C_02 ⇒ Low Level Vs High Level Languages

Levels of Programming Languages :-

→ Low Level (Machine Lang!- and Assembly Lang

→ High Level (C, C++, Java, ..etc

Low Level Languages :-

→ Machine Language

→ Assembly Language.

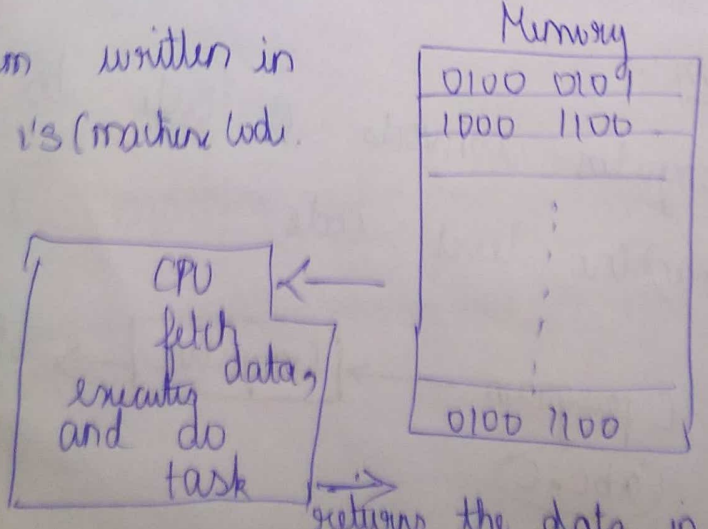
Machine Language ⇒ * Very close to Hardware, So it is called low level.

* For this we need to learn Architecture of CPU and configuration.

* Architectures of CPU are different for different machines.

* CPU is responsible to do the specific task.

Program written in 0 and 1's (machine code.

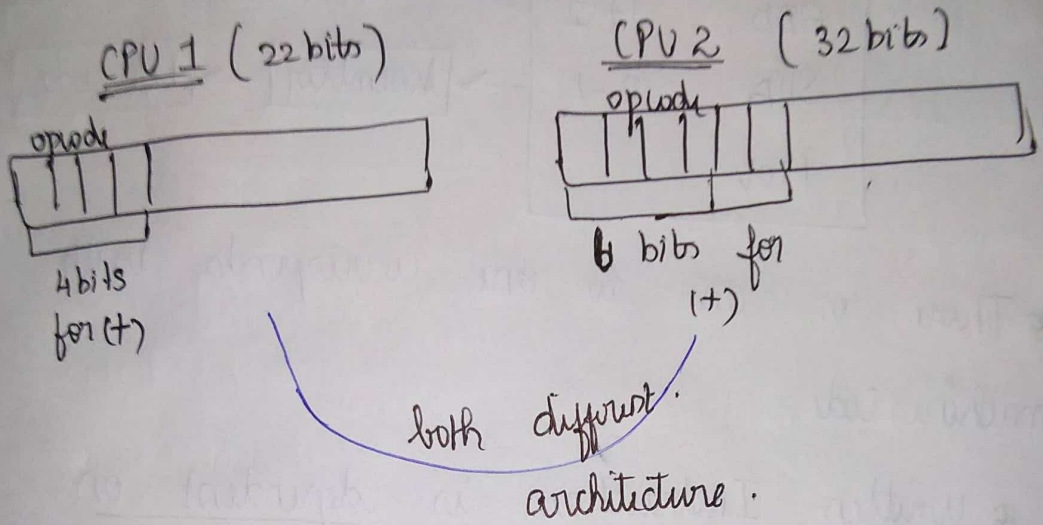


Eg:-

Hello World ⇒ converted to 0's and 1's

↳ 01001011 010001000 0010.....

* To learn this we should learn CPU Architectures



Drawbacks:-

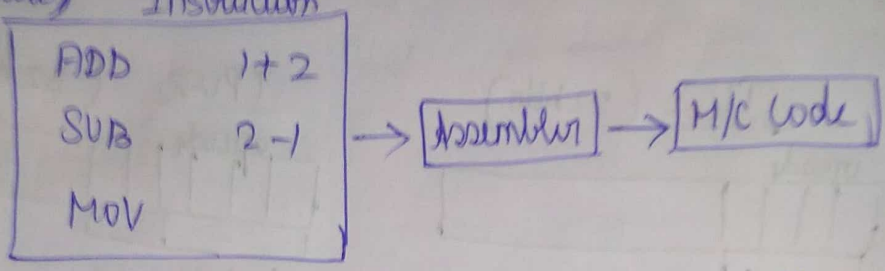
* Not understandable by humans.
* Computer can directly execute the program; because it is written in machine understandable code and it executes fast but the program may vary on set of CPU architectures.
* Same program cannot be executed in different machines.
* To overcome this drawback we have Assembly level Languages.

Assembly Language:

* It uses symbols and numbers called PNUMONICS (human readable instructions)

Eg: For addition, subtraction,

we write, Instruction



* There is one to one corresponds with machine code.

* Written Instruction is dependent on Computer Architecture and it is drawback.

* Here also programs cannot be executed on different machines

* Programs are not portable, so we overcome this drawback by using high level languages.

High Level Languages:

* C, C++, COBAL, FORTON, JAVA, Python, Perl, PHP, Ruby

* High Level Languages are not machine dependent architecture.

* It is close to humans, understandable by humans.

* Rather than dealing with registers, memory address, machine codes; High Level languages deals with Mathematical Notations, Variables, Keywords etc.

* C has lower level of abstraction; we need not want to learn low level or system details like specification of CPU, processor and computer system.

* For addition, subtraction we need not want to write ADD, SUB, MOV symbols. instead we write directly +, − ... etc.

Compiler and Interpreter:

* Programs written in high level lang:- are Converted to machine code

* Compiler takes complete source code and converts to object code; after converting the machine executes the code.

* Interpreter will not convert ~~compiler~~ complete code; instead it reads line by line and do parallel task by converting source code to object code and execution of code.

* Conversion & Execution is done parallelly by Interpreter.

Drawback:

* Pgms written in high level lng are converted to machine code which takes ~~taaaa~~ time

* But easy to read, write, portable (12) can run on different machine, independent of computer architectures.