Unit – I

Syllabus:

- **Knowing the Computer**

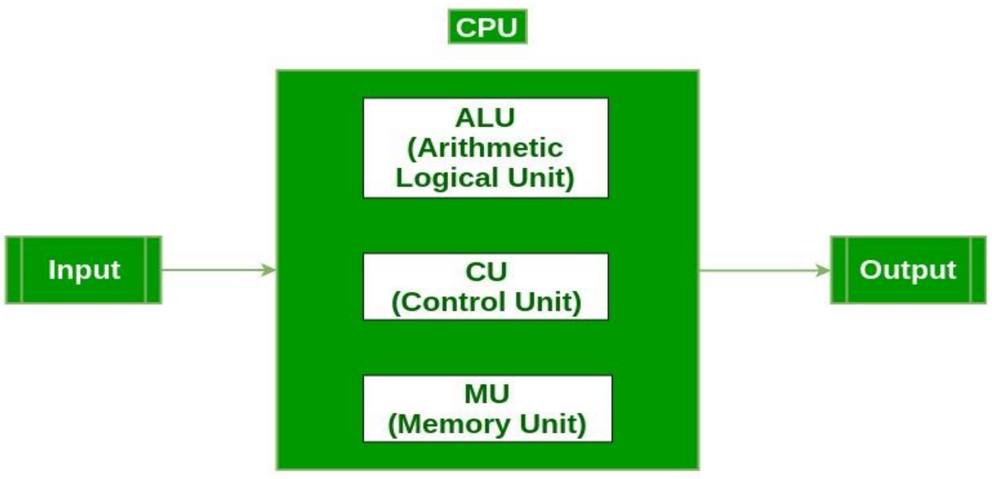
Definition and Block Diagram of a Computer. Basic parts of a computer (Memory, CPU, Input, and Output), Memory hierarchy, Circuits and Logic, Hardware vs Software, Representation of Data in memory (integer (including negative), floating points etc. to text, images, audio), Principle of Abstraction, Operating System, Language Hierarchy - Machine Language to High Level Language, Compiler, Interpreter, The Command Line Interface (basic linux commands).

What is Computer?

A computer is a programmable electronic device that accepts raw data as input and processes it with a set of instructions (a program) to produce the result as output.

It can process numerical as well as non-numerical calculations. The term "computer" is derived from the Latin word "computare" which means to calculate.

Block Diagram of a Computer:



Input Unit:

The input unit consists of input devices that are attached to the computer. These devices take input and convert it into binary language that the computer understands. Some of the common input devices are keyboard, mouse, joystick, scanner etc.

Central Processing Unit (CPU) :

Once the information is entered into the computer by the input device, the processor processes it. The CPU is called the brain of the computer because it is the control center of the computer. It first fetches instructions from memory and then interprets them so as to know what is to be done. If required, data is fetched from memory or input device. Thereafter CPU executes or performs the required computation and then either stores the output or displays on the output device. The CPU has three main components which are responsible for different functions – Arithmetic Logic Unit (ALU), Control Unit (CU) and Memory registers.

- **Arithmetic and Logic Unit (ALU) :**

  The ALU, as its name suggests performs mathematical calculations and takes logical decisions. Arithmetic calculations include addition, subtraction, multiplication and division. Logical decisions involve comparison of two data items to see which one is larger or smaller or equal.
- **Control Unit :**

  The Control unit coordinates and controls the data flow in and out of CPU and also controls all the operations of ALU, memory registers and also input/output units. It is also responsible for carrying out all the instructions stored in the program. It decodes the fetched instruction, interprets it and sends control signals to input/output devices until the required operation is done properly by ALU and memory.
- **MEMORY UNIT:**

  Memory unit is a component of a computer system. It is used to store data, instructions and information. It is actually a work area of computer, where the CPU stores the data and instruction. It is also known as a main/primary/internal-memory.

  There are two types of memory: -

1. **Read only memory (ROM):-** ROM is a part of the memory unit. This is read only memory. It cannot be used to written. ROM is used in situations where the data must be held permanently.
2. **Random access memory (RAM):-** RAM is also part of memory unit. It is used for temporary storage of program data. Its data is lost when power is turned off.

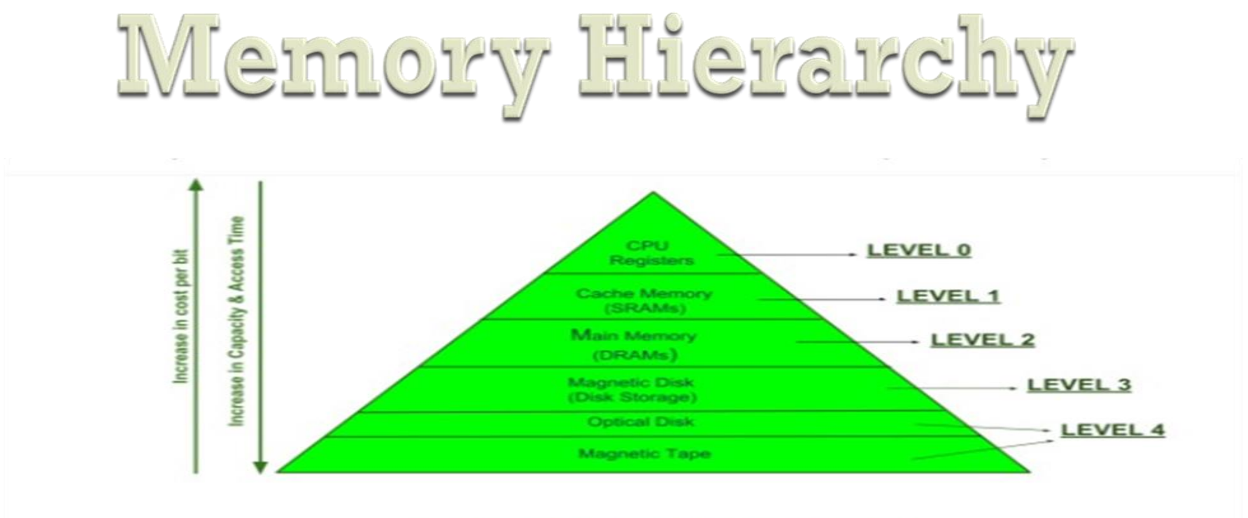
Output Unit:

The output unit consists of output devices that are attached with the computer. It converts the binary data coming from CPU to human understandable form. The common output devices are monitor, printer, plotter etc.

Memory hierarchy:

In the Computer System Design, Memory Hierarchy is an enhancement to organize the memory such that it can minimize the access time. The Memory Hierarchy was developed based on a program behavior known as locality of references.

The figure below clearly demonstrates the different levels of memory hierarchy:



MEMORY HIERARCHY DESIGN

- This Memory Hierarchy Design is divided into 2 main types:

1. **External Memory or Secondary Memory –**
   Comprising of Magnetic Disk, Optical Disk, Magnetic Tape i.e. peripheral storage devices which are accessible by the processor via I/O Module.
2. **Internal Memory or Primary Memory –**
   Comprising of Main Memory, Cache Memory & CPU registers. This is directly accessible by the processor.

We can infer the following characteristics of Memory Hierarchy Design from above figure:

1. **Capacity:**
   It is the global volume of information the memory can store. As we move from top to bottom in the Hierarchy, the capacity increases.
2. **Access Time**:
   It is the time interval between the read/write request and the availability of the data. As we move from top to bottom in the Hierarchy, the access time increases.
3. **Performance:**
   Earlier when the computer system was designed without Memory Hierarchy design, the speed gap increases between the CPU registers and Main Memory due to large difference in access time. This results in lower performance of the system and thus, enhancement was required. This enhancement was made in the form of Memory Hierarchy Design because of which the performance of the system increases. Performance is minimizing how far down the memory hierarchy one has to go to manipulate data.

