جامعة الانبار

كلية علوم الحاسوب وتكنولوجيا المعلومات

قسم نظم المعلومات

COMPILER

Principles & Techniques

م.م يقين سعد علي

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 from 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 …

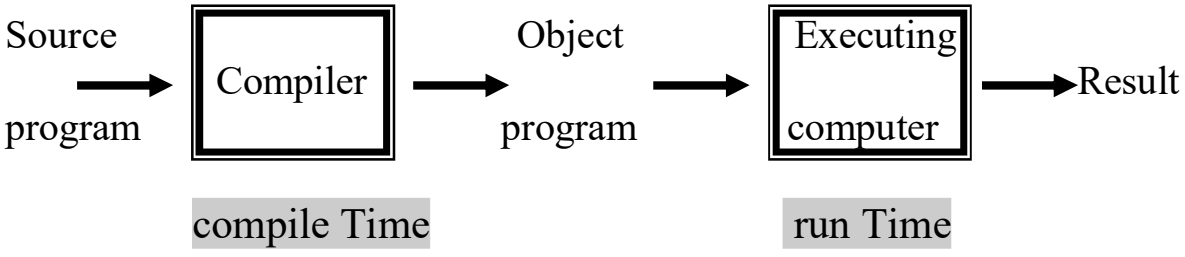
4- problem-oriented language : It provides for the expression of problems in a specific application . Examples of such language are **SQL** for Database application and **COGO** for civil engineering applications .

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 .

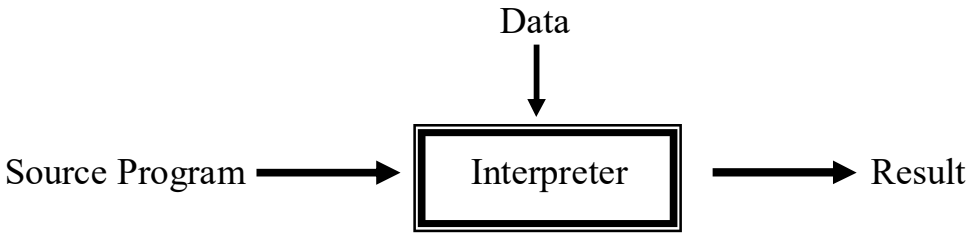
1- If the source program is written in *assembly language* and the target program in machine language .the translator is called " **Assembler** "
2- If the source language is H.L.L. and the object language is L.L.L. ,then the translator is called " **Compiler** " .
3- If the source language is L.L.L.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



(compilation Process)

- 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 .

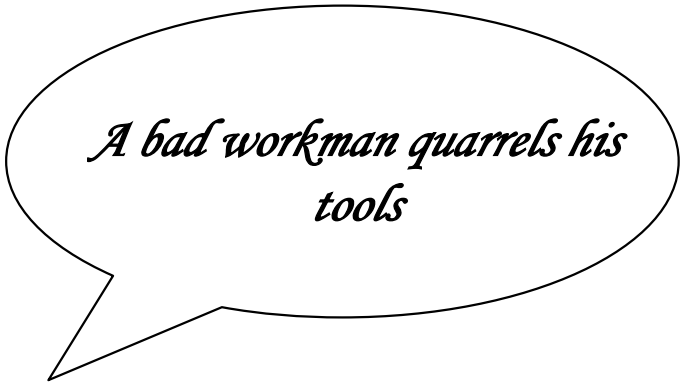
Another kind of translator ,called an " **Interpreter** " in which processes an internal form of source program and data at the same time . that is interpretation of the internal source from occurs at run time and no object program is generated .



(Interpretive process)

Compiled programs usually run faster than interpreter ones because the overhead of understanding and translating has already been done .However ,Interpreters are frequent easier to write than Compilers , and can more easily support interactive debugging of program .

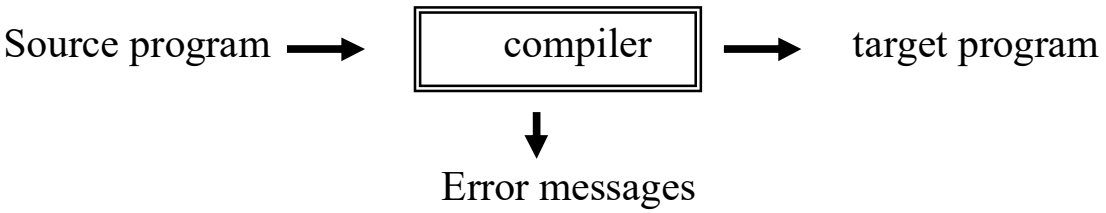
Remark :Some programming language implementations support both interpretation and compilation.



Compilation concepts

What is compiler?

A compiler is a program that translates a computer program(source program) written in H.L.L (such as Pascal,C++) into an equivalent program (target program) written in L.L.L.

