UNIT 1: INTRODUCTION TO COMPUTER SYSTEMS AND BASICS PROGRAMMING FUNDAMENTALS

Syllabus:

- Introduction to the computer system, ALU, registers, and Memory
- Concept of finite storage, bits, bytes, kilo, mega and gigabytes
- The idea of program execution at the micro level
- Introduction to system software: Operating System, Compilers, Assemblers, Interpreter and multi-user environment.
- Concept of flow charts and algorithms, Algorithms to program.
- Logic development for solving problems
- Development of flow charts and algorithms

Computer system: A computer system can be defined as a digital electronic machine that can be programmed to perform some operations as per the instructions.

OR

A computer system can also be defined as a digital electronic device that can be programmed to accept some inputs in term of data, then process this data as per the program instructions and provides the output in the desired format that can be used for some meaningful work.

OR

The Computer is a programmable finite-state machine which can perform precise arithmetic and logic operations. (A programmable finite-state machine is which can take only a fixed range of values)

The computer system consists of both hardware and software components.

- Hardware Components: They are the physical components mounted within the computer case and some are also connected externally.
- Software Components: A collection of related programs is referred to as Software.

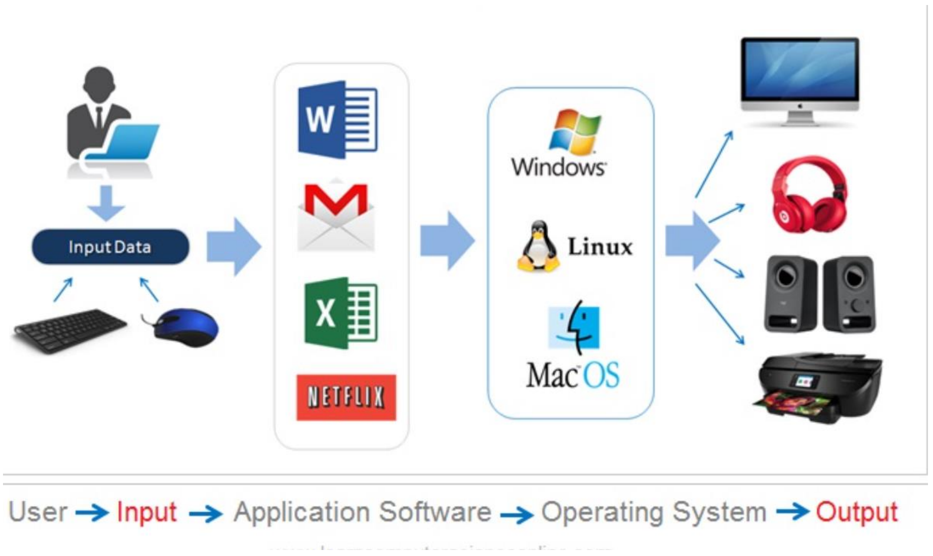

Input Data
Windows
Linux
Mac OS
NETFLIX
User → Input → Application Software → Operating System → Output

Figure 1. Introduction to Computer System

As shown in Figure 1, the computer users interact with the application software and provide the input data. The application software in turn interacts with the operating system to process the data and the processed data is sent to the output device.

The Computer Systems can be categorized into 5 main blocks:

