Computer Architecture

&

Operating System Fundamentals

Contents

Introduction to Computer

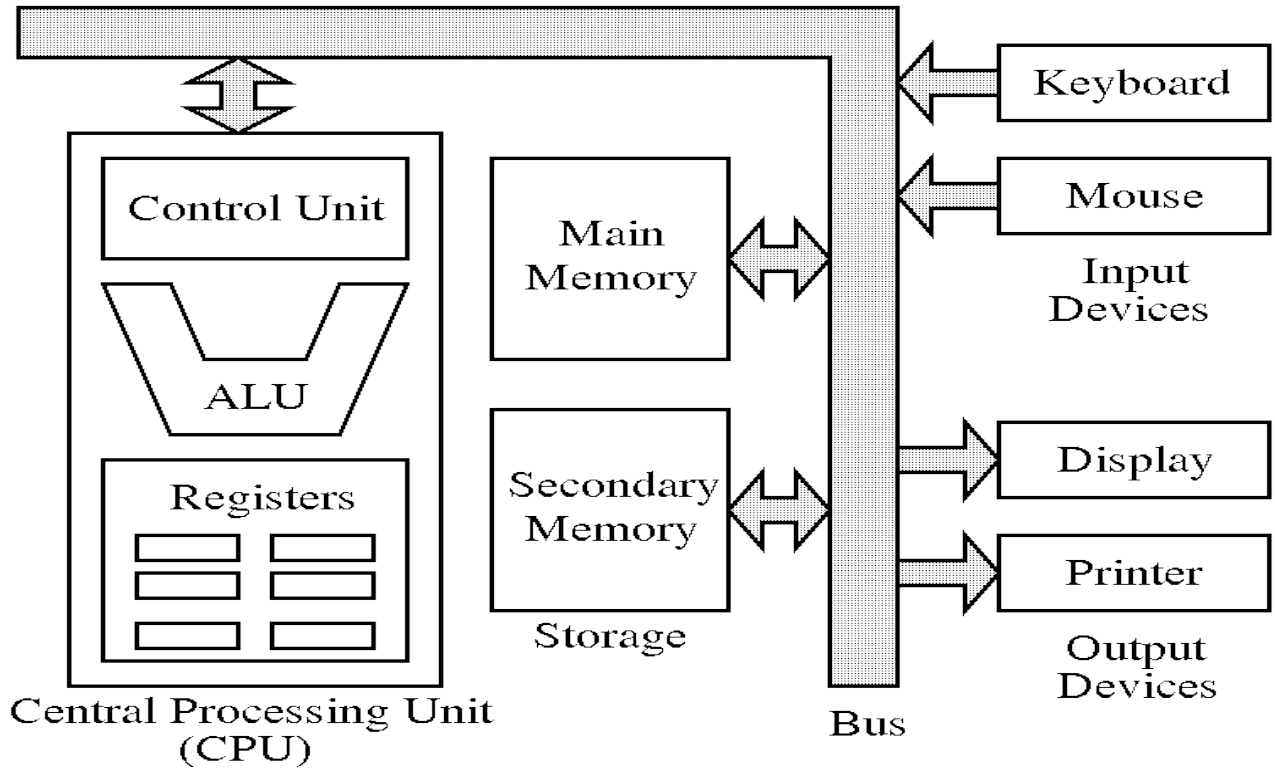
Computer:

A **computer** is a general purpose device that can be programmed to carry out a set of arithmetic or logical operations.

The two principal characteristics of a computer are: it responds to a specific set of instructions in a well-defined manner and it can execute a pre-recorded list of instructions (a program).

Computers can be generally classified by size and power as Personal Computer, Workstation, Minicomputer, Mainframe and supercomputer.

Von Neomon Architecture



The Von Neumann architecture is computer architecture which is designed by computer scientist **John von Neumann**.

This describes a design architecture for an computer with subdivisions of a

- Processing unit consisting of an arithmetic logic unit and processor registers
- Control unit containing an instruction register and program counter
- Memory to store both data and instructions
- External mass storage
- Input and Output mechanisms

Programming Languages

It is an artificial language designed to communicate with a computer. Programming languages can be used to create programs that control the behaviour of a computer.

The description of a programming language is usually split into the two components of syntax and semantics. Syntax means rules of writing the program and semantics means rules of meaning.

Types of Programming Language

Procedural Programming Languages

- Procedural programming specifies a list of operations that the program must complete to reach the desired state.
- Each program has a starting state, a list of operations to complete, and an ending point.
- This approach is also known as imperative programming. Integral to the idea of procedural programming is the concept of a procedure call.
- Procedures, also known as functions, subroutines, or methods, are small sections of code that perform a particular task.
- Ex. FORTRAN and BASIC.

Structured Programming Languages

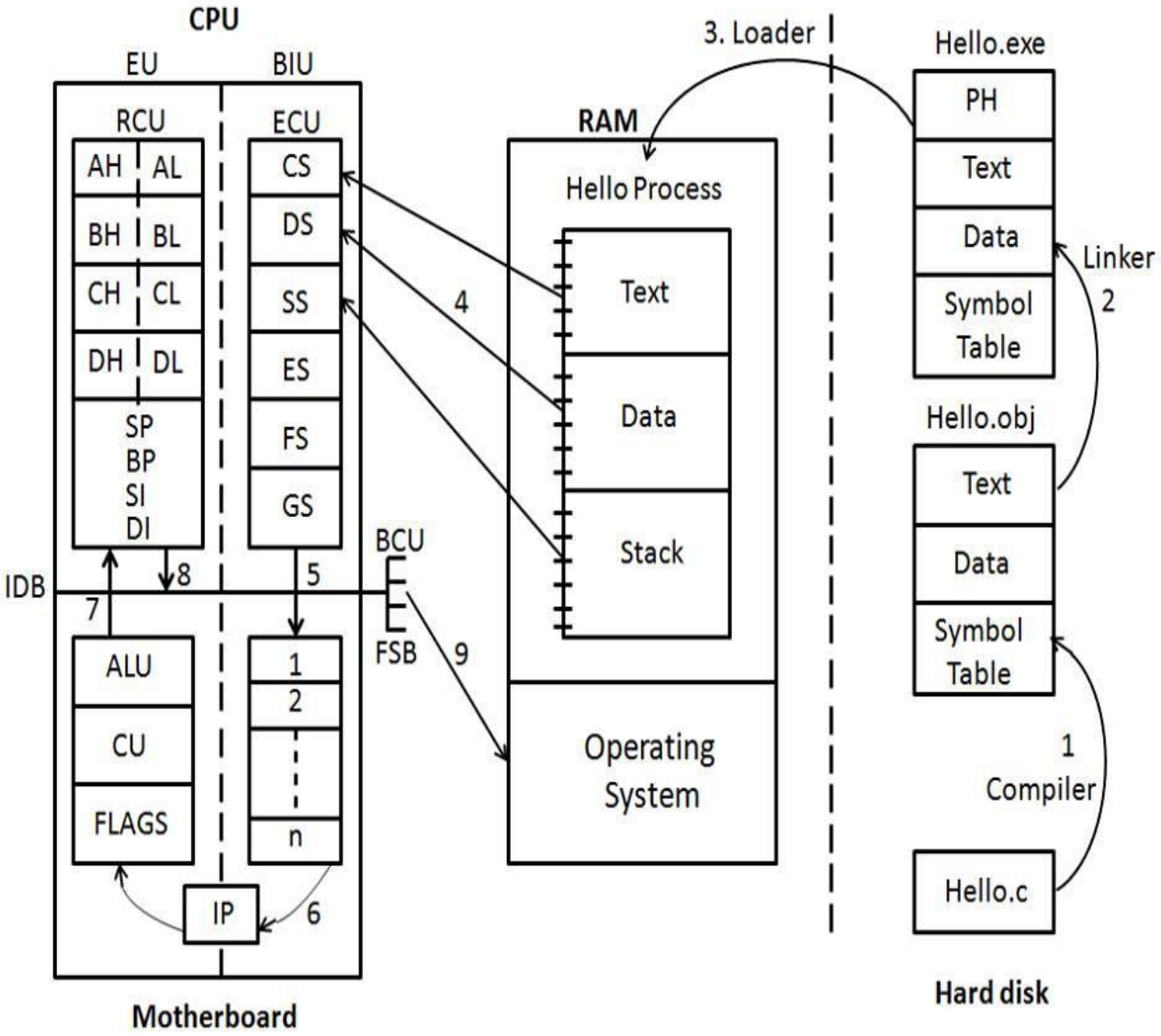
- Structured programming is a special type of procedural programming. It provides additional tools to manage the problems that larger programs were creating.
- Structured programming requires that programmers break program structure into small pieces of code that are easily understood.
- It also frowns upon the use of global variables and instead uses variables local to each subroutine.
- Ex. C, Ada, and Pascal.

Object-Oriented Programming Languages

- Object-oriented programming (OOP) is a programming paradigm that represents concepts as "objects" that have data fields (attributes that describe the object) and associated procedures known as methods.
- Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs.
- Ex. Java, Visual Basic, c#, C++, and Python.

X86 Toolchain

- A Toolchain is the set of Programming Tools that are used to createsoftware.
- The Tools are used in a chain, so that the output of each tool becomes input for the next tool.



Components of Toolchain

Editor:

- Editor is program in which we can write our code and edit that code.
- Gedit, kwrite are the examples of editors.

