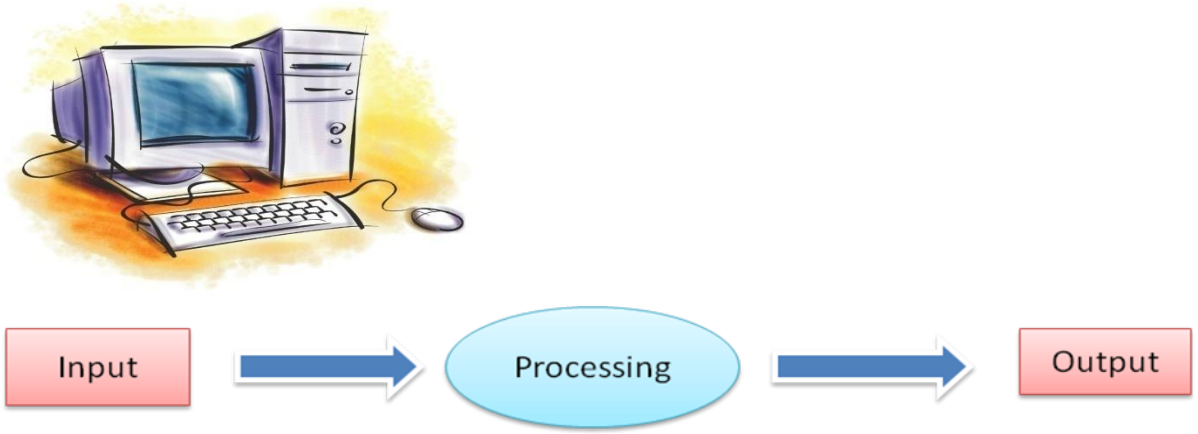
Unit – 1: Knowing the Computer

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 and video), Principle of Abstraction, Language Hierarchy - Machine Language to High Level Language, Compiler, Interpreter, The Command Line Interface (basic linux commands)

Computer:

A computer is an electronic device which accepts data as input, performs operations on the data based on the instructions stored in the memory and produces output.

General representation of a computer is as shown below:



- **Data** is collection of raw facts and figures.
- **Information** comprises processed data to provide answers to *who, what, where* and *when* type of questions.
- Early computers used vacuum tubes. Technological advances led to new generation of computers that were considerably smaller, faster and less expensive than previous ones.
- The elements of a computer fall into two major categories
    1. Hardware
    2. Software
- **Hardware** is the equipment used to perform the necessary computations and includes central processing unit, monitor, keyboard, mouse, printer and speakers.
- The **software** is the collection of programs that allow the hardware to serve its purpose. The purpose of the software is to use the underlying hardware components.

- **Program** – a list of instructions that enable a computer to perform a specific task
- **Instructions** are the commands given to the computer that tells what it has to do.

Characteristics of a computer:

- ✓ Speed
- ✓ Accuracy
- ✓ Automatic
- ✓ Diligence
- ✓ Versatile
- ✓ Memory
- ✓ No IQ
- ✓ Economical

Basic applications of computer:

- ✓ Communication
- ✓ Desktop Publishing
- ✓ Government
- ✓ Traffic control
- ✓ Legal systems
- ✓ Retail business
- ✓ Music
- ✓ Movies
- ✓ Travel and tourism
- ✓ Business and industry
- ✓ Hospitals
- ✓ Simulation
- ✓ Geology
- ✓ Astronomy
- ✓ Weather forecasting
- ✓ Education
- ✓ Online banking
- ✓ Industry and Engineering
- ✓ Robots
- ✓ Sports

Block Diagram of a Computer

The five major operations performed by a computer are

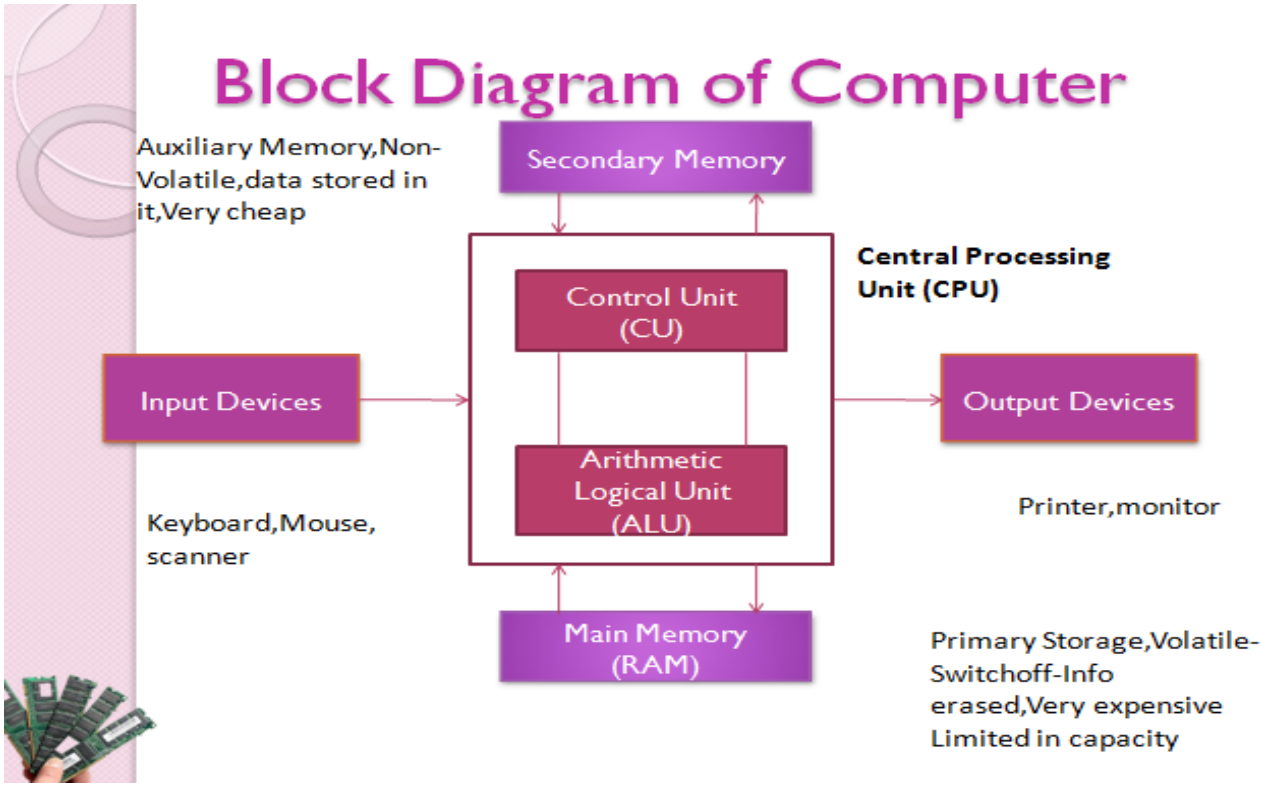
1. Accepting data or instructions (input)
2. Storing data
3. Processing data
4. Displaying results (output)
5. Controlling and coordinating all operations inside a computer.

- ✓ Before execution of the program, it should be transferred from secondary storage to main memory.
- ✓ Data need to be processed is entered through an input device and stored in main memory, where it can be accessed and manipulated by the CPU. Results are stored back in main memory.
- ✓ The information in main memory can be displayed through an output device.

