2020

GATEIESMENTOR

GATEIESMENTOR

# GATE CSE MADEEASY HAND NOTES-2020

# 9.SUBJECT NAME- OPERATION SYSTEM

GATEIESMENTOR

# Operating System

# GATE NOTES



GATEIESMENTOR

## COMPUTER SCIENCE
## MADEEEASY CLASS HAND NOTES

## Page-137

NAME: Rakesh Nama. STD.: _____ SEC.: _____ ROLL NO.: _____ SUB.: O.S

| S. No. | Date | Title (1.01.2018) 3 Day training 1 to 3 | Page No. | Teacher's Sign / Remarks |
|---|---|---|---|---|
| 1. | | • Introduction of Operating system. | 1-6 | |
| 2. | | • Process Management : | | |
| | | (a) Process concept. (1) | | |
| | | (b) Threads. (1) | | |
| | 40% | (c) Interprocess communication. (2) | | |
| | | (d) Concurrency. (CS) (2) | | |
| | | (e) Synchronization. (2) | | |
| | | (f) Deadlock. (3) | | |
| | | (g) CPU Scheduling. (*) (1) | | |
| 3. | | • Memory Management : | | |
| | 40% | (a) Memory mangement & virtual memory. (4) | | |
| 4. | | • File System : | | |
| | | (a) File and Device management. (5) | | |

(www.gatementor.in)

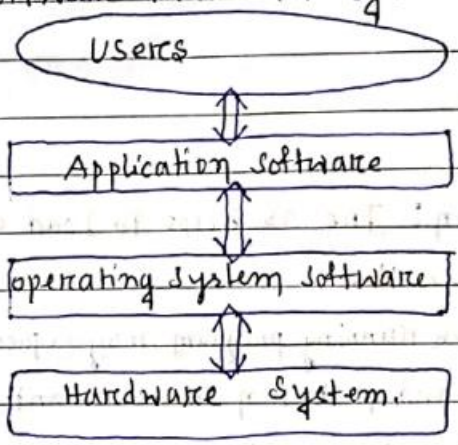
Saathi

Date 06/04/2019

# INTRODUCTION OF OS

- What is Operating System?

→ An Operating System acts as an Intermediary between the user of a computer and the computer hardware. An operating System in a software that manage the computer hardware.




(Basic Diagram)

- Hardware: It provides the basic computing recources for the System. It consists of CPU, Memory and I/O Devices.

- Application programs: Define the ways in which these recources are used to solve user's computing problems. e.g Word processors, Spreadsheets, Compilers, and web browsers, Accounting software.

- Components of Operating System:
  - process management.
  - Main memory management.
  - File management.
  - I/O System management.
  - Secondary-storage Management.
  - protection System.
  - Networking.
  - Command-Interpete system.

- Function of operating System:
  1. program execution.
  2. I/O operation.

Date __/__/__

3. File system manipulation.
4. Communication.
5. Error Detection.
6. Resource allocation.
7. Accounting.
8. protection system.

- program execution: The OS helps to load a program into main memory and run it.
- I/O operation: Each running program may request for I/O operation and for efficiency and protection the user can't control I/O devices directly. Thus, the OS must provide some means to do I/O operation.
- File system manipulation: file are the most important part which needed by programs to read and write the files and files may also be created and deleted by name or by the programs. The OS is responsible for the file management.
- Communication: OS performs the communication among various types of processes in the form of shared memory. In the multitasking environment, the processes needed to communicate with each other and to exchange their information. These processes are created under a hierarchical structure where the main process is known as parent process and the subprocess are known as child processes.
- Error detection: It is neeessary that OS must be aware of possible errors and should take the appropriate action to ensure correct and consistent computing.
- Resource Allocation: Allocation of resources to the various processes is managed by O.S.
- Accounting: OS may also be used to keep track of the various computer resources and how much and which users are using these resources.
- protection: If a computer system has multiple users and allows

Date ___/___/___

the concurrent execution of multiple processes, then the various processes must be protected from one another's activities.

- System call:

```
main()
{
  pf("....");
  sf("....");
}
```

Internally calls write() system calls in order to communicate with the monitor.

It is the request mode by the user program to the OS in order to get any kind of service.

- Types of Operating System:

The Operating System can perform a single operation and also Multi-ple operations at a time. So there are many types of Operating Systems those are organized by using their working Techniques.

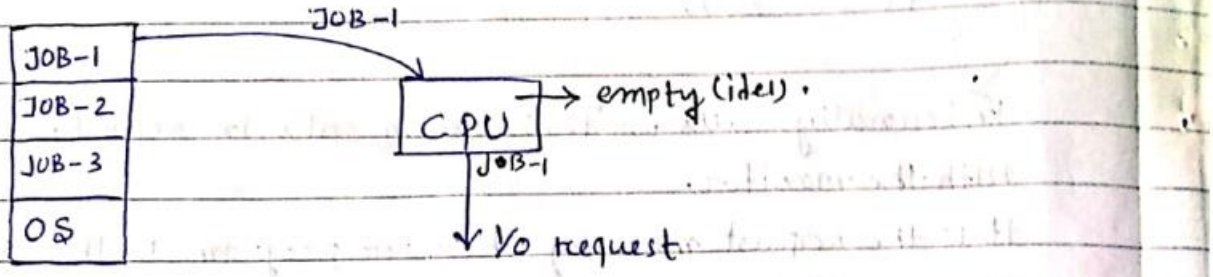
1. Serial processing: The serial processing Operating System are those which performs all the instructions into a sequence manner. ore the instructions those are given by the user will be executed by using the FIFO Manner. means first in first out. Mainly the punch cards are used for this. In this all the jobs are firstly prepared and stored on the card and after that card will be entered in the system and after that all the instructions will be executed one by one. But the main problem is that a user doesn't interact with the system while he is working on the system, means the user can't be able to enter the data for execution.

2. Batch processing (O.S): The Batch processing is same as the serial processing Technique. But in the Batch processing similare types of job are firstly prepared and they are stored in the card and that card will be submit to the system for the processing. ~~The main~~

JOB = (CPU time + Yo time)
execution time

Date ___/___/___

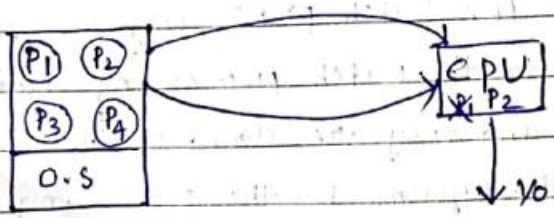
The main problem is that the jobs those are prepared for execution must be the same type and if a job requires for any type of input then this will not be possible of the user. The batch contains the jobs and all those jobs will be executed without the user intervention.




Memory

- If the job is fully completed, then only another job will be scheduled.
- Increased CPU idelness.
- The Throughput of the system will decreased.
  Throughput (the number of jobs completed per unit time is called throughput of the system).

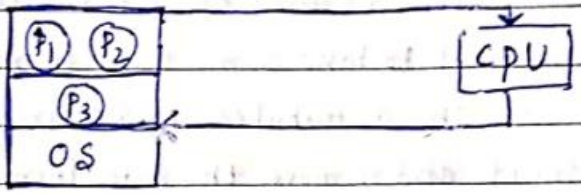
3. Multiprogramming Operating System: Execute Multiple programs on the system at a time and in the Multiprogramming the cpu will never get idels, because with the help of Multiprogramming we can execute many programs on the System and when we are working with the program then we can also submit the second or another program for running and the CPU will then execute the second program after the completion of the first program. And in this, we can also specify our input means a user can also interact with the system.




- Increased CPU Utilization.
- Increased throughput.

Date ___/___/_____

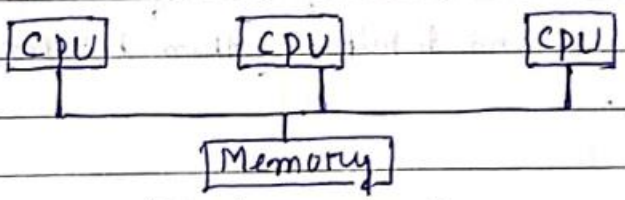
4. Multitasking operating System: Multiple applications ~~app~~ operates at the same time.
- Multitasking is an extension to Multiprogramming.
- The jobs will be executed in the time sharing mode.




(We see illusion)

Memory (Multitasking)

5. Multiprocessing operating System: In the Multi processing there are two or more CPU in a single operating System, if one CPU will fail, then other CPU is used to providing backup to the first CPU. With the help of ~~put~~ multiprocessing, we can execute many jobs at a time. All the operations are divided into the number of CPU's. If first CPU completed his work before the second CPU, then the work of second CPU will be divided into the first and second CPU.




(Multiprocessing)

6. Realtime Operating System: In this, Response time is already fixed. Means time to display the results after processing has fixed by the processor or CPU. Real time system is used at those places in which we requires higher and timely response.
- Hard Real-time System: In the hard real-time system, time is fixed and we can't change any moments of the time of process the data as we enters the data.
- Soft Real-time System: In the soft real-time system, some moments can be changed. Means after giving the command to the CPU, CPU performs the operation after a microsecond.

Date __/__/____

7. Distributed Operating System: Distributed Means data is Stored and processed on Multiple locations. When a data is stored on to the multiple computers, those are placed in different locations. Distributioned means in the network, Network collections of computers are connected with each Others. Then If you we want to take some data from other Computer, then we uses the distributed processing system. And we can also insert and remove the data from out location to another location. In this Data is shared between many users. And we can also Access all the input and output devices are also accessed by Multiple users.

8. parallel Operating System: These are used to interface multiple networked computers to complete task in parallel. parallel OS are able to use Software to manage all of the different resources of the computers running in parallel, Such as memory, caches, storage space, and processing power. A parallel OS works by dividing sets of calculations into smaller parts and distributing them between the machines on a network.

Saathi

Date / /

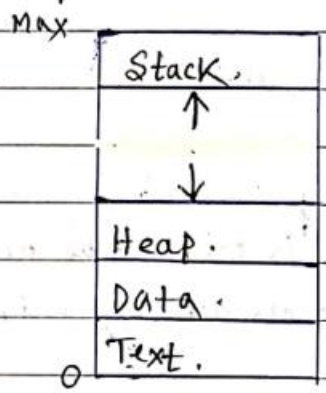
# PROCESSES - (Process management)

RBR

- **Process:** A process can be defined in any of the following ways:
  - A process is a program in execution.
  - It is a asynchronous activity.
  - It is the entity to which processors are assigned.
  - It is dispatchable unit.
  - It is the unit of work in a system.

A process is more than the program code. When we execute a program, it becomes a process which performs all the task mentioned in the program.

When a program is loaded into memory and it becomes a process, it can be divided into four sections - stack, heap, text and data. The following image shows a simplified layout of a process inside main memory -




(process in memory)

- **Stack:** The process stack contains the temporary data such as function parameters, return address, and local variables.
- **Heap:** This is dynamically allocated memory to process during its run time.
- **Data:** This section contains the global and static variable.
- **Text:** This includes the current activity represented by the value of program counter and the contents of the processo's register. (Code section)

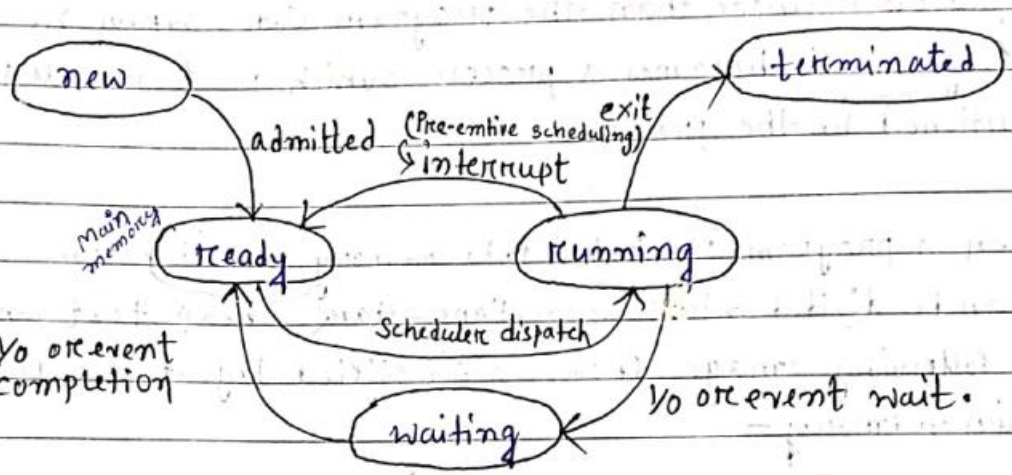
Date __/__/__

RBR

## Process State Diagram:

When a process executes, it passes through different states. This stages may differ in different OS, and the name of these states are arbitrary.

A process can have one of the following five states at a time.




(Diagram of process state)

1. Start: The process is being created..
2. Running state: A process is said to be running if it has the CPU, that is, process actually using the CPU at that particulate instant.
3. Waiting (or Blocked) state: A process is said to be blocked if it is waiting for some event to happen such that as an I/O completion before it can proceed. Note that, a process unable to run until some external event happens.

4. Ready: A process is said to be ready if it use a CPU if one where available. A ready state process is runable but temporarily stopped running to let another process run.
5. Terminated State: The process has finished execution.

Date __/__/____

RBR

## Process Control Block (PCB):

A process in an OS is represented by a data structure known as process control block (PCB) or process descriptor.

A PCB keeps all the information needed to keep track of a process.

The architecture of PCB is completely dependent on operating system and may control different information in different OS.

| Process ID |
|---|
| p-state |
| Pointer |
| Priority |
| Program counter |
| CPU register |
| I/O information |
| Accounting information |
| etc... |

(PCB Diagram)

1. process State: The current state of process i.e, whether it is ready, running, waiting, or whatever.
2. process privileges: This is required to allow/disallow access to system resources.
3. process ID: Unique identification for each of the process in OS.
4. pointer: A pointer to parent process.
5. program Counter: PC is a pointer to the address of the next instruction to be executed for this process.
6. CPU registers: various CPU registers where process need to be stored for execution for running state.
7. Accounting information: This includes the amount of CPU used for process execution, time limits, execution ID etc.
8. I/O status information: This includes a list of I/O devices allocated to the process.
9. CPU-scheduling information: process priority and other scheduling Informatio

Multiprogramming

with preemption — without preemption.

Date __/__/____

Saathi

which is required to schedule the process.

10. Memory management information: This includes the information of page table, memory limits, segment table depending on memory used by the ~~oper~~ OS.

RBR

## Schedulers:

Schedulers are special system ~~S~~ software which handle process scheduling in various way. Their main task is to select the jobs to be submitted into the system and to decide which process to run.

Schedulers are of three types -

1. long Term scheduling.
2. short Term scheduling.
3. Medium Term scheduling.

1. Long Term Scheduler or job scheduler:

LTS is responsible for creating new processes and bringing them into System. LTS select processes from job pool (where process are kept for later execution) and load them into memory for execution. LTS control degree of Multiprogramming.

New —LTS→ Ready

2. Short term Scheduler or CPU scheduler:

STS ~~re~~ is responsible for scheduling one of the process from ready sate to running State:

ready —STS→ run

STS selects from main memory among the process that are ready to execute and allocate the CPU to one of them.

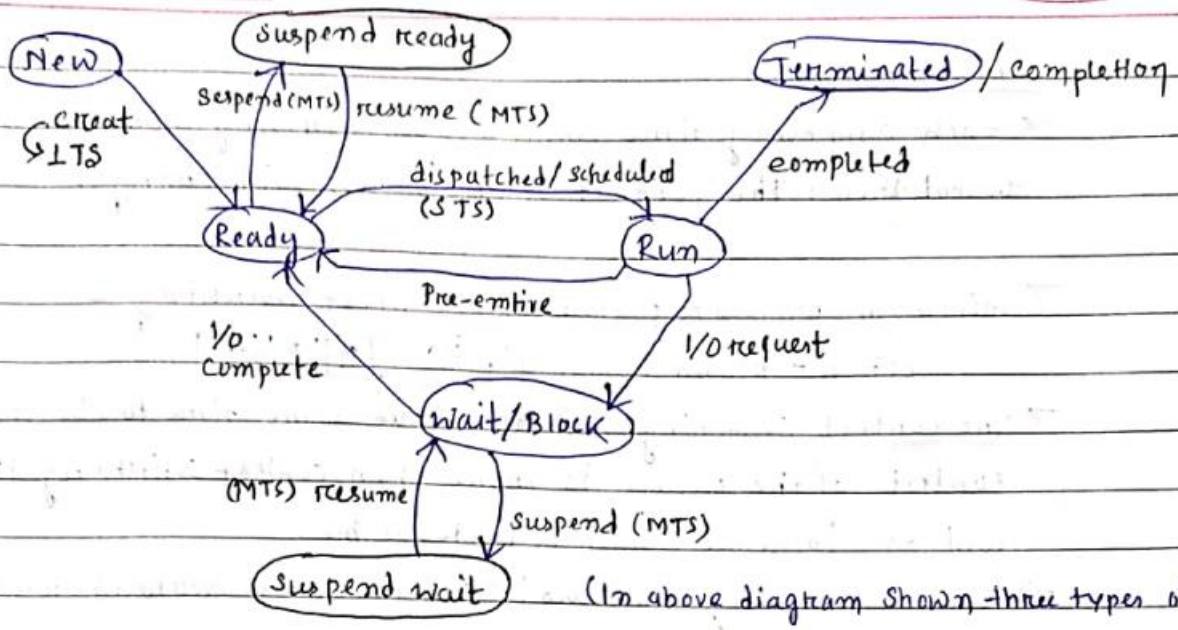
3. Medium Term scheduler:

MTS is responsible of suspending and resuming the processes.

main memory ⟷ secondary memory

The medium term scheduler are also called a SWAPPER.

Date ___/___/_____



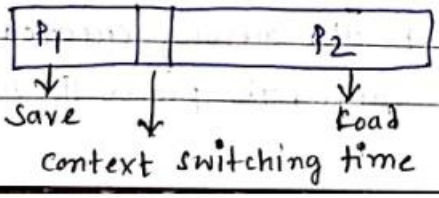

(In above diagram shown three types of schedulers)

| Aspects : | LTS | MTS | STS |
|---|---|---|---|
| called as | It is Job scheduler | It is a process swapping | It is a cpu scheduler |
| Speed | Spee is less than STS | Speed is in between both LTS and STS. | Speed is fastest among two Schedulers. |
| Multiprogramming | It Control the Degree of Multiprogramming | It reduce the Degree of multiprogramming. | It provids less control over Degree of multiprogramming |
| Time-sharing system. | It is almost absent or minimal in time sharing System. | It is part of Time sharing System. | It is also ~~nominal~~ minimal. in time Sharing System. |
| processes | It Selects process from pool and loads them into memory for execution. | It can reintroduce the process into memory and execution can be continued. | It selects those process which are ready to execute. |

PBR

## Context Switching :

Saving the context of 1 process and loading the context of another process is called Context Switching.



Date __/__/____

Some points:

→ Each and every time when process is moving from one state to another, the context of the process will change.

→ Minimum process required for context switching '2'.
Exception - Round Robin '1'. (P1) [P1 | P1 | ..]

→ The context switching will also take some time to, so if the context of the process is more then context switching time will also increase which is undesirable.

→ Context switching time also is considered as overhead for the system.

• Dispatcher:
Dispatcher is responsible for saving the context of 1 process and loading the context of another process. The context switching is done by dispatcher.

RBR • The fork():

→ System call fork() is used to creat processes. It takes no arguments and returns a process ID.

→ The purpose of fork() is to create a new process, which becomes the child process of the caller.

→ After a new child processes is created, both processes will executed the next instruction following the fork() system call.

→ Therefore, we have to distinguish the parent from the child.

This can be done by testing the returned value of fork():

→ • If fork() return a negative value, the creation of a child process was unsuccessful.

→ • If fork() return a zero to the newly created child process.

→ • If fork() return a positive value, the process ID of the child process.

Date __/__/____

to the parent. Normally the process ID is an Integer. Moreover, a process can use function ~~getpdf~~ getpid() to retrieve the process ID assigned to this process.

• Conclusions –

→ If the program contains n fork() calls it will creat $2^n - 1$ child processes.

→ When the child process is created by using fork system call, both parent and children will have.

→ Relative address = Same for both parent and child process.

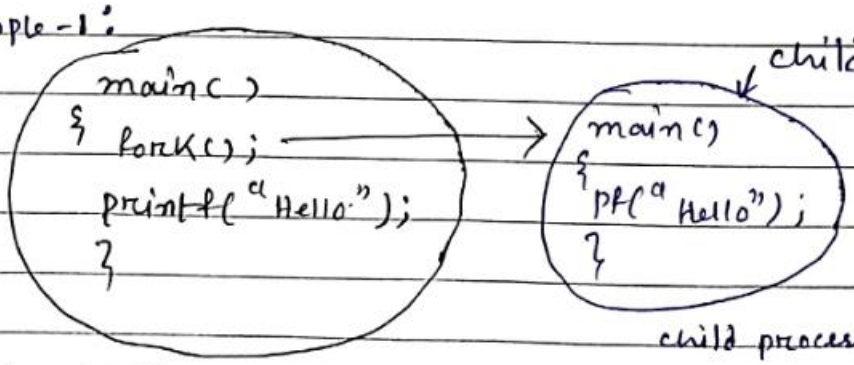
→ Absolute address = Different for both parent and child proces.

• fork() system call Implementation –

```
main
{
 int pid;
 pid = fork();
 if (pid < 0)
  { pf("child process creation failed");
 else if (pid == 0)
  { pf("child process");
  }

 else
  { pf("parent process");
  }
}
```

• Example-1:




output: Hello
Hello

2 process / 2 time Hello. / 1 child ($2^n - 1 = 2^1 - 1 = 1$).

Saathi

Date __/__/____

RBR

Q. Question on process states —

Consider a system with 'N' CPU processors and 'M' processes, then,

| | min | max |
|---|---|---|
| ready | 0 | M |
| running | 0 | N |
| block or wait | 0 | M |






Date __/__/____

# THREADS

## Thread:

A Thread is a single sequence stream within a process. Because threads have some of the properties of processes, they are sometimes called lightweight processes.

→ A thread can be any several states (Running, Blocked, Ready, or Terminated)

→ Each thread has its own stack.

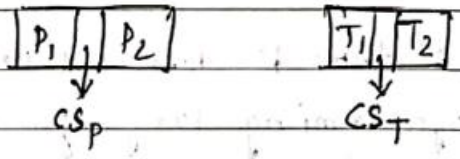
→ A thread has or consists of a program counter (PC), a register set, and a stack space.

→ Threads are not independent of one other like processes a result threads shares with other threads their code section, data section, OS resources also known as task, such as open files and signals.

## Advantages of Thread:

1. Responsiveness:
   → If the process is divided into multiple threads, then 1 thread is completed, then immediately output will be responded.

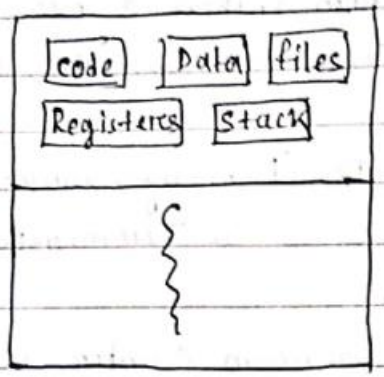
2. Faster context switching:
   → The context switching time between the threads is very very less as compared to context switching time between the process. Because the threads will have less context compared to process.



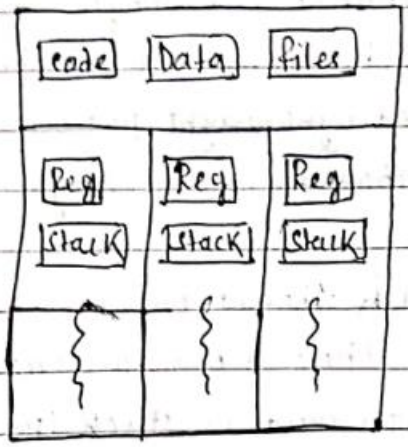

$$CS_T << CS_P$$

Date __/__/____

3. Resource Sharing:




single threaded process




Multi threaded process

→ The resources like code, data, and files will be shared among all the threads with in the process but every thread will have its own stack and registers.

4. Effective Utilization of Multiprocessor Systems:

→ If the process is divided into multiple threads, then different threads of the process can be scheduled onto a different CPU, so that the process execution will be faster.

5. Enhanced throughput of the System:

→ If the process is divided into multiple threads, if we consider 1 thread as 1 job then number of job completed per unit time will increased and throughput of the system will be enhanced.

6. Economical:

→ The implementation of threads does not requires any cost. There are various programming languages API's which support implementation of thread e.g JAVA's API, thread API

Saathi

Date / /

• Types of Threads :

Threads are categoriged into 2 types —

→ User level threads.
→ Kernel level threads.

• Difference between them —

| User level thread | Kernel level thread |
|---|---|
| → These are created by user or programmers. | → These are created by operating System (OS). |
| → OS can't recognige the user level threads. | → OS recogniges the Kernel level threads. |
| → If one user level thread performing Block system call, then entire process will be blocked. | → If one Kernel level thread performing Blocking System call, then another thread will continue the execution. |
| → Dependent threads. | → Independent threads. |
| → Designing user level threads is easy. | → Designing Kernel level threads is complicated. |
| → Less context. | → More Context. |
| → NO H/W support required. | → Scheduling of Kernel level threads requires H/W support. |

*. Thread Scheduling also known as GANG Scheduling.

• There are three common ways of establishing relationship between user threads and kernel threads —

→ Multi ... are ~~two~~ three types —
- Many-to-many model.
- one-to-one model.
- Many-to-one model.

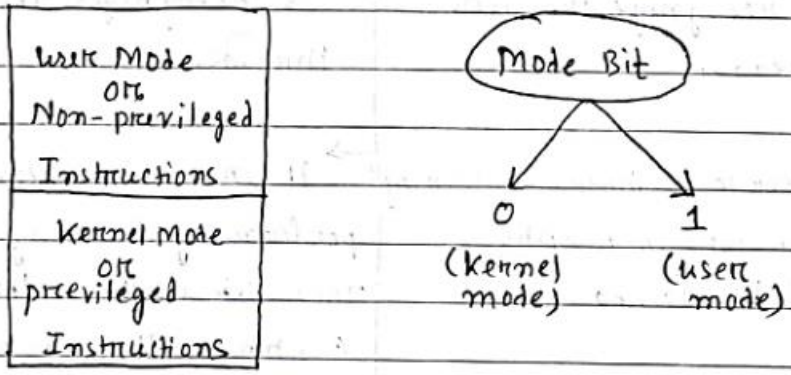
Date __/__/____

1. Many-to-many model -
→ Allows many user level threads to be mapped to many Kernel threads.

2. One-to-one model -
→ Each User level threads maps to Kernel thread.

3. Many-to-one model -
→ Many user level threads mapped to single Kernel thread.

## Double Mode of Operation :




→ The O.S executes the Instructions in 2 different modes -
1) User mode.
2) Kernel mode.

→ The dual mode of Operation is required to protection and security to the O.S and also to the user programs from Errant users (Concepted Users).

→ In which particular mode the current instruction is executing will be decided by mode bit. → 0 / 1

→ At boot time the system always starts in the Kernel mode.

→ Depending Upon the type of Instruction the O.S will be decide In which particular mode Instruction has to be executed.

Date ___/___/_____

→ Generally privileged instructions will be executed in the Kernel mode, and Non-privileged instructions will be executed in the User mode.

- privileged Instructions: (examples)
  1) Context Switching.
  2) Disabling Interrupts.
  3) set the time of clock.
  4) Changing the memory map.
  5) I/O operation (Reading/file data from disc)
- Non-privileged instructions: (examples)
  1) Reading time of the clock.
  2) Sending the final print to the printer.
  3) Reading the status of the cpu.

✓ DONE

Date ___ / ___ / ___

Saathi

Date

# Inter-Process Communication

→ process executing concurrently in the operating system may be either independent or cooperating processes.

→ A process is independent; if it can't affect or be affected by other processes executing in the system.

→ A process ~~is cooperating process, if~~ that shares data with other processes is a cooperating process.

→ Advantages of process cooperation are –
  → Information sharing.
  → Computation speed up.
  → modularity and convenience to work on ~~may~~ many task at ~~the~~ same time.

→ Cooperating processes require an inter process communication (IPC) mechanism that will allow that to exchange data and information.

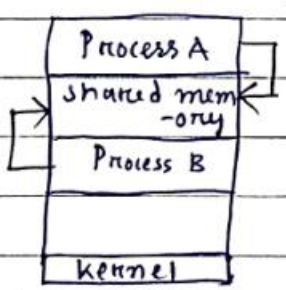
→ There are 2 fundamental models of IPC:
  I) Shared Memory.
  II) Message passing.

1. Shared memory:

→ In this method 2 or more processes shared a single chunk of memory to communicate.

→ used for exchanging larg amount of data.

→ faster than message passing.



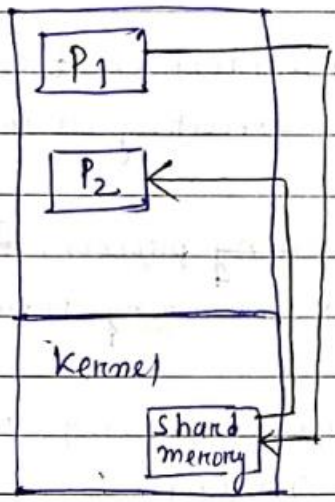

(Shared memory)

Date __/__/____

## 2. Message passing:

→ shared memory created in the kernel.

