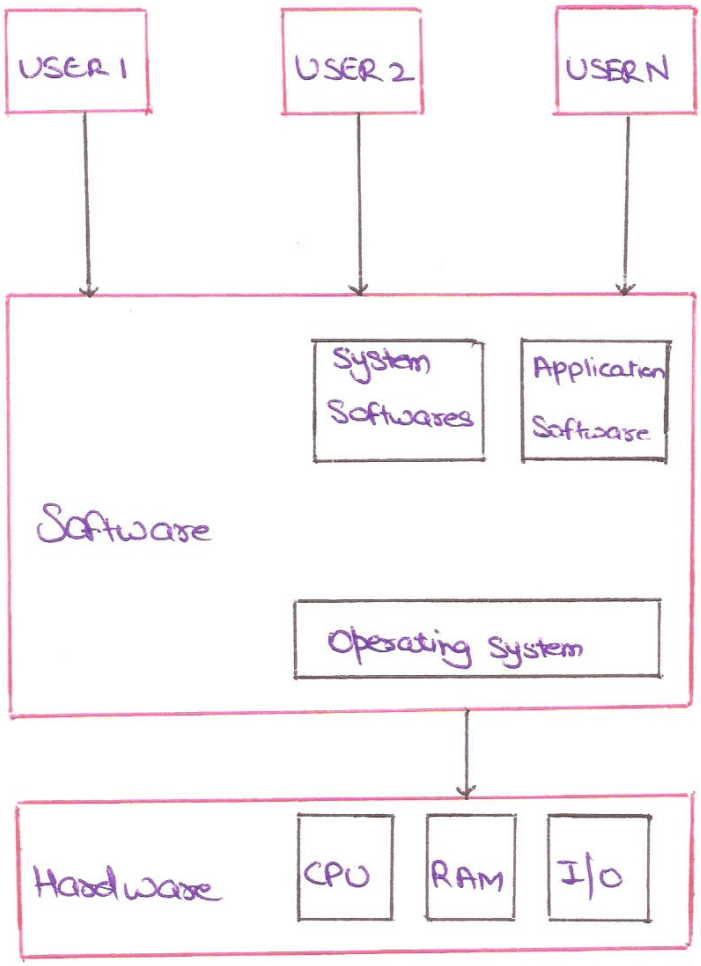
Operating System Functions and Characteristics :-

An operating System is a program that acts as an interface between user and the computer hardware and controls the execution of all kind of programs.



Functions of an Operating System :-

- Memory Management
- Device Management
- Security
- Job Accounting
- Coordination between other Software and users.
- process Management
- File Management
- Control over System Performance
- Error Detection aids

Memory Management :- It refers to Management of primary Memory. Main memory is a large array of words or bytes where each word or byte has its own address.

Main Memory provides a fast storage that can be accessed directly by the CPU. For a program to be executed, it must in the main memory.

- Operating System keeps track of primary memory, ie what part of it are in use by whom, what part are not in use.
- In multiprogramming, the OS decides which process will get memory when and how much.
- Allocates the memory when a process requests it to do so.
- De-allocates the memory when a process no longer needs it or has been terminated.

Process Management :- In this OS decides which process gets the processor when and for how much time. This function is called Process Scheduling. An Operating System does the following activities for processor Management:

⇒ Keeps tracks of processor and status of process. The Program responsible for this task is known as Traffic Controller.

⇒ Allocates the processor (CPU) to a process.

⇒ De allocates processor when a process is no longer required.

Device Management :- An operating System manages device Communication via their respective device-drivers. Os does

⇒ Keep tracks of all devices. The program responsible for this task is known as the I/O Controller.

⇒ Allocates the device in the most efficient way.

⇒ De-allocates devices.

File Management :- A file System is normally Organized into directories for easy navigation and usage. These directories may Contain files and other directories.

O/s does

⇒ Keep track of information, location, users, status etc. The Collective facilities often as known as File System.

⇒ Decides who gets the Resources.

⇒ Allocates the resources.

⇒ De- Allocates the Resources.

Security :- By Means of passwood and similar other techniques, it prevents unauthorized Access to programs and data.

Control over System Performance :- Records delay between Request for a Service and Response from the System.

Job Accounting:- Keeps track of time and resources used by various jobs and users.

Error Detection Aids:- Production of dumps, traces, error messages and other debugging and error detecting Aids.

Co-ordination between other Software and users:- Coordination and assignment of Compilers, Interpreters, assemblers and other Software to the various users of the Computer Systems.

Properties or Characteristics Of Operating System :

1) **Reliability** :- Reliability is that a system does exactly what it is designed to do.

At Software level OS must be more Reliable on detecting Virus or malicious code as they take control of the System for their own purposes by exploiting design or implementation errors in the operating System's defenses.