File Name: Demo.c

```
// Header file inclusion
# include<stdio.h>
// Macro defination
# define MAX 10

int Add(int No1, int No2)
{
        // Local variable
        intAns;

        Ans = No1 + No2 + MAX ;
        printf("Addition of Two Numbers is %d",Ans);
        returnAns;
}
```

Preprocessor:

- The Preprocessor provides the ability for the inclusion of header files,macro expansion, conditional compilation, and line control.
- Preprocessor takes lines beginning with '#' as directives. Because it knows nothing about the underlying language.
- After Preprocessing it will give the expanded code as a output having extension .i.
- Expanded output given by preprocessor is in human understandable format.

File Name: Demo.i

```
intprintf( const char * format, ...);
intscanf( const char * format, ...);

int Add(int No1, int No2)
{
        intAns;
        Ans = No1 + No2 + 10;
        printf("Addition of Two Numbers is %d",Ans);
        returnAns;
}
```

Compiler:

- Compiler is a program which is used for translating source code from a high level programming language (like c,c++) to lower level programming language (like assembly language, machine language).
- In this case we consider that output is in assembly language having extension .asm or .s.
- Code which is generated by the compiler is machine dependent.

File Name: Demo.asm

```
Add: PUSH ECX
PUSH EDX
ADD ECX,EDX
ADD ECX,10
MOV EAX,ECX
RETN
```

Assembler:

- Assembler is a program which translates assembly language program to an object file, which contains code in machine language.
- Object file contains machine code but still it is non-executable having extension .obj.

File name: Demo.obj

```
10110110111000101001010
10101110100010100101010
10111011101110101110110
00101110111011101110101
10101110100010100101010
10111011101110101110110
00101110111011101110101
10110110111000101001010
10101110100010100101010
10111011101110101110110
```

Linker:

- A Linker is a program that takes one or more object files and combines them into a single executable program having extension .exe.
- Linker adds primary header over executable file which contains address of entry point function, magic number, time-date stamp, type of executable (Executable can be self-executable or dependable executable).

File name: Demo.exe

Demo.obj		Other.obj
10110110111000101001010		10110110111000101001010
10101110100010100101010		10110110111000101001010
10111011101110101110110		10110110111000101001010
00101110111011101110101	+	00100010000100001001010
00100010000100001001010		00100010000100001001010
10111011101110110111101		00100010000100001001010
01000100001111101110101		10111011101110101110110
00100010000100001001010		00100010000100001001010
10111011101110110111101		00100010000100001001010
01000100001111101110101		10111011101110101110110

Demo.exe

10101110100010100101010
10111011101110101110110
00101110111011101110101
00100010000100001001010
10111011101110110111101
01000100001111101110101
10111011101110101110110
00101110111011101110101
00100010000100001001010
10111011101110110111101

Loader:

- A Loader is a program which is part of operating system that is responsible for loading programs.
- It places program in memory (RAM) and prepares them for execution.
- Loading a program involves reading the contents of executable file like text, data into the memory.

Demo.exe running in RAM

10101110100010100101010
10111011101110101110110
00101110111011101110101
00100010000100001001010
10111011101110110111101
01000100001111101110101
10111011101110101110110
00101110111011101110101
00100010000100001001010
10111011101110110111101

Stepwise Description of Toolchain

Step 1: First we write a program in any programming language like C / C++ on editor and then save that program. When we save the program with appropriate file extension like .c / .cpp it gets stored in hard disk in format of file (File is unformatted uniform stream of bytes).

In this step program is in human understandable format. And our task is to convert this format in to machine understandable format i.e. Machine language.

Step 2: To achieve this first we pass our program to the preprocessor. Preprocessor gives the expanded source code as a output. But still our program is in human understandable format.

Step 3: After preprocessing we pass the expanded program to the compiler which gives the output in assembly language which is hardware dependent language. But still our program is not in machine language.

Step 4: This assembly language program is pass to the assembler which converts assembly language program into machine language called as object code.

This is not pure executable code. This object file is divided into three parts as Text, Data, and Symbol table.

Text section is a section which contains compiled instructions of our program in machine understandable format.

Data section contains all the global variable and static variables of our program. Data section is internally divided into two parts as bss(Block starting with Symbol) and non bss(Block starting with value).

bss section contains all non-initialized global variables and non bss section contains all initialized global variables.

Symbol table is a data structure which is used to maintain all the information of identifiers used in our program.

Step 5: After getting the object code we pass this file to the linker. Linker is responsible for creating executable file.

Linker links our object code with other dependent object code. It also adds primary header to our executable file. Primary header is a structure which contains magic number, type of executable, address of entry point, time date stamp etc.

Magic number is a number which is used by operating system to uniquely identify the executable.

After getting output of linker our file is in executable format but it is stored in hard disk.

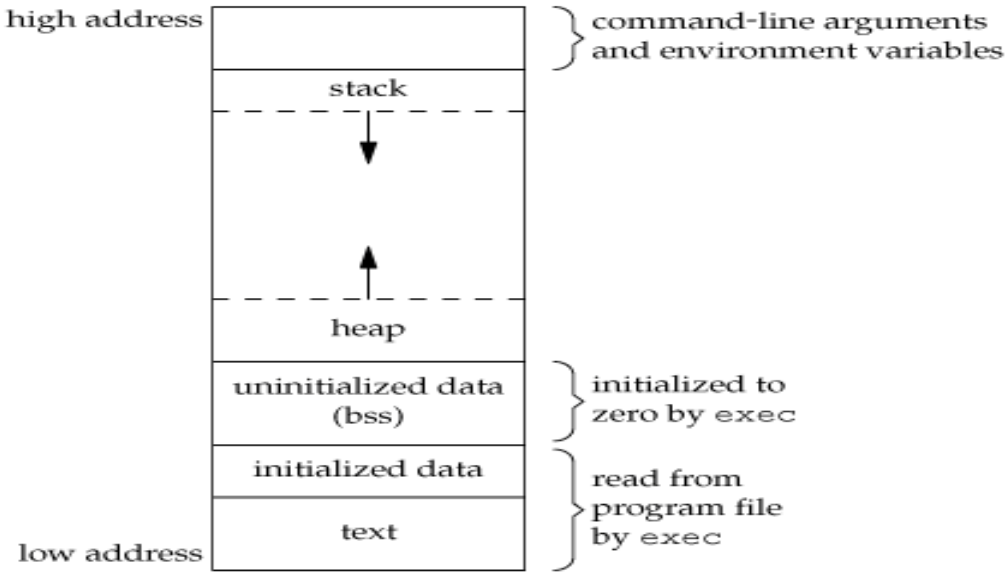
But if we want to run this program it must be copied into the main memory i.e. RAM.

Step 6: Loader is part of operating system which is responsible for loading the program from hard disk to RAM.

When our program gets loaded into the ram its primary header gets removed and instruction pointer is set to the address of entry point.

New section gets added which is called as stack which contains information of function. There is separate stack frame for each and every function which contains information like local variables, argument of function, old value of EBP, address of next instruction to be executed etc.

Memory layout of process



Step 7: Now our program gets copied into the memory. For further processing we have to pass that program to CPU. But it is not posssible to pass whole program to CPU because of memory constraints. Because of which each and every section of program get converted into small segment of small size.

Segment of text section gets copied into CS(Code Segment), segment of data segment gets copied into DS(Data Segment), and stack segment gets copied into SS(Stack Segment).

If there is more space required to store the segment then ES(Extra segment),DS,GS(There is no full forms) are used.

Step 8: Now contents of this segment gets converted into set of instructions. And these instructions are stored in queue called as instruction queue.

There is a IP (Instruction Pointer) who's responsible for fetching the instruction from instruction queue and pass for further processing.

Step 9: Now this instruction is accepted by CU (Control Unit).If the instruction is arithmetic instruction then it is handled by ALU (Arithmetic Logic Unit).

Step 10: To process the instructions temporary memory locations of processor are used called as CPU registers.

There are different types of registers as

General Purpose Registers (GPR)

- EAX (Accumulator Register)
- EBX (Base Register)
- ECX (Count Register)
- EDX (Data Register)

Index Registers

- ESI (Source Index Register)
- EDI (Destination Index Register)

Segment Registers

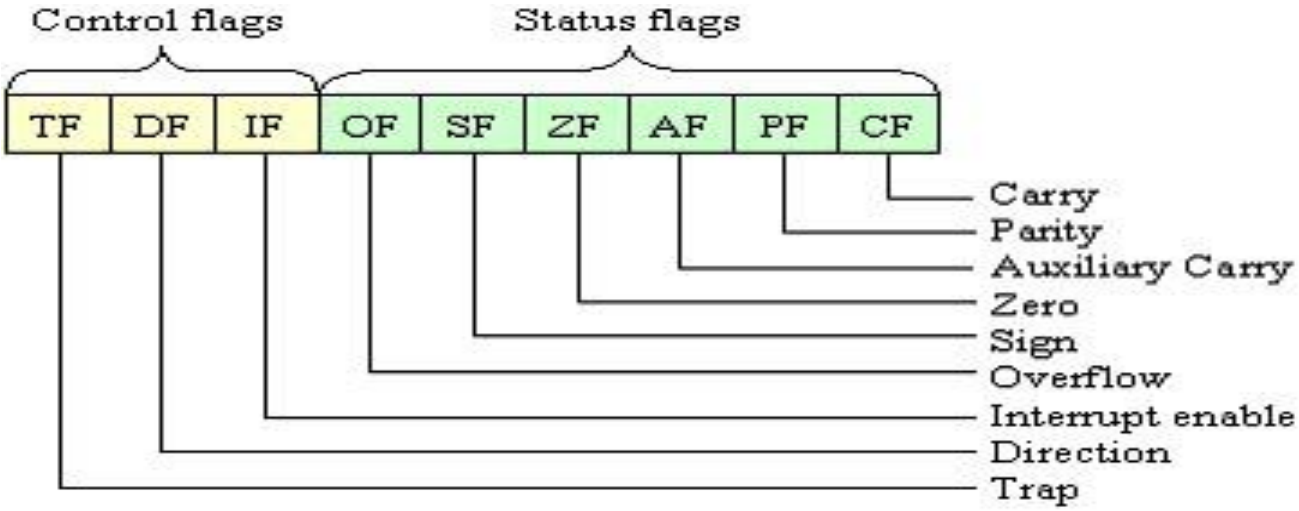
- CS (Code Segment Register)
- DS (Data Segment Register)
- SS (Stack Segment Register)
- ES (Extra Segment Register)
- FS & GS (There is no Full form)

Pointer Register

- ESP (Stack Pointer Register)
- EBP (Base Pointer Register)
- EIP (Instruction Pointer Register)

Flag Register

The FLAG register is the single status register of microprocessors that indicate the current state of the processor. Single bit of Flag register has its own meaning.



