→ Dynamic link libraries are linked (Need support of OS).

4. Fixed Partitioning :-

- Contiguous memory allocation :-



5. Relocation and protection in contiguous memory Allocation :-



6. Dynamic Partitioning :- (variable Partitioning)

- The user space/memory is not divided into fixed size partitions. It is left as it is.
- If I go for dynamic partitioning there willn't be any internal fragementation.
- If there is internal fragementation then there is external fragementation.
- If there is no internal fragementation then there may or may not be external fragementation.
- External fragementation is present.
- advantages :-
  (a) Degree of multiprogramming is not fixed.
  (b) Size of the process is not limited by the size of partition.

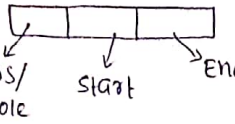
7. Bit map for Dynamic programming :-

→ Bit map :-
The memory is divided into small parts called "Allocation units".

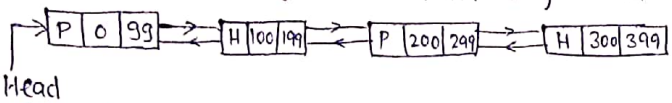
→ For every allocation unit the single bit 0 or 1 is assigned '0' represents the allocation unit is open/free and 1 represents the allocation Bit unit is occupied.

8. Linked list for Dynamic partitioning :-

- The structure of the linked list will be like this.



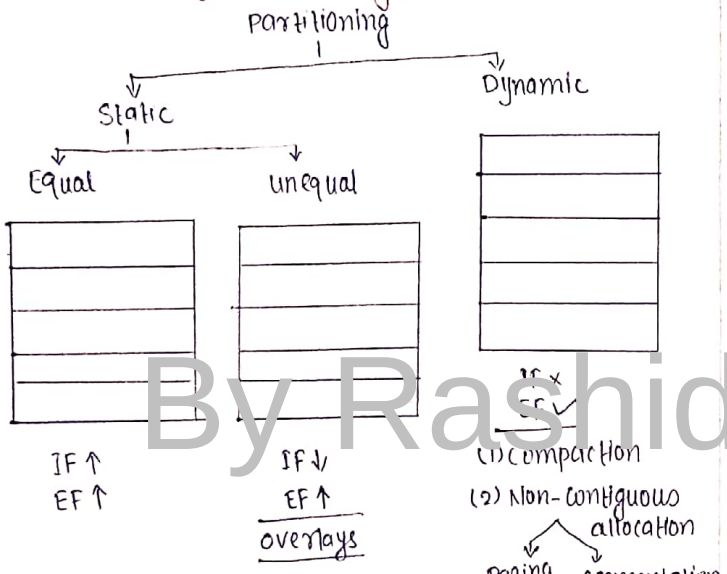
- For the above diagram the structure of linked list:



9. Various algorithm that are used on the linked list :-

- First fit
- Next fit
- Best fit
- worst fit
- Quick fit.

10. Summary of partitioning :-



11. Overlays :-

- Sometimes the size of the biggest partition will be less than the size of the process, in that case we should go with "OVERLAYS."
- only a part of the program is loaded in the memory at any given instant of time.
- The example of overlays is "assembler."

12. Need for paging :-

- The main problem that even the Dynamic partitioning could not eliminate is External fragementation.
- They have proposed is divide the process into small parts called pages and also divide the memory into small parts called frames.
- The division should be in such a way that the page size and frames size are equal.
- CPU generates the logical address and it is divided into 2 parts <page no, page offset>.

13. Basics of Binary address :-

$2^{10} = K$, $2^{20} = M$, $2^{30} = G$, $2^{40} = T$.

14. Physical address space and logical address space

→ The smallest addressable unit in the computer is "word."

- Physical address space :-

→ Physical address space is nothing but size of main memory.

PAS = 'm' words $\quad PA = \lceil \log_2 (m) \rceil$ bits

- logical address space / virtual address space :-

→ logical address space = size of a process.

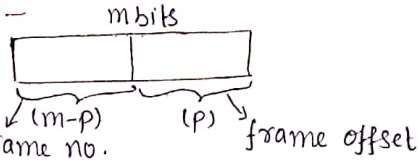
LAS = 'm' words $\quad LA = \lceil \log_2 (m) \rceil$ bits

→ page size = frame size = 'p' words

Page offset = $\lceil \log_2 P \rceil$ bits

15. Page table :-

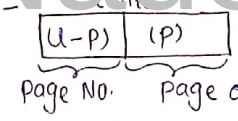
→ PA :-



Frame = 2^P words, PAS = 2^m words

No. of frames = $\frac{PAS}{\text{Frame size}} = \frac{2^m}{2^P} = 2^{m-P}$ frames

→ LA :-



Page size = 2^P words, LAS = 2^l words

No. of pages = $\frac{2^l}{2^P} = 2^{l-P}$ pages.

