Components of a Computer

There are basically three important components of a computer:

1. Input Unit
2. Central Processing Unit (CPU)
3. Output Unit

1. Input Unit:

The input unit consists of input devices that take input and convert it into binary language. The input unit is used to provide data to the processor for further processing.

2. Central Processing Unit:

Once the information is entered into the computer by the input device, the processor processes it. The CPU first fetches instructions from memory and then interprets them so as to know what is to be done. The CPU has three main components, which are responsible for different functions: Arithmetic Logic Unit (ALU), Control Unit (CU) and Memory registers.

A. Arithmetic and Logic Unit (ALU): The ALU performs mathematical calculations and takes logical decisions.

- Arithmetic Logical Unit is the main component of the CPU
- Arithmetic and Logical Unit is a digital circuit that is used to perform arithmetic and logical operations.

B. Control Unit: The Control unit coordinates and controls the data flow in and out of the 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.

C. Memory Registers: A register is a temporary unit of memory in the CPU. These are used to store the data, which is directly used by the processor. Accumulator (ACC) is the main register in the ALU and contains one of the operands of an operation to be performed in the ALU.

3. Output Unit :

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

Components of Operating System

An Operating system is an interface between users and the hardware of a computer system. It is a organized collection of software consisting of procedures and functions, providing an environment for the execution of programs. The operating system manages resources of system software and computer hardware resources.

Important Components of the Operating System:

- Process management
- Files management
- Network management
- I/O device management
- Memory management

Language Processors

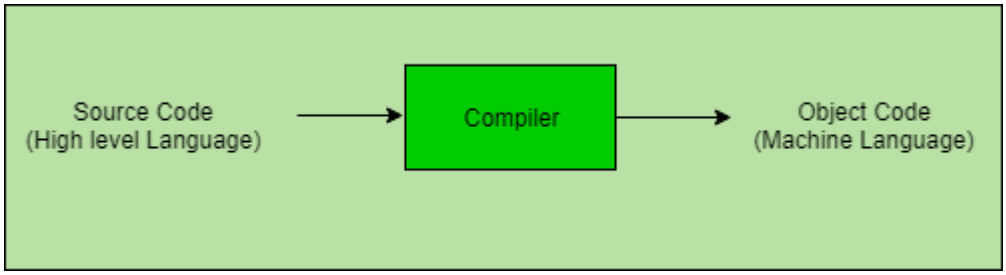
Compilers, interpreters, translate programs written in high-level languages into machine code that a computer understands and assemblers translate programs written in low-level or assembly language into machine code.

The programs are written mostly in high-level languages like Java, C++, Python etc. and are called source code.

Types of Language Processors

1. Compiler

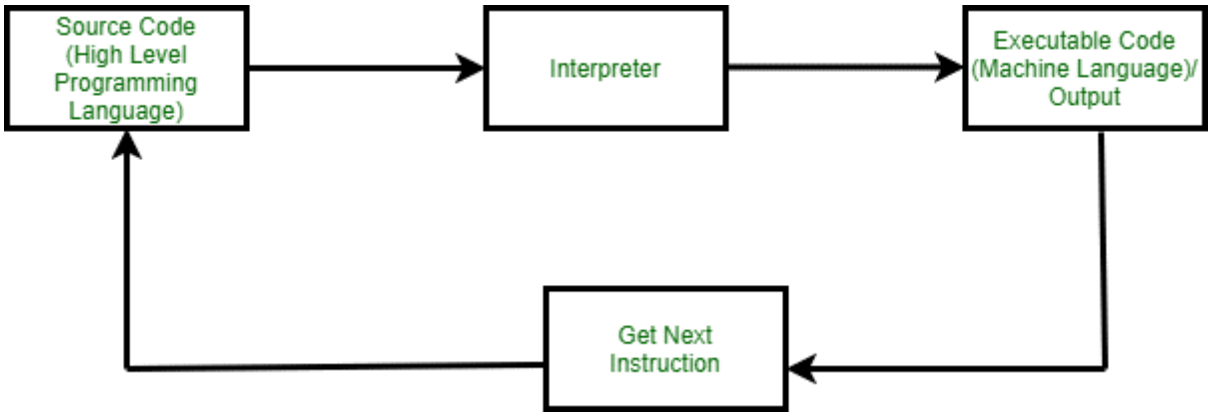
The language processor that reads the complete source program written in high-level language as a whole in one go and translates it into an equivalent program in machine language is called a Compiler. Example: C, C++, C#.



2. Interpreter

The translation of a single statement of the source program into machine code is done by a language processor and executes 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 the removal of the error.

Example: Perl, Python and Matlab.



