OPERATING SYSTEM

PAGE No.
DATE

→ OS is an interface b/w User & Computer System

Interface ⇒ Working Environment —(Easy way)→ Hardware Access

→ OS is a software which manages Computer Resources

-
- BOOTING → The process of starting a computer is known as Booting. The various steps involved in Booting are :—

1) BIOS → Basic ~~Inp~~ I/P O/P Service
This is a program which is pre-installed in the ROM or EEPROM.
Note → This is not a part of OS.
This is the 1st & foremost program to run when booting.

~~This~~

2) Now, the BIOS reads & locates all the ~~per~~ computer peripheral devices attached to the computer (such as Processor, Keyboard, Memory etc —) & i.e activates them & makes them ready for use.

3) After this a program known as BOOTSTRAP LOADER locates the hard disk (Secondary Storage) & the OS inside it.
Once OS is located, the Kernel of the OS is fetched from the Sec. storage to the Memory (RAM)

* This is because, as w.k.t.

the size of complete OS — 2, 4, 6, 8 ... etc GB

& Size of RAM is also — 2, 4, 6, 8 ... — GB

So the complete OS can't be put in RAM

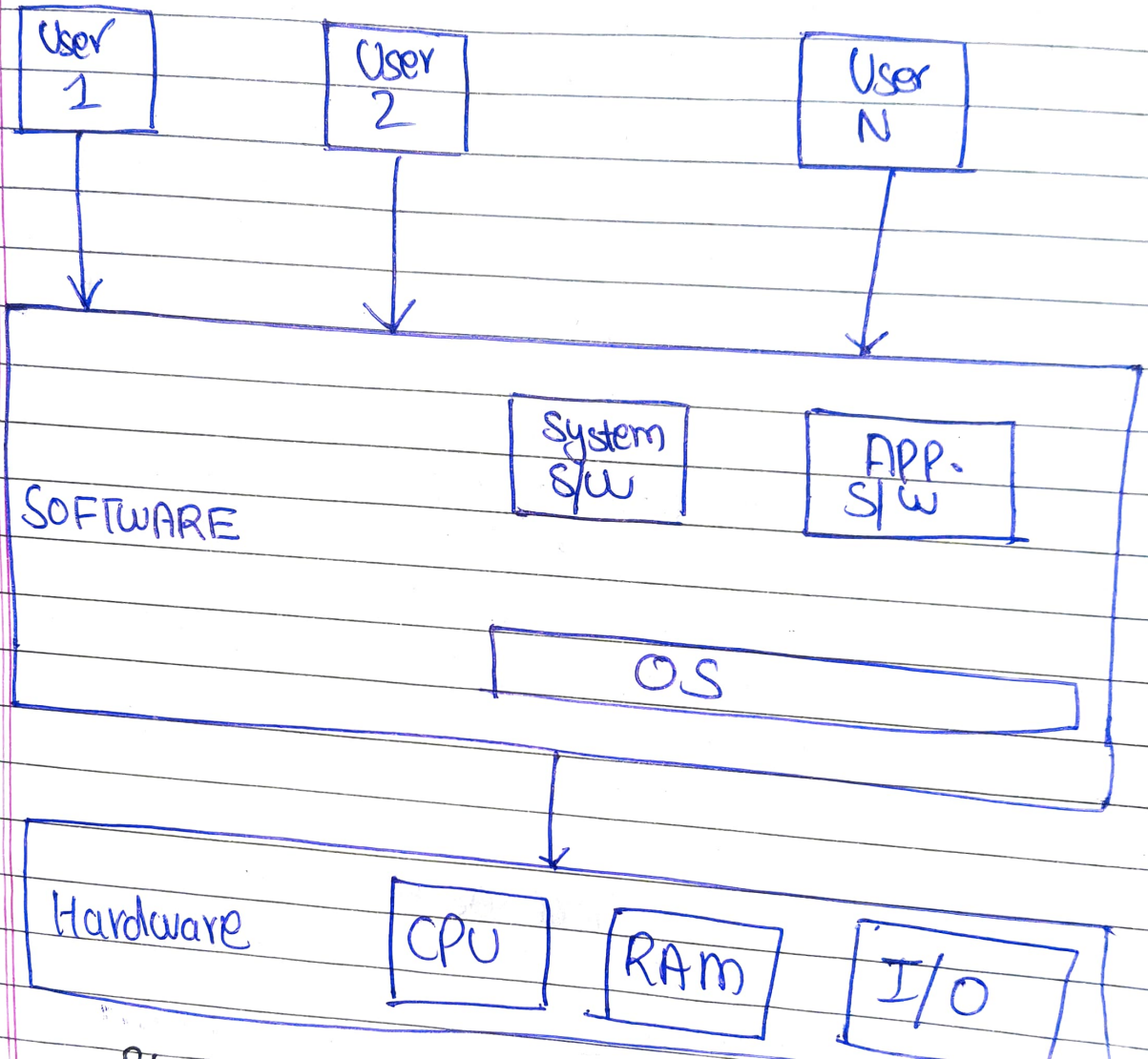
So to resolve this, only the main program of OS known as Kernel is loaded into RAM

*** FIRST OS → Developed by General Motor Research (G.M.R.) in 1956

known as GM-NAA I/O for IBM 704
by Robert L. Patrick

*** FIRST BATCH OS → OS/360 for IBM 360 in 1964.

→ An OS is a program that acts as an interface b/w user & the computer hardware & controls execution of all kinds of programs.



PAGE No.
DATE

→ FUNCTIONS OF OS

1) MEMORY MANAGEMENT → It refers to the management of primary memory. Main mem. provides a fast storage that can be accessed directly by the CPU.

* For a Prog. to be executed it must be in main mem.

a) OS keeps track of Primary Mem. i.e what parts of it are in use by whom & what parts aren't in use.
b) In multiprogramming, the OS decides which process will get memory when & how much
c) Allocates the memory when a process requests it to do so.
d) De-allocates the memory when a process no longer needs it or has been terminated.

2) PROCESS MANAGEMENT → Here the OS decides ~~which~~ ~~processor~~ gets the processor ~~to~~ when & for how much time. ~~The~~ This fn is called Process Scheduling.

a) OS keeps track of processor & status of processes. The program responsible for this task is known as Traffic Controller

b) OS allocates the processor (CPU) to a process.

c) OS also De-allocates processors when a process is no longer required.

