COS831- Operating System II

Course Contents

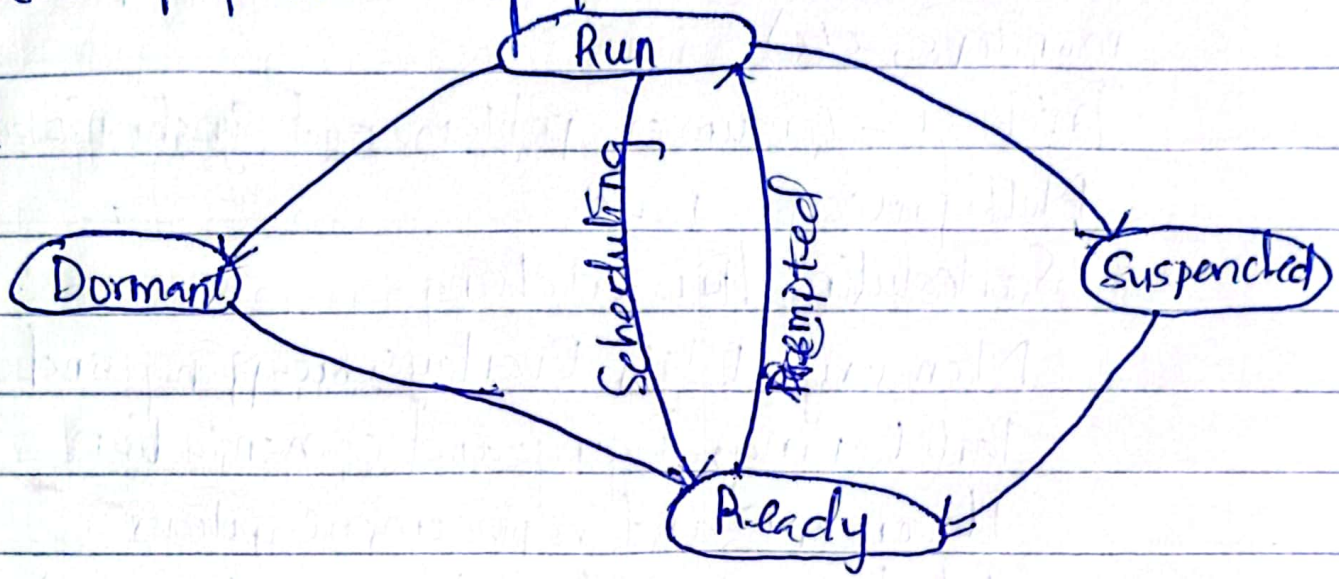
Concurency, States and State diagram, Structures
Dispatching and Context switching, interrupts
Execution, Mutual exclusion
problem an some solutions.
Deadlock; Models and mechanisms (semaphores, monitors, etc).
Producer - Consumer problem and Synchronization
Multiprocessor issues
Scheduling / Dispatching
Memory Mgt, Overlays, Swapping and
Partitioning, Paging and segmentation
Placement and replacement policies
Working Set, and trashing, caching

Concurrency

Concurrency usually happens in ~~the~~ a unit processor system. Means executing multiple task at the same time but its not necessarily simultaneous.

In a single core system concurrency can be achieved through a process called context switching.

Tasks are performed using process state transition method



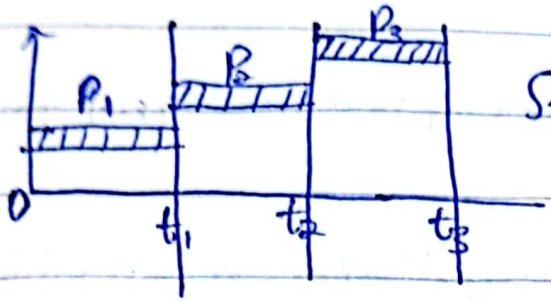
In a multicore environment concurrency takes place through parallelism

∴ Note concurrency relates to ~~single core~~ through context switching whereas parallelism takes place in a multicore processor.

In which case multiple task are executed simultaneously

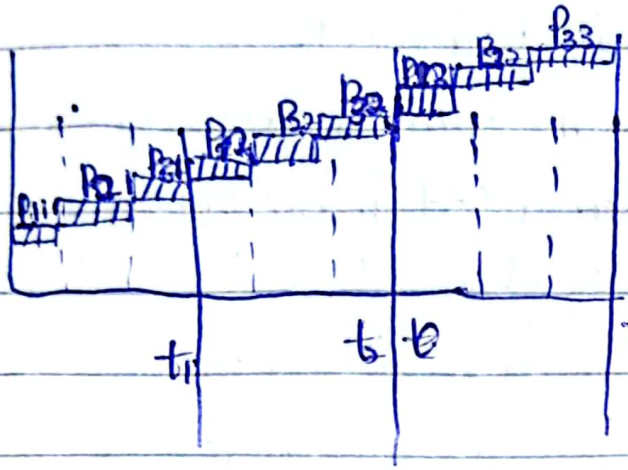
Can one say that a timesharing system is concurrent.

Draw a diagram to show the difference blw concurency and Parallelism

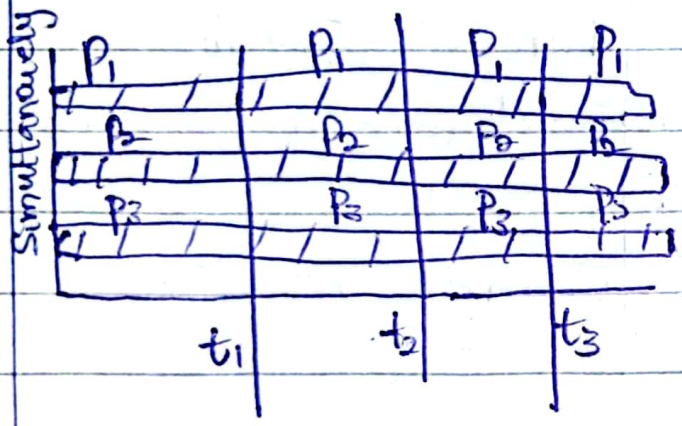


Serial processing
Outdated

Explain the difference blw Parallelism and Concurrence



Concurrent processing
Stimulated or Virtual parallel



True parallel Processing

Process States and State diagrams

process concept: A process is a program in execution. A process is a dynamic entity, while a progam is static. processes exist in a limited time span, while the program that spills out the processes remains.