- Input Unit
- Output Unit
- Arithmetic Logic Unit (ALU)
- Control unit
- Memory

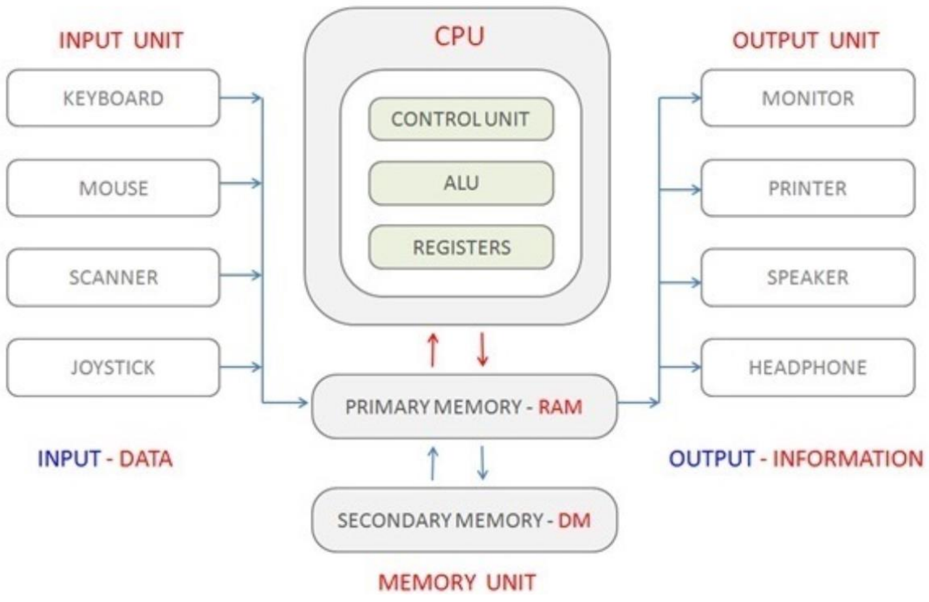


Figure 2. Computer System Block Diagram

1) **Input Unit:** It is the device used to enter data and instructions into the computer. The data read from the input device is stored in the computer's memory. Examples: Keyboard, mouse, scanner, and light pen
2) **Output Unit:** The information generated by the computer is displayed using an output device. Examples: Screen, printer, speaker etc.
3) **Arithmetic Logic Unit (ALU):** An arithmetic-logic unit is the part of a central processing unit that carries out arithmetic and logic operations on the operands in computer instruction words.
   - Typically, the ALU has direct input and output access to the processor controller, main memory (random access memory or RAM in a personal computer) and input/output devices. Inputs and outputs flow along an electronic path that is called a bus.
   - The input consists of an instruction word, sometimes called a machine instruction word, that contains an operation code or "opcode," one or more operands and sometimes a format code. The operation code tells the ALU what operation to perform and the operands are used in the operation
4) **Control unit**: Also called Microprocessor.
   - The Arithmetic Logic Unit (ALU) and the control unit together comprise the Central Processing Unit (CPU).

- It provides processing power to the computer system.
- It performs a number of vital operations where one of the main functions is to execute the computer programs.
- It is responsible to decode the machine instructions.
- Some popular microprocessors are Intel, Pentium, Pentium Pro etc.

5) **Memory:** It is used to store data, instructions and information. It is actually a work area of a computer, where the CPU stores the data and instructions.

The computer's memory is constructed out of semiconducting materials and stores information in binary form (The binary form is composed of two symbols 0 and 1, called binary digits or bits).

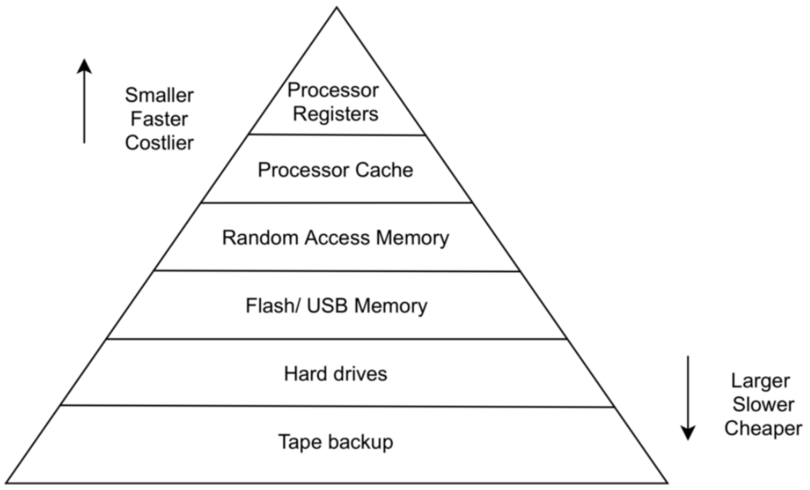


Figure 3. Memory Hierarchy

The memory can be divided into distinct parts as below:

1. **Registers:** Registers are the locations within the microprocessor where the data is stored temporarily during processing. They are the smallest data-holding elements that are built into the processor itself. These are the memory locations that are directly accessible by the processor. It may hold an instruction, a storage address or any kind of data such as a bit sequence or individual characters.
2. **Processor Cache:** Processor cache is a small onboard memory used by the processor to store data. It is a small amount of high-speed memory that stores copies of the most recent data from the RAM. So, when you want to open a program, your computer looks in the cache first and if it finds what it needs, it retrieves it from there. Processor cache is directly related to how fast your laptop can run programs. The more cache, the faster the system will be. Laptop processors are measured in GHz. A higher GHz means a faster CPU.
    a. Internal Cache: It contains frequently used data. The use of a cache avoids repeated reading of data from the slower main memory.
    b. External Cache: It is used to supplement the internal cache. It is used when the internal cache is not present. It is placed between the CPU and the main memory.
3. **Main/Primary Memory:** The main memory is the fundamental storage unit in a computer system.