Difference Between Compiler and Interpreter

Compiler	**Interpreter**
A compiler is a program that converts the entire source code of a programming language into executable machine code for a CPU.	An interpreter takes a source program and runs it line by line, translating each line as it comes to it.
The compiler takes a large amount of time to analyze the entire source code but the overall execution time of the program is comparatively faster.	An interpreter takes less amount of time to analyze the source code but the overall execution time of the program is slower.
The compiler generates the error message only after scanning the whole program, so debugging is comparatively hard as the error can be present anywhere in the program.	Its Debugging is easier as it continues translating the program until the error is met.

Compiler	Interpreter
The compiler requires a lot of memory for generating object codes.	It requires less memory than a compiler because no object code is generated.
Generates intermediate object code.	No intermediate object code is generated.
Examples: C, C++, C#	Examples: Python, Perl, JavaScript, Ruby.

3. Assembler

The Assembler is used to translate the program written in Assembly language into machine code. The output generated by the assembler is the object code or machine code understandable by the computer.

4.Linker :

A linker is a special program that combines the object files and other pieces of code to originate an executable file that has a .exe extension.

i. The linker optimizes the code generated by the compiler to reduce code size and improve program performance.
ii. The linker assigns memory addresses to the code and data sections of the program.
iii. The linker can link external libraries into the executable file to provide additional functionality.

5.Loader :

It is special program that takes input of executable files from linker, loads it to main memory, and prepares this code for execution by computer. Loader allocates memory space to program.

Differences Between Linker and Loader

Linker	Loader
The main function of Linker is to generate executable files.	Whereas main objective of Loader is to load executable files to main memory.

Linker	Loader
The linker takes input of object code generated by compiler/assembler.	And the loader takes input of executable files generated by linker.
Linking can be defined as process of combining various pieces of codes and object code to obtain executable code.	Loading can be defined as process of loading executable codes to main memory for further execution.
Another use of linker is to combine all object modules.	It helps in allocating the address to executable codes/files.

Algorithm

An **algorithm** is a finite sequence of well-defined instructions that can be used to solve a computational problem. It provides a step-by-step procedure that convert an input into a desired output.

Characteristics of an Algorithm:

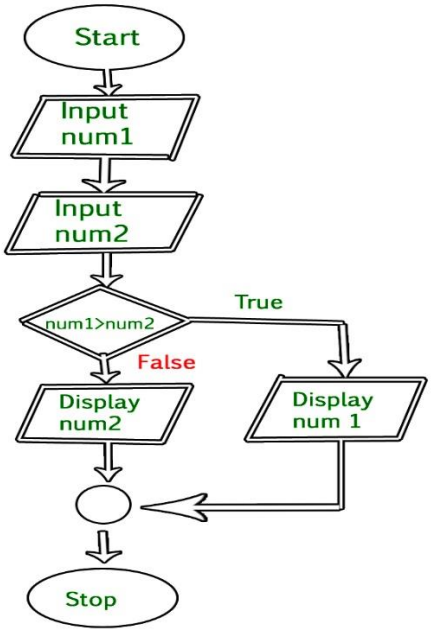
- **Clear and Unambiguous:** The algorithm should be unambiguous. Each of its steps should be clear in all aspects and must lead to only one meaning.
- **Well-defined Inputs:** It may or may not take input.
- **Well-defined Outputs:** It should produce at least 1 output.
- **Finiteness:** The algorithm must be finite, i.e. it should terminate after a finite time.
- **Feasible:** The algorithm must be simple, generic, and practical, such that it can be executed using reasonable constraints and resources.
- **Language Independent:** Algorithm must be language-independent, i.e. it must be just plain instructions that can be implemented in any language, and yet the output will be the same, as expected.

Flowchart

Flowchart is a graphical representation of an algorithm. Programmers often use it as a program-planning tool to solve a problem. It makes use of symbols which are connected among them to indicate the flow of information and processing.

Rules For Creating Flowchart :

Rule 1: Flowchart opening statement must be 'start' keyword.
Rule 2: Flowchart ending statement must be 'end' keyword.
Rule 3: All symbols in the flowchart must be connected with an arrow line.



Pseudo Code

A Pseudocode is defined as a step-by-step description of an algorithm. Pseudocode does not use any programming language in its representation instead it uses the simple English language text as it is intended for human understanding rather than machine reading.

```
FOR each index of string
    IF string(index) is equal to hash THEN
        RETURN "Hash found"
    END IF
END FOR
```

Error

Error is an illegal operation performed by the user which results in abnormal working of the program.

- **Syntax errors:** Errors that occur when you **violate the rules** of writing C/C++ syntax are known as syntax errors.
- **Run-time Errors:** Errors which occur during program execution(run-time) after successful compilation are called run-time errors.
- **Logical Errors:** On compilation and execution of a program, desired output is not obtained when certain input values are given.

Object Code and Executable Code

Object code is a sequence of statements in binary that is generated after compiling the source program. In contrast, an executable code is a file or a program that indicates tasks according to encoded instructions which are directly executed by the CPU.