A process has Object program, data, resources and status of the process on execution.
A process is not a closed system, there is a communication blw processes through a shared enviroment.
A process can be defined as an entity representing the basic unit of work to be implemented in the system.
Resources can be classified into physical and logical resources

Difference between process and program

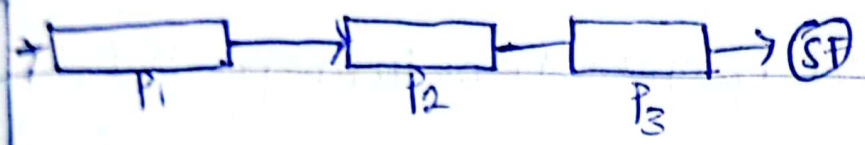
	Program	Process
i	Static entity	Dynamic Entity
ii	Exist in a single state and continue to exist	Exist in a limited time span
iii	Programs contain Instructions	Execute Instructions.

Types of Processes

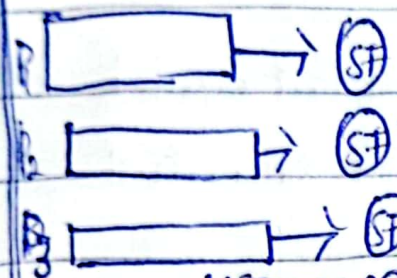
i Sequential processes
ii Concurrent processes parallel processes Concurrent process

In Sequential processes are executed one after another
In parallel two or more process are executed simultaneously

SF → Service facilities



Sequential



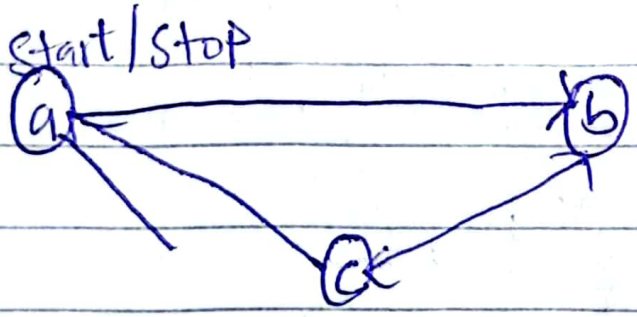
~~Concus~~ Parallelism

multiprocessing (Parallelism)

multiprograming (context switching)

A state Diagram

A State diagram is used in computer science and related fields to describe the behaviour of systems



We use a directed graph to show different state of the machine.

A classic form of state diagram for a finite automaton is a directed graph with the following elements $(Q, \Sigma, Z, \delta, \%, F)$

Q - Vertices, a finite set of states normally represented by circles

Σ → Input symbols - a finite collection input symbols

Z → Output Symbols - a finite collection of output symbol

δ → Edges

q_0 → Initial State

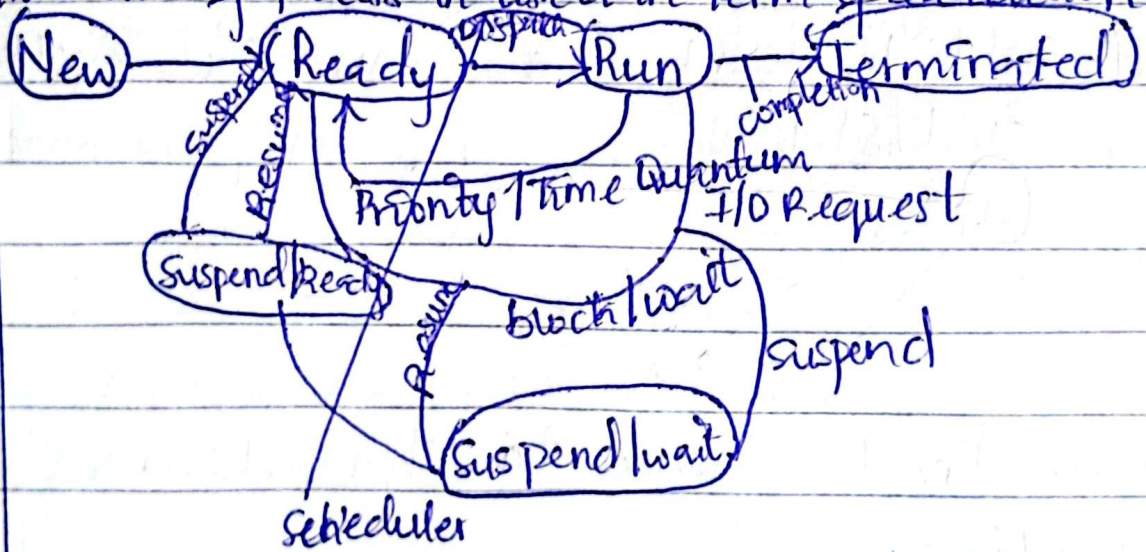
F → Final state.

State Diagram of OS

A program is an orded set of instruction - programmer's POV

A program is an executable file — O/S's POV

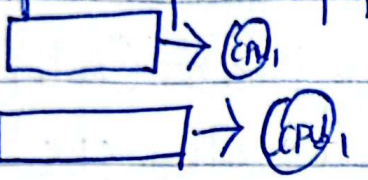
The notion of process in used in term of execution.



COSC 331

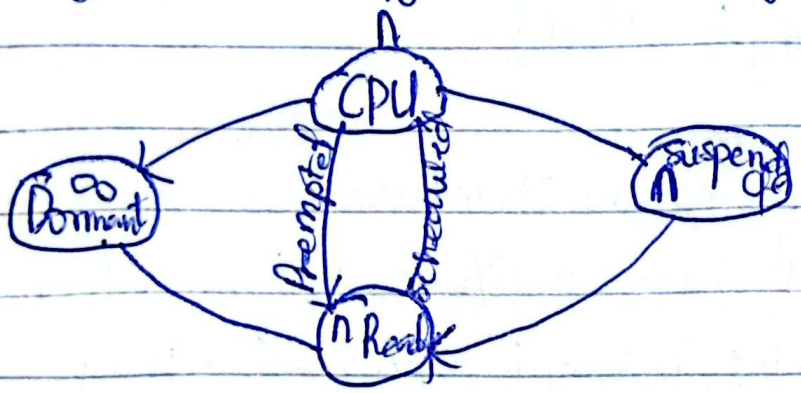
New state

Ready state
Running state
Block / wait / suspended state
Completion / termination / Dormant
Complete parallel processes



} it's called a multiprocessor system.

