1.2 Programming Languages

Different category of programming languages

There are a lot of programming languages out there, for example Python, C, Scala, Java, Cobol, Pascal etc. Some of them are easier for humans to understand while others are more computer friendly.

At this point of time it is worth mentioning that computers can understand only 0 and 1, then how can they understand languages like python written in English? Before answering this question, it is imperative to categorize programming languages. We can classify programming languages into following three categories.

Low Level

These are programming languages which are closest to computers in term of easiness with which they are converted to 0 and 1. But they are more difficult for humans to read, understand and are less tractable to maintain the code. They can further be divided into two categories.

Machine Language

Before we dive to explore what machine language is, let us know how a program is being executed by a computer.

> A computer program is a set of instructions which can be executed by CPU of a computer sequentially (one by one) or in parallel. There are different CPU architectures like ARM, Intel's 8051 etc. and each architecture has its own instruction set, which includes data types, instructions, registers etc. These instruction sets are understood and executed by CPU in the form of a sequence of 0's and 1's. So, computer programs are converted into instruction sets and these instruction sets are then executed by CPU.

Machine language is Bit encoding (sequence of 0's and 1's) of instruction set e.g.

```
1110 0001 1010 0000 0011 0000 0000 1001
```

Above bit sequence is an ARM instruction to copy values from register 9 into register 3.

Assembly Language

They are the Symbolic encoding of the instruction set e.g. MOV R3, R9 This instruction copies a value

from register 9 into register 3.

They are more convenient to read and understand. A code written in Assembly language is first converted into the machine level language i.e. sequence of 0's and 1's and then they get executed by CPU.

For example, below code is written for Intel's 8086 architecture to compare two numbers and print whichever number is greater.

```
.8086

.MODEL SMALL

.DATA

CHAR DB " "

MESSAGE1 DB 0AH,0DH,'****FIRST IS GREATER****','$'

MESSAGE2 DB 0AH,0DH,'****SECOND IS GREATER****','$'

MESSAGE3 DB 0AH,0DH,'****BOTH ARE EQUAL****','$'


INPUT_M DB 0AH,0DH,'ENTER NUMBER',0DH,0AH,'$'

.CODE

MAIN PROC

MOV AX,@DATA

MOV DS,AX


MOV DX,OFFSET INPUT_M

MOV AH,09

INT 21H


MOV AH,01

INT 21H


MOV CHAR,AL

MOV DX,OFFSET INPUT_M


MOV AH,09

INT 21H
```

```
MOV AH,01

INT 21H


CMP CHAR,AL

JE EQUAL

JGE GREAT


MOV DX,OFFSET MESSAGE2

JMP PRINT

EQUAL: MOV DX,OFFSET MESSAGE3

JMP PRINT

GREAT: MOV DX,OFFSET MESSAGE1

PRINT: MOV AH,09

INT 21H

MOV AH,4CH

INT 21H

MAIN ENDP

END MAIN
```

Don’t worry if you cannot understand this code, I have put it here purposefully to give you a feeling of the amount of code needed just to do a simple task and how difficult it is to understand code written in a low level language.

Both, Machine language and Assembly language are dependent on the architecture of CPU, i.e., to write code in low level one needs to know the hardware structure like what are the registers, flags, and bus etc. This means that a program written, in a low-level language, for PC cannot run on mac and vice versa. So, there was a need for a programming language which was easy to port, understand and maintain. This necessity gave birth to high-level programming languages.

High-Level Language

They are written in simple English, they have their reserved vocabulary called keywords e.g. if, else, while, break etc. and associated grammar i.e. syntax, dictating how to use these keywords. For example, if we want to compare two numbers in Python then we can write a simple code like below.

```
VarA=10 #variable
```

```
VarB=15
if varA>varB:
print “varA is greater than varB”
elif varA<varB:
print “varB is greater than varA”
else :
print “varB and VarA are equal”
```

In above code highlighted words are key words of Python, while other words are either a string literal or variable (Soon we will learn what are string literal and variables).

Comparing code 1 and code 2 we can observe that high-level languages are easy to write and understand by humans. High-level languages are converted into binary form so that they can be executed by CPU. This conversion can be done either via a compiler or an interpreter.

Compiler: A compiler takes an entire program and converts it into optimized object code or machine code. This object code is then run on CPU.

Interpreter: executes source code line by line i.e. read one line execute it and then proceed to the next line. It will terminate the execution either when it reaches end of the code or if it finds an error.