software:

Another type of computer system is software . A software is computer program that tells the computer how to perform particular tasks.

There are two kinds of software :

A- **Application software** : such as word processing , excel , …and programming language that writing by the user.

B- **System software:** such as operating system , complier,…

A) Application software:

1- Programming Language:

It is list of instructions(program) that enable a computer to perform a specific task. A computer programs can be written in **low level language** and **high level language.**

1.1 Low level language :

Low level language is a type of programming language that used to write program that relate to the specific architecture and hardware of particular type of computer . Low level languages are closer to the native language of computer (binary data) , making them harder for programmers to understand .

***Two common types of low-level programming languages** are **machine language** and **assembly language** _

A:- Machine language(Machine Code) :

A machine language is low- level programming language designed for specific type of processor .

The **machine language** is made up of **instructions** and **data** that written in **binary numbers**. It is the only language understood by a computer .

It is very difficult to understand by the user , but it is the only language that the computer can work with . All programming languages must be **converted** into **machine language** , since the computer is digital device , it only recognize **binary data program** Machine language instruction that has a binary form can be **directly executed by a computer's central processing unit (CPU).** Each instruction causes the CPU to perform a very specific task, such as a **load**, a **store**, a **jump**, or an **arithmetic/ logic** operation on one or more data in the CPU's **registers** or **memory.**

Every CPU(Processor) has its own **unique machine language** , Because machine languages are designed to one specific processor architecture, so they are **not portable**.

Below is an example of machine language (binary) for print text " Hello Worid"

```
01001000 01100101 01101100 01101100 01101111 00100000
01010111 01101111 01110010 01101100 01100100
```

Note: Machine language is called **object code** or **target code**

B-Assembly language :

An assembly language is low- level programming language designed for specific type of processor .

As the machine code is written in binary format, it is very difficult to read, so using **assembly language** programmers can write human –readable program that correspond almost exactly to machine language.

It contains the same instruction as a machine language , but the instruction and variable have **abbreviation English names called**(**mnemonic codes**) instead of using the **patterns of bits** and more **understandable form** . For example, it is much easier to remember that “AND” refers to an instruction that performs logic “and” operation, instead of memorizing a code “100001”.

A mnemonic code is a **symbolic name of abbreviation of specific an operation** . such as MOV (move), ADD (add), AND(and) and SUB (subtract). These commands perform basic operations, such as moving values into memory registers and performing calculations.

EX: TO add the 8 to the value in accumulator register A .
IN Machine language as flowing :
10100000 00001000
While in assembly language as following
ADD A, 8h

Because assembly languages are **designed to one specific processor architecture,** they are **not portable**.

An assembly language is a low-level programming language that requires a **software** called an **assembler** to convert it into **machine code**, and then **executed** by a **computer's central processing unit (CPU).**

The Assembly language is **an intermediate language** between **high-level language** and **machine code.**

The **main difference** between **machine code** and **assembly language** is that the **machine code** is a language consisting of **binaries** data that can be **directly executed by a cpu(Processor) while an assembly language** is language that has **syntaxes similar to English, word such as ADD ,MOV…,** and requires a software called an **assembler** to convert it into **machine code** and then **executed by cpu (processor**). Since computers are digital devices, they only recognize binary data

1.2; High –level – language (HLL)

High level language is another type of programing language .High level language is easier to learn and understood than low level language , because high level language uses English and mathematical symbols in instructions .

Some of common high level language are ;

- Fortran (Formula Translation) for engineers.
- Cobol (Common Business Oriented) for business programmers.
- Basic(Beginner's All- purpose Symbolic Code)
- Pascal
- C++

Unlike assembly and machine language , high level languages programs may be used with different makes of computers ,so High level languages are portable .

High-level **languages** programs, must be compiled (translated) into **machine language** by using a program called **Compiler** or **Interpreter** before the **program** is run on a computer. Since computers are digital devices, they only recognize binary data.

Both compilers and interpreters are used to convert a program written in a high-level language(source code) into machine language(object code) understood by computers. However, there are differences between how an interpreter and a compiler works.

The advantages of high level languages are :

1) They are easier to learn than low- level languages
2) They are easier to use for problem solving .
3) They require less time to write a program
4) They provide better documentation .