2) **Availability** :- It is the percentage of time that the System is usable. Some System crashes frequently, losing the user's work. Both Reliability & Availability are desirable. Availability is affected by two factors :- the frequency of failure called the mean time to failure (MTTF) - and the time it takes to restore a system to a working state after failure (for example, to reboot), the MTTR (Mean time to Repair)

Availability Can be improved by increasing the MTTF or reducing the MTTR.

3) **Ease of Management** :- OS must be easy to Management as these are lots of help & tutorials are available to rectify the errors.

4) **Security** :- As OS has administrative users that Can make changes to the Computer but for other user to make changes he has enter the password of administrator.

Associated Utilities :- O/s must have lots of Associated utilities available like windows O/s has lots of version and According to their version lots of free utilities are available like MS Office + MS SQL + Developing Languages.

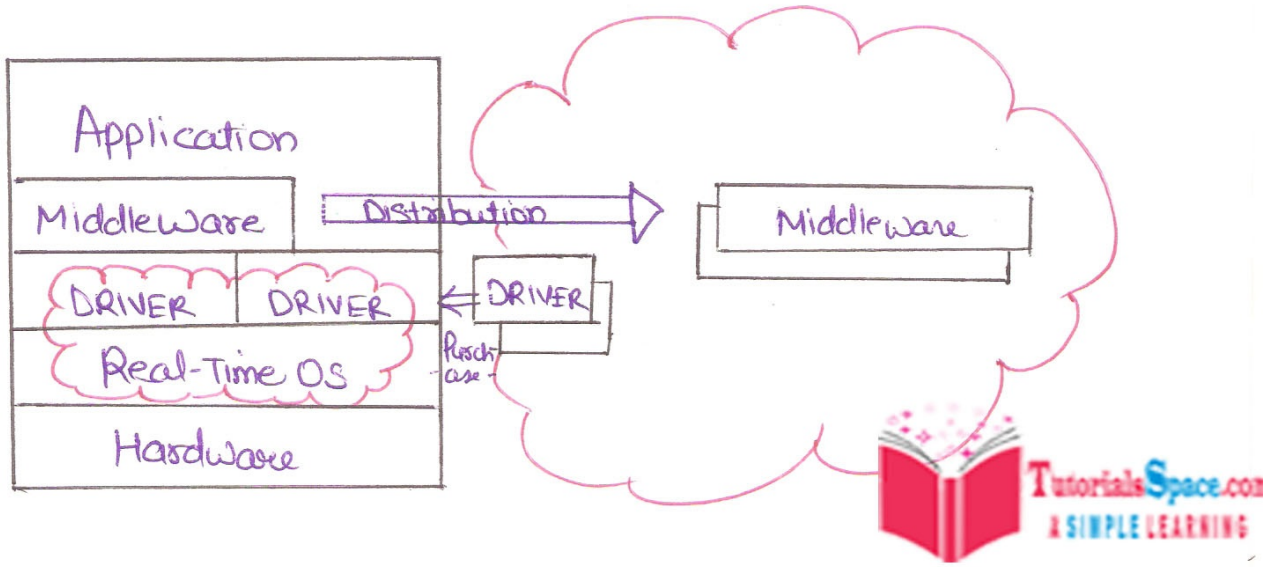
Cost :- O/s Cost should be accordance of its features like Windows has around £100 where Dos is free of charge as so is Linux

Support for the users :- O/s should have help section for users if any error, any problem occurs. like Windows has proper help section and websites. Where Linux has book in pdf format available and Dos has websites and forum for help as Dos is not much popular.

Real-Time Operating System

A Real time System is defined as a data processing System in which the time interval required to process and response to inputs is so small that it controls the environment.

Time taken from Input to output task is Response time.



→ A real-time operating System must have well-defined, fixed time Constraints, otherwise the System will fail.

For Example:- Scientific experiments, medical imaging systems, industrial control systems, weapon Systems, Robots, air traffic control System etc.

These are two Real-time Operating Systems:-

- Hard Real time System
- Soft Real-time System

Hard Real Time Systems:-

It guarantees the 'ontime' Completion of 'CRITICAL' tasks.

In this Secondary Storage is limited or missing and the Data is stored in ROM. In this Systems, Virtual memory is almost never found.

Soft Real Time Systems:-

A Critical Real-Time task gets priority over other tasks and retains the priority until it Completes.

These Systems have limited utility than hard Real-Time Systems. for Example:-

Multimedia, Virtual Reality, Advanced Scientific Projects like undersea Exploration and planetary Rovers etc.

Advantage:- • Maximum Consumption :- Gives more o/p while using all the resources and keeping all device active.

Task Shifting :- Less time is needed in shifting the task which is around 3 micro seconds.

Focus on Application :- focus on applications which are running and usually give less importance to the other application residing in waiting stage of life Cycle.