3) DEVICE MANAGEMENT → OS manages device communication via their respective device drivers.

a) OS keeps track of all devices. The Program responsible for this task is known as the I/O Controller

b) Allocates the device in the most efficient way

c) De-allocates the devices

4) FILE MANAGEMENT → A file system is normally organized into directories for easy navigation & usage. These directories may contain file & other directories.

a) OS keeps track of information location, user, status etc. The collective facilities often as known as file System

b) OS decides who gets the resources

c) Allocate & de-allocate the resources.

5) SECURITY → By means of passwords & similar other techniques, it prevents unauthorized access to programs & data

6) CONTROLS OVER SYSTEM PERFORMANCE → Records delay b/w req. for a service & response from the System

7) JOB ACCOUNTING → Keeps track of time & resources used by various jobs & users.

8) ERROR DETECTION AIDS → Production of dumps, traces, errors, messages & other debugging & error detecting aids.

9) CO-ORDINATION B/w other S/W & Users → Co-ordination & assignment of Compiler, Interpreters, assemblers & other S/w to various users of the Computer Systems.

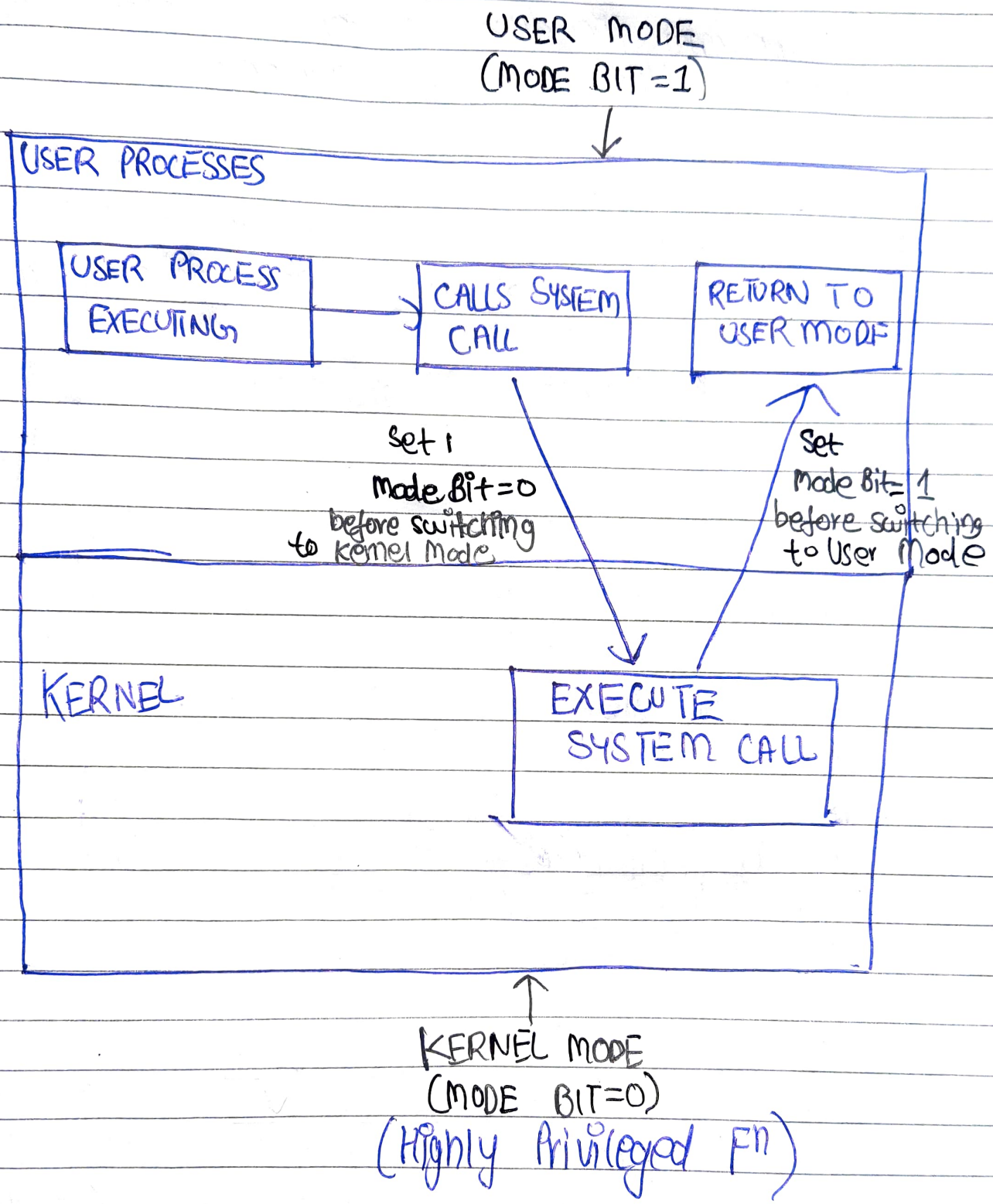
* USER MODE & KERNEL MODE

These are the 2 modes in which a CPU works in.
To achieve the Goals of Convenience & Efficiency ~~Provide~~ promised by the OS, the OS needs these 2 modes of the CPU.

So suppose the CPU is executing a User Process E.g. running a User App. like msword ppt etc then it needs to use some highly privileged fn of the OS then it makes System calls & sets the ModeBit = 0 & indicating that it is switching to Kernel mode.

Now if OS is able to complete those calls, it ~~process~~ executes those system calls.

to Return to User Mode, it sets the mode Bit = 1 indicating that it wants to switch to User Mode



* Kernel Mode is also known as :-
  a) Supervisor Mode
  b) System ,,
  c) Privileged ,,

SYSTEM CALLS

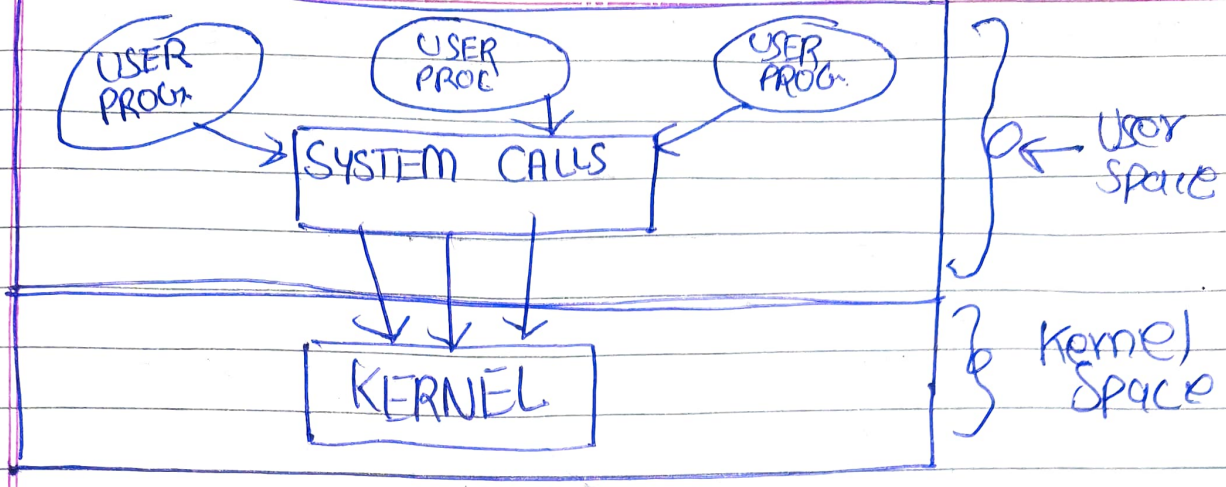
Basically a System call is an instruction that requests the OS to perform the desired operations that need "Hardware Access or other privileged operations.