Draw this state diagram for a multiprocessor system.
The thing is to modify the number of CPU and queue to n



} Process lifecycle the stages in which a process must go through

Designing computer with no particular user
The running state is the state where the CPU starts executing the instructions of that system.
Each process must have an identity. There's a datastructure to store the information about a process. PCB and TCB (Process control Block)

In terms of PCB and TCB describe a Process

PCB	TCB
Process ID	Thread ID
Process Privileges	Stack Pointer (points to thread stack in the process)
Pointer	Program counter (points to current instruction)
Process State	Registers (thread CPU registers)
CPU Registers	Points to Originating process (PCB)
CPU scheduling inform...	Thread state (runs, ready, etc)
Memory mgt information	Thread priority
Accounting Information	pointer (other threads that were created together)
I/O states information	

Operating System Structures

Is a blueprint on how an O/S is organized and how the different parts interact with each other

Monolithic

Micro-kernel Client-Server

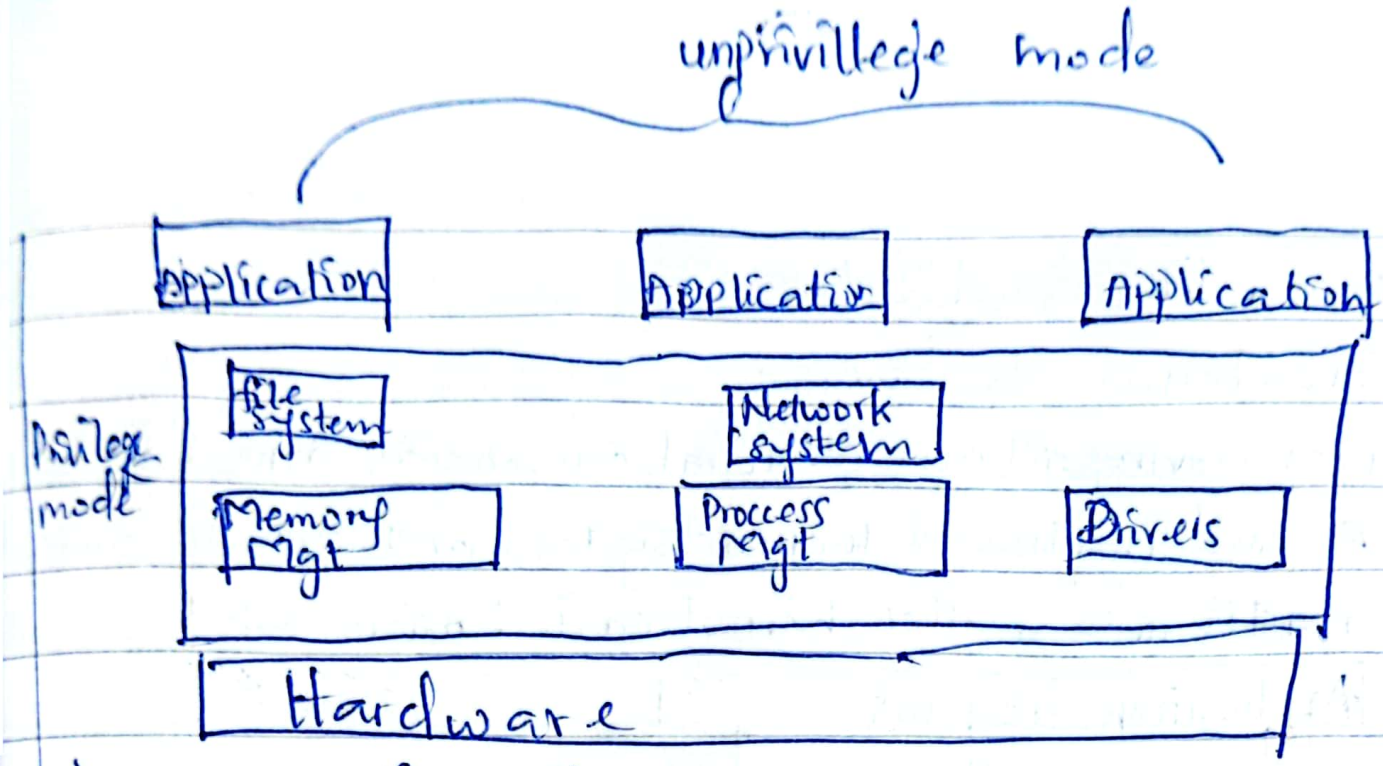
Hybrid-kernel

Exo-kernel

layered

Modular

Virtual Machine



2) In a monolithic structure

In a monolithic OS, the entire OS is implemented as a single large process in the kernel mode.

Essentially, OS services such as process mgt., memory input, block filesystems and device drivers are combined into a single code.

Advantages

(1) Performance of monolithic structure is fast since everything runs in a single block, therefore communication between components is quick

(2) It is easier to build because all parts are in one code block

Disadvantages

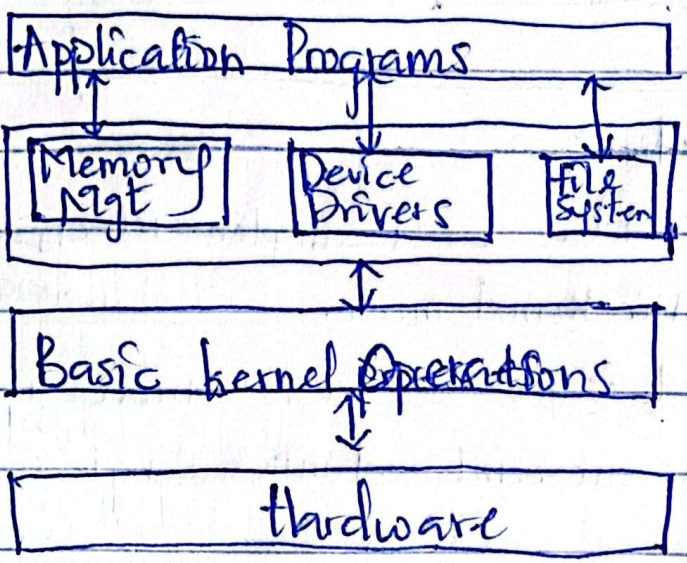
(1) It is hard to maintain as small error can affect the entire system

