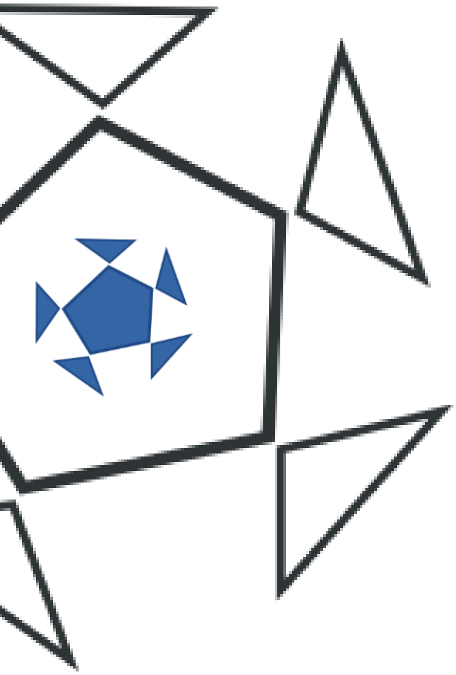
Raza's Simplified

Chapter 1



INTRODUCTION

In this chapter contain:

- OS Defined
- Machine Hardware
- Machine Cycle
- Execution in Multimode
- OS Structure
- OS Types
- Processing Traps and interrupts
- The kernel
- Boot Process

- ÷ OS Defined
- ÷ Machine Hardware
- ÷ Machine Cycle
- ÷ Execution in Multimode
- ÷ OS Structure
- ÷ OS Types
- ÷ Processing Traps and interrupts
- ÷ The kernel
- ÷ Boot Process

OS Defined

It is the collection of many small and large programs which govern computer operations. They control applications, they control users, they manage hardware, they secure the computer, they manage memory, they provide user interface and much more…. You can call it the soul of computer system and hardware its body-

Some OS are:

1. DOS
2. Windows
3. LINUX
4. UNIX
5. Solaris
6. IOS
7. Android
8. Chromium
9. UBUNTU

Machine Hardware

If computer was a human then CPU was surely to be called the heart and brain of it. It is the brain of computer because it has a component called CU which processes all instructions and data. It is the heart because a special circuit called clock inside it generates continuous pulses which synchronize the execution of an instruction as done by a real heart by pumping the blood in the body.

CPU components include CU[Control Unit], ALU[Arithmetic and Logic Unit], clock and special purpose short memory called **registers**.

Registers are the small sized, high speed temporary memory locations inside CPU which store data and instructions being executed.

All the I/O devices and memory are connected to the CPU through a bus.

A bus is link through which data flows between computer components. The CPU acts on instruction which it fetches/gets from memory and then directs the I/O devices according to that instruction.

The CU of CPU follows a basic set of actions while executing an instruction. No matter what the type of instruction is, the CU always performs these steps. It continues in a loop called machine cycle.

Following registers play a key role during this cycle:

Program Counter (PC)

Itstoresthe address of the next instruction to be fetched from memory and executed.

CU works by fetching the instruction at the current address in PC. After which the PC is changed to point to the next instruction.

Instruction Register (IR)

It stores the instruction recently fetched from memory. Then it is executed.

Machine Cycle

The machine cycle goes on in the following fashion:

1. Before executing an instruction, it is fetched from a memory location given in the PC. All instructions are available in memory for the CPU.
2. The fetched instruction is put in IR. Where it waits to be decoded.
3. The value in PC is adjusted and incremented so it would point to the next instruction.
4. The instruction is decoded. This is done by the CU. Decoding gives explanation about the instruction. This is done by examining the special bits. These bits contain **opcode** [operation code] of the instruction. Opcode specifies an action to perform.
5. An instruction may specify some data to do its operation, the CPU fetches needed data from memory. These data values are called **operands** and after fetching them, they are stored in other registers.
6. Now the operation given in opcode is performed on operands. The types of operations include:
    a. **Movement**: a value or data is transferred from one place to another. It can move from:
        i. memory to registers
        ii. registers to memory
        iii. memory to memory
        iv. registers to registers.
    b. **Computation**: an operand is given to the ALU to perform some calculation.
    c. **Conditional branch**: means conditional execution. It happens when a condition is encountered during program execution. If that condition is true, then PC is reset. Now, PC has the address of the new instruction which will be run for that condition. Unconditional branching is a special case in which the condition is always considered to be true.

d. **Procedure call:** means execution of a sub program/function/method, which would be executed from somewhere else. To execute this, the state of PC is saved, and it is modified to point to a new location. Now PC has the start address of the subroutine. When the procedure execution is finished, a branch instruction is executed which resets the PC to point to the old saved state. In this way the execution of the old program resumes.

e. The value of PC would be saved in a special register (PSW), or in memory or a special data structure called stack.

f. **Input/output**: this instruction directs an operation between CPU and I/O devices. Mostly its result is an I/O operation.