→ W.R.t there are 2 modes of operation of CPU :—
a) USER MODE → All user processes are executed
b) SYSTEM MODE → All privileged operations are executed ***

→ So the User Prog. & Kernel fn are executed in their respective space allocated to them in main memory partitions

→ If the User Mode programs need to execute some privileged operations, which aren't permitted in User Mode fn, User Mode programs must use an Interface which forms the only permitted interface b/w the User mode & Kernel mode. This interface is called "SYSTEM CALLS". So System Calls act as an interface b/w User Prog. & OS.

→ System calls generate an INTERRUPT that cause the OS to gain control of the CPU. The OS then finds out the type of System Calls & corresponding Interrupt Handler Routine to be executed to perform operation.



*** System Calls are inherently used for Security Reasons. Due to the Use of System Calls, a User Prog. is not able to enter into OS or any other User's Regions.
Similarly I/O devices are also safe from any misuse by the User.

→ Thus through the Use of System Calls, Kernel, Other user programs & I/O devices are safe & secure from malicious User Programs.

→ Making a System Call

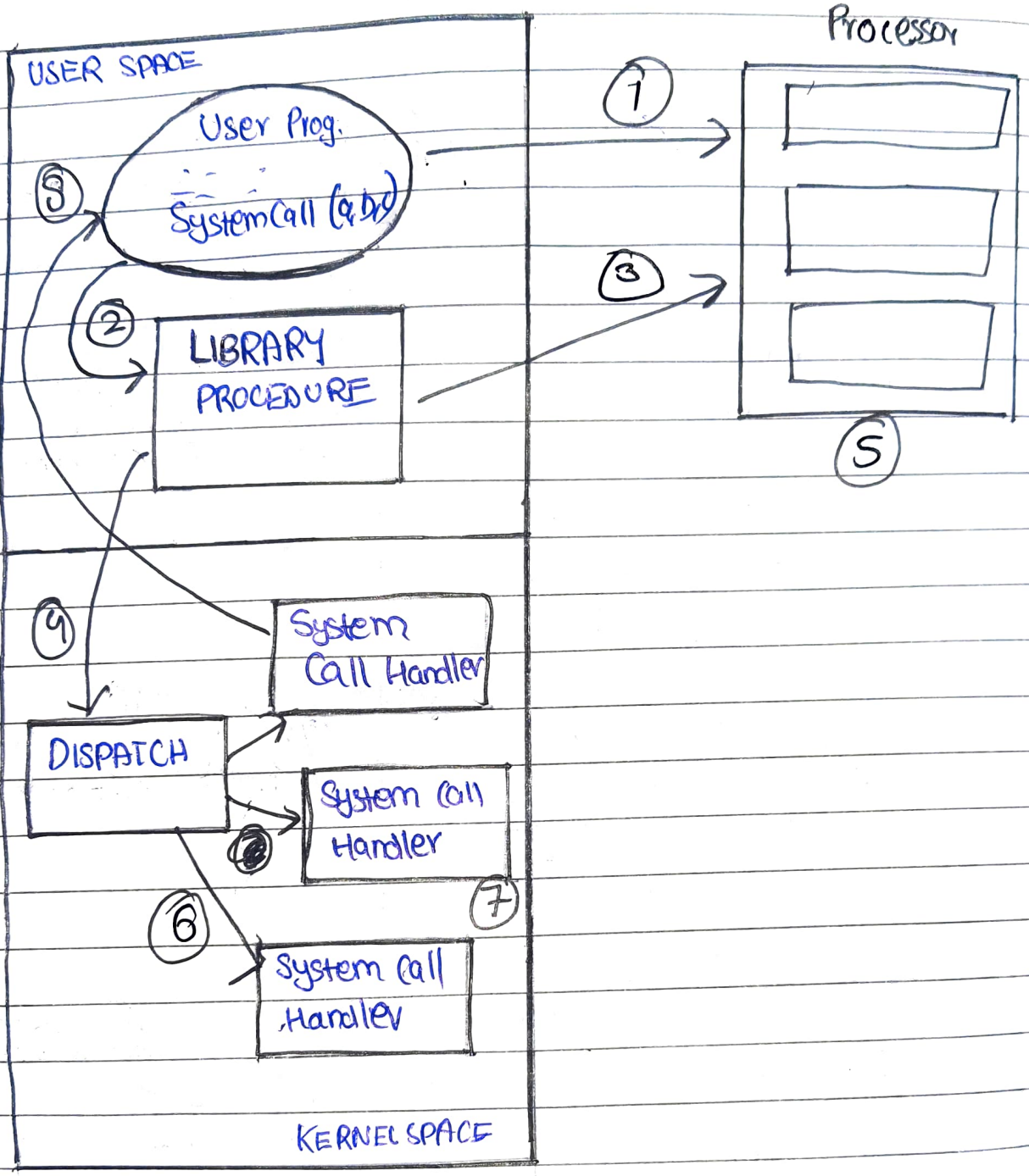
Now System Calls are directly available & are in High-Level Languages like C & C++. So it has become easy to use them.

For a Programmer, System Calls are same as Calling a Procedure or fn.

*** V.Imp

The diff. b/w a System Call & a normal fn Call is that a System Call enters the Kernel but a normal fn Call does NOT.

→ SEQUENCE IN WHICH SYSTEM CALL IS EXECUTED



(1) In the User Prog., when the System call is Executed first of all, its parameters are pushed onto the stack & later saved ~~on~~ in processor registers

(2) The corresponding library procedure ~~is~~ for that system call is Executed i.e the corresponding code is extracted

(3) There is a particular code for every system call by which the Kernel identifies which system call or handler needs to be executed ∴ Library procedure places the system call no. in processor registers.