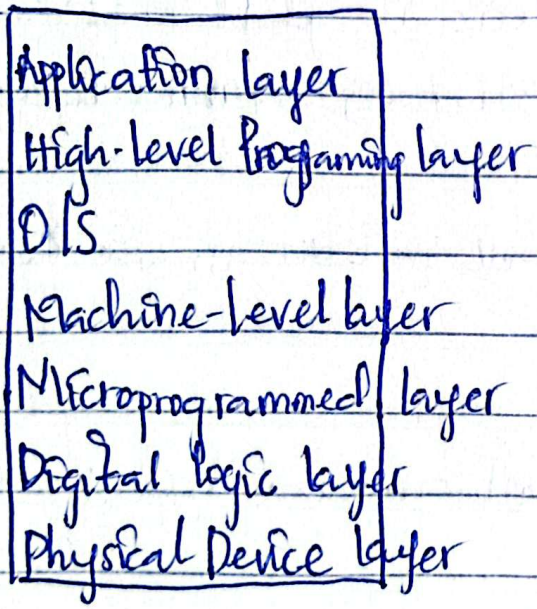
(2) There are also some security risks in the

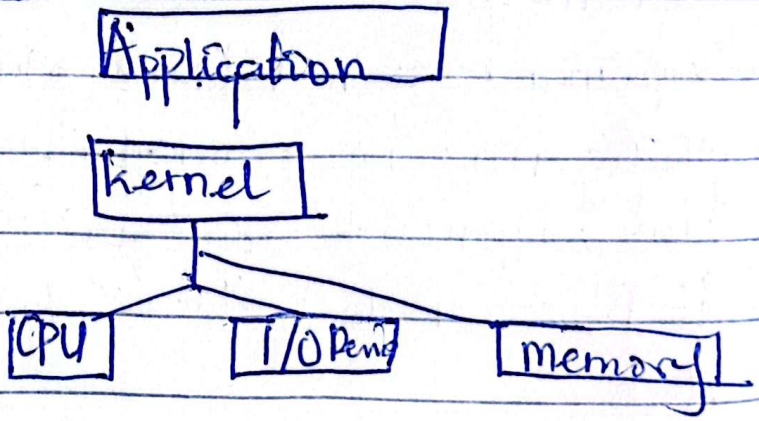
In the monolithic architecture

3) Micro-kernel Structure

Involves removing all non essential components from the kernel and implement them as system and user programs this results in a smaller kernel called micro-kernel







Advantages of Micro-kernel

It makes the OS portable to various Platforms

As microkernels are small so theses can be tested effectively

Disadvantages

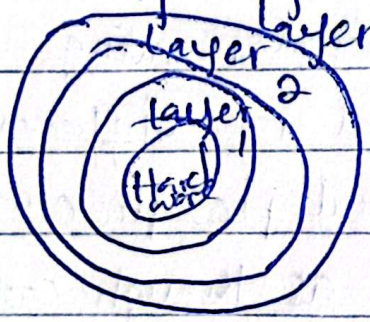
Increased level of inter module communication degrades system performance

EXO-Kernel Structure

Is an OS that was first developed in MIT to provide application-level management of hardware resources

Layered Structure

Is an OS that is broken into pieces and retain much more control over the system. In this structure, OS is broken down into a number of layers (levels).



n — user-interface Layer

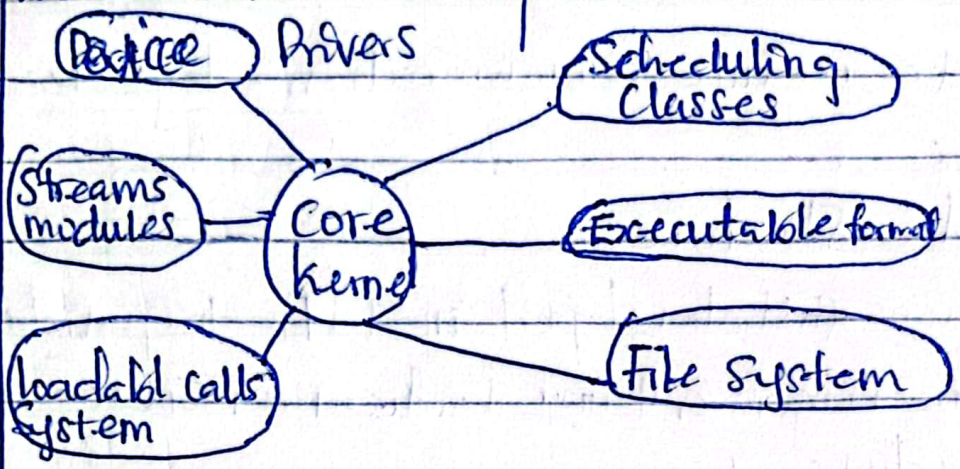
Linux uses layered structure

~~Windows~~ To make the processor access speed ≤ memory access speed a memory is added called Cache.

The entire Design of the Computer System is modular Design

Modular Structure

A modular O/S structure is based on the idea of dividing the system into smaller and independent Units, called modules, that can be loaded and unloaded as need arises during runtime or boot-time



Virtual Machines

A Virtual Machine abstracts the Hardware of the PC, such as CPU, disc drives, RAM and NIC, into a variety of different execution contexts, giving an impression that each execution enviroment is a different computer.

VM enables us to run multiple processes concurrently while making it appear as though each one is using a different processor and Virtual memory by using CPU scheduling and Virtual memory techniques

It is used in implementing client processes

Client - Server Model

In Client - Server model most of the OS is implemented in user processes. To request a service such as reading block of a file, a user process called (client Process) sends the request to a server process, which then does the work and sends back the answer

Client Process	Client Process	Process Server	Terminal server	...
File server	Memory Server	micro kernel		

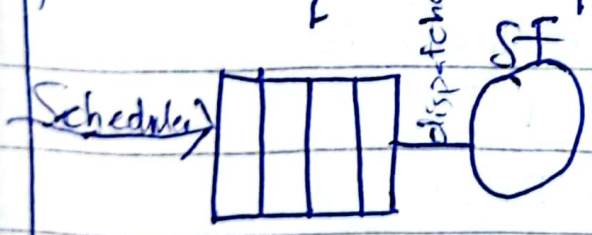
In this model, all the kernel does is handled the communications b/w clients and servers.

