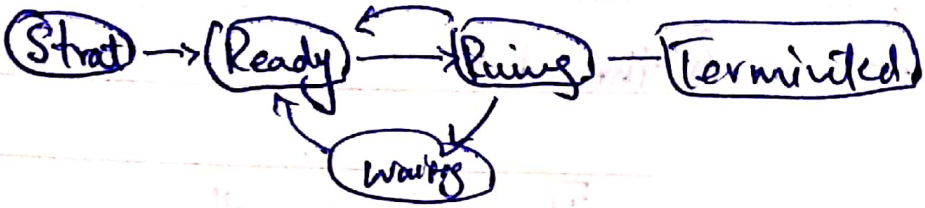
OS

Diff b/w process & threads

Process	Threads
Process is program in execution	Threads are the segment of program.
More exe time	Less exe time
Don't share res	Shares resource
P don't shares data	Threads shares data
P don't share memory	Threads shares Memory
Slower than threads	Faster than process.

Process



PCB

Data str in OS that stores info about P

Sockets

are software obj that establishes a bidirectional network communication blw client and server.

Sockets are strategies for communication. They are mostly used in client-server network. They acts at the end-point on either side for comm

Multicore programing:

is the development of concurrent systems for deployment on multicore processors. A multicore processor is a single processor containg multiple execution chips in it.

Multiprocess System

have multiple processor connected with mother board

Q Making Connection b/w user & Kernal thread

System call

is the programatic way to request a system service from a Kernel that is running in the OS

fork(), .write()/openFile()

Q Primary Thread Libraries

are the threads that are provided to the programer for creating and managing threads by using API application program interface

Example Win 32, Java, Pthreads.

Q CPU scheduling

is the process of allocating CPU to a process in ready state and switching b/w process on basis of diff sheduling algo.

The purpose is whenever its idle, OS switch the loaded prems and load the pr ready for exe in CPU.

Critical Section

is the segment of code of a process that is accessing a shared common variable and perform operation on it.

When one program is in its critical section, no other program is allowed to execute its C.R.

Cont

Section Problem :

A procedure or protocol that is used to overcome C.S.P. Each P. that wants to enter in its C.R check that if anyother process is in C.S.

Four entry section, exit section, remainder

Solution

Multiple Ex:

Progress,

Bound

Q Internal & External

Q Preemptive & Non Preemptive Kernel

It allows one process to be replaced by other process	It doesn't allows context switch
It is suitable for real-time Prograing	It is not suitable for real time programming
The process with high priority are given to CPU	Each process that is given to CPU continues its exe until its exe is Terminated
It allows preemption of process	It doesn't allows preemption of processes

Load Balancing:

is the core network sol that provides the distribution of client request to different servers in a server. Frame.

This allows the availabity of application and responsiveness the the client request and prevents overloading

Longs

Critical Section

Def critical section

Def critical section-problem

↳ developing rules and protocols that are followed by a process in which it makes a request for every parallel program in execution to access critical section

Race condition This request is entertained by entry section which is the entry point of critical section

Entry Section consist of implementation of such code that if critical section is already acquired by some other process, the request for access is denied by entry section

After entry section, there is code for access shared variables or resource.

Exit Section is followed by entry section. this section allows the exit of process from critical section, make free the available resources and informs all other processes that the critical section is free

Remainder Section

the execution of code after critical section is remainder section.

Solution to the critical section problem must satisfy

Mutual Exclusion

Progress

Bounded Waiting

Some Solution to the critical Section Problem

Most of the solution utilizes lock implementation.

➡ Mutex Lock:

provides acquire and release function that is automatically executed

➡ Semaphore:

semphores utilize 2 basic function wait() and signal() that is automatically executed on Semaphore S, an integer variable.

➡ Conditional Variables: utilizes a sequence of processes that are waiting to enter in critical section.