(4) Then the library procedure traps to the kernel by executing Interrupt instruction. With this int. ~~Instr.~~ execution, the user mode switches to Kernel mode by loading Program Status word (PSW) register to the OS.

(5) The h/w saves the current content of CPU registers so that after executing ~~the~~ System calls, execution of ~~Rest~~ of the program can be resumed.

(6) The Kernel identifies the System call by examining its no. & dispatches the control to the corresponding System call handler

(7) The System Call handler executes

(8) On completion of System Call handler, the control is returned to the User's program & it resumes its execution.

TYPES OF SYSTEM CALLS

1) PROCESS CONTROL SYSTEM CALL →

A process is a basic entity in the system. The processes in the system need to be created, deleted & aborted. Besides these many operations are required on the processes for their management.

WINDOWS	UNIX
CreateProcess()	fork()
ExitProcess()	exit()
WaitforSingleObject()	wait()
	kill() ← exit abruptly
	nice() ← Increase priority of a process

2) FILE MANAGEMENT System Calls →

Creation, deletion, opening, closing & reading & writing are some general operations on files. Similarly for organizing files, there is a directory system & thereby system calls for managing them.

WINDOWS	UNIX
CreateFile()	create()
ReadFile()	read()
WriteFile()	write()
CloseHandle()	close()
	open() Read/Write
	Change Pos. of Pointer → lseek()
	Give Another name to file → Link()

Delete a file in a Directory → Unlink()
Create a New Directory → mkdir()

3) DEVICE MANAGEMENT System Calls →

WINDOWS	UNIX
SetConsoleMode()	ioctl()
ReadConsole()	read()
WriteConsole()	write()
SetTimer()	

4) INFORMATION MAINTENANCE System Calls →
Many System Calls exist simply for the purpose of transferring info. b/w the User Programs & OS

WINDOWS	UNIX
GetCurrentProcessID	getPid()
SetTime()	alarm()
SetSleep()	sleep()
	getFile()

5) COMMUNICATION SYSTEM CALLS → there is a need for Communication among the Processes in the System.

→ General Operations are →
- Opening & Closing the connection
- Sending & Receiving Messages
- Reading & Writing Messages & so on

MsgSnd() → Sending a message
MsgRcv() → Receiving a message

These System Calls may be related to the communication b/w processes either on same machine or b/w processes on diff nodes of a network. Thus inter-process communication is provided by OS through these communication-related System Calls.

WINDOWS	LINUX
CreatePipe() ← creates a communication link b/w 2 points	pipe()
CreateFileMapping()	Shmget()
MapViewOfFile()	mmap()

Transfer data from RAM to Virtual Memory

6) SECURITY & PROTECTION System Calls →

Windows	UNIX
SetFileSecurity()	chmod()
InitializeSecurityDescriptor()	umask()
SetSecurityDescriptorGroup()	chown()

SYSTEM PROGRAMS

PAGE No.
DATE

System programs are utilities programs that help the user & may further call system calls. System programs provide a convenient environment for program development & execution. Some are simply user interfaces to System Calls & others are considerably more complex. They can be divided into these categories:-

1) FILE MANAGEMENT → These programs create, delete, copy, rename, print, dump, list & generally manipulates files & directories.

2) Status Information → Some System Programs simply ask information related to system like Date/Time/ Size of DISK/ No. of Users.

3) FILE MODIFICATIONS → Several Text Editors maybe available to create & modify the content of files stored on disk or other storage devices. These may also be special commands to search contents of files as perform transformation of text.

4) PROGRAMMING LANGUAGE SUPPORT → Compilers, assemblers debuggers & interpreters for common Programming Languages (such as C, C++) are often provided to the User with the OS.

5) Program loading & Execution → Once a program is assembled or compiled, it must be loaded into memory to be executed. The System may provide absolute loaders relocatable loaders, linkage editors & overlay loaders. Debugging System for either higher-level languages or machine language are needed as well.

6) Communication → These programs provide the mechanism for creating virtual connections among processes, users & computer systems. They allow users to send messages to one another's screens, to browse web pages, to send e-mail messages to log on remotely or to transfer files from one machines to another machine

Some System program supplied with as are Text formatters, Spreadsheets, Compiler, Web Browsers, database System, Games etc. ——

INTERRUPTS

Interrupt is a mechanism by which computer components, like memory or I/o modules, may interrupt the normal processing of processor & req. processor to perform other specific tasks

Types of INT.

1) PROGRAM → Generated by some condition that occurs as a result of an instruction exec. such as:-
   - Arithmetic Overflow
   - Page Fault
   - Outside memory space Ref
   - Div. by 0
   - Invalid Instr.

2) TIMER → Gen. by a timer within the processor. This allows OS to perform certain fn on a regular basis
   Eg Shutdown, Sleep etc

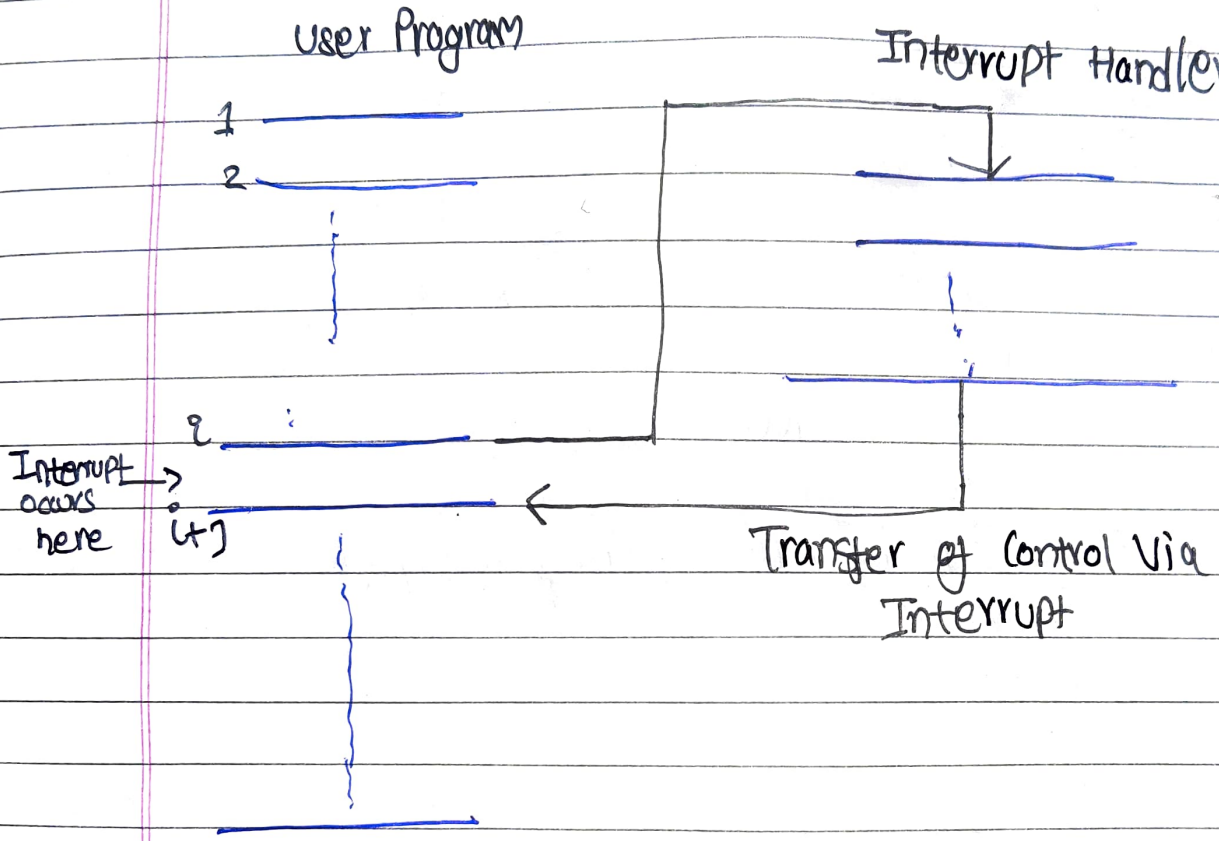
3) EXTERNAL or I/O → by an I/o controller, I/o devices tell the CPU that an I/o req. has completed by sending an interrupt signal to the processor