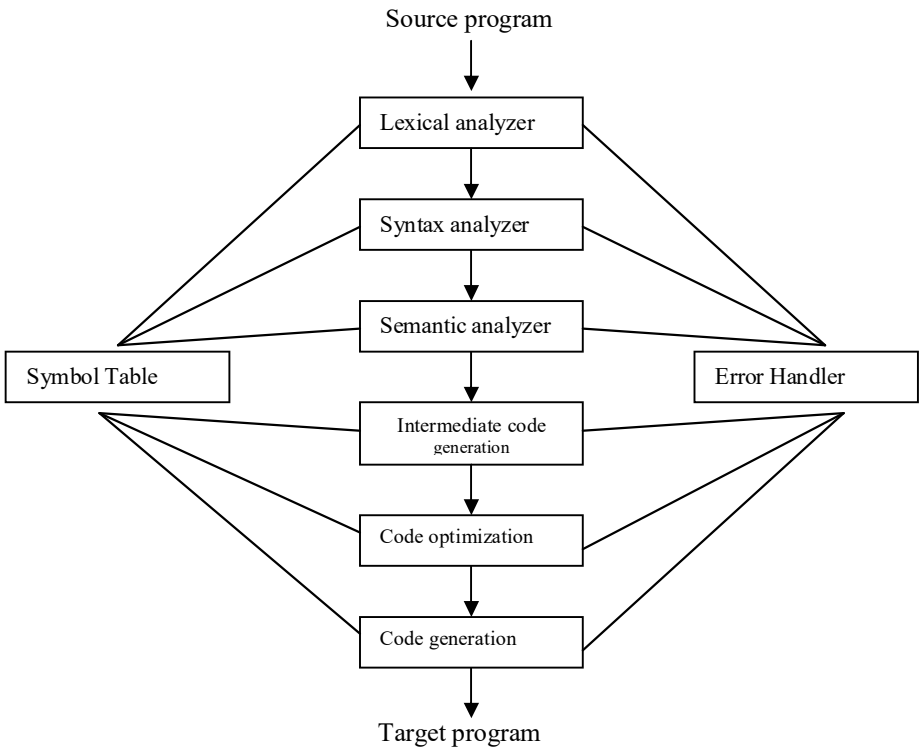


Model of Compiler:

The task of constructing a compiler for a particular source language is complex. The complexity of the compilation process depend on the source language.A compiler must perform two major tasks:

1. Analysis :deals with the decomposition of the source program into its basic parts.
2. Synthesis:builds their equivalent object program using these basic parts.

To perform these tasks, compiler operates in phases each of which transforms the source program from one representation to another. A typical decomposition of a compiler is shown in the following figure.



Compiler phases

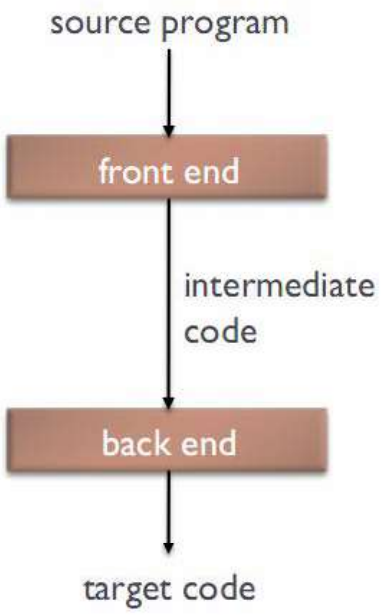
1. **Lexical Analyzer**: whose purpose is to separate the incoming source code into small pieces (tokens) , each representing a single atomic unit of language, for instance "keywords", "Constant "," Variable name" and "Operators".
2. **Syntax Analyzer** : whose purpose is to combine the tokens into well formed expressions (statements) and program and it check the syntax error
3. **Semantic Analyzer**: whose function is to determine the meaning of the source program.
4. **Intermediate Code Generator**: at this point an internal form of a program is usually created.For example:

Y=(a+b)*(c+b) ⟹
(+,a,b,t1)
(+,c,d,t2)
(*,t1,t2,t3)

5. **Code Optimizer** :Its purpose is to produce a more efficient object program (Run faster **or** take less space **or** both)
6. **Code Generator**: Finally, the transformed intermediate representation is translated into the target language.

The grouping of phases : the phases of compiler are collection into :

1. **Front-End** :It consists of those phases that depend on the *source language* and are largely independent of the *target machine* ,those include : (**lexical analysis** ,**syntax analysis** , **semantic analysis**, and **intermediate code generation**)
2. **Back-End** : Includes those phases of compiler that depend on the *target machine* and not depend on the *source language* . these include:(**code optimization phase** and **code generation phase**)



The grouping of Compiler Phases