Process Scheduling

When the process is loaded into memory the Heap section contains program code

Scheduling: Is the act of arranging a set of processes in some ordered way according to a well-defined strategy

Dispatching: Is the operation of selecting a process from the queue and providing it with access to CPU



Service facility.

Dispatching latency is the amount of time
a To execute a particular process
b to stop one process and start another
c A process been waiting in the ready queue
d from when the request was submitted until time response was produced.

Types of Scheduler

3 types of schedulers
(i) Long term Schedulers
(ii) Medium-term
(iii) Short term Dispatcher

Figure 16

Preemptive & Non preemptive

Interupts & It's roles

OS steps in Context Switching

1. Saves registers P_1, in the stack to eventually return back to the calling function
2. Stores the stack pointer of P_1 in the current block PCB_1
3. load the stack pointer of P_2 from the PCB_2
4. Restores the P_2 registers from the stack
5. Returns to registers P_1 from PCB_1 and columns

Interupts are signals sent to the CPU by external devices, normally I/O devices.
They tell the CPU to stop it's current activities and execute the appropriate part of the OS:
Maskable Interupts & non-maskable interupts.
Hardware Interupts refers to interupts; maybe key strokes
Software
Traps
Ethernet card is a card in the device that enables it connect to the internet
ieee 802.11a|b|c|d - wireless ethernet card
ieee 802.3 } wired.
ieee 802.5
Software Interupts are generated by program.
Traps these are generated by CPU itself. where assistance is needed
Interupts are important because they allow the user control over the computer.

Concurrency

Deadlock Resource starvation

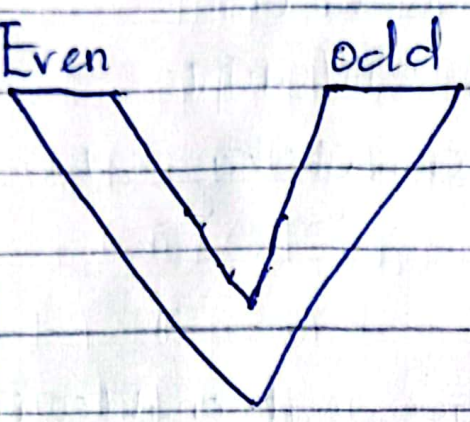
Concurrency

Dinning Philosophy Problem

Interleaved Processes

Overlapped processes

both interleaved an overlapped ca



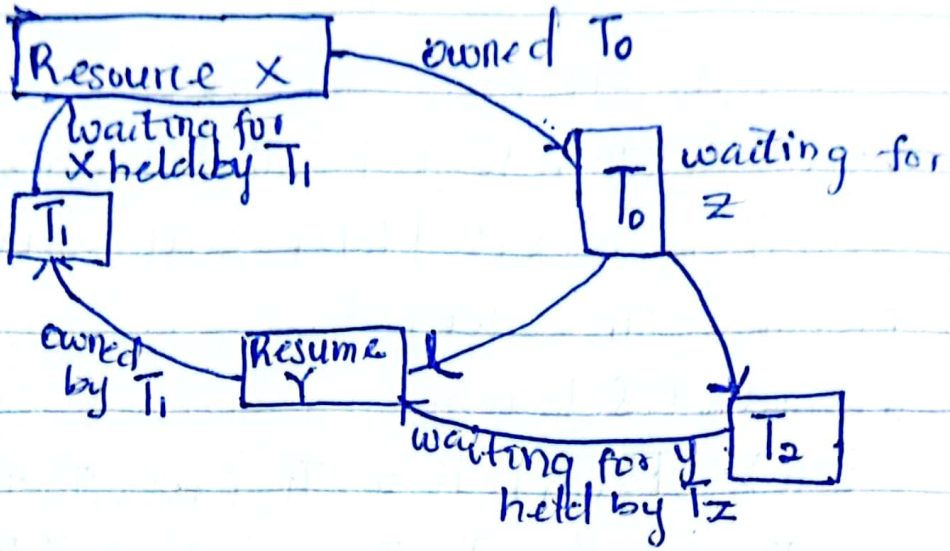
Problems of concurrency

Deadlock

Causes, Conditions, Prevention

Deadlock is an unwanted situation that happens in a share resource environment. Eg of a share resource is a global library. In a multiprocess system is where a deadlock can occur.

A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause.



Race condition (Problem)
Mutual Exclusion. (Solution)

Deadlock is not healthy for a system. the transaction required for
O/S
can kill the processes to free up resources

Deadlock prevention

To prevent any deadlock situation in the system, the DBMS aggresively inspects all the operations and analyze if they can create a deadlock situation. There are deadlock prevention schemes ~~that uses~~

(i) Wait die Scheme

If TS (Ti) < TS(Tj) - that is Ti, which is requesting a conflicting lock, is older than Tj then Ti is allowed to wait

until the data-item is available.
- If TS(Ti) > TS(Tj) - that is Ti is younger than Tj - then Ti dies. Ti is restarted later with a random delay but with the same timestamp.

Wound-wait scheme:
If TS(Ti) < TS(Tj), then Ti forces Tj to be rolled back - that is Ti wounds Tj. Tj is restarted later with a random delay but with the same timestamp.
If TS(Ti) > TS(Tj), then Ti is forced to wait until the resource is available

- The following four conditions must hold for deadlock to take place:

i) Mutual exclusion condition
ii) Hold & wait condition
iii) No preemption
iv) Circular wait condition

The Dinning Phylosophers problem

Bankers Algorithm

Key things: The bankers alg. is a combination of two algorithms. Safty algorithm and resource request algorithm

for SA find two vectors Work & finish

① Initialize work

work = Available

finish[i] = false; (for i = 0, 1, 2, 3, ..., n-1)

② Need[i] <= Work

Finish[i] = false

If the i does not exist go to 4

③ Work = work + Allocation[i]

Finish[i] = true

④ If finish[i] == true; then the system is safe;

Process	Allocation (A B C)	Max (A B C)	Available
P_1	0 1 0	7 5 3	3 3 2
P_2	2 0 0	3 2 2	
P_3	3 0 2	9 0 2	
P_4	2 1 1	2 2 2	
P_5	0 0 2	4 3 3	