→ System call such as send and receive used for communication.




→ used for exchanging small amount of data.

→ slower than shared memory.

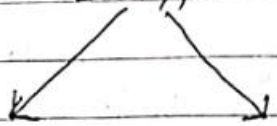
(Message passing)

Saathi

Date __/__/____

# PROCESS SYNCHRONIZATION

*. The processes with respect to synchronization are of 2-types



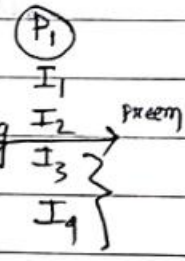
Co-operating process | Independant process.

The execution of 1 process affects or affected by other process then these processes are said to be co-operative processes, Other wise they are said to be independant process.

→ Affection occurs due to shared variable, common shared Resource / Data.

- Some Import points:

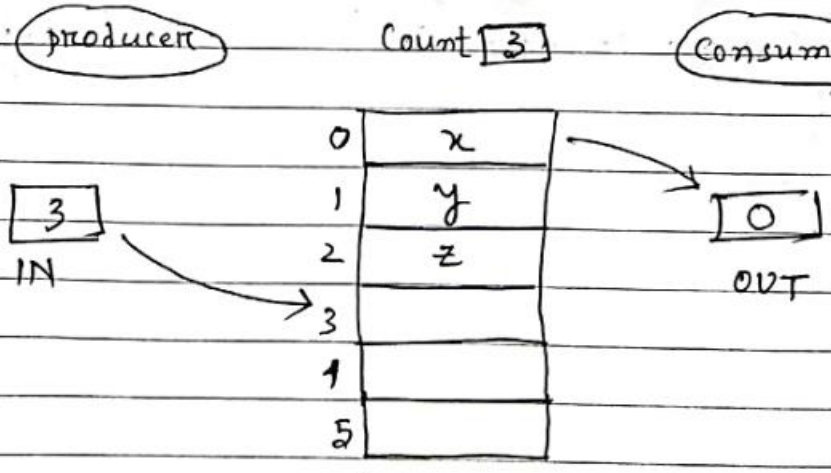
→ The pre-emption is just a temporary stop, the will come back and continue the execution.




→ If there is any possibility of solution becoming wrong by taking the pre-emption.

→ If any solution has the be deadlock, then the progress is not satisfied.

- producer-consumer problem:



Date

IN → is a variable used by producer to Identify the next empty slot in the Buffer.

OUT → is a variable used by Consumer to Identify the slot from where it has to consume the item.

Count → is a variable used by consumer and producer to identify the no of items present in the Buffer at any point of time.

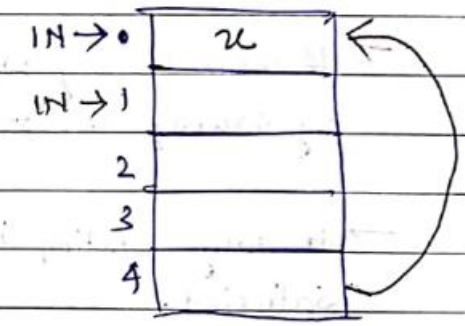
Conditions :-

1) → When the buffer is full, the producer is not allowed to produce the item.

2) → When the buffer is empty consumer is not allowed to consume the item.

• Producer Code →

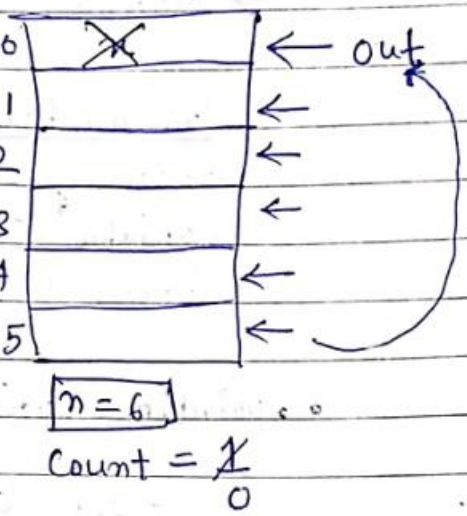
```
int count = 0;
void producer (void)
{
    int temp;
    while (TRUE)
    { produce_item (Itemp);
      while (count == N);
       Buffer [IN] = Temp;
       IN = (IN+1) MOD N;
       Count = count + 1;
    }
}
```



Date

- Consumer Code →

```
void consumer(void)
{
    int itemc;
    while(TRUE)
    {
        while (count==0);
        Item c = Buffer[OUT];
        OUT=(OUT+1) MODN;
        count = count -1;
    }
}
```




- producer consumer problem Analysis:

⊛ producer

count=count+1;

⇓

I. LOAD $R_p$, m[COUNT];
II. INCR $R_p$
III. STORE m[COUNT], $R_p$

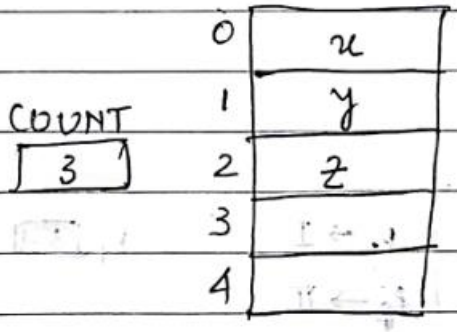
⊛ Consumer

count=count-1;

⇓

I. LOAD $R_c$, m[COUNT]
II. DECR $R_c$
III. STORE m[COUNT], $R_c$

Analysis – Race around condition




{different count value for Producer and consumer}

Now,

P→I $R_p$ 3
preempt P→II $R_p$ 4

C→I $R_c$ 3
C→II $R_c$ 2
✓C→III 2 ✓

✓P→III 4 ✓

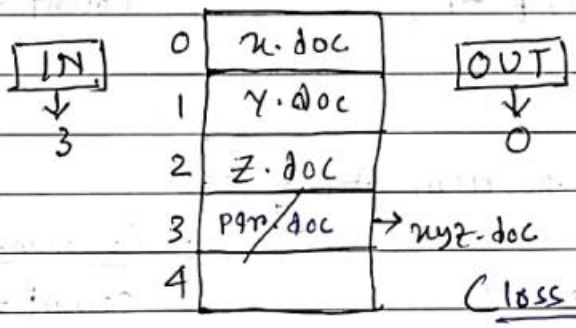
Date ___ / ___ / _____

- producer Consumer problem Conclusion –

→ problems of Non-Synchronigation.

Inconsistency :- The producer and consumer are not properly Synchroniged while ~~sharing~~ shearing the common variable COUNT. Hence it is leading inconsistency. No proper Synchronigation between producer and consumer

- pointer - Spooler Domain problem –



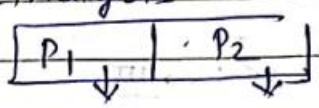

(IN) → is a variable used by all processes to identify the next empty place in the spooler directory.

(OUT) → is a variable used ~~by~~ only by printer to identify the slot from where it has to Print the document.

Enter file –

1) load $R_i$, m[IN]
2) STORE SO[$R_i$], "file name".
3) INCR $R_i$.
4) STORE m[IN], $R_i$.

Analysis –




$P_1 \rightarrow I$ $R_1$ [84]
$P_2 \rightarrow II$
$P_1 \rightarrow III$ → preempt
$P_2 \rightarrow I$ $R_1$ [3]
$P_2 \rightarrow II$

Date

- problem of Non-Synchronigation:

→ Inconsistency: The producer and consumer are not properly Synchroniged while sheaning the common variable COUNT. Hence it is leading to inconsistency. No proper synchronigation between producer and consumer.

→ Data Loss of Data: The process are not properly synchroniged while sharing the common variable [DR], hence it is leading to lose of data.

→ Deadlock: If the process are not properly synchroniged the there is also a possibility of deadlock.

- Definations:

1) Critical Section (C.S): The portion of program text where sheared variables are placed.

example- In producer/consumer problem:

count = count + 1 → C.S

2) Non-Critical section: The portion of program text where the Independent code of the process will be placed.

3) Race Condition: The final value of any variable depends on the execution sequence of process. This condition is called as Race condition.

- The Critical Section ~~problem~~ - Structure -

```
do
{
    entry section
        critical section.
    exit section
        reminder Section / NCS.
}
while (TRUE);
```

Date __/__/____

- Synchronization conditions :-

1) MUTUAL EXCLUSION: No two process may be simultenously present inside the critical section at any point of time. Only 1 process is allowed into critical section at any point of time.

2) progress : No process running outside the critical section should block the other interested process from entering into c.s when the c.s if free.

3) Bounded Waiting: No process should have to wait forever from entering into critical section. There should be a bound on getting chance to enter into critical section. If Bounded waiting is not satisfied, then it is possible for starvation.

- Solution Types :

1) Software Type :-
   a. Lock variables.
   b. strict Alternation (or) Decker's Algorithm.
   c. peterson's Algorithm.

2) Hardware type :-
   a. TSL Instruction set (Test & set lock)

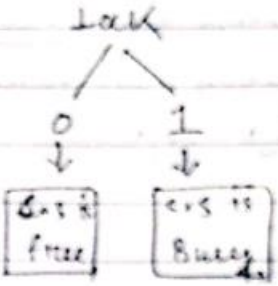
3) O.S Type :-
   a. Counting Semaphore.
   b. Binary semaphore.

4) programming language compiler support type :-
   a. Monitors.

## Lock variables : → [M·E → X, Progress → ✓, B·W → X]

Entry section -

I. load $R_i$, m[lock]
II. CMP $R_i$, #0
III. JNZ tostep①
IV. STORE m[lock] #1
V. [C S]
VI. STORE m[lock], #0



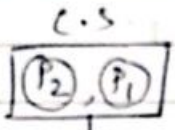

let, lock ⇒ 0 X 1

R·Q | $P_1$ | $P_2$

$P_1 \to$ I  R [0]
$P_1 \to$ II
$P_1 \to$ III → preempt.

$P_2 \to$ I  R [0]
$P_2 \to$ II
$P_2 \to$ III
$P_2 \to$ IV, V

$P_1 \to$ IV,
$P_1 \to$ V




M·E-X

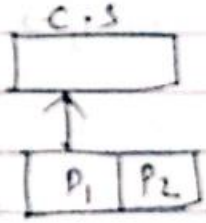
✓ (Mutual exclusion not satisfy) - X

---

Entry section.

I. LOAD $R_i$, m[lock]
II. CMP $R_i$, #0
III. JNZ tostep①
IV. STORE m[lock] #1
V. [C S] } critical section
VI. STORE m[lock] #0. } remainder section

} entry section




$P_2 \to$ I  R [0]
$P_2 \to$ II, III
$P_2 \to$ IV → preempt

$P_1 \to$ I  R [1]
$P_1 \to$ II, III

(Same no. of process infinite)

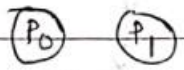
✓ process are satisfy ✓
✓ Bounded Waiting - X
(If process are finite then B·W satisfy other wise not satisfy)

Date ___/___/_____

- Strict Alternation (or) Decker's Algorithm :- M.E → ✓, P → ✗, B.W → ✓

→ process takes turn to enter into critical solution.

→ solution will work for 2 - process.




turn

0 → $P_0$ (entry)

1 → $P_1$ (entry)

| $P_0 \rightarrow 0$ | $P_1 \rightarrow 1$ |
|---|---|
| while (True) | while (True) |
| { | { |
| Non c.s ; | Non c.s ; |
| while (turn != 0) ; | while (turn != 1) ; |
| c.s | c.s |
| turn ⇒ 1 ; | turn ⇒ 0 ; |
| } | } |

Analysis (for M.E, P, B.W) —

turn ⇒ 0

c.s

$P_0$ → M.E ✗

for finite process → B.W ✓

turn → ∅, 1, 0

$P_0$, $P_1$, $P_1$

here $P_0$ not interested to go c.s but for turn=0 $P_1$ can't go c.s. (no stored $P_1$) → Progress ✗

- 2nd defination of progress :-

Which process will go next into c.s is decided by only those process, who want to go into cs. I this decision the process in the remainder section and the process which is not interested to go into the c.s should not take participation.

Saathi

Date

- Peterson's Algorithm :- (ME → ✓, P → ✓, B.W → ✓)

(2 process solution)

```
#define N 2
#define TRUE 1
#define FALSE 0

int turn;
int interested[N];
void enter_Region(int process)
{
  1. int other;
  2. other = [1-Process];
  3. Interested[process] = TRUE;
  4. turn = process;
  5. while (turn == process && interested[other] == TRUE);
       CS;
}

void leave_Region(int process)
{
  interested[process] = false;
}
```

Initially –

Interested[0] = false.
Interested[1] = false.

P0 → other 1
P1 → other 0

- Analysis :-

(next page)

Date ___/___/_____

| $P_0$ | $P_1$ |
|---|---|
| 1. | 1. → |
| 2. other → 1-0 = 1 | 2. other = 0 |
| 3. Interested[0] = True. | 3. Interested[1] = True. |
| 4. turn = 0 | 4. turn = 1 |
| 5. → CS [$P_0$] | 5. Loop CS [$P_0$] |

So, M.E = ✓.
progress = ✓
Bounded waiting = ✓

* peterson practise Question :-

Code –

```
void enter_Region()
{
1. int other ;
2. other = 1- process ;
3. Interested[process] = True;
4. turn = process ;
5. While (turn == process && Interested[other] == True);
6. c.s ;
}
```

| $P_0$ | $P_1$ ✓ |
|---|---|
| 1. → | 1. → |
| 2. → other = 1 | 2. other = 0 |
| 3. → Interested[0] = True | 3. Interested[1] = True. |
| 4. → turn = 0; | 4. turn = 1; |
| | 5. ↻ |
| | C.S [$P_1$] |

Q. If both the process $P_0$ and $P_1$ are trying to enter in cs at same time, then which process will go into cs first.

a. The process which executes statement 2 first.
b. " " " " " 3 " .
c. " " " " " 4 " .
d. We cannot say.

→ Ans → (c).

Date ___/___/_____

- TSL Introduction: M.E → ✓, P → ✓, B.W → X

→ TSL ~~flag~~ Register flag :- Copies the current value of flag into register and Store the value of 1 into the flag in a single atomic cycle, without any preemption.

→ Disabling the interrupt is at H/W level that is why it is called H/W approach.

→ Entry section –

flag

0 — CS is free

1 — CS is busy

*1) TSL $R_i$, m[flag],
2) cmp $R_j$, #0
3) JNZ to step (1).
4) C.S
5) STORE m[flag], #0

Analysis –

flag = 0 & 1

RQ = [$P_0$ | $P_2$]

$P_1 \rightarrow$ I  $R_1$ [0]

$P_1 \rightarrow$ II

$P_1 \rightarrow$ III

$P_1 \rightarrow$ IV → pre-empt  CS [$P_1$]  M.E → ✓

$P_2 \rightarrow$ I  $R_2$ [1]

$P_2 \rightarrow$ II, III

[$P_1$ | $P_2$]  flag = 0 & 1

CS [$P_1$]

$P_1 \rightarrow$ I  $R_1$ [0] → preemptive

$P_2 \rightarrow$ I  $R_2$ [1]  progress → ✓

$P_2 \rightarrow$ II, III

Where no. of process infinite so, B.W → X.
(Bound, waiting)

Date

*. Some Important points :- (TSL)

→ In Lock variables mutual exclusion is not satisfied because loading and storing were not done in same step.

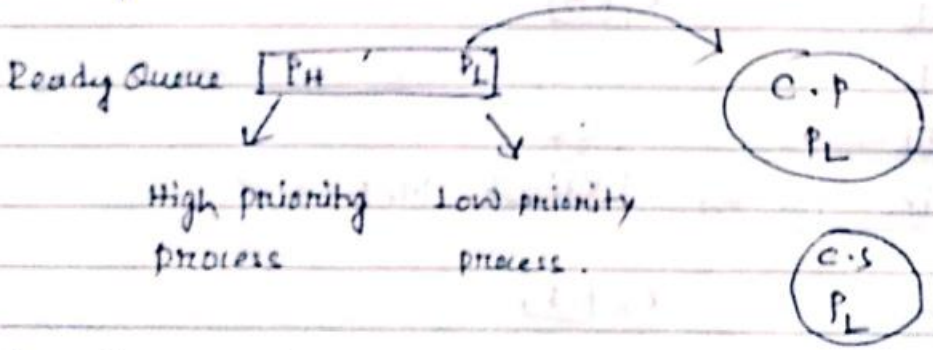
→ progress is satisfied in TSL.

→ Bounded waiting is not satisfied Because no. of processes is not countable.

*. Summary :-

| Solution | Mutual exclusion | progress | Bounded waiting |
|---|---|---|---|
| Lock variable | X | ✓ | X |
| Strict Alternation | ✓ | X | ✓ |
| Peterson's Solution | ✓ | ✓ | ✓ |
| TSL | ✓ | ✓ | X |

• Priority Inversion problem —




→ $P_L$ is currently executing CS code. Now suddenly $P_H$ coms, then we are going to pre-empt $P_L$. ~~$P_H$ is~~ even though $P_H$ is scheduled by CPU but it can not enter in C.S because it is locked by $P_L$.

↓

Problem is called live-Lock

↓

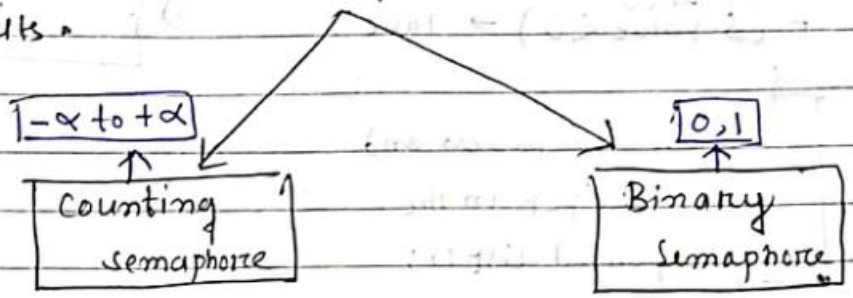
Solution → Priority Inversion / Inheritance.

Date ___ / ___ / _____

→ This allows that $P_1$ can continue its execution in c.s and then Leave.

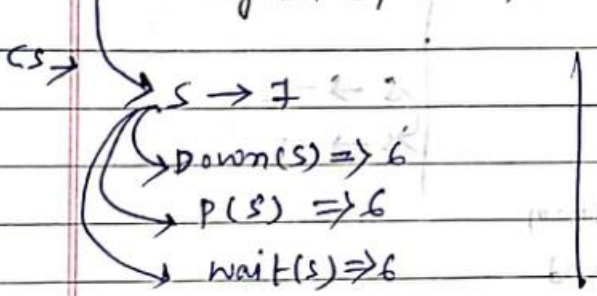
## * Semaphore :

→ is an Integer variable which is used by various process in a mutual exclusive manner to achive Synchronigation. The ~~proper~~ Improper usage of semaphore will be given Improper results.




Two different operations performed on semaphore variable are –

1) Down Operation on P() or wait().
2) signal Operation on V() or release() or Up().




bs ⇒ 1
down(bs) ⇒ 0

bs ⇒ 0
up(bs) ⇒ 1

Date ___/___/_____

Saathi

## Counting Semaphore –

→ Down operation –

```
Down operation (semaphore S)
{
  S.value = S.value - 1;
  If (S.value < 0) → True
  {
    Block the process and
    Place the PCB in the
    suspended list ();
  }
}
```

S → 2
1) Down (1) → 1
2) Down (1) → 0
3) Down (1) → -1

→ up operation – (always successful)

```
up operation (semaphore S)
{
  S.value = S.value + 1;
  If (S.value ≤ 0)
  {
    Select the process from
    suspanded list and
    wakeup ();
  }
}
```

S → 2
S → 3

S → -2
S → -1

→ After performing the down operation, if the process is getting suspended, then it is called unsuccessfull down operation.

→ After performing the down operation, if the process is not getting suspended, then it is called successfull down operation.

Date ___/___/_____

→ If it is successfull down Operation, then only the process will continue the execution. The down operation on the counting semaphore is successfull only if the semaphore value is $\geq 1$

→ There is no unsuccessfull up operation, the up Operation is always successfull because it does not suspend any process.

→ If $S = +6$ it means we can perform 6 successfull down Operation.

→ If $S = -6$ it means that there are already 6 suspended processes.

*. Question -

Consider a system where a counting semaphore value is initialiged to +17.

The various semaphore operations like-

23P, 18V, 16P, 14V, 1P are performed.

what is the final value of semaphore.

$\Rightarrow 17 - 23 + 18 - 16 + 14 - 1$

$\Rightarrow 9 - 6 + 2 + 13$

$\Rightarrow$ (F)

Binary Semaphore: 0 ↔ 1

→ Down Operation

```
Down (Semaphore S)
{
  if (S.value == 1)
    S.value = 0;
  else
  { Block the process and
    place the PCB in the
    Suspended list();
  }
}
```

→ Up Operation

```
up(semaphore S)
{
  if (suspended list() is empty)
    S.value = 1;
  else
  {
    Select a process from
    suspended list and WeakUp()
  }
}
```

→ After performing down Operation if the process is getting suspended then it is called unsuccessfull down Operation.

→ After performing down Operation, if the process not getting ~~were~~ suspended then it is called successfull down Operation.

→ The down Operation on the Binary semaphore is successfull only if the semaphore value is 1.

→ If it is successfull down operation, then only the process will continue the Execution.

→ There is no unsuccessfull up operation. The up Operation will be always successfull. The process performing the up Operation will definately continue the execution.

Question : (I) (II) (III)

Each process $P_i = 1$ to 9 executes the following code -

```
repeat
    P(mutex);
      C.S
    V(mutex);
forever
```

The process $P_{10}$ execute the following code.

```
repeat
    V(mutex);
      C.S
    V(mutex);
forever
```

⇒ The initial value of Binary semaphore mutex = 1.

⇒ What is the maximum no. of process may be present inside the cs at any point of time ?

mutex ⇒ 1

mutex → 1 0 1 0 1 0 1 0

| $P_1$ | $P_{10}$ | $P_2$ | $P_3$ | $P_4$ | $P_5 \cdots P_9$ |
|---|---|---|---|---|---|

C.S ↓

→ Ans : (10) process. ✓

P(mutex); then find ?

→

mutex = 1 0 1 0

| $P_1$ | $P_{10}$ | $P_2$ |
|---|---|---|

C.S ↓ — 3(P).

→ Ans : (3) process. ✓

P(mutex); then find ?

mutex = 1 0

| $P_1$ |
|---|

C.S

→ Ans : (1) process.

- GATE-2003 Question :-

Consider the two concurrent process 'P' and 'Q' executing their ~~ex~~ respective codes.

| process 'P' code | process 'Q' code |
|---|---|
| While (TRUE) | While (TRUE) |
| { | { |
| ① W : P(T) | ③ y : P(s) |
| printf('0'); | printf('1'); |
| printf('0'); | printf('1'); |
| ② X : V(s) | ④ Z : V(T) |
| } | } |

What should be the semaphore operations on W, X, Y, Z respectively. And what should be the initial value of Binary semaphore 's' and 'T' inorder to get the output as 0011 0011 00 ...

(a) W = P(T), X = V(T), Y = P(s), Z = V(s), S = T = 1 ;

(b) W = P(T), X = V(T), Y = P(s), Z = V(s), S = 1, T = 0

(c) W = P(T), X = V(s), Y = P(s), Z = V(T), S = 1, T = 1 ;

(d) W = P(T), X = V(s), Y = P(s), Z = V(T), S = 0, T = 1 ;

→ PQPQ ... ..

- Question :- process code same.

Which of the following will ensure that the output string never contains a substring of the form $01^n0$ or $10^n1$ where n is odd.

(a) W = P(s), X = V(s), Y = P(T), Z = V(T), S = T = 1 ;

(b) W = P(s), X = V(T), y = P(T), Z = V(s), S = T = 1 ;

(c) W = P(s), X = V(s), y = P(s), Z = V(s), S = 1 ;

(d) W = P(s), X = V(T), y = P(s), Z = V(T), S = T = 1 ;

S →

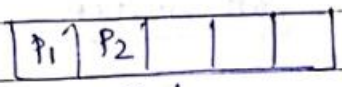
→ 00 11 0011 ... ..

• Producer & consumer problem solve with the help of semaphore.

semaphore S = 1

semaphore E = n  here n = is of size of buffer

Semaphore F = 0




C.S

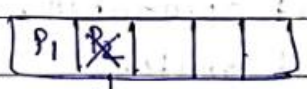
```
* void producer()
  {
     While(T)
     {
        produce()
        wait/down(E)
        wait(s)/down(s)
        append()
        signal(s)/up(s)
        signal(F)/up(F)
     }
  }
```

Let S = 1 0 1 0 1

E = ~~5~~ ~~4~~ 3

F = 0 1 2

```
* void consumer()
  {
     While(T)
     {
        wait(F)/down(F)
        wait(s)
        take()
        signal(s)/up(s)
        signal(E)
        use()
     }
  }
```




S = 1 0 1

E = ~~3~~ 3 4

F = 0 1 1

- READ-WRITE PROBLEM:

(Solve by Synaphorce)

(there using used 3 semaphore).

for writer

| | |
|---|---|
| I. wait(wrt) (1) | wrt = 1 0 |
| II. write operation | |
| III. signal(wrt)/up(wrt). | |

for read

| | |
|---|---|
| I. wait (mutex) (1) | I. mutex = 1 0 1 0 1. |
| II. readcount ++ | II. readcount = 1 0 |
| III. If (readcount == 1) wait (wrt) | III. wrt = 0 1 (no writer can enter in critical Section :). |
| IV. signal (mutex) | |
| V. read operation. | |
| VI. wait (mutex) | |
| VII. readcount -- | |
| VIII. If (readcount == 0) signal (wrt) | |
| IX. signal (mutex). | |

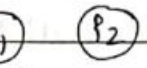
*. At a time more than one read operation done in critical section.

*. At a time only one write operation done in C.S.

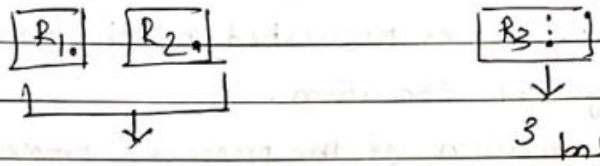
# DEADLOCK

- Defination:- If two or more processes are waiting for some event to occure, which never happens, the processes are said to be involved in deadlock.
- Resource Allocation Graph (RAG):

→ Basics:- process will be represented with circle.



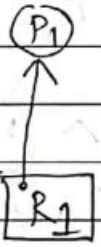

Resources will be represented with rectangle.



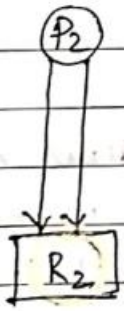

*. Requesting edge


process $P_1$ is requesting 1 Instance of $R_1$

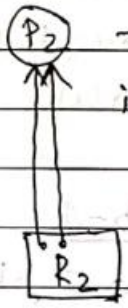
*. Allocation edge.




one Instance of '$R_1$' is allocated to '$P_1$'.




process $P_2$ is requesting for 2 instences of '$R_2$'.




Two Instance of '$R_2$' is allocated to '$P_2$'.

→ Resource request and Resource allocation will be ~~requested in the~~ represented in the resource allocation graph (RAG).

$(R.A.G) = G_r(V, E)$

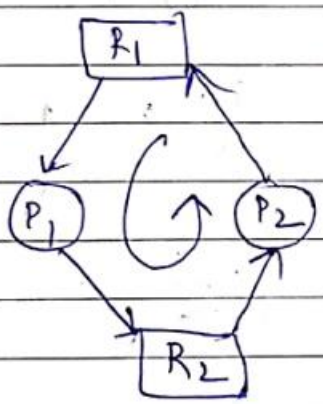
V → processes & Resources. (vertex)

E → Requesting and Allocation edge.

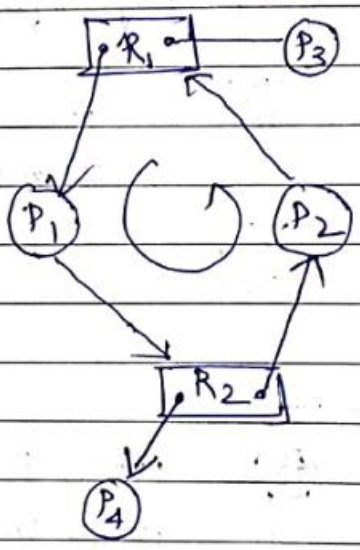
• Resource Request and Release life cycle.

→

1) The process will request for the resource. $(P_1) \rightarrow [R_1]$
2) O.S validates the request of the process.
3) If the request made by process is valid, the O.S check for availability of the resource.
4) If the resource is freely available, it will be allocated to the process, otherwise process has to wait.
5) If all the resources requested by the process are allocated then it will go into execution.
6) Once the execution of the process completed, then the process will release all the resources.




Deadlock is involved in (RAG)




No Deadlock even if cycle is present.

• Deadlock Q-1

Consider a system with n processes and 6 tape drives. If each process requires 2 tape drives to complete their execution, then what is the maximum value of n which ensures deadlock free operation?