General structure

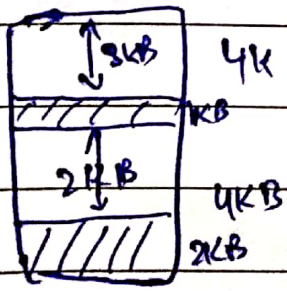
```
do {
    Entry Section
    C.S
    Exit Sec
    R.S
} while(true);
```

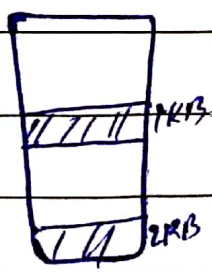
Example

		R_1	R_2	C
T_0	$R_1 = C$	5		5
T_1	$R_1 = C+1$ $R_2 = C$	6	5	5
T_2	$C = R_1$ $R = C-1$	6	4	6

Q Internal & External Fragmentation

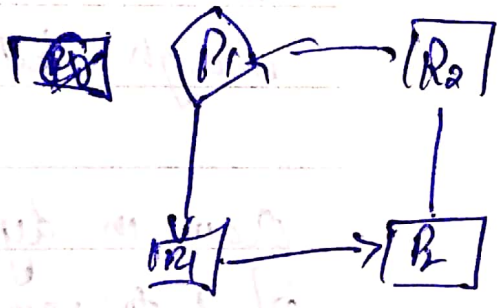
Internal Frag	External Frag
Def	Def
occurs when P is allocated to greater memory block	when available size is enough but not allocated
occurs in fixed partitioning of M	Occurs in dynamic allocation of P to variable size.
can be handeled by variable/dynamic partition	can be handeled by contention and pegin
Memory Block is greater the P size	Avaiable M enough but cannot be allocated





Q: Deadlock

is the situation in which a number of processes are blocked because each process has been acquiring / holding a resource and waiting for other resources which has acquired by other process

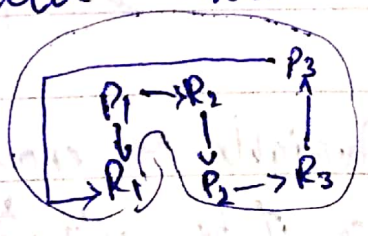


Mutual Exclusion => Non Sharable Resource

Non-Preemption =>

Hold & wait =>

Circular wait =>



Deadlock	Starvation
DeQ	DoQ
wait is infinite	wait is long but not infinite
Every deadlock is starv	Every starv is not deadlock
Resource are blocked	CPU continously assigned to higher P

Methods to Handle Deadlock

(i) Deadlock Avoidance (ostrige algo)
(ii) Deadlock Prevention (Donot let 4 states/con to occur)
(iii) Deadlock Avoidance (Banker's Algo)
(iv) Deadlock Detection & Recovery

Deadlock Avoidance

=> For every resource allocation, we check for safe & unsafe sequence.
=> By using Banker's Algo (Dijikstra's Algo)

Resource Allocation Graph

Graphically rep of allocation of resources

4 components

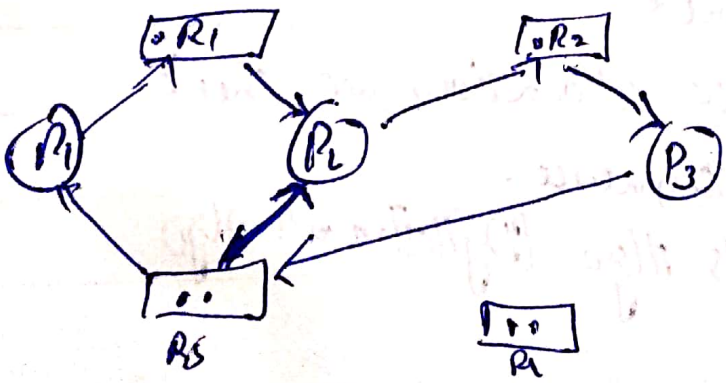
1. ○ → processes ✓
2. □ → Resources ✓
   - [•] → Non-Sharable
   - [••] → Sharable, •• shows no of instances
3. R '—>' P → R allocated to P ✓
4. P —> R → R requested by P ✓

Can find if Deadlock occur or not?

Ex

∴ No cycle ⇒ deadlock never occur

∴ Cycle ⇒ may or may not deadlock occur



Banker's Algo

Banker's Algo checks for the safe or unsafe state before allocating resource to a process.