Step 11: Now the instructions gets evaluated and output of program is passed to the operating system. Then OS displays output of our program on to the screen.

Set of commands which will elaborate the Toolchain components

Step 1: Create file names as Demo.c which contains simple C program.

Step 2: Compile that file using command
gcc -S -o Demo.S Demo.c
This command gives output file in assembly language named as Demo.S

Step 3: Pass that Assembly file to assembler which gives output as aobjectfile named as Demo.o
as -o Demo.oDemo.S

Step 4: Pass that object file to Linker which gives output as a executable file.
ld -o Demo -lc -dynamic-linker /lib/ld-linux.so.2 Demo.o -e main

Step 5: Run that executable using
./Demo

Components of Computer

A computer system consists of mainly four basic components as

- Input Devices
- Output Devices
- Storage Devices
- Central Processing Unit

Microprocessor:

The microprocessor is the heart of any normal computer, whether it is a desktop machine, a server or a laptop.
A microprocessor also known as a CPU or central processing unit is a complete computation engine that is fabricated on a single chip.
The first microprocessor to make it into a home computer was the Intel 8080, a complete 8-bit computer on one chip, introduced in 1974.
The first microprocessor to make a real splash in the market was the Intel 8088, introduced in 1979 and incorporated into the IBM PC (which first appeared around 1982).
Since 2004, Intel has introduced microprocessors with multiple cores and millions more transistors. But even these microprocessors follow the same general rules as earlier chips.
A microprocessor executes a collection of machine instructions that tell the processor what to do. Based on the instructions, a microprocessor does three basic things:

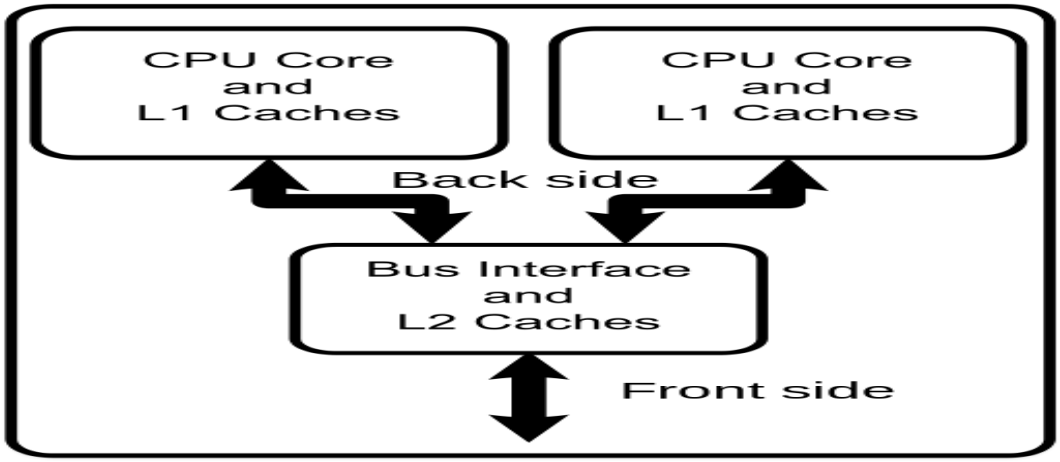
- Using its ALU (Arithmetic/Logic Unit), a microprocessor can perform mathematical operations like addition, subtraction, multiplication and division. Modern microprocessors contain complete floating point processors that can perform extremely sophisticated operations on large floating point numbers.
- A microprocessor can move data from one memory location to another.
- A microprocessor can make decisions and jump to a new set of instructions based on those decisions.

If we have an address bus 8 bits wide and a data bus 8 bits wide. That means that the microprocessor can address (2^8) 256 bytes of memory, and it can read or write 8 bits of the memory at a time.
When the microprocessor starts, it begins executing instructions it finds in the BIOS. The BIOS instructions do things like test the hardware in the machine, and then it goes to the hard disk to fetch the boot.
This boot sector is another small program, and the BIOS stores it in RAM after reading it off the disk. The microprocessor then begins executing the boot sector's instructions from RAM.
The boot sector program will tell the microprocessor to fetch something else from the hard disk into RAM, which the microprocessor then executes, and so on.
This is how the microprocessor loads and executes the entire operating system.

Each microprocessor has its own large set of instructions that it can perform. The collection of instructions is implemented as bit patterns, each one of which has a different meaning when loaded into the instruction register.
It is not possible to remember bit patterns, so a set of short words are defined to represent the different bit patterns. This collection of words (mnemonics) is called the assembly language of the processor.

Multicore Processor:
A multi-core processor is a single component with two or more independent actual central processing units (called "cores"), which are the units that read and execute program instructions.
The instructions are ordinary CPU instructions such as add, move data, and branch, but the multiple cores can run multiple instructions at the same time, increasing overall speed for programs.

Manufacturers typically integrate the cores onto a single integrated circuit die or onto multiple dies in a single chip package.



Processors were originally developed with only one core. Multi-core processors were developed in the early 2000s by Intel, AMD and others.

Multicore processors may have

- Two cores (Dual core): AMD Phenom II X2, Intel Core Duo
- Four cores (Quad core): AMD Phenom II X4, Intel's quad-core processors, i5, i7
- Six cores: AMD Phenom II X6, Intel Core i7 Extreme Edition 980X
- Eight cores: Intel Xeon E7-2820, AMD FX-8350
- Ten cores: Intel Xeon E7-2850

Homogeneous multi-core systems include only identical cores, heterogeneous multi-core systems have cores that are not identical.

Input Devices:

An input device is any peripheral used to provide data and control signals to a computer.
Ex: Keyboard, mouse and scanners.

Keyboard:

It is one of the major input devices. Inside a keyboard, there is a microcontroller that keeps monitoring to check if a key has been depressed or not. When key is depressed, some electrical connections are made, some amount of current get generated and thus the microprocessor comes to know about the key depressed.

Mouse:

A mouse is an input device that detects two-dimensional motion relative to a surface.
Apart from the keyboard mouse provides very easy way to communicate with the computer. A mouse is normally used to position the cursor at desired location on to the screen.
There are two types of mouse mechanical mouse and optical mouse. In case of mechanical mouse trackball is used to generate coordinates.
An optical mouse has no mechanical parts like ball. Instead it has built in photo detector. When we move the mouse photo detector sense the position and sends information to the computer in terms of change in x and y coordinates.

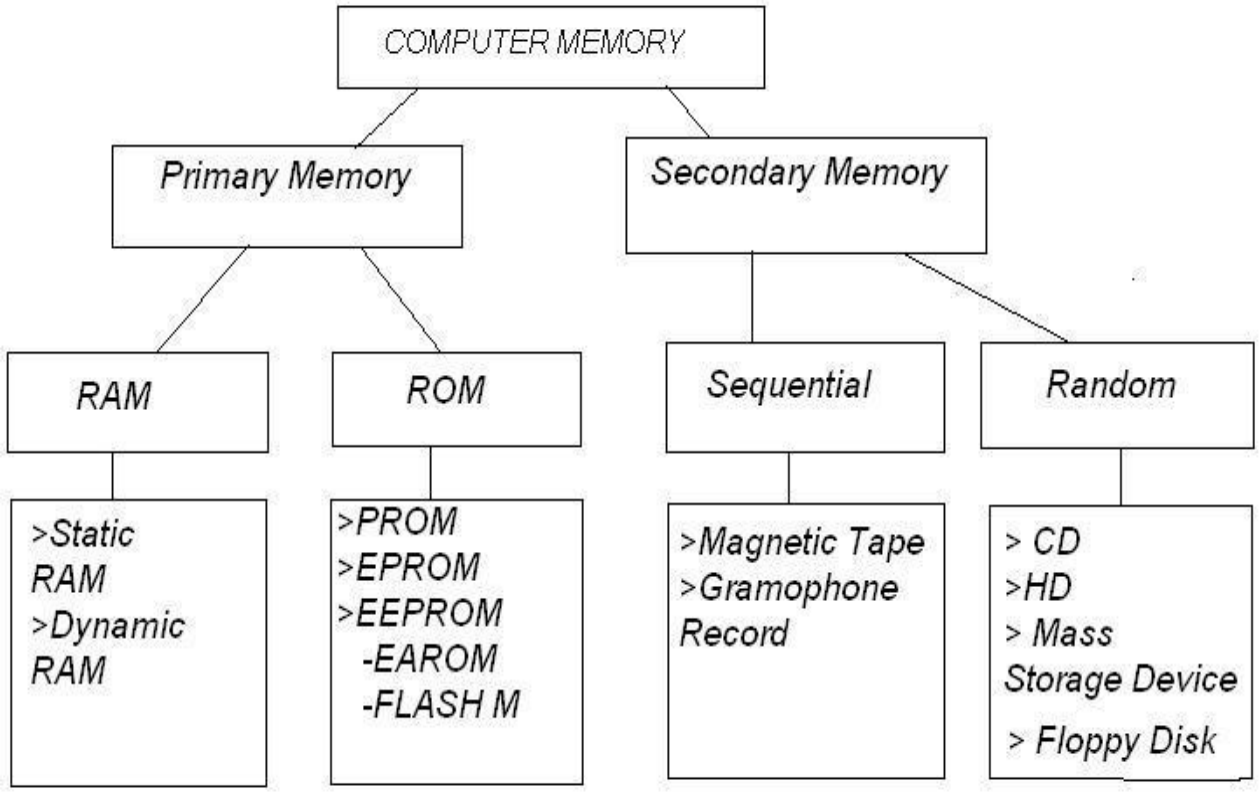
Output Devices:

An **output device** is any piece of computer hardware equipment used to communicate the results of data processing carried out by a computer which converts the electronically generated information into human-readable form.
Ex. Monitor, Printer, Projector.

Storage Devices:

It is a recording media used to retain data. There are two types of storage devices as

- Primary storage devices (RAM,ROM)
- Secondary storage devices (Hard disk,CD)



Random Access Memory (RAM):

- It is a form of data storage which allows stored data to be accessed directly in any random order.
- RAM is electronic storage medium.
- RAM is normally associated with volatile types of memory where its stored information is lost if the power is removed.
- In RAM each cell is connected in rows and column manner.

Types of RAM:

- **SRAM (Static RAM)**: In SRAM, a data is stored using the state of a flip-flop. This form of RAM is more expensive to produce, but is generally faster and requires less power than DRAM and, it is often used as cache memory for the CPU.
  - SRAM is more faster that the DRAM.

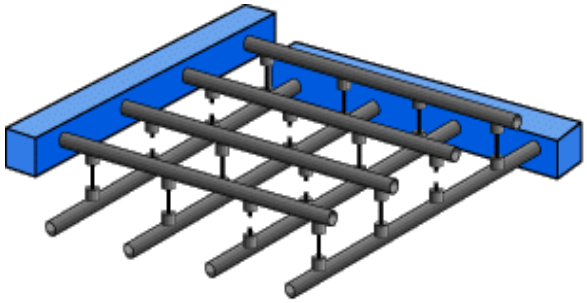
- **DRAM (Dynamic RAM)**: DRAM stores a data using a transistor and capacitor pair, which together comprises a memory cell.
  - The capacitor holds a high or low charge (1 or 0), and the transistor acts as a switch that lets the control circuitry on the chip read the capacitor's state of charge or change it.
  - This form of memory is less expensive to produce than SRAM.

Read Only Memory (ROM):

It is an integrated circuit programmed with specific data when it is manufactured.

Data stored in these chips is nonvolatile means it is not lost when power is removed.

Data stored in these chips is either unchangeable or requires a special operation to change.



ROM

ROM uses a diode to connect the lines if the value is 1. If the value is 0, then the lines are not connected at all.

Types of ROM:

- **PROM (Programmable ROM)**: Every intersection of a column and row in a PROM chip has a fuse connecting them. A charge sent through a column will pass through the fuse in a cell to a grounded row indicating a value of 1.
  - Since all the cells have a fuse, the initial (blank) state of a PROM chip is all 1s. To change the value of a cell to 0 send a specific amount of current to the cell.
  - The higher voltage breaks the connection between the column and row by burning out the fuse. This process is known as burning the PROM.

- **EPROM (Erasable Programmable ROM)**: This type of ROM is similar as PROM, but the only difference is that we can erase the contents from EPROM.

- **EEPROM (Electrically Erasable Programmable ROM):** This is an EPROM in which we can edit the contents electrically.

- **UVEPROM (Ultra Violate Erasable Programmable ROM):** This is an EPROM in which we can edit the contents using ultra violate rays.

Flash memory:

- Flash memory is an electronic non-volatile computer storage medium that can be electrically erased and reprogrammed.
- Flash memory was developed from EEPROM.

- There are two main types of flash memory, which are named after the NAND and NOR logic gates like NAND flash memory and NOR flash memory.
- All pen drives store data in drives with very minute electricity by means of signal 0 and 1, where 0 is absence of electricity and 1 is presence of electricity.
- Pen drives works on principle of nonvolatile memory ie Flash memory which is nothing but EEPROM.
- USB flash drives are basically external hard drives that don't require an external power source to operate because they get their power from the computer they are plugged in.

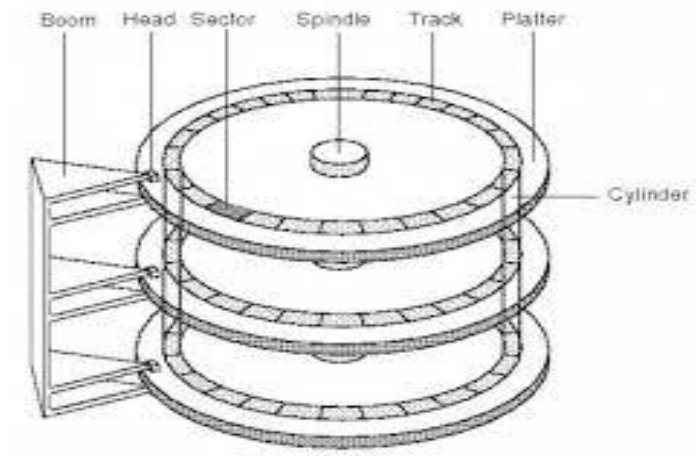
Cache Memory:

- There is a speed mismatch of CPU and main memory.
- To avoid the problem of speed mismatch cache memory is used.
- The cache is a smaller, faster memory which stores copies of the data from frequently used main memory locations.
- Most CPUs have different independent caches, including instruction and data caches, where the data cache is usually organized as a hierarchy of more cache levels (L1, L2, L3 etc.)

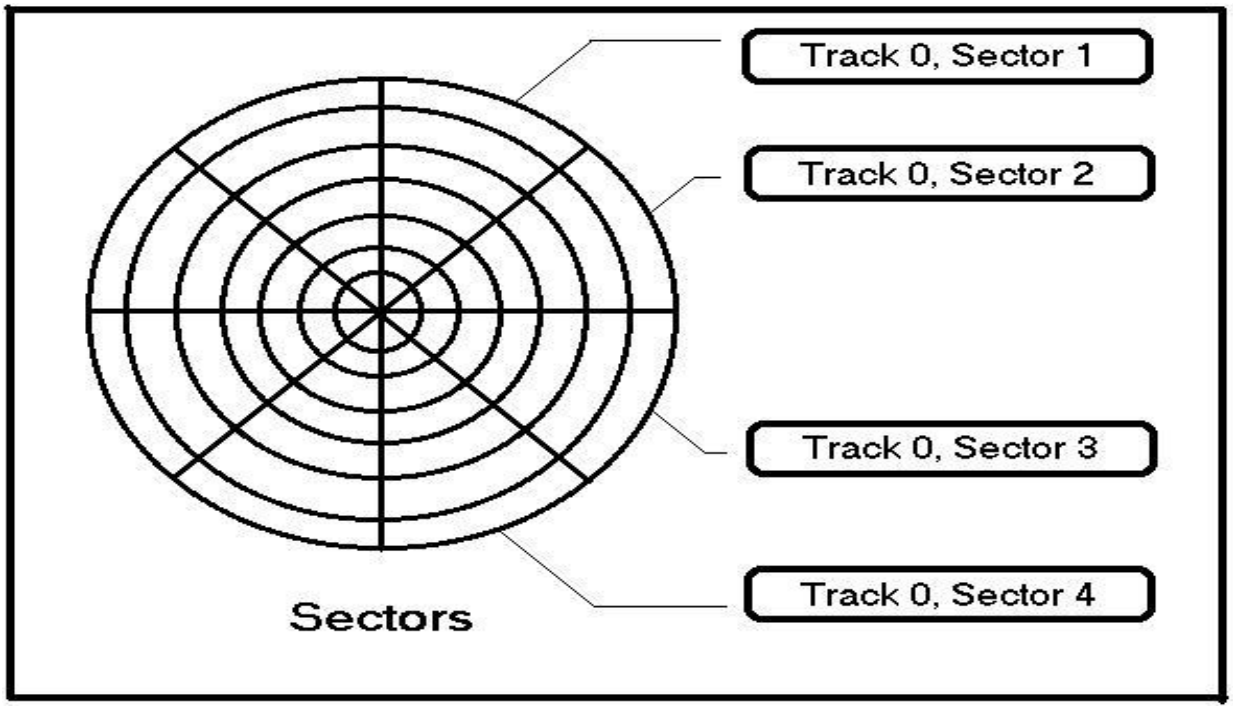
Direct Memory Access (DMA):

- Direct memory access (DMA) is a method that allows an input/output (I/O) device to send or receive data directly to or from the main memory, bypassing the CPU to speed up memory operations.
- The process is managed by a chip known as a DMA controller (DMAC).
- Without DMA, when the CPU is using programmed input/output, it is typically fully occupied for the entire duration of the read or write operation, and is thus unavailable to perform other work.
- With DMA, the CPU initiates the transfer, does other operations while the transfer is in progress, and receives an interrupt from the DMA controller when the operation is done.