- It is associatively large and quick memory and saves programs and information during computer operations.
- Main memory is fast but expensive. However, it is volatile which means the content stored will be lost when the power supply is cut off.
- The operating system controls the usage of this memory.
- Specifications such as 4GB, 8GB, 12GB and 16GB almost always refer to the capacity of RAM. In contrast, disk or solid-state storage capacities in a computer are typically 128GB or 256GB and higher.

Primary memory is further divided into two parts:

3.1. RAM (Random Access Memory): It is one of the faster types of main memory accessed directly by the CPU. It is the hardware in a computer device to temporarily store data, programs or program results. It is used to read/write data in memory until the machine is working. It is volatile, which means if a power failure occurs or the computer is turned off, the information stored in RAM will be lost. All data stored in computer memory can be read or accessed randomly at any time.

Advantages of RAM

- It is a faster type of memory in a computer.
- It requires less power to operate.
- Program loads much faster
- More RAM increases the performance of a system and can multitask.
- Perform read and write operations.
- The processor can read information faster than a hard disc, floppy, USB, etc.

Disadvantages of RAM

- Less RAM reduces the speed and performance of a computer.
- Due to volatile property, it requires electricity to preserve the data.
- It is more expensive than ROM
- It is unreliable as compared to ROM
- The Size of RAM is limited.

3.2.ROM (Read Only Memory): ROM is a memory device or storage medium that is used to permanently store information inside a chip. It is a read-only memory that can only read stored information, data or programs, but we cannot write or modify anything. A ROM contains some important instructions or program data that are required to start or boot a computer. It is a non-volatile memory; it means that the stored information cannot be lost even when the power is turned off or the system is shut down.

Advantages of ROM

- It is a non-volatile memory in which stored information can be lost even power is turned off.
- It is static, so it does not require refreshing the content every time.
- Data can be stored permanently.
- It is easy to test and store large data as compared to RAM.
- These cannot be changed accidently
- It is cheaper than RAM.

- It is simple and reliable as compared to RAM.
- It helps to start the computer and loads the OS.

Disadvantages of ROM

- Store data cannot be updated or modify except to read the existing data.
- It is a slower memory than RAM to access the stored data.
- It takes around 40 minutes to destroy the existing data using the high charge of ultraviolet light.