- ✓ **Input:** is the process of entering data and instructions into the computer. Various input devices used are keyboard, mouse, scanner, trackball etc.

- ✓ **Storage**: is the process of storing data and instructions in the computer. Computer has two types of storage areas.

The architecture or structure of a computer is as shown below:



1. Primary storage :

- ✓ Is also known as main memory.
- ✓ Directly accessible to the CPU at a very fast speed.
- ✓ Used to store data, programs, intermediate and final results.
- ✓ Very expensive therefore limited in capacity.
- ✓ Volatile – as soon as computer is switched off, the information stored in it gets erased.

2. Secondary storage:

- ✓ Also known as secondary memory or auxiliary memory.
- ✓ It is cheaper, non-volatile and used to permanently store data and programs

Processing: The process of performing operations on the data as per the instructions specified by the user (program). Main aim of this is to transform data into information. Data and instructions are taken from the primary memory and are transferred to the Arithmetic and Logic Unit, which performs all sorts of calculations. When the processing completes, the final result is transferred to the main memory.

Output: is the process of giving the result of data processing to the outside world. Various output devices are monitor, printer etc.

Controlling : The function of managing, coordinating, and controlling all the components of the computer system is handled by the control unit. The control unit decides the manner in which the instructions will be executed and the operations will be performed.

Computer Hardware

The essential hardware components of a computer are:

1. CPU (Central Processing Unit)
2. Memory
3. I/O (Input/Output) Devices

Central Processing Unit:

- Coordinates all computer operations.
- Performs arithmetic and logical operations on data.
- CPU follows the instructions contained in a computer program to determine which operations should be carried out and in what order.
- The CPU can perform arithmetic operations such as addition, subtraction, multiplication and division.
- The CPU can also compare the contents of two memory cells and make decisions based on the results of that comparison.
- The circuitry of modern CPU is housed in a single integrated circuit or chip.
- An IC that is full CPU is called a microprocessor.
- CPU contains special high speed memory locations called registers.

Control Unit(CU):

- ✓ It is responsible for controlling the transfer of data and instructions among other units of a computer.
- ✓ It manages and coordinates all the units of the computer.
- ✓ It communicates with Input/Output devices for transfer of data or results from storage.

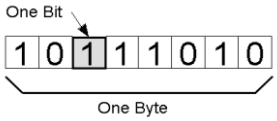
- ✓ It does not process or store data.

Arithmetic Logical Unit(ALU):

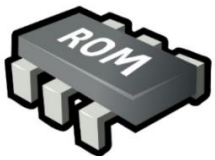
- ✓ It performs arithmetic operations like addition, subtraction, multiplication, and division.
- ✓ It performs logic operations such as comparing, selecting, matching, and merging of data.
- ✓ Examples of logic operations are comparisons of values such as NOT, AND, and OR.

Memory:

- Memory is an essential component in any computer.
- Memory cell is an individual storage location in memory.
- Address of memory cell is the relative position of a memory cell in the computers main memory.
- The data stored in the memory cell are called contents of the cell.
- The ability to store programs as well as data is called stored program concept. A program's instructions must be stored in main memory before they can be executed.
- A memory cell is actually a grouping of smaller units called bytes. A byte is the amount of storage required to store a single character. A byte is even composed of smaller units of storage called bit. Bit refers **B**inary dig**it.**
- Binary refers to a number system based on two numbers, 0 and 1. Therefore a bit is either 0 or 1.
- There are eight bits in a byte.



- **Main Memory:** stores programs, data and results. Most computers have two types of Main Memory.
  1. Random Access Memory(RAM)
  2. Read-Only Memory (ROM)



- **RAM:**
  - Offers temporary storage of programs and data while the programs are temporarily being executed by the computer.
  - RAM is usually volatile memory, which means that everything in RAM will be lost when the computer is switched off.
  - Accessible to the programmer.
- **ROM:**
  - Stores information permanently in the computer.

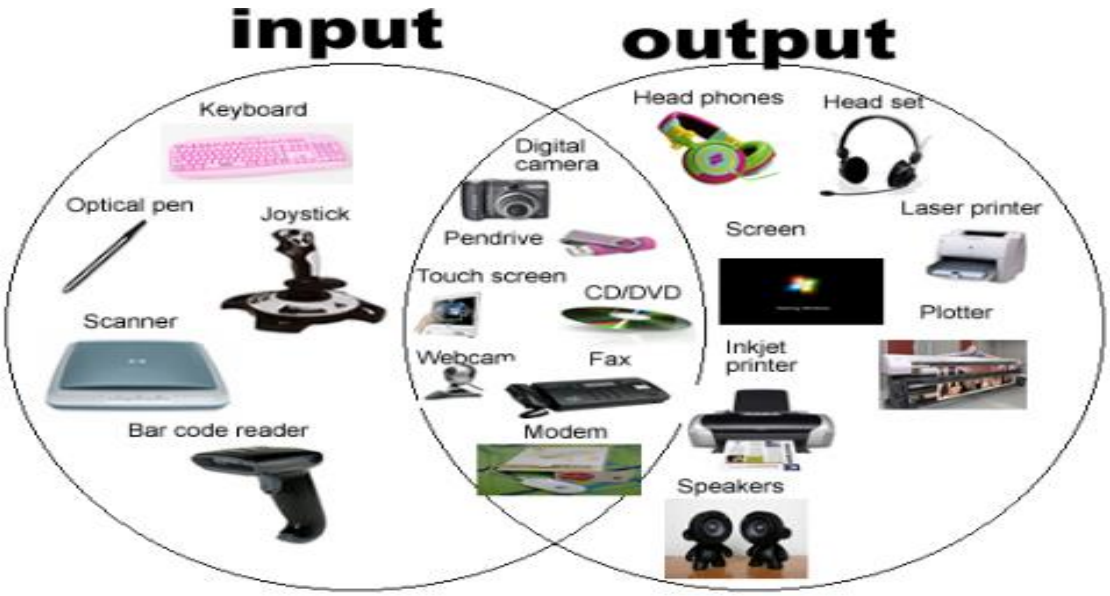
- Its name is read-only, because computer can read but cannot write information in ROM.
- ROM is non-volatile. Data stored in ROM do not disappear when the computer is switched off.

- **Secondary storage devices :** these devices are needed for the following reasons.

- ✓ Computers need permanent / semi permanent storage, so that information can be retained during a power-loss or when the computer is turned off.
- ✓ Systems typically store more information than will fit in memory.
- ✓ Some of the most frequently used secondary storage devices : compact disk, magnetic tape, floppy disk, hard disk, zip disk, digital video disk.

Input / output devices:

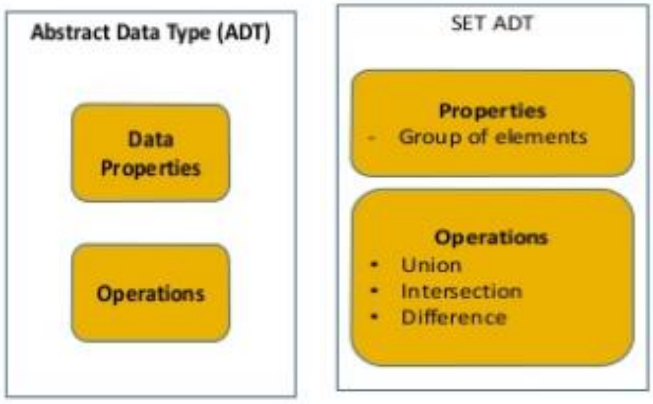
- Input / output devices are used to communicate with the computer.
- They allow us to enter data for a computation and to observe the results of that computation.