7. 7- The output is stored in memory. This step is optional. It may not be needed every time. E.g. the data movement instruction will not return a result.

Execution in Multimode

There are two sets of instructions, one that an application program can execute, second which only an OS can execute. The execution of OS specific instructions must be protected from an application program. This is done using different modes of execution. Most OS support only two modes, kernel mode and user mode. To record the mode of execution, a special bit in PSW is used. If some application program attempts to execute a privileged action, while running in user mode, a trap will be the result. The privileged instructions include:

- specific instructions reserved for OS
- read and write access to certain registers
- read and write access to I/O devices.

During the operations of a program, a system can enter kernel mode (also called privileged mode or supervisor mode) by: -

First is issuing an SVC (supervisor call). It is a special type of system call. It is similar to procedure call but shifts the execution mode to supervisor mode. This call does not have a branch address. Instead, the branch address is obtained from a special vector, much similar to interrupt vector. Every SVC has a unique number, called index or operand of SVC. The OS matches this number in the vector and finds branch address. Now the execution continues from that address.To keep the things working properly, the OS controls this vector. After an SVC the mode shift, vector searching and branching occur.

Second and third mode shifting methods are the traps and interrupts. They set the mode to kernel. When mode shift happens, it jumps to OS kernel entry point.

Application programs cannot be executed in kernel mode orchange the mode. So they continue executing their own code.

There is an interesting case of UNIX OS which has a special user called **superuser**. This user can read/write it. It can also terminate a running process. But it does not mean that a process or application run by superuser is executed in kernel mode. Actually the superuser is given special and above average rights. But an application running under it always uses SVCs to get its work done from the OS.

OS Structure

A virtual machine is a conceptual machine. It is an environment facilitating a user process where many SVCs can be issued and processed.

When an application is run, the OS provides a virtual machine for that application.

The instruction set provided by the hardware is raw and limited. The SVCs of OS expand these capabilities by providing new abstractions like processes and files and virtual memory. They build an environment where a process runs smoothly.

An application program cannot access any hardware directly. They must request the OS to access hardware for them. Also, there are many applications running parallel and requesting the same hardware resources. In such a case the OS must:

- fulfill the needs of individual applications
- it must also protect the hardware from direct access
- manage the concurrent requests in a manner that applications' integrity remains intact.

OS Functions

There are following areas of OS functions: -

Managing Processes

When a program is executing, it is called a process. A process has its code, data, resources and flows of execution (called threads). It is the OS which allocates resources to a process. Then provides and manages SVCsof it. If there are multiple processes running together on the same machine, then OS must protect them from each other's interference, and manage a separate virtual environment for each process.

Memory Management

In a very simple scheme, the memory is shared at least between OS and an application program.

But today's multitasking systems run many processes simultaneously. So the memory must be shared between all these processes. OS manages memory by:

a. Allocating it to the processes

b. Recording which process has which area in memory

c. protecting a process' memory from other processes

File System Management

A file system is a set of objects used to record and manipulate stored data.

A computer is a data processor, which gives information as output. This information must be saved somewhere and transmitted. It may be processed again if needed. The storage is done on some persistent media including disks. A file system organizes many objects to facilitate file management functions. The OS must provide functions and procedures to manage these objects

Device Management

A hardware device cannot be allowed direct access by an application. I/O devices are necessary for communication between users and computers. To fulfill this need, SVCs are implemented and provided to the applications. OS tries to manage these resources in an efficient and concurrent manner.

OS Types

Batch Processing OS

A batch OS executes group of similar non-interactive jobs. Non interactive job , once executed, does not allow user interaction.

In early days of computing, the OS were simple. A job was simply a program and its input data. A programmer used to manage its jobs on some media. This media was mostly paper tape or a punched card. Then this job was given to the computer operator who would gather similar jobs in a batch. Then this batch was given to the computer. Computer processed the batch. It took minutes or even hours to process a job. Once a job was executing, no other job could be entertained. Executing job had full control over the machine. An OS supporting this type of job execution is called ***batch processing OS*.** This systemprovidesleast functionality. They are not interactive and when a job is waiting for an I/O, the time of machine is wasted. But such systems are extremely simple as they do not need to manage sharing of resource among different processes.

