Machine language

Sometimes referred to as **machine code** or **object code**, **machine language** is a collection of binary digits or bits that the computer reads and interprets. Machine language is the only language a computer is capable of understanding. Therefore all instructions and data should be written using binary codes 1 and 0. The binary code is called the machine code or machine language. The exact machine language for a program or action can differ by operating system on the computer.

Computer programs are written in one or more programming languages, like C++, Java, or Visual Basic. A computer cannot directly understand the programming languages used to create computer programs, so the program code must be compiled. Once a program's code is compiled, the computer can understand it because the program's code has been turned into machine language.

A compiler checks the entire user-written programme (known as source programme) and produces a complete programme in machine language (known as object programme). The source programme is retained for possible modifications and corrections and the object programme is loaded into the computer for execution.

For example, a program instruction may look like this:
1011000111101

Typical Machine language Instruction format

1. **OPCODE (Operation code)**
   OPCODE tells the computer which operation to perform from the instruction set of the computer.
2. **OPERAND (Address/Location)**
   OPERAND tells the address of the data on which the operation is to be performed.

Advantage of Machine Language:

The only advantage is that program of machine language run very fast because no translation program is required for the CPU.

Disadvantages of Machine language

- It is machine dependent i.e. it differs from computer to computer.
- It is difficult to program and write.
- It is prone to errors
- It is difficult to modify.

Assemble language/Symbolic language

It is a low level programming language that allows a user to write a program using alphanumeric mnemonic of instructions. It requires a translator as assembler to convert language into machine language so that it can be understood by the computer. It is easier to remember and write than machine language.

Using alphanumeric mnemonic codes instead of numeric cods for the instruction in the instruction set e.g. using ADD instead of 1110 (binary) or 14 (decimal) for instruction to add.

Allowing storage location to be represented in form of alphanumeric address instead of numeric address e.g. representing memory locations 1000, 1001 etc.

Assembler

It is a computer programme which converts or translate assembly language into machine language. It assembles the machine language program in the main memory of the computer and makes it ready for execution.

Assembly language program $\xrightarrow{\text{INPUT}}$ **ASSAMBLER** $\xrightarrow{\text{OUTPUT}}$ **Machine lang. program**

The main memory in a computer is called Random Access Memory. It is known as RAM. This is the part of the computer that stores operating system software, software application and other information for the CPU to have fast and direct access when needed to perform tasks.

Advantages of Assembly language

1. The symbolic programming of Assembly Language is easier to understand and saves a lot of time and effort of the programmer.
2. It is easier to correct errors and modify program instructions.
3. Assembly Language has the same efficiency of execution as the machine level language. Because this is one-to-one translator between assembly language program and its corresponding machine language program.

Disadvantages AssemblyLanguage:

1. One of the major disadvantages is that assembly language is machine dependent. A program written for one computer might not run in other computers with different hardware configuration. Long programs written in such languages cannot be executed on small sized computers.

2. It takes lot of time to code or write the program, as it is more complex in nature.

High level languages

High level language is abbreviated as **HLL**. High level languages are similar to the human language. Unlike low level languages, high level languages are programmers friendly, easy to code, debug and maintain.