Principle of Abstraction:

Abstraction is the process of trying to identify the most important or inherent qualities of an object or model, and ignoring or omitting the unimportant aspects.(information hiding)
It consists of two parts: data and Operations

Real Life Example of Abstraction



Memory Hierarchy

- ✓ Memory hierarchy is the hierarchy of memory and storage devices found in a computer system.
- ✓ It ranges from the slowest but high capacity auxiliary memory to the fastest but low capacity cache memory.

Need-

There is a trade-off among the three key characteristics of memory namely-

- ✓ Cost
- ✓ Capacity
- ✓ Access time

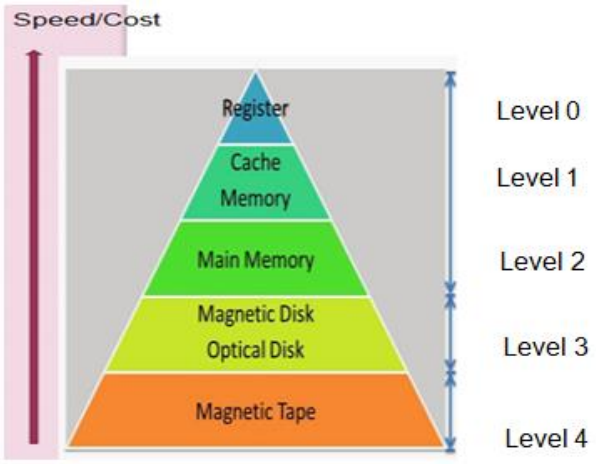
Level-0:

- ✓ At level-0, registers are present which are contained inside the CPU.
- ✓ Since they are present inside the CPU, they have least access time.
- ✓ They are most expensive and therefore smallest in size (in KB).
- ✓ Registers are implemented using **Flip-Flops**.

Level-1:

- ✓ At level-1, **Cache Memory** is present.
- ✓ It stores the segments of program that are frequently accessed by the processor.
- ✓ It is expensive and therefore smaller in size (in MB).

- ✓ Cache memory is implemented using static RAM.



Level-2:

- ✓ At level-2, main memory is present.
- ✓ It can communicate directly with the CPU and with auxiliary memory devices through an I/O processor.
- ✓ It is less expensive than cache memory and therefore larger in size (in few GB).
- ✓ Main memory is implemented using dynamic RAM.

Level-3:

- ✓ At level-3, secondary storage devices like **Magnetic Disk** are present.
- ✓ They are used as back up storage.
- ✓ They are cheaper than main memory and therefore much larger in size (in few TB).

Level-4:

- ✓ At level-4, tertiary storage devices like magnetic tape are present.
- ✓ They are used to store removable files.
- ✓ They are cheapest and largest in size (1-20 TB).

Observations-

The following observations can be made when going down in the memory hierarchy-

- ✓ Cost / bit decreases
- ✓ Frequency of access decreases
- ✓ Capacity increases
- ✓ Access time increases

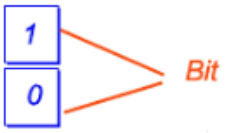
Units of storage in Computer

Computers are electronic machines which operate using binary logic. These devices use two different values (0 and 1) to represent data. This corresponding number system is referred as Binary number system.

Differences between binary and decimal number systems.

	Decimal	binary
base	10	2
Digits	0 to 9	0, 1

A ***bit*** is the smallest unit of information that can be stored or manipulated on a computer; it consists of either zero or one. We can also call a *bit* a *binary* digit. A single bit can store two different values (either 0 or 1)

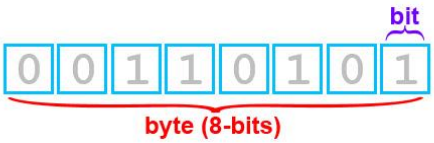


A **Nibble** is a group of four bits.

Byte : is a group of eight bits.

- A byte is a simply a fixed-length sequence of bits. Modern computers organize data into bytes to increase the data processing efficiency of network equipment, disks and memory. Computers by convention set one byte to equal eight (8) bits.
- A byte can store 2^8 or 256 different values.

A **word** is a group of 16 bits.



Besides bytes, data is also specified using the units shown in the table

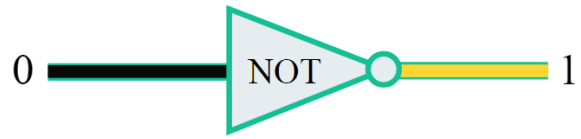
Unit	Abbreviation	Equals to	Bytes
Byte	Byte	8 bits	2^0 bytes
Kilobyte	KB	1024 bytes	2^{10} bytes
Megabyte	MB	1024 KB	2^{20} bytes
Gigabyte	GB	1024 MB	2^{30} bytes
Terabyte	TB	1024 GB	2^{40} bytes

Circuits and Logic:

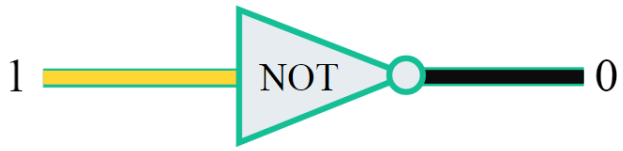
The basic digital electronic circuit that has one or more inputs and single output is known as **Logic gate.**
Logic gates are divided into three types.They are
1.Basic Gates:NOT,AND,OR
2.Universal Gates :NAND,NOR
3.Arithmetic Gates:EXOR,EXNOR

NOT Gate:

The simplest gate is the NOT gate, also known as an inverter. It accepts a single input and outputs the opposite value.
If the input is 0, the output is 1:



If the input is 111, the output is 000:



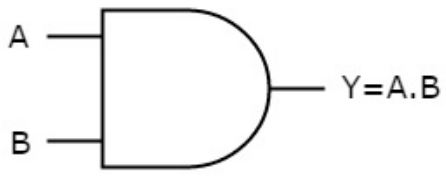
Truth table for NOT

Input	Output
A	Y
0	1
1	0

AND Gate:

If both inputs are '1', then only the output, Y is '1'. For remaining combinations of inputs, the output, Y is '0'. It is optional to represent the **Logical AND** with the symbol '.'.

The following figure shows the **symbol** of an AND gate, which is having two inputs A, B and one output, Y.



The following table shows the **truth table** of 2-input AND gate.

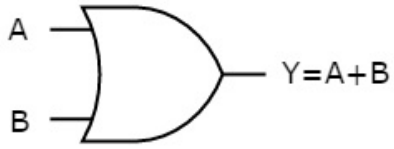
A	B	Y=A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate:

This **logical OR** is represented with the symbol '+'.

If both inputs are '0', then only the output, Y is '0'. For remaining combinations of inputs, the output, Y is '1'.

The following figure shows the **symbol** of an OR gate, which is having two inputs A, B and one output, Y.



The following table shows the **truth table** of 2-input OR gate.

