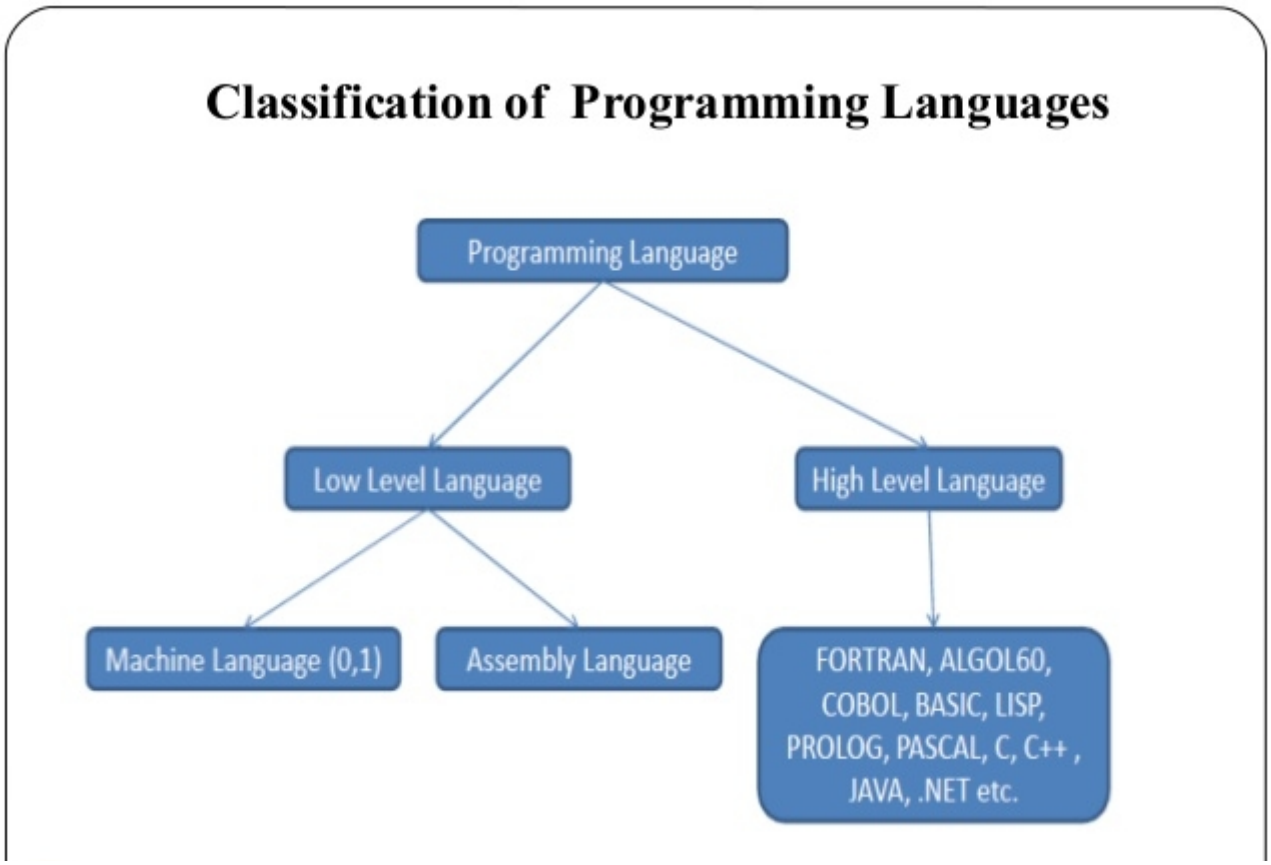
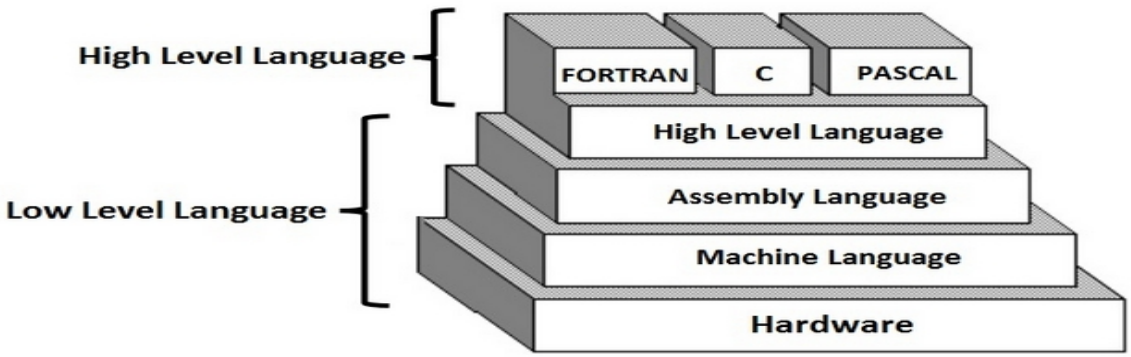
Computer Language:

- A computer language is an artificial language designed to work for different applications in different environments.
- A programming language is a vocabulary and set of grammatical rules for instructing a computer or computing device to perform specific tasks. Languages that programmers use to write code are called "high-level languages." This code can be compiled into a "low-level language," which is recognized directly by the computer hardware.
- Each programming language has a unique set of keywords (words that it understands) and a special syntax for organizing program instructions.