Hard Disk Drive (HDD):



- It is a storage device used for storing and retrieving information using rapidly rotating disks (platters) coated with magnetic material.
- An HDD retains its data even when powered off. An HDD consists of one or more rigid ("hard") rapidly rotating disks (platters) with magnetic heads arranged on a moving arm to read and write data to the surfaces.
- Hard disk has many platters, each of which has two surfaces. The platters rotate around the mounting shaft and there is a Read/Write head one per surface.

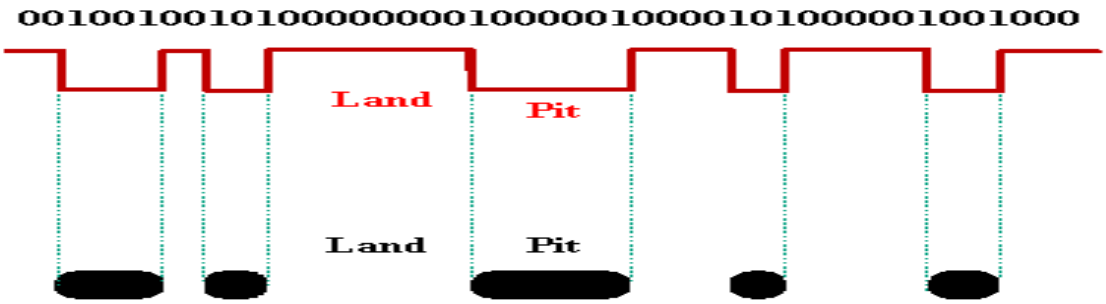


- In case of hard disk concept called cylinder is used. A cylinder essentially consist of tracks on all surfaces which have same diameter.

- Each surface is divided into number of tracks and each track is divided into a sector of equal size. Size of sector can vary but it is generally 512 bytes. A given sector is specified by three components of an address made up of cylinder(Track) number, surfaces number and sector number.

Optical Disk:

- An optical disk is a plastic disk coated with highly reflective material like polycarbonate. It uses laser beams for writing and reading data.
- Like harddisk optical disk also stores the data on tracks. But tracks of optical disk are spiral and tracks of harddisk are made up of consentric circales.
- It uses a laser of greater intensity to write data by itching microscopic pits on disk surface. Based on os and 1s to be written on the disk surface, the intensity of the leaser beam is intensified.



- When high intensity laser beam falls on surface of CD(Compact Disk) it burns the pit into surface. Pits are not formed in the portion where high intensity is not applied and the coating at that place is intact and it is called as land.
- Means the contents which are written on to the optical in term of pits and land.

PCI Cards

Graphics card(Video card):

- A graphics card is an expansion card which generates a feed of output images to a computer monitor. Most video cards offer various functions such as accelerated rendering of 3D scenes and 2D graphics, MPEG-2/MPEG-4 decoding, TV output, or the ability to connect multiple monitors (multi-monitor).
- As an alternative to the use of an external graphics card, video hardware can be integrated into the motherboard.

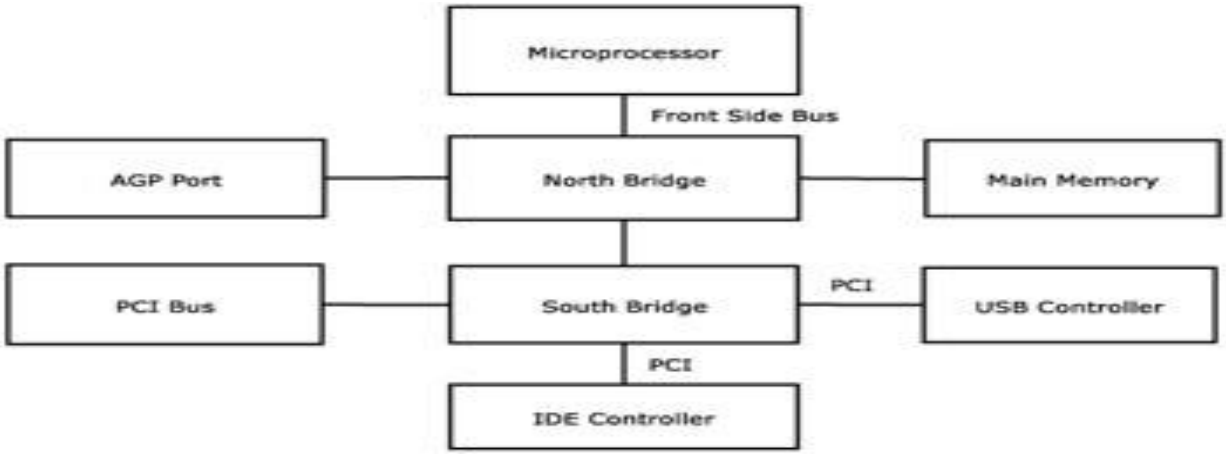
- Motherboard-based implementations are sometimes called "on-board video". Sometimes both the integrated graphics and a dedicated graphics card can be used simultaneously to feed separate displays.
- The main advantages of integrated graphics include cost, compactness, and simplicity and low energy consumption. The performance disadvantage of integrated graphics arises because the graphics processor shares system resources with the CPU.
- A dedicated graphics card has its own random access memory (RAM), its own cooling system, and dedicated power regulators, with all components designed specifically for processing video images.
- Upgrading to a dedicated graphics card offloads work from the CPU and system RAM, so not only will graphics processing be faster, but the computer's overall performance may also improve.

Sound Card:

- It is a hardware device that allows a computer to playback or record sound.
- It is of two types internal sound card and external sound card.
- An internal sound card is integrated into the motherboard of a computer while the external sound card is plugged into one of the expantion slot of the motherboard.
- The main task of a sound card is to convert digital sound data into analog sound.

Motherboard:

A motherboard is the main printed circuit board (PCB) used in computers.
It holds many of the crucial electronic components of the system, such as the central processing unit (CPU) and memory, and provides connectors for other peripherals.



This board is the "mother" of all components attached to it, which often include sound cards, video cards, network cards, hard drives, TV tuner cards, BIOS chip etc.

North Bridge:
It is in charge of controlling transfers between the processor and the RAM, which is way it is located physically near the processor.

South Bridge:
It handles communications between peripheral devices. It is also called the ICH (*I/O Controller Hub*).
The term bridge is generally used to designate a component which connects two buses.

Bus:
A collection of wires through which data is transmitted from one part of a computer to another are known as Bus.
Bus connects all the internal computer components to the CPU and main memory.

Types of bus:

- **Address bus**:
  Transports memory addresses which the processor wants to access in order to read or write data.
  When a processor or DMA needs to read or write to a memory location, it specifies that memory location on the address bus (the value to be read or written is sent on the data bus).
  The width of the address bus determines the amount of memory a system can address. For example, a system with a 32-bit address bus can address 2^{32} (4,294,967,296) memory locations.
  If each memory address holds one byte, the addressable memory space is 4 GB.

- **Data bus:**
  As name tells that it is used to transfer data within Microprocessor and Memory/Input or Output devices.
  It is bidirectional as Microprocessor requires sending or receiving data.
  Data bus is used to send the value to be written to/read off the memory.
  If I have a data bus of size 32 bits, it means a maximum of 4 bytes can be written to/read off the memory at a time.
  The word length of a processor depends on data bus, that's why Intel 8085 is called 8 bit Microprocessor because it has an 8 bit data bus.

- **Control bus**:
  Transports orders and synchronization signals coming from the control unit and travelling to all other hardware components.

It is a bidirectional bus, as it also transmits response signals from the hardware.
It is a computer bus, used by CPUs for communicating with other devices within the computer.
The control bus carries commands from the CPU and returns status signals from the devices.

BIOS (Basic Input Output System):
Basic Input/output System, theBIOS,ROM BIOS,orSystem BIOSis a chip located on all motherboards that contain instructions and setup for how your system should boot and how it operates.
It works as an interface between the operating system and the motherboard.
The BIOS software has a number of different roles, but its most important role is to load the operating system.
When we turn on our computer and the microprocessor tries to execute its first instruction, it has to get that instruction from somewhere.
It cannot get it from the operating system because the operating system is located on a hard disk, and the microprocessor cannot get to it without some instructions that tell it how. The BIOS provides those instructions**.**

Some of the other common tasks that the BIOS perform include:

- A power-on self-test (POST) for all of the different hardware components in the system to make sure everything is working properly
- Activating other BIOS chips on different cards installed in the computer - For example, SCSI and graphics cards often have their own BIOS chips.
- Providing a set of low-level routines that the operating system uses to interface to different hardware devices - It is these routines that give the BIOS its name. They manage things like the keyboard, the screen, and the serial and parallel ports, especially when the computer is booting.
- Managing a collection of settings for the hard disks, clock, etc.

When we turn on our computer, the BIOS do several things. This is its usual sequence:

1. Check the CMOS Setup for custom settings
2. Load the interrupt handlers and device drivers
3. Initialize registers and power management
4. Perform the power-on self-test (POST)
5. Display system settings
6. Determine which devices are bootable
7. Initiate the bootstrap sequence

Operating Systems Basics

Operating System:
It is a set of programs that act as a interface between a user of a computer and the computer hardware.