A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1

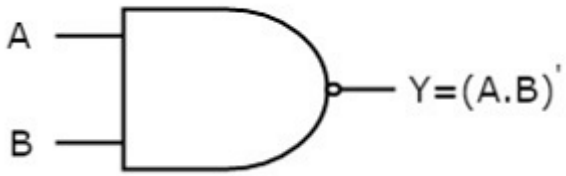
Universal gates

NAND & NOR gates are called as **universal gates**. Because we can implement any Boolean function, which is in sum of products form by using NAND gates alone. Similarly, we can implement any Boolean function, which is in product of sums form by using NOR gates alone.

NAND gate

NAND gate is a digital circuit that has two or more inputs and produces an output, which is the **inversion of logical AND** of all those inputs.

The following image shows the **symbol** of NAND gate, which is having two inputs A, B and one output, Y.



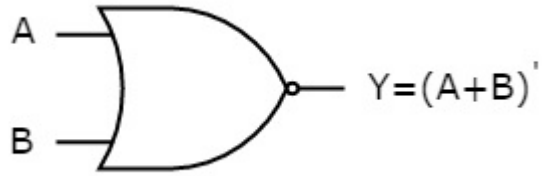
The following table shows the **truth table** of 2-input NAND gate.

A	B	Y=A.B	$(A.B)^I$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

NOR Gate:

NOR gate is a digital circuit that has two or more inputs and produces an output, which is the **inversion of logical OR** of all those inputs.

The following figure shows the **symbol** of NOR gate, which is having two inputs A, B and one output, Y.



The following table shows the **truth table** of 2-input NOR gate

A	B	Y=A+B	Y^I
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

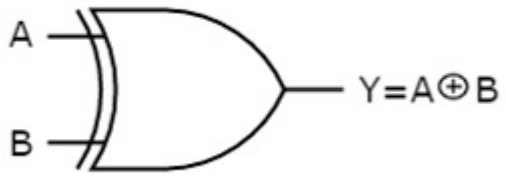
Arithmetic Gates:EXOR,EXNOR

Ex-OR & Ex-NOR gates are called as Arithmetic gates. Because, these two gates are special cases of OR & NOR gates.

Ex-OR gate

The full form of Ex-OR gate is **Exclusive-OR** gate. If both the inputs are 1 or both the inputs are 0,then obtained output is 0.And for the remaining cases the output is taken as 1.

Below figure shows the **symbol** of Ex-OR gate, which is having two inputs A, B and one output, Y.



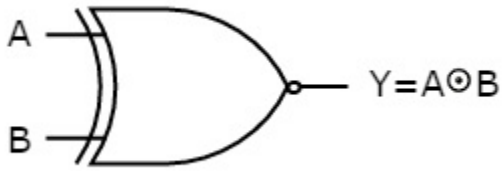
The following table shows the **truth table** of 2-input Ex-OR gate.

A	B	Y = A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Ex-NOR:

The full form of Ex-NOR gate is **Exclusive-NOR** gate. It is a inversion of Ex-OR

The following figure shows the **symbol** of Ex-NOR gate, which is having two inputs A, B and one output, Y.



The following table shows the **truth table** of 2-input Ex-NOR gate.

A	B	Y = A⊙B
0	0	1
0	1	0
1	0	0
1	1	1

Advantages of Logic Gates

- Logic gates are quick yet use low energy.
- Logic gates don't get overworked.
- Logic gates can lessen the prescribed number of I/O ports needed by a microcontroller.
- Logic gates can bring about straightforward data encryption and decryption.

Applications of Logic gates

- NAND Gates are used in Burglar alarms and buzzers.
- They are basically used in circuits involving computation and processing.
- They are also used in push button switches. E.g. Door Bell.
- They are used in the functioning of street lights.

Operating System:

The collection of computer programs that control the interaction of the user and computer hardware is called the operating system.

Its responsibilities are

- ✓ Communicating with the user- receiving commands and carrying them out.
- ✓ Managing allocation of memory, processor time, other resources for various tasks.
- ✓ Collecting input from the keyboard, mouse and other input devices, and providing this data to the currently running program.
- ✓ Conveying program output to the screen, printer or other output device.
- ✓ Accessing data from secondary storage.
- ✓ Writing data to secondary storage.

Widely used OS are – windows, linux.

The relationship between hardware, system software and application software is as shown in the below diagram:

Representation of Data in memory (integer (including negative), floating points etc. to text, images, audio and video):

Computer uses *a fixed number of bits* to represent a piece of data, which could be a number, a character, or others. For example, a 3-bit memory location can hold one of these eight binary patterns: 000, 001, 010, 011, 100, 101, 110, or 111.

Refer to Units of storage in Computer page no:9

Integer Representation:

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:

Unsigned Integers: We can represent zero and positive integers.

n-bit Unsigned Integers

An *n*-bit pattern can represent 2^*n* distinct integers. An *n*-bit unsigned integer can represent integers from 0 to (2^*n*)-1

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)

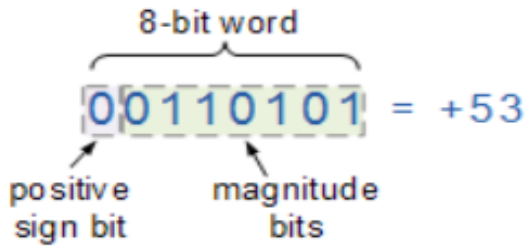
Signed Integers: We can represent zero, positive and negative integers. Three representation schemes had been proposed for signed integers:

- ✓ Sign-Magnitude representation
- ✓ 1's Complement representation
- ✓ 2's Complement representation

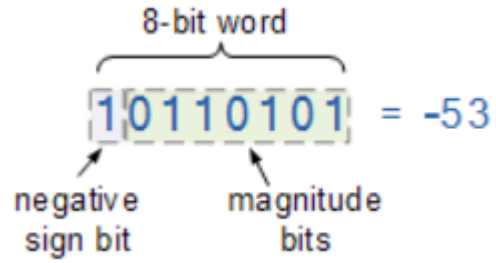
Sign-Magnitude representation:

Signed Binary Numbers use the MSB(most significant bit) as a sign bit to display a range of either positive numbers or negative numbers

Positive Signed Binary Numbers:

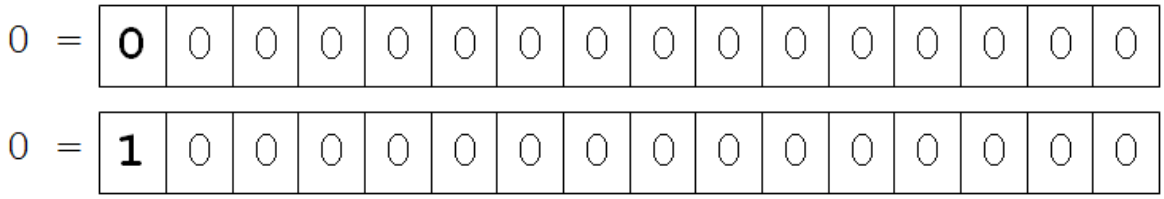


Negative Signed Binary Numbers



Disadvantages:

An unfortunate feature of Sign-Value representation is that there are two ways of representing the value zero: all bits set to zero, and all bits except the sign bit set to zero.