Used in Embedded System also :- Due to small size of program RTOS can also be used in embedded System like in transport and Others.

Error free :- RTOS is error free so it has no chance of errors in performing tasks.

24-7 System :- RTOS can be used for any applications which run 24 hours and 7 days b/c it do less task Shifting and give maximum output.

Memory Allocation :- Best managed in these.

Disadvantage :-

→ Use heavy System Resources → Expensive

→ Low Multitasking → Not easy to program

→ Complex- Algorithm → Low priority task : LPT not get time to run.

→ Device Driver and interrupt signals

→ Thread priority

Introduction
Distributed Operating System.

Distributed Systems use multiple Central processors to Serve multiple Real-time applications and multiple users. Data processing jobs are distributed among the processors accordingly.

Processors Communicate with one another through various Communication lines (Such as high-speed buses or telephone lines). These are refered as loosely coupled systems. or distributed Systems.

processors in a distributed system may vary in size and function. These processors are reffered as sites, nodes, Computers and so on.

The advantages of distributed systems are follows:-

→ Resource sharing facility:- by this an user at one site may be able to use the resources available at another.

→ Exchange of data by email:- Speedup the exchange of data with one another via Email.

→ One fails other works:- If one site fails in these System, the remaining Sites can potentially Continues operating.

→ Better Service to the Customers

→ Reduction of the load on the host Computer

→ Reduction of delays in data processing.

Methodologies for implementation of o/s Service System Calls :-

For performing any operation a user must have to Request for a Service Call which is also known as the System Call.

Or we Can Say

→ Programming interface to the Services provided by the OS.

→ They are typically written in a High-level Language (C or C++)

→ There are two modes in the operation of System which is user mode or System mode

In User mode → all user processes are executed.
In System mode → All privileged operations are executed

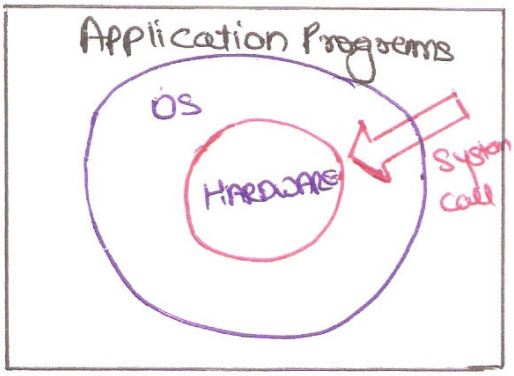
The user programs and kernal functions are being executed in their respective space allotted in the main memory partitions.

User mode programs need to exute some privileged operations, which are not permitted in user mode functions, user mode programs must use an interface, which forms the only permitted interface between user mode and kernal mode. This interface is called System Calls. So System calls is an interface b/w the user programs and the o/s

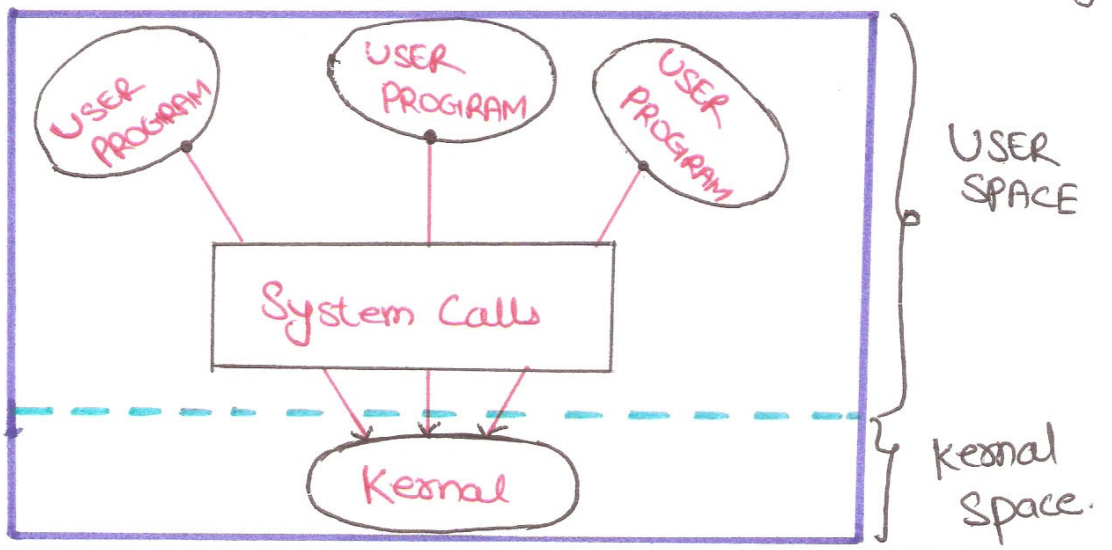
System calls expose all kernal functionalities that user mode programs require.