① What is the reference of the need matrix

② Determine if the system is safe or not

(ii) What happens if the resource request (1, 0, 2) for process P_1, can the system accept the request immediately

Work > Available

~~Need[i,j] =~~

Need[i][j] = Max[i][j] - Allocation[i][j]

P_1 : (7, 5, 3) - (0, 1, 0) = (7, 4, 3)

P_2 : (3, 2, 2) - (2, 0, 0) = (1, 2, 2)

P_3 : (9, 0, 2) - (3, 0, 2) = (6, 0, 0)

P_4 : (2, 2, 2) - (2, 1, 1) = (0, 1, 1)

P_5 : (4, 3, 3) - (4, 0, 2) = (0, 3, 1)

} Need

Step 1: For Process P_1

Need <= Available

(7, 4, 3) <= (3, 3, 2)

Step 2: For process 2

1, 2, 2 <= 332

New Available = Available + Allocate = (6, 0, 0)

Now Available = 332 + 200 = (0, 1, 1)

= 532 (4, 3, 1)

Step 2

Need <= available

600 <= 532 Process is false

P_4

Need <= Available

(0,1,1) <= 5 3 2

New available = Available + Allocation

5,3,2 + 2,1,1 = 7,4,3

For Process 5

Need <= Available

New available 7,4,3 + 0,0,2 = 7,4,5

For P_1 : 7,4,3 <= 7,4,5

7,4,5 + 0,1,0 = 7,5,5

For P_3

6,0,0 <= 7,5,5

7,5,5 + 3,0,2 = 10,5,7

Here the system is safe

Condition variables are synchronization primitives that allows threads to wait until a particular condition occurs. Examples of condition variables are:

i) Monitors
Semaphores:
Sleep and wakeup:

} Shared resources and Critical Section

Global variables are examples of shared resources multicore system is called shared memory system

starvation in Os: Starvation is almost closely related to deadlock, occurs when a pro

Producer-Consumer Problem

there is one producer and one consumer consuming the produced goods. Two processes share a common, fixed-size buffer. One of them, the producer puts information into the buffer, and the other one takes it out. Trouble arises when the producer wants to put a new item in the buffer, but it is already full.

Multiprocessor Systems

Ens low, A multiprocessor system must satisfy the following prope

(i) Must contain two processor that have approximately ~~equal~~ Comparable capabilities

(ii) All processors share access to a common memory. It does not mean that each processor does not have