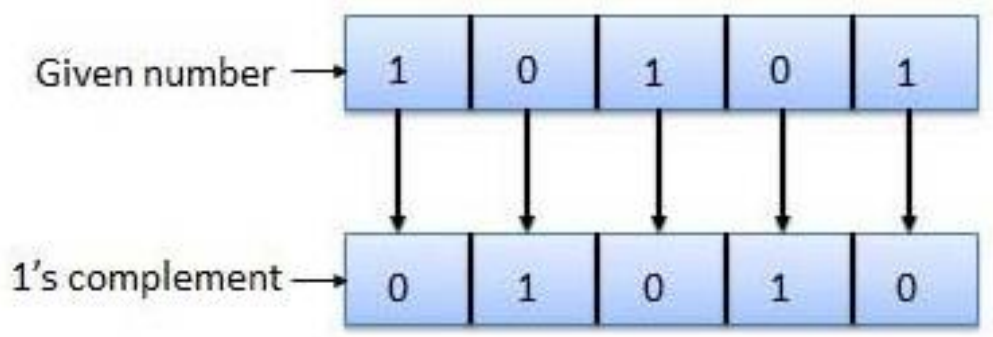
This makes the math a bit confusing.

One's Complement of a Signed Binary Number

One's Complement or **1's Complement** as it is also termed, is another method which we can use to represent negative binary numbers in a signed binary number system. In one's complement, positive numbers (also known as non-complements) remain unchanged as before with the sign-magnitude numbers.

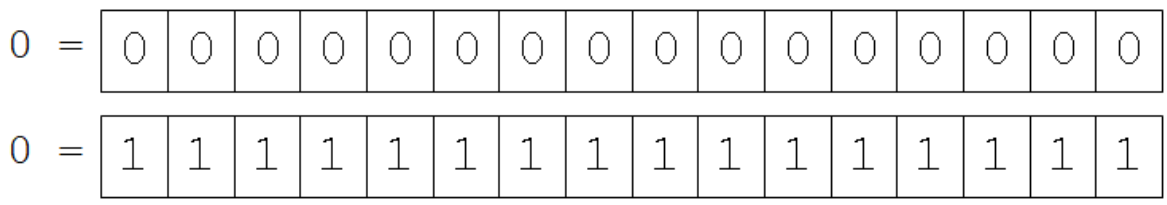
Negative numbers however, are represented by taking the one's complement (inversion, negation) of the unsigned positive number.

1's Complement of 21



Disadvantage:

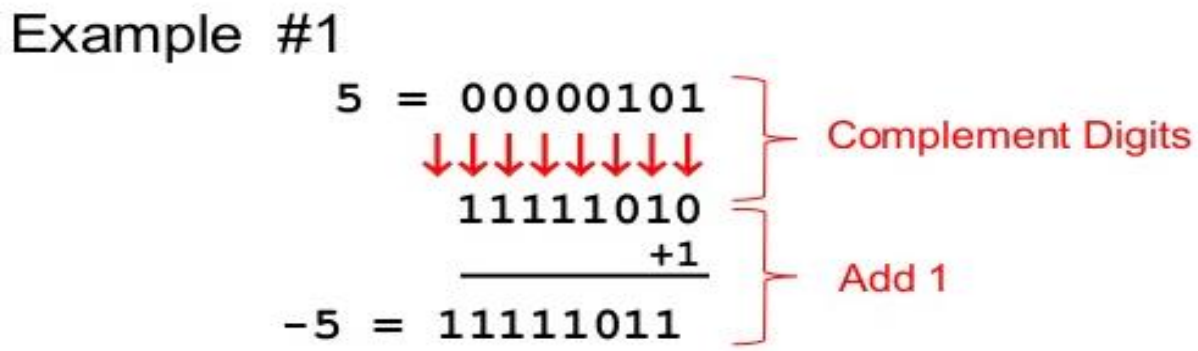
An unfortunate feature of One's Complement representation is that there are two ways of representing the value zero: all bits set to zero, and all bits set to one.



Two's Complement of a Signed Binary Number

Two's Complement or **2's Complement** as it is also termed, is another method like the previous sign-magnitude and one's complement form, which we can use to represent negative binary numbers in a signed binary number system.

2's complement= 1's complement + 1



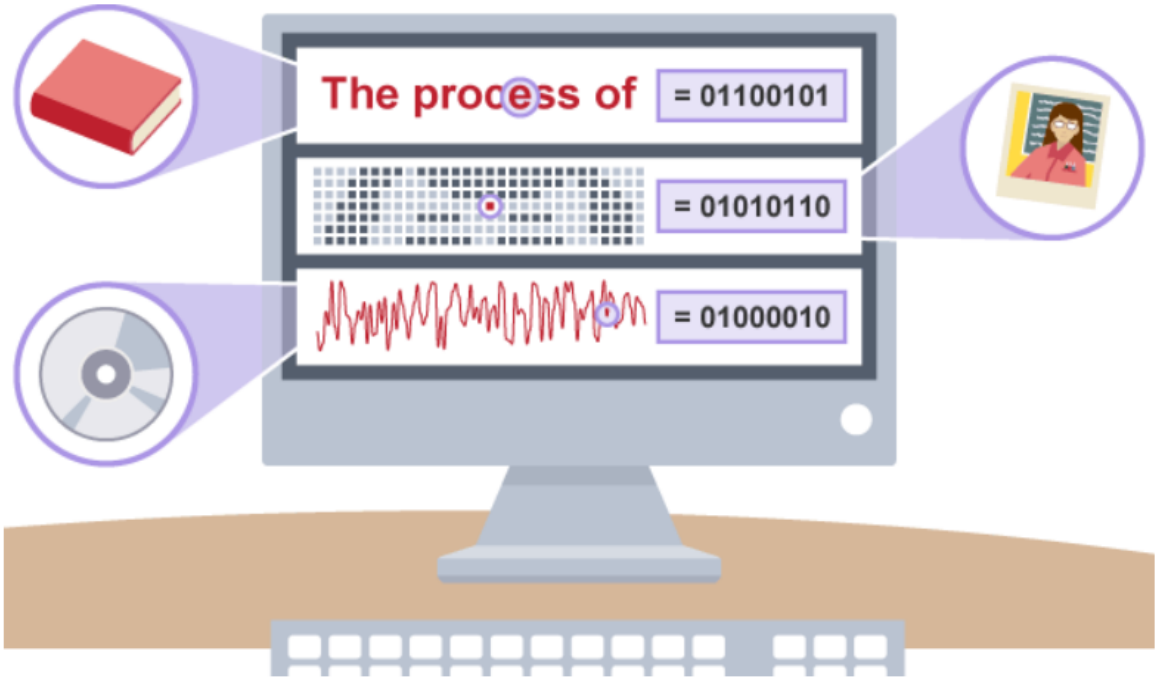
Advantage:

In 2' s complement, there is only one representation for zero – 00000000 (+0) because if we add 1 to 11111111 (-1), we get 00000000 (+0) which is the same as positive zero. This is the reason why 2' s complement is generally used.