4. **Cost per bit:**
As we move from bottom to top in the Hierarchy, the cost per bit increases i.e. Internal Memory is costlier than External Memory.

✓ **Register Memory:**

- Register memory is the smallest and fastest memory in a computer. It is not a part of the main memory and is located in the CPU in the form of registers, which are the smallest data holding elements.
- A register temporarily holds frequently used data, instructions, and memory address that are to be used by CPU. They hold instructions that are currently processed by the CPU. All data is required to pass through registers before it can be processed. So, they are used by CPU to process the data entered by the users.
- Registers hold a small amount of data around 32 bits to 64 bits. The speed of a CPU depends on the number and size (no. of bits) of registers that are built into the CPU. Registers can be of different types based on their uses. Some of the widely used Registers include Accumulator or AC, Data Register or DR, the Address Register or AR, Program Counter (PC), I/O Address Register, and more.

✓ **Cache Memory**

- Cache memory is a high-speed memory, which is small in size but faster than the main memory (RAM). The CPU can access it more quickly than the primary memory. So, it is used to synchronize with high-speed CPU and to improve its performance.
- Cache memory can only be accessed by CPU. It can be a reserved part of the main memory or a storage device outside the CPU. It holds the data and programs which are frequently used by the CPU. So, it makes sure that the data is instantly available for CPU whenever the CPU needs this data.
- In other words, if the CPU finds the required data or instructions in the cache memory, it doesn't need to access the primary memory (RAM). Thus, by acting as a buffer between RAM and CPU, it speeds up the system performance.

✓ **Primary Memory**

- Primary Memory is of two types**:** **RAM** and **ROM**.
- **RAM (Volatile Memory**)
- It is a volatile memory. It means it does not store data or instructions permanently. When you switch on the computer the data and instructions from the hard disk are stored in RAM.
- CPU utilizes this data to perform the required tasks. As soon as you shut down the computer the RAM loses all the data.

- **ROM (Non-volatile Memory)**
- It is a non-volatile memory. It means it does not lose its data or programs that are written on it at the time of manufacture. So it is a permanent memory that contains all important data and instructions needed to perform important tasks like the boot process.

✓ **Secondary Memory**

- The secondary storage devices which are built into the computer or connected to the computer are known as a secondary memory of the computer. It is also known as external memory or auxiliary storage.
- The secondary memory is accessed indirectly via input/output operations. It is non-volatile, so permanently stores the data even when the computer is turned off or until this data is overwritten or deleted.
- The CPU can't directly access the secondary memory. First, the secondary memory data is transferred to primary memory then the CPU can access it.

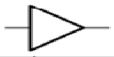
Circuits and Logic

- Logic gates perform basic logical functions and are the fundamental building blocks of digital integrated circuits.
- Most logic gates take an input of two binary values, and output a single value of a 1 or 0.

- There are seven different types of logic gates

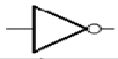
- **AND** - True if A and B are both True
- **OR** - True if either A or B are True
- **NOT** - Inverts value: True if input is False; False if input is True
- **XOR** - True if either A or B are True, but False if both are True
- **NAND** - AND followed by NOT: False only if A and B are both True
- **NOR** - OR followed by NOT: True only if A and B are both False
- **XNOR** - XOR followed by NOT: True if A and B are both True or both False

YES



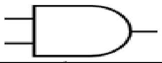
INPUT	OUTPUT
A	
0	0
1	1

NOT



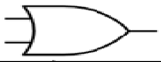
INPUT	OUTPUT
A	
0	1
1	0

AND



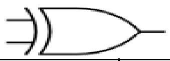
INPUT		OUTPUT
A	B	
0	0	0
1	0	0
0	1	0
1	1	1

OR



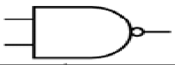
INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	1

XOR



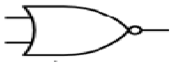
INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	0

NAND



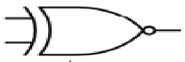
INPUT		OUTPUT
A	B	
0	0	1
1	0	1
0	1	1
1	1	0

NOR



INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	0

XNOR



INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	1

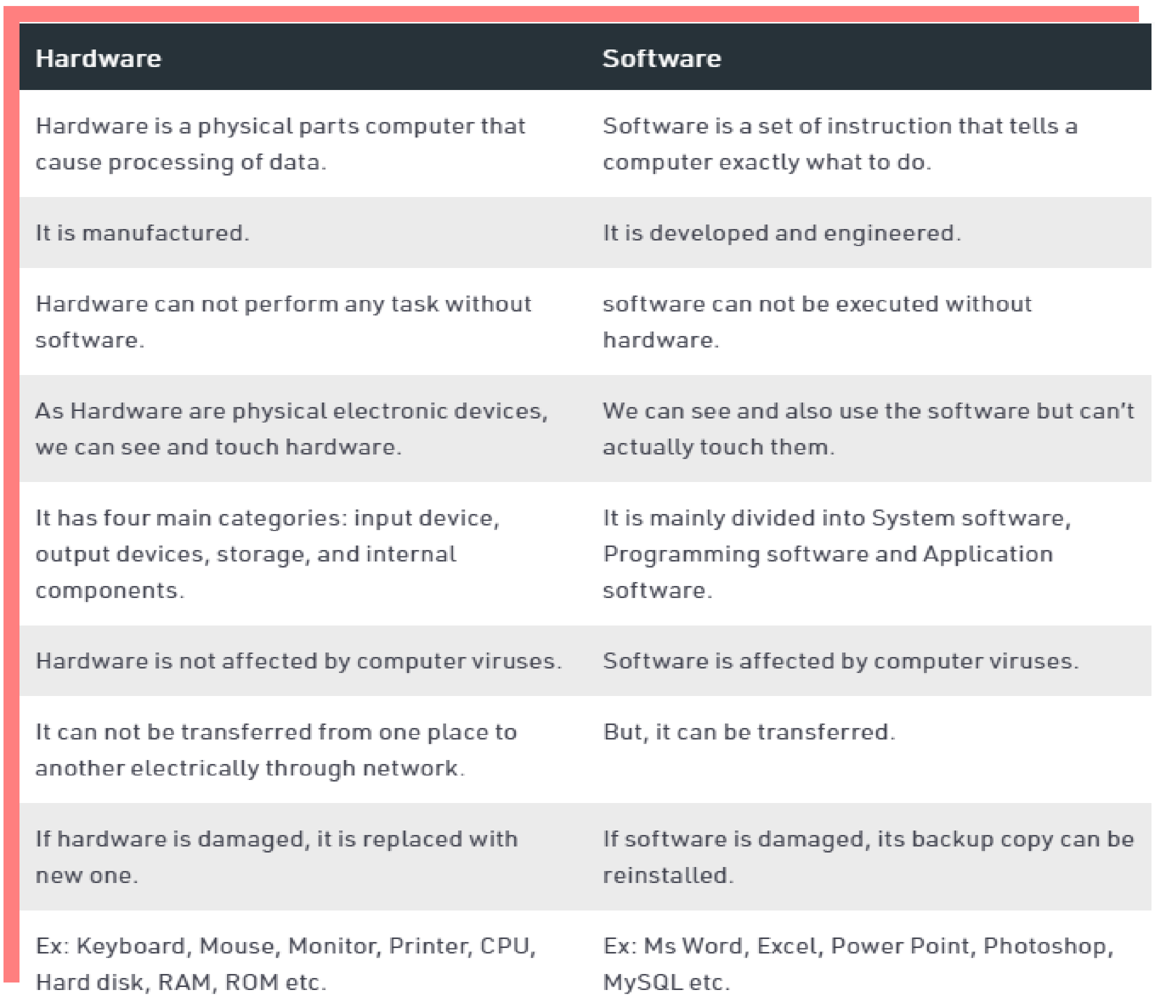
Hardware vs Software

Computer Hardware:

- Hardware refers to the physical components of a computer. Computer Hardware is any part of the computer that we can touch these parts.
- These are the primary electronic devices used to build up the computer.
- Examples of hardware in a computer are the Processor, Memory Devices, Monitor, Printer, Keyboard, Mouse, and the Central Processing Unit.