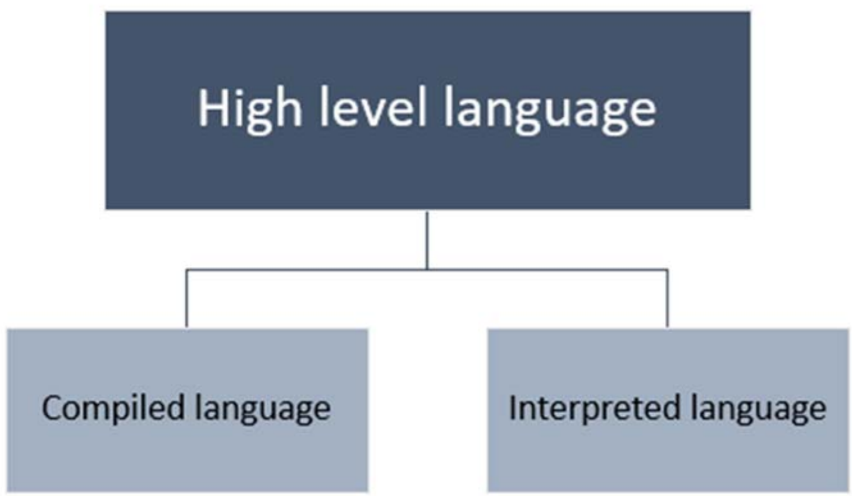
High level language provides higher level of abstraction from machine language. They do not interact directly with the hardware. Rather, they focus more on the complex arithmetic operations, optimal program efficiency and easiness in coding.

Low level programming uses machine friendly language. Programmers writes code either in binary or assembly language. Writing programs in binary is complex and cumbersome process. Hence, to make programming more programmers friendly. Programs in high level language is written using English statements.

High level programs require compilers/interpreters to translate source code to machine language. We can compile the source code written in high level language to multiple machine languages. Thus, they are machine independent language.

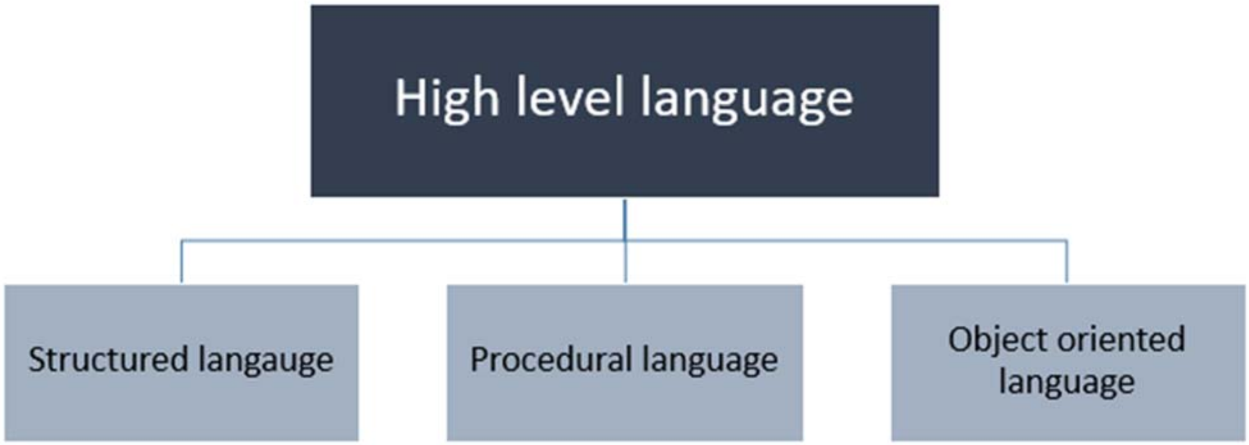
Today almost all programs are developed using a high level programming language. We can develop a variety of applications using high level language. They are used to develop desktop applications, websites, system software's, utility software's and many more.

High level languages are grouped in two categories based on execution model – compiled or interpreted languages.



Classification of high level language on the basis of execution model

We can also classify high level language several other categories based on [programming paradigm](#).



Classification of high level language on the basis of paradigm

Advantages of High level language

1. High level languages are programmer friendly. They are easy to write, debug and maintain.
2. It provide higher level of abstraction from machine languages.
3. It is machine independent language.
4. Easy to learn.
5. Less error prone, easy to find and debug errors.
6. High level programming results in better programming productivity.

Disadvantages of High level language

1. It takes additional translation times to translate the source to machine code.
2. High level programs are comparatively slower than low level programs.
3. Compared to low level programs, they are generally less memory efficient.
4. Cannot communicate directly with the hardware.