Introduction

Programming languages :-

Interactions involving humans are most effectively carried out through the medium of language . language permits the expression of thoughts and ideas , and without it , communication as we know it would be very difficult indeed .

In computer programming , programming language serves as means of communication between the person with a problem and the computer used to solve it . programming language is a set of symbols , words , and rules used to instruct the computer .

A hierarchy of programming languages based on increasing machine independence include the following :-

1- machine language : is the actual language in which the computer carries out the instructions of program . otherwise , " it is the lowest form of computer language , each instruction in program is represented by numeric cod , and numeric of addresses are used throughout the program to refer to memory location in the computer memory .

2- Assembly languages : is a symbolic version of a machine language ,each operation code is given a symbolic code such a **Add** , **SUB** ,…. Moreover , memory location are given symbolic name , such as **PAY** , **RATE** .

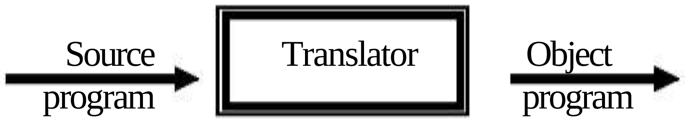
3-high - level language :Is a programming language where the programming not require knowledge of the actual computing machine to write a program in the language .H.L.L . offer a more enriched set of language features such as control structures , nested statements , block …

4- problem-oriented language : It provides for the expression of problems in a specific application . Examples of such language are **SQL** for Database application and **COGO** for civil engineering applications .

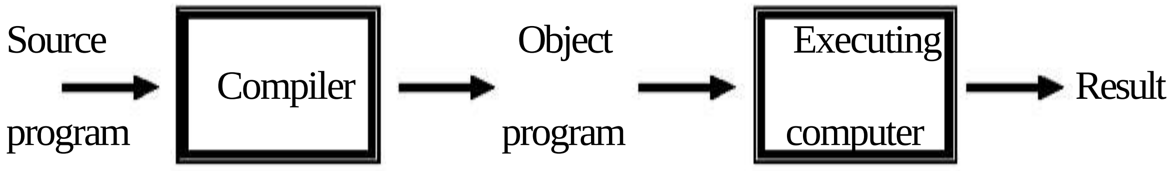
Advantages of H.L.L over L.L.L include the following :

1- **H.L.L** are easier to learn then **L.L.L**
2- A programmer is not required to know how to convert data from external from to internal within memory .
3- Most **H.L.L** offer a programmer a variety of control structures which are not available in **L.L.L**
4- Programs written in H.L.L are usually more easily **debugged** than **L.L.L.** equivalents.
5- Most **H.L.L** offer more powerful data structure than **L.L.L.**
6- Finally ,High level languages are relatively **machine-independent**. Consequently certain programs are portable

Translator: High- Level language programs must be translated automatically to equivalent machine- language programs .
A translator input and then converts a " source program" into an object or target program . the source program is written in a source language and the object program belong to an object language .



1- If the source program is written in *assembly language* and the target program in machine language .the translator is called " **Assembler** "
2- If the source language is H.L.L. and the object language is L.L.L. ,then the translator is called " **Compiler** " .
3- If the source language is L.L.L. and the object language is H.L.L., then the translator is called "**Decompiler**"



- The time at which conversion of a source program to an object program occurs is called " **Compile time** " .The object program is executed at " **Run time** " ,note that the source program and data are process at different time .

Another kind of translator ,called an " **Interpreter** " in which processes an internal form of source program and data at the same time . that is interpretation of the internal source from occurs at run time and no object program is generated .

Interpreters

An interpreter is another way of implementing a programming language. Interpretation shares many aspects with compiling. Lexing, parsing and type-checking are in an interpreter done just as in a compiler. But instead of generating code from the syntax tree, the syntax tree is processed directly to evaluate expressions and execute statements, and so on. An interpreter may need to process the same piece of the syntax tree (for example, the body of a loop) many times and, hence; interpretation is typically slower than executing a compiled program. But writing an interpreter is often simpler than writing a compiler and the interpreter is easier to move to a different machine, so for applications where speed is not of essence, interpreters are often used.