Computer Software:

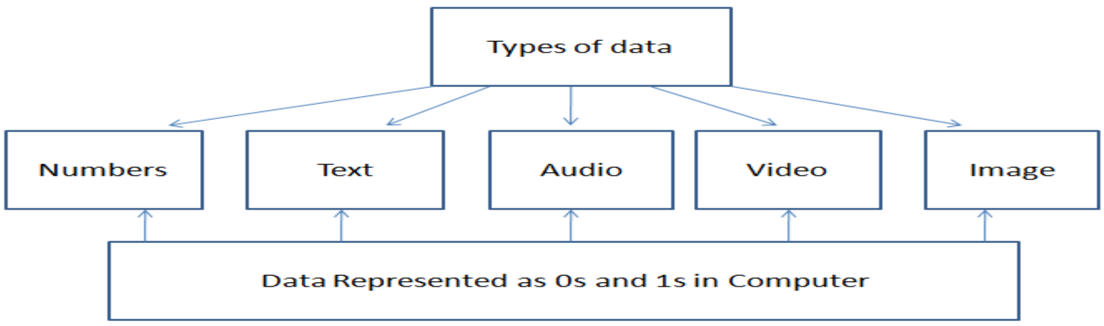
- Software is a collection of instructions, procedures, documentation that performs different tasks on a computer system.
- we can say also Computer Software is a programming code executed on a computer processor. The code can be machine-level code or the code written for an operating system.
- Examples of software are Ms Word, Excel, Power Point, Google Chrome, Photoshop, MySQL etc.



Hardware	Software
Hardware is a physical parts computer that cause processing of data.	Software is a set of instruction that tells a computer exactly what to do.
It is manufactured.	It is developed and engineered.
Hardware can not perform any task without software.	software can not be executed without hardware.
As Hardware are physical electronic devices, we can see and touch hardware.	We can see and also use the software but can't actually touch them.
It has four main categories: input device, output devices, storage, and internal components.	It is mainly divided into System software, Programming software and Application software.
Hardware is not affected by computer viruses.	Software is affected by computer viruses.
It can not be transferred from one place to another electrically through network.	But, it can be transferred.
If hardware is damaged, it is replaced with new one.	If software is damaged, its backup copy can be reinstalled.
Ex: Keyboard, Mouse, Monitor, Printer, CPU, Hard disk, RAM, ROM etc.	Ex: Ms Word, Excel, Power Point, Photoshop, MySQL etc.

Representation of Data in memory:

- Computer uses a fixed number of bits to represent a piece of data, which could be a number, a character, or others.
- A n-bit storage location can represent up to 2^n distinct entities.
- For example, a 3-bit memory location can hold one of these eight binary patterns: 000, 001, 010, 011, 100, 101, 110, or 111. Hence, it can represent at most 8 distinct entities.
- You could use them to represent numbers 0 to 7, numbers 8881 to 8888, characters 'A' to 'H', or up to 8 kinds of fruits like apple, orange, banana; or up to 8 kinds of animals like lion, tiger, etc.



1. Number Systems

- Human beings use ***decimal*** (base 10) and ***duodecimal*** (base 12) number systems for counting and measurements (probably because we have 10 fingers and two big toes).
- Computers use ***binary*** (base 2) number system, as they are made from binary digital components (known as transistors) operating in two states - on and off.
- In computing, we also use ***hexadecimal*** (base 16) or ***octal*** (base 8) number systems, as a ***compact*** form for representing binary numbers.

1.1 Decimal (Base 10) Number System

- Decimal number system has ten symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9, called digits. It uses positional notation. That is, the least-significant digit (right-most digit) is of the order of 10^0 (units or ones), the second right-most digit is of the order of 10^1 (tens), the third right-most digit is of the order of 10^2 (hundreds), and so on, where ^ denotes exponent.
- For example,

```
➢   735 = 700 + 30 + 5 = 7×10^2 + 3×10^1 + 5×10^0
```

1.2 Binary (Base 2) Number System

- Binary number system has two symbols: 0 and 1, called ***bits***. It is also a ***positional notation***,
- for example,

10110B = 10000B + 0000B + 100B + 10B + 0B = 1×2^4 + 0×2^3 + 1×2^2 + 1×2^1 + 0×2^0

- We shall denote a binary number with a suffix B. Some programming languages denote binary numbers with prefix 0b or 0B (e.g., 0b1001000), or prefix b with the bits quoted (e.g., b'10001111').
- A binary digit is called a ***bit***. Eight bits is called a ***byte*** (why 8-bit unit? Probably because 8=23).

1.3 Hexadecimal (Base 16) Number System

- Hexadecimal number system uses 16 symbols:
  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F, called ***hex digits***. It is a ***positional notation***.
- for example,

- A3EH = A00H + 30H + EH = 10×16^2 + 3×16^1 + 14×16^0

- We shall denote a hexadecimal number (in short, hex) with a suffix H.
- Some programming languages denote hex numbers with prefix 0x or 0X (e.g., 0x1A3C5F),

1.4 Octal (Base 8) Number System

- In the octal number system, the base is 8 and it uses numbers from 0 to 7 to represent numbers.
- Octal numbers are commonly used in computer applications.
- Octal numbers are indicated by the addition of either a 0o prefix.

Integer Representation

- Integers are ***whole numbers*** or ***fixed-point numbers*** with the radix point ***fixed*** after the least-significant bit.
- Computers use a fixed number of bits to represent an integer. The commonly-used bit-lengths for integers are 8-bit, 16-bit, 32-bit or 64-bit.
- Besides bit-lengths, there are two representation schemes for integers:

1. **Unsigned Integers**: can represent zero and positive integers.
2. **Signed Integers:** can represent zero, positive and negative integers. Three representation schemes had been proposed for signed integers:
    a. Sign-Magnitude representation
    b. 1's Complement representation
    c. 2's Complement representation

1 *n*-bit Unsigned Integers

- Unsigned integers can represent zero and positive integers, but not negative integers.
- The value of an unsigned integer is interpreted as "***the magnitude of its underlying binary pattern***".
- Example 1: Suppose that ***n***=8 and the binary pattern is 0100 0001B, the value of this unsigned integer is 1×2^0 + 1×2^6 = 65D.
- Example 2: Suppose that ***n***=16 and the binary pattern is 0001 0000 0000 1000B, the value of this unsigned integer is 1×2^3 + 1×2^12 = 4104D.
- Example 3: Suppose that ***n***=16 and the binary pattern is 0000 0000 0000 0000B, the value of this unsigned integer is 0.
- An n-bit pattern can represent 2^n distinct integers. An n-bit unsigned integer can represent integers from 0 to (2^n)-1, as tabulated below:

n	Minimum	Maximum
8	0	(2^8)-1 (=255)
16	0	(2^16)-1 (=65,535)
32	0	(2^32)-1 (=4,294,967,295) (9+ digits)
64	0	(2^64)-1 (=18,446,744,073,709,551,615) (19+ digits)

2 Signed Integers

- Signed integers can represent zero, positive integers, as well as negative integers. Three representation schemes are available for signed integers:

1. Sign-Magnitude representation
2. 1's Complement representation
3. 2's Complement representation

- In all the above three schemes, the ***most-significant bit*** (msb) is called the ***sign bit***.
- The sign bit is used to represent the ***sign*** of the integer -: with 0 for positive integers and 1 for negative integers.
- The ***magnitude*** of the integer, however, is interpreted differently in different schemes.