→ R ⟺ 6 (tape)

P ⇒ need {2}

↓

* (n-1)

↓

5

| $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | $P_6$ |
|---|---|---|---|---|---|
| $x_2$ | 1 | 1 | 1 | 1 | 1 |

$n = 1$

↳ execution complete.

first takes, $n = 6$ (Deadlock)

if $n = 5$ (P) then it will be deadlock free operation.

• Deadlock Q-2:

Consider a system with 3 processes, where each process requires 2 tape drives to complete there execution, then what is the minimum number of resources required to ensure deadlock free execution. (1)

→ here, process = 3,

| | $P_1$ | $P_2$ | $P_2$ |
|---|---|---|---|
| P → | | | |
| R → | 1 | 1 | 1 |

$\therefore R = 3+1 = 4$

minimum number of resources required to ensure deadlock free execution are 4(R).

• Deadlock Q-3:

consider a system with n processes and 6 tape drives. If each process require 3 ~~process~~ tape drives to complete their execution. What is the maximum value of n which ensures deadlock free execution.

→

| $P_1$ | $P_2$ | $P_3$ |
|---|---|---|
| 2 | 2 | 2 |
| 3 | 3 | X |

$R \rightarrow 6$

$n_1 \rightarrow ?$ need(3) so, n will be 2(P).

• Deadlock Question-1 :

Consider a system with 3 process $P_1, P_2, P_3$. The peak demand of every process is 5, 9, 13 respectively for the Resource (R). Then for what minimum no of resources, the execution is deadlock free. (25)

→ $P_1 \to 5$
$P_2 \to 9$
$P_3 \to 13$
} (n-1)

| $P_1$ | $P_2$ | $P_3$ |
|---|---|---|
| 4 | 8 | 12 |

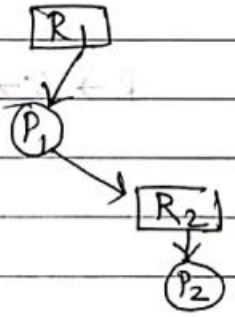
$R \to 4+8+12$
$= (24)$
$+1$
$R = (25)$ ✓

• Deadlock characteristics Introduction –
(Behaviour of Deadlock)

(I) Mutual exclusion.
(II) Hold and wait.
(III) NO – Preemption.
(IV) circular wait.

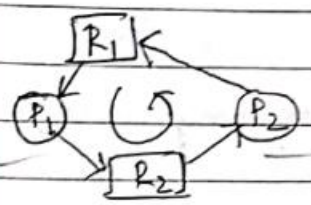
1. Mutual exclusion :- The resource has to be allocated to only one process or it is freely available. There should be a one to one relation ship between the resource and process.

2. Hold and wait :- The process is holding some resources and waiting on some other resource simultenously.



3. No- Preemption :- The resource has to be voluntarily released by the process after the completion of execution. It is not allowed to forcefully pre-empt the resources from process.

4. Circular Wait :-
The process are circularly waiting on each other for the resources.




$R_1 \rightarrow P_1 \rightarrow R_2 \rightarrow P_2$

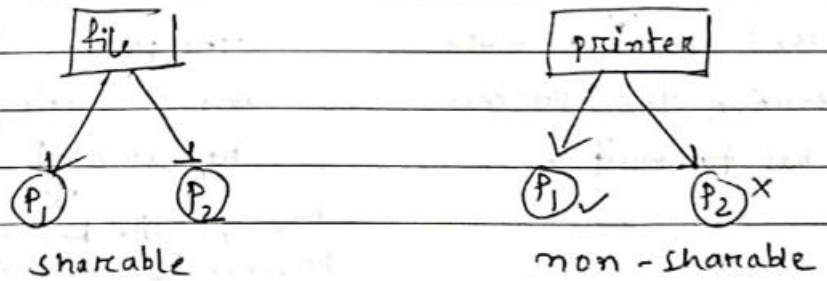
All 4 MUST OCCUR simultaneously. Then Definately Deadlock is there.

All the above 4 conditions are not truely independent because circular wait includes hold & wait.

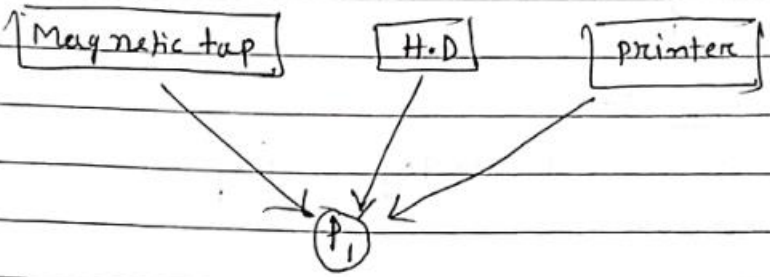
- Deadlock prevention schemes :

(1) Deadlock prevention (M.E): It is not always possible to dissatisfy mutual exclusion because of shareable and non-shareable resource.

→ If there are shareable resources then mutual exclusion can be dissatisfy and if they are non shareable resource then Mutual exclusion cannot be dissatisfied.



(2) Hold and wait :- (i) Allocate all the ~~process~~ Resources required by the process before start of execution.




→ Low device utiligation.

(ii) The process should release all existing resource before making new request.

$P_1 \Rightarrow 100\ R$

| | |
|---|---|
| 1) $P_1 \rightarrow 100R$ | $P_1 \leftarrow 85\ R$ |
| 2) $P_1 \rightarrow 100\ R$ | $P_1 \leftarrow 40\ R$ |
| 3) $P_1 \rightarrow 100R$ | $P_1 \leftarrow 100R$ |

→ (starvation).

(3) No preemption : The process '$P_1$' is requesting for resource '$R_1$'. $(P_1) \rightarrow [R_1]$

If resource $R_1$ is free then it will be allocated to process '$P_1$'.

$[R_1] \rightarrow (P_1)$

If the process $P_1$ is in the execution then process $P_1$ has to wait.

The Resource $R_1$ is not free, and it is allocated to process $P_2$.

$(R_1) \rightarrow (P_2)$

The process $P_2$ is not executing and is waiting for the other resource $R_2$.

Prempt the resource $R_1$ from process $P_2$ and Asign it to process $P_1$.

(4) Circular Wait: The resources will be assigned with a numerical number. The process can request for the resource only in increasing order of enumeration.

$R_1, R_2, R_3, \ldots R_{12}$

$P_1 \rightarrow R_5$ ✓

$P_1 \rightarrow R_7$ ✓

$P_1 \rightarrow R_8$ ✓

$P_1 \rightarrow R_6$ ✗

$P_1 \rightarrow R_9$ ✓

## Bankers Algorithm:

- Deadlock Avoidance (BANKERS ALGORITHM)

Total Resources ⇒

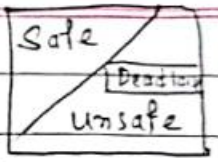
| A | B | C |
|---|---|---|
| 10 | 5 | 7 |

= (1 instance)

| | Max Need | | | current Allocation | | | Remaining Need | | | Current Available | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | A | B | C | A | B | C | A | B | C |
| (III) $P_0$ | 7 | 5 | 3 | 0 | 1 | 0 | 7 | 4 | 3 | (3) | (3) | (2) Free |
| X (I) $P_1$ | 3 | 2 | 2 | (2) | 0 | 0 | 1 | 2 | 2 | 5 | 3 | 2 |
| (IV) $P_2$ (II) | 9 | 0 | 2 | 3 | 0 | 2 | 6 | 0 | 0 | 7 | 4 | 3 |
| X $P_3$ | 2 | 2 | 2 | 2 | 1 | 1 | 0 | 1 | 1 | 7 | 5 | 3 |
| X (V) $P_4$ | 4 | 3 | 3 | 0 | 0 | 2 | 4 | 3 | 1 | 10 | 5 | 5 |
| | | | | 7 | 2 | 5 | | | | 10 | 5 | 7 |

$P_1, P_3, P_0, P_2, P_4$

safe sequence




- Deadlock Avoidance Conclusion:

*. Remaining need = Max - Current Allocation.

→ If we can satisfy the remaining need of the processes with the current available resources, then system is said to be in the safe state other wise, the system is said to be in the unsafe state.

→ If the system is in the unsafe state, then it is possible for deadlock.

→ The order in which we satisfy the remaining need of all the process is called safe sequence.

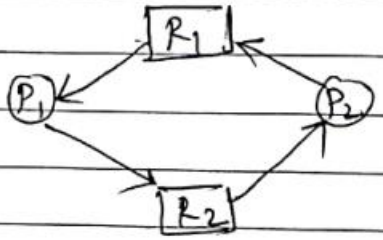
→ The safe sequence may not be unique. We can have multiple safe sequence.

→ Deadlock Avoidance is more restrictive than deadlock ~~pre~~ perention.

→ The unsafe state purely depends on the behaivour of the process.

- Deadlock Detection:

(i) All the resources are of single Instance type.




If all the resources are of single Instance type, then cycle in the RAG is nessery and sufficient condition for occuring of

deadlock. If all the resources are of more than single Instance type, then cycle in the RAG is just a necessary condition but not sufficient condition for occuring of deadlock.

(ii) The Resources are of multiple Instance type.

total resources

| A | B | C |
|---|---|---|
| 7 | 2 | 6 |

| | Current Allocation | | | Remaining need | | | current Available | | |
|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | A | B | C | A | B | C |
| X (P0) → | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| X3 (P1) → | 2 | 0 | 0 | 2 | 0 | 2 | 0 | 1 | 0 ✓ |
| X2 (P2) → | 3 | 0 | 3 | 0 | 0 | 0 | 3 | 1 | 3 ✓ |
| X4 (P3) → | 2 | 1 | 1 | 1 | 0 | 0 | 5 | 1 | 3 ✓ |
| X5 (P4) → | 0 | 0 | 2 | 0 | 0 | 2 | 7 | 2 | 4 ✓ |
| | 7 | 2 | 6 | | | | 7 | 2 | 6 |

→ $P_0, P_2, P_1, P_3, P_4$ safe sequence. ✓

* What happens if $P_2$ is demanding for 001 resources?

total resources

| A | B | C |
|---|---|---|
| 7 | 2 | 6 |

| | current Allocation | | | Remaining need | | | current Available | | |
|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | A | B | C | A | B | C |
| X (P0) | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| P1 | 2 | 0 | 0 | 2 | 0 | 0 | 0 | 1 | 0 |
| P2 | 3 | 0 | 3 | 0 | 0 | 1 | | | |
| P3 | 2 | 1 | 1 | 1 | 0 | 0 | | | |
| P4 | 0 | 0 | 2 | 0 | 0 | 2 | | | |
| | 7 | 2 | 6 | | | | | | |

→ deadlock

(P0) ↗ (P1, P2, P3, P4) → deadlock.

## • Deadlock Recovery :-

i) Killing the process :

(a) Killing all the processes involved in the deadlock.

(b) Kill one by one till the deadlock is cracked / resolved.

- i) low priority process.
- ii) % of process completion $P_1 \rightarrow 95\%$ $P_2 \rightarrow 5\%$ ✗
- iii) no of resources the process is holding.
  - $P_1 \leftarrow 20R$ $P_2 \leftarrow 2R$
- iv) no of resources the process is requisting.
  - $P_1 \rightarrow 1R$ $P_2 \rightarrow 20R$ ✓

ii) Resource pre-emption : The resources from the process which are involved in the deadlock will be preempted and the pre-empted resources will be allocated to other process so that there is a possibility of ~~resource~~ recovering the system from deadlock.

→ If resource pre-emption is followed, there is a possibility of starvation.

1) selection of victim
2) Roll back.
3) starvation.

Saathi

Date __/__/____

# CPU SCHEDULING - (numerical)

RBR • Basic Introduction:

Where? → In the ready state to running.

Who? → Short term Scheduler / CPU scheduler.

When? →

1) Run → termination, Run → wait, Run → Ready.
2) Wait → Ready.
3) New → Ready.

• Functionality: To allocate the CPU to the process.

• GOAL: 1) Increase the CPU Utiligation and throughput of the System.
2) Minimize the Average waiting time, averag turn around time of the process.

RBR • Some Basic terms:

① Arrival time (A.T) → The time when the process is arriveal into ready state is called as Arrival time.

② Burst time (B.T) → The time is required By the process for its execution is called as Burst time of the process.

③ Completion time (C.T) → The time when the process is completed or completes its execution is called completion time.

④ Turn around time (T.A.T) → The time difference between completion time and arrival time is called turn around time of process.

$$T.A.T = C.T - A.T$$

⑤ waiting time (W.T) → The difference between T.A.T and B.T

$$W.T = T.A.T - B.T$$

⑥ Response time (R.T) → The difference between 1st response and arrivel time is called as Respons time.

Date __/__/____

RBR

- First Come First Served (FCFS) Scheduling:

selection criteria: Arrival time.

Mode: Non-pre-emptive.

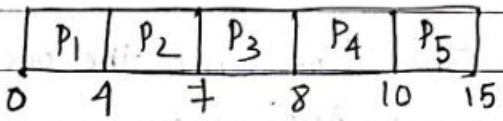
$T.A.T = C.T - A.T$
$= W.T + B.T$

- Question-1:

| process.no | A.T | B.T | C.T | T.A.T | W.T |
|---|---|---|---|---|---|
| 1 | 0 | 4 | 4 | 4 | 0 |
| 2 | 1 | 3 | 7 | 6 | 3 |
| 3 | 2 | 1 | 8 | 6 | 5 |
| 4 | 3 | 2 | 10 | 7 | 5 |
| 5 | 4 | 5 | 15 | 11 | 6 |

$W.T = T.A.T - B$

Gnantt chart




Average waiting time $= \frac{0+3+5+5+6}{5}$

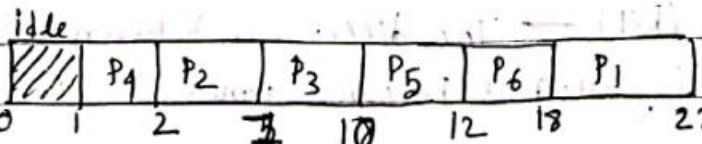
$= \frac{19}{5}$

$= 3.8$ ms. ✓

- Question-2:

| P.NO | A.T | B.T | C.T | T.A.T | W.T |
|---|---|---|---|---|---|
| 1 | 6 | 4 | 22 | 16 | 12 |
| 2 | 2 ✓ | 5 | 7 | 5 | 0 |
| 3 | 3 ✓ | 3 | 10 | 7 | 4 |
| 4 | (1) 1 ✓ | 1 | 2 | 1 | 0 |
| 5 | 4 ✓ | 2 | 12 | 8 | 6 |
| 6 | 5 | 6 | 18 | 13 | 7 |

Gnantt chart




average waiting time $= \frac{12+4+6+7}{6}$

$= \frac{29}{6} = 4.8$

Saathi

Date __/__/____

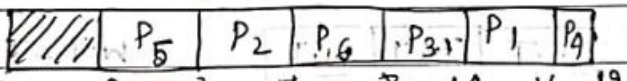
(*) If the Arrival time of the process are same / matching then schedule the process which has lowest process id.

- Question no-3:

| | P.NO | A.T | B.T | C.T | T.A.T | W.T |
|---|---|---|---|---|---|---|
| | 1 | 8 ✓ | 2 | 16 | 8 | 6 |
| * | 2 | 3 ✓ | 4 | 7 | 4 | 0 |
| | 3 | 7 ✓ | 6 | 14 | 7 | 1 |
| | 4 | 10 ✓ | 3 | 19 | 9 | 6 |
| | 5 | 2 ✓ | 1 | 3 | 1 | 0 |
| * | 6 | 3 ✓ | 1 | 8 | 5 | 4 |

T.A.T = C.T − A.T
W.T = T.A.T − B.T

Gnantt chart




So, average waiting time = $\frac{6+1+6+4}{6}$

$= \frac{17}{6}$

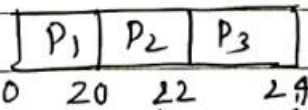
$= 2.83$ ms.

- Question no-4:

(a)

| P.NO | A.T | B.T | W.T |
|---|---|---|---|
| 1 | 0 | 20 | 0 |
| 2 | 1 | 2 | 19 |
| 3 | 2 | 2 | 20 |

Gnantt chart




average waiting time,

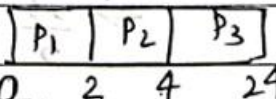
$\frac{0+19+20}{3} \Rightarrow (13)$ ✓

CONVOY Effect

(b)

| P.NO | A.T | B.T | W.T |
|---|---|---|---|
| 1 | 0 | 2 | 0 |
| 2 | 1 | 2 | 1 |
| 3 | 2 | 20 | 2 |

Gnantt chart




average waiting time,

$\frac{0+1+2}{3} = (1)$ ✓

Date __/__/____

RBR

(*) In FCFS, if the 1st process is having large burst time then it will be have drastic effect on average waiting time of all the process. This effect is called CONVOY Effect.

RBR

Shortest Job first (SJF) ~~Sho~~ scheduling:

selection Criteria :- Burst time

Mod :-

↳ Non-pre-emphive.

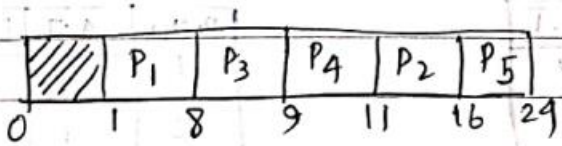
*↳ ✓pre-emptive. * (imporatant for gate)

• Question-1:- SJF Non pre-emphive.

| P.No | A.T | B.T | C.T | T.A.T | W.T |
|---|---|---|---|---|---|
| ✓1 | 1 | 7 | 3 | 7 | 0 |
| ✓2 | 2 | 5 | 16 | 14 | 9 |
| ✓3 | 3 | 1 | 9 | 6 | 5 |
| ✓4 | 4 | 2 | 11 | 7 | 5 |
| ✓5 | 5 | 8 | 24 | 19 | 11 |

W.T = T.A.T − B.T

Gruantt Chart




average T.A.T. (Turn around time) :-

$\Rightarrow \frac{7+14+6+7+19}{5}$

$\Rightarrow \frac{53}{5}$.

average W.T (waiting time),

$= \frac{0+9+5+5+11}{5} \Rightarrow \frac{30}{5} \Rightarrow 6$

Saathi

Date ___/___/___

• Question-2 :- SJF Non pre-emptive.

| P.No | A.T | B.T | C.T | T.A.T | W.T |
|---|---|---|---|---|---|
| 1 | 3 | 4 | 13 | 10 | 6 |
| 2 | 4 | 2 | 9 | 5 | 3 |
| 3 | 5 | 1 | 7 | 2 | 1 |
| 4 | 2 | 6 | 19 | 17 | 11 |
| 5 | 1 | 7 | 26 | 25 | 17 |
| 6 | 2 | 4 | 6 | 4 | 0 |

Gcuantt chart

| idle | $P_5$ | $P_6$ | $P_6$ | $P_6$ | $P_6$ | $P_3$ | $P_2$ | $P_1$ | $P_4$ | $P_5$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 9 | 13 | 19 | 26 |

Average T.A.T = $\frac{10+5+2+17+25+4}{6}$

$= \frac{63}{6} \Rightarrow 10.5$.

Average W.T = $\frac{6+3+1+11+17+0}{6}$

$= \frac{38}{6}$

$= 6.33$.

PBR

• Question-1 :- SJF preemptive.

*. SJF is also called SRTF.

SRTF => Shortest Remaining time first.

| P.NO | A.T | B.T | C.T | T.A.T | W.T |
|---|---|---|---|---|---|
| 1 | 0 | 7 | 19 | 19 | 12 |
| 2 | 1 | 5 | 13 | 12 | 7 |
| 3 | 2 | 3 | 6 | 4 | 1 |
| 4 | 3 | 1 | 4 | 1 | 0 |
| 5 | 4 | 2 | 9 | 5 | 3 |
| 6 | 5 | 1 | 7 | 2 | 1 |

avg T.A.T = $\frac{19+12+4+1+5+2}{6}$

$= \frac{43}{6} \Rightarrow 7.16$

avg W.T = $\frac{12+7+1+0+3+1}{6}$

$= 4$

Gcuantt chart :-

| $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ | $P_3$ | $P_6$ | $P_5$ | $P_2$ | $P_1$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 9 | 13 | 19 |

Page No. 57

Saathi

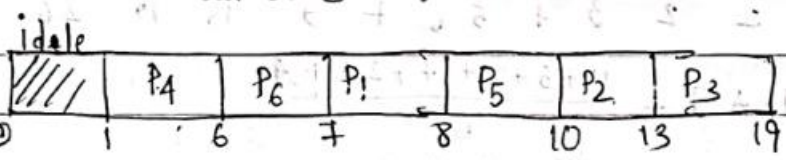
Date __/__/____

* Note:- If the Burst time of the process are same/matching then schedule the process which has lowest Arrival time.

• Question: SJF → Non-pre-emptive.

| P.NO | A.T | B.T | C.T | T.A.T | W.T |
|---|---|---|---|---|---|
| 1 | 6 | 1 | 8 | 2 | 1 |
| 2 | 3 | 3 | 13 | 10 | 7 |
| 3 | 4 | 6 | 19 | 15 | 9 |
| 4 | 1 | 5 | 6 | 5 | 0 |
| 5 | 2 | 2 | 10 | 8 | 6 |
| 6 | 5 | 1 | 7 | 2 | 1 |

Gruantt Chart




$$\text{avg T.A.T} = \frac{2+10+15+5+8+2}{6} \Rightarrow \frac{42}{6} \Rightarrow 7$$

$$\text{avg W.T} = \frac{1+7+9+6+1}{6} \Rightarrow \frac{24}{6} \Rightarrow 4$$

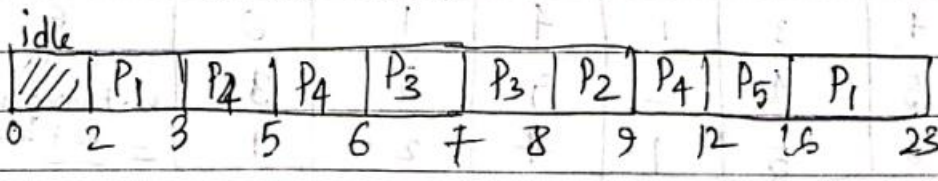
* • Question :- SJF pre-emptive. W.T = T.AT - B.T

| p.NO | A.T | B.T | C.T | T.A.T | W.T |
|---|---|---|---|---|---|
| 1 | 2 | 8 | 23 | 21 | 13 |
| 2 | 7 | 1 | 9 | 2 | 1 |
| 3 | 6 | 2 | 8 | 2 | 0 |
| 4 | 3 | 6 | 12 | 9 | 3 |
| 5 | 5 | 4 | 16 | 11 | 7 |

$$\text{Avg T.A.T} = \frac{21+2+2+9+11}{5} = 9$$

$$\text{Avg W.T} = \frac{13+1+0+3+7}{5} = \frac{24}{5} = 4.8$$

Guantt chart



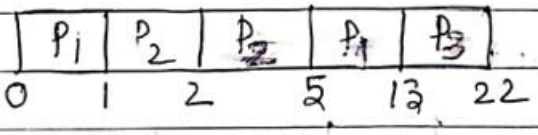
Date __/__/____

PBR

- GATE 2011 Question :- SJF pre-emptive.

| P.NO | A.T | B.T | C.T | T.A.T | W.T |
|---|---|---|---|---|---|
| 1 | 0 | 9 | 13 | 13 | 4 |
| 2 | 1 | 4 | 5 | 4 | 0 |
| 3 | 2 | 9 | 22 | 20 | 11 |

Gcuantt Chart



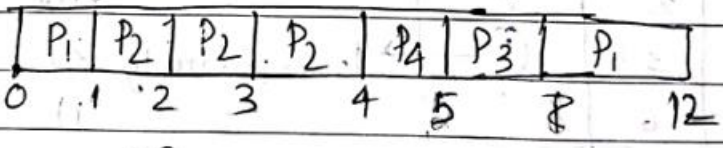

avg T.A.T = $\frac{13+4+20}{3} \Rightarrow \frac{37}{3} \Rightarrow 12.33$.

avg W.T = $\frac{15}{3} = 5$.

- GATE 2004 :→ SJF pre-emptive.

| P.NO | A.T | B.T | C.T | T.A.T | W.T |
|---|---|---|---|---|---|
| 1 | 0 | 5 1 | 12 | 12 | 7 |
| 2 | 1 | 3 2 | 4 | 3 | 0 |
| 3 | 2 | 3 | 8 | 6 | 3 |
| 4 | 3 | 1 | 5 | 2 | 1 |

Gcuantt chart




Avg T.A.T = $\frac{23}{4} \Rightarrow 5.75$

Avg W.T = $\frac{11}{4} = 2.75$

Throughput = No. of Process / C.T

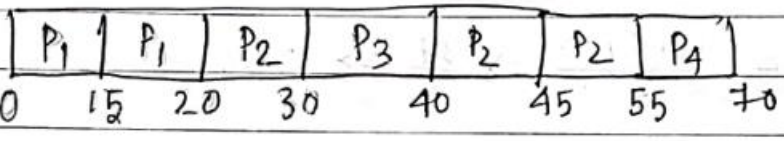
Saathi

Date __/__/____

- GATE 2006 Question :- SJF pre-emptive.
  What is the waiting time of process $P_2$?

| P.No | A.T | B.T | C.T | T.A.T | W.T |
|---|---|---|---|---|---|
| 1 | 0 | 20 | 20 | 20 | 0 |
| 2 | 15 | 25 | 55 | 40 | (15) → Waiting time of ($P_2$) |
| 3 | 30 | 10 | 40 | 10 | 0 |
| 4 | 45 | 15 | 70 | 25 | 10 |

Guantt chart




So, W.T of $P_2$ process is 15.

PBR

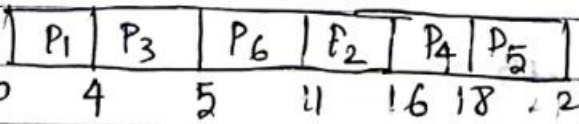
- priority Based Scheduling :
  Selection criteria :- priority. (high-priority)
  Mode :- Non-pre-emptive.

Q.1 →

| P.No | A.T | B.T | priority |
|---|---|---|---|
| 1 | 0 | 4 | 4 |
| 2 | 1 | 5 | 5 |
| 3 | 2 | 1 | (1) → high priority |
| 4 | 3 | 2 | 2 |
| 5 | 4 | 3 | (1) → low-priority |
| 6 | 5 | 6 | 6 |

| C.T | T.A.T | W.T |
|---|---|---|
| 4 | 4 | 0 |
| 16 | 15 | 10 |
| 5 | 3 | 2 |
| 18 | 15 | 13 |
| 21 | 17 | 14 |
| 11 | 6 | 0 |

Guantt chart




Avg Waiting time (W.T) = $\frac{0+10+2+13+14+0}{6}$

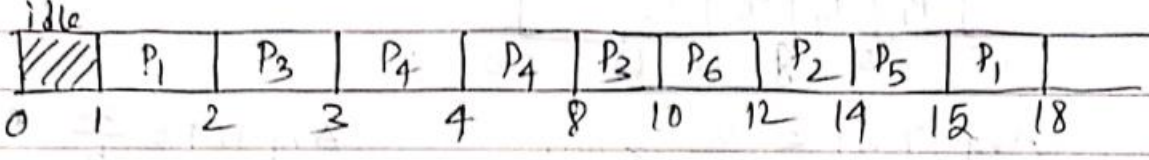
$= \frac{39}{6} \Rightarrow 6.5$

Date

- priority Based : pre-emptive. (Q1)

| priority | P.NO | A.T | B.T | C.T | T.A.T | W.T |
|---|---|---|---|---|---|---|
| 4 ← low | 1 | 1 | 4 | 13 | 17 | 13 |
| 5 | 2 | 2 | 2 | 14 | 12 | 10 |
| 7 | 3 | 2 | 3 | 10 | 8 | 5 |
| 8 ← high | 4 | 3 | 5 | 8 | 5 | 0 |
| 5 | 5 | 3 | 1 | 15 | 12 | 11 |
| 6 | 6 | 4 | 2 | 12 | 8 | 6 |

Gruantt Chart




Avg W.T,

$$= \frac{13+10+5+0+11+6}{6}$$

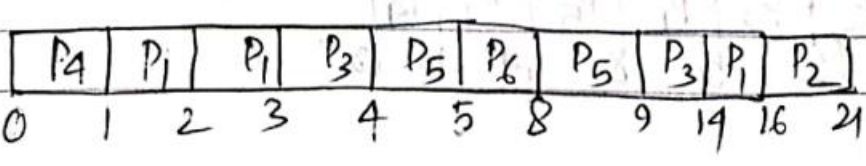
$$= \frac{45}{6} \Rightarrow 7.5$$

- priority Based : pre-emptive. (Q2)

T.A.T = C.T − A.T
W.T = T.A.T − BT

| priority | P.NO | A.T | B.T | C.T | T.A.T | W.T |
|---|---|---|---|---|---|---|
| 5 | 1 | 1 | 4 3 2 | 16 | 15 | 11 |
| 2 ← low priority | 2 | 2 | 5 | 21 | 19 | 14 |
| 6 | 3 | 3 | 6 5 | 14 | 11 | 5 |
| 4 | 4 | 0 | 1 | 1 | 1 | 0 |
| 7 | 5 | 4 | 2 1 | 9 | 5 | 3 |
| 8 ← high priority | 6 | 5 | 3 | 8 | 3 | 0 |

Gruantt Chart




So Avg W.T = $\frac{33}{6}$

$= 5.5$

Date ___/___/_____

RBR

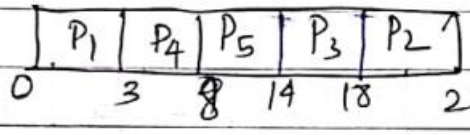
Largest Job First : opposit of SJF.