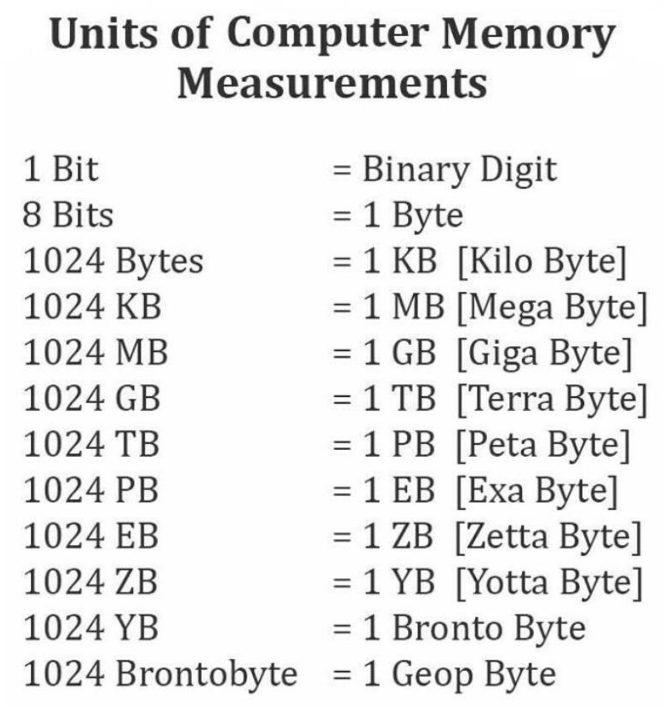
RAM Vs. ROM

RAM	ROM
It is a Random-Access Memory.	It is a Read Only Memory.
Read and write operations can be performed.	Only Read operation can be performed.
Data can be lost in volatile memory when the power supply is turned off.	Data cannot be lost in non-volatile memory when the power supply is turned off.
It is a faster and expensive memory.	It is a slower and less expensive memory.
Storage data requires to be refreshed in RAM.	Storage data does not need to be refreshed in ROM.
The size of the chip is bigger than the ROM chip to store the data.	The size of the chip is smaller than the RAM chip to store the same amount of data.
Types of RAM: DRAM and SRAM	Types of ROM: MROM, PROM, EPROM, EEPROM

4. **Secondary Memory:** Secondary Memory is used to store the data which is not currently required by the microprocessor.
   - It is slower but less expensive than the main memory.
   - It is also not volatile and larger in size.
   - The microprocessor cannot access the main memory directly. Therefore, data from the secondary memory has to be brought to the main memory so that the processor can use it. Memory management techniques are required to transfer information between the main memory and secondary memory and this function are performed by the operating system.
   - Examples of Secondary memory are hard disks, floppy disks, CD-ROMS etc.

<u>Concept of finite storage, bits, bytes, kilo, mega and gigabytes:</u>

Memory units can be defined as a way to measure the quantity of data collected together under a single memory location, which is termed the ‘storage unit’. These memory units are used to indicate the number or amount of data that are picked and used for individual computation processes performed in a computing device.



Units of Computer Memory Measurements

1 Bit	= Binary Digit
8 Bits	= 1 Byte
1024 Bytes	= 1 KB [Kilo Byte]
1024 KB	= 1 MB [Mega Byte]
1024 MB	= 1 GB [Giga Byte]
1024 GB	= 1 TB [Terra Byte]
1024 TB	= 1 PB [Peta Byte]
1024 PB	= 1 EB [Exa Byte]
1024 EB	= 1 ZB [Zetta Byte]
1024 ZB	= 1 YB [Yotta Byte]
1024 YB	= 1 Bronto Byte
1024 Brontobyte	= 1 Geop Byte

The memory units of the data in the storage systems are classified into the below different types,

1. **Bit:** The bit is a term shortly used instead of the word ‘Binary digit’, which is nothing but the 0’s and 1’s that are used for the indication of passive or active states of any component involved in an electric system. This is the least of the memory units that are used to represent the storage occupancy of the data in the memory space, that is, the data shown either in 0 or in 1.
2. **Nibble:** Nibble can be defined as the collection of four bits, which is characterized as a hexadecimal number to store the data in the memory. It takes up the same space as a hexadecimal number or a group of four bits of data. It is sometimes called as ‘Nybble’ or ‘Nyble’ or ‘hex digit’. There are also memory units called as ‘low nibble’ and ‘high nibble’ that are used to denote the contents of the nibble memory unit, where the low nibble shows the lesser momentous bit and the high nibble shows the highly momentous bit inside the memory unit.
3. **Byte:** A byte can be defined as a group of eight binary digits, which generally corresponds to the other compounds of 8 bits. This is the commonly used unit terminology, as it comes with the prefixes for the multiples of bytes, such as the prefixes as kilo-, mega-, giga-, tera-, peta-, etc.
4. **Kilobyte:** Kilobyte can be defined as one of the many multiples of the byte memory units. The prefix kilo-, normally, represents the unit to be ‘Kilo’ and it comes in multiples of ‘1000’. Though in terms of international standards kilo means 1000, in a typical memory unit, the binary digit as seen as Base 2 and so the ‘Kilo’ here means 1024 of binary digits or bits. In short, 1Kilobyte or 1KB is equal to 1024 bits or binary digits.
5. **Megabyte:** Similar to the Kilobyte memory unit, the megabyte memory unit is used when the bits are large in number and when it makes it easy to say it the number to

be in lesser length. That is, in international standards the prefix ‘mega- ‘is used for showing the 10^6 or the 1000000. This also means that a typical MB is shown with Base 2, and so the 1MB or the 1 Megabyte equals to the 1048576 bytes of the memory unit. It can also be calculated as (1024) ^2bytes or the 2^20 bytes.