1. Signed Magnitude Method:

- In the signed magnitude method number is divided into two parts: Sign bit and magnitude.
- The most-significant bit (msb) is the sign bit, with value of 0 representing positive integer and 1 representing negative integer.
- The remaining n-1 bits represent the magnitude (absolute value) of the integer. The absolute value of the integer is interpreted as "the magnitude of the (n-1)-bit binary pattern".
- **Example 1**: Suppose that n=8 and the binary representation is 0 100 0001B.
  Sign bit is 0 -> positive
  Absolute value is 100 0001B = 65D
  Hence, the integer is +65D
- **Example 2**: Suppose that n=8 and the binary representation is 1 000 0001B.
  Sign bit is 1 => negative
  Absolute value is 000 0001B = 1D
  Hence, the integer is -1D

2. Sign Integers in 1's Complement Representation:

In 1's complement representation:

- The most significant bit (msb) is the sign bit, with value of 0 representing positive integers and 1 representing negative integers.
- The remaining n-1 bits represents the magnitude of the integer, as follows:
  - ✓ For positive integers, the absolute value of the integer is equal to "the magnitude of the (n-1)-bit binary pattern".
  - ✓ For negative integers, the absolute value of the integer is equal to "the magnitude of the complement (inverse) of the (n-1)-bit binary pattern" (hence called 1's complement).
- **Example 1**: Suppose that n=8 and the binary representation 0 100 0001B.
  Sign bit is 0 ⇒ positive
  Absolute value is 100 0001B = 65D
  Hence, the integer is +65D
- **Example 2**: Suppose that n=8 and the binary representation 1 000 0001B.
  Sign bit is 1 ⇒ negative
  Absolute value is the complement of 000 0001B, i.e., 111 1110B = 126D
  Hence, the integer is -126D

3. Sign Integers in 2's Complement Representation:

In 2's complement representation:

- The most significant bit (msb) is the sign bit, with value of 0 representing positive integers and 1 representing negative integers.
- The remaining n-1 bits represents the magnitude of the integer, as follows:
  - ✓ For positive integers, the absolute value of the integer is equal to "the magnitude of the (n-1)-bit binary pattern".

- ✓ For negative integers, the absolute value of the integer is equal to "the magnitude of the complement of the (n-1)-bit binary pattern plus one" (hence called 2's complement).

- **Example 1**: Suppose that n=8 and the binary representation 0 100 0001B

  Sign bit is 0 ⇒ positive

  Absolute value is 100 0001B = 65D

  Hence, the integer is +65D

- **Example 2:** Suppose that n=8 and the binary representation 1 000 0001B.

  Sign bit is 1 ⇒ negative

Absolute value is the complement of 000 0001B plus 1, i.e., 111 1110B + 1B = 127

Hence, the integer is -127D

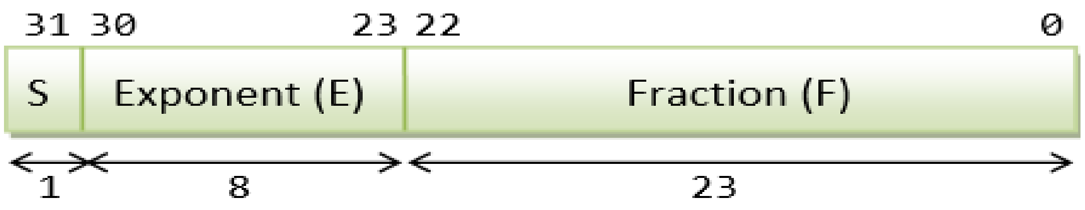
Floating-Point Number Representation:

- A floating-point number (or real number) can represent a very large (1.23×10^88) or a very small (1.23×10^-88) value.
- It could also represent very large negative number (-1.23×10^88) and very small negative number (-1.23×10^88), as well as zero.
- A floating-point number is typically expressed in the scientific notation, with a fraction (F), and an exponent (E) of a certain radix (r), in the form of F×r^E. Decimal numbers use radix of 10 (F×10^E);
- While binary numbers use radix of 2 (F×2^E).
- Representation of floating point number is not unique.
- For example, the number 55.66 can be represented as 5.566×10^1, 0.5566×10^2, 0.05566×10^3, and so on.
- The fractional part can be normalized. In the normalized form, there is only a single non-zero digit before the radix point.
- For example, decimal number 123.4567 can be normalized as 1.234567×10^2;
- Binary number 1010.1011B can be normalized as 1.0101011B×2^3.
- In computers, floating-point numbers are represented in scientific notation of fraction (F) and exponent (E) with a radix of 2, in the form of F×2^E.
- Both E and F can be positive as well as negative. Modern computers adopt IEEE 754 standard for representing floating-point numbers.
- There are two representation schemes: 32-bit single-precision and 64-bit double-precision.

1. **IEEE-754 32-bit Single-Precision Floating-Point Numbers:**

In 32-bit single-precision floating-point representation:

- ✓ The most significant bit is the sign bit (S), with 0 for positive numbers and 1 for negative numbers.
- ✓ The following 8 bits represent exponent (E).
- ✓ The remaining 23 bits represents fraction (F).

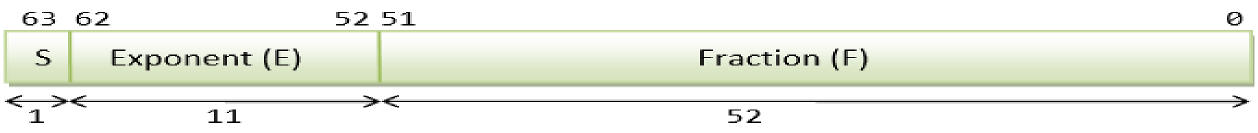


32-bit Single-Precision Floating-point Number

2. IEEE-754 64-bit Double-Precision Floating-Point Numbers:

The representation scheme for 64-bit double-precision is similar to the 32-bit single-precision:

- ✓ The most significant bit is the sign bit (S), with 0 for positive numbers and 1 for negative numbers.
- ✓ The following 11 bits represent exponent (E).
- ✓ The remaining 52 bits represents fraction (F).



64-bit Double-Precision Floating-point Number

Character representation in memory:

- ➢ Like numeric data even a character can’t be stored as it is, it will be first converted into its equal binary and then stores into the bits of memory.
- ➢ Every character on the keyboard has its equal binary value. The decimal equal to that binary value is called ASCII (American standard code for information interchange) value.
- ➢ Say for example equal binary value to character ‘A’ is 01000001, the decimal equal to which is 64. So the ASCII value of ‘A’ is 64.
- ➢ Let us see another example, say equal binary value to character ‘a’ is 01100001, the decimal equal to which is 97. So the ASCII value of ‘a’ is 97.

Operating System

- An operating system is a program that controls the execution of application programs and acts as an interface between the user of a computer and the computer hardware.

