UNIT-II

A machine language:→ It is the basic low-level Programming language designed to be recognized by a computer. Actually the language is written in a binary code of 0's and 1's that represent off and on States respectively.

A group of such digits is called an instruction and it is translated into a command that the Central Processing Unit or CPU understand.

eg Binary Program to Add two numbers.

Location/Address	Instruction Code
0	0010 0000 0000 0000
1	0001 0000 0000 0101
10	0011 0000 0000 0110
11	0111 0000 0000 0001
100	0000 0000 0101 0011
101	1111 1111 1110 1001
110	0000 0000 0000 0000

Program written for a computer may be in one of the following categories:

(i) **Binary Code :→** This is a sequence of instructions and operands in binary that list the exact representation of instruction as they appear in computer memory.

(ii) **Octal or hexadecimal Code:→** This is an equivalent translation of the binary code to octal or Hexa-decimal representation

(iii) **Symbolic Code :-** The user employs Symbols (letters, numerals, or special characters) for the operation part, the address part, and the other part of the instruction code. Each Symbolic instruction can be translated into one binary coded instruction

This translation is done by a special Program Called An Assembler. And this type of Symbolic Program is referred to as an Assembly language Program.

e.g Hexadecimal Program to Add two numbers

Location/Address	Instruction
000	2004
001	1005
002	3006
003	7001
004	0053
005	FFE9
006	0000

e.g Program with Symbolic operation codes,

Location/Address	Instruction	Comments
000	LDA 004	Load 1st operand in A
001	ADD 005	Add Second operand to A
002	STA 006	Store Sum in location 006
003	HLT	Halt computer
004	0053	1st operand
005	FFE9	Second operand
006	0000	Store Sum here

(iv) High-level Programming languages :→ These are Special languages developed to reflect the Procedures used in the Solution of a Problem rather than be concerned with the computer hardware behaviour. eg fortan

The Program that translates a high-level language Program to binary is called Compiler.

Assembly language :→ It is a low-level language because of the one-to-one relationship between Symbolic instruction and its binary equivalent. This uses Symbols (English language ~~to write~~ words) to write instructions, referred as mnemonics.

An Assembler :→ It is a special Software that Converts Assembly language Program into binary language Program.

Pseudo Instruction (Assembler Directive) :-

These are false (Pseudo) instructions that do not have equivalent binary form but Serve various functions. They do not refer to an operation that will be Performed by the Program during execution, rather it is a message to the assembler to help the Assembler in the Assembly Process.

e.g ORG N, N is the memory location where the first instruction of the Program must be Stored.

END, Specifies the end of the Program.

DEC N, N is a decimal number that needs to be Converted into binary by the Assembler.

HEX N, Hexadecimal number N to be converted into its binary equivalent.

Rules of the Assembly language :-

- Each line of Code must be divided into four fields i.e Label, instruction, operand & comment.

* Label field :- This is a one to three alphanumeric characters (Symbolic Address) that specifies the location of the instruction in memory. It should be terminated by a Comma (,) to enable the Assembler

recoganize it as a label. The first character should be an alphabet and the rest either alphabets or numerals

* Instruction field :→ This specifies a mnemonic (Pseudo Instructions

*. Comment field :→ Explains what each line of code does for easy understanding and explanation. Each Comment must be preceded by /. This helps the assembler recognize the beginning of a Program. It can be left empty.