Representing text, images and sound:

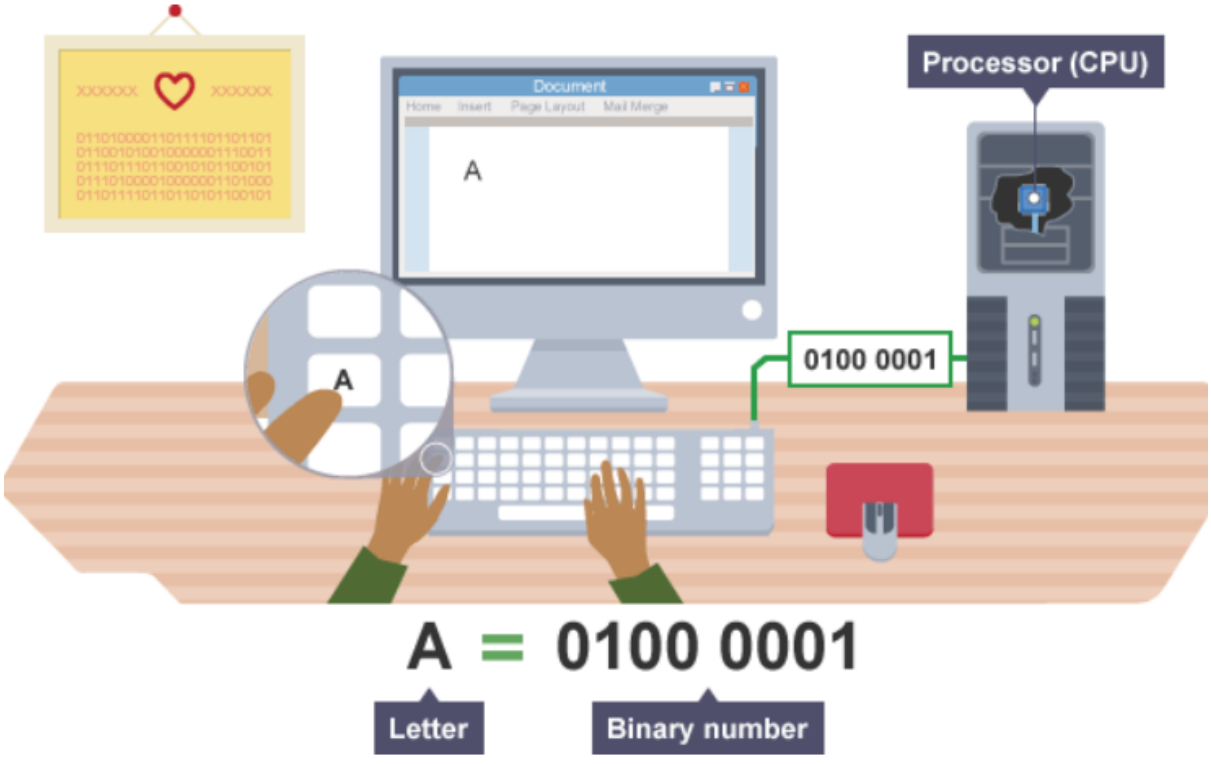
Representing data

All data inside a computer is transmitted as a series of electrical signals that are either **on** or **off**. Therefore, in order for a computer to be able to process any kind of data, including text, images and sound, they must be converted into binary form. If the data is not converted into binary – a series of 1s and 0s – the computer will simply not understand it or be able to process it.



Representing text

When any key on a keyboard is pressed, it needs to be converted into a binary number so that it can be processed by the computer and the typed character can appear on the screen.



A code where each number represents a character can be used to convert text into binary. One code we can use for this is called ASCII. The ASCII code takes each character on the keyboard and assigns it a binary number. For example:

- the letter 'a' has the binary number 0110 0001 (this is the denary number 97)
- the letter 'b' has the binary number 0110 0010 (this is the denary number 98)
- the letter 'c' has the binary number 0110 0011 (this is the denary number 99)

Text characters start at denary number 0 in the ASCII code, but this covers special characters including punctuation, the return key and control characters as well as the number keys, capital letters and lower case letters.

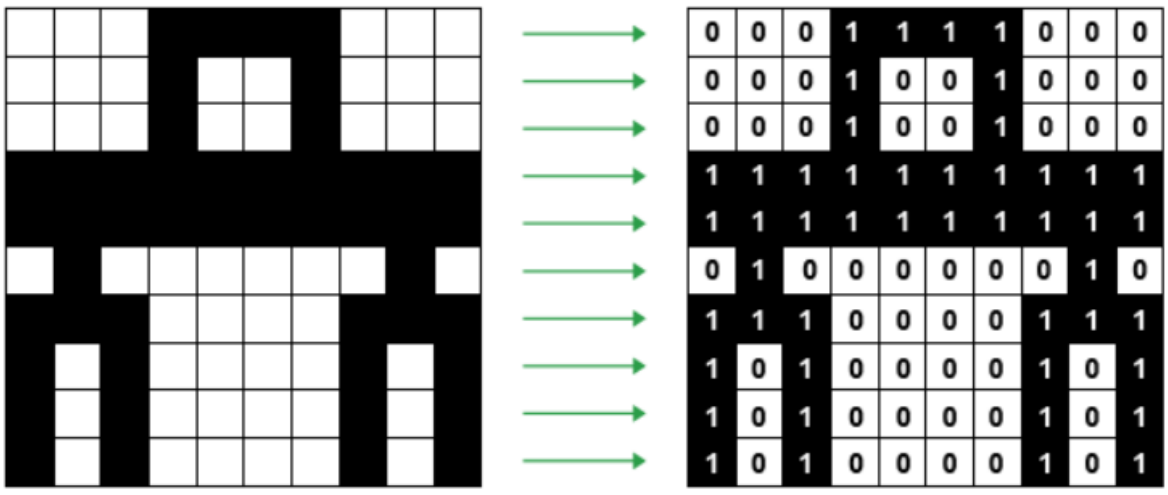
ASCII code can only store 128 characters, which is enough for most words in English but not enough for other languages. If you want to use accents in European languages or larger alphabets such as Cyrillic (the Russian alphabet) and Chinese Mandarin then more characters are needed. Therefore another code, called Unicode, was created. This meant that computers could be used by people using different languages.

Representing images

Images also need to be converted into binary in order for a computer to process them so that they can be seen on our screen. Digital images are made up of pixels. Each pixel in an image is made up of binary numbers.

If we say that 1 is black (or on) and 0 is white (or off), then a simple black and white picture can be created using binary.

To create the picture, a grid can be set out and the squares coloured (1 – black and 0 – white). But before the grid can be created, the size of the grid needs be known. This data is called metadata and computers need metadata to know the size of an image. If the metadata for the image to be created is 10x10, this means the picture will be 10 pixels across and 10 pixels down.This example shows an image created in this way:



Adding colour

The system described so far is fine for black and white images, but most images need to use colours as well. Instead of using just 0 and 1, using four possible numbers will allow an image to use four colours. In binary this can be represented using two bits per pixel:

- 00 – white
- 01 – blue
- 10 – green
- 11 – red

While this is still not a very large range of colours, adding another binary digit will double the number of colours that are available:

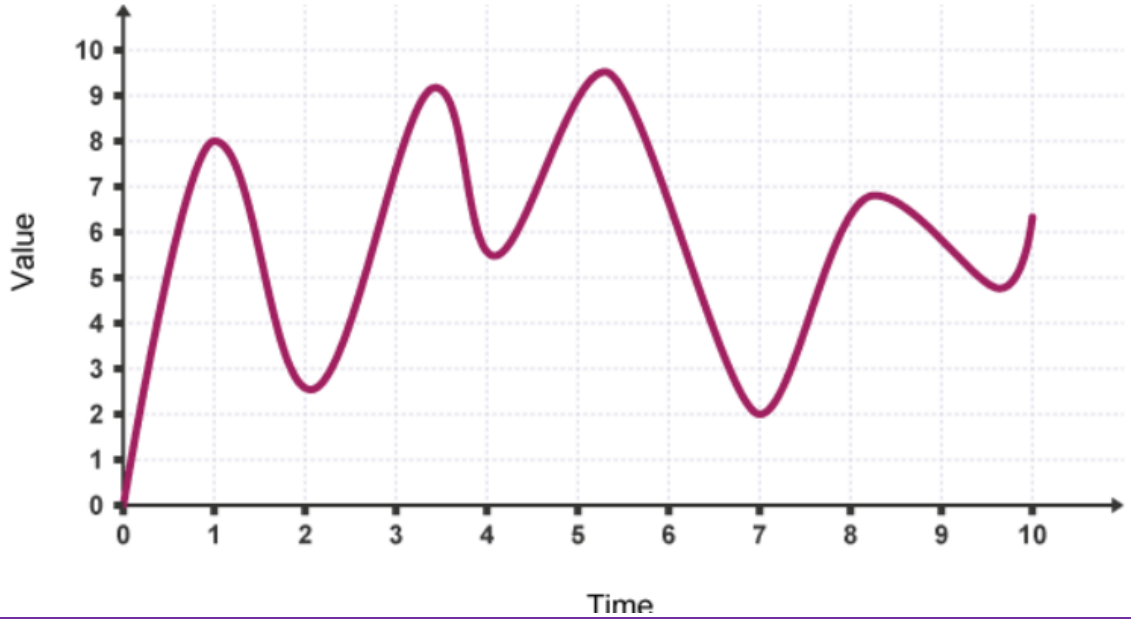
- 1 bit per pixel (0 or 1): two possible colours
- 2 bits per pixel (00 to 11): four possible colours
- 3 bits per pixel (000 to 111): eight possible colours
- 4 bits per pixel (0000 – 1111): 16 possible colours
- …
- 16 bits per pixel (0000 0000 0000 0000 – 1111 1111 1111 1111): over 65 000 possible colours

The number of bits used to store each pixel is called the colour depth. Images with more colours need more pixels to store each available colour. This means that images that use lots of colours are stored in larger files.

Representing sound

Sound needs to be converted into binary for computers to be able to process it. To do this, sound is captured - usually by a microphone - and then converted into a digital signal.

An analogue to digital converter will sample a sound wave at regular time intervals. For example, a sound wave like this can be sampled at each time sample point:

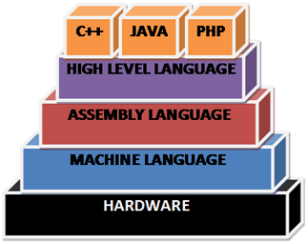


The samples can then be converted to binary. They will be recorded to the nearest whole number.

Time sample	1	2	3	4	5	6	7	8	9	10
Denary	8	3	7	6	9	7	2	6	6	6
Binary	1000	0011	0111	0110	1001	0111	0010	0100	0110	0110

Language Hierarchy:

- There are generally three types of computer languages. They are:
  1. Machine Language,
  2. Assembly Language and
  3. High-level languages.



Machine Language: (1940's)

- It is the lowest-level programming language.
- The fundamental/native language of the computer's processor, also called Low Level Language i.e., each computer has its own machine language.
- All programs are converted into machine language before they can be executed.

- It consists of combination of 0's and 1's that represent high and low electrical voltage.
- Machine language is a sequence of instructions written in the form of binary number to which computer responds directly.
- It is considered as first generation language.
- A machine language instruction generally has three parts as shown below.

Operation Code	Mode	Operand

Part1 : command or operation code, it conveys the computer what function has to be performed by the instruction.

Part2 : it either specifies the operand contains data on which the operation has to be performed or it specifies that the operand contains a location, the contents of which have to be subject to the operation.

Table 9.1 Code in machine language to add two integers

Hexadecimal	Code in machine language			
$(1FEF)_{16}$	0001	1111	1110	1111
$(240F)_{16}$	0010	0100	0000	1111
$(1FEF)_{16}$	0001	1111	1110	1111
$(241F)_{16}$	0010	0100	0001	1111
$(1040)_{16}$	0001	0000	0100	0000
$(1141)_{16}$	0001	0001	0100	0001
$(3201)_{16}$	0011	0010	0000	0001
$(2422)_{16}$	0010	0100	0010	0010
$(1F42)_{16}$	0001	1111	0100	0010
$(2FFF)_{16}$	0010	1111	1111	1111
$(0000)_{16}$	0000	0000	0000	0000

Advantages: Though it is not a human friendly language, it offers following advantages.

1. **Translation Free:** Translation is not required as the CPU directly understands machine instructions.
2. As conversion is not needed, programs developed using these languages takes less execution time.
3. More control on hardware.

Disadvantages:

1. Difficult to use: It is difficult to understand and develop a program using machine language as it is very complex to write using binary numbers.
2. Machine dependent: Computer architectures differ from one another. The programmer has to remember machine characteristics while preparing a program.
3. Error Prone: since the programmer has to remember all the opcodes and the memory locations, machine language is bound to be error prone.
4. Difficult to debug and modify.

Assembly Language (1950's):

- The next evolution in programming came with the idea of replacing binary code for instruction and addresses with symbols or mnemonics.
- Because they used symbols, these languages were first known as symbolic languages. The set of these mnemonic languages were later referred to as assembly languages.
- This language is also referred as low-level language.
- Every computer has its own assembly language that is dependent upon the internal architecture of the processor.
- An assembler is a translator that takes input in the form of the assembly language program and produces machine language code as its input.

Table 9.2 Code in assembly language to add two integers

Code in assembly language				*Description*
LOAD	RF	Keyboard		Load from keyboard controller to register F
STORE	Number1	RF		Store register F into Number1
LOAD	RF	Keyboard		Load from keyboard controller to register F
STORE	Number2	RF		Store register F into Number2
LOAD	R0	Number1		Load Number1 into register 0
LOAD	R1	Number2		Load Number2 into register 1
ADDI	R2	R0	R1	Add registers 0 and 1 with result in register 2
STORE	Result	R2		Store register 2 into Result
LOAD	RF	Result		Load Result into register F
STORE	Monitor	RF		Store register F into monitor controller
HALT				Stop

Assembler

- An assembler is a translator that takes input in the form of the assembly language program and produces machine language code as its input.

- An assembler is a system software which converts an assembly language program to its equivalent object code.
- The input to the assembler is a source code written in assembly language and the output is an object code.
- Basic assembler functions are
  - Translating mnemonic language to its equivalent object code.
  - Assigning machine address to symbolic labels.

Advantages:

- It is easy to write and maintain programs in assembly language than in machine languages

- Improved readability than machine language.
- Less Error prone when compared with machine language.
- Assembly programs can run can run much faster and use less memory and other resources.
- More control on hardware.

Disadvantages:

- Machine dependent: It is specific to particular machine architecture.
- Hard to learn
- Assembly language program is less efficient than machine language program.
- Programming is difficult and time consuming.

High-Level Languages:

- Although assembly languages greatly improved programming efficiency, they still required programmers to concentrate on the hardware they were using. Working with symbolic languages was also very tedious, because each machine instruction had to be individually coded. The desire to improve programmer efficiency and to change the focus from the computer to the problem being solved led to the development of high-level languages.
- These languages have instructions that are similar to human languages and have a set grammar that makes it easy for a programmer to write programs and identify correct errors in them.