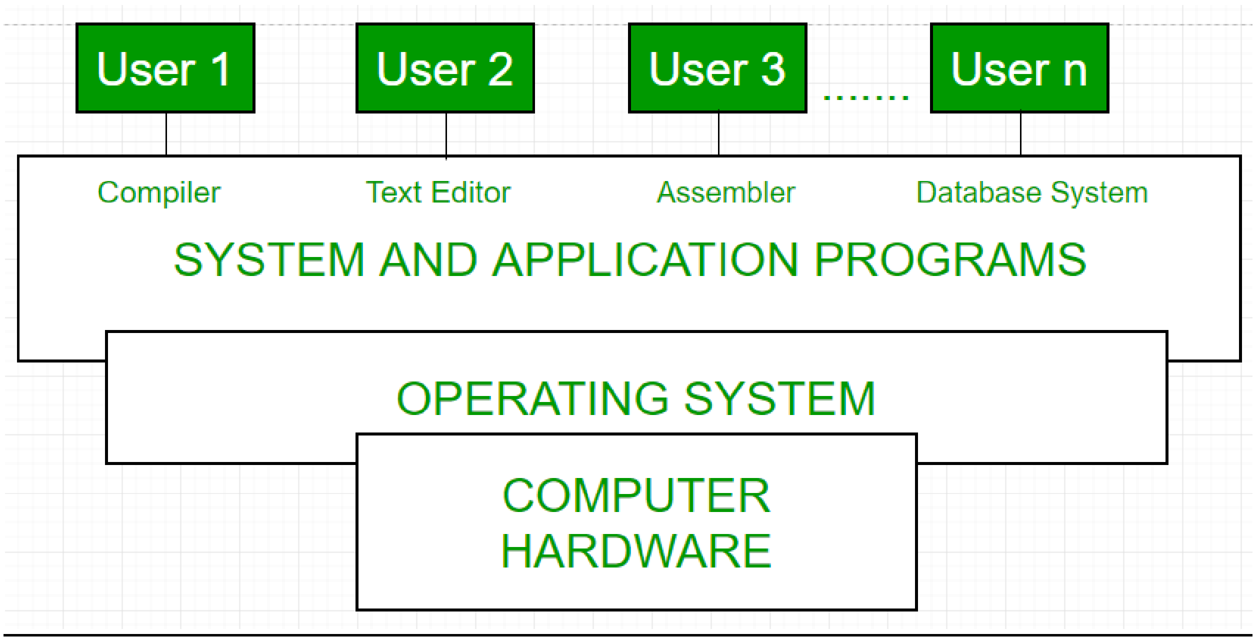
Functions of Operating system – Operating system performs three functions:

- **Convenience:** An OS makes a computer more convenient to use.
- **Efficiency:** An OS allows the computer system resources to be used in an efficient manner.
- **Ability to Evolve:** An OS should be constructed in such a way as to permit the effective development, testing and introduction of new system functions at the same time without interfering with service.

Operating system as User Interface –

1. User
2. System and application programs
3. Operating system
4. Hardware

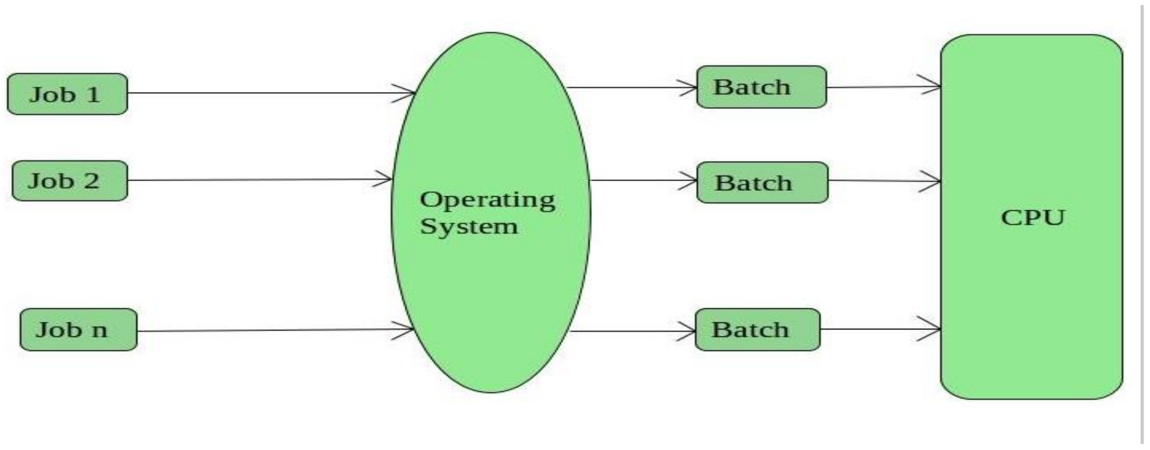
Every general-purpose computer consists of the hardware, operating system, system programs, and application programs. The hardware consists of memory, CPU, ALU, and I/O devices, peripheral device, and storage device. System program consists of compilers, loaders, editors, OS, etc. The application program consists of business programs, database programs.



Types of Operating Systems:

1. Batch Operating System –

- This type of operating system does not interact with the computer directly. There is an operator which takes similar jobs having the same requirement and groups them into batches. It is the responsibility of the operator to sort jobs with similar needs.



Advantages of Batch Operating System:

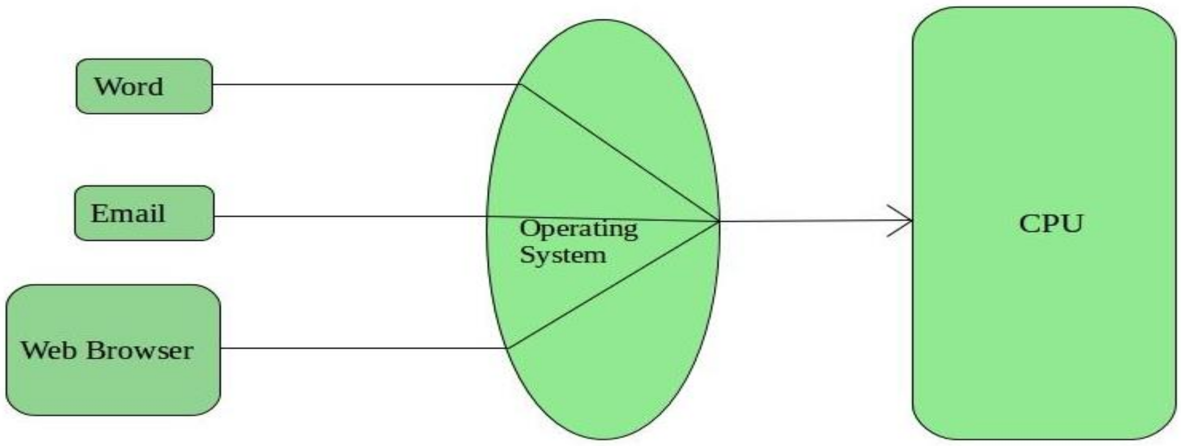
- ✓ It is very difficult to guess or know the time required for any job to complete. Processors of the batch systems know how long the job would be when it is in queue.
- ✓ Multiple users can share the batch systems
- ✓ The idle time for the batch system is very less
- ✓ It is easy to manage large work repeatedly in batch systems

Disadvantages of Batch Operating System:

- ✓ The computer operators should be well known with batch systems
- ✓ Batch systems are hard to debug
- ✓ It is sometimes costly
- ✓ The other jobs will have to wait for an unknown time if any job fails.
- ✓ Examples of Batch based Operating System: Payroll System, Bank Statements, etc.

2. Time-Sharing Operating Systems –

- ➢ Each task is given some time to execute so that all the tasks work smoothly. Each user gets the time of CPU as they use a single system. These systems are also known as Multitasking Systems.
- ➢ The task can be from a single user or different users also. The time that each task gets to execute is called quantum. After this time interval is over OS switches over to the next task.



Advantages of Time-Sharing OS:

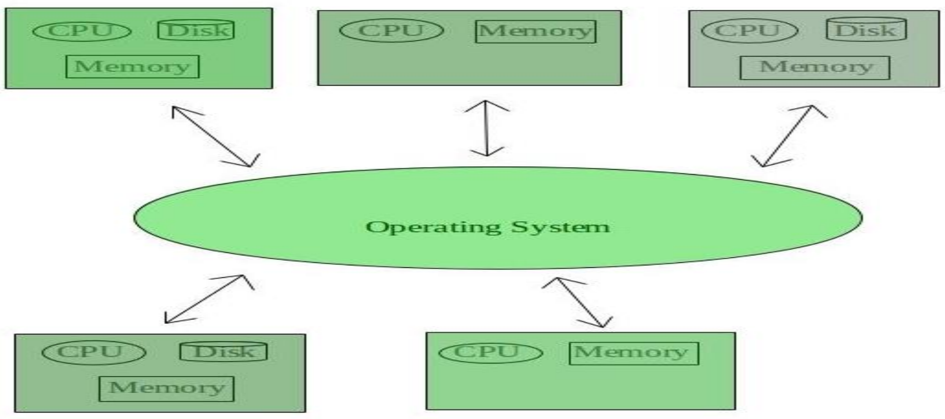
- ✓ Each task gets an equal opportunity
- ✓ Fewer chances of duplication of software
- ✓ CPU idle time can be reduced

Disadvantages of Time-Sharing OS:

- ✓ Reliability problem
- ✓ One must have to take care of the security and integrity of user programs and data
- ✓ Data communication problem
- ✓ Examples of Time-Sharing OSs are: Multics, Unix, etc.

