COMPUTER LANGUAGE

MACHINE LANGUAGE

Machine language is a language that a processor understands. Machine language consists of strings of binary numbers (0, 1). 1 stands for the presence of signals.
Any set of instruction in a machine language can be divided into the following four categories.

1. Arithmetic – Add, subtract, multiply, divide.
2. Control- load, store, jump.
3. Input output- Read and write.
4. Direct use- Start, halt and end.

Machine language instructions are represented by binary numbers i.e, sequencing consists of 0's &1's. For example, the binary sequence 001010011010 could represent a 12 bit machine language instruction. The instruction is divided into two parts, operation code (or op code) and an operand.

Op code	**operand**
10	10011010

This op code specifies the operations such as add, multiply, etc and the operand is the address of the data item that is to be operated upon.
Thus, while using a machine language, we have to remember the code numbers for the operations and also keep tract of the addresses of all the data items. Hence, machine language is highly complicated and subject to error. Also, the programs written in machine language are machine dependent. That has been developed for a particular machine can not be run on another machine.
The program written in machine language are also called object program.
For a particular computer, the following operation codes are used for different functions.

0001	Addition
0010	Subtraction
0011	Multiply
0100	Division
0101	Place numbers from central memory in accumulator.
0110	Place content of accumulator in memory.
0111	Perform input from accumulator A to output device.

ADVANTAGE

1. Machine language instruction is directly executed, as there is no compilation translation procedure involved.

2. Machine language makes efficient use of computers memory space.
3. Shorter execution time is required for machine language program.
4. This language is suitable for computers having limited memory.

Disadvantage

1. Machine language is a machine dependent language.
2. Instructions of this language are written in binary language. Since it is very difficult to remember the codes, this leads to error.
3. During modification, the address of al the instructions have to be changed. It becomes a difficult task.
4. Remembering the address of all the storage locations is a very difficult task.
5. High programming skill is required to do programming in machine language.

ASSEMBLY LANGUAGE

The program written in the symbolic language is called source program. This source program acts as an input to the assembler, which is loaded in the computer memory. Then the assembler performs the translation and generates the equivalent machine code which is called object code.

The symbolic language is called as low level language because it is a designed for a particular machine. It can not be developed without knowing the size of the control memory and the size of the location word.

The assembly language compromise between a high level language and a machine language developed in 1995. These languages permit the use of alpha numeric operation codes and addresses. The computer operating system automatically translates these symbols with the help of symbolic equivalence table in order to obtain the numeric code that it needs to execute the program. This is also called symbolic machine language.

MACHINE CODE ASSEMBLY LANGUAGE

Machine code	**Assembly language (mnemonic)**
0000	**LDA** – Load accumulator contents of specific address.
0001	**STA** - Store accumulate contents in specific address.
0010	**ADD** – ADD contents of specific address to the accumulator contents.
0011	Subtract contents of the specific address from the accumulator
0100	AND- Perform an AND operation as contents of accumulator.

0101	ORA- Perform an OR operation as contents of accumulator.
0110	JPU- Jump Unconditionally to specified address.
1110	HLT- Stop the program
1111	IOP- Perform specified output operation.
1101	NOT- Perform the NOT operation.
0111	JAZ- Jump to specific address if accumulator contents are zero.
1000	JAN- Jump to specific address if accumulator contents are not zero.
1011	SAI- Swap the contents of the accumulator with the contents of the index register

HIGH LEVEL LANGUAGE

1. They are easier to learn than symbolic language.
2. They required less time to write.
3. They are easier to maintain.
4. They required better documentation.
5. The programs written in such a language can be executed in any computer.
6. Four or five low level instructions are reduced to a simple high level language statement.

EXAMPLE

1. **BASIC (1965)**
   Beginners All purpose symbolic instruction code, Gemmy & Kuntz
2. FORTRAN 3. COBOL 4.PASCAL

High level language may be further subdivided into procedure oriented language, problem oriented language and interactive programming language.
EXAMPLE
(1)COBOL is a procedure oriented language which is used extensively in business application.

(2)FORTRAN

PROBLEM ORIENTED LANGUAGE

It is attempt to solve processing requirement several programming effort allowing the user focus on what results are designed rather than on the individual steps needed to get those results.
Example –RPG (Report program generator)

INTERATIVE PROGRAMING LANGUAGE

It allows the user to interact with program in conversational fashion.
EXAMPLE – CAD& CAM, BASIC PASCAL.

GENERAL PURPOSE LANGUAGE (e.g. basic & Pascal) is suitable for any application.

SPECIAL PURPOSE LANGUAGE (e. g. COBOL) is suitable for special application areas.