6. **Gigabyte:** The Gigabytes are used to represent the data to be in multiples of 10^9 or 1000000000 binary digits, and when the memory unit is in terms of Base 2 then the storage space is calculated as 2^30 bytes or as 1024^3 bytes. In other words, the memory units used by the processor that runs on the Base 2 are termed as the Gigabytes and are equal to 1073741824 bytes.
7. **Terabyte:** Terabytes are applied when the memory requirement is as high as 2^40, that is the binary digits exponential to the number 40. It can also be seen as equal to 1024 terabytes and is represented as TB. Or other words equal to 10^12 bytes by international standards and that can also be termed as the trillion bytes.

<u>Introduction to system software: Operating System, Compilers, Assemblers, Interpreter and multi-user environment</u>

There are two main types of software: systems software and application software.

Application Software: The term “application software” refers to software that performs specific functions for a user. When a user interacts directly with a piece of software, it is called application software. The sole purpose of application software is to assist the user in doing specified tasks. Microsoft Word and Excel, as well as popular web browsers like Firefox and Google Chrome, are examples of application software. It also encompasses the category of mobile apps, which includes apps like WhatsApp for communication and games like Candy Crush Saga. There are also app versions of popular services, such as weather or transportation information, as well as apps that allow users to connect with businesses. Global Positioning System (GPS), Graphics, multimedia, presentation software, desktop publishing software, and so on are examples of such software.

Need of application software:

– Helps the user in completing specified tasks
– Manages and manipulates data
– Allows users to effectively organize information

System Software: System software is software that provides a platform for other software. Some examples can be operating systems, assemblers, interpreters, compilers, linkers and loaders. It includes the programs that are dedicated to managing the computer itself. This software consists of programs written in low-level languages, used to interact with the hardware at a very basic level.

The most important features of system software include:

– Closeness to the system
– Fast speed
– Difficult to manipulate
– Written in a low-level language
– Difficult to design

Operating System: An operating system (OS) is a type of system software that manages a computer’s hardware and software resources. It acts as a link between the software and the hardware. It controls and keeps a record of the execution of all other programs that are present in the computer, including application programs and other system software.

The most important tasks performed by the operating system are:

- Memory Management: The OS keeps track of the primary memory and allocates the memory when a process requests it.
- Processor Management: Allocates the main memory (RAM) to a process and de-allocates it when it is no longer required.
- File Management: Allocates and de-allocates the resources and decides who gets the resources.
- Security: Prevents unauthorized access to programs and data by means of passwords.
- Error-detecting Aids: Production of dumps, traces, error messages, and other debugging and error-detecting methods.
- Scheduling: The OS schedules the process through its scheduling algorithms.

Compiler and Interpreters:
A typical computer program usually exists in high-level languages that a human can understand. Thus, they contain various phrases and words from the English language (or any other in common use). Computers, on the other hand, cannot understand these languages as we do- but can comprehend a program in binary codes. As a result, we first write a program in a high-level language (source code), convert them into machine language, and make them readable for the computers. It is exactly when we need interpreters and compilers.

Compiler: A compiler is a computer program that reads a program written in a high-level language such as FORTRAN, PL/I, COBOL, etc. It can translate it into the same program in a low-level language including machine language. The compiler also finds out the various errors encountered during the compilation of a program.
The compiler converts high-level language into low-level language using various phases. A character stream inputted by the customer goes through multiple stages of compilation which at last will provide the target language.

Advantages of Compiler

- There are various advantages of the compiler which are as follows −
- A compiler translates a program in a single run.
- It consumes less time.
- CPU utilization is more.
- Both syntactic and semantic errors can be checked concurrently.
- It is easily supported by many high-level languages like C, C++, JAVA, etc.

Interpreter: An interpreter is a program that executes the programming code directly instead of just translating it into another format. It translates and executes programming language statements one by one.
Programming languages that use interpreters include Python, Ruby, and JavaScript, while programming languages that use compilers include Java, C++, and C.
In most cases, a compiler is preferable since its output runs much faster compared to a line-by-line interpretation. Rather than scanning the whole program and translating it into machine code like a compiler does, the interpreter translates code one statement at a time.