4) H/w FAILURE → Gen. by a failure such as :—
   POWER FAILURE
   MEMORY "
   RISE IN TEMP

INTERRUPT MECHANISM

DATE

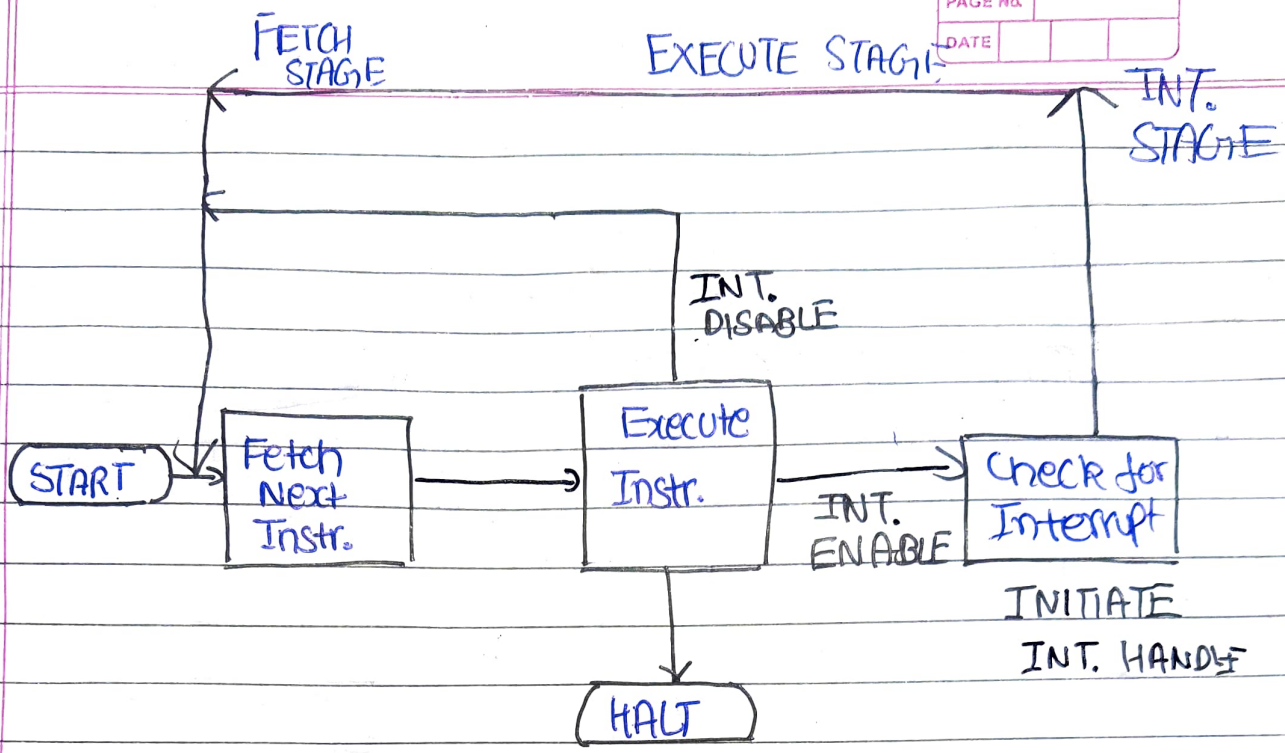
For the user prog., an int. suspends the normal sequence of exec. When the int. processing is completed, exec. resumes. Thus the user prog. doesn't have to contain any special code to accomodate interrupts



To accomodate int., an interrupt stage is added to the inst. cycle. In the int. stage, the processor checks to see if any int. have occurred, indicated by the presence of an int. signal.

If no int. are pending, the processor proceeds to fetch stage & fetches next instr. of current prog.