Advantages:

- Readability : programs written in these languages are more readable than those written in assembly and machine languages.
- Portability: High level programming languages can be run on different machines with little or no change.
- Easy debugging: Errors can be easily detected and removed.
- Development of software is easy.

Disadvantages:

- Poor control on hardware
- Less efficient

Compiler vs interpreter.

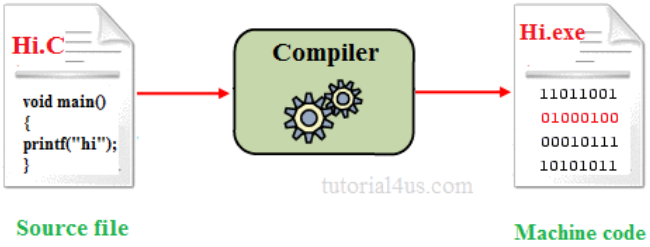
Translation

Programs today are normally written in one of the high-level languages. To run the program on a computer, the program needs to be translated into the machine language of the computer on which it will run.

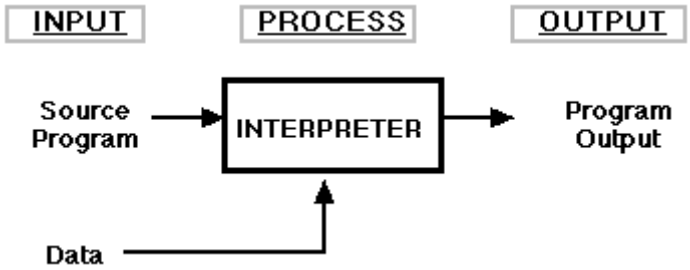
- The program in a high-level language is called the source program.
- The translated program in machine language is called the object program.

Two methods are used for translation: compilation and interpretation.

Compiler: A compiler is a translator which converts the program written in high-level language into assembly code or into another form of intermediate code or directly into machine code. A compiler converts the whole source code at once into object code and then executes it. So, compiler is faster than a interpreter.



Interpreter: An interpreter is a translator which converts the source program into object or machine code. The interpreter converts the source code line-by-line and executes it immediately, which results in less performance. Thus, an interpreter is slower than a compiler.



Difference between compiler and interpreter

Comparison	Compiler	Interpreter
Input	It takes an entire program at a time.	It takes a single line of code
Speed	Comparatively fast	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	Displays all errors after compilation ,all at the same time	Displays error of each line one by one

Error Detection	Difficuilt	Easy
Examples	C,C++,Java,Scala,etc.,	Python,Perl,PHP,Ruby,etc.,

Command Line Interface

The command line is a text interface for your computer. It's a program that takes in commands, which it passes on to the computer's operating system to run.

From the command line, you can navigate through files and folders on your computer, just as you would with Windows Explorer on Windows or Finder on Mac OS. The difference is that the command line is fully text-based.

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.

Basic Windows CLI Commands

Command	Description
dir	List the directory (folder) system.
cd *pathname*	Change directory (folder) in the file system.
cd \	Move to the root folder of the file system.
cd ..	Move one level up (one folder) in the file system.
copy	Copy a file to another folder.
move	Move a file to another folder.
type *filename*	Type a file.
mkdir or md	Creates a new directory (folder).
rmdir or rd	Removes a directory (folder).
cls	Clears the CLI window.
exit	Closes the CLI window.
help *command*	Shows the manual for a given command.

Linux Commands

1. **pwd –** When you first open the terminal, you are in the home directory of your user. To know which directory you are in, you can use the "pwd" command. It gives us the absolute path, which means the path that starts from the root. The root is the base of the Linux file system. It is denoted by a forward slash(/). The user directory is usually something like "/home/username".

```
nayso@Alok-Aspire:~$ pwd
/home/nayso
```

2. **ls –** Use the "ls" command to know what files are in the directory you are in. You can see all the hidden files by using the command "ls -a".

```
nayso@Alok-Aspire:~$ ls
Desktop           itsuserguide.desktop  reset-settings     VCD_Copy
Documents         Music                 School_Resources   Videos
Downloads         Pictures              Students_Works_10
examples.desktop  Public                Templates
GplatesProject    Qgis Projects         TuxPaint-Pictures
```

3. **cd –** Use the "cd" command to go to a directory. For example, if you are in the home folder, and you want to go to the downloads folder, then you can type in "cd Downloads". Remember, this command is case sensitive, and you have to type in the name of the folder exactly as it is.

```
nayso@Alok-Aspire:~$ cd Downloads
nayso@Alok-Aspire:~/Downloads$ cd
```

4. **mkdir&rmdir –** Use the mkdir command when you need to create a folder or a directory. For example, if you want to make a directory called "DIY", then you can type "mkdir DIY". Remember, as told before, if you want to create a directory named "DIY Hacking", then you can type "mkdir DIY\ Hacking". Use rmdir to delete a directory. But rmdir can only be used to delete an empty directory. To delete a directory containing files, use rm.

```
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$ mkdir DIY
nayso@Alok-Aspire:~/Desktop$ ls
DIY
nayso@Alok-Aspire:~/Desktop$ rmdir DIY
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$
```

5. **rm -** Use the rm command to delete files and directories. Use "rm -r" to delete just the directory. It deletes both the folder and the files it contains when using only the rm command.

```
nayso@Alok-Aspire:~/Desktop$ ls
newer.py  New Folder
nayso@Alok-Aspire:~/Desktop$ rm newer.py
nayso@Alok-Aspire:~/Desktop$ ls
New Folder
nayso@Alok-Aspire:~/Desktop$ rm -r New\ Folder
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$
```

6. **man**& --help – To know more about a command and how to use it, use the man command. It shows the manual pages of the command. For example, "man cd" shows the manual pages of the cd command. Typing in the command name and the argument helps it show which ways the command can be used (e.g., cd –help).

```
TOUCH(1)                        User Commands                        TOUCH(1)

NAME
       touch - change file timestamps

SYNOPSIS
       touch [OPTION]... FILE...

DESCRIPTION
       Update  the  access  and modification times of each FILE to the current
       time.

       A FILE argument that does not exist is created empty, unless -c  or  -h
       is supplied.

       A  FILE  argument  string of - is handled specially and causes touch to
       change the times of the file associated with standard output.

       Mandatory arguments to long options are  mandatory  for  short  options
       too.

       -a     change only the access time

Manual page touch(1) line 1 (press h for help or q to quit)
```

7. **cp** – Use the cp command to copy files through the command line. It takes two arguments: The first is the location of the file to be copied, the second is where to copy.

```
nayso@Alok-Aspire:~/Desktop$ ls /home/nayso/Music/
nayso@Alok-Aspire:~/Desktop$ cp new.txt /home/nayso/Music/
nayso@Alok-Aspire:~/Desktop$ ls /home/nayso/Music/
new.txt
```

8. **mv** – Use the mv command to move files through the command line. We can also use the mv command to rename a file. For example, if we want to rename the file "text" to "new", we can use "mv text new". It takes the two arguments, just like the cp command.

```
nayso@Alok-Aspire:~/Desktop$ ls
new.txt
nayso@Alok-Aspire:~/Desktop$ mv new.txt newer.txt
nayso@Alok-Aspire:~/Desktop$ ls
newer.txt
```