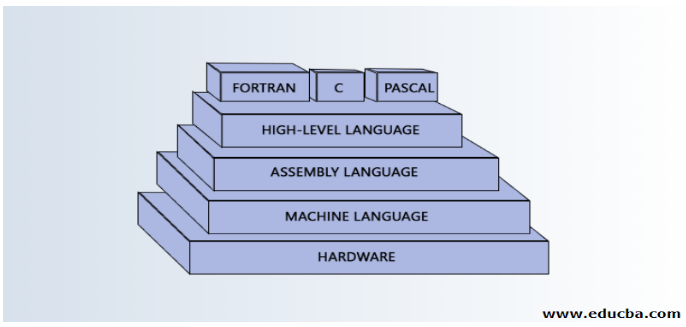


Fig: hierarchy programing language

Let's see the difference between high level and low level languages:

	HIGH LEVEL LANGUAGE	LOW LEVEL LANGUAGE
1.	It is programmer friendly language	It is a machine friendly language.
2.	It is easy to understand.	It is tough to understand.
3.	It is simple to debug.	It is complex to debug .
4.	It is portable.	It is non-portable.
5.	It needs compiler or interpreter for translation.	It needs assembler for translation.

.

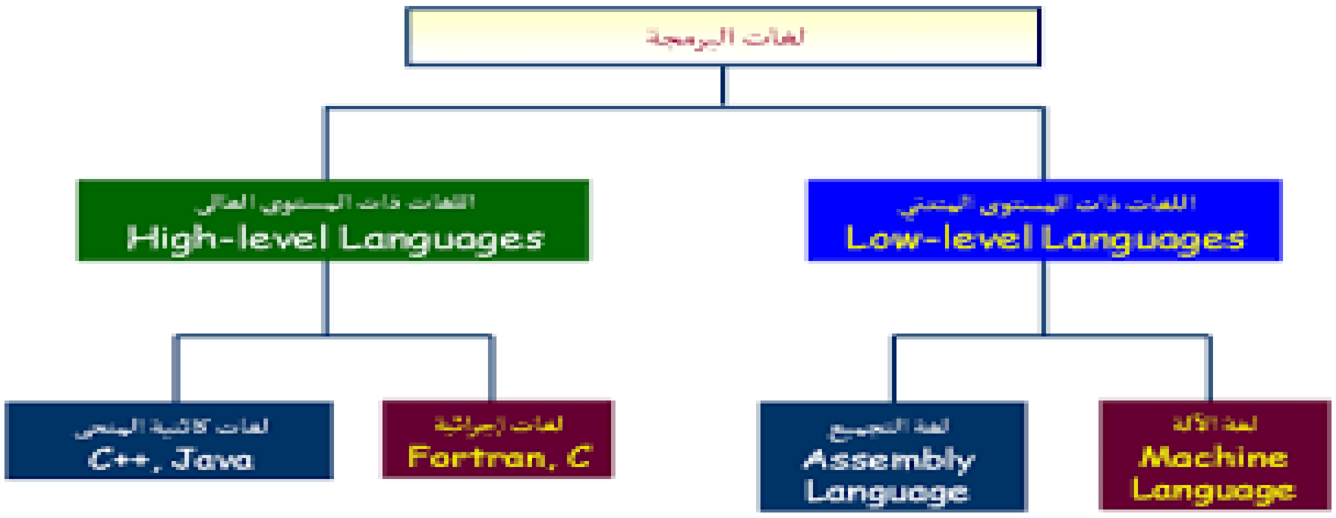


Fig: programming languages type

Note: Machine language is called **target code** or **object code**

Note ; **code** is also called **a program**

Note; a program written in **high level language** and in **assembly language** is called **source program**

B)System softwre

1- A language translator: is a software which translates the a source language(high level language) into an object language(machine language).

compilers, interpreters and assemblers are examples of a language translator which convert source program into object program , but each term has specific work.

Type of translator:

1-1 complier: is a computer program that **scan(read)** a program written in the **high-level language(source program)** and converts it into the **machine code(object program)**

The work of compler

The complier **scans(read)** the entire program , **analysis it** and then **translates** it as a whole into **machine code .**

A compiler generates error message only after scanning and analyzing the whole program . Hence debugging is comparatively hard . After removing these errors , a complier generate intermediate object code , and then convert it to machine code which is ready to execute by CPU.