Selection Criteria → Brust / Mode → Non-preemptive and preemptive.

• Q-1 (Non-preemptive).

| P.No | A.T | B.T | C.T | T.A.T | W.T |
|---|---|---|---|---|---|
| 1 | 0 | 3 | 3 | 3 | 0 |
| 2 | 1 | 2 | 20 | 19 | 17 |
| 3 | 2 | 4 | 18 | 16 | 12 |
| 4 | 3 | 5 | 8 | 5 | 0 |
| 5 | 4 | 6 | 14 | 10 | 4 |

Guantt chart




Avg T.AT = $\frac{53}{5}$ ⇒ 10.6 Avg W.T = $\frac{33}{5}$ = 6.6.

• Q-2 SJF → Non-preemptive.

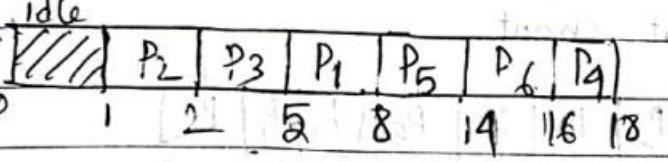
* → If the Brust time of the process are matching, then schedul the process which has the lowest arrival time.

| P.NO | A.T | B.T | C.T | T.A.T | W.T |
|---|---|---|---|---|---|
| 1 | 3 | 3 | 8 | 5 | 2 |
| 2 | 1 | 1 | 2 | 1 | 0 |
| 3 | 2 | 3 | 5 | 3 | 0 |
| 4 | 4 | 2 | 18 | 14 | 12 |
| 5 | 6 | 6 | 14 | 8 | 2 |
| 6 | 2 | 2 | 16 | 14 | 12 |

Avg T.A.T = $\frac{45}{6}$ ⇒ 7.5

Avg W.T = $\frac{28}{6}$ = 4.66

Guantt chart



Saathi

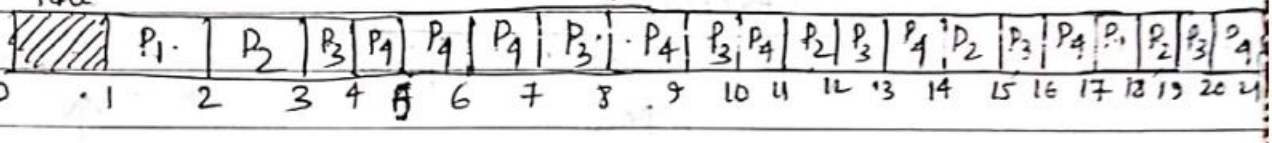
Date ___/___/_____

## * (LRTF)

• LJF → pre-emptive ⇒ (Largest Remaining time first).

| P.No | A.T | B.T | C.P | T.A.T | W.T |
|---|---|---|---|---|---|
| 1 | 1 | 2 | 18 | 17 | 15 |
| 2 | 2 | 4 | 19 | 17 | 13 |
| 3 | 3 | 6 | 20 | 17 | 11 |
| 4 | 4 | 8 | 21 | 17 | 9 |

Gnantt time




Avg T.A.T = 17

Avg W.T = $\frac{48}{4}$ ⇒ 12

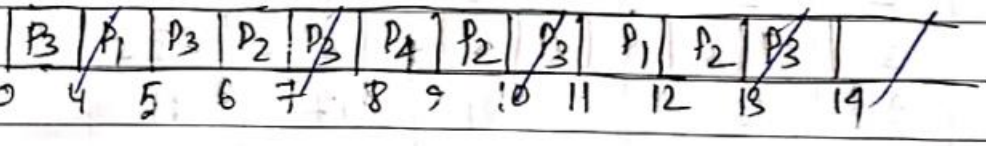
(2006)

* • GATE Problem :- LRTF.

* = What is avg T.A.T using LRTF.

| P.NO | A.T | B.T | C.T | T.A.T | W.T |
|---|---|---|---|---|---|
| 1 | 0 | 2 | 12 | 12 | 10 |
| 2 | 0 | 4 | 13 | 13 | 9 |
| 3 | 0 | 8 | 19 | 19 | 6 |

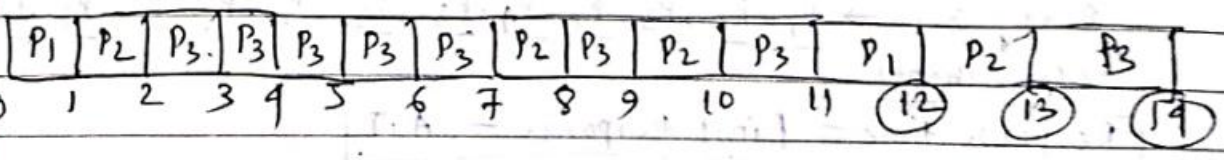
Gnant chart




Avg T.A.T = $\frac{39}{3}$ ⇒ 13.

Avg W.T = $\frac{25}{3}$ ⇒ 8.3.

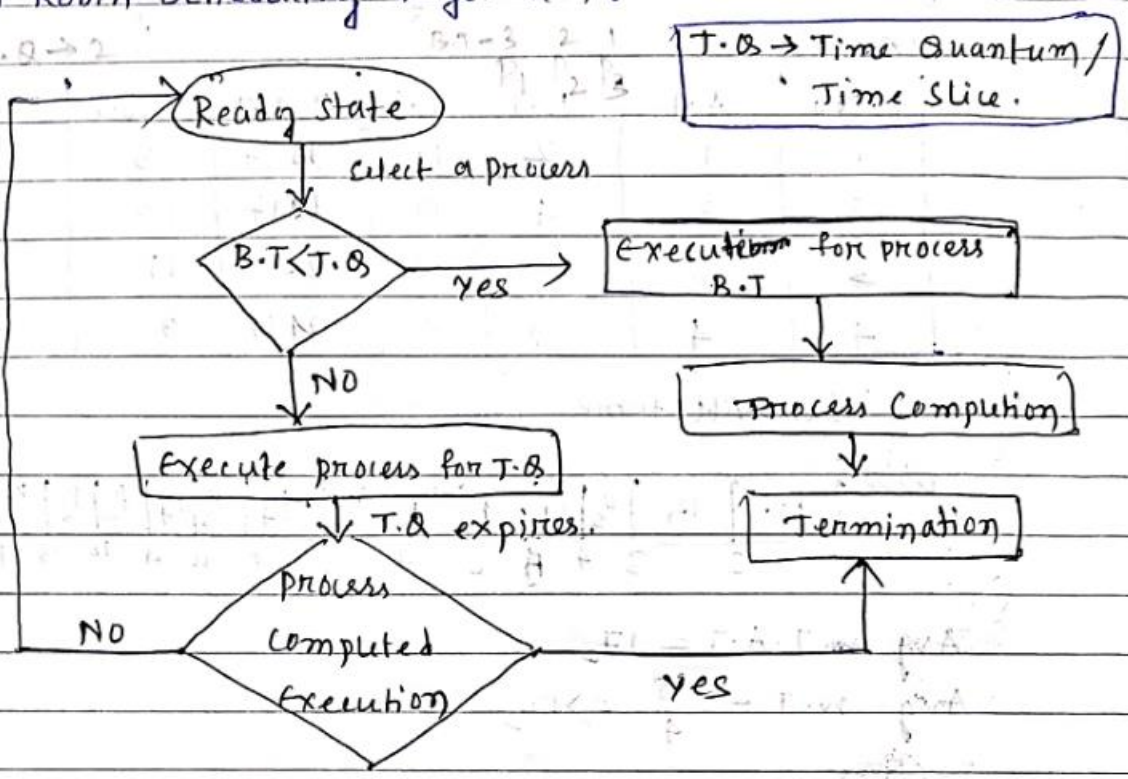
Gnantt chart



Date ___/___/_____

RBR

## Round Robin Scheduling Algorithm:




- Round Robin scheduling :- (Q-1)

*. Selection criteria: Time Quantum or Time slice.

*. Mode : pre-emptive.

(T.Q = 2)

| R.T | P.No | A.T | B.T | C.T | T.A.T | W.T |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 4 | 8 | 8 | 4 |
| 1 | 2 | 1 | 5 | 18 | 17 | 12 |
| 2 | 3 | 2 | 2 | 6 | 4 | 2 |
| 5 | 4 | 3 | 1 | 9 | 6 | 5 |
| 5 | 5 | 4 | 6 | 21 | 17 | 11 |
| 10 | 6 | 5 | 3 | 19 | 14 | 11 |

Avg R.T = $\frac{23}{6} \Rightarrow 3.8$

Gruantt chart

[Avg = $\frac{66}{6}$ = 11

[Avg = $\frac{45}{6}$ = 7.5

| P1 | P2 | P3 | P1 | P4 | P5 | P2 | P6 | P5 | P2 | P6 | P5 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2 | 4 | 6 | 8 | 9 | 11 | 13 | 15 | 17 | 18 | 19 | 21 |

Ready queue → P1, P2, P3, P1, P4, P5, P2, P6, P5, P2, P6, P5

(flow chart)

**Response time = First Response − A.T**

Date

- Round Robin Scheduling :- (Q-2)

T.Q = 3

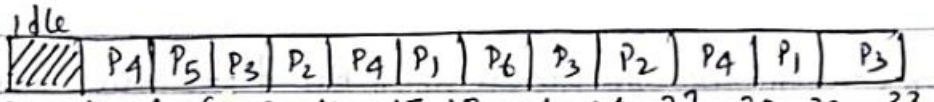
| P.NO | A.T | B.T |
|---|---|---|
| 1 | 5 | 5 |
| 2 | 4 | 6 |
| 3 | 3 | 7 |
| 4 | 1 | 9 |
| 5 | 2 | 2 |
| 6 | 6 | 3 |

| C.T | T.A.T | W.T |
|---|---|---|
| 32 | 27 | 22 |
| 27 | 23 | 17 |
| 33 | 30 | 23 |
| 30 | 29 | 20 |
| 6 | 4 | 2 |
| 21 | 15 | 12 |

(time sharing process)

Ready Queue → $P_4^{\checkmark}, P_5^{\checkmark}, P_3^{\checkmark}, P_2^{\checkmark}, P_4^{\checkmark}, P_1^{\checkmark}, P_6^{\checkmark}, P_3^{\checkmark}, P_2^{\checkmark}, P_4^{\checkmark}, P_1^{\checkmark}, P_3^{\checkmark}$

Gruantt chart




Avg-T.A.T = $\frac{128}{6} \Rightarrow 21.3$

Avg-W.T = $\frac{96}{6} \Rightarrow 16$

- Round Robin scheduling :- (Q-3)

TQ = 2

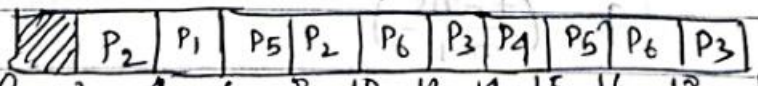
| P.NO | A.T | B.T |
|---|---|---|
| 1 | 3 | 2 |
| 2 | 2 | 4 |
| 3 | 6 | 3 |
| 4 | 8 | 1 |
| 5 | 4 | 3 |
| 6 | 5 | 4 |

| C.T | T.A.T | W.T |
|---|---|---|
| 6 | 3 | 1 |
| 10 | 8 | 4 |
| 19 | 13 | 10 |
| 15 | 7 | 6 |
| 16 | 12 | 9 |
| 18 | 13 | 9 |

Gruantt chart

avg = $\frac{56}{6}$ = 9.3

avg = $\frac{39}{6} \Rightarrow 6.5$




Ready Queue :-

→ $P_2^{\checkmark}, P_1^{\checkmark}, P_5^{\checkmark}, P_2^{\checkmark}, P_6^{\checkmark}, P_3^{\checkmark}, P_4^{\checkmark}, P_5^{\checkmark}, P_6^{\checkmark}, P_3^{\checkmark}$

Saathi

Date __/__/____

(1)

• Question :- Consider a system with 4 processes $P_1, P_2, P_3, P_4$. The Burst time Requirements of these process are 4, 1, 8, 1 respectively. Then what is the Completion time of process $P_1$ assuming Round Robin scheduling.

(T.Q / TS = 1)

→ $P_1 \rightarrow 4$ Let, Arrival time of all process '0'.

$P_2 \rightarrow 1$ then,

$P_3 \rightarrow 8$ Ready queue → $P_1, P_2, P_3, P_4, P_1, P_3, P_1, P_3, P_1, P_3$

$P_4 \rightarrow 1$

| $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_1$ | $P_3$ | $P_1$ | $P_3$ | $P_1$ | $P_3$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | (9) |

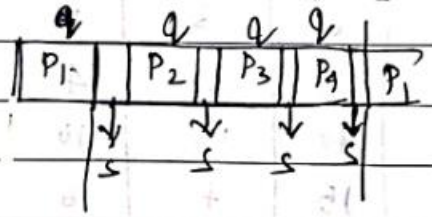
So, the Completion time of $P_1$ is = 9.

(2)

• Question :- Consider the process which has 'n' processes sharing the cpu in Round Robin fashion. The Context switching time is 's' units. Then what must be the time Quantum 'q' such that each process is guaranteed to get its turn at the cpu for every t seconds of time.

(i) $q = \frac{t-ns}{n+1}$ (ii) $q = \frac{t+ns}{n-1}$ (iii) $q = \frac{t-ns}{n-1}$ (iv) $q = \frac{t-ns}{n+1}$

→ Let, n = 4 ($P_1, P_2, P_3, P_4$)

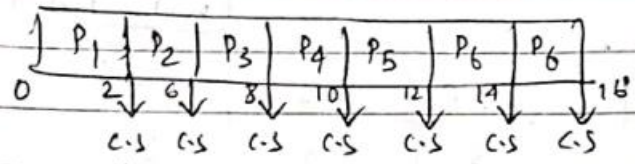



total time, $t = 3q + 4s$

So, for n Processes - $t = ns + (n-1)q$

$$q = \left(\frac{t-ns}{n-1}\right)$$

Date ___ / ___ / _____




*. Time Quantum (T.Q) ↓ → C.S ↑
→ Response time decrease.

*. T.Q ↑ → C.S ↓
→ Response time Increase.

*. T.Q ↑↑↑ → FCFS.

- Round Robin scheduling conclusion:

→ If the time Quantum is less, then the no of context switching -es will Increase and response time will be less.

→ If the time Quantum is large, then number of context switches will decrease and response time will be more.

→ If the time Quantum is very very large, the algorithm degenerates to FCFS Algorithm.

→ Round Robin is used to decrease the Response time.

RR

- Highest Response Ratio Next (HRRN)

selection criteria → Response Ratio. (high)

Mode → Non-pre-emptive.

$$\text{Response ratio (RR)} = \frac{W+S}{S}$$

W → waiting time.

S → service time or Burst time.

The HRRN favours the shortest jobs and also limits the waiting time of larger jobs.

Saathi

Date / /

$W.T = T.A.T - B.T$

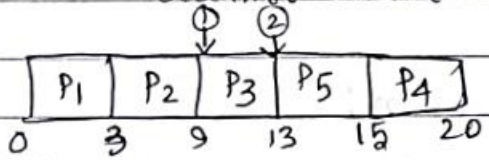
• Question 1: ~~HNA~~ HRRN.

| P.NO | A.T | B.T |
|---|---|---|
| ✓1 | 0 | 3 |
| ✓2 | 2 | 6 |
| ✓3 | 4 | 4 |
| 4 | 6 | 5 |
| ✓5 | 8 | 2 |

| C.T | T.A.T | W.T |
|---|---|---|
| 3 | 3 | 0 |
| 9 | 7 | 1 |
| 13 | 9 | 5 |
| 20 | 14 | 9 |
| 15 | 7 | 5 |

Gruantt chart

avg $= \frac{20}{5} \Rightarrow (1)$




(i) $RR_3 = \frac{W+S}{S} = \frac{(9-4)+4}{4} = \text{ans } (2.5)$

$RR_4 = \frac{W+S}{S} = \frac{(9-6)+5}{5} = \frac{8}{5} \Rightarrow 1.6$

$R.R_5 = \frac{(9-8)+2}{2} = \frac{3}{2} \Rightarrow 1.5$

(ii) $RR_4 = \frac{(13-6)+5}{5} = \frac{12}{5} = 2.$

$RR_5 = \frac{(13-8)+2}{2} = \frac{7}{2} = (3.5)$

So, Average waiting time (W.T) = 4.

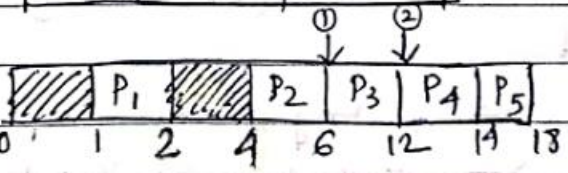
• Question 2: HRRN.

| P.NO | A.T | B.T |
|---|---|---|
| ✓1 | 1 | 1 |
| ✓2 | 4 | 2 |
| ✓3 | 5 | 6 |
| 4 | 6 | 2 |
| 5 | 7 | 9 |

| C.T | T.A.T | W.T |
|---|---|---|
| 2 | 1 | 0 |
| 6 | 2 | 0 |
| 12 | 7 | 1 |
| 14 | 8 | 6 |
| 18 | 11 | 7 |

avg W.T $= \frac{14}{5} = 2.8.$

Gruantt chart




(i) $RR_3 = \frac{(6-5)+6}{6} = \frac{7}{6} = 1.$

$RR_4 = \frac{(6-6)+2}{2} = 1$

(ii) $RR_4 = \frac{(12-6)+2}{2} = 4$

$RR_5 = \frac{(12-7)+4}{4} = \frac{9}{4} = 2.1$

Page No. 68

Date ___ / ___ / _____

RBR

## CPU Scheduling & I/O Scheduling –

• Question :-) what is completion time of process $P_1$, $P_2$, $P_3$ using SRTF.

| P.NO | A.T | cpu time | Yo time | cpu time |
|---|---|---|---|---|
| 1 | 0 | 1 | 2 | 2 |
| 2 | 1 | 2 | 4 | 5 |
| 3 | 2 | 3 | 6 | 8 |

Note :-

(1) The process first spends cpu time followed by I/O time, followed by cpu time.

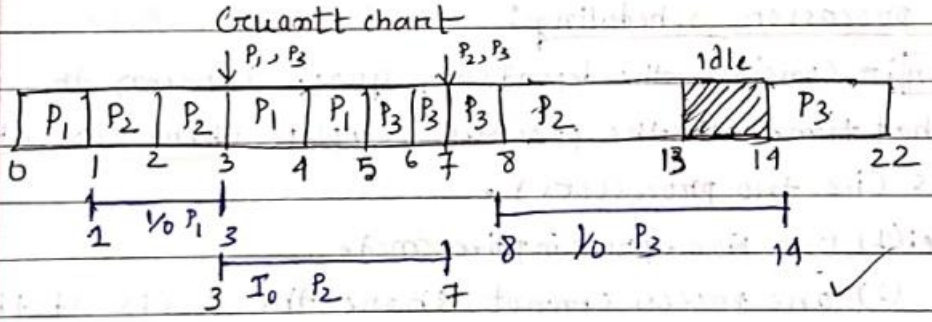
(2) I/O time of the process can be overlapped as much as possible.

→

| P.NO | A.T | cpu time | Yo time | cpu time | e.T |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 2 | 2 | 5 |
| 2 | 1 | 2 | 4 | 5 | 13 |
| 3 | 2 | 3 | 6 | 8 | 22 |

– completion time

Gruantt chart

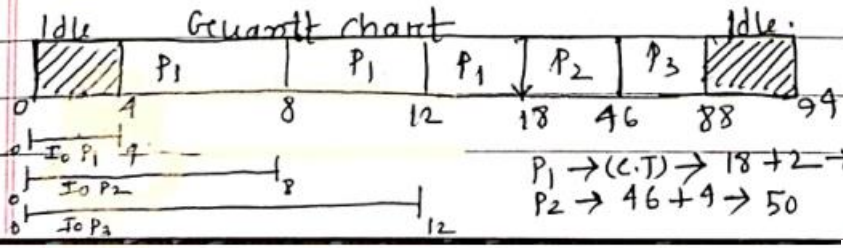



• Question 2 :-) Using SRTF

| P.NO | A.T | Yo time | cpu time | Yo time |
|---|---|---|---|---|
| 1 | 0 | 4 | 14 | 2 |
| 2 | 0 | 8 | 28 | 4 |
| 3 | 0 | 12 | 42 | 6 |

e.T

Idle Gruantt chart Idle.




$P_1 \rightarrow (C.T) \rightarrow 18+2 \rightarrow 20$

$P_2 \rightarrow 46+4 \rightarrow 50$

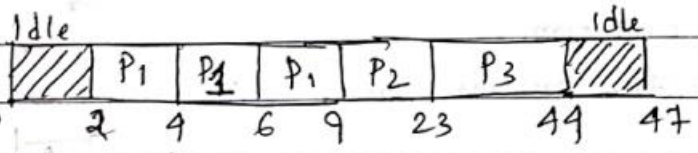
$P_3 \rightarrow 88+6 \rightarrow 94$

Date __/__/____

GATE problem: 2006 Question. → SRTF.

| P.NO | A.T | B.T Total | I/O time(20%) | CPU time(70%) | I/O time(10%) |
|---|---|---|---|---|---|
| $P_1$ | 0 | 10 | 2 | 7 | 1 |
| $P_2$ | 0 | 20 | 4 | 14 | 2 |
| $P_3$ | 0 | 30 | 6 | 21 | 3 |

Gruant chart




So,

completion time of $P_1 = 9+1 \Rightarrow 10$

$P_2 = 23+2 \Rightarrow 25$

$P_3 = 44+3 \Rightarrow 47$

Total c.p.u idle time, $\Rightarrow 2+3$

$\Rightarrow 5 \Rightarrow \frac{5}{47} \times 100 \Rightarrow$ 10.6.%

Multi processors Scheduling:

Question:- Consider the dependancy graph between the process. At what time all the processes complete there execution using 2cpu's (i.e two processors).

Note: (1) Use Non-preemptive mode.

(2) one process cannot share the 2 cps at the same time.

(a) 17 (b) 18 (c) 19 (d) 20.

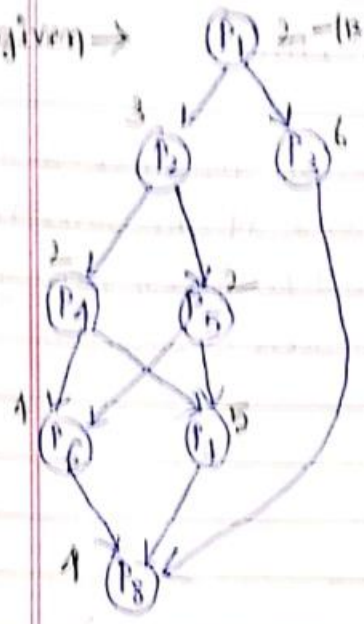
given-

→ (Next page).

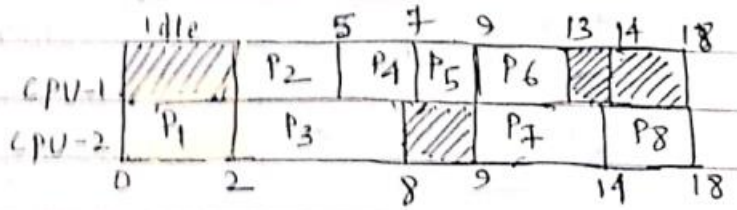
Saathi

Date

given →




Gruant chart




So, Completion time → 18.

## Multilevel Queue Scheduling :

(Ready Queue)

RQ1 →

RQ2 →

RQ3 →

RQ4 →

→ Depending on the proprity of the process, In which particular Queue, the process has to be placed will be decided.

→ The high priority process will be placed in the topa level ready Queue and low priority process will be placed in the Bottom level ready queue.

→ only after completion of all the process-es from the top level ready queue, the further level ready queue process will be scheduled.

→ If this is the strategy which is followed then the processes which are placed in the bottom level ready queue will suffer from STARVATION.

STARVATION : The Indefinate waiting of a process is called as STARVATION.

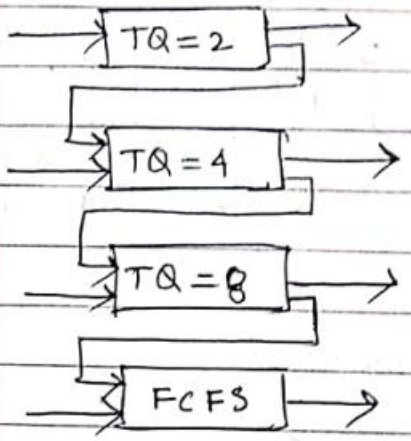
→ It is also possible to use different scheduling algorithms in the different ready queue.

Date / /

Saathi

RBP

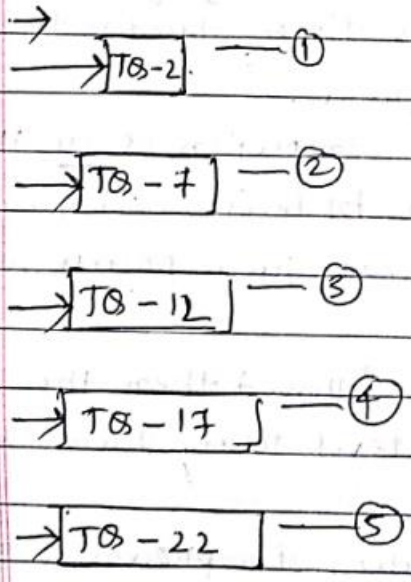
## MULTILEVEL FEEDBACK QUEUE SCHEDULING -




→ Algorithm avoids the problem of starvation and at the same time preference will be given to high priority process.

• Question :- Consider the system which has CPU Bound ~~Bound~~ process which requires Burst time of 40 time Units. Multilevel feedback queue scheduling is used. The time quantum is 2 units and it will be incremented by 5 units in each level. How many time the process will be interrupted and in which queue, process will complete the execution ?

(a) 4, 5 (b) 5, 6 (c) 3, 4 (d) 5, 5.




| | remain |
|---|---|
| 2 | 38 |
| 9 | 31 |
| 21 | 19 |
| 38 | 2 |
| 40 | 0 |

• 5th level complete.

• 4 time Interrupted.

Saathi

Date

Conclusion :-

| Algorithm | Starvation |
|---|---|
| 1) FCFS | (NO) |
| 2) SJF (NP) | yes |
| 3) SRTF | yes |
| 4) R.R | (NO) |
| 5) LJF (NP) | yes |
| 6) LRTF | yes. |
| 7) HRRN | (NO) |
| 8) priority (NP) | yes. |
| 9) priority (P) | yes |
| 10) MLQ | yes |
| 11) MLFQ | (NO) |

Date ___ / ___ / _____

# MEMORY MANAGEMENT

## /* INTRODUCTION */ (1)

- Functionality :- Allocating and deallocating memory to the process.
- GOAL :- Efficient utiligation of memory by using minimiging the internal and external fragmentation.

$2^{10} \Rightarrow 1024 \approx 10^3 \Rightarrow$ (1K)
$2^{20} \Rightarrow$ (1M)
$2^{30} \Rightarrow$ (1G)
$2^{40} \Rightarrow$ (1T)

- Memory Organization Map -

| | | |
|---|---|---|
| 0 | 1010 1101 | 000 |
| 1 | ---- ---- | 001 |
| 2 | ---- ---- | 010 |
| 3 | ---- ---- | 011 |
| 4 | ---- ---- | 100 |
| 5 | ---- ---- | 101 |
| 6 | ---- ---- | 110 |
| 7 | ---- ---- | 111 |

(Memory)

8-Word $\rightarrow 2^3$

1 word $\rightarrow$ 8 bits

Capacity of the memory in Bytes,

$\Rightarrow$ no of words * word size

$\Rightarrow 8 * 8$

$\Rightarrow$ 64 bit

$\Rightarrow$ 8 byte.

- **Q-1** Consider a system which has 256 KW and each has size of 64 bits. Then what is the capacity of memory in bytes.

→ # no of words $\Rightarrow$ 256 K ——— (1)

size of word = 64 bit.
= 8 byte. ——— (2)

Capacity = (1) * (2)
= (256 X 8) byte. $\Rightarrow 2^8 * 2^{10} * 2^3 \Rightarrow 10^{21}$ byte
$= 2 \times 2^{20} =$ 2 MB.

**Q-2** Consider a system which has 512 M words and each word has the size of 16 bytes then capacity of memory is -

→ # no of words = 512 M — ①

Size of each word = 16 bytes — ②

Capacity of memory = ① * ②

$= 512M * 16$

$= 512 * 2^{20} * 2^4$

$= 2^9 * 2^{20} * 2^4$

$= 2^{33} \Rightarrow 2^{30} * 2^3$

$=$ **8 GB**

**Q-3** Consider a System where 32 binary bits are used to represent all the words of memory and each word has the size of 32 bits. Then what is the capacity of memory in Bytes -

→ # no of words $= 2^{32}$

word size = 32 bit

$\Rightarrow$ 4 bytes.