Operating system performs following tasks:

- **File management:**
  - It used to maintain information about the files and allows user to create file, read contents of file, write contents in file etc.

- **Process management:**
  - A program does nothing unless its instructions are executed by CPU. A process is defined as program in execution or running instance of program.
  - Process management part of operating system is responsible for creating and terminating process, provides mechanism for process synchronization, suspending and resuming the process, maintaining context of a process etc.

- **Memory management:**
  - This part of OS is responsible for allowing user to allocate memory, deallocate memory and maintaining the record of allocated and unallocated memory.

- **Hardware abstraction:**
  - It means hiding the internal complexity of hardware from user. This is most important task of operating system.

- **CPU scheduling:**
  - Schedule a particular task from set of multiple task and assign this task to CPU is the responsibility of CPU scheduler.

Types of Operating Systems

- **Batched Operating system:**
  - Batch operating system is the operating system which analyzes input and groups them into batches.
  - That is data in each batch is of similar characteristics. And then it performs operation on each individual batch.

- **Multiprogrammed Operating system:**
  - In this type of operating system there are multiple jobs which are loaded in memory.
  - OS picks and execute one of the job from the memory. In non-multiprogrammed system the CPU would sit idle.

- **Time sharing operating system:**
  - Time sharing operating system is a operating system in which available CPU time is divided into equal slots.
  - Then these slots are assigned to all the tasks. The selected task gets executed in specified time slot.

- **Desktop operating system:**
  - This is not a multiuser operating system. Goal of this type of operating system is different, instead of maximizing CPU and peripheral utilization these operating systems used for maximizing user convenience and responsiveness.

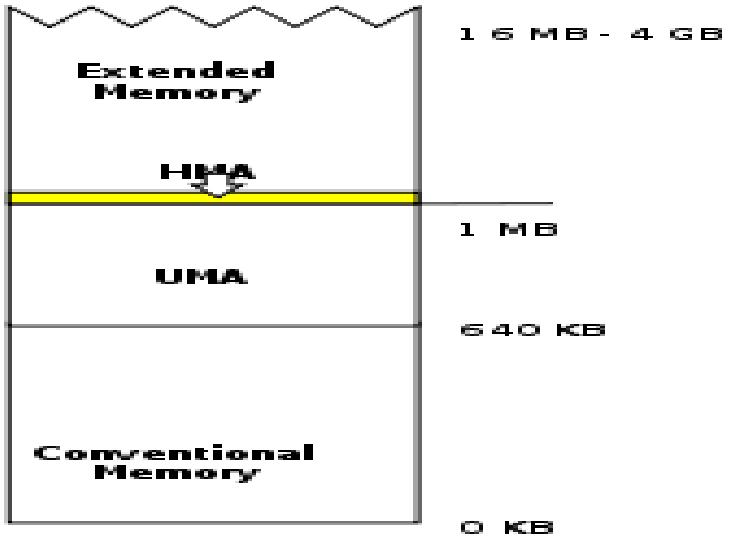
- **Distributed Operating system:**
  - It is an OS where distributed applications are running on multiple computers linked by communications.
  - A distributed operating system is an extension of the network operating system that supports higher levels of communication and integration of the machines on the network.

- **Real Time Operating system:**
  - It is a special form of OS. A real-time operating system (RTOS) is an operating system (OS) intended to serve real-time application requests.
  - It must be able to process data as it comes in, typically without buffering delays. Processing time requirements (including any OS delay) are measured in tenths of seconds or shorter.

Real mode and Protected mode

Real mode:

- Real mode is an operating mode of all x86-compatible CPUs.
- Real mode is characterized by a 20-bit segmented memory address space.
- In this mode only 20 address lines are accessible, due to which operating system access only 2^{20} i.e. 1MiB memory is accessible.
- Real mode provides no support for memory protection, multitasking, or code privilege levels.
- Before the release of the 80286, which introduced protected mode, real mode was the only available mode for x86 CPUs.
- In the interests of backwards compatibility, all x86 CPUs start in real mode when reset.



- In real mode operating system is single tasking means it can perform only single task at a time.
- Every operating system first boot in real mode then enters into the protected mode.
- To enter into the protected mode it require to open A20(Address line 20) gate.
- When the A20 gate opens all the address lines are accessible. If the address bus is of 32 bit then it has range from A0 to A31.
- If operating system is real mode operating system then it uses only first 640 kb part of RAM.
- If a process wants more memory than the 640kb then OS takes some part of memory after 640kb in temporary purpose then this type of memory management is called as expanded memory scheme.

Protected mode:

- Protected mode is an operational mode of x86-compatible CPU.
- It allows system software to use features such as virtual memory, paging and safe multi-tasking designed to increase an operating system's control over application software.
- When a processor that supports x86 protected mode is powered on, it begins executing instructions in real mode, in order to maintain backwards compatibility with earlier x86 processors.
- Protected mode may only be entered after the system software sets up several descriptor tables and enables the Protection Enable (PE) bit in the control register 0 (CR0).
- Protected mode was first added to the x86 architecture in 1982, with the release of Intel's 80286 (286) processor, and later extended with the release of the 80386 (386) in 1985.

Concepts of Multi Paradigms

- **Multitasking:**
  - Performing multiple task simultaneously is called as multitasking. Multiprocessing and multithreading are two types of multitasking.

- **Multiprocessing:**
  - When there are more than one process concurrently running then such type of situation is called as multiprocessing.

- **Multithreading:**
  - When there are multiple parts of a single process performing multiple task simultaneously then it is called as multithreading.

- **Multiprogramming:**
  - If multiprocessing and multithreading is performed with the help of multiple CPU then it is called as multiprogramming.

- **Multicore programming:**
  - If multiprocessing and multithreading is performed with the help of multiple parts (cores) of CPU then it is called as multicore programming.

- **Multiuser:**
  - If single copy of operating system is shared between multiple users then it is called as multiuser operating system.

Kernel

Operating System is divided into two parts*kernel (core part + system services) + user interface (command line or graphical)***.**

The *kernel* is a piece of software that constitutes the central core of a computer operating system. It has complete control over everything that occurs in the system.

- **Kernel Mode:**
  - In Kernel mode, the executing code has complete and unrestricted access to the underlying hardware.
  - It can execute any CPU instruction and reference any memory address.
  - Kernel mode is generally reserved for the lowest-level, most trusted functions of the operating system.

- **User Mode:**
  - In User mode, the executing code has no ability to *directly* access hardware or reference memory.
  - Code running in user mode must delegate to system APIs to access hardware or memory.
  - Due to the protection afforded by this sort of isolation, crashes in user mode are always recoverable.

Kernel and its types

Monolithic Kernel:

- Monolithic kernel got simple design and it is a single large processes running entirely in a single address space and in the kernel space.
- It is a single static binary file.
- All kernel services exist and execute in kernel address space. The kernel can invoke functions directly.
- but one consequence of all parts of the kernel running in the same address space is that if there is an error somewhere in the kernel, it will have an effect on the entire address space; in other words, a bug in the subsystem that takes care of networking might crash the kernel as a whole, resulting in the user needing to reboot his system.

Example: MS-DOS, BSD, Windows 98, Linux, etc.

Micro Kernel:

- Micro kernels designed to fix the above problem, it try to limit the amount of damage a bug can cause.
- They do this by moving parts of the kernel away from the dangerous kernel space into user space, where the parts run in isolated processes (so-called 'servers') as a consequence, they do not influence each other's functioning.
- The bug in the networking subsystem which crashed monolithic kernel will have far less severe results in a microkernel design: the subsystem in question will crash, but all other subsystems will continue to function.
- In fact, many microkernel operating systems have a system in place which will automatically reload crashed servers.
- In a microkernel design, only a small subset of the tasks a monolithic kernel performs reside in kernel space, while all other tasks live in user space.
- Generally, the part residing in kernel space (the actual 'microkernel') takes care of the communication between the servers running in user space; this is called 'inter-process communication (IPC).
- These servers provide functionality such as sound, display, disk access, networking, and so on.