Basically the System call is an instruction that request the OS to perform the desired Operation that needs hardware access or other privileged Operations.

* System call generates an interrupt that Causes the OS to gain Control of the CPU. The OS then finds out the type of System Call and the Corresponding Interrupt Handler Routine is executed to perform the operation.



System Calls are inherently used for security reasons. Due to the use of System Calls, a user program is not able to enter into OS or any other's user's Region. Similarly, I/O devices are also safe



From any misuse by the users. Thus, through the use of System calls, Kernel, other user programs, and I/O Devices are safe and secure from malicious user programs

Making a System Call

Now System Calls are directly available and used in high level languages like C & C++. So it has become easy to use System Calls in programs.

For a programmer, System Calls are same as calling a procedure or function.

The difference between a System Call and a normal function call is that a System Call enters a Kernal but a normal function call does not.

Executing the System Call :-

There is a sequence of steps to execute a System Call. for this, there is the need to pass various parameters of System Call to the OS. For passing these parameters to the OS, three methods are used as follow :-

1) Register Method, wherein the parameters are stored in registers of the CPU.
2) If parameters are not in number, Compared to the size of registers, a block of memory is used and the address of that block is stored in Register
3) Stack Method, where in parameters are pushed onto the Stack and popped off by the OS

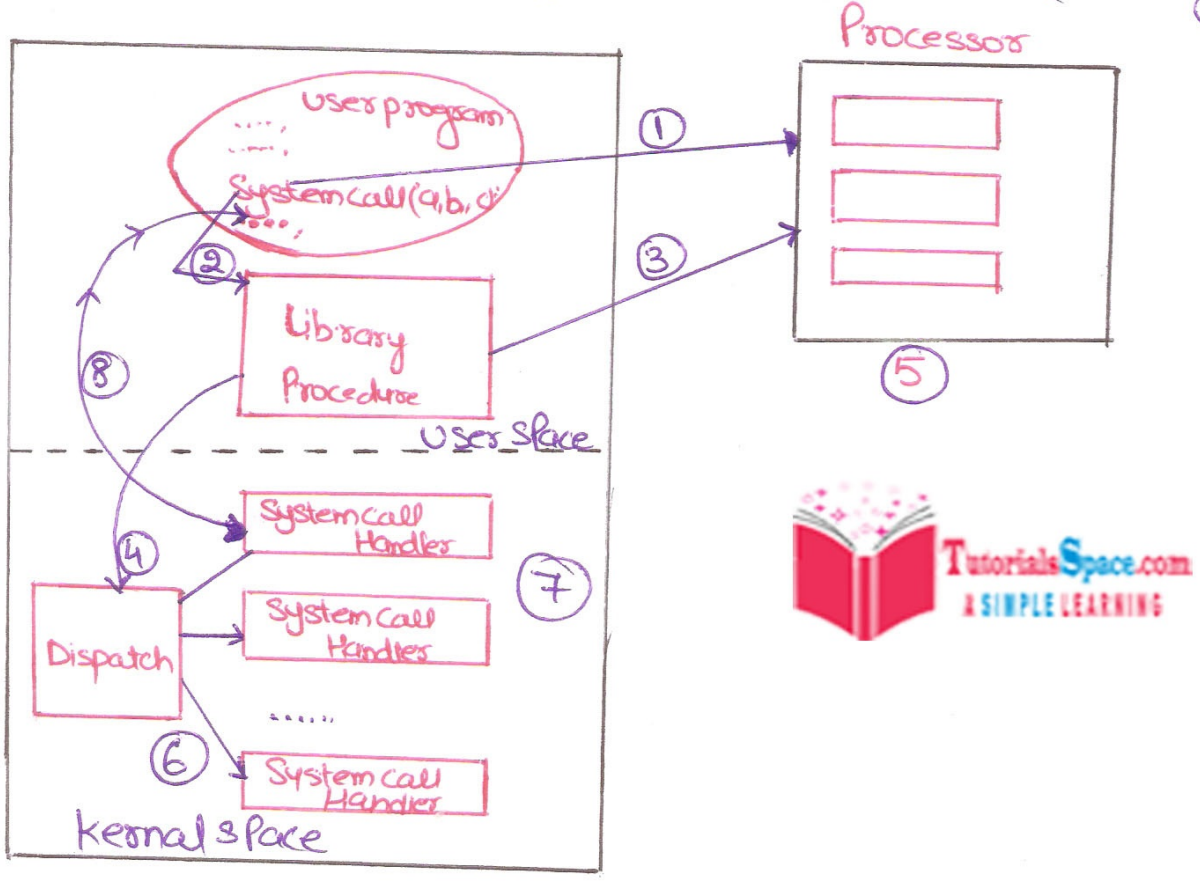
Sequence in which System Call is executed

1) In the User program when the System Call is executed, first of all, its parameters are pushed onto the stack and later on saved in processor Registers.