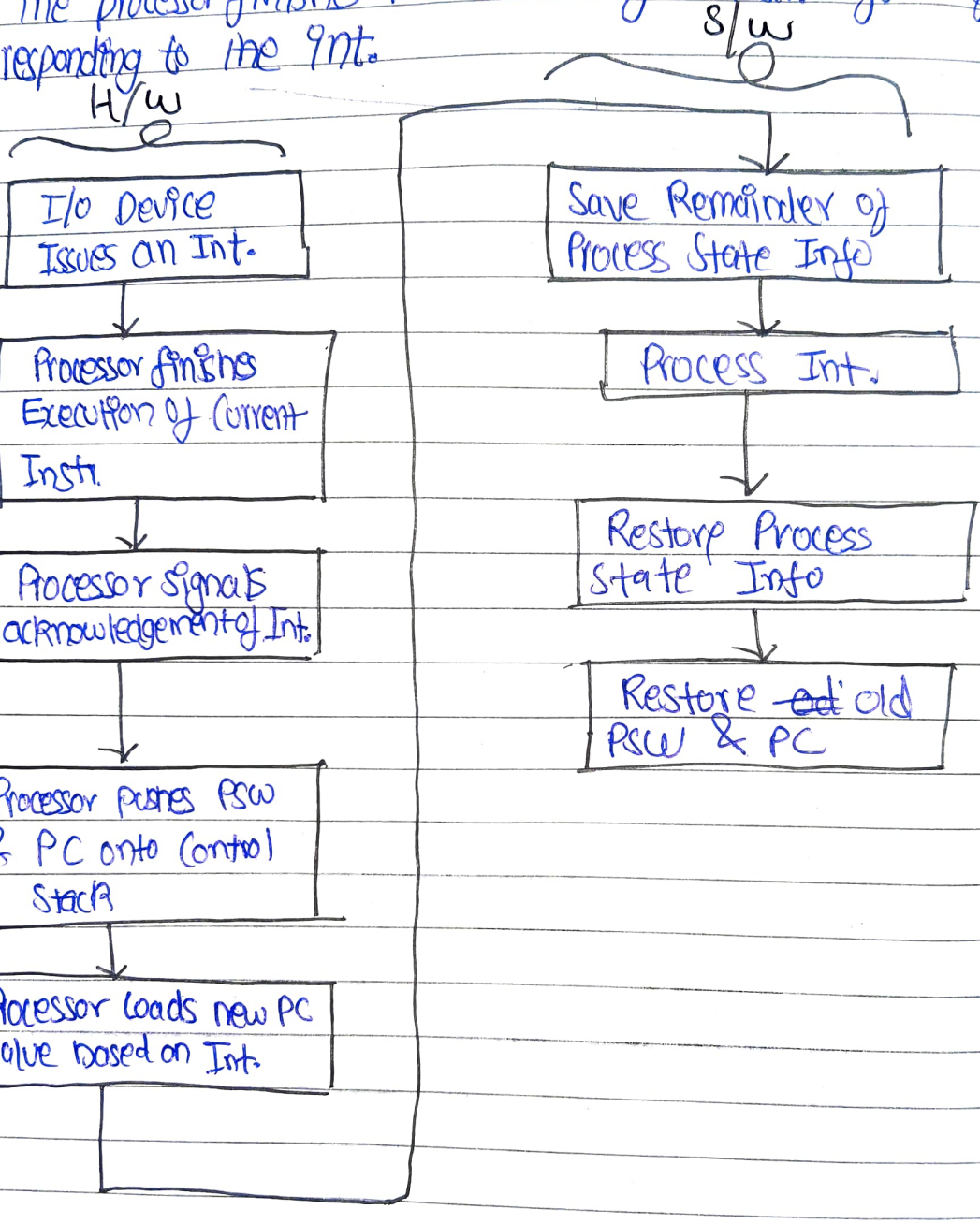
→ If an int. is pending, the processor suspends execution of current prog. & executes an 'Interrupt Handler Routine'



INSTRUCTION CYCLE WITH INT.

→ Interrupt Processing Procedure →

→ I/o device issues the interrupt
→ The processor finishes the execution of an instr. cycle before responding to the Int.



BATCH OPERATING SYSTEMS

→ Was mainly used in mainframe computers e.g. IBM 360

→ Jobs which are of similar type are grouped together & as "Batch".

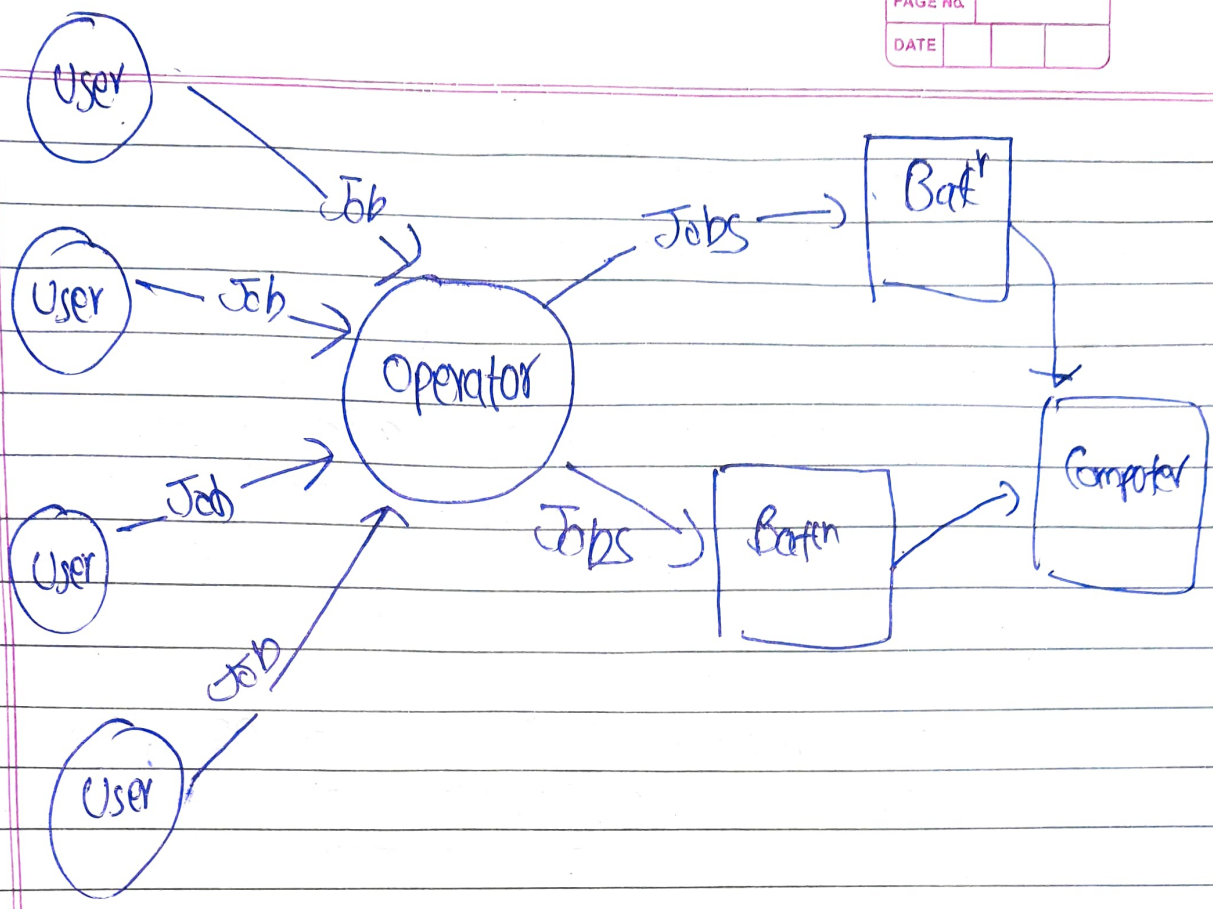
→ The OS is termed as "batch operating" because the input data (job) are collected into batches or set of records with similar needs & each batch is processed as a unit (group). The o/p is another batch that can be reused for computation.