1.2 Interpreter:

an **interpreter** is a **computer program** that directly **translates** and **executes i**nstructions written in a high level programming**(source program),**without producing **any object code** first. It dose this by fetching the source program instructions **one by one , analyzing them one by one** and then **executing** them **one by one .**

<u>The work of interpreter</u>

The interpreter translates the source code(high level program) instruction-by-instruction during RUN Time. Since interpreter take each instruction one by one in source program scan it, translates it and executes it directly at the same time .,i.e translation and execution take place for each instruction at the same time .

interpreter generates the error message after scanning and analyzing of each instruction one by one in the source program , it's easier to detect errors. when an error takes place in the instruction , the interpreter prevents its translation and after removing the error from instruction , translation and execution resumes

The difference between compiler and interpreter

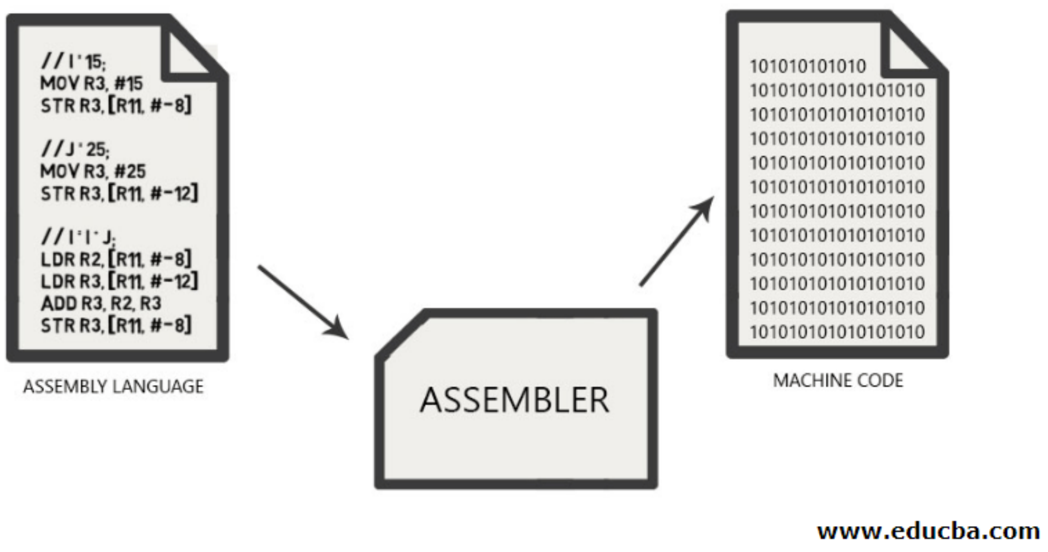
1- The compiler takes a program as a whole and translates	**1- interpreter translates a program instruction by instruction.**

2- The translation is done before execution.	2- translation and execution take place at same time
3- Display all errors after analyzing the whole program, all errors at the same time and it’s difficult to detect the errors	3- Displays error of each instruction one by one and it’s easier to detect errors.
4- In compiler when an error occurs in the program, it stops its translation and after removing error whole program is translated again.	4- when an error takes place in the instruction, it prevents its translation and after removing the error, translation and execution resumes.

5- It generates intermediate object code.	**5- It does not produce any intermediate object code.**
6- Memory requirement is more due to the creation of intermediate object code.	**6 It requires less memory as it does not create intermediate object code**

1.3 – Assembler

An assembler is a **translator program** used to **translate** a program written in **assembly language**(source program) into **machine languag**e(object code)



```
//I'15;
MOV R3, #15
STR R3, [R11, #-8]

//J'25;
MOV R3, #25
STR R3, [R11, #-12]

//I'I'J;
LDR R2, [R11, #-8]
LDR R3, [R11, #-12]
ADD R3, R2, R3
STR R3, [R11, #-8]
```

```
101010101010
1010101010101010
1010101010101010
1010101010101010
1010101010101010
1010101010101010
1010101010101010
1010101010101010
1010101010101010
1010101010101010
1010101010101010
1010101010101010
```

.

المحاضرة السابعة

المرحلة الاولى

تقنيات الحاسبة

م. وداد عبد الخضر