Programming the Basic Computer
Lecture: 4
Assembly Language

Assembly Language:

A programming language is defined by a set of rules. Users must conform with all format rules of the language if they want their programs to be translated correctly. Almost every commercial computer has its own particular assembly language.

The rules for writing assembly language programs are documented and published in manuals which are usually available from the computer manufacturer.

The following are the requirements for an effective assembly language programming:

- Programming model of the processor,
- complete instruction set details of the processor,
- memory map and I/O map of the computer system,
- and complete details of the assembler including rules of the language.

Programming model specifies the program accessible registers.

The assembler converts the assembly language into machine language for execution.

The basic unit of an assembly language program is a line of code. The specific language is defined by a set of rules that specify the symbol that can be used and how they may be combined to form a line of code.

Rules of language:

Each line of an assembly language program is arranged in three columns called fields. The field specify the following information:

1. The label field may be empty or it may specify a symbolic address.
2. The instruction field specifies a machine instruction or a pseudoinstruction.
3. The comment field may be empty or it may include a comment.

Symbolic Address: A symbolic address consists of one, two or three but not more than three alphanumeric characters.

- The first character must be letter, the next two may be letters or numerals.
- The symbol can be chosen arbitrarily by the programmer.

- A symbolic address in the label field is terminated by a comma so that it will be recognized as a label by the assembler.

The instruction field in an assembly language program may specify one of the following items:

1. A memory-reference instruction (MRI)
2. A register-reference or input-output instruction (non-MRI).
3. A pseudoinstruction with or without an operand

A pseudo instruction is not a machine instruction but rather an instruction to the assembler giving information about some phase of the translation. e.g.

DEC N	Signed decimal number N to be converted to binary
ORG N	Hexadecimal number N is the memory location for the instruction or operand listed in the following line

The third field in a program is reserved for comments. A line of code may or may not have a comment, but if it has, it must be preceded by a slash for the assembler to recognize the beginning of a comment field.

- Assembly Language Program to Subtract Two Numbers

```
        ORG 100     /Origin of program is location 100
        LDA SUB     /Load subtrahend to AC
        CMA         /Complement AC
        INC         /Increment AC
        ADD MIN     /Add minuend to AC
        STA DIF     /Store difference