2) The corresponding library procedure for the System call is executed.

3) There is a particular code for every System Call by which the Kernal identifies which System call function or handler needs to be executed. Therefore, library procedure places the System call number in the processor Register.

4) Then the library procedure traps to the kernal by executing interrupt instruction. With this interrupt execution, the user mode switches to kernal mode by loading 'Program Status Word' (PSW) register to 0.



5) The hardware saves the current contents of CPU registers, so that after executing the system call, the execution of the rest of the program can be resumed.

6 → The kernal identifies the system call by examining its numbers and 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 program and it resumes its execution.

WaitForSingleObject(): my process waits for the newly created process

CloseHandle() : *Handle* is a utility that displays information about open handles for any process in the system. ：The **CloseHandle** function closes handles to the following objects:

- Communications device
- Console input
- Console screen buffer
- Event
- File
- File mapping
- Job
- Mailslot
- Mutex
- Named pipe
- Process
- Semaphore
- Socket
- Thread
- Token

The term **console** usually refers to a terminal attached to a minicomputer or mainframe and used to monitor the status of the **system**.

The **SetConsoleMode()** function can be used to set the input and output modes of the console's input and output buffers.

The **ioctl**() function manipulates the underlying device parameters of special files. In particular, many operating characteristics of character special files (e.g. terminals)

may be controlled with **ioctl**() requests. The argument *d* must be an open file descriptor.

GetCurrentProcessID() : Retrieves the process identifier of the calling process.

Settimer(): to use `SetTimer` API to call a function every X minutes. So, i have written this test code.

An operating system “pipe” is an abstraction which implements interprocess communications through an interface which behaves like an open file descriptor.

CreatePipe() : a pipe is a connection between two processes, such that the standard output from one process becomes the standard input of the other process.

The transformation of data from main memory to cache memory is called mapping.

File mapping is the association of a file's contents with a portion of the virtual address space of a process. The system creates a *file mapping object* (also known as a *section object*) to maintain this association.
A *file view* is the portion of virtual address space that a process uses to access the file's contents. File mapping allows the process to use both random input and output (I/O) and sequential I/O. It also allows the process to work efficiently with a large data file, such as a database, without having to map the whole file into memory. Multiple processes can also use memory-mapped files to share data.

CreateFileMapping():The first step in mapping a file is to open the file by calling the **CreateFile** function. To ensure that other processes cannot write to the portion of the file that is mapped, you should open the file with exclusive access. In addition, the file handle should remain open until the process no longer needs the file mapping object. An easy way to obtain exclusive access is to specify zero in the *fdwShareMode* parameter of **CreateFile**. The handle returned by **CreateFile** is used by the **CreateFileMapping** function to create a file mapping object.

The **CreateFileMapping** function returns a handle to the file mapping object. This handle will be used when creating a file view so that you can access the shared memory. When you call **CreateFileMapping**, you specify an object name, the number of bytes to be mapped from the file, and the read/write permission for the mapped

memory. The first process that
calls **CreateFileMapping** creates the file mapping object. Processes calling **CreateFileMapping**for an existing object receive a handle to the existing object. You can tell whether or not a successful call
to **CreateFileMapping** created or opened the file mapping object by calling
the **GetLastError** function. **GetLastError** returns **NO_ERROR** to the creating process
and **ERROR_ALREADY_EXISTS** to subsequent processes.

MapViewOfFile(): Maps a view of a file mapping into the address space of a calling process.

shmget(): shared memory segment : . In order to create a new message queue, or access an existing queue, the **shmget()** system call is used.

InitializeSecurityDescriptor()

security descriptor

A structure and associated data that contains the security information for a securable object. A security descriptor identifies the object's owner and primary group. It can also contain a DACL that controls access to the object, and a SACL that controls the logging of attempts to access the object.

SetSecurityDescriptorGroup():

The **SetSecurityDescriptorGroup** function sets the primary group in formation of an absolute-format security descriptor, replacing any primary group information already present in the security descriptor.

Chmod() : change mode: to change the file mode

The **chmod**() function changes permissions of the specified file.

Unmask(): In computing, **umask** is a command that determines the settings of a mask that controls how file permissions are set for newly created files.

chown() — Change the owner or group of a file or directory - IBM

Methodologies for implementing o/s services :-

System program :-

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

File Management :- these programs create, delete, copy, rename print, dump, list, and generally manipulates files and directories.

Status information :- Some System programs simply ask information related to system like Date|time| Size of disk| No of users.

File Modification :- Several text editors may be available to create and modify the content of files stored on disk or other storage devices. These may also be special commands to search contents of files or perform transformations of text.