Example:Mach, Minix

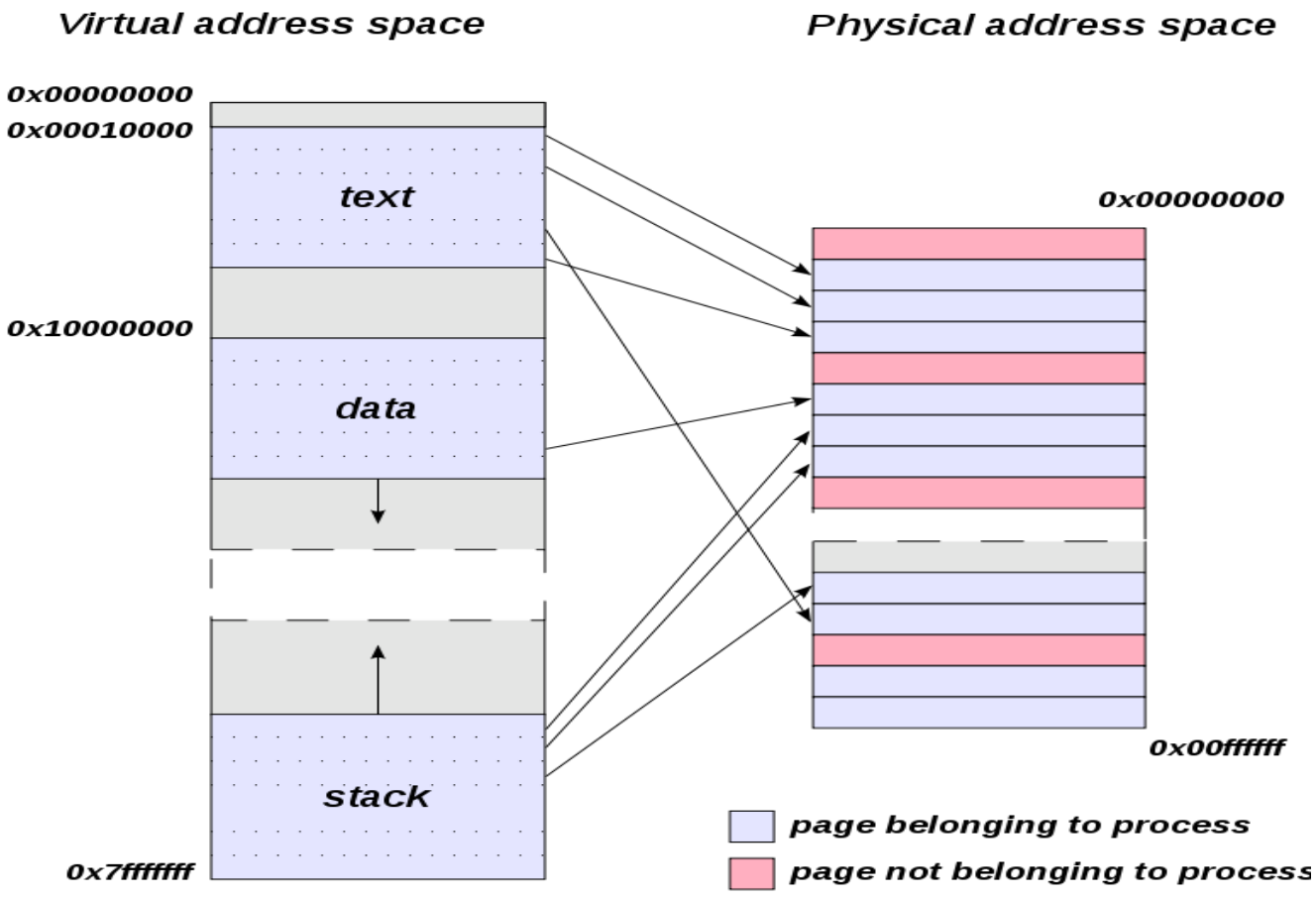
Hybrid Kernel:

- This design combines the monolithic and microkernel design in that it has characteristics of both.
- It keeps some subsystems in kernel space to increase performance, while keeping others out of kernel space to improve stability.

- That part of a hybrid kernel running in kernel space is in fact structured as if it were a microkernel; as a consequence, parts which run in kernel space can actually be 'moved out' of it to user space relatively easily.
- Hybrid kernels are used in most commercial operating systems such as Microsoft Windows NT, 2000, XP, Vista, and 7**.** Apple Mac OS X uses a hybrid kernel called XNU which is based upon code from Carnegie Mellon's Mach kernel and FreeBSD's monolithic kernel.

Address space of a process

- ✓ Address space is the amount of memory allocated for all possible addresses for a computational entity, such as a process.
- ✓ Address space may refer to a range of either physical or virtual addresses accessible to a processor or reserved for a process.
- ✓ On a computer, each computer device and process is allocated address space, which is some portion of the processor's address space.
- ✓ There are two types of address space as Virtual address space and Physical address space.



Virtual Address space:

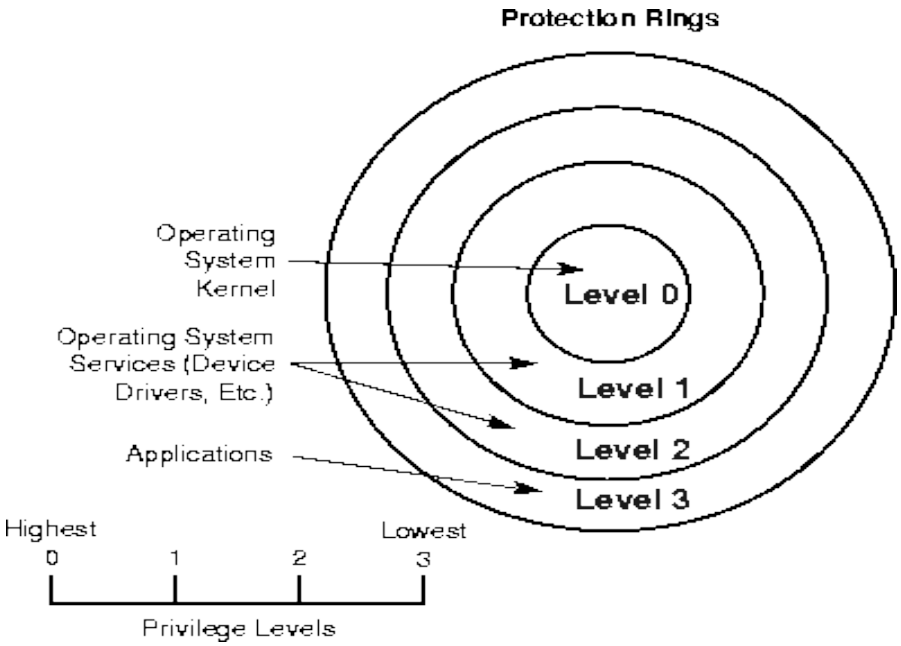
- The virtual address space for a process is the set of virtual memory addresses that it can use.
- The address space for each process is private and cannot be accessed by other processes unless it is shared.
- A virtual address does not represent the actual physical location of an object in memory instead, the system maintains a *page table* for each process, which is an internal data structure used to translate virtual addresses into their corresponding physical addresses.

Physical address space:

- The physical address space for a process is the set of physical memory addresses that it can use.
- A physical address represent the actual physical location of an object in memory

Operating system ring model

- Ring model isa mechanism to protect data and functionality from faults and malicious behavior.
- Computer operating systems provide different levels of access to resources. A protection ring is one of two or more hierarchical *levels* or *layers* of privilege within the architecture of a computer system.



- Rings are arranged in a hierarchy from most privileged (most trusted, usually numbered zero) to least privileged (least trusted, usually with the highest ring number).
- Special gates between rings are provided to allow an outer ring to access an inner ring's resources in a predefined manner, as opposed to allowing arbitrary usage.

Booting process

- Booting process is the initial set of operations that a computer performs after electrical power to the CPU is switched on or when the computer is reset.
- This can take tens of seconds and typically involves performing power-on self-test, locating and initializing peripheral devices, and then finding, loading and starting an operating system.
- A boot loader is a computer program that loads the main operating system for the computer after completion of the self-tests.
- The computer term *boot* is short for *bootstrap*or *bootstrap load* and derives from the phrase *to pull oneself up by one's bootstraps*.
- The usage calls attention to the requirement that, if most software is loaded onto a computer by other software already running on the computer, some mechanism must exist to load the initial software onto the computer.

Booting Process - Windows XP.

Step 1: Power supply switched on.

- The power supply performs a selftest.
- When all voltages and currentlevels are acceptable, the supply indicates that the power is stable and sends the Power Good signal to the processor.
- The time from switch-on to Power Good is usually between .1 and .5 seconds.

Step 2: The microprocessor timer chip receives the Power Good signal.

- With the arrival of the Power Good signal the timer chip stops sending reset signals to the processor allowing the CPU to begin operations.

Step 3: The CPU starts executing the ROM BIOS code.

- The CPU loads the ROM BIOS starting at ROM memory address FFFF:0000 which is only 16 bytes from the top of ROM memory.
- As such it contains only a JMP (jump) instruction that points to the actual address of the ROM BIOS code.

Step 4: The ROM BIOS performs a basic test of central hardware to verify basic functionality.

- Any errors that occur at this point in the boot process will be reported by means of 'beep-codes' because the video subsystem has not yet been initialized.

Step5: The BIOS searches for adapters that may need to load their own ROM BIOS routines.

- Video adapters provide the most common source of adapter ROM BIOS.
- The start-up BIOS routines scan memory addresses C000:0000 through C780:0000 to find video ROM.

- An error loading any adapter ROM generates an error such as: XXXX ROM Error where XXXX represents the segment address of the failed module.

Step 6: The ROM BIOS checks to see if this is a 'cold-start' or a 'warm-start'

- To determine whether this is a warmstart or a cold start the ROM BIOS startup routines check the value of two bytes located at memory location 0000:0472.
- Any value other than 1234h indicates that this is a coldstart.

Step 7: If this is a cold-start the ROM BIOS executes a full POST. If this is a warm-start the memory test portion of the POST is switched off.

- The POST can be broken down into three components: The Video Test initializes the video adapter, tests the video card and video memory, and displays configuration information or any errors.
- The BIOS Identification displays the BIOS version, manufacturer, and date. The Memory Test tests the memory chips and displays a running sum of installed memory.
- Errors occur during the POST can be classified as either 'fatal' or 'nonfatal'.
- A non-fatal error will typically display an error message on screen and allow the system to continue the boot process.
- A fatal error, on the other hand, stops the process of booting the computer and is generally signaled by a series of beep-codes.

Step 8: The BIOS locates and reads the configuration information stored in CMOS.

- CMOS (which stands for Complementary Metal-Oxide Semiconductor) is a small area of memory (64 bytes) which is maintained by the current of a small battery attached to the motherboard.
- Most importantly for the ROM BIOS startup routines CMOS indicates the order in which drives should be examined for an operating systems - floppy first, CD-ROM first, or fixed disk first.

