Make Comparative analysis of Assembly language, Machine Language and High level languages respectively.

## Levels of Programming Languages

Programming is the act of developing a piece of software. And software contains instructions that tell a computer what to do. Therefore, programming is the process of developing instructions that tell a computer what to do.

And you can tell the computer what to do with a programming language. Each programming language was invented for a specific purpose. Each succeeding programming language builds on the strength of it’s predecessors. For example, Machine Language is succeeded by Assembly Language, which is succeeded by high-level languages. In other words, High-level programming languages are build on Assembly Language, which is built on Machine Language. The idea is that higher level languages calls functions of a lower level languages with a single-simple short code.

### What is Machine Language?

A computer’s native language is called Machine Language. Machine language is the most primitive or basic programming language that starts or takes instructions in the form of raw binary code. So that if you wanted to give a computer an instruction in its native or Machine language, you have to manually enter the instructions as binary code.

For example, adding two numbers together in machine language would look like this:

1101101010011010

### What is Assembly Language?

Programming in Machine language is tedious (you have to program every command from scratch) and hard to read & modify (the 1s and 0s are kind of hard to work with…). For these reasons, Assembly language was developed as an alternative to Machine language.

Assembly Language uses short descriptive words (mnemonic) to represent each of the Machine Language instructions.

For example the mnemonic **add** means to add numbers together, and **sub** means to subtract the numbers. So if you want to add the numbers 2 and 3 in assembly language, it would look like this:

add 2, 3, result

So Assembly Languages were developed to make programming easier. However, the computer cannot directly execute the assembly language. First another program called the **assembler**is used to translate the Assembly Language into machine code.



### What is a High-Level Language?

High-Level languages are platform independent, meaning that you can write & run High-Level Languages on different types of machines. High-Level Languages are English like and therefore easier to learn and use. Note that instructions in a High-Level Language are called **statements**.

Note that a program written in a high-level language is called the **source code**. Note that the Source Code must be translated into machine code before the computer can execute the source code. And the translations are done by programming tools called an **interpreter** or **compiler**.

Here’s an example of a High-Level Language statement that calculates the area of a circle with a radius of 5:

area = 5 \* 5 \* 3.14159;

Examples of High-Level Programming Languages include Ada, BASIC, C, C++, C#, COBOL, FORTRAN, Java, Pascal, Python, and Visual Basic.

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.  The symbol 0 stands for the absence of an electric pulse  1 stands for the presence of an electric pulse.

5. Advantages Disadvantages Machine language makes fast and efficient use of the computer. All operation codes have to be remembered It requires no translator to translate the code. It is directly understood by the computer. All memory addresses have to be remembered. It is hard to amend or find errors in a program written in the machine language.

6.  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.

7. Advantages Disadvantages Assembly language is easier to understand and use as compared to machine language. Like machine language, it is also machine dependent/specific. It is easy to locate and correct errors. Since it is machine dependent, the programmer also needs to understand the hardware. It is easily modified.

8. 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.

9. Advantages Disadvantages High-level languages are user-friendly A high-level language has to be translated into the machine language by a translator, which takes up time They are easier to learn They are easier to 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

10. Types of High-Level Languages  1) Algebraic Formula-Type Processing  These languages are oriented towards the computational procedures for solving mathematical and statistical problems.  Examples include:  BASIC (Beginners All Purpose Symbolic Instruction Code)  FORTRAN (Formula Translation)  PL/I (Programming Language, Version 1)  ALGOL (Algorithmic Language)  APL (A Programming Language)  2. Business Data Processing  These languages are best able to maintain data processing procedures and problems involved in handling files. Some examples include:  COBOL (Common Business Oriented Language)  RPG (Report Program Generator)  3. String and List Processing  These are used for string manipulation, including search patterns and inserting and deleting characters. Examples are:  LISP (List Processing)  Prolog (Program in Logic)  4. Object-Oriented Programming Language  In OOP, the computer program is divided into objects. Examples are:  C++  Java