Programming Language Support :- Compilers, assemblers, debuggers and interpreters for Common programming Languages (such as C, C++, Java & more

are often provided to the users with the operating System

program loading and execution :- Once a program is assembled or compiled, it must be loaded into memory to be executed. The System may provide absolute loaders, relocatable loaderss, linkage editors, and overlay loaders. Debugging Systems for either higher-level languages or machine language are needed as well.

Communications :- These programs provide the mechanism for creating Virtual connections among processes, users and computer systems. They allow users to send messages to one another's screens, to browse web pages, to send electronic-mail messages, to log in remotely, or to transfer files from one machine to another machine.

Some System program supplied with O/s are Text formatters, Spread sheet, Compilers, Web Browser, database System, Games and many more...

INTERRUPT MECHANISM

Interrupt is a mechanism by which Computer Components, like memory or I/O modules, may Interrupt the normal processing of the processor and request the processor to perform other specific action.

According to the source where they are generated, interrupts may be categorized into four classes:-

1) program : Generated by some condition that occurs as a result of an instruction execution, such as
   - Arithmetic Overflow
   - Division by zero
   - page fault
   - Invalid instruction
   - Outside memory space Reference

2) Timer : Generated by a timer with in the processor. This allow operating System to perform Certain functions on a regular basis

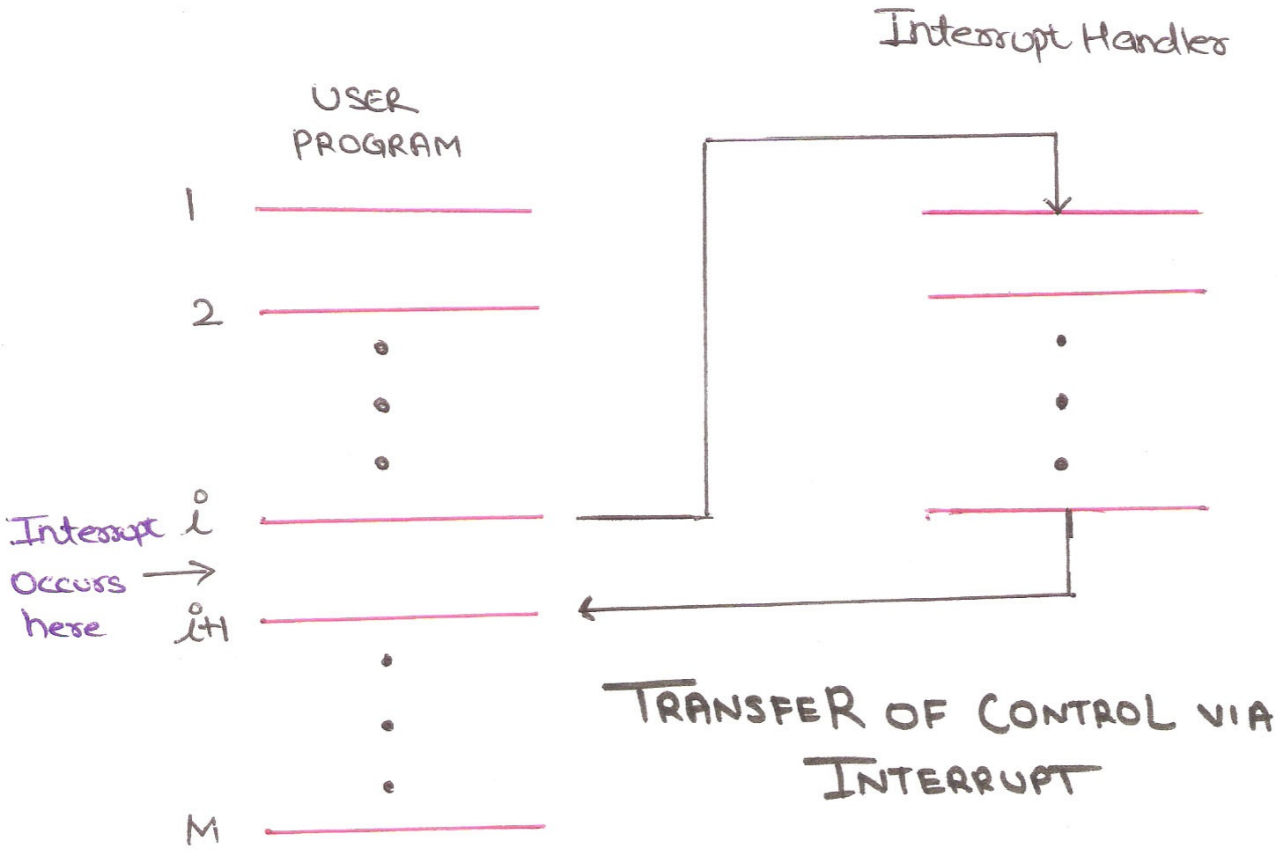
3) External or I/O : by an I/O Controller, I/O devices tells the CPU that an I/O request has Completed by sending an interrupt signal to the processor.