⇒ No. of entries in the page table = No. of pages in the process = 2^{l-P}.

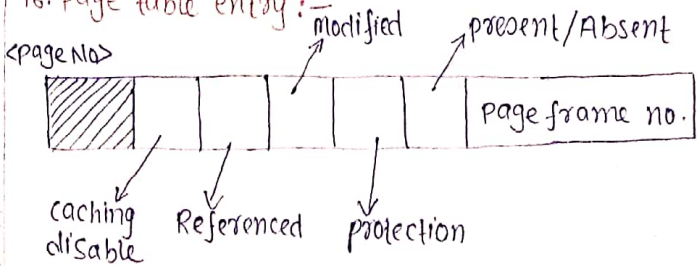
let 'e' be the size of each entry,

Page table size = $2^{l-P} * (e)$ bytes.

If the value of 'e' is not given then,

Page table size = $2^{l-P} * (m-P)$ bytes

16. Page table entry :-



17. Need for multilevel paging :-

→ If page table size > the page size then we go for multilevel paging.

18. Finding optimal page size :-

No. of pages = $\frac{VAS}{PS}$

$$\text{overhead} = \left[\frac{np}{2} + s\left(\frac{1}{p}\right) * e\right]$$

$$p = \sqrt{se}$$

where, p → optimal page size,

s → Average VAS

e → PTE

n → no. of pages the section is divided.

19. Virtual Memory Introduction :-

→ The VM concept is used when the process size is greater than the MM.

→ VM is provided by operating system but overlays is by manual intervention.

20. Translation lookaside buffer (TLB) :-

→ The cache is faster than MM and cheaper than Registers.

→ The TLB also sometimes called ASSOCIATIVE MEMORY because every element of this memory is having 'TAG'.

Effective Access time (EAT) = $P(t+m) + (1-P)(t+2m)$ (single level page table)

where,

t → Time for checking the key in TLB.

m → Time for accessing the frame in MM.

TLB Hit = P => TLB MISS = 1-P.

If there are K levels then,

$EAT = P(t+m) + (1-P)(t + km + m)$

21. Page fault :-

- Refering to a page that is not present in the memory is called page fault.
- Demand Paging :- Don't load any page until it is required.
- No. of page references equal to No. of page faults then it is called thrashing.
- EMAT in case of page fault is =

P * (service time / page fault time)

+ (1-P) (memory Access time)

Page fault rate.

EMAT = P * (service time) + (1-P) (memory access time)

Also called page fault time.

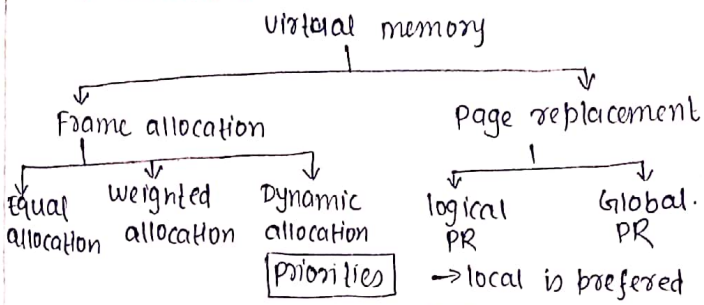
EMAT = P * (service time + MA) + (1-P) (MA)

↳ very Negligible.

22. Inverted page table :-

IPTS = (No. of frames) * IPTE

23. Importance of Frame allocation and page Replacement :-



→ local is prefered more than global as global can cause more thrashing.

24. Page Replacement :-

- Optimal page Replacement :- Replace the page that will not be refered for long time.
- least Recently used (LRU) :- Replace the page which has not been referred for a long time.
- FIFO :- First inserted page should be replaced.
- Belady Anamoly :- If we increase the number of frames FIFO is behaving strangly for some examples.

25. Stack Algorithm :-

→ The main reason for Belady Anamoly is 'stack properity'.

→ optimal property is a stack Algorithm.

→ whenever a algorithm is a stack algorithm it does not follow Belady Anamoly.

26. Some interesting behaviour of optimal page Repl:-

→ In case of loops optimal and MRU (mostly Recently used) are same.

→ In case of loops both LRU and FIFO behaves in worst manner (more PF).

27. WORKing set algorithm :-

→ Initially the process requires less frames but gradually the neccecity of the frames increases

→ This also has the parameter 'window size(A)'.

working set size ≤ window size

28. Segmentation :-