[An example of batch processing is the way that credit card companies process billing. The customer does not receive a bill for each separate credit card purchase but one monthly bill for all of that month's purchases. The bill is created through batch processing, where all of the data are collected and held until the bill is processed as a batch at the end of the billing cycle.

http://www.webopedia.com/TERM/B/batch_processing.html]

Multiprogrammed Batch OS

On these systems, all jobs are read and stored in a job pool. This job pool is managed on some disk. When an executing job is unable to run due to a pending I/O or something else, it is unloaded and another job is selected and run. This scheme increases system efficiency on one hand, and on the other, the system complexity also increases. These systems too do not allow user interaction. The data must be given in both types of batch systems in the beginning. And later nothing can be input. They are suited for the systems printing payrolls and paychecks and alike. For modern systems, user interaction is very necessary. This is done in time shared OS.

Time Shared OS

These systems are highly interactive and suit the needs of today's interactive and user focused computer industry. They allow smooth user interaction with the computer. And the system responds quickly to the user input. Such a system is running many processes at a time. This is called multitasking. The concurrent running processes share all resources and OS manages this sharing. It is achieved by continuous and quick transfer of control between processes. Every process is executed for a short time and then the next one is executed.

Network OS

Today's computers have support for networking. In networking a computer is not a stand alone machine, rather it works in coordination with other machines. This does not affect the basic function of the OS: process, memory, hardware and user management. To transfer the network traffic, a communication device is used. To provide security and reliability in communications, protocols are used.The support for these protocols is built directly into the OS. These systems are also time shared. In some cases, these systems also support batch processing.

Real Time OS

These systems are capable of completing a task within a given time duration.

In common work environment, the user expects the task be done as early as possible by the computer, but a rigid time constraint is not given.

In real time OS, the quality of job done depends upon (a)the completion of job, and (b) time taken to complete it. If it is not done in a given time, the results would be catastrophic.

For example, if the temperature in a nuclear reactor begins to rise, then the computer must initiate a cooling system in a matter of few milliseconds. Otherwise it will blow the whole reactor.

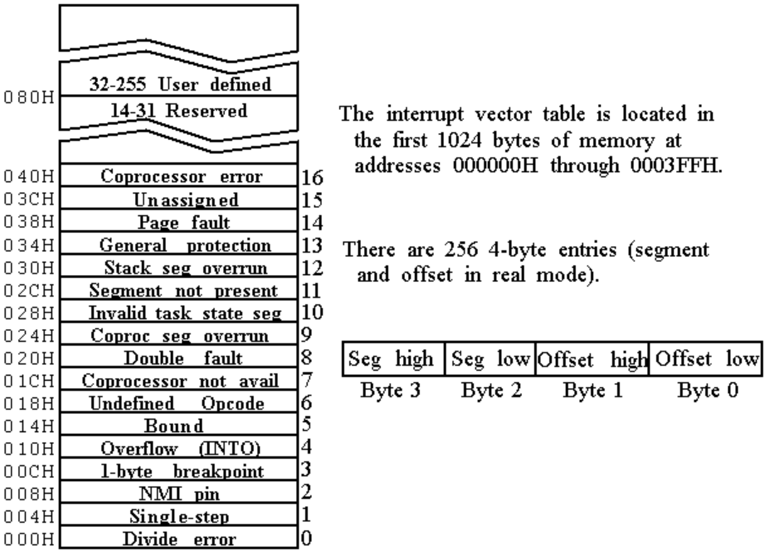
In real time OS, the sharing of resources is highly discouraged to avoid delays in processing. The low level access to hardware is encouraged to reduce response time. So, the principles governing the design of a general purpose time share OS are much different from those governing the design of a real time OS.

These OS are used in industry, production systems, machinery, weapon systems, auto pilots, navigation system of satellites etc.

Distributed OS

This is a sophisticated OS in which a single large job is divided into smaller chunks. These chunks are given to different physical machines to reduce processing time. This is different from a network OS. On a network OS every machine manages its own resources and processes and communication is done using a communication device making network I/O. Whereas on a distributed OS, all OS running on all machines join together to complete a single task. They manage the resources on all computers together. This collective single distributed OS manages the work and resources on the whole network.