4) Hardware failure : Generated by a failure, such as power failure or memory parity error.

INTERRUPT MECHANISM

INTERRUPTS & INSTRUCTION CYCLE

For the user program, an interrupt suspends the normal sequence of execution. When the interrupt processing is completed, execution resumes. Thus the user program does not have to contain any special code to accommodate interrupts.

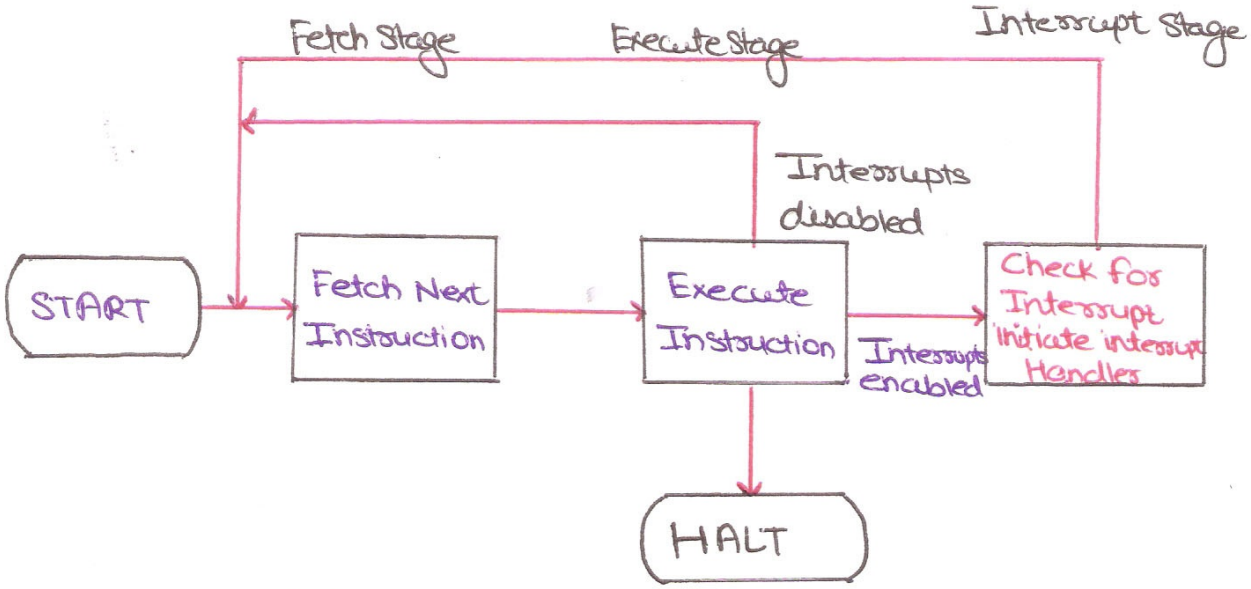


TRANSFER OF CONTROL VIA INTERRUPT

To accomodate interrupts, an interrupt stage is added to the Instruction Cycle. In the interrupt stage, the processors checks to see if any interrupts have occurred, indicated by the presence of an interrupt signal.

If no interrupts are pending, the processor proceeds to the fetch stage and fetches the next instruction of the current program.

→ If an interrupt is pending, the processor suspends execution of the current program and executes an 'interrupt-handler' routine.