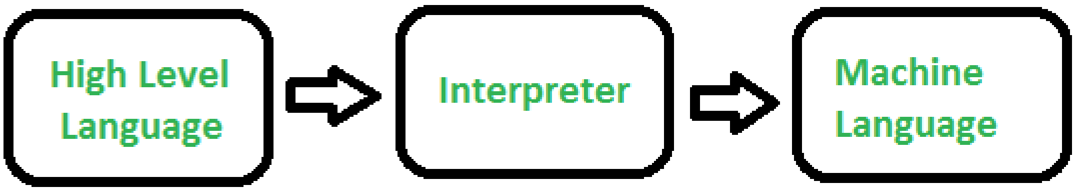
While the time to analyze source code is reduced, especially a particularly large one, execution time for an interpreter is comparatively slower than a compiler. On top of that, since interpretation happens per line or statement, it can be stopped in the middle of execution to allow for either code modification or debugging.

Compilers must generate intermediate object code that requires more memory to be linked, contrarily to interpreters which tend to use memory more efficiently.

Since an interpreter reads and then executes code in a single process, it very useful for scripting and other small programs. As such, it is commonly installed on Web servers, which run a lot of executable scripts. It is also used during the development stage of a program to test small chunks of code one by one rather than having to compile the whole program every time.

Every source statement will be executed line by line during execution, which is particularly appreciated for debugging reasons to immediately recognize errors. Interpreters are also used for educational purposes since they can be used to show students how to program one script at a time.

The software by which the conversion of the high level instructions is performed line-by-line to machine level language, other than compiler and assembler, is known as **INTERPRETER**. If an error is found on any line, the execution stops till it is corrected. This process of correcting errors is easier as it gives line-by-line error but the program takes more time to execute successfully. Interpreters were first used in 1952 to ease programming within the limitations of computers at the time. It translates source code into some efficient intermediate representation and immediately execute



this.
Source programs are compiled ahead of time and stored as machine independent code, which is then linked at run-time and executed by an interpreter. An Interpreter is generally used in micro computer. It helps the programmer to find out the errors and to correct them before control moves to the next statement. Interpreter system performs the actions described by the high level program. For interpreted programs, the source code is needed to run the program every time. Interpreted programs run slower than the compiled programs. **Self-Interpreter** is a programming language interpreter which is written in a language which can interpret itself. For Example- **BASIC** interpreter written in **BASIC**. They are related to self-hosting compilers. Some languages have an elegant and self-interpreter such as Lisp and Prolog. **Need of an Interpreter :** The first and vital need of an interpreter is to translate source code from high-level language to machine language. However, for this purpose Compiler is also there to satisfy this condition. The compiler is a very powerful tool for developing programs in high-level language. However, there are several demerits associated with the compiler. If the source code is huge in size, then it might take hours to compile the source code, which will significantly increase the compilation duration. Here, Interpreter plays its role. They can cut this huge compilation duration. They are designed to translate single instruction at a time and execute them immediately. **Advantage and Disadvantage of Interpreter :**

- **Advantage of interpreter** is that it is executed line by line which helps users to find errors easily. Therefore, Error localization is easier.
  - **Disadvantage of interpreter** is that it takes more time to execute successfully than compiler.

Applications of Interpreters :

- Each operator executed in a command language is usually an invocation of a complex routine, such as an editor or compiler so they are frequently used to command languages and glue languages.
- Virtualization is often used when the intended architecture is unavailable.
- Sand-boxing
- Self-modifying code can be easily implemented in an interpreted language.
- Emulator for running Computer software written for obsolete and unavailable hardware on more modern equipment.

Some examples of programming languages that use interpreters are Python, Ruby, Perl, PHP and Matlab.

Comparison between the Compiler and the Interpreter

Compiler	Interpreter
A compiler translates the entire source code in a single run.	An interpreter translates the entire source code line by line.
It consumes less time i.e.; it is faster than an interpreter.	It consumes much more time than the compiler i.e., it is slower than the compiler.
It is more efficient.	It is less efficient.
CPU utilization is more.	CPU utilization is less as compared to the compiler.
Both syntactic and semantic errors can be checked simultaneously.	Only syntactic errors are checked.
The compiler is larger.	Interpreters are often smaller than compilers.
It is not flexible.	It is flexible.
The localization of errors is difficult.	The localization of error is easier than the compiler.
A presence of an error can causes the whole program to be re-organized.	A presence of an error causes only a part of the program to be reorganised.
The compiler is used by language such as C, and C++.	An interpreter is used in languages such as Java.

