**Programming Languages**

To write a program (tells what to do) for a computer, we must use a computer language. Over the years computer languages have evolved from machine languages to natural languages. The following is the summary of computer languages

1940‘s -- Machine Languages

1950‘s -- Symbolic Languages

1960‘s -- High Level Languages

Machine Language

In the earliest days of computers, the only programming languages available were machine languages. Each computer has its own machine language which is made of streams of 0‘s and 1‘s. The instructions in machine language must be in streams of 0‘s and 1‘s. This is also referred as binary digits. These are so named as the machine can directly understand the programs

Advantages:

1. High speed execution

2. The computer can understand instructions immediately

3. No translation is needed.

Disadvantages:

1. Machine dependent

2. Programming is very difficult

3. Difficult to understand

4. Difficult to write bug free programs

5. Difficult to isolate an error

Symbolic Languages (or) Assembly Language

In the early 1950‘s Admiral Grace Hopper, a mathematician and naval officer, developed the concept of a special computer program that would convert programs into machine language. These early programming language simply mirrored the machine languages using symbols or mnemonics to represent the various language instructions. These languages were known as symbolic languages. Because a computer does not understand symbolic language it must be translated into the machine language. A special program called an **Assembler** translates symbolic code into the machine language. Hence they are called as Assembly language.

Advantages:

1. Easy to understand and use

2. Easy to modify and isolate error

3. High efficiency

4. More control on hardware

Disadvantages:

1. Machine Dependent Language

2. Requires translator

3. Difficult to learn and write programs

4. Slow development time

5. Less efficient

**High-Level Languages**

Although symbolic languages greatly improved programming efficiency, they still required programmers to concentrate on the hardware. 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 problems being solved led to the development of high-level languages. High-level languages are portable to many different computers allowing the programmer to concentrate on the application problem at hand rather than the intricacies of the computer. Examples of high level languages include C, C++, C#, Python etc.

Advantages:

1. Easy to write and understand

2. Easy to isolate an error

3. Machine independent language

4. Easy to maintain

5. Better readability

6. Low Development cost

7. Easier to document

8. Portable

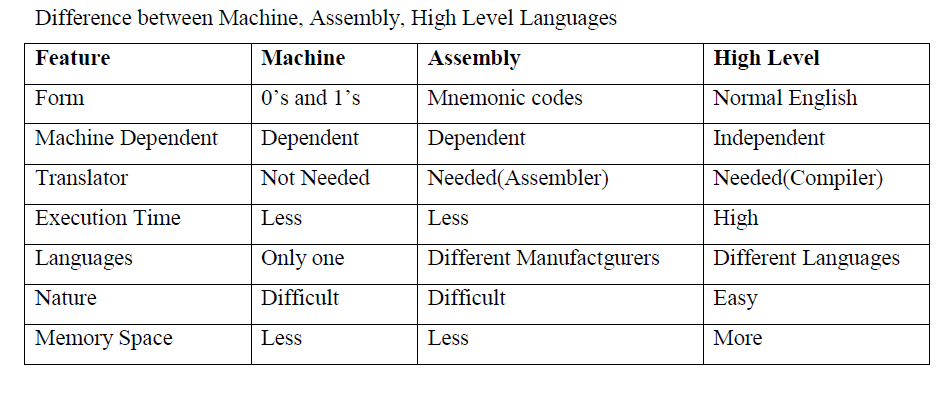
Disadvantages:

1. Needs translator

2. Requires high execution time

3. Poor control on hardware

4. Less efficient



**Language Translators**

These are the programs which are used for converting the programs in one language into machine language instructions, so that they can be executed by the computer.

* **Compiler**: It is a program which is used to convert the high-level language programs into machine language .
* **Assembler**: It is a program which is used to convert the assembly level language programs into machine language.
* 3) **Interpreter**: It is a program, it takes one statement of a high-level language program, translates it into machine language instruction and then immediately executes the resulting machine language instruction and so on.

**Comparison between compiler and interpreter**

|  |  |
| --- | --- |
| **Compiler** | **Interpreter** |
| A Compiler is used to compile an entire program and an executable program is generated through the object program | An interpreter is used to translate each line of the program code immediately as it is entered |
| The executable program is stored in a disk for future use or to run it in another computer | The executable program is generated in RAM and the interpreter is required for each run of the program |
| The compiled programs run faster | The Interpreted programs run slower |
| Most of the Languages use compiler | A very few languages use interpreters |