Capacity $= 2^{32} * 4$

$= 2^{32} * 2^2$

$= 2^{34}$

$= 2^{30} * 2^4$

$= 16$ GB.

2010

**Ram Chip Introduction :-**

**RAM chip implementation -**

**Q-1**

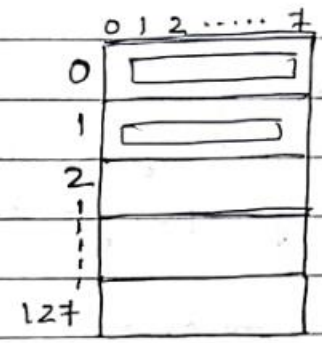
Ram chip size = 128 Bytes.

Organize the memory capacity of 16 KB by using above Ram chip -

a) No of RAM chips required.

b) Draw the memory Organization Map.

→ 128 B
- 128 : no of word
- B : word size.




✓

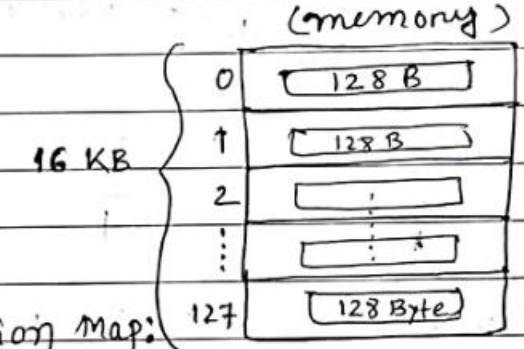
(a) NO of RAM chip required,

$= \frac{16\ KB}{128\ B}$

$= \frac{2^4 \times 2^{10}}{2^7}$

$= 2^7$

$= 128$




(b) Memory organigation Map:

• Q-2 (Ram chip implementation)

RAM chip :- 256 * 1 bit.

Memory capacity = 16 MB.

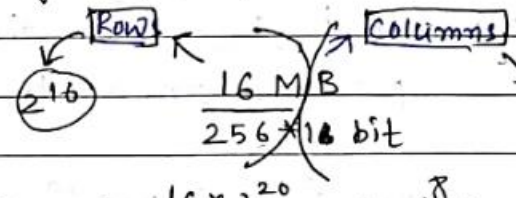
i) NO of RAM chip require $= \frac{16\ MB}{256 * 1\ bit} \Rightarrow \frac{2^4 * 2^{20} * 2^3}{2^8 * 2^1}$

$\Rightarrow 2^{19}$

$\Rightarrow 2^9 * 2^{10}$

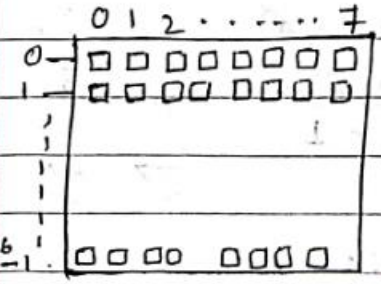
$\Rightarrow$ 256 K

ii) Memory organigation Map:




$\frac{16 \times 2^{20}}{2^8}$ $\quad \frac{8}{1}$

$= 2^{16}$.




memory (16MB)

RAM chip (no) * RAM chip (size) = 16MB

$\Rightarrow$ RAM chip no $= \frac{16\ MB}{256 * 1\ bit}$

$= 256K$

• **8-3** Ram chip Implementation -

Ram chip Size = 512 * 2 bit.

Memory capacity = 32 MB.

1) No of Ram chip = $\frac{32\,MB}{512*2\,bit} \Rightarrow \frac{2^5*2^{20}*2^3}{2^9*2^1}$

$= 2^{18}$.

$(2^{16})$ Row

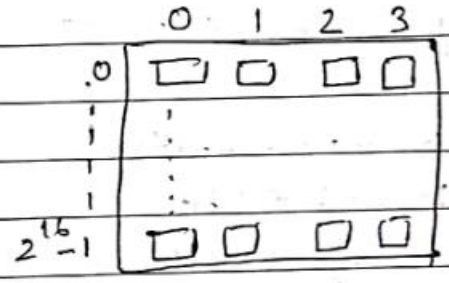
32 MB → Column (4)

512 * 2 bit

$\frac{2^5*2^{20}}{2^9} = 2^{16}$

$\frac{8}{2} = 4$

2) Map -



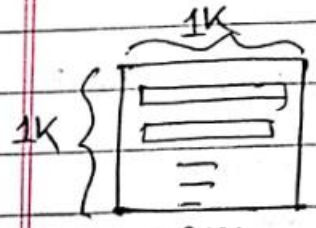

main memory (32 MB) :

GATE (2010)-8 :

A main memory unit with capacity of 4MB is built using 1M * 1 bit DRAM chips. Each DRAM chip has 1K rows of cells with 1K cells in each row. The time taken for single refresh operation is 100 ns. The time required to perform one refresh operation on all the cells in the memory unit is -

a) 100 ns b) $100*2^{10}$ ns c) $100*2^{20}$ ns d) $3200*2^{20}$ ns

→ no of RAM chip = $\frac{4\,MB}{1M*1\,bit} \Rightarrow \frac{2^2*2^{20}*2^3}{2^{20}*2^0} \Rightarrow 2^5 \Rightarrow 32$




cells in one DRAM chip, = 1K * 1K

$= 2^{10}*2^{10} \Rightarrow 2^{20}$ cells.

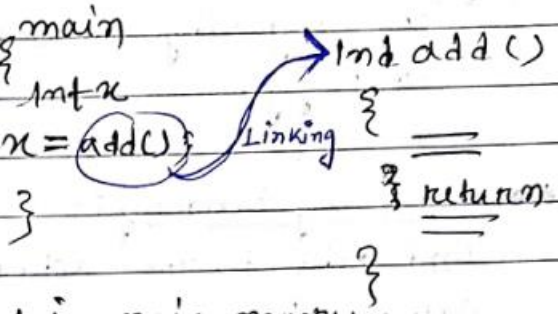
Total cells to built (4MB) main memory = $32*2^{20}$

$\Rightarrow 32*2^{20}*100$ ns

$= 3200*2^{20}$ ns

## Loading, Linking and Address Binding:

→ Loading & Linking
- Static.
- Dynamic.




Loading:- When program Load in main memory.

- **Static :-**

→ Loading the entire program into memory before the start of the program execution is called static loading.

→ Inefficient utilization of memory because whether it is required or not required, the entire program is brought into main memory.

→ The Program execution will be faster.

Linking → Linking all the modules / functions of the program.

→ If the static loading is used, then static linking will be applied.

- **Dynamic :-**

→ Loading the program into memory on demand is called as dynamic loading.

→ The efficient utilization of memory.

→ The program execution will be slower.

→ If the dynamic loading is used accordingly the dynamic

-linking will be applied.

→ The megority of O.S will use dynamic loading/linking strategy.

• Addressing Binding :- Association of program instruction and data to the actual physical memory locations is called Address Binding.

Program P1

$I_1$ → 1000
$I_2$ → 2000
$I_2$ → 3000

• Type of Address Binding :-

→ Compile time Address Binding (by compiler)
→ Load time A.B (Loader)
→ Dynamic or Execution time A.B (processor).

*• Compile time A.B :-

→ If the compiler is responsible of association of program instructions and data to the actual physical memory locations is called compile time A.B.

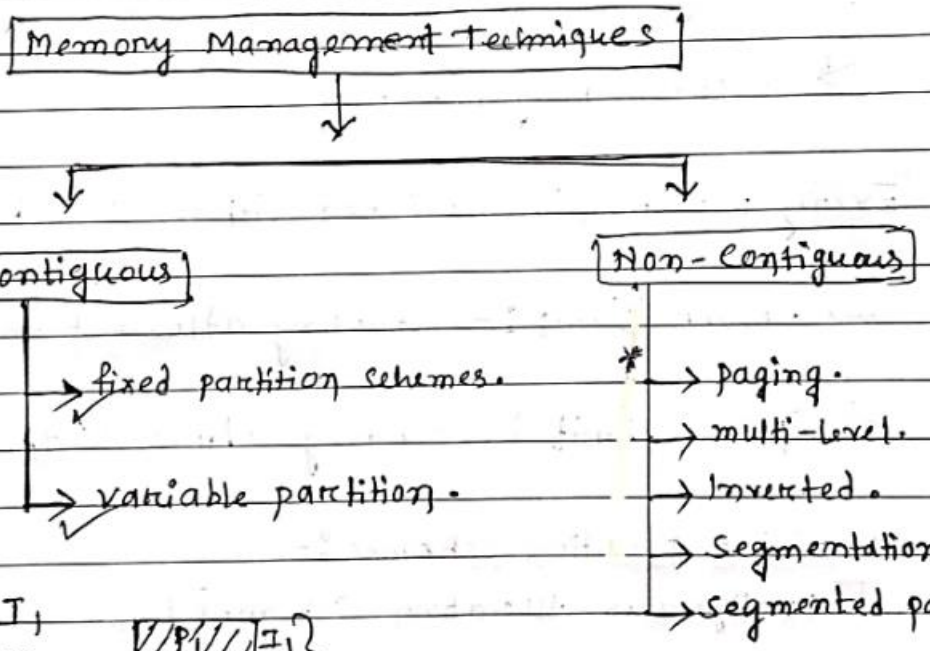
→ The compile time Address Binding will be done before loading the program into main memory.

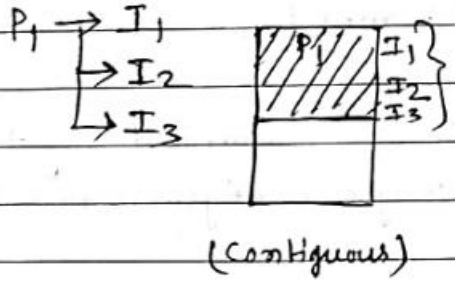
→ The compiler leads to interrupt with O.S memory manager to perform compile time address binding.

*. Load time A.B :- The Load time A.B will be done after the program is loaded into main memory.

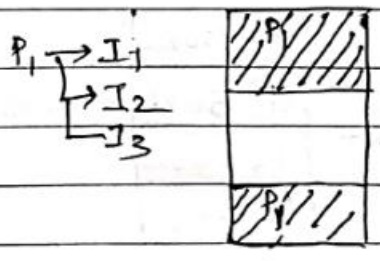
*. Execution time or dynamic A.B :- The Address binding will be postponded even after the loading the program into main memory.
→ This type of A.B will be done at the time of program execution.
→ processors does this type of A.M.
→ The Majority of the O.S uses the Dynamic A.B.

Memory Management Techniques Introduction :-



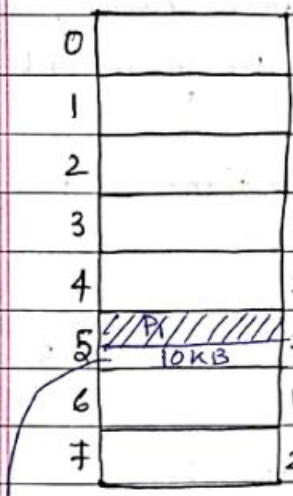

(Contiguous)




Non contiguous.

- **Fixed partition Schemes :-**

[Contiguous Allocation Schemes]




memory = 8

→ Internal fragmentation.

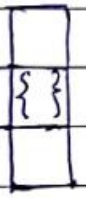
→ In fixed partition Schemes memory will be devided into fixed no of partitions. $P_1 = 60KB$

→ fixed means no of partitions are fixed not the size of partition.

→ In one partition only 1 process will be accomadated. = 8 process.

→ The degree of multiprogramming is restricted by the no of partitions in the main memory.

Every partition is associated with the limit Registers-

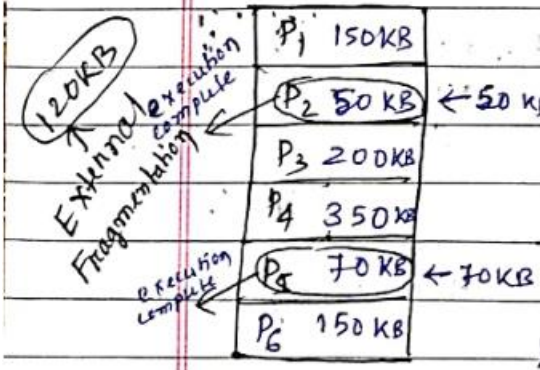



1) Lower limit :- starting address of the partition.

2) Upper limit :- Ending address of the partition.

- **Variable partition Schemes :-**

[Contiguous Allocation Schemes]




→ In this scheme, initially the memory will be single contiguous free block.

→ Whenever the request by the process in accordingly the partition will be made.

$P_1 \rightarrow 150KB$ | $P_4 \rightarrow 350$ KB
$P_2 \rightarrow 50KB$ | $P_5 \rightarrow 70$ KB
$P_3 \rightarrow 200$ KB | $P_6 \rightarrow 150KB$
($P_7 \rightarrow 100$ KB)

To Avoid the problem of External fragmentation:

*. Compaction :- is undesirable. Moving all the process towards the top or towards the bottom to make the free available memory in a single contiguous place is called as compaction.

→ Compaction is undesirable to implement because it disturbs all the running process in the memory.

• Partition Allocation Schemes:

When more than one partition is freely available to satisfy the request of the process then for the decision making of selecting a partition will be done by partition allocation methods.

1) First fit :- Allocate the process in the partition which is first sufficient partition from the top of the memory.

2) Best fit :- Allocate the process in the partition which has the smallest sufficient partition among the free available partition.

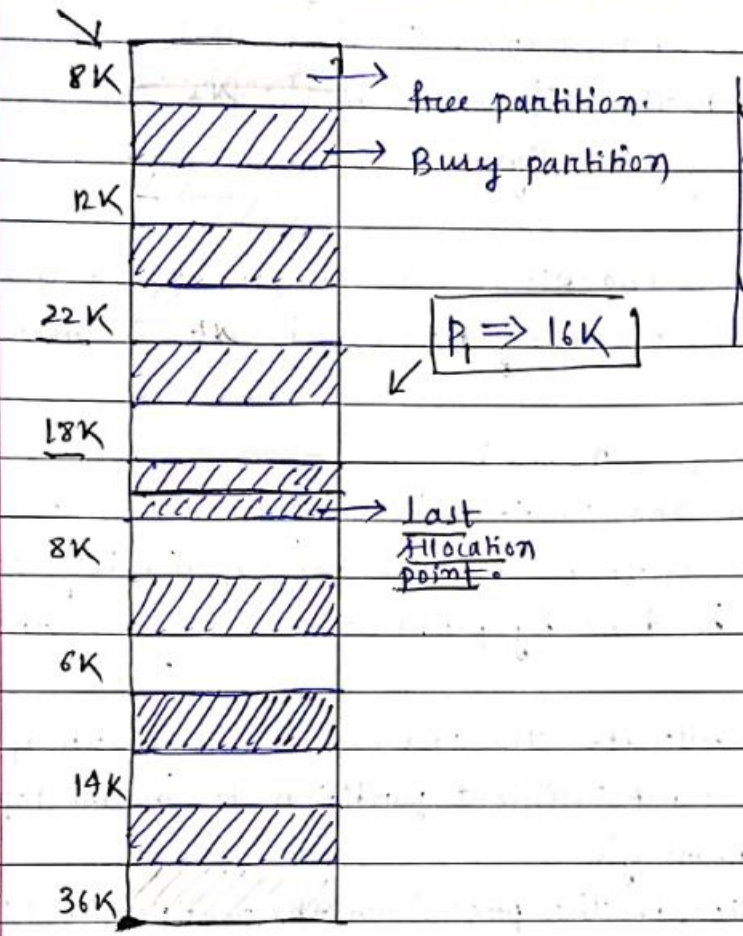
To do this, we need to do Searching. To find out the smallest partition, it requires searching all the partions in the memory.

3) Worst fit :- Allocate the process in a partition which is largest sufficient among all the free available partition.

To find out the largest sufficient, it requires to search all the partitions in the memory.

4) Next fit :- Next fit also works like the first fit but it will search for first sufficient partition from the last allocation point.

• Best fit, first fit, worst fit, & Next fit (Question-1) :-




1) first fit ⇒ 22K ($P_1$)
2) Best fit ⇒ 18K ($P_1$)
3) worst fit ⇒ 36K ($P_1$)
4) Next fit ⇒ 36K ($P_1$)

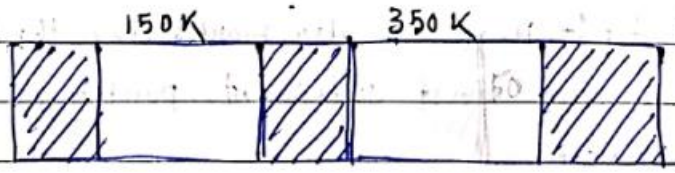
(Question-2) :- How many successive request of 6K will be satisfied by using the first fit and assuming variable partition scheme ? (from above diagram)

a) 16 b) 17 c) 18 d) 19.

→ 1+2+3+3+1+1+2+6 ⇒ 19

Gate (Question-3) :-

Request from the process are 300K, 25, 125K, and 50K respectively.




The above request could be satisfied with (Assume variable partition scheme)

a) Both best fit and first fit.

b) First fit but not Best fit.

c) Best fit but not first fit.

d) neither ~~best~~ first fit nor best fit.

(first fit)

| 150K | 350K |
|---|---|
| 125K | 50K |
| 0K | 0K |

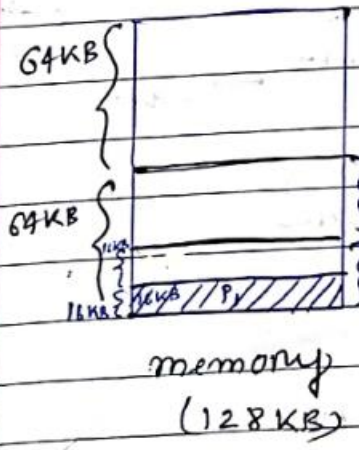
(best fit)

| 150K | 350K |
|---|---|
| 25K | 50K |
| | 25K |

Buddy System Theory: In the Buddy System initially the the memory will be single continous free block.

→ whenever the request by process comes in the memory will be devided into 2 half blocks.

→ If the request still small, the lower block of the memory is further divided into 2 half block again. In the buddy system, the memory will be allocated from the lower level blocks to the higher blocks.




Question-1 (Buddy System) -
In a Buddy system of memory management, successive request of 50K, 25K, and 35K are satisfied with 256K of Available memory. How many blocks (with size) are left.

Two Block (32K, 64K).

b) One block (94K)

c) One block (145K)

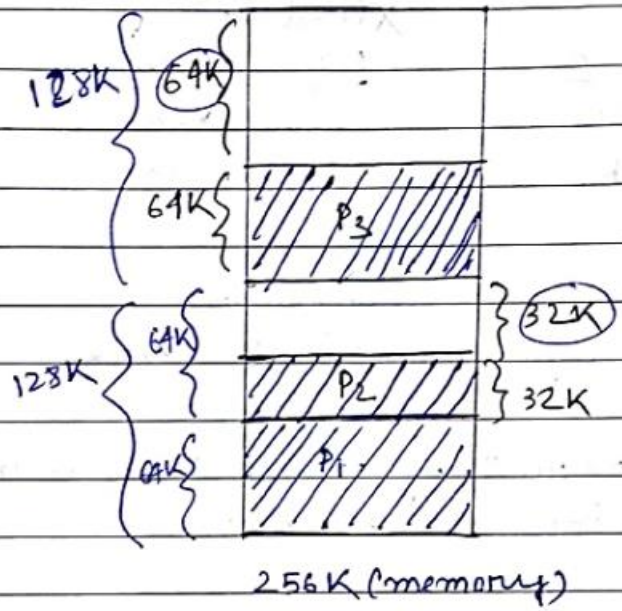
d) Memory blocks with varying sizes.

→

$P_1 - 50K$

$P_2 - 25K$

$P_3 - 35K$




GATE **

- PAGING (Non Contiguous) (11)

Logical Address Space (LAS) or virtual Address space (VAS) → Words → Bytes.

Logical Address or virtual Address (VA) → Bits.

physical Address space (PAS) → Words → Bytes.

Physical Address (PA) → Bits.

(1) L.A.S = 128 MW
$\Rightarrow 2^7 * 2^{20}$
$\Rightarrow 2^{27}$
L.A = 27 bits

(2) L.A = 38 bits
LAS = $2^{38}$
$= 2^8 * 2^{30}$
= 256 GW.

(3) P.A = 16 bits
PAS = $2^{16}$
$= 2^6 * 2^{10} \Rightarrow$ 64 KW.

(4) PAS = 32 MW
$= 2^5 * 2^{20}$
$= 2^{25}$
PA = 25 bits

## • LAS and PAS :

*. CPU always generate Logical Address.

L A >> PA

→ The Technique of Mapping processor generated logical Address to physical Address is called as paging.

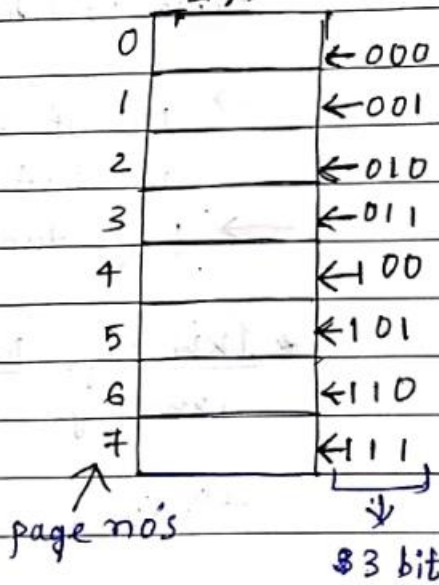
## • LAS division into PAGES –

L.A => 13 bits → L.A.S => $2^{13}$ => $2^3 * 2^{10}$ => 8KW => 8 pages.
P.A => 12 bits → PAS => $2^{12}$ => 4 KW.

① L.A.S will be devided into equal sige pages.

Page sige = 1KW

# pages = $\frac{\text{L.A.S}}{\text{page sige}}$ => $\frac{8\text{KW}}{1\text{KW}}$ => 8



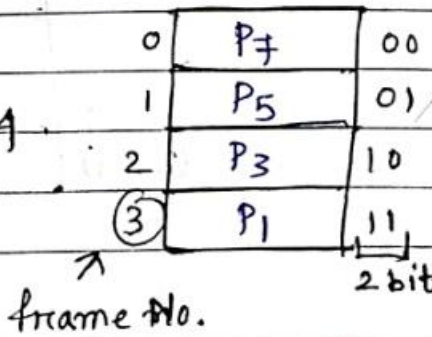

## • PAS division into Frames –

PAS is divided into equal sige frames.

* Page SIZE is Always same as frame size

Frame size => 1KW

# Frames => $\frac{\text{P.A.S}}{\text{frame sige}}$ => $\frac{4\text{KW}}{1\text{KW}}$ => 4.

## (*) Some Important Conclusions —

→ $\#\text{pages} = \frac{\text{L.A.S}}{\text{Page size}}$ — L.A.S is divided into equal size pages.

→ $\#\text{frames} = \frac{\text{P.A.S}}{\text{frame size}}$ — P.A.S is divided into equal size frames.

**PAGE size is Always Same as FRAME size**

Whenever the paging is applied the page the page table will be maintained.

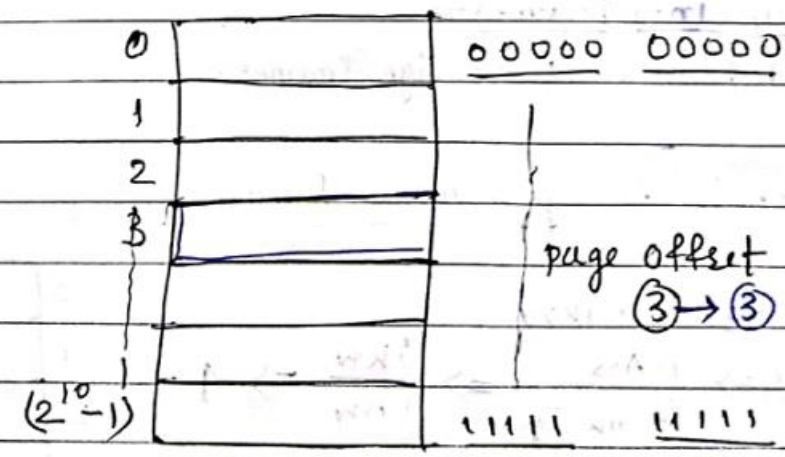
→ No of entries in the page table is same as no of page in L.A.S.

→ page table entry contains the frame no.

- **1kW page Details:**

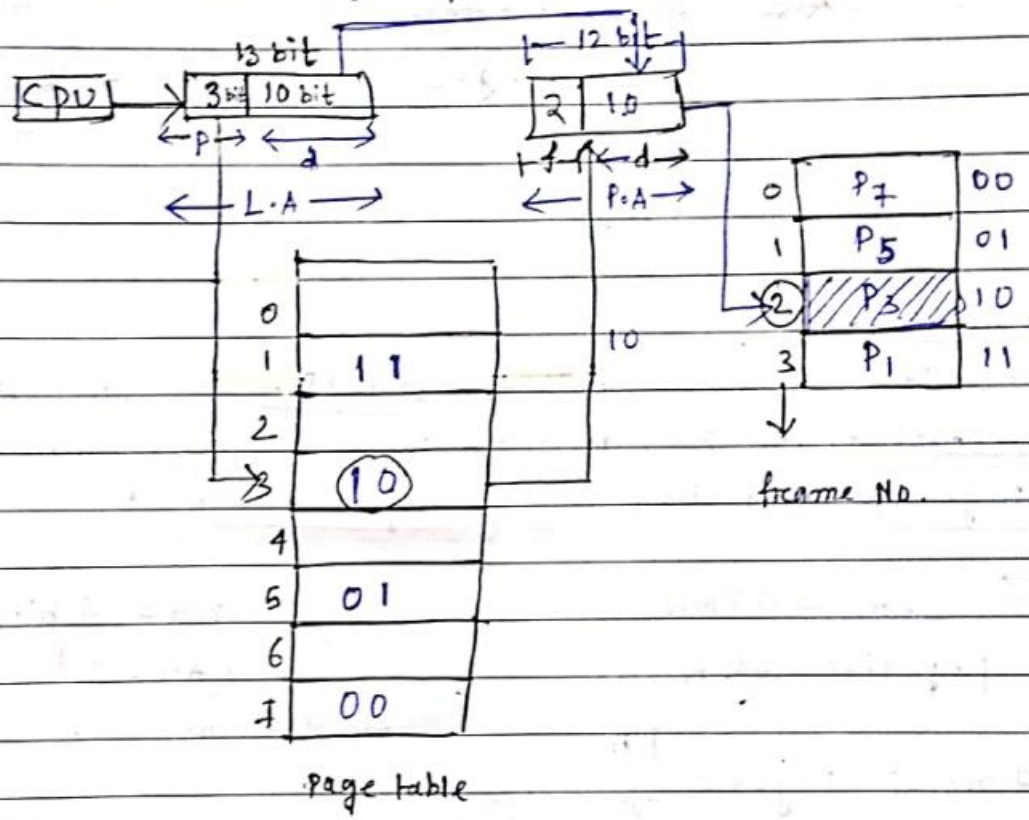
1kW page will look like this.

$1K = 2^{10}$ → (10 bits)



## • Paging Diagram :-

When; L.A => 13 bits.
P.A => 12 bits.
p → no of bits required to represent pages of L.A.S or page No.
d → no of bits required to represent the pages offset.
f → The no of bits required to represent the frame no.




## • Paging Technique Question - (1) :-

Consider a system which has logical Address = 27 bits and physical address = 21 bits, and page size = 4KW. calculate no of pages and no of frames. [page size = frame size]

→ L.A = 27 bits
L.A.S => $2^{27}$

# no of pages = $\frac{L.A.S}{4\,KW}$

$= \frac{2^{27}}{2^2 * 2^{10}} \Rightarrow (2^{15})$

$\Rightarrow 2^5 * 2^{10} \Rightarrow 32K$

P.A = 21 bits
P.A.S = $2^{21}$ bits

# No of frames = $\frac{P.A.S}{4\,KW} \Rightarrow \frac{2^{21}}{2^2 * 2^{10}}$

$\Rightarrow (2^9)$

• Question-(2).

Consider a system which has no of pages = 2K and page size is 4KW. The physical Address 18 bits then calculate L.A and no of frames.

→ # no of pages = $\frac{L.A.S}{page\ size}$

$\Rightarrow$ L.A.S = 2K * 4KW

$= 2 * 2^{10} * 2^{2} * 2^{10}$

$= 2^{23}$

L.A = 23 bits

P.A = 18.

# no of frames = $\frac{P.A.S}{no.\ of\ space}$

$= \frac{2^{18}}{4KW} \Rightarrow \frac{2^{18}}{2^2 * 2^{10}}$

$= 2^6$

$= 64$

• Question-(3)

Consider a system with L.A.S of 128MW and physical Address = 24 bits. The P.A.S divided into 8K frames. Then what is the page size and how many pages in L.A.S.

→ L.A.S = 128MW

page size = 2KW

# no of pages = $\frac{P.A.S}{size\ of\ page}$

$= \frac{128 * 2^{20}}{2 * 2^{10}}$

$= \frac{2^7 * 2^{20}}{2^{11}}$

$= 2^{16}$

$= 2^6 * 2^{10}$

= 64K

P.A = 24 bits

P.A.S = $2^{24}$

no of frames = 8K

# no of frams = $\frac{P.A.S}{frame\ size}$

$\Rightarrow$ frame size = $\frac{2^{24}}{8K}$

$= \frac{2^{24}}{2^3 * 2^{10}} \Rightarrow 2^{11}$.