Instruction Cycle with Interrupts

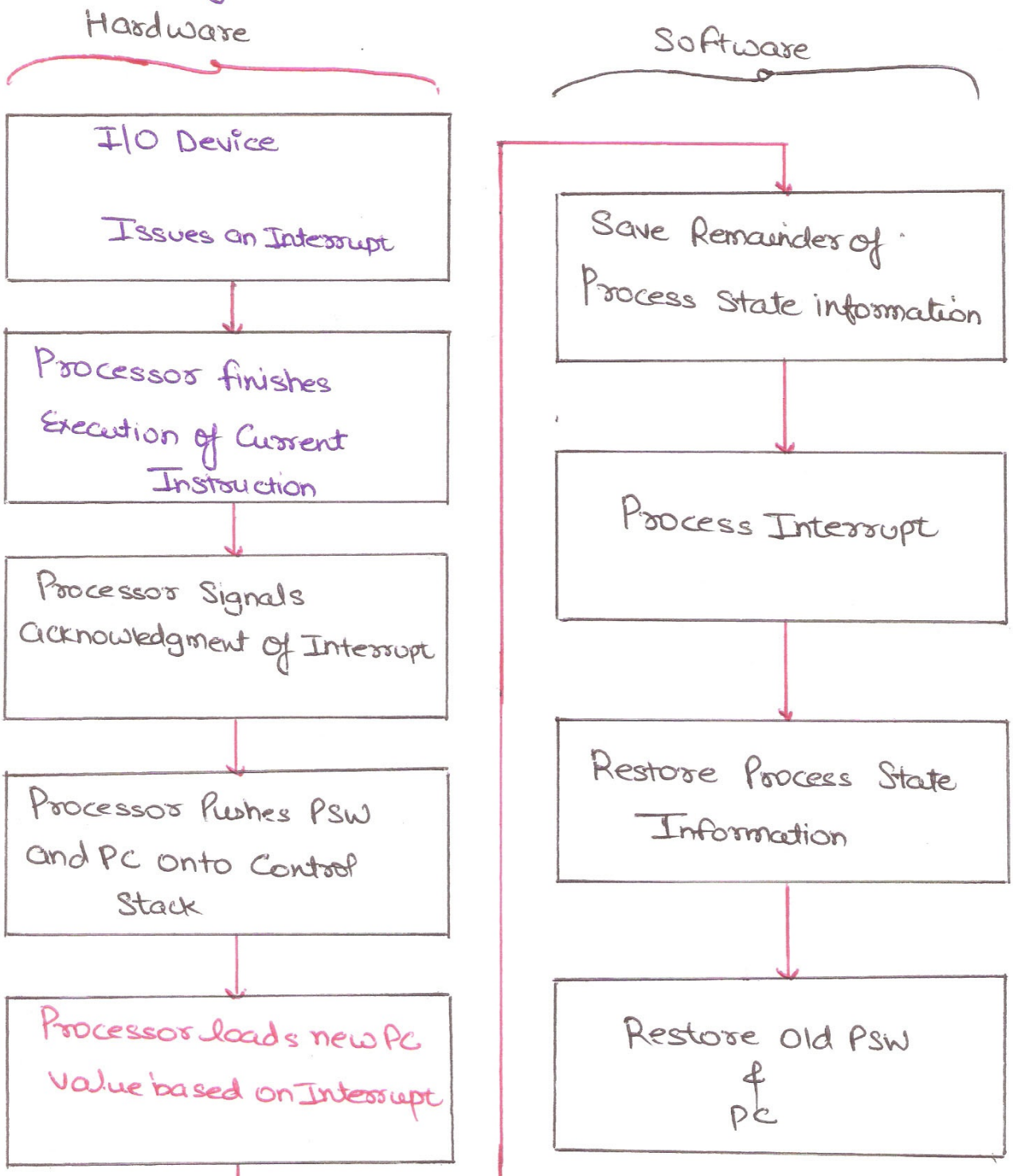
This routine determines the nature of the interrupts and performs what ever actions are needed.

Interrupt Processing

Interrupt processing procedure

→ I/O device issues the interrupts

→ The processor finishes the execution of an instruction before responding to the interrupt.



→ The processor checks for an interrupt. If there is one, it then sends an acknowledgement signal to the I/O device that issued the interrupt. The signal allows the device to remove its interrupt signal.

→ To switch to run interrupt handler, information about the current program is stored, so that its execution may be resumed later including Program Status word (PSW) and Program Counter (PC)

→ The processor loads the program counter with the entry location of the interrupt handler. A typical case is there are a set of routines, each for one type of interrupt, or each for one device.

→ The interrupt handler may continue to save other information that is considered as part of process state.

→ The handler performs the interrupt processing.

→ When handler p finishes, the saved register values are stored into the registers that originally hold them when interrupt handler returns.

→ Finally PSW and PC values of the interrupted program are restored, thus the program may continue to execute.

Operating System : Processes

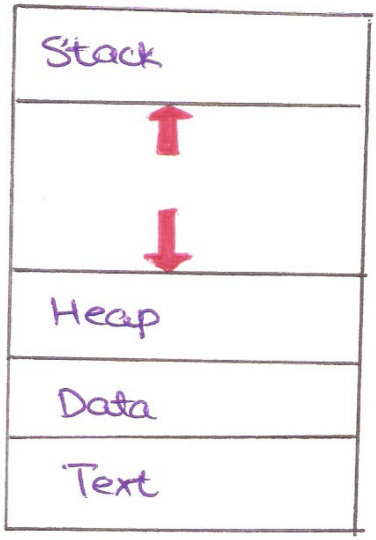
What is a process?

A process is a program in execution including the current values of the program counter, registers and variables.

The difference between a process and a program is that the program is the group of instruction where as the process is the activity.

We write our computer programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program.

When a program is loaded into the memory and it becomes a process, it can be divided in to four sections- Stack, Heap, text and Data.



Main Memory

S.NO	Component & Description
1	Stack: The process Stack Contains the temporary data such as Method/function, Parameters, return address, and local variables.
2.	Heap :- This is dynamically allocated memory to a process during its runtime.
3.	Text : This includes the current activity represented by the value of Program Counter & Contents of the processor's Registers.
4.	Data: This section contains the all global and static variables.