Step 9: If the first bootable disk is a fixed disk the BIOS examines the very first sector of the disk for a Master Boot Record (MBR). For a floppy the BIOS looks for a Boot Record in the very first sector.

- On a fixed disk the Master Boot Record occupies the very first sector at cylinder 0, head 0, sector 1.
- It is 512 bytes in size. If this sector is found it is loaded into memory at address 0000:7C00 and tested for a valid signature.
- A valid signature would be the value 55AAh in the last two bytes.
- Lacking an MBR or a valid signature the boot process halts with an error message which might read: NO ROM BASIC - SYSTEM HALTED.
- A Master Boot Record is made up of two parts - the partition table which describes the layout of the fixed disk and the partition loader code which includes instructions for continuing the boot process.

Step 10: With a valid MBR loaded into memory the BIOS transfers control of the boot process to the partition loader code that takes up most of the 512 bytes of the MBR.

- The process of installing multiple operating systems on a single PC usually involves replacing the original partition loader code with a Boot Loader program that allows the user to select the specific fixed disk to load in the next step of the process

Step 11: The partition loader (or Boot Loader) examines the partition table for a partition marked as active. The partition loader then searches the very first sector of that partition for a Boot Record.

- The Boot Record is also 512 bytes and contains a table that describes the characteristics of the partition (number of bytes per sectors, number of sectors per cluster, etc.) and also the jump code that locates the first of the operating system files (IO.SYS in DOS).

Step 12: Boot Record- The active partition's boot record ischecked for a valid boot signatureand if found the boot sector code isexecuted as a program.

- The loading of Windows XP is controlled by the file NTLDR which is a hidden, system file that resides in the root directory of the system partition. NTLDR will load XP in four stages:
  - Initial Boot Loader Phase
  - Operating System selection
  - Hardware Detection
  - Configuration Selection

Step 13: During the initial phase NTLDR switches the processor from realmode to protected mode which places the processor in 32-bit memory mode and turns memory paging on.

- It then loads the appropriate mini-file system drivers to allow NTLDR to load files from a partition formatted with any of the files systems supported by XP.
- Windows XP supports partitions formatted with either the FAT-16, FAT-32, or NTFS file system.

Step 14: NTLDR OS Selection

- If the file BOOT.INI is located in the root directory NTLDR will read it's contents into memory.
- If the file BOOT.INI is not found in the root directory NTLDR will continue the boot sequence and attempt to load XP BOOT.INI contains entries for more than one operating system NTLDR will stop the boot sequence at this point, display a menu of choices, and wait for a specified period of time for the user to make a selection from the first partition of the first disk, typically C:\.

Step 15: NTLDR Hardware Detection

- If the selected operating system is XP, NTLDR will continue the boot process by locating and loading the DOS based NTDETECT.COM program to perform hardwaredetection.
- NTDETECT.COM collects a list of currently installed hardware components and returns this list for later inclusion in the registry under the HKEY_LOCAL_MACHINE\ HARDWARE key.

Step 16: Kernel Load

- After selecting a hardware configuration (if necessary) NTLDR begins loading the XP kernel(NTOSKRNL.EXE).
- During the loading of the kernel (but before it is initialized) NTLDR remains in control of the computer.
- The screen is cleared and a series of white rectangles progress across the bottom of the screen.
- NTLDR also loads the Hardware Abstraction Layer (HAL.DLL) at this time which will insulate the kernel from hardware.
- Both files are located in the \system32 directory.

Step 17: NTLDR Boot Device Drivers

- NTLDR now loads device drivers that are marked as boot devices. With the loading of these drivers NTLDR relinquishes control of the computer.
- Every driver has a registry subkey entry under HKEY_LOCAL_MACHINE \SYSTEM\Services.
- Any driver that has a Start value of SERVICE_BOOT_START is considered a device to start at boot up.
- A period is printed to the screen for each loaded file (unless the /SOS switch is used in which case file names are printed.

Step 18: Kernel Initialization

- NTOSKRNL goes through two phases in its boot process – phase 0 and phase 1. Phase 0 initializes just enough of the microkernel and Executive subsystems so that basic services required for the completion of initialization become available.
- At this point, the system display a graphical screen with a status bar indicating load status.
- XP disables interrupts during phase 0 and enables them before phase 1.
- The HAL is called to prepare the interrupt controller; the Memory Manager, Object Manager, Security Reference Monitor, and Process Manager are initialized.
- Phase 1 begins when the HAL is called to prepare the system to accept interrupts from devices.
- If more than one processor is present the additional processors are initialized at this point.
- All Executive subsystems are reinitialized in the following order:
    1. Object Manager
    2. Executive
    3. Microkernel
    4. Security Reference Monitor
    5. Memory Manager
    6. Cache Manager
    7. LPCS
    8. I/O Manager
    9. Process Manager

Step 19: I/O Manager

- The initialization of I/O Manager begins the process of loading all the systems driver files.

- Picking up where NTLDR left off, it first finishes the loading of boot devices. Next it assembles a prioritized list of drivers and attempts to load each in turn.
- The failure of a driver to load may prompt NT to reboot and try to start the system using the values stored in the Last Known Good Configuration.

Step 20: SMSS (Session Manager Subsystem)

- The last task for phase 1 initialization of the kernel is to launch the Session Manager Subsystem (SMSS).
- SMSS is responsible for creating the usermode environment that provides the visible interface to NT.
- SMSS runs in user-mode but unlike other user-mode applications SMSS is considered a trusted part of the operating system and is also a native application (it uses only core Executive functions).
- These two features allow SMSS to start the graphics subsystem and login processes.

Step 21: win32k.sys

- SMSS loads the win32k.sys device driver which implements the Win32 graphics subsystem.
- Shortly after win32k.sys starts it switches the screen into graphics mode. The Services Subsystem now starts all services mark as Auto Start.
- Once all devices and services are started the boot is deemed successful and this configuration is saved as the Last Known Good Configuration.

Step 22: Logon

- The XP boot process is not considered complete until a user has successfully logged onto the system.
- The process is begun by the WINLOGON.EXE file which is loaded as a service by the kernel and continued by the Local Security Authority (LSASS.EXE) which displays the logon dialog box.
- This dialog box appears at approximately the time that the Services Subsystem starts the network service.

Some important Short Forms and their Full Forms

- ACP : Accelerated Graphics Port
- AMD : Advanced Micro Devices
- ARM : Advanced RISC Machines
- AVR : Advanced Virtual RISC
- BCU : Bus Control Unit
- BIOS : Basic Input Output System
- BIU : Basic Instruction Unit
- BRD : Blu Ray Disc
- CD : Compact Disc
- CISC : Complex Instruction Set Computing
- CPU : Central Processing Unit
- CRT : Cathode Ray Tube
- DDR RAM : Double Data Rate RAM
- DMA : Direct Memory Access
- DRAM : Dynamic Random Access Memory
- DVD : Digital Versatile Disc
- ECU : Execution Control Unit
- EDO RAM : Extended Data Output RAM
- EEPROM : Electrically Erasable Programmable ROM
- EPIC : Explicit Parallel Instruction Computing
- EPROM : Erasable Programmable ROM
- eSATA : External SATA
- EU : Execution Unit
- FSB : Front Side Bus
- HDMI : High Definition Multimedia Interface
- HTTP : Hyper Text Templates
- ICH : Intel Chipset
- IDB : Internal Data Bus
- IDE : Integrated Drive Electronics
- IRQ : Interrupt ReQuest
- IVT : Interrupt Vector Table
- Kbps : Kilo Bytes per second
- LAN : Local Area Network
- LCD : Liquid Crystal Display
- LD : LASER Disc
- LED : Light Embedded Diode
- LPDDR RAM : Low Power DDR RAM
- M68K : Motorola 68000
- Mbps : Mega Bytes Per second

- MICR : Magnetic Ink Character Recognizer
- MIPS : Microprocessor with Interlocked Pipelined
- MIPS : Millions of Instruction Per Second
- OCR : Optical Character Recognizer
- PATA : Parallel Advanced Technology Attachment
- PCI : Peripheral Component Interface
- PnP : Plug and Play
- POST : Power ON Self Test
- PROM : Programmable ROM
- RAID : Redundant Array of Inexpensive Disk
- RAM : Random Access Memory
- RCU : Register Control Unit
- RISC : Reduced Instruction Set Computing
- ROM : Read Only Memory
- SATA : Serial Advanced Technology Attachment
- SCSI : Small Computer System Interface
- SDRAM : Synchronous Dynamic RAM
- SH4 : Super HITACHI 4
- SMPS : Switch Mode Power Supply
- SPARC : Sun Processor Architecture
- SRAM : Static Random Access Memory
- SSD : Solid State Drive
- TFT : Thin Film Transistors
- USB : Universal Serial Bus
- UVEPROM : Ultra Violet Erasable Programmable ROM
- VAX : Virtual Address eXtention
- VDU : Visible Display Unit
- VGA : Video/Visual Graphic Adapter
- WAN : Wide Area Networks

Units To Measure Data

4 Bits	1 Nibble
8 Bits	1 Byte
1024 Bytes	1 Kilo Byte (KB)
1024 KB	1 Mega Byte (MB)
1024 MB	1 Gyga Byte (GB)
1024 GB	1 Tera Byte (TB)
1024 TB	1 Peta Byte (PB)
1024 PB	1 Exa Byte (EB)
1024 EB	1 Zetta Byte (ZB)
1024 ZB	1 Yotta Byte (YB)