3. Distributed Operating System –

- ➢ These types of the operating system are a recent advancement in the world of computer technology and are being widely accepted all over the world and, that too, with a great pace.
- ➢ Various autonomous interconnected computers communicate with each other using a shared communication network. Independent systems possess their own memory unit and CPU.
- ➢ These are referred to as **loosely coupled systems** or distributed systems. These system's processors differ in size and function.



Advantages of Distributed Operating System:

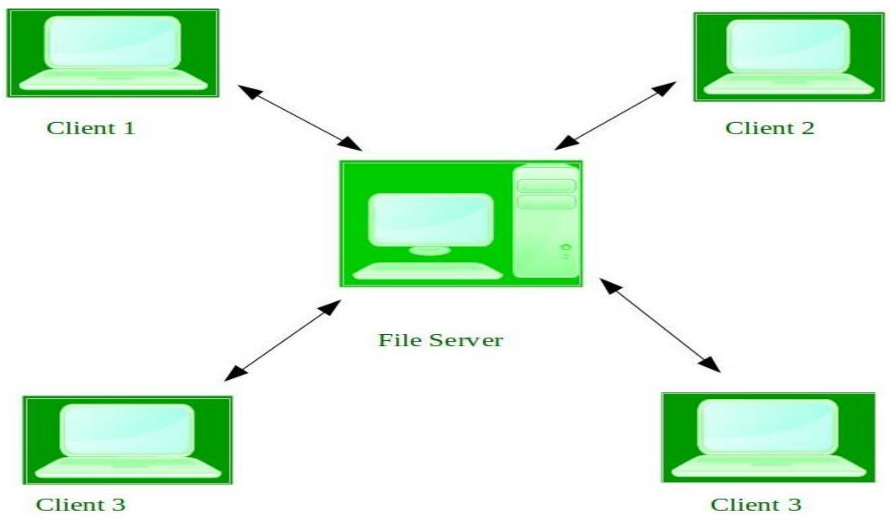
- ✓ Failure of one will not affect the other network communication, as all systems are independent from each other
- ✓ Electronic mail increases the data exchange speed
- ✓ Since resources are being shared, computation is highly fast and durable
- ✓ Load on host computer reduces
- ✓ These systems are easily scalable as many systems can be easily added to the network
- ✓ Delay in data processing reduces

Disadvantages of Distributed Operating System:

- ✓ Failure of the main network will stop the entire communication
- ✓ To establish distributed systems the language which is used are not well defined yet
- ✓ These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well yet
- ✓ Examples of Distributed Operating System are- LOCUS, etc.

4. Network Operating System –

- ➢ These systems run on a server and provide the capability to manage data, users, groups, security, applications, and other networking functions. These types of operating systems allow shared access of files, printers, security, applications, and other networking functions over a small private network.
- ➢ One more important aspect of Network Operating Systems is that all the users are well aware of the underlying configuration, of all other users within the network, their individual connections, etc. and that's why these computers are popularly known as **tightly coupled systems**.



Advantages of Network Operating System:

- ✓ Highly stable centralized servers
- ✓ Security concerns are handled through servers
- ✓ New technologies and hardware up-gradation are easily integrated into the system

- ✓ Server access is possible remotely from different locations and types of systems

Disadvantages of Network Operating System:

- ✓ Servers are costly
- ✓ User has to depend on a central location for most operations
- ✓ Maintenance and updates are required regularly
- ✓ Examples of Network Operating System are: Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD, etc.

In an operating system software performs each of the function:

1. **Process management**: - Process management helps OS to create and delete processes. It also provides mechanisms for synchronization and communication among processes.

2. **Memory management**: - Memory management module performs the task of allocation and de-allocation of memory space to programs in need of these resources.

3. **File management**: - It manages all the file-related activities such as organization storage, retrieval, naming, sharing, and protection of files.

4. **Device Management**: Device management keeps tracks of all devices. This module also responsible for this task is known as the I/O controller. It also performs the task of allocation and de-allocation of the devices.

5. **I/O System Management**: One of the main objects of any OS is to hide the peculiarities of those hardware devices from the user.

6. **Secondary-Storage Management**: Systems have several levels of storage which includes primary storage, secondary storage, and cache storage. Instructions and data must be stored in primary storage or cache so that a running program can reference it.

7. **Security:-** Security module protects the data and information of a computer system against malware threat and authorized access.

8. **Command interpretation**: This module is interpreting commands given by the and acting system resources to process that commands.

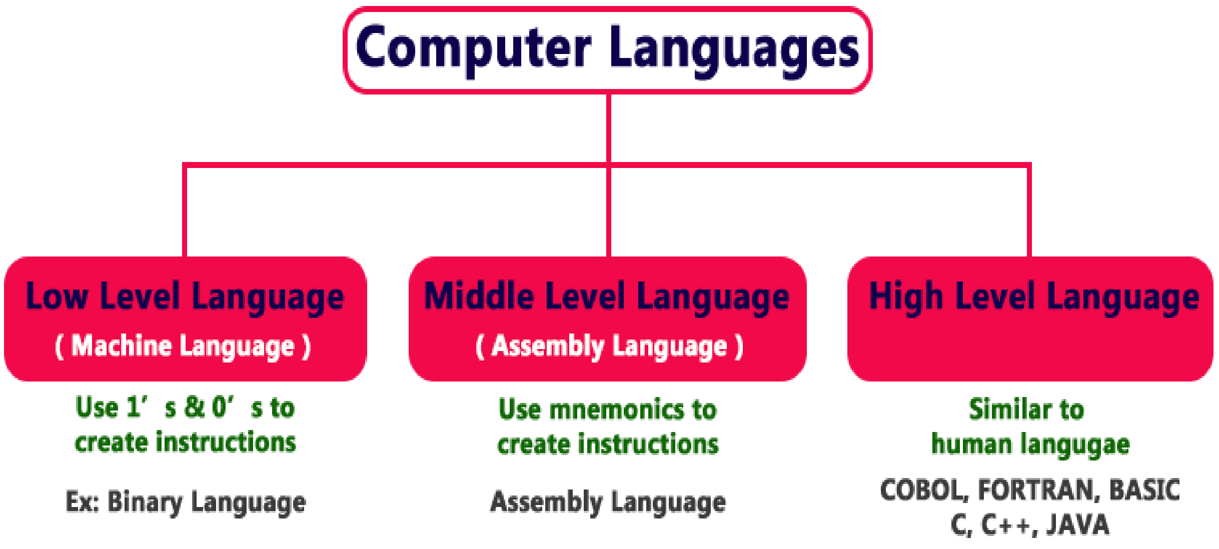
9. **Networking:** A distributed system is a group of processors which do not share memory, hardware devices, or a clock. The processors communicate with one another through the network.