$= 2 * 2^{10}$

= 2K

(page size = frame size)

Gate-2002

• Question — (4)

Consider a system with L.A equal to 32 bits. The P.A.S is 64 MB. The page size is 4KB. The memory is byte addressable. The page table entry size of 2 bytes. what is the approximate size of page table in bytes.

(a) 1MB (b) 2MB (c) 4MB (d) 8MB.

→ [No of entries in page table = No of pages in L.A.S]

No. of pages = $\frac{\text{L.A.S}}{\text{page size}}$

$= \frac{2^{32}}{4KB}$

$= \frac{2^{32}}{2^2 * 2^{10}}$

$= 2^{20}$

L.A = 32 bits

L.A.S = $2^{32}$ bi

So approximate size of page table = No of entries * entry size

$= 2^{20} * 2$

$= 2 * 2^{20} = 2M.$

• Question — (5)

consider a system having a page table with 4K entries. The L.A is 29 bits. what is the physical Address if the System has 512 frames.

→ L.A = 29 bits.

L.A.S = $2^{29}$.

# no of page = $\frac{\text{L.A.S}}{\text{page size}}$

page size = $\frac{2^{29}}{2^2 * 2^{10}}$ (4K)

$= 2^{17}$

[# of entries in P.T = # of page is L.A.S]

no of frames = 512

# no of frames = $\frac{\text{P.A.S}}{\text{frame size}}$

frame size = $\frac{\text{P.A.S}}{512}$

⇒ P.A.S = $512 * 2^{17}$

$= 2^9 * 2^{17}$

$= 2^6 * 2^{20}$

$= 2^{26}$

[frame size = page size]

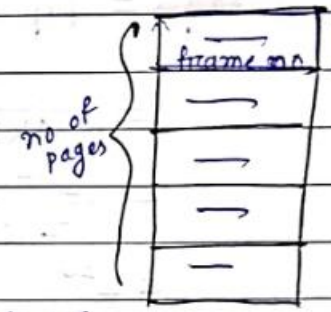
✓ [P.A = 26].

• Question - (6)

Consider a Question system where the L.A.S = P.A.S = $2^{16}$ bytes and page size => 512 bytes. The memory is byte addressable. The page table entry size is 2 bytes. The page table entry also contains other information like 1 bit for valid and invalid, 1 bit for reference, 3 bit for page information/protection and 1 bit for dirty bit. How many bits are still available in page table entry to store Ageing information.

→ 1 bit
1 bit
3 bit
1 bit
6 bit

16 bits




no of frame (pg entry) = P.A.S / frame size

= $\frac{2^{16}}{512} \Rightarrow \frac{2^{16}}{2^9} \Rightarrow 2^7$

7 bit require to identify a particular frame no.

16 − (6+7)
= 3 bits (freely available) to store Ageing information.

• Question - (7)

Consider a system which has L.A.S of 256 MB. The physical Address = 24 bits. The memory is byte addressable. The P.A.S is divided into 8 KB frames. Then how many pages are there in the L.A.S.

→ L.A.S = 256 MB
P.A = 24 bits.
~~no of frames = 8 KB~~

no of page = L.A.S / page size $\Rightarrow \frac{2^8 * 2^{20}}{8\,KB}$

$\Rightarrow \frac{2^8 * 2^{20}}{2^{13}}$

$\Rightarrow 2^{15}$

$\Rightarrow$ 32K

• Question-(8)

*[The internal fragmentation in the paging is considered as, (P/2) where p is the page size]

Consider a system L.A.S = P.A.S = 'S' Bytes. and
page size = 'P' Bytes.
page table entry sig = 'e' Bytes.
The memory is Byte addressable. what is the optimal value of page size by minimizing the memory overhead of maintaing page table and internal fragmentation in the paging.

(a) $P=\sqrt{2se^2}$ (b) $P=\sqrt{2s^2e}$ (c) ✓ $P=\sqrt{2se}$ (d) $P=\sqrt{2(se)^2}$.

→ No of pages $= \frac{L.A.S}{page\ size}$

$= \frac{S}{P}$

P.T = $\frac{S}{P}$ * sig of entry

$= \frac{Se}{P}$

P.T + I.F

$\frac{Se}{P} + P/2 \Rightarrow \frac{d}{dp}$

$\Rightarrow \frac{d}{dp}\left(\frac{Se}{P}+\frac{P}{2}\right)=0$

$\Rightarrow -P^{-2}Se+\frac{1}{2}=0$

$\Rightarrow P^{-2}Se=\frac{1}{2}$

$\Rightarrow P^{-2}=\frac{1}{2Se}$

$= P^2=2Se$

$\boxed{P=\sqrt{2Se}}$ ✓

• Performance of paging:

→ The page table of the processes will be stored in the main memory.

→ The main memory Access time = M,

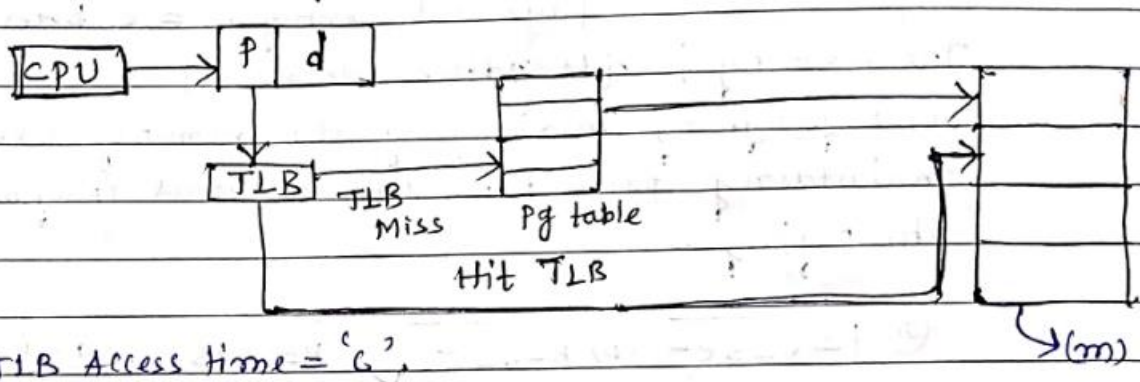
→ If the page table stored in the main memory, the formula for effective memory Access time (EMAT).

$\boxed{EMAT = 2m}$

- TLB Introduction:
(Translation Lookaside Buffer).

→ This is added to improve the performance of paging.

→ The TLB contains frequently referred page numbers and corresponding frame numbers.




TLB Access time = 'c'.
TLB hit ratio = $\alpha$

$$EMAT \Rightarrow \alpha(c+m) + (1-\alpha)(c+m+m)$$

$$EMAT \Rightarrow \underbrace{\alpha(c+m)}_{\text{TLB hit}} + \underbrace{(1-\alpha)(c+2m)}_{\text{TLB miss}}$$

- Q-(1)
Consider a system which has main memory Access time = 100 ns and TLB access time = 20 ns and TLB hit ratio 95% then what is the effective memory access time with TLB and without TLB.

→ m = 100 ns.
TLB = 20 ns
$\alpha$ = 95% => .95

(i) With TLB:

$$EMAT = \alpha(m+c) + (1-\alpha)(c+2m)$$
$$= 0.95(120) + (1-0.95)(20+200)$$
$$= 0.95 * (120) + 0.05 * 220$$
$$= 114 + 11$$
$$= 125 \text{ ns}.$$

(ii) without TLB:

EMAT = 2m

= 2 * 100

= 200 ns

Q-(2)

What hit ratio is required to reduce effective memory access time from 300ns without TLB to 250 with TLB. TLB Access time is 60ns.

→

without TLB = 300ns

EMAT ⇒ 2m ⇒ 300ns

m ⇒ 150ns.

with TLB = 250

EMAT = x(c+m) + (1-x)(c+2m)

⇒ 250 = x(60+150) + (1-x)(60+300)

⇒ 250 = 210x + 360 - 360x

⇒ 150x = 110

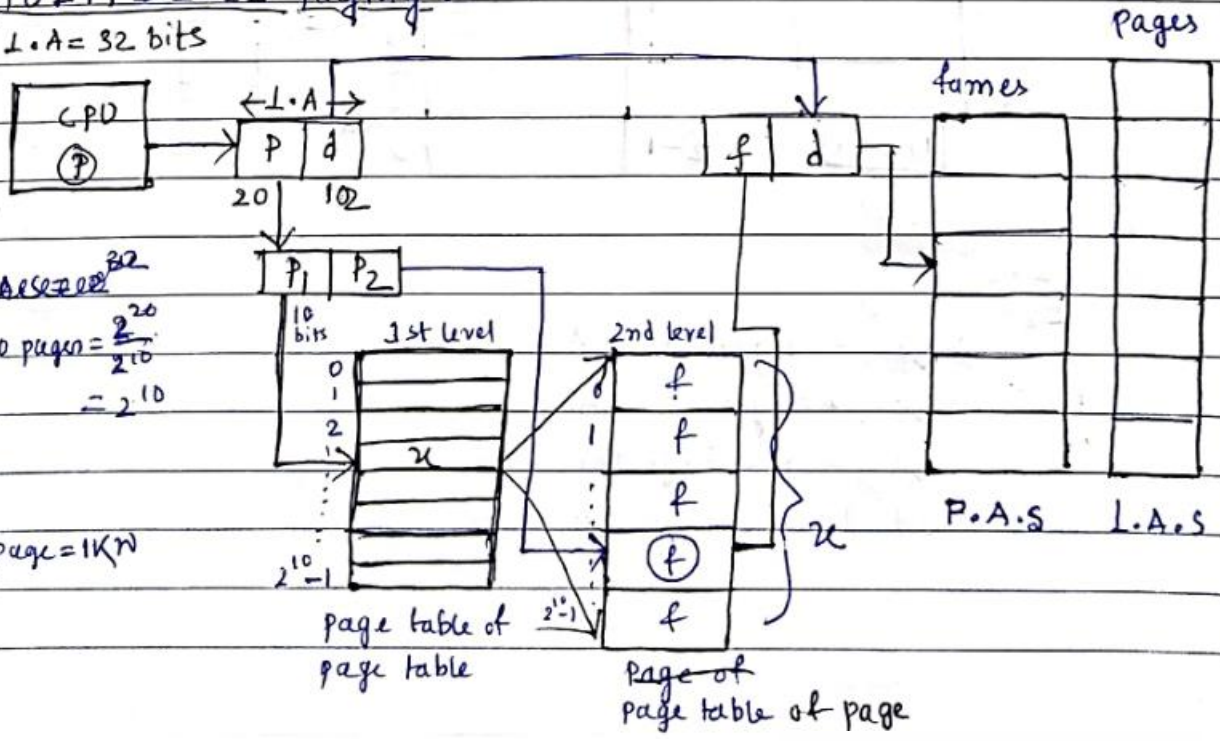
⇒ x = 11/15

⇒ x = .733

⇒ x = 73.3%.

MULTI LEVEL Paging:

L.A = 32 bits



→ TO Avoid overhead of maintaining large page tables, the multilevel paging will be Implemented.

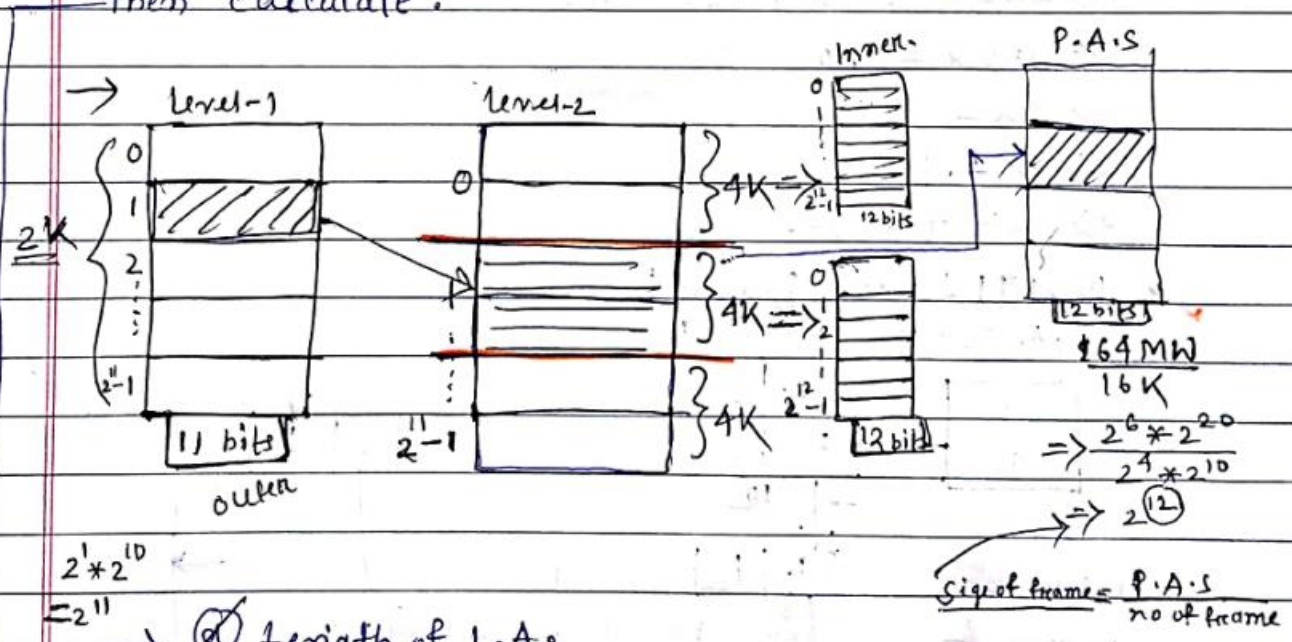
→ In the multilevel paging, the paging will be applied on page table.

$P_1$ → NO of bits required to represent the page of page table.

$P_2$ → The no of bits required to represent the page size of page table.

x → Address of page of page table.

**Q-1** consider a system with 2 level paging applicable. The page table has been divided into 2K pages. Each of size 4K words. If P.A.S is having 64 MW, which is divided into 16K frames, and the page table entry size is 4 Bytes, then calculate.




→ (A) Length of L.A.

(B) Length of PA.

(C) Outer page table size.

(D) Inner page table size.

(a) (11+12+12) ⇒ 35 bits.

(b) 64 * MB ⇒ $2^6 * 2^{20}$ ⇒ $2^{26}$

= 26 bits

(c) 2K * 1 Byte

⇒ $2^{11} * 1$

⇒ $2^{13}$ Byte.

(d) 4K * 4 Bytes.

= $2^{12} * 2^2$

= $2^{14}$ Bytes.

Q-2

Consider a system with 2 level paging appliable. The page table is devided into 8K pages, each of 16KB. The memory is ~~devi~~ byte addressable. If the p.A.s is 128 megabytes which is devided into 4KB frames. The page table entry sige of outer page table is 32 bits and page table entry size of inner page table is 64 bits. The calculate -

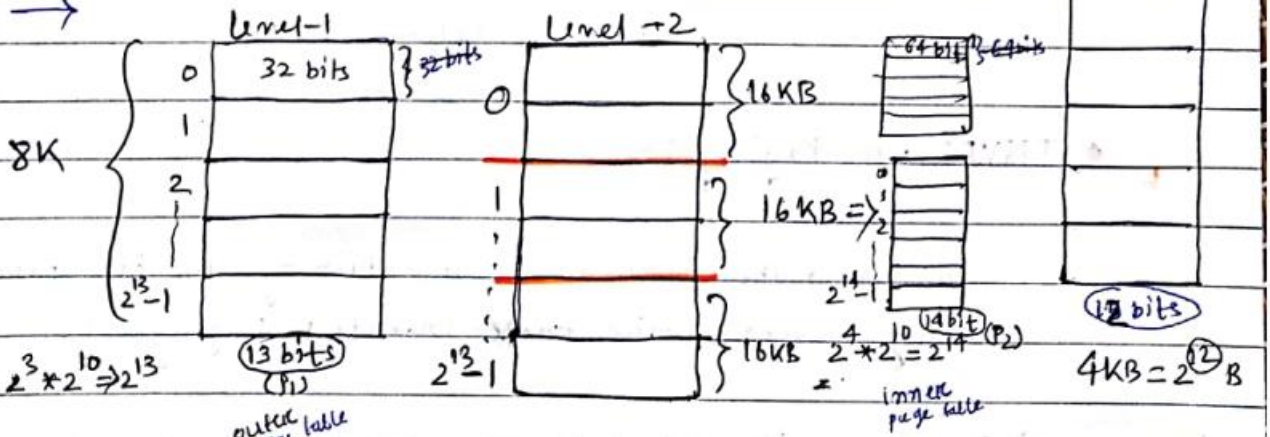
a) Length of LA. (13+14+12) bits ⇒ 39 bits.

b) Length of PA. $2^{27}$ ⇒ 27 bits.

c) Outer page table sige. $32 * 8K \Rightarrow 2^5 * 2^3 * 2^{10} \Rightarrow 2^{18}$ bits ⇒ $2^{15}$ Byts

d) Inner page table sige. $\frac{64}{\text{bit}} * 16KB \Rightarrow (2^{6-3} * 2^{4+10})B$ ⇒ $2^{17}$ Byte.

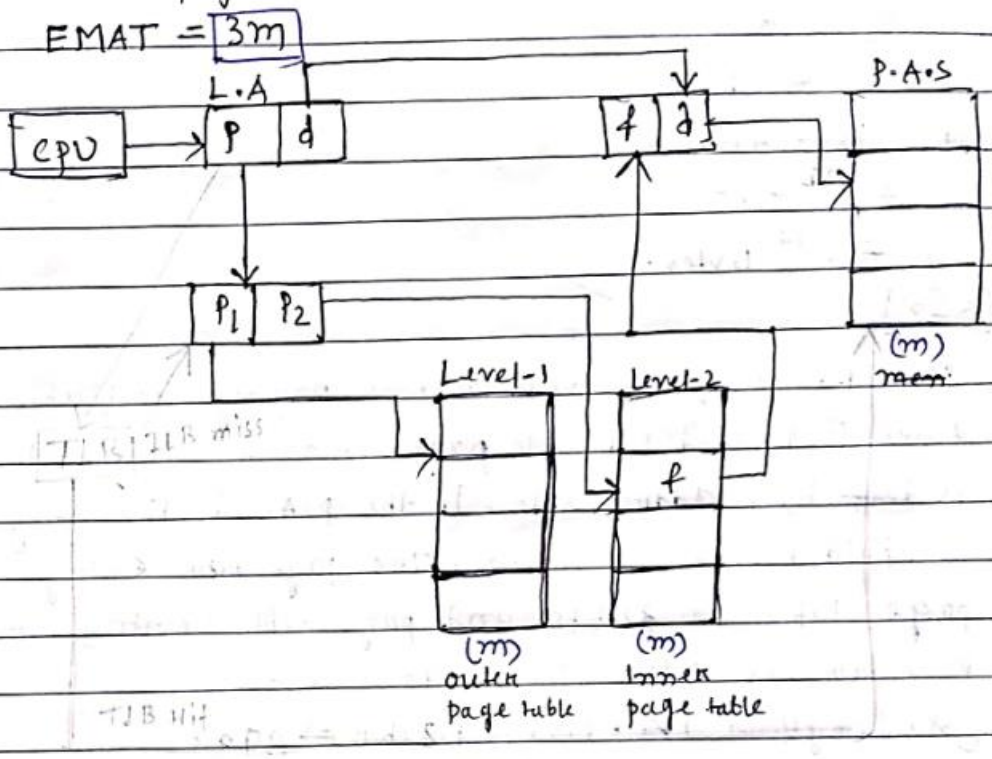
→



- Performance of 2 Level paging with out TLB :-

Let the main memory access time = 'm'.

→ If all the page tables are stored in the main memory.




→ If the TLB is added to improve the performance. The TLB contains frequently refferred page NO's and corresponding frame no.

EMAT = $x(c+m) + (1-x)(c+3m)$.

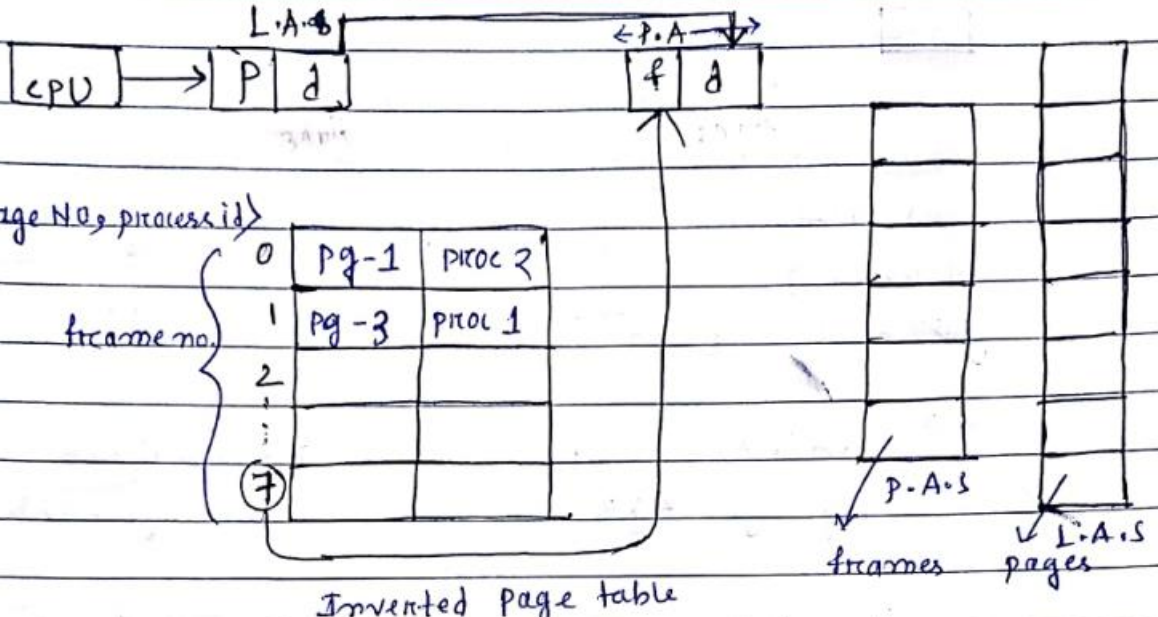
'c' = TLB access time

$x(c+m)$ : TLB Hit, $(1-x)(c+3m)$ : TLB miss.

- INVERTED Paging :-

→ To Avoid the overhead of maintaining the page table of every process, the ~~inert~~ Inverted paging will be implemented.

→ In ~~Inert~~ Inverted paging only 1 page table is maintained for all the process.




Inverted page table
OR
FRAME table

**Q-1**

Consider a system with Logical (L.A) Address = 34 bits and physical Address = 29 bits. The page size is 16 KB. The memory is byte Addressable. The page table entry size is 8 bytes. The calculate -

1) Conventional page table size.
2) Inverted page table size.

PgT size = No of entry in page table * size of entry

→ 1) Conventional p.T size,

No of entry = no of page

$$\text{No of entry} = \frac{\text{P.A.S}}{\text{page size}} \Rightarrow \frac{34 \text{ bits}}{16 \text{ KB}} \Rightarrow \frac{2^{34}}{2^{14}} \Rightarrow 2^{20}$$

$$\text{P.T size} = 2^{20} * 8 \text{ byte}$$

$$= 8 \text{ MB.}$$

2) Inverted page table size,

No of entry = no of frames

$$\text{No of entry} = \frac{\text{L.A.S}}{\text{frame size}} \Rightarrow \frac{2^{29}}{16 \text{ KB}} \Rightarrow \frac{2^{29}}{2^{14}} \Rightarrow 2^{15} \text{ Bytes}$$

$$\text{p.T size} = 2^{15} * 8 \text{ b} \Rightarrow 2^{18} \text{ B}$$

$$\Rightarrow 2^8 * 2^{10} \text{ B} \Rightarrow 256 \text{ KB.}$$

**Q-2**

Consider virtual Address space of 32 bits and page sige of 4KB. Consider RAM of 128 KB. The what will be the ratio of page table and inverted page table sige of each entry in Both is of 4 Bytes ?

(a) $2^{15}:1$ (b) $2^{20}:1$ (c) $2^{10}:1$ (d) None of the above.

→ L.A (v.A) = 32 bits
page sige = 4 KB.

P.A.S = 128KB
frame sige = 4KB

Page table sige = no of entrys * sige of entrys

$P.T_N = \left(\frac{2^{32}}{4KB}\right) * 4B$

$= \left(\frac{2^{32}}{2^{12}}\right) * 4B$

$= 2^{20} * 4B.$

$P.T_{INV} = \left(\frac{2^7 * 2^{10}}{4K}\right) * 4B$

$= \frac{2^{17}}{2^{12}} * 4B$

$= 2^5 * 4B.$

ratio (N : INV) $\Rightarrow \frac{P.T_N}{P.T_{INV}} \Rightarrow \frac{2^{20}*4}{2^5*4} \Rightarrow \frac{2^{15}}{1} = 2^{15}:1$

**Q-3**

Suppose that a certain computer with paged virtual memory has 4KB pages, A 32-bit byte Addressable virtual Address and 30 bit Byte Addressable physical address. The system manages an inverted page table. Where each entry includes The page number plus 12 overhead bits. How Big is the basic Inverted page table including number and overhead bits.

(a) $2^{10}$ Bytes (b) $2^{20}$ Bytes (c) $2^{30}$ Bytes (d) $2^{32}$ Bytes.

→ Page sige = 4KB P.A = 30 bit
L.A = 32 bits

P.T size = no of entrys * size of entry

$= \left(\frac{P.A.S}{\text{Size of frame}}\right) * \left(\overset{20\text{ bit}}{\text{Pg.NO}} + 12 \text{ bits}\right)$

$\frac{2^{32}}{2^{12}} \Rightarrow 2^{20}$

$= \left(\frac{2^{30}}{2^{12}}\right) * (20 \text{ bit} + 12 \text{ bit})$

$= 2^{18} * 32 \text{ bit}$

$= 2^{18} * 4 \text{ Bytes}.$

P.T size = $2^{20}$ Bytes

Q-4 Assume a machine with 64MB physical memory and 32 bit virtual Address. If the page size is 4KB, then what is the Approximate size of the page table?

(a) 16 MB (b) 1 MB (c) 24 MB (d) 2 MB ✓

→ P.T size = no of entrys * P.T entry size

(I) P.A.S = 64 MB | $2^6 * 2^{20} = 2^{26}$

(II) P.A = 26 bit |

(III) virtual address (V.A) = 32 bits

(IV) Page Size = 4KB = $2^{12}$ Byte.

no of entry per page = $\frac{2^{32}}{2^{12}} \Rightarrow 2^{20}$

P.T entry size = 14 bit (= 16 bit)
(↓ frame size)

no of frame = $\frac{P.A.S}{\text{Frame size}}$

$= \frac{2^{26}}{2^{12}}$

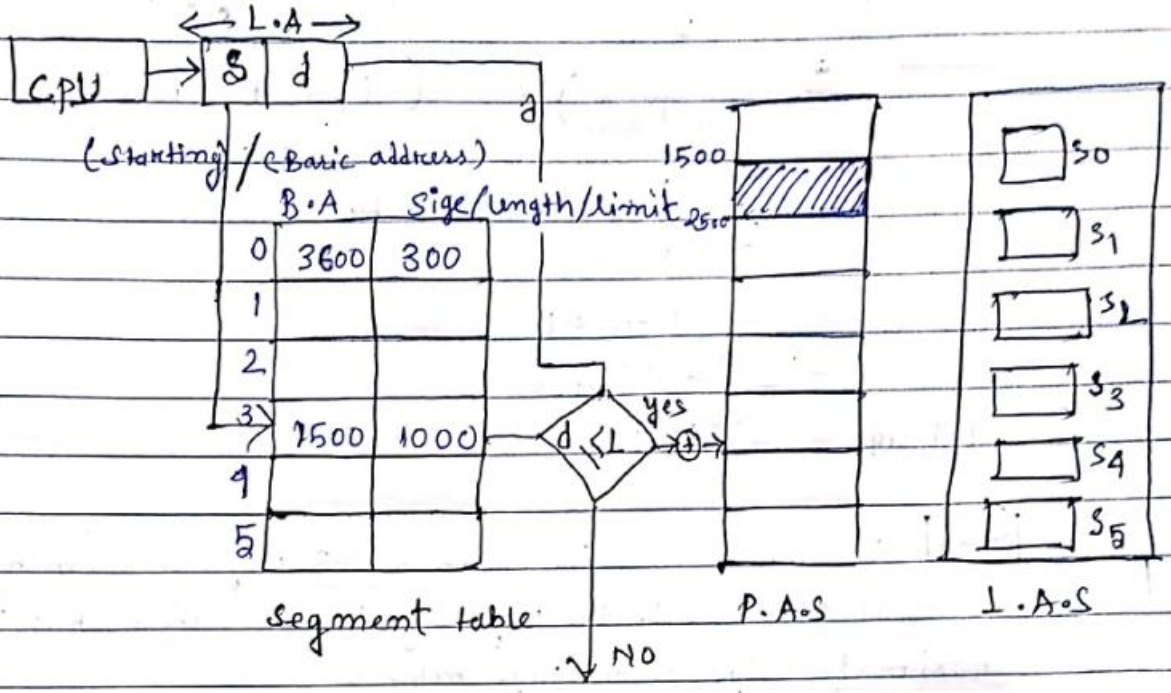
$= 2^{14}$

P.T size = $2^{20} * 16$ bit

$= 2^{20} * 2$ byte.

= 2 MB.

# SEGMENTATION & SEGMENTED PAGING:




- Some points :-

→ No of entries in segment table = No of segments in L.A.S.

→ The paging does not follow the users view of memory allocation.

→ To achive user's view of memory allocation, the ~~of~~ segmentation will be implemented.

→ In segmentation the L.A.S is devided into segments.

→ The segments will vary in Sige.

(S) → No of bits required to represent the segments of L.A.S or segment No.

(d) → segment offset No of bits required to represent segment Sige are word no of the segment.

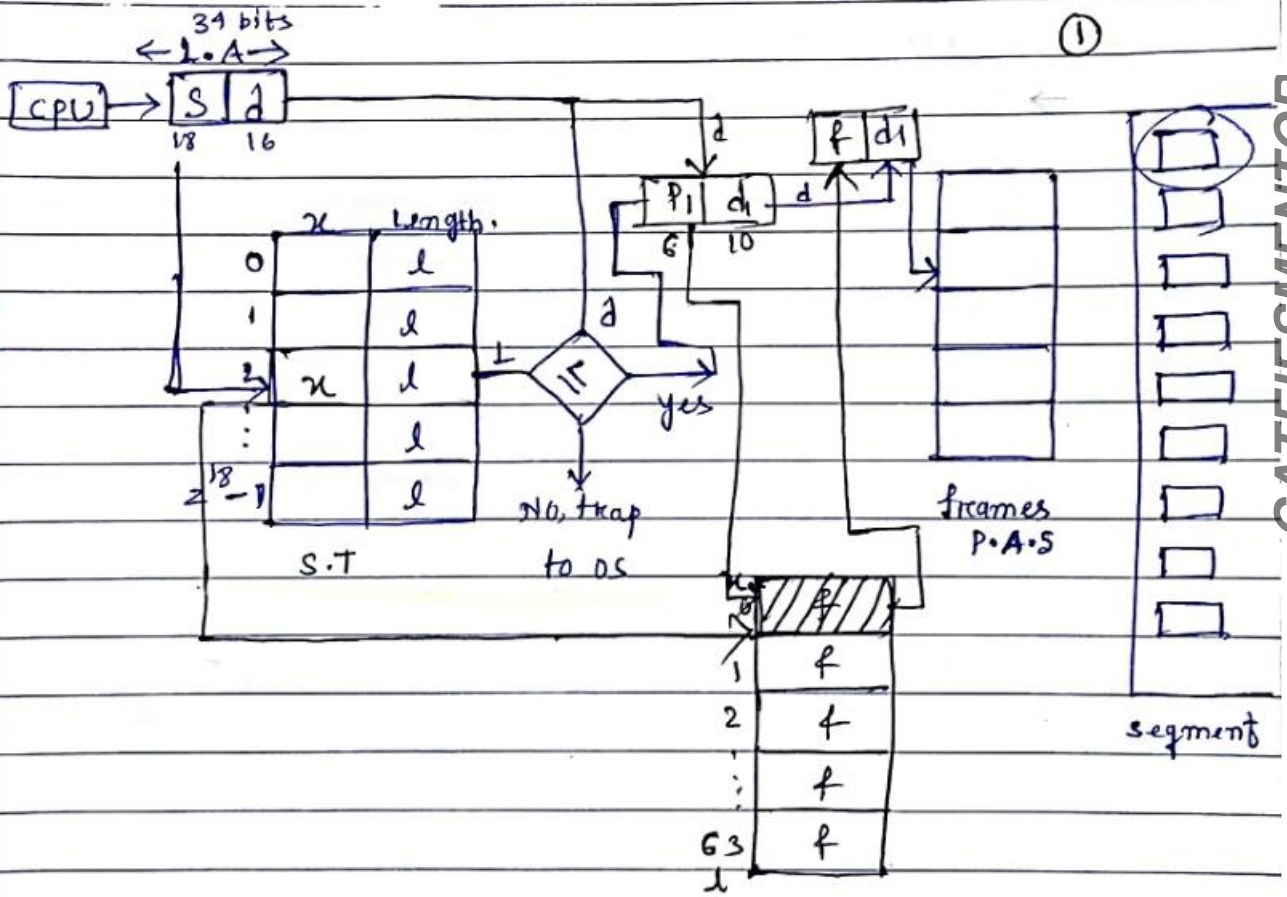
→ The variable size segments are brought into p.A.s so that the segmentation is just behaving like variable partition scheme.

⇓

Hence it suffers from External fragmentation

- paging on segmentation when segment size increase:

→ problems of external fragmentation and lengthy search times can be solve by paging the segments.




paging on segments

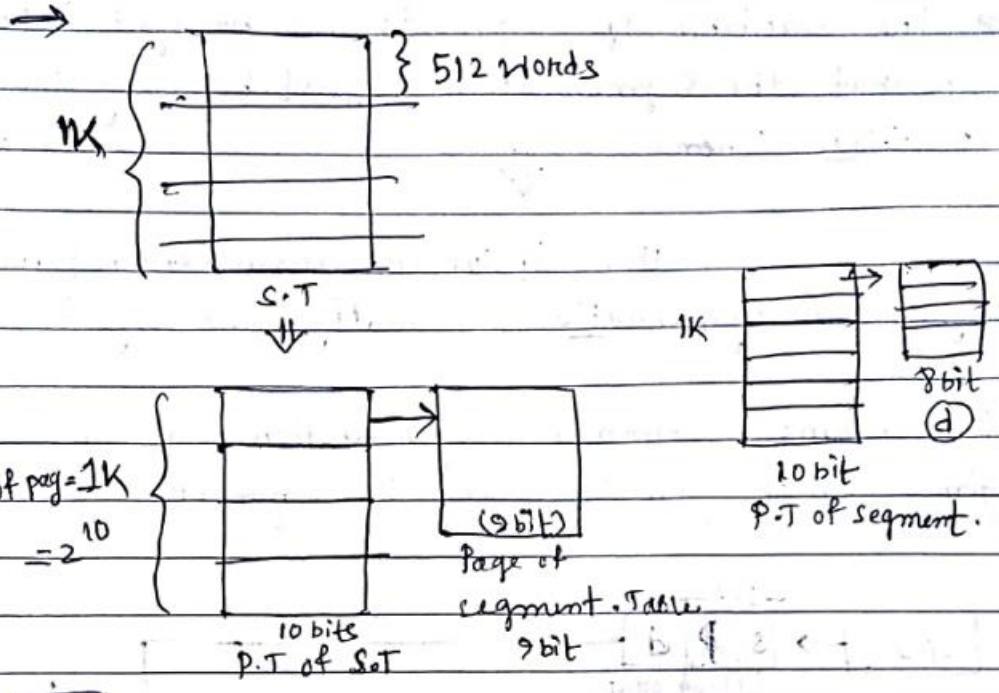
① calculate →

1) length of L.A. → (10+9+10+8) ⇒ 37 bit

2) length of p.A. → 18 + 8 → 26 bit

3) Page table size of segment → 1K * 4byte → 4KB.

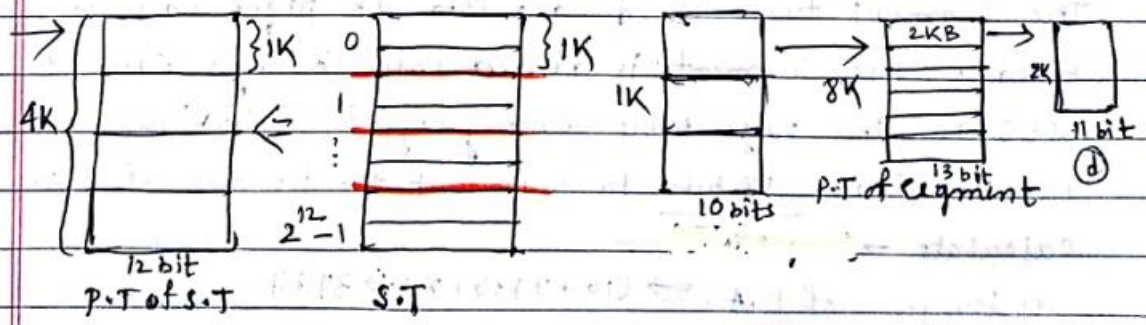
4) Page table size of segment table → 1K * 4B → 4KB




Q-2

Consider a system using segment paging Architecture. The segment is divided into 8K pages each of size 2KB and segment table is divided into 4K pages each of size 1KB. The memory is Byte Addressable. The P.A.S is 64 MB. The page table entry size is 32 bits. Then calculate -

1) Length of L.A. → (12+10+13+11) → 46 bit.
2) Length of P.A. → 26 bits.
3) page table size of segment → $2^{13}$ * 4byte → $2^{15}$ Byte → 32KB
4) page table size of S.T. → 4K * 4B → 16KB
5) The no of frames in P.A.S → $2^{15}$ Byte.




P.A.S = 64 MB
= $2^{26}$ B

no of frame = $\frac{\text{P.A.S}}{\text{Size of frame}}$

= $\frac{2^{26}\text{ B}}{2\text{KB}}$ => $2^{15}$

$\frac{2^{22}}{2^2\text{M}}$

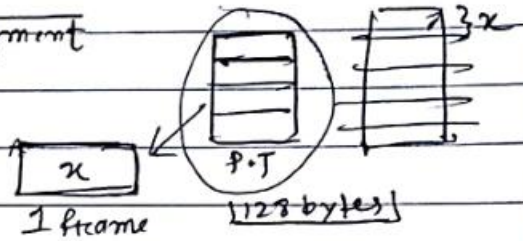
Q-3

Consider a system using segmented paging architecture where L.A.S = P.A.S = $2^{16}$ Bytes.

L.A.S is divided into 8 equal size segments. The segment is divided into equal size pages which are in the powers of 2 in term of page size. The memory is byte addressable. The page table entry size is 2 bytes. The page tables are stored in the main memory. What must be the page size of segment in Bytes so that the page table of the segment exactly fits in 1 page frame?

→ size of segment = $\frac{2^{16}}{8} = \frac{\text{L.A.S}}{\text{no of segment}}$

$= 2^{13}$ byte.




P.T size = $\left(\frac{2^{13}}{x}\right) * 2$ byte

[P.T size = no of entry (no of page) * size of entry]

$= \left(\frac{2^{13}}{x}\right) * 2$

$\Rightarrow \frac{2^{14}}{x} = x$

$\Rightarrow \frac{2^{14}}{2^Y} = 2^Y$

$\Rightarrow 2^{2Y} = 2^{14}$

$Y = 7$

$2^7 = 128$

GATEIESMENTOR

## VIRTUAL MEMORY:

Virtual Memory gives an illusion to the programmer that programs of largere sige than actual physical memory can be executed.

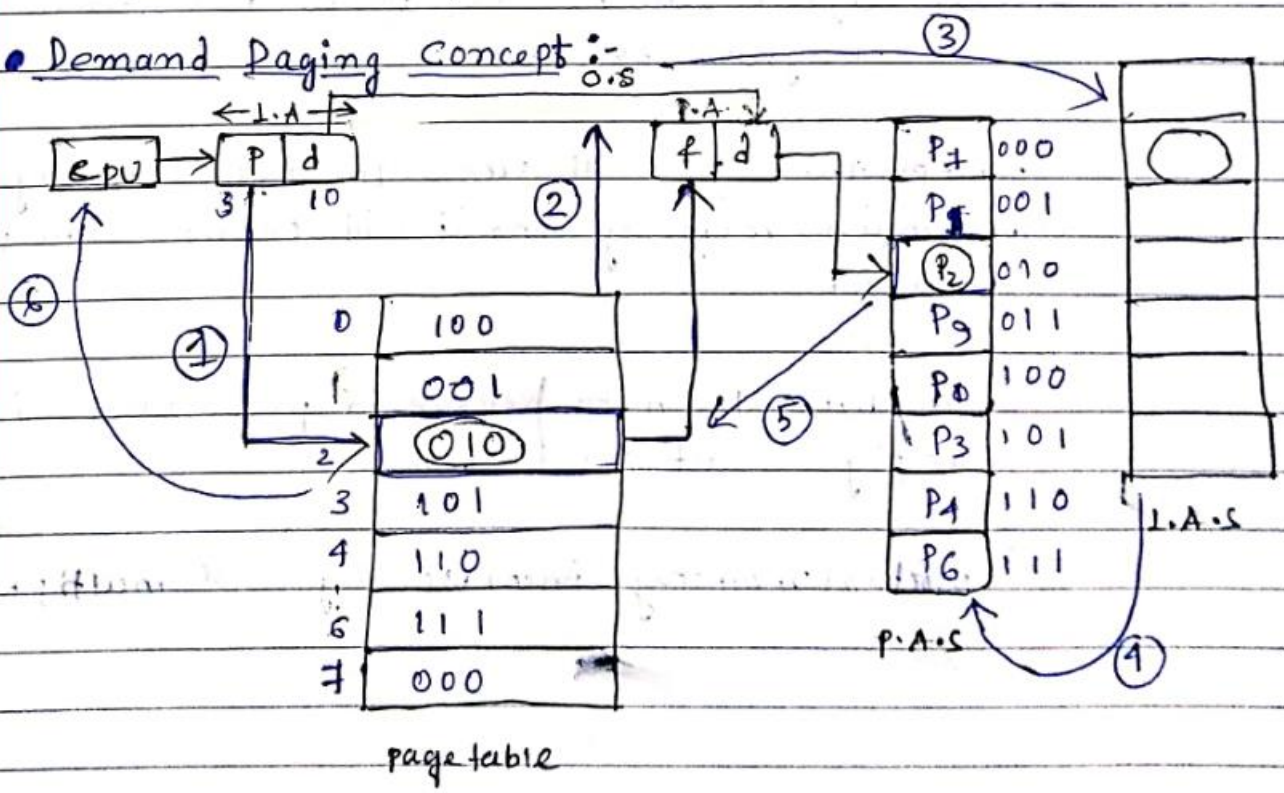
Example :- If program sige = 200KB
And Available P.A.S = 100KB.

- virtual Memory will be implemented using -
  1. Demand paging.
  2. Demand segmentation.

→ If we are using paging Technique as memory managemen-nt technique we use demand paging.

→ If we are using segmentation as Memory Management technique, we use demand segmentation.

- Demand Paging Concept :-



• Demand paging Conclusion :

Step 1) :- The CPU is trying to access a page which is not available in the main memory, hence page fault occures.

Step 2) :- The program execution will be stopped and the signal will be send to the O.S regarding the page fault. Then the O.S authenticates the validity of the process.

Step 3) :- Then O.S will be search for required page in the L.A.S.

Step 4) :- The required page will be brought from L.A.S to P.A.S, The page replacement Algorithms will be used, for decision making of replacing of page in the P.A.S.

Step 5) :- The page table will be uploaded accordingly.

Step 6) :- Signal will be sent to CPU to continue the program execution.

Then the CPU will access the required page from the main memory and it will continue the program execution.

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

→ Virtual memory increases degree of multiprogramming.

## • Demand paging performance:

→ Loading the page into memory on Demand.
(whenever the page fault occurs)

→ The page fault service time includes the time required to perform all the 6 steps.

→ If page fault service time = S
Main memory access time = M.
page fault rate = P.
Then $\boxed{EMAT = p*S + (1-p)M}$

As $S >> M$.
page table access time is negligible in comparision to S, so we ignore it for page fault.

Virtual memory — 8-1

Consider a system which has main memory access time = 10ns (m)
page fault service time = 200 ns (S). page hit ratio is 95% (P)
What is EMAT.

→ $\boxed{EMAT = p*S + (1-p)*m}$

(Page fault)
$p = 1 - 0.95$
$= 0.05$

$\Rightarrow 0.05 * 200 + (1-0.05) * 10$
~~$= 2 * 95 + 0.5 * 10 \Rightarrow 10 + 9.5$~~
~~$= 190 + 5$~~ EMAT $\Rightarrow 19.5$ ns.
~~$= 195$ ns.~~

* $1 ms = 10^6 ns$

**Q-2**

Suppose if an instruction takes i μseconds and additional j μsec if the page fault occures. What is the effective instruction Access time. If the page fault occures on an average for every K instruction.

(a) $i + \frac{K}{j}$ (b) $j + \frac{i}{K}$ (c) $K + \frac{i}{j}$ ✓ (d) $i + \frac{j}{K}$.

→ m ⇒ i —(1)

S ⇒ i+j μsec —(2)

$P \Rightarrow \frac{1}{K}$ —(3)

$EMAT = P * S + (1-P) * M$

$= \frac{1}{K}(i+j) + (1-\frac{1}{K}) * i$

$= \frac{(i+j)}{K} + \frac{(K-1)}{K} * i$

$= \frac{1}{K}(i+j+Ki-i)$

$= \frac{j}{K} + i$

$EMAT = i + \frac{j}{K}$

GATE-2011 **Q-3** The page fault service time is 10 ms and the main memory access time is 20 ns. If one page fault is generated for every $10^6$ memory access. Then what is the effective Access time for memory.

(a) 21 ns (b) ✓ 30 ns (c) 23 ns (d) 35 ns.

→ S ⇒ 10ms ⇒ $10^7$ ns.

m ⇒ 20ns.

$P = \frac{1}{10^6}$

$EMAT = P * S + (1-P) * m$

$= \frac{1}{10^6} * 10^7 + (1-\frac{1}{10^6}) * 20$

$= 10 + 20 - \frac{20}{10^6}$

$= 30 - \frac{20}{10^6} \Rightarrow 30 - 0$

$\approx 0 \Rightarrow 30.$

Q-4

Consider a demand paging environment and it takes 8ms to service a page fault, if either an empty frame is available or replaced page is not modified and it takes 20 ms, if the replaced page is modified. Assuming main memory access time is 1ms. Further Assume that page to be replaced is modified 70% of time. Then what is the maximum Acceptable page fault rate to get an EMAT not more than 2ms.

(a) P = 0.32 (c) P = 0.064

(b) P = 0.16 (d) P = 0.72.

→

S → 8 ms (pg modified) ✗ } 70% P = ?
20 ms (pg modified) ✓ }

m → 1 ms

E.MAT = P * S + (1-P) * m

≤ P * [(.70 * 8ms) + (.30 * 20ms)] + P(1-P) 1. ≤ 2

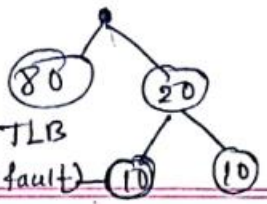
≤ 15.4P ≤ 1.

P ≤ 0.064.

Q-5 Consider a system where the page fault service time is 200 ms and main memory access time = 10 ms. The TLB is added to improve the performance. The 80% references are found in the TLB and that of remaining 10% cause page fault. The TLB access time and page table access time are negligible, then what is EMAT.

(a) 12.8 ms (b) 13.8 ms (c) 14.8 ms (d) 15.8 ms

→ S → 200 ms
m → 10 ms.




classmate
Date
Page 110

EMAT = $.80 * (10ms) + .20 [p*s + (1-p)*m]$

$= 8 + .20 [.1 * 200 + .9 * 10]$

$= 8 + 5.8$ ~~8.2~~ **13.8 ms**

## Page Replacement: (Algorithms) –

1) FIFO → First in First out.
2) Optimal page Replacement.
3) Least Recently Used (LRU).
4) Most Recently Used (MRU)

Page replacement FIFO –

**Question-1**

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

Assume 4 frames are 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 | X | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | 0 | 0 | 0 | 0 | 0 | X | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | X | 7 | 7 | 7 |
| | 7 | 7 | 7 | 7 | X | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | X | 2 | 2 | 2 | 2 | 2 | 2 |
| | * | * | * | * | | * | | * | | * | | | | * | * | | | * | | |

(page fault)

**10 page faults**

**Q-2**

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

Assume 3 frames are 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 |
| * | * | * | * | | * | * | * | * | * | * | | | * | * | | | * | * | * |

(Page fault)

total page fault – 15 ✓

Q-3

Reference String: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.

(1) Using 3 frames.

| 1 | 2 | 3 | 4 | 1 | 2 | 5 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 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 |
| * | * | * | * | * | * | * | | | * | * | |

total 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 |
| * | * | * | * | | | * | * | * | * | * | * |

total page fault – 10 ✓

- CONCLUTION –

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

3 frames → 9 page fault.

4 frames → 10 page fault.

## • BELADY'S ANAMOLY:

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

## • Page Replacement (LRU):

(Least Recently Used)

8-1

[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.

(1) using 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 | 2 | 4 | 4 | 4 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| * | * | * | * | | * | * | * | * | * | * | | | * | | | * | | | |

page fault → 12.

## • Page Replacement (Optimal PR):

8-1 [In the event of page fault replace the page 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.

(1) using 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 |
| | | 1 | 1 | 1 | 1 | 1 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 7 | 7 | 7 |
| | 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 |
| * | * | * | * | | * | | * | | | | | | * | | | * | | | |

FCFS

Total page fault → 8

- Page Replacement
- Most Recently Used (MRU):

In the event of page fault, replace the page which is most recently used.

Q-1

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

(i) Using 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) | 2 | 2 | (2) | 0 | 0 | 0 | 0 | 0 |
| | | 1 | 1 | 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 | 4 |
| 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| * | * | * | * | | * | * | * | | * | * | * | * | | | * | | | | |

Total page fault → 12

- Q- practice (Record)

The Address Sequence generated by tracing a particular program executing in a pure Demand paging system with 100 records per pages with 1 free memory frame is re-written as follows.

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

What is the number of page fault?

(a) 7 (b) 8 (c) 9 (d) 10.

| Records → | page no |
|---|---|
| 0–99 | 0 |
| 100–199 | 1 |
| 200–299 | 2 |
| 300–399 | 3 |
| 400–499 | 4 |
| 500–599 | 5 |

here frame → 1

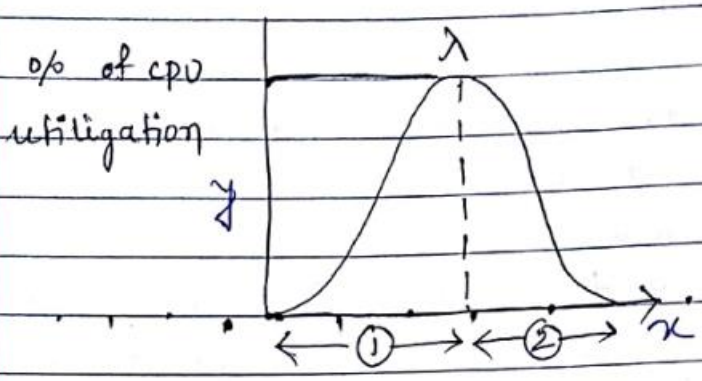
reference strings –

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

| 1 | 2 | 4 | 4 | 5 | 5 | 5 | 1 | 2 | 2 | 2 | 3 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 4 | 4 | 5 | 5 | 5 | 1 | 2 | 2 | 2 | 3 | 3 |
| * | * | * | | * | | | * | * | | | * | |

Total page fault → (7)

## • Thrashing Introduction:




→ In the initial degree of multiprogramming the cpu utiligation is very high and the system resources are utiliged. 100%.

→ If you further increase the degree of multiprogramming after some extent of point i.e λ the cpu utiligation will drastically fall down and time taken to complete every request will increase and system will spend more time only in page replacement. This is called Thrashing.

Example - Let free frames = 300.

Phase ① - # of process = 100 (3)

Phase ② - # of process = 300 (1)

## • Thrashing Causes And Recovery

### ⊛ Causes :-

① High degree of multiprogramming.

② Lack of frames.

### ⊛ Recovery from Thrashing :-

① Don't allow the system to go into thrashing. Instruct the long term shedular, not to bring process beyond the point of λ.

② Allow the system to go into thrashing, then insert the mid term shedular to suspend the processes. so that system may recover from thrashing.

- Thrashing practice Question – 1

Consider a system with below statistics observation.

1) CPU utiligation → 10 %.
2) paging on disk → 90 %. → (page replacement).

Then which one of following will increase the cpu utiligation=

1) Install the faster CPU → NO
2) Install the faster disk → NO
3) Increase degree of multiprogramming → NO
4) Decrease the degree of multiprogramming → yes.
5) Install more main memory → yes
6) Decreas the page sige → NO
7) Increase the page sige → yes.

GATEIESMENTOR

# FILE AND DEVICE MANEGEMENT

- File :→ collection of logically related entities.

Attributs :-

File context ⇓ FCB ↓ File Control Block

1) Name.
2) size.
3) type.
4) Permissions
5) Location.
6) Creation date.
7) Last Modified date.
8) owner / Author.
9) password.

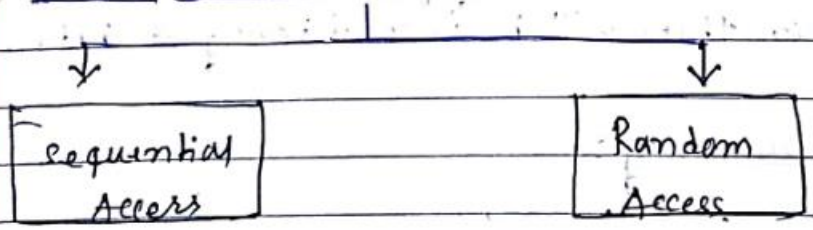
Types of files :-

1) .doc
2) .jpg
3) .pdf
4) .dll
5) .obj

- Operations on the file -

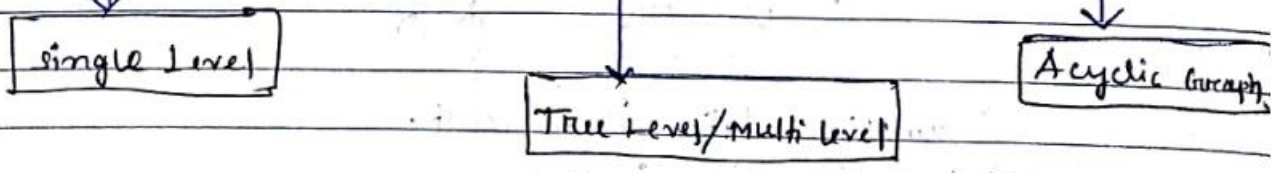
1) creat.
2) open.
3) read.
4) write.
5) update.
6) truncate.
7) scan
8) copy
9) delete
10) Rename
11) close.

- Access Methods of files -

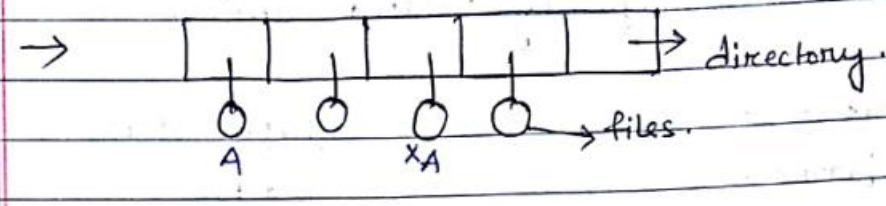



* For better classification of files they are stored in the Directories

Directory Structure



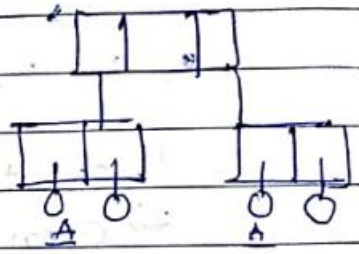
- **Single level Directory :-**




Advantage:- Implementation is Easy.
Disadvantage:-→ Searching time for a specific file will be more.
→ we cannot have two files of same name.

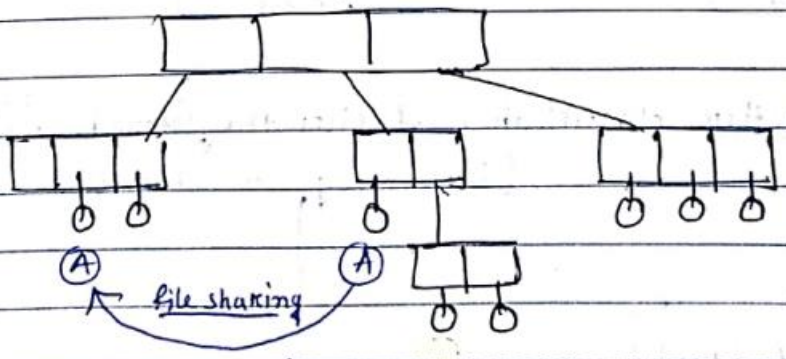
- **Tree level Directory :-**

Advantage:→ Searching time for a specific file will be less.
→ Better classification of files.




Disadvantage:
→ Implementation is difficult.
→ If same file entries in 2 different directories and if file is updated, the other file has to be updated accordingly. If other file is not updated, then there will be inconsistency of data.

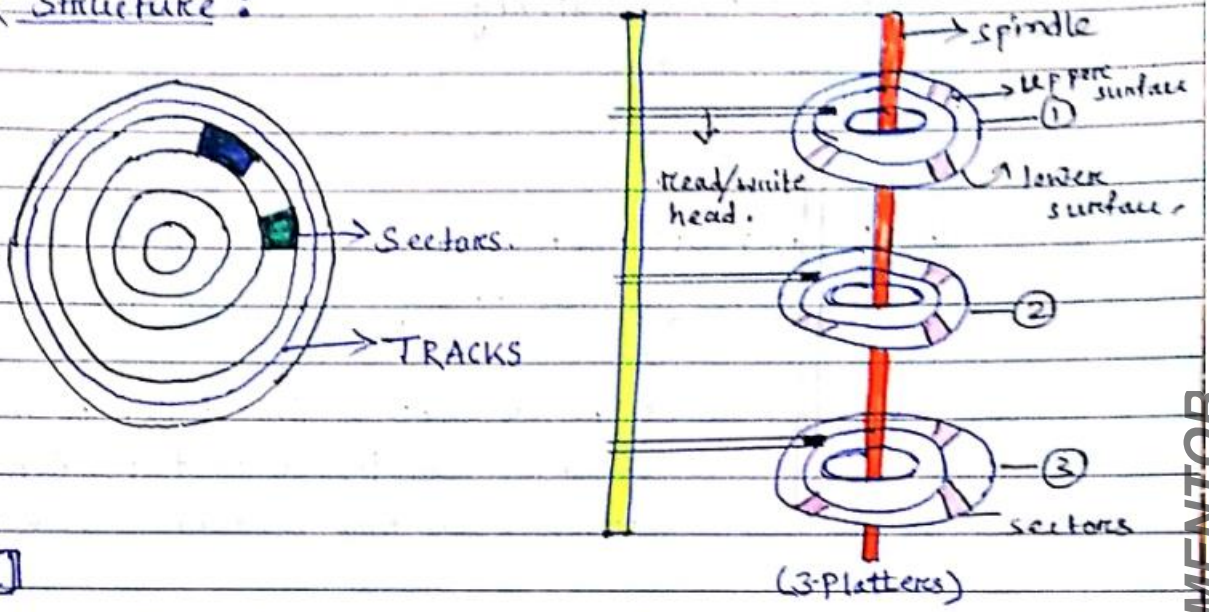
- **Acylic Graph Directory :-**




→ Implimentation is difficult.
→ Searching time is less.

→ If the same file exist in 2 different directories and 1 file is updated, then other file will be ~~up~~ updated automatically by using the concept of file sharing.

• Disk Structure :




Q-1

Consider a disc which has 16 platters. Each platters has 2 surfaces. Every surface is having 1K tracks. Each track is divided into 512 sectors. Every sector can store the data of 2KB.

calculate

1) Capacity of disk. (32 GB)

2) How many bits are required to identify a particular sector of a disk. (24 bit)

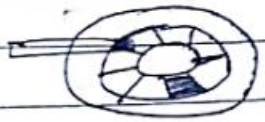
→ # of platters → 16
platters have 2 surface.
1K track in each surface.
track divided into 512 sector

(1) →
(16 * 2 * 1K * 512 * 2KB)
$\Rightarrow 2^4 * 2^1 * 2^{10} * 2^9 * 2^{11}$
$\Rightarrow 2^{35}$
$\Rightarrow$ 32 GB.

(2) →
Bits to identify platter. → 4 bit
+ Bits to identify surface → 1 bit
+ Bits to identify track → 10 bit
+ Bits to identify sector → 9 bit
→ 24 bit

- Disc Terminology:
  - DISK I/O Operation:

→ seek time
→ Rotational Latency.
→ Transfer time.
→ Transfer rate.



The Read/write header can never be outside the tracks. It will always point to any particular track of the surface.

- Seek time → The amount of time taken to move the R/W header from its current position to desided track. This time is called as seek time.

- Rotational Latency → The amount of time taken to rotate the R/W header on the track so that the header comes to exact position (sector). The rotational latency is considered as 1/2 of rotation time.

ie $\boxed{\text{Rotational latency} = \frac{1}{2} * \text{Rotation time}}$

- Transfer time → The amount of time required to transfer the required data, once the read/write header comes on the ~~desired~~ desired sector.

→ The transfer time depends on rotational rate of the disk and total size of the track.

- Transfer Rate → The no of bytes transfered per unit time is called as transfer rate of the disk.

**Q-1**

Consider a disk system which has an average seek time of 30 ms and rotational rate of 360 rpm. Each track of the disk has 512 sectors, each of size 512 B.

(i) What is time taken to read 4 successive sectors ?
(a) 0.32 sec (b) 0.16 sec (c) 0.084 sec (d) 0.52 sec.

(ii) What is the data transfer rate ?
(a) 1536 KBps (b) 1636 KBps (c) 1436 KBps (d) 1736 KBps.

(i) Solution = (seek time + Rotational latency + Transfer time)

= $30ms + \frac{1}{2} * \frac{1}{6} + 0.0013$ ($\frac{1}{2}$ * rotation time)

= $30 * 10^{-3}$ sec + 0.083 sec + 0.0013

= 0.084

1 m = 360 R

1 R = $\frac{60}{360}$ sec

1 R = $\frac{1}{6}$ sec

Transfer time logic = In one rotation time, we can transfer the total data of the track.

→ To transfer the required data how much time required

Total size of the track = (Sectors * Sector size)

= 512 * 512

= $2^{18}$

= 256 KB.

required data = 4 * Sector

= 4 * 512 B

= 2 KB.

256 KB → $\frac{1}{6}$ secs.

2 KB = $\frac{2}{6 * 256}$ sec

2 KB = 0.0013 secs

(ii) Data transfer rate : In one Rotation → we can transfer total size of track.

$\frac{1}{6}$ sec = 256 KB.

1 sec = 6 * 256 KB

= 1536 KB.

Q-2

Consider a disk having average seek time of 60ns and rotational rate of 3600 rpm. Each track of the disk has 512 sectors, each of size of 2KB.

(i) What is approx time required to read 1200 random sectors
(a) 10 sec (b) 20 sec (c) 30 sec (d) 40 sec.

(ii) What is effective data transfer rate.
(a) 30 MBPS (b) 40 MBps (c) 50 MBPS (d) 60 MBPS.

→ (1) Seek time = 60 ns
= $60 * 10^{-9}$ sec.

3600 R = 60 sec.
1R = $\frac{1}{60}$ sec.

(2) R.L = $\frac{1}{2}$ (R.T)
= $\frac{1}{2} * \frac{1}{60}$
= $\frac{1}{120}$

Total track size = 512 * 2KB
= $2^9 * 2^{11}$ B
= $2^{20}$ B

$2^{20}$ B = $\frac{1}{60}$

1B = $\frac{1}{60 * 2^{20}}$ (T.T)

2KB = $\frac{2^{11}}{60 * 2^{20}}$

2KB = $3.255 * 10^{-5}$ sec.

Required time to read 1200 random sectors —
(seek time + rotational latency + Transfer time) * 1200
= $(60 * 10^{-19}$ sec + $8.33 * 10^{-3}$ sec + $3.25 * 10^{-5}$ sec) * 1200
= 10sec Approx —— (i)

(ii) Effective data transfer rate —

$\frac{1}{60}$ sec = $2^{20}$ B (data transfer per second)

1 = $2^{20} * 60$ ⇒ 60 MBPS —— (ii)

Q-4

How long does it take to load 64KB of program from a disk whose 1 rotation time is 20 ms (R.T), and seek time is 30 ms and the total sige of the track is 32KB. The paging of is applied to the program and program is divided into page of sige 2KB. Assume that pages are spread randomly over the disc.

(a) 1120 ms (c) 1320 ms
(b) 1230 ms (d) 1450 ms

→ (1) Seek time = 30ms
(2) R.T.L = $\frac{1}{2}$ * (R.T)
= $\frac{1}{2}$ * 20
= 10 ms

# of pages 32
sige of the track = 32KB

20 ms → 32 KB
9 → 2KB (1 pg transfer time)
(1.25 ms) (1 page)

Total time = (30 ms + 10 ms + 1.25 ms) * 32
= 1320 ms

Q-5

How long does it take to load 128KB of program from a disk whose rotation time is 40 ms, and seek time = 60 ms. The total sige of the track is 64KB. The paging is applied on the program and it is divided into 128 pages. Assume that the pages are randomly spread over the disc.

(a) 30 sec (b) 20.4 sec (c) 10.25 sec (d) 40 sec.

→
(1) Seek time = 60 ms.
(2) Rotation time = 40 ms
(2) R.T.L = $\frac{1}{2}$ * 40
= 20 ms.
sige of page = 1 KB.

track sige = 64KB
64 KB = 40 ms
1KB = $\frac{40}{64}$ => 0.625 msec.

Total time = (60 ms + 20 ms + .625 ms) * 128
= 80.625 * 128
= 102.625 ms.
= 10.24 sec

## Disc last Question – 6

Consider a disc with below specifications.

1) No of surface → 8.
2) Inner diameter → 4 cm.
3) Outer diameter → 12 cm.
4) Inner track distance → 0.2 mm.
5) No of sector per tracks → 20.
6) Sector size → 4KB
7) Rotational rate of the disc → 3600 rpm.

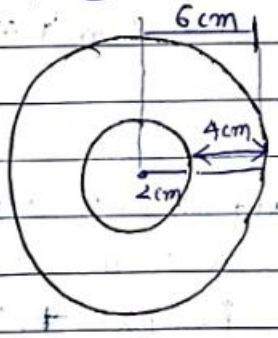
Qs-1 What is the capacity of the disk.

(a) 128 MB (b) 256MB (c) 512 MB (d) 1 GB.

Qs-2 what is the data transfer rate of the disc.

(a) 1800 KBPS (b) 2800 KBPS (c) 3800 KBPS (d) 4800 KBPS

→




No of tracks = $\frac{\text{Data Storage}}{\text{I.T.D}}$

$= \frac{4\text{ cm}}{0.2\text{ mm}}$

$= 200$

capacity of 1 ~~Disc~~ surface = 200 * 20 * 4KB

① capacity of 8 surface = 8 * 200 * 20 * 4KB

= 128 MB

② Data transfer rate –

60 sec = 3600 R

1 R = $\frac{1}{60}$ sec.

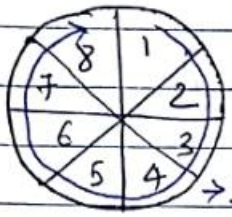
1 track = 80 KB

$\frac{1}{60}$ sec = 80 KB

1 sec = 80 * 60 R

= 4800 KBPS

## Disk Interleaving:

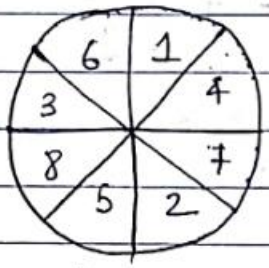



→ 1 Rotation need

No Interleaving


→ 2 Rotation need

Single Interleaving




→ 2.75

Double Interleaving

**Why Interleaving:**

Disc Rotation is very fast & hence it cannot detect (Read/write head)

**Q-1**

Consider the disc with double interleaving where track is divided into 8 sectors and seek time of the disc is 30ms and rotational rate of the disc is 3600 rpm. The sector size is 2KB and rotational latency is still considered as 1/2 of rotation time. Then calculate-

Q1) How much time is ~~used~~ required to read all 8 sectors of the track using double interleaving disk.

(a) 74ms (b) 84ms (c) 94ms (d) 104 ms

Q2) What is the time difference to read all 8 sectors of the track if the disk is non-interleaving disk.

(a) 19 ms (b) 55 ms (c) 29 ms (d) 37 ms

Q3) What is the data transfer rate if it is double interleaving disc-

(a) 249 KBPS (b) 349 KBPS (c) 149 KBPS (d) 449 KBPS

→ Q1)

2.75 R (Rotation need)

Seek time + R.L + Transfer time

30ms + 0.5R + + 2.75

= ~~88.25R~~

3.25 R

= 30 ms + 54 ms.

= 84 ms

3600R = 60 sec

1R = 1/60 sec.

3.25R = 3.25/60

= 0.054 sec

= 54 ms

→ Q2)

Total time = (seek time + R.L + T.T)

= 30 ms + 0.5R + 1R.

1.5R → 25ms

= 55 ms.

1R = 1/60 sec

1.5R = 1.5/60 sec

= 0.025 sec

= 25 ms.

84 ms − 55 ms = 29 ms time difference.

→ Q3)

2.75R * 60 sec / 3600 = 16KB

1 sec = 16KB * 60 sec / 2.75 R

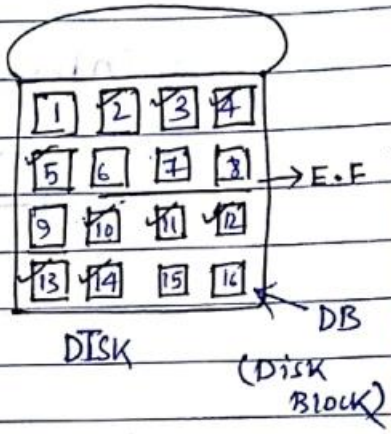
= 349 KBPS.

- Disk Space Allocation Methods:

1) Contiguous Allocation
2) Non-Contiguous Allocation / Linked Allocation.
3) Indexed allocation.

## (1) Contiguous Allocation =

| file | starting DBA | size |
|---|---|---|
| ABC.doc | 2 | 4 |
| XYZ.doc | 10 | 5 |




→ In contiguous Allocation disc blocks will be allocated in a contiguous manner.

→ Increasing the file size is not possible, always.

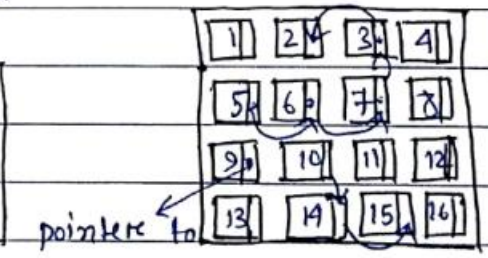
→ It is suffering from external fragmentation.

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

→ It supports sequential and Random Access of the file.

## (2) Non-Contiguous Allocation =

| file | starting DBA | ending DBA |
|---|---|---|
| abc.doc | 5 | 2 |
| xyz.doc | 10 | 15 |

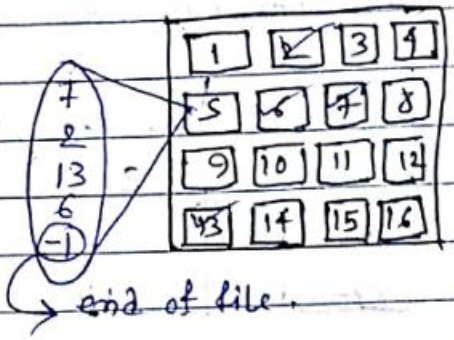



→ Increasing the file size is possible if free disc block is available.

→ No external fragmentation.
→ Internal fragmentation may exist in the last disc block of block.
→ There is overhead of maintaining the pointer in every disc block. It supports only sequential access of file.

③ Indexed Allocation –

| file | Index |
|---|---|
| abc.doc | 5 |



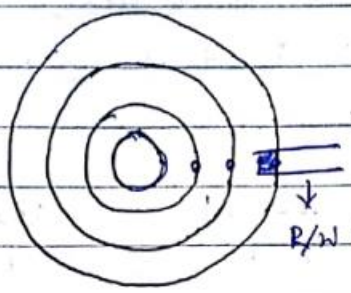

→ In Index Allocation 1 disc block is used just to maintain the disc block address of the file.

→ Every file is associated with its own Index node.

→ If the file is very large 1 disc block may not be sufficient to store disc block address of the file.

→ If the file is very very small, wastage of disk block to store the address to maintain the disc block address.

## DISC SCHEDULING:




GOAL ⟹ To minimize the Average seek time of the disc.

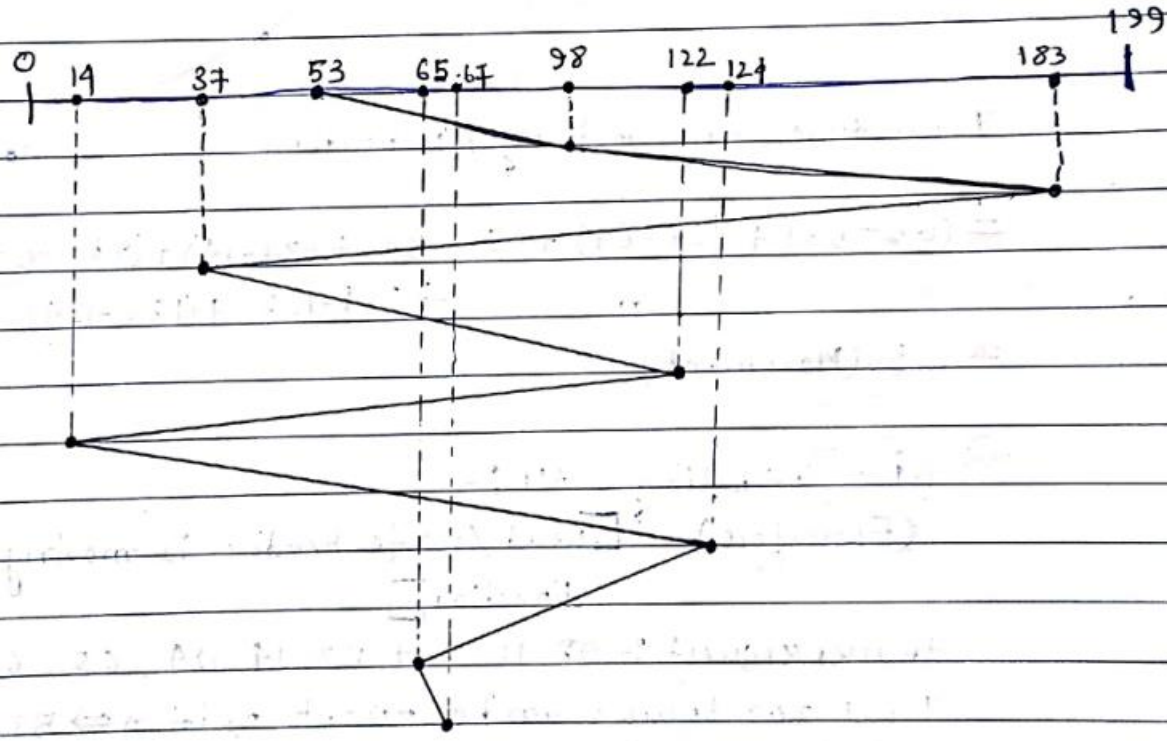
1) FCFS 2) SSTF
3) SCAN 4) C-SCAN
5) LOOK 6) C-LOOK.

### (1) FCFS:-

Track Request :- 98, 183, 37, 122, 14, 124, 65, 67.
(1) (2) (3) ..... (8)

surface of disc : 200 tracks. (first come first serve)

current track : 53.




total track movement by read/write header,

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

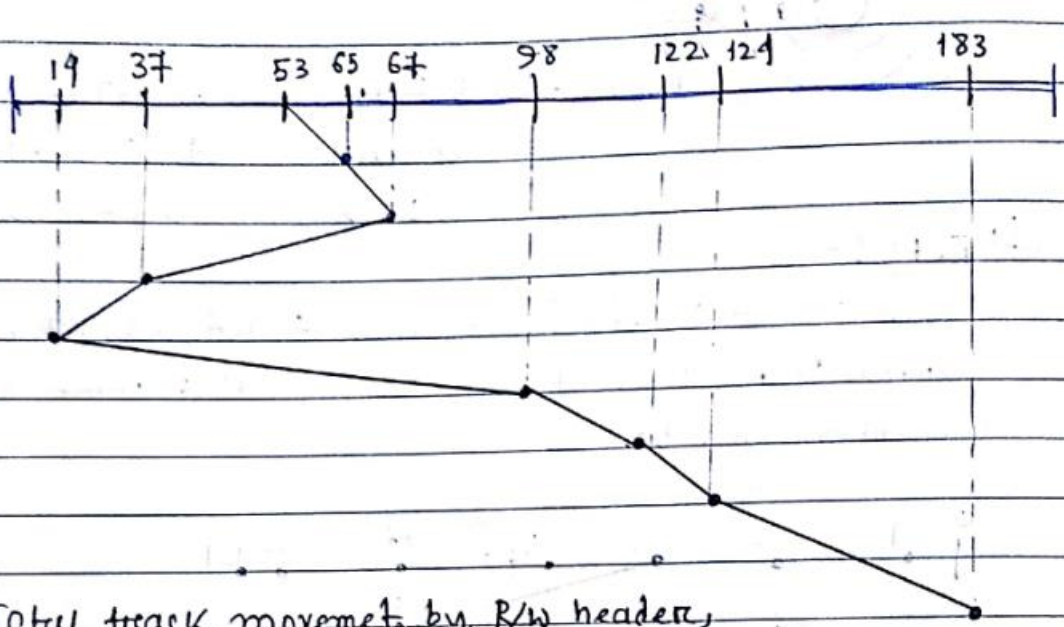
= 640

## ② Disc Scheduling SSTF :-

(Shortest Seek time first or Nearest track Next)

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

total 200 Tracks and current position => 53.




Total track movemet by R/W header,

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

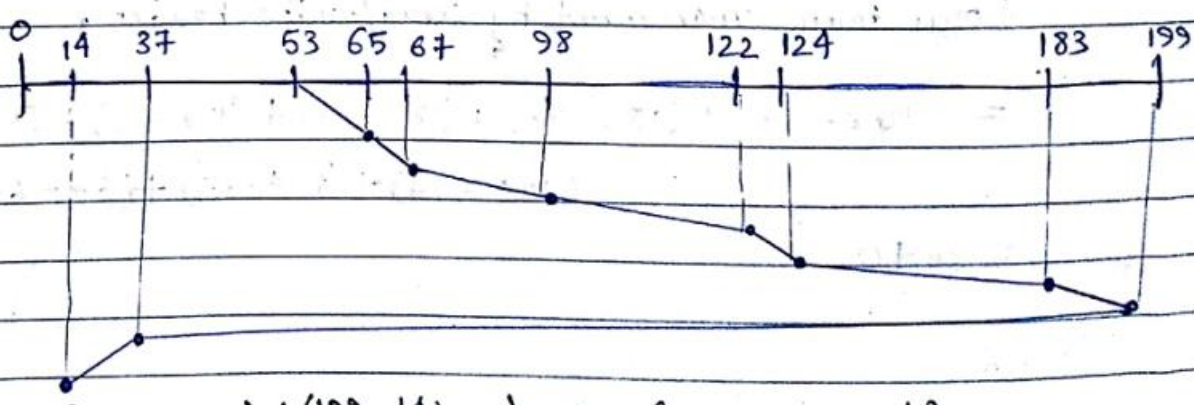
$= 236$ (Movement)

## ③ Disc Scheduling SCAN :-

(Elevator) [read/write header is moving towards right direction]

Tracks request :- 98, 183, 37, 122, 14, 124, 65, 67.

Total 200 tracks and current position => 53.

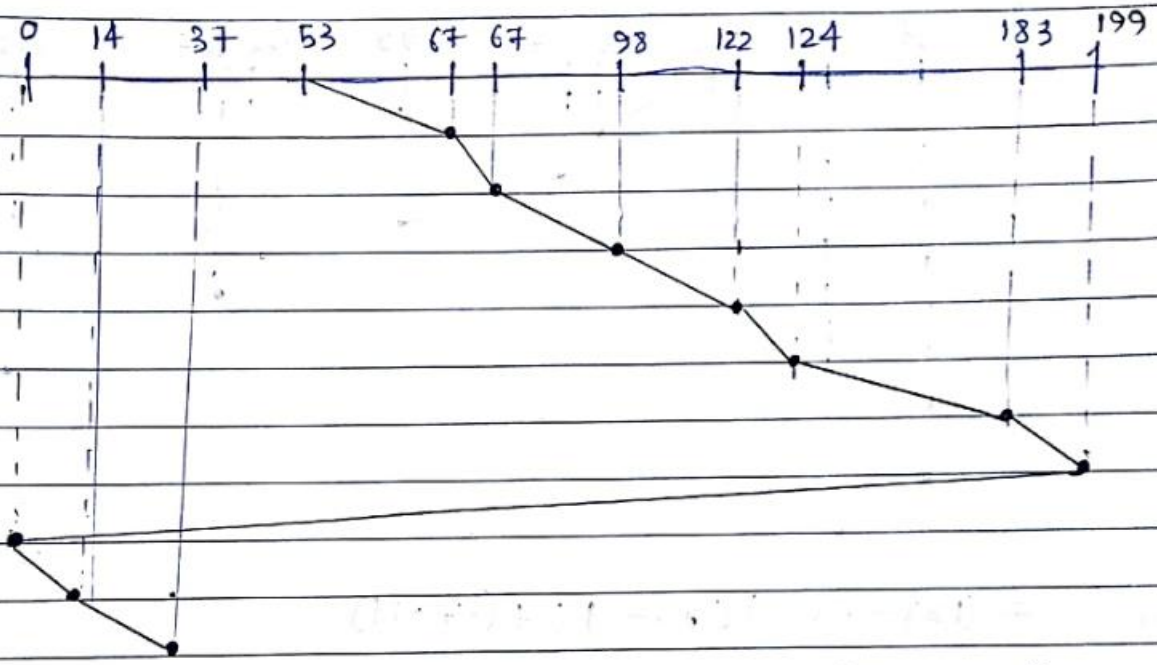



$= (199-53) + (199-14)$ => 331 (movement).

## (4) Disc Scheduling Circular-SCAN :- (C-SCAN)

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

Total 200 track request and current position ⇒ 53.




Total R/W head movement = (199 - 53) + (199 - 0) + (37 - 0)

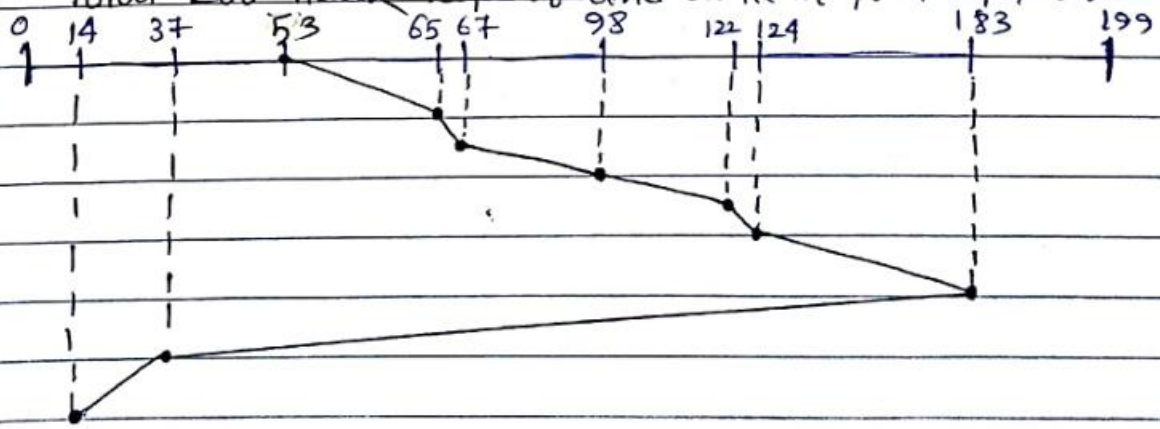
= 382 (Movement)

## (5) Disc scheduling LOOK :-

Look for the last pending request in the direction of r/w header.

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

Total 200 track request and current position ⇒ 53.




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

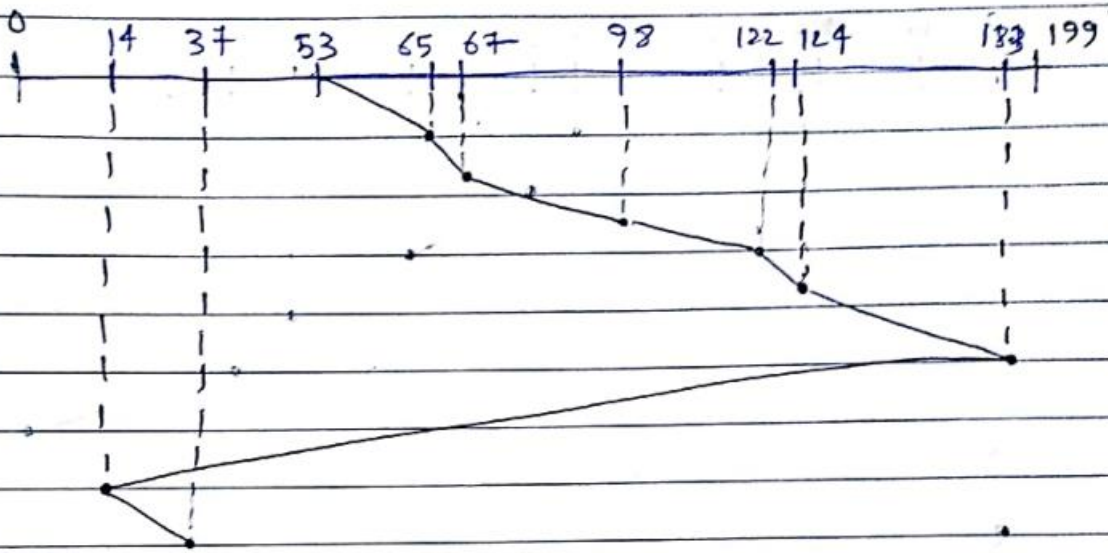
= 130 + 169.

= 299

## ⑥ Disc Scheduling Circular LOOK :-
## [C-LOOK]

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

Total tracks => 200 and current position => 53.




$= (183 - 53) + (183 - 14) + (37 - 14)$

$= 322$ (Movement)