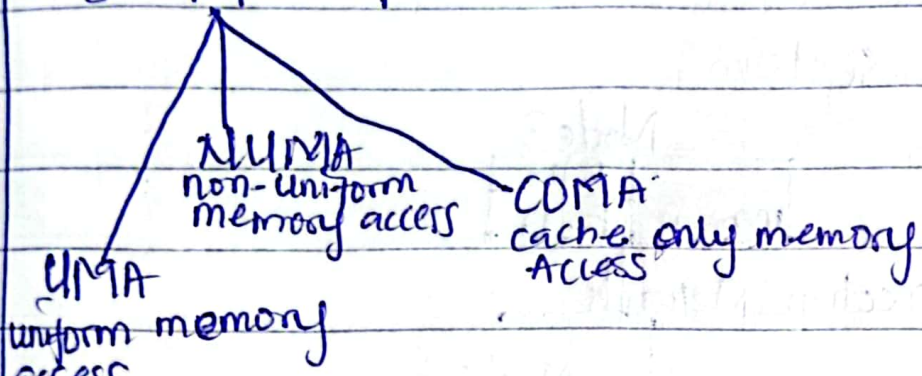
SMP — Share

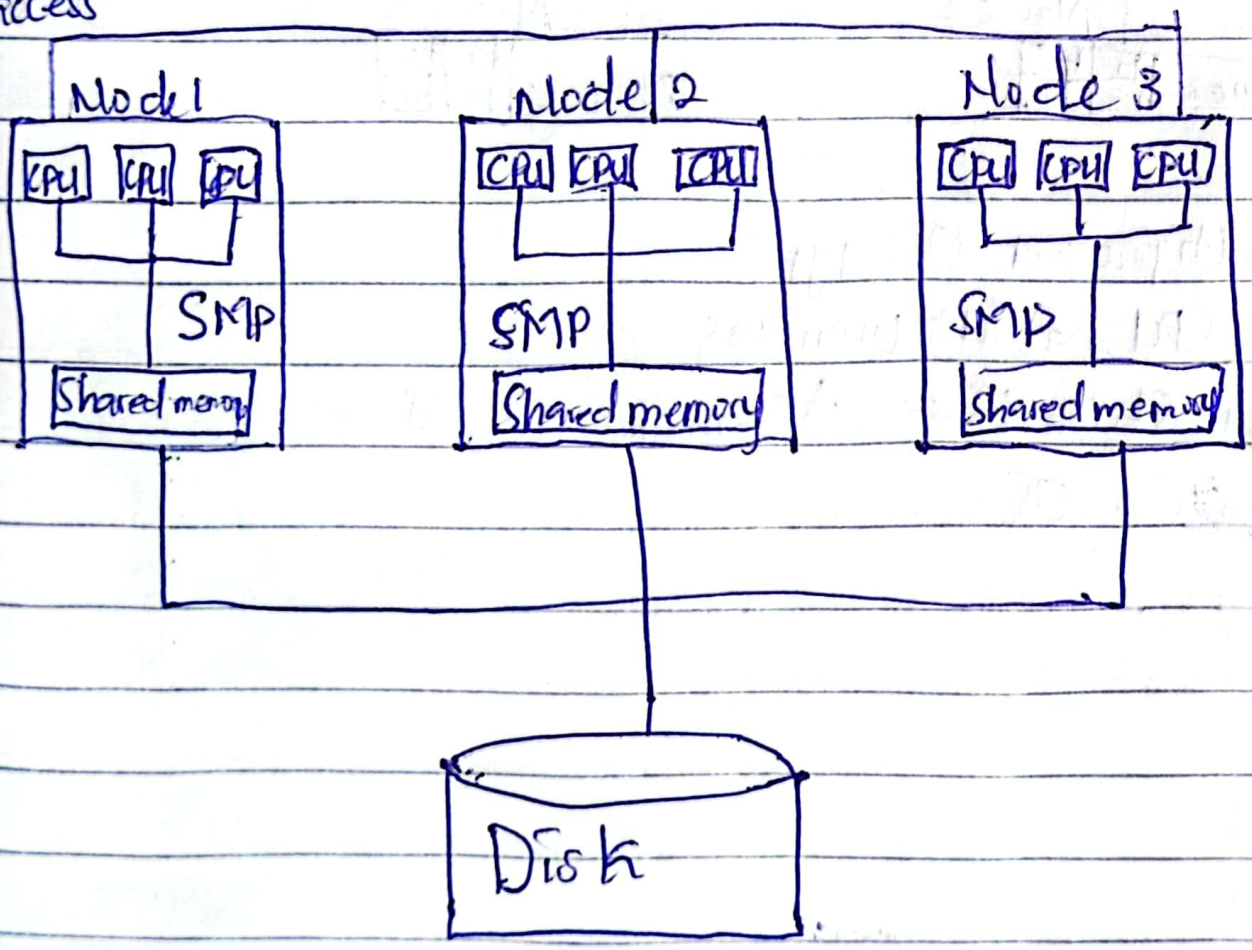
Disadvantages

Credibility is limited by bandwidth, the higher the bandwidth the faster the system is.

Two types of Shared Memory

S M P





detoriat Thrasing: A system runing multiprocessor begins to

(b)

A tightly coupled multiprocessor does not conform to the above definition of a multiprocessor

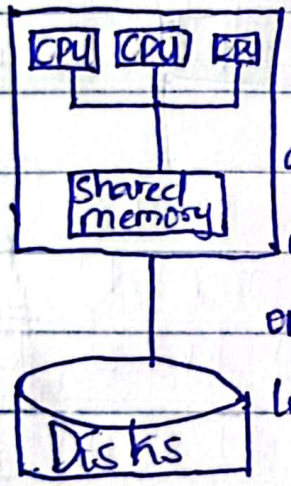
Asymetric uses master slave architecture -

~~Memory architecture of Parallel~~ 3 Architectures of parallel computing

The memory architecture are: shared memory

(ii) Distributed Memory

(iii) Hybrid Distributed Shared.



it can be 7.24 but performance detoriater

It consists of set of CPU one single memory any of the processor can access that one memory communicatio:

only one processor can access the shared memo location

Shared memory is limited to some number of processors

Advantages

(i) Memory Access is less expensive

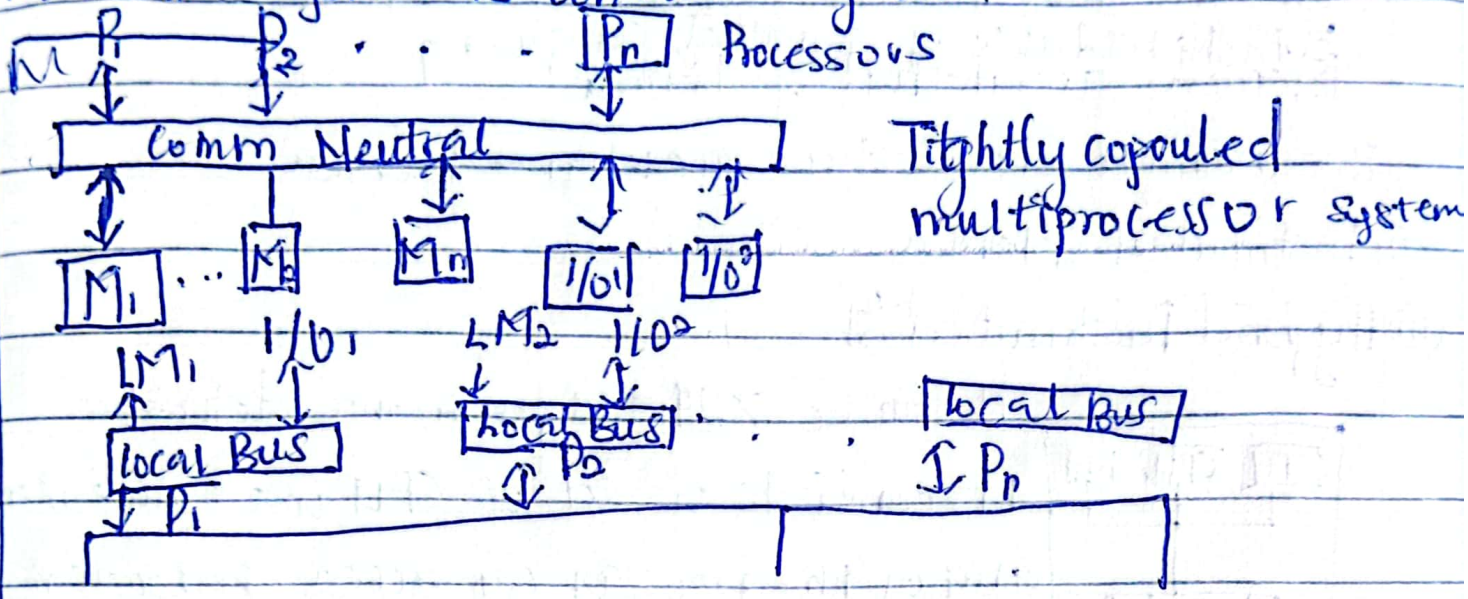
(ii) They are easier to administer

(iii) It is easier to program

Cache is for pre-fetched data
Cache miss/Hit

It's own local memory.
All processors shared access to I/O channels, control units & devices. It does not mean that it doesn't have local I/O Interfaces

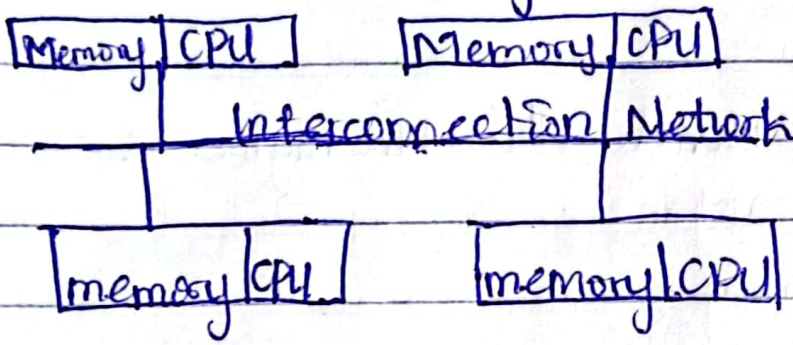
The entire system is controlled by one OS