Computer Language and its Types

Two Basic Types of Computer Language

• **Low-Level Languages:** A language that corresponds directly to a specific machine
• **High-Level Languages:** Any language that is independent of the machine

Low-Level Languages:

- Low-level computer languages are either machine codes or are very close them.
- A computer cannot understand instructions given to it in high-level languages or in English.
- It can only understand and execute instructions given in the form of machine language i.e. binary.
- There are two types of low-level languages:
  Machine Language: a language that is directly interpreted into the hardware
  Assembly Language: a slightly more user-friendly language that directly corresponds to machine language

➢ **Machine language:**

- Machine language is the lowest and most elementary level of programming language and was the first type of programming language to be developed.
- Machine language is basically the only language that a computer can understand and it is usually written in hex.
- In fact, a manufacturer designs a computer to obey just one language, its machine code, which is represented inside the computer by a string of binary digits (bits) 0 and 1.

Advantages:

- Machine language makes fast and efficient use of the computer.
- It requires no translator to translate the code. It is directly understood by the computer.

Disadvantages:

- All operation codes have to be remembered.
- All memory addresses have to be remembered.
- It is hard to find errors in a program written in the machine language.

➢ Assembly Language:

- Assembly language was developed to overcome some of the many inconveniences of machine language.
- This is another low-level but very important language in which operation codes and operands are given in the form of alphanumeric symbols instead of 0's and l's .
- These alphanumeric symbols are known as mnemonic codes and can combine in a maximum of five-letter combinations e.g. ADD for addition, SUB for subtraction, START, LABEL etc. Because of this feature, assembly language is also known as 'Symbolic Programming Language.'
- The instructions of the assembly language are converted to machine codes by a language translator and then they are executed by the computer.

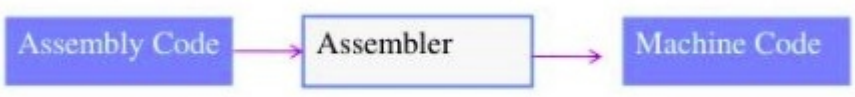
Advantages:

- Assembly language is easier to understand and use as compared to machine language.
- It is easy to locate and correct errors.
- It is easily modified.

Disadvantages:

- Like machine language, it is also machine dependent/specific.
- Since it is machine dependent, the programmer also needs to understand the hardware.

Assembly Language	*Machine Language*
`ST 1,[801]`	`00100101 11010011`
`ST 0,[802]`	`00100100 11010100`
`TOP: BEQ [802],10,BOT`	`10001010 01001001 11110000`
`INCR [802]`	`01000100 01010100`
`MUL [801],2,[803]`	`01001000 10100111 10100011`
`ST [803],[801]`	`11100101 10101011 00000010`
`JMP TOP`	`00101001`
`BOT: LD A,[801]`	`11010101`
`CALL PRINT`	`11010100 10101000`
	`10010001 01000100`



High Level Language

- High-level computer languages use formats that are similar to English.
- The purpose of developing high-level languages was to enable people to write programs easily, in their own native language environment (English).
- High-level languages are basically symbolic languages that use English words and/or mathematical symbols rather than mnemonic codes.
- Each instruction in the high-level language is translated into many machine language instructions that the computer can understand.
- **Examples of high-level programming languages** in active use today include Python, Visual Basic, BASIC, FORTRAN, ALGOL, COBOL, C++, Java , Perl, PHP, Ruby, C# and many others.

Advantages:

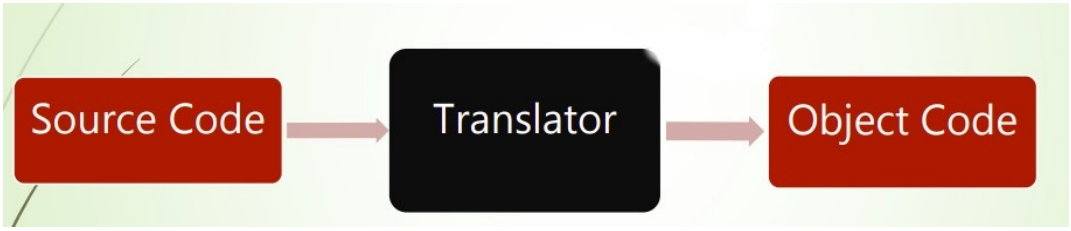
- High-level languages are user-friendly.
- They are easier to read, write and maintain.
- They are problem-oriented rather than 'machine'-based.
- A program written in a high-level language can be translated into many machine languages and can run on any computer for which there exists an appropriate translator.
- The language is independent of the machine on which it is used.
- High Level Language is portable, i.e. they can work on different operating system.
- Length of the program is also small compared with low level.
- Less error prone, easy to find and debug errors.