Processing Traps and Interrupts



080H	32-255 User defined	
	14-31 Reserved	
040H	Coprocessor error	16
03CH	Unassigned	15
038H	Page fault	14
034H	General protection	13
030H	Stack seg overrun	12
02CH	Segment not present	11
028H	Invalid task state seg	10
024H	Coproc seg overrun	9
020H	Double fault	8
01CH	Coprocessor not avail	7
018H	Undefined Opcode	6
014H	Bound	5
010H	Overflow (INTO)	4
00CH	1-byte breakpoint	3
008H	NMI pin	2
004H	Single-step	1
000H	Divide error	0

The interrupt vector table is located in the first 1024 bytes of memory at addresses 000000H through 0003FFH.

There are 256 4-byte entries (segment and offset in real mode).

Seg high	Seg low	Offset high	Offset low
Byte 3	Byte 2	Byte 1	Byte 0

Well, let us see how our OS and CPU respond to user inputs and errors!

Normally a CPU goes on executing its assigned task smoothly. It never leaves execution of such a task until a trap or an interrupt occurs. Both situations require attention of the CPU. So the CPU leaves behind its own current task and serves them.

A trap is an indication of some error in system. Such errors include:

1. division by zero
2. an attempt to access memory locations which do not exist
3. an attempt to access memory locations which are restricted
4. Attempt to execute a privileged instruction.

An interruptis a situation when an I/O device has issued a signal to CPU requesting its attention. This signal goes to the CPU and it almost immediately responds. It first completes an instruction in progress and does not fetch the next one. Then it checks the need of the I/O device that requested CPU time.

The response of CPU to a trap or an interrupt requires the CPU to save its current state. It must save the values of many registers including program counter (PC) so that it could resume the execution of this task once the trap or interrupt has been served. Other information is also saved. A register is dedicated for this purpose on many machines. This register is termed as PSW (Program Status Word). Also, PC and other registers must be loaded with the new values for the trap or interrupt routine. Then CPU determines the address to start execution of this trap or interrupt. It is done using system's hardware architecture.

To do this, a data structure named interrupt vector is maintained. This vector has index and memory location fields. Every trap or interrupt is given a unique index. When an interrupt or trap occurs, the CPU jumps to this interrupt vector and presents the index of underlying trap or interrupt. The index of interrupt vector is matched and control jumps to the specified memory location. So, the execution resumes at the new memory location.

Here PSW plays an important role. It stores the priority level of CPU. Every interrupt or trap also has a priority level. The CPU will service only those traps or interrupts which have higher priority than CPU. In many cases, the CPU will need to lower its priority so that a lower priority trap or interrupt gets executed. To change the priority of CPU, certain bits of PSW are reset.

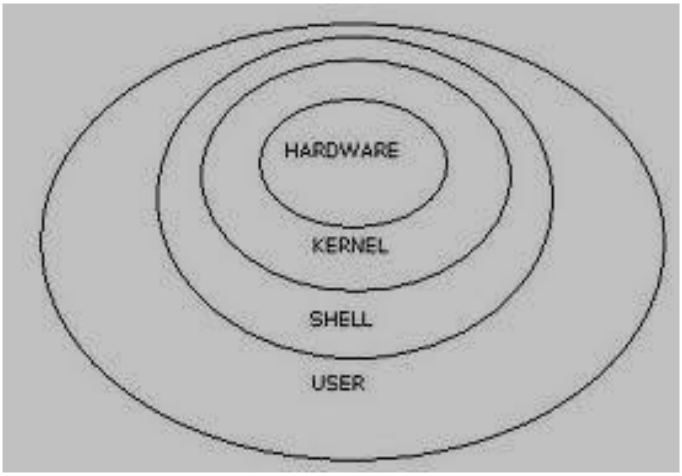
One thing to understand about traps and interrupts is their occurrence behavior.

We call traps synchronous and interrupts asynchronous.

Traps are synchronous because keeping the state of machine, data and program same, the trap will occur at exactly the same point in program execution flow.

But interrupts, which are coming form I/O devices, are based upon the timings and input of the user, which is unpredictable. That's why it is difficult to debug those errors which are influenced by relative timing of the interrupt. Such traps are also difficult to repeat.