→ In this jobs which are of similar type are grouped together & treated as a batch.

→ Now, they are stored on the Punch Card (a stiff paper on which digital data is stored & represented using some specific sequence of holes) which will be submitted to the system for processing.

→ The system will then perform all the req. operations in a sequence. So, we consider this as a type of Serial Processing.
E.g. Bank Statements.

* During Batch OS, Secondary Storage wasn't available i.e only Main Memory was available.



- **Advantages**

1) Suppose a job takes a very long time (1 day or so) Then such processes can be performed even in the absence of humans

2) They don't require any special hardware & system support to I/P data

- **Disadvantages**

1) Very difficult to debug batch systems
2) Lack of interaction b/w user & OS.
3) Suppose an error occurs in one of the jobs of the batch. Then all the remaining jobs get affected ie they have to wait until the error is

MULTIPROGRAMMING OS

→ In a multiprogramming system there are or more programs loaded in main memory which are ready to execute.

Only 1 program at a time is able to get the CPU for executing its instructions. (i.e there is at most one process running on the system) while all the others are waiting for their turn.

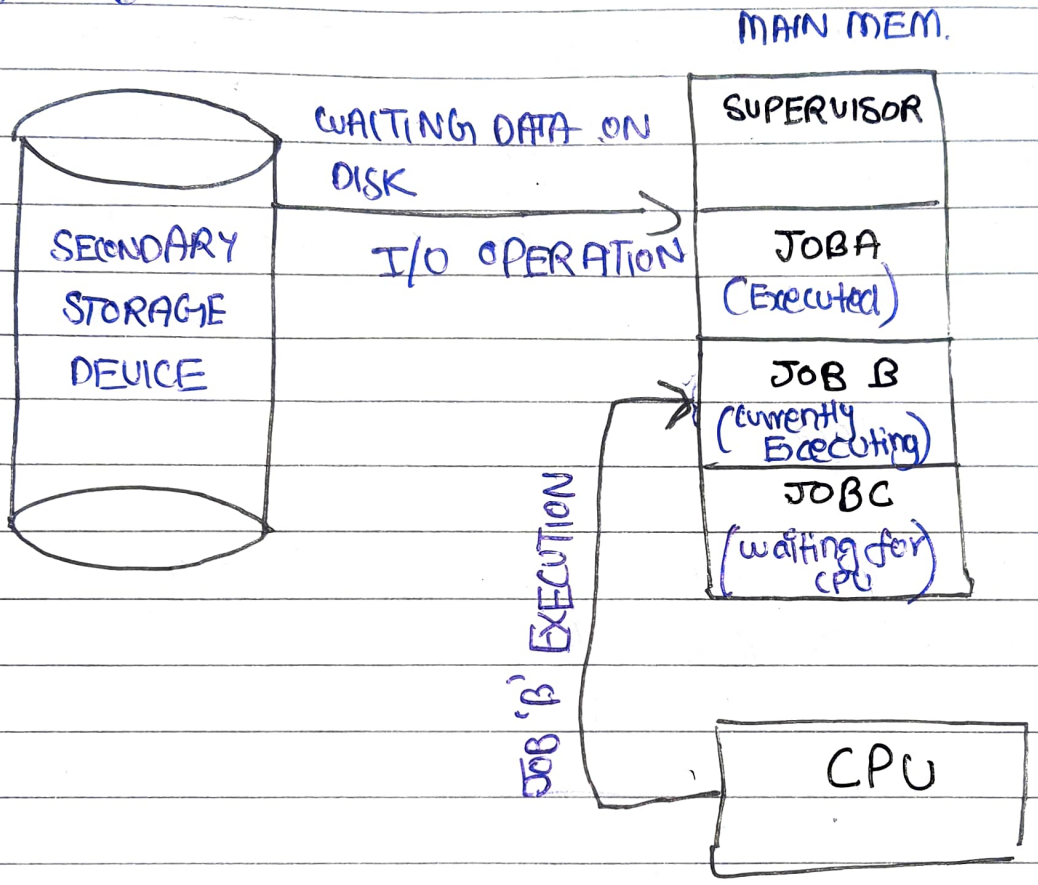
→ The main idea of multiprogramming OS is to maximize the use of CPU Time

→ If currently running process is performing an I/O Task (which doesn't need the CPU), then the OS may interrupt that process & give control to one of the other in-main-memory that are ready to execute. (It is known as Process Context Switching)

In this way, no CPU Time is wasted by the system waiting for the I/O Task to be completed, & a running process keeps executing until either it voluntarily releases the CPU or when it blocks for an I/O operation

→ ∴ the ultimate goal of multiprogramming is to keep the CPU busy as long as there are processes ready to execute.

* Note that in order for such a system to function properly, the OS must be able to load multiple programs into separate area of the main memory & provide the required protection to avoid the chance of one process being modified by another one



→ Other prob. that need to be addressed when having multiple prog. in memory is fragmentation as programs enter are leave the memory

→ Another issue that needs to be handled as well is that large programs may not fit in memory which can be solved using paging or virtual memory

→ If there are 'N' ready processes & all of those are highly CPU-bound (i.e they mostly execute CPU tasks)

& none or very few I/O operations) in the worst case one program might wait till the other 'N' processes to complete before executing.

- **Advantages**

1) High CPU utilization
2) Increased throughput
3) Shorter Response Times
4) Shorter waiting Time
5) Ability to assign priority to Jobs
6) Many Scheduling Algo. can be used-