Disadvantages

- A high-level language has to be translated into the machine language by a translator, which takes up time.
- High Level Language are comparatively slower than low-level programs.
- Compared to low level programs, they are generally less memory efficient.
- Cannot communicate directly with the hardware.

Language Translator

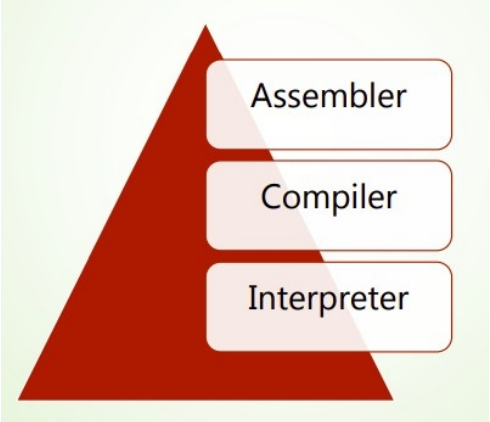
- Language translator is a program that converts the source code in to the object code.



Why Language Translators?

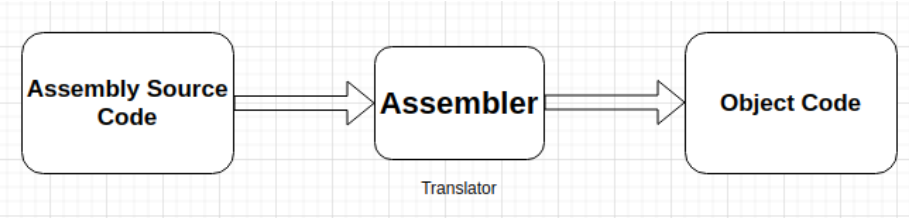
- Computer only understands object code (machine code).
- It does not understand any source code.
- There must be a program that converts source code in to the object code so that the computer can understand it.
- The language translator is one which does this job.
- The programmer writes the source code and then translator converts it in machine readable format (object code).

Types of Language Translators
There are Three types of Language Translator:



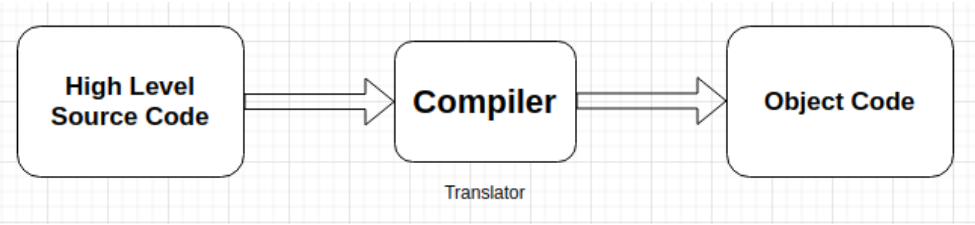
Assembler

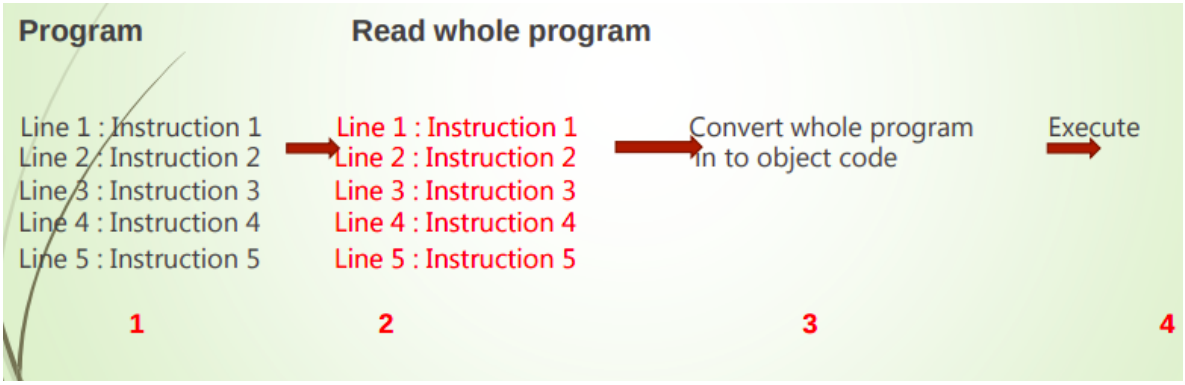
- Assembler is the language translator that converts assembly language code in to the object code (machine code).