Why it is named

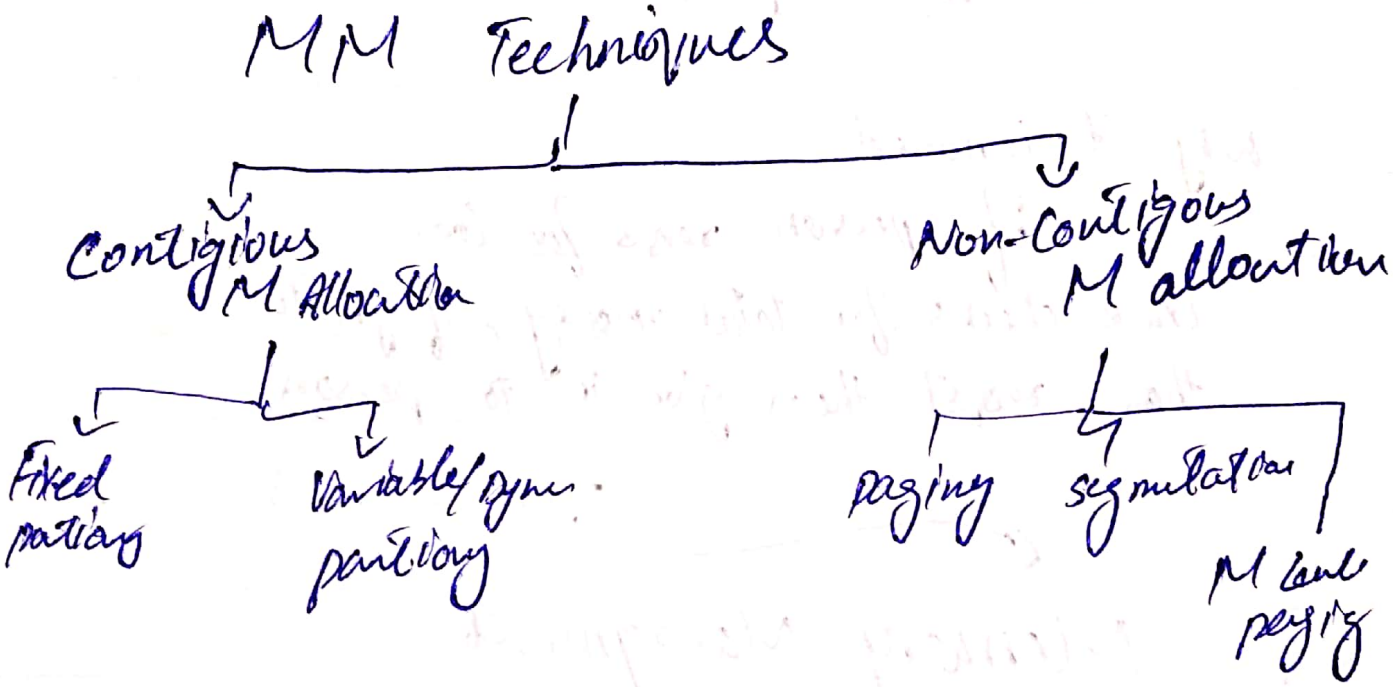
if person reqs for loan
Bank checks for total money, if greater than reqst then give it to person.

Memory Managment

=> is the type of resource menegmnt.
=> handles the swapping of pages from main memory to secondary storage
=> increase efficiency

Base Reg = Smallest legal unit of physical addrs
Limit Reg = size of address or range

CPU direct access to Main Memory
Process are brig to Main Memory from secondary storage
CPU access registers, caches and MM



Static Loading
all the libraries and modules are loaded into memory for the exe to start.

Dynamic loading
all the libaries, modules and data are loaded into memory when req by the program.

Static Linking

the linker combines all the data, libraries and program objects into a single executable program to avoid runtime execution

∴ The linker combines the object modules with object program and convert them into absolute executable program.

Dynamic Linking

In dynamic linking, instead of linking actual modules and libraries, a reference is linked with the dynamic module at the time of compilation

Swapping

is the process of moving a process out of Main and storing it to Secondary & vice versa.
Swapping is also called Technique for memory compaction.
To run large program by swapping pages.