- **Disadvantages**

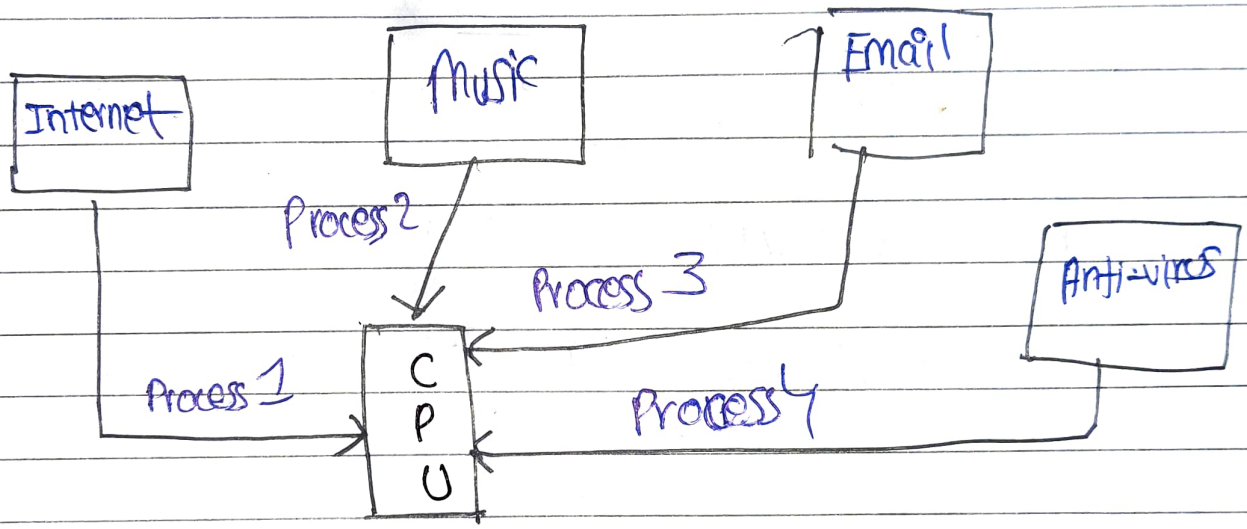
1) Scheduling Implementation is not Easy.
2) More Management is Required
   - (a) Memory Management
   - (b) Context Switching
   - (c) Scheduling.

MULTITASKING OS

OR FAIR-SHARE or TIME-SHARING OS

It is quite similar to MULTIPROGRAMMING but in this CPU is allocated to process for fixed timing i.e "TIME QUANTUM/TIME SLICE" after that CPU 'context switches' to another PROCESS.

In this User can Interact with the System (we can type a letter while Printing Task is going on)



- **Advantages**

(1) Shorter Response Time (2) Logical Parallelism
(3) CPU Utilization (4) Parallely using other Software

- **Disadvantages**

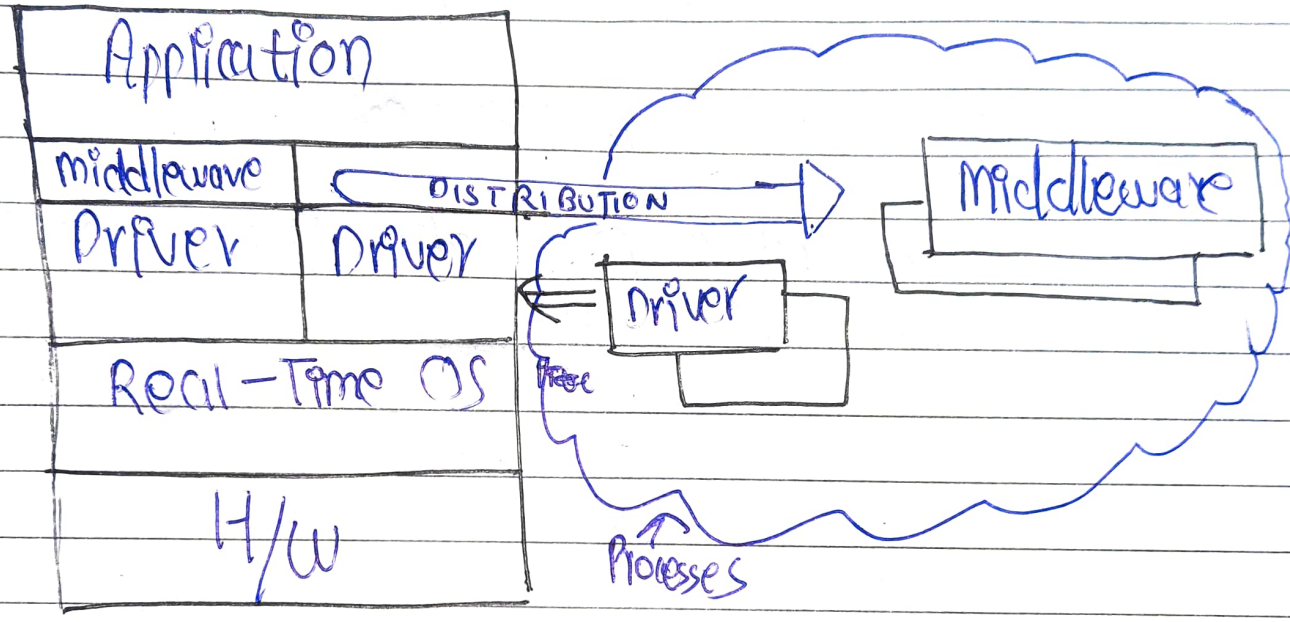
(1) Context Switching Overhead
(2) Can't be implemented on very Slow Processor Speed CPU

DIFF. B/W

MULTIPROGRAMMING	MULTITASKING
1) CPU is not allocated to processes for fixed Time	1) Time Quantum concept is used for CPU allocation
2) Process leaves the CPU at its will.	2) OS takes CPU from processes after fixed time is elapsed
3) Can't run other prog. simultaneously while CPU processing	3) We can do other tasks during CPU processing
4) Can be used on slow Processors	4) Can't be used on slow Processors
5) Can be used on low memory	5) Large memory Req.
6) Less used in PC	6) Highly Used in PC.
7) Less switching overhead	7) High switching overhead.

A Real-Time OS is defined as a data processing system in which the time interval required to process & response to inputs is so small that it controls the environment

→ Time Taken from input task to o/p Task is Response Time



⇒ A Real-Time OS is an OS in which each task performs its function within 'Prescribed Deadline' & failure to these deadlines may cause severe consequences so it has fixed Time Constraints & ∴ User convenience & Resource Utilization are secondary concerns

→ 2 types of RTOS

1) Hard RTOS →
→ Guarantees ontime completion of Critical Tasks

→ Secondary storage is limited or missing & data is stored in ROM.

→ No Virtual Memory

2) Soft RTOS

→ A critical Real-Time task gets priority over other tasks & retains priority until it completes

→ small delay can be accepted

→ mainly used in Multimedia, Virtual Reality, undersea exploration etc

- Advantages of RTOS

a) Maximum Consumption
b) Shorter Task Shifting Time
c) Best Memory Allocation
d) Error Free
e) Can be Used in Embedded Systems due to small size of program

- Disadvantages of RTOS

a) Use heavy System Resources
b) Expensive
c) Low Multitasking
d) Complex Scheduling Algorithms.
e) Severe consequences on missing Deadline
f) Low priority task delays