Compiler

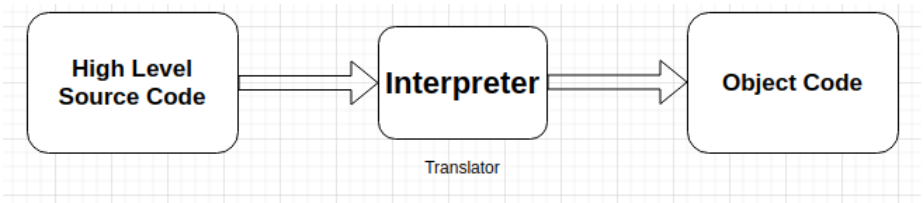
- Compiler is the language translator that converts high level language code in to the object code (machine code).
- It converts the whole code at a time.

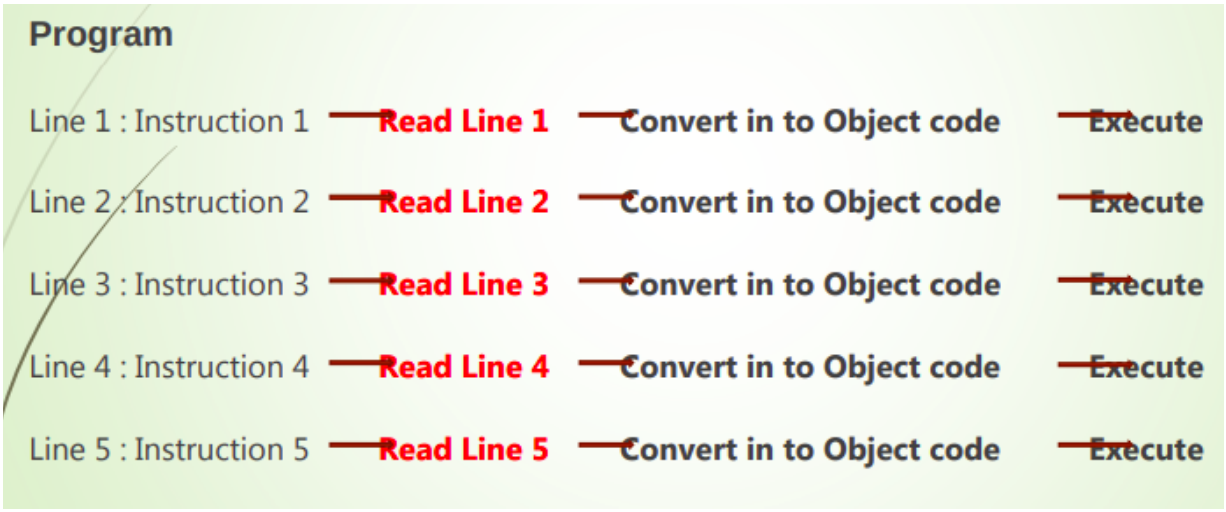


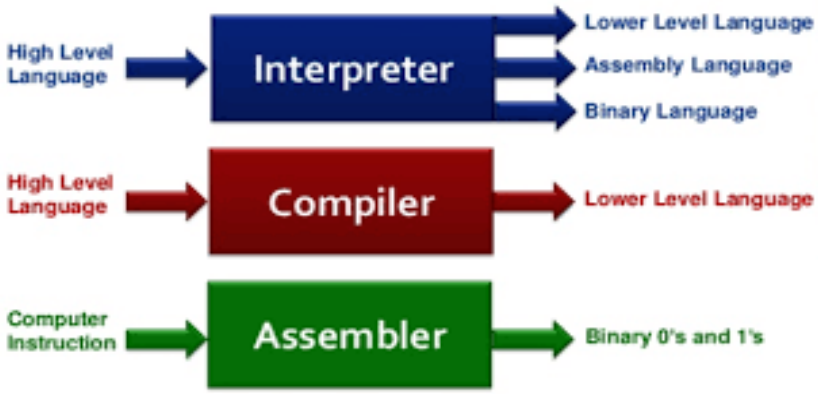


Interpreter

- Interpreter is the language translator that converts high level language code in to the object code (machine code).
- It converts the code line by line.







COMPILER VS INTERPRETER VS ASSEMBLER

Software that converts programs written in a high level language into machine language	Software that translates a high level language program into machine language	Software that converts programs written in assembly language into machine language
Converts the whole high level language program to machine language at a time	Converts the high level language program to machine language line by line	Converts assembly language program to machine language
Used by C, C++	Used by Ruby, Perl, Python, PHP	Used by assembly language