Assembler: An assembler is a program that converts the assembly language into machine code. It takes the basic commands and operations and converts them into binary code specific to a type of processor. Assemblers produce executable code that is similar to compilers. However, assemblers are more simplistic since they only convert low-level code (assembly language) to machine code. Since each assembly language is designed for a specific processor, assembling a program is performed using a simple one-to-one mapping from assembly code to machine code. On the other hand, compilers must convert generic high-level source code into machine code for a specific processor.

An assembly language is **a type of low-level programming language that is intended to communicate directly with a computer's hardware**. Unlike machine language, which consists of binary and hexadecimal characters, assembly languages are designed to be readable by humans.

Example of assembly language

In this example of an assembly language, "1:" is the label which lets the computer know where to begin the operation. The "MOV" is the mnemonic command to move the number "3" into a part of the computer processor, which can function as a variable. "EAX," "EBX" and "ECX" are the variables. The first line of code loads "3" into the register "eax." The second line of code loads "4" into the register "ebx." Finally, the last line of code adds "eax" and "ebx" and stores the result of the addition, which is seven, in "ecx."

Altogether, this assembly language looks like this:

```
1: MOV eax, 3
MOV ebx, 4
ADD eax, ebx, ecx
```

Comparison between Compiler, Interpreter and Assembler

PARAMETERS	**COMPILER**	**INTERPRETER**	**ASSEMBLER**
Conversion	It converts the high-defined programming language into Machine language or binary code.	It also converts the program-developed code into machine language or binary code.	It converts programs written in the assembly language to machine language or binary code.
Scanning	It scans the entire program before converting it into binary code.	It translates the program line by line to the equivalent machine code.	It converts the source code into the object code and then converts it into the machine code.
Error Detection	Gives the full error report after the whole scan.	Detects error line by line. And stops scanning until the error in the previous line is solved.	It detects errors in the first phase after fixation the second phase starts.

Code generation	Intermediate code generation is done in the case of the Compiler.	There is no intermediate code generation.	There is an intermediate object code generation.
Execution time	It takes less execution time compared to an interpreter.	An interpreter takes more execution time than the compiler.	It takes more time than the compiler.
Examples	C, C#, C++	Python, Perl, VB, PostScript, LISP, etc…	GAS or GNU Assembler

<u>Multi-User Environment:</u>
A multiuser environment is one in which other users can connect and make changes to the same database that you are working with. As a result, several users might be working with the same database objects at the same time.

<u>Concept of flow charts and algorithms, Algorithms to program</u>

Algorithm Definition: An algorithm is a well-defined, finite set of computational instructions that accomplishes a particular task, which may or may not take inputs and produce some values or a set of values as output. For example, the internet runs on algorithms and all online searching is accomplished through them.
In addition, they must satisfy the following criteria:

1) Zero or more quantities are externally supplied: Input
2) At least one quantity is produced: Output
3) Each instruction is clear and unambiguous: Definiteness
4) The algorithm terminates after a finite number of steps: Finiteness
5) For every input instance, it halts with the correct output: Correct

Conventions used in writing algorithms:

1) Name of the algorithm: Every algorithm should be given a name which reflects the task performed by it.
2) Introductory comments: The task performed by the algorithm is described briefly. Any assumptions made by the algorithm are mentioned here.
3) Steps: Sequence of steps
4) Comments: Comments can be included within the body of the step by enclosing them in parentheses.

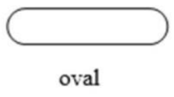
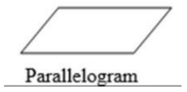
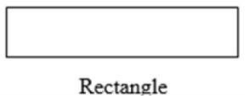
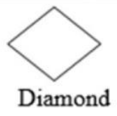
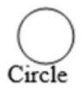
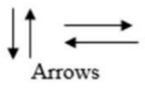
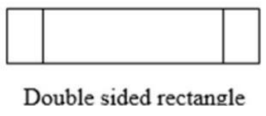
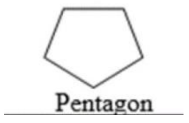
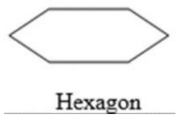
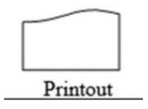
Example of an algorithm that computes the largest number:

Algorithm: Largest
This algorithm computes the largest of three numbers. The variables used here are:

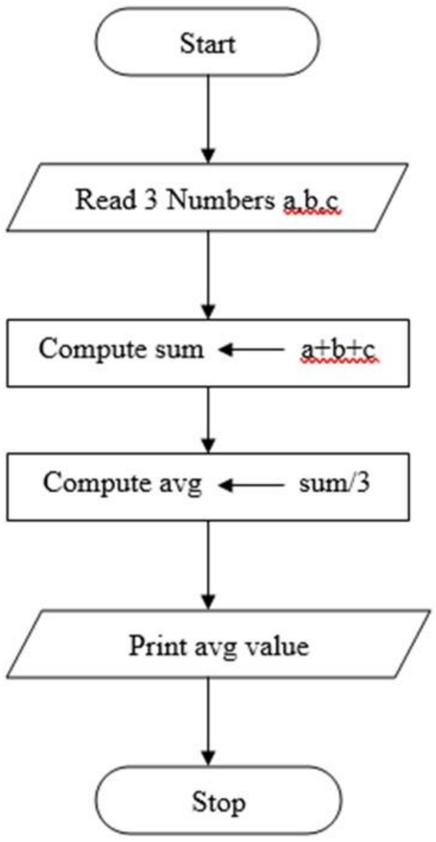
```
x, y, z  :      type integer
big      :      storing the value of the largest number, type integer
Step 1   :      read x, y, z      [Input the three integers]
Step 2 :    big = x           [Compute the largest of three numbers]
            if (y > big) big = y
            if (z > big) big = z
Step 3 :    write (big)       [Write the largest number]
Step 4 :    exit              [Finished]
```

Flowchart

Diagrammatic representation of an algorithm is called flow chart. Symbols used in flowchart are mentioned below −

Name	Symbol	Purpose
Terminal		start/stop/begin/end
Input/output		Input/output of data
Process		Any processing to be performed can be represented
Decision box		Decision operation that determine which of the alternative paths to be followed
Connector		Used to connect different parts of flowchart
Flow		Join 2 symbols and also represents the flow of execution
Pre-defined process		Module (or) subroutines specified elsewhere
Page connector		Used to connect flowcharts in 2 different pages
For loop symbol		shows initialization, condition and incrementation of loop variable
Document		Shows the data that is ready for printout

Example of flowchart



Difference Between Algorithm and Program

Algorithm is a step by step logical approach and program is a set of instructions for a given task.

What is an Algorithm?

An algorithm is a step-by-step and logical approach that defines a systematic process for computers to solve a specific problem. It consists of a set of rules that define how a task is to be executed to get the expected results. Algorithms are conceptual and can be described using language or flowcharts. We can implement them in different programming languages.

For example, here's an algorithm to add two numbers:

1. Start
2. Take two number inputs
3. Add both the numbers using the + operator
4. Display the result
5. End

What is a Program?

A program is a set of instructions that a computer follows to perform a specified task. Many programming languages can be used to write computer programs. Some of the popular programming languages include Python, Java, C++, JavaScript, PHP, and Ruby. These high-level programming languages are human-readable and writable. These languages are converted into low-level machine languages by compilers, interpreters, and assemblers within the computer system. A program tells the computer how to accept input, manipulate that input, and display the output in some form that humans find useful.

Algorithm vs Program: Difference Between Algorithm and Program

Computer algorithms solve the problem while computer programs implement them in a form that a computer can execute. Here are the main differences between algorithms and programs:

Algorithm	**Program**
It is a well-defined, step-by-step, logical procedure for solving a given problem.	It refers to a set of instructions for a computer to follow. A program can be an implementation of many algorithms, or a program can even contain no algorithms.
An algorithm provides abstract steps for processing one sequence of related information into a different sequence of derived information.	The constituents of a program may not be conceptually related.
It is written using plain English language and can be understood by those from a non-programming background.	It could be written in any programming language such as Python, Java, C++, JavaScript, or any other language, depending on the particular task the program is being designed for.
It can be expressed in natural language, flow charts, pseudocode, and in a variety of programming languages.	We write computer programs in a computer language. Then a compiler or interpreter translates it into a language that is understandable by any computer system.
An algorithm can be executed by a person.	A program is always executed by a Computer.