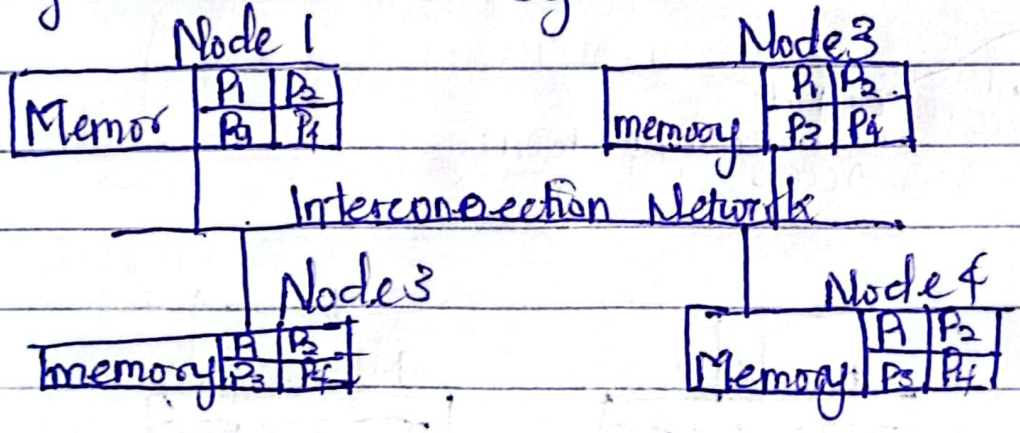
Tightly copouled multiprocessor system

If distributed system has
A multiplicity of general purpose, physical and logical resources that can be assigned to specific tasks on a dynamic basics

② Distributed Memory



③ Hybrid / Distributed system



Multiprocessor OSs types

(i) Each CPU has its own OS

(ii) Master Slave OS = Asymmetric OS

(iii) Symmetric OS

Memory Management System 14/09/2025

How O/S manages Memory: RAM

Set of Memory

CPU registers, cache L1 and L2, RAM / ROM → HDD → External Disk

How OS manages content of RAM, physical Address is address of memory:

physical Address = logical Address + offset

logical Address is the address in the users program

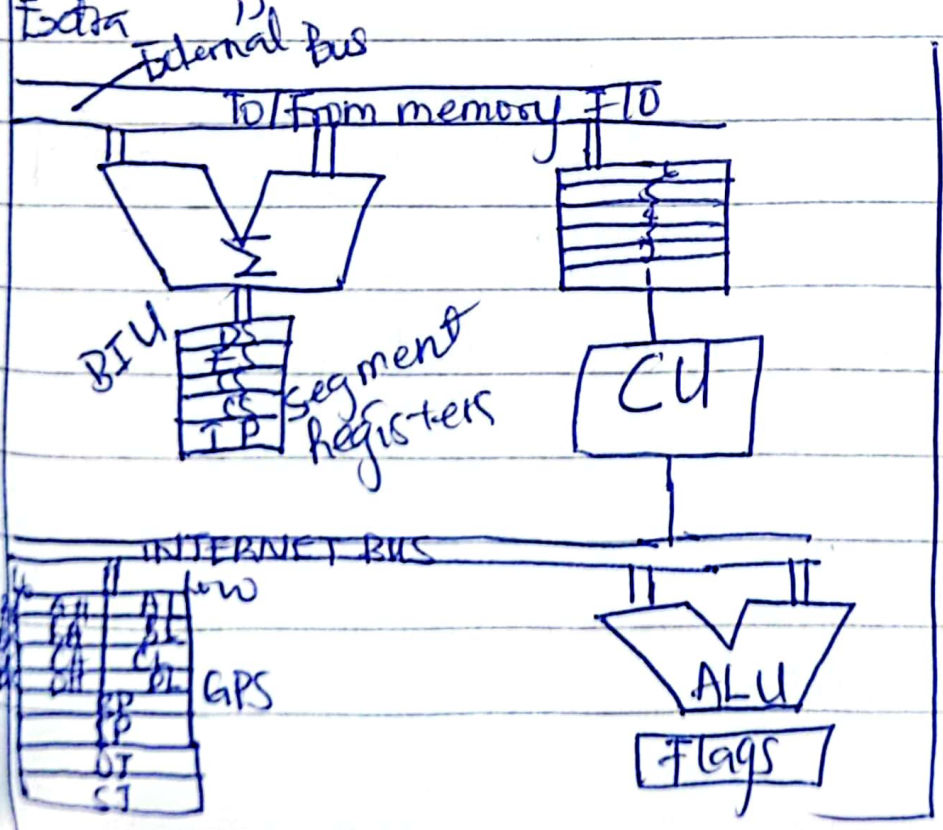
The registers used here are

Code Segment

Data ,,

stack ,,

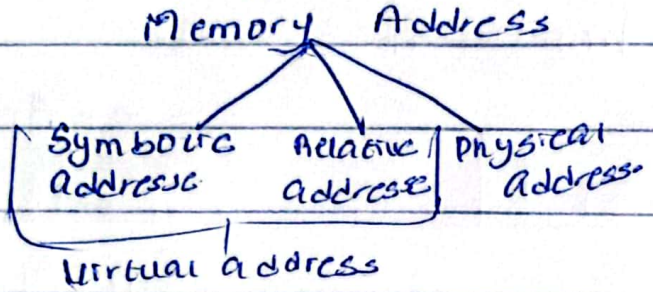
Extra ,,



The CPU Diagram

The memory management system is the functionality of an operating system which handles or manages primary memory memory mgt keep track of each and every memory location
If the size of the CPU is 32 bit or 64bit the size of the RAM $2^{32} = 4,294,967,296 \approx 4GB$
$2^{31} = 2GB$ is what process use
The O/S tracks unused memory

There are three types of memory addresses



Physical Address = logical address + offset
(Virtual Address)

Symbolic addresses are addresses used in the sourcode
Mov [Bakpo], 100 (table)
In the process of runing a program there is:
- Load time
- Compile
- Runtime Execution

Virtual and physical addresses in compile and load

The runtime mapping from virtual to physical is done by the MMU. The user program uses virtual addresses, not the physical address

Choice of static and dynamic loading is decided when the program is being developed.

Swapping is done b/w main memory and hard disk.