6- Memory Management Techniques

Fixed - Partitioning
Variable Partioning
} Contigious Memory Allo

Paging
Multilvl Paging
Segmentation
} Non-Contigious M A

Contigious Memory Allocation

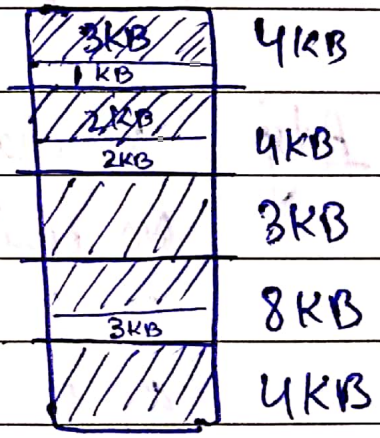
allows you to store the process in a contigious fashion

=> Fixed partioning

in fixed partioning

- No of partitions are fixed
- size of partitions are may or may not be fixed
- Spanning is not allowed in contigous M A

∴ Spanning is the process of dividing process in parts



Limitations

=> Internal Fragmentation
=> Limit of process size
=> Limit in degree of Multiprograming due to fixed no of partitions.
=> External Frag

Variable Partitioning

partitions are made when need by process instead of during system configuration

=> Divide ram into parts variably on run time

Adv : No chance of internal Frag
No limit for no of partitions
" " " size of "
" " " no of processes.

Dis Adv

External Frag

Non-Contigous Memory Allocation

=> processes are allocated in different portions of main memory in non-contigous form

=> reduces the waste of memory but increases the time for context switching and translation of logical address.

=> Page Table is used to maintain the pages of programs allocated to frames of MM

=> **Paging**

is the process of dividing process into parts.

=> Program is first divided into parts before bringing them to MM.

=> Main Memory is also divided into frames

=> Size of page = Size of Frame.

=> Removes External Fregmt

=> Logical address
generated by CPU for program

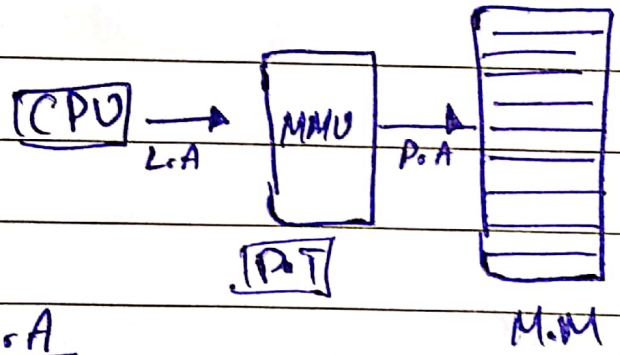
=> Physical Address
are actual addresses of Memory

=> MMU is responsible for conversion of logical address into physical address

=> MMU uses page table for this purpose

=> Page Table contains the page no and page offset

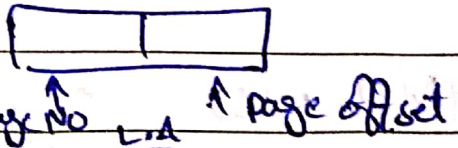
PageNO	Page offset
P_2	0
P_1	1
P_3	2
P_5	3
P_4	4



L.A

Page NO : bits represents the no of pages of program

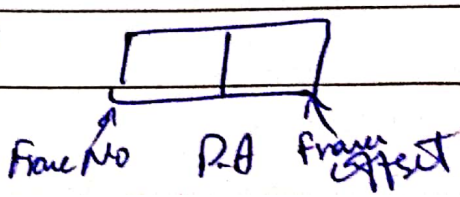
Page offset : bits represent the word or page size



P.A

Frame NO : Bits rep no of frame

Frame offset : Bits rep page offset



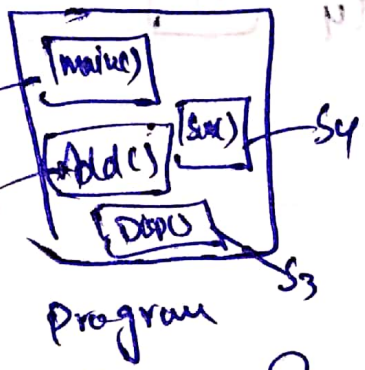
Page Rep Algos

=> **Segmentation**

is the process of dividing program in parts and allocating them in MM

It is similar to paging, instead it divides parts of program, Paging divides the codes, It divides the functionality

=>



=> Segments are of diff size

=> pages are of same size.