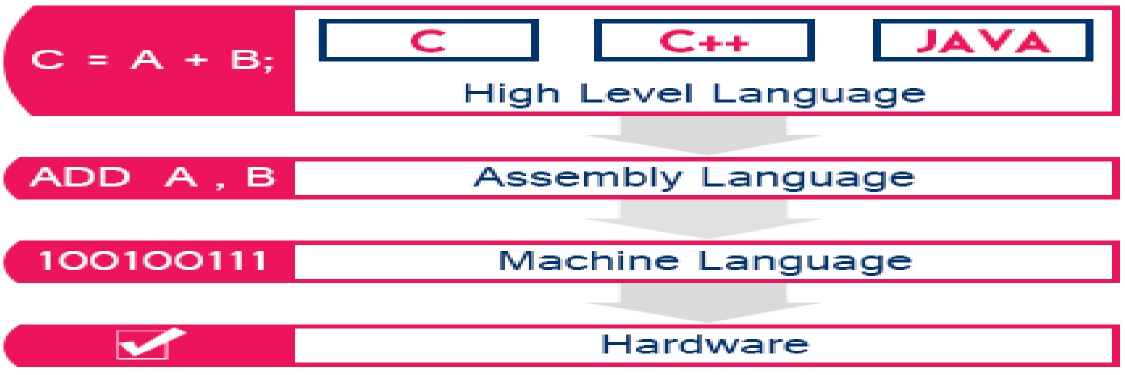
10. **Job accounting**: Keeping track of time & resource used by various job and users.

11. **Communication management**: Coordination and assignment of compilers, interpreters, and another software resource of the various users of the computer systems.

Language Hierarchy:

Computer languages have been evolved from Low-Level to High-Level Languages. In the earliest days of computers, only Binary Language was used to write programs.





Machine-level language:

- The machine-level language is a language that consists of a set of instructions that are in the binary form 0 or 1.
- As we know that computers can understand only machine instructions, which are in binary digits, i.e., 0 and 1, so the instructions given to the computer can be only in binary codes.

- ➢ Creating a program in a machine-level language is a very difficult task as it is not easy for the programmers to write the program in machine instructions.
- ➢ It is error-prone as it is not easy to understand, and its maintenance is also very high.
- ➢ A machine-level language is not portable as each computer has its machine instructions, so if we write a program in one computer will no longer be valid in another computer.

Advantages of Machine Language:

- No translation required.
- Takes less execution time.

Disadvantages of machine Language:

- Difficult to use
- Machine dependent
- Error prone
- Difficult to debug and modify

➢ **Assembly Language:**

A program written with mnemonic codes forms an assembly language program.

- ➢ The assembly language contains some human-readable commands such as mov, add, sub, etc.
- ➢ The problems which we were facing in machine-level language are reduced to some extent by using an extended form of machine-level language known as assembly language.
- ➢ Since assembly language instructions are written in English words like mov, add, sub, so it is easier to write and understand.
- ➢ As we know that computers can only understand the machine-level instructions, so we require a translator that converts the assembly code into machine code.
- ➢ The translator used for translating the code is known as an assembler.

Advantages of Assembly Language:

- More convenient than Machine language.
- Written in the form of symbolic instructions this improves readability.

Disadvantages of Assembly Language:

- Specific to a Machine architecture.(machine dependent)
- Programming is difficult and time consuming.
- The programmer should know all about the logical structure of the computer.

Machine-level language	Assembly language
The machine-level language comes at the lowest level in the hierarchy, so it has zero abstraction level from the hardware.	The assembly language comes above the machine language means that it has less abstraction level from the hardware.
It cannot be easily understood by humans.	It is easy to read, write, and maintain.
The machine-level language is written in binary digits, i.e., 0 and 1.	The assembly language is written in simple English language, so it is easily understandable by the users.
It does not require any translator as the machine code is directly executed by the computer.	In assembly language, the assembler is used to convert the assembly code into machine code.
It is a first-generation programming language.	It is a second-generation programming language.

- **High-Level Language:**
- The high-level language is a programming language that allows a programmer to write the programs which are independent of a particular type of computer.

- The high-level languages are considered as high-level because they are closer to human languages than machine-level languages.
- When writing a program in a high-level language, then the whole attention needs to be paid to the logic of the problem.
- A compiler is required to translate a high-level language into a low-level language.
- The high-level languages are designed to overcome the limitation of low-level language, i.e., portability.
- The high-level language is portable; i.e., these languages are machine-independent.

Advantages:

- Code is machine independent
- Easy to learn and use.
- Less errors and easy to maintain.
- Easy to detect and correct errors.

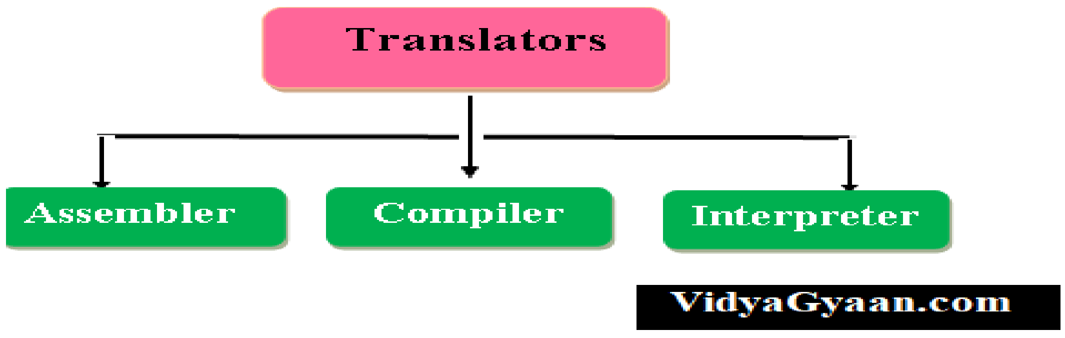
Disadvantages:

- Code may not be optimized.
- Difficult to write some logic.

Low-level language	High-level language
It is a machine-friendly language, i.e., the computer understands the machine language, which is represented in 0 or 1.	It is a user-friendly language as this language is written in simple English words, which can be easily understood by humans.
The low-level language takes more time to execute.	It executes at a faster pace.
It requires the assembler to convert the assembly code into machine code.	It requires the compiler to convert the high-level language instructions into machine code.
The machine code cannot run on all machines, so it is not a portable language.	The high-level code can run all the platforms, so it is a portable language.
It is memory efficient.	It is less memory efficient.
Debugging and maintenance are not easier in a low-level language.	Debugging and maintenance are easier in a high-level language.

Language Processors:

- The programs are written mostly in high level languages like Java, C++, Python etc. and are called source code.
- These source codes cannot be executed directly by the computer and must be converted into machine language to be executed.
- Hence, a special translator system software is used to translate the program written in high-level language into machine code is called **Language Processor** and the program after translated into machine code.

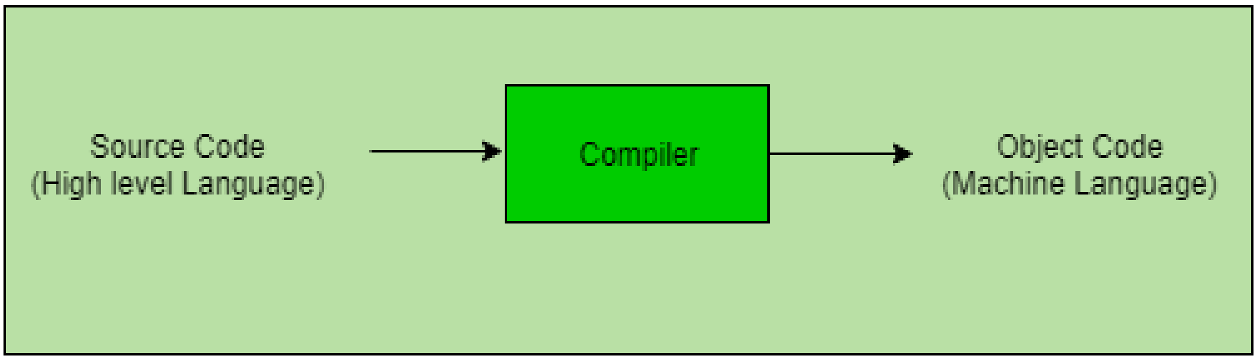


➢ **Compiler –**

➢ The language processor that reads the complete source program written in high level language as a whole in one go and translates it into an equivalent program in machine language is called as a Compiler.