The kernel



Kernel is the code that executes when the system has switched into kernel mode. It is not a single program. It is a complete library of procedures and one of its procedures executes when the hardware is executing in kernel mode. When a trap, interrupt or an SVC is issued, a procedure from kernel is executed as a response to it. When the execution of the procedure is over, the mode is reset to user mode and execution resume. Now the executing program may or may not be the one which caused the switch to the kernel mode. For example, a program which caused a trap would have been terminated as a result of kernel procedure execution due to some possible situation in the system

All the areas organized for OS are also managed for the kernel. For example, device management functions are called when an interrupt occurs. A memory management code is run when a trap is generated by memory management hardware. These and possibly other situations, which cause the switch to kernel mode, generate an SVC which resets the mode. When the service to an SVC is ended, the process scheduler is called prior to switching to the user mode to determine which process to run next.

Kernels are classified as being monolithic and micro.

Monolithic kernels are Large and have Full OS Functions.

A micro kernel has only essential OS features, which include memory management, interprocess communication, handling traps and interrupts. Some high level services such as file system management are not provided into micro kernel. They are accessed using some server process. If an application needs file system facilities, the kernel will provide them by executing a server process.

This scheme of providing a micro kernel with a server process is highly beneficial, especially in networked environments. It gives flexibility and the following advantages: -

If a new server process is needed it can be created and run without restarting the computer. So the number of server processes can be increased or decreased as needed.

The micro kernels have small size. So, their migration and implementation to new systems is easy.

On distributed systems, where a single task is executed on many systems, micro kernels are extremely helpful. A process running on a local or on a remote machine is run efficiently because micro kernel utilizes the native routines of the interprocess communication.

There are some services, which are hard to classify as being user level or OS level. For example, print spooling. It is true that the distinction between many system programs and OS is becoming poorly defined as more and more services provided by the system programs are now being provided by the OS. But some experts argue to define them as being part of OS device management functionalities.

Boot Process

Let us suppose that you go to your desk and turn on your PC. It will become alive and a beautiful desktop will appear. But how did all this happen? What magic did the power button cast on the dead hardware to bring it to life.

When power is turned on, the programs in ROM are read and copied to RAM. The CPU executes them. Some checking and diagnostic functions are performed on other hardware. Later, a program called stage 0 boot program is loaded and executed.

The function of stage 0 boot program is to find a boot device on PC.???? Such a device can be a floppy, hard disk, a CD ROM, and on modern computers it can be a USB flash or even a DVD. Network boot is an option too.

When a boot device is located on PC, its first sector is read and loaded into main memory. This first sector is call boot sector. The size of it is 512 bytes. This sector contains a stage 1 boot program. When once the boot stage 1 boot program is in memory, the stage 0 boot program branches to it and its execution begins.

But wait; there can be an error… the boot sector may not contain a stage 1 boot program. Then? If the stage 0 boot program continues and tries to copy from unset and uninitialized boot sector, some random and unrecognizable bits will be copied to the memory. So how to prevent this situation? Simple: before executing the bits received form the boot sector, the stage 0 boot program will check the end of boot sector for a special pattern of bits name "Magic bit pattern". The presence of these bits ensures the integrity and availability of stage 1 boot program.

Next stage is to load the OS from boot device. This is done in two ways. In some systems, the stage 1 boot program can do it. On others some boot strap programs are loaded in a sequence and executed, and they load the OS. Whatever the method is chosen out of these two, the OS is finally loaded into memory and it is ready to run. At this stage, the last boot program transfers control to the OS's initialization entry point and the full system begins to execute

The situation is slightly different for operating systems loading from harddisk. The boot sector of harddisk also contains a partition table. This table has information about partitions on harddisk. The partition is a division on harddisk, which acts as a separate harddisk unit. There are three types of partitions: Primary, Extended and Logical. A system can have four primary partitions at maximum. The partition table contains, along other information, the start and end points of each partition. This partition table is also loaded into memory. A specific partition is marked as being active in the table. The stage 1 boot program reads the first sector of the active partition and copies in into memory. This sector contains a stage 2 boot program which works with the OS present in the same or other partition. It finds and loads the OS from harddisk partitions.

When all this has finished, the OS is in memory and ready to run in the kernel mode. No user interaction is permitted. Many data structures including interrupt vectors, system registers and devices are initialized as needed.

When OS is ready, special processes are created and run that offer OS services to users. Some processes, which can initialize system programs, are also created and run.

On DOS, this is very simple. Only a command interpreter is run to take commands from user and parse them to OS

On complex systems, a special program *init* is run which creates more processes according to a database, containing configurations.

After all this, the kernel mode is switched to the user mode. And now the OS is ready to serve the users.

Well this was simple. Isn’t it? Think and rethink.