=> Logial $\xrightarrow{MMU}$ Physical

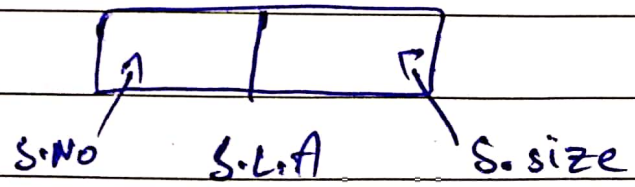
=> MMU uses segment tabl

Segment Table

No of segment	B.A	Size
0	3300	200
1	1800	400
2	2700	800
3	2200	700

∵ Base Address = starting address of S
∵ Size = size of segment

CPU ⟶ L.A



Components of a computer system

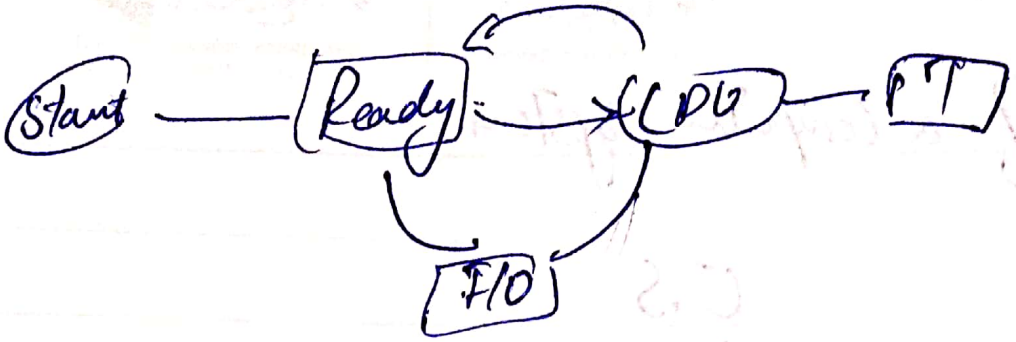
Hardware — O.S
Application Software — Endusers

Services)
User Interface (
Program Execution)
I/O operation (
Resource Management)
Allocation of R (
Communication (
Privacy)
Security (

Computing Enviro
Personal Computer
Time Sharing
Client-Server
Distributed
Cloud based

System calls

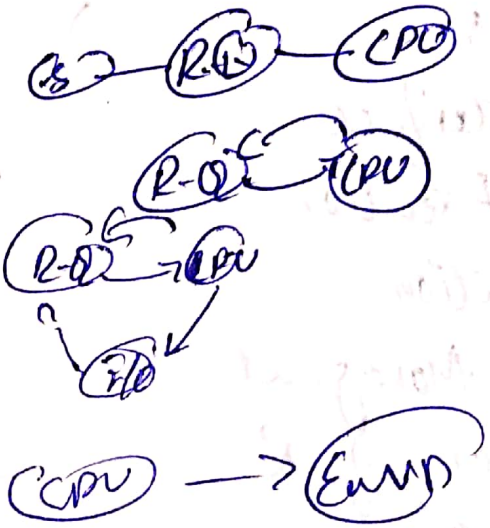
Process Control — CreateProcess() ExitProcess()
File Management — CreateFile() RdFile() WrFile() ClFile()
Device Management — ReadConsole(), WriteConsole
Info " — GetProcessID(), setTime()
Communication — CreatePipe()



Q Explain operation on process

Process Creation

- " preemption
- " blocked
- " Terminal



Thread

Many to One — Multiple lvls mapped to single kernel thread.

May to Many — Thread Magmt is done by thread library but are blocked by making a blocking system call

One to One