➢ Example: C, C++, C#, Java In a compiler, the source code is translated to object code successfully if it is free of errors.

➢ The compiler specifies the errors at the end of compilation with line numbers when there are any errors in the source code. The errors must be removed before the compiler can successfully recompile the source code again.

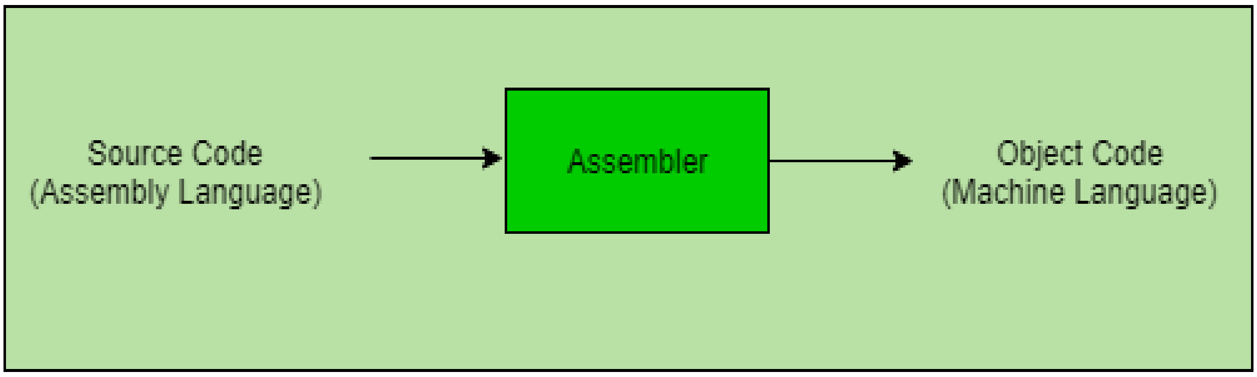


➢ **Assembler –**

➢ The Assembler is used to translate the program written in Assembly language into machine code.

➢ The source program is a input of assembler that contains assembly language instructions.

➢ The output generated by assembler is the object code or machine code understandable by the computer.



➢ **Interpreter –**

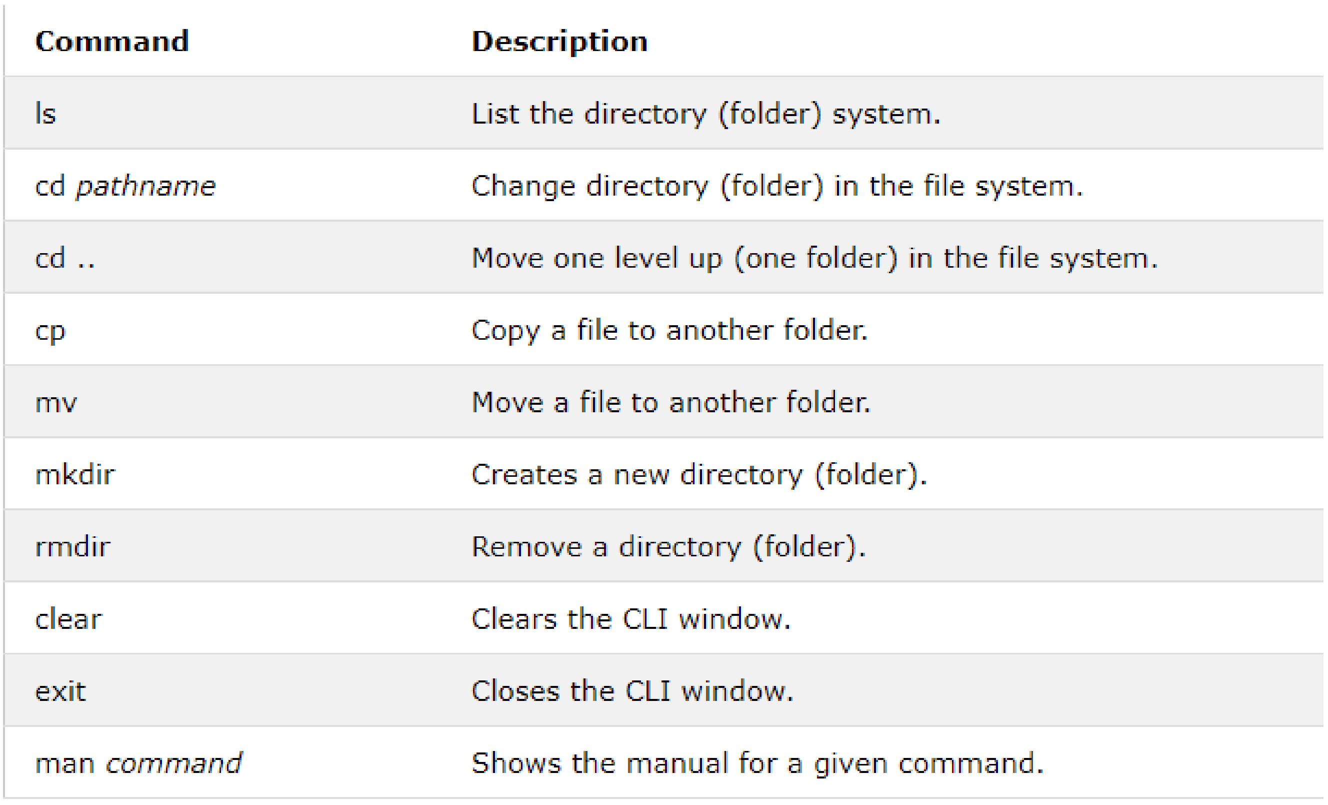
- ➢ The translation of single statement of source program into machine code is done by language processor and executes it immediately before moving on to the next line is called an interpreter.
- ➢ If there is an error in the statement, the interpreter terminates its translating process at that statement and displays an error message.
- ➢ The interpreter moves on to the next line for execution only after removal of the error.
- ➢ An Interpreter directly executes instructions written in a programming or scripting language without previously converting them to an object code or machine code.
- ➢ Example: Perl, Python and Matlab.

COMPARISON	COMPILER	INTERPRETER
Input	It takes an entire program at a time.	It takes a single line of code or instruction at a time.
Output	It generates intermediate object code.	It does not produce any intermediate object code.
Working mechanism	The compilation is done before execution.	Compilation and execution take place simultaneously.
Speed	Comparatively faster	Slower
Memory	Memory requirement is more due to the creation of object code.	It requires less memory as it does not create intermediate object code.
Errors	Display all errors after compilation, all at the same time.	Displays error of each line one by one.
Error detection	Difficult	Easier comparatively
Pertaining Programming languages	C, C++, C#, Scala, typescript uses compiler.	PHP, Perl, Python, Ruby uses an interpreter.

The Command Line Interface:

- CLI is a command line program that accepts text input to execute operating system functions.
- In the 1960s, using only computer terminals, this was the only way to interact with computers.
- In the 1970s an 1980s, command line input was commonly used by Unix systems and PC systems like MS-DOS and Apple DOS.
- Today, with graphical user interfaces (GUI), most users never use command-line interfaces (CLI).
- However, CLI is still used by software developers and system administrators to configure computers install software, and access features that are not available in the graphical interface.

Basic Linux CLI Commands



Command	Description
ls	List the directory (folder) system.
cd *pathname*	Change directory (folder) in the file system.
cd ..	Move one level up (one folder) in the file system.
cp	Copy a file to another folder.
mv	Move a file to another folder.
mkdir	Creates a new directory (folder).
rmdir	Remove a directory (folder).
clear	Clears the CLI window.
exit	Closes the CLI window.
man *command*	Shows the manual for a given command.

FOR DETAIL EXPLAINATION WITH OUTPUTS ABOUT BASIC LINUX COMMNADS, VISIT THE WEBSITE GIVEN BELOW

https://www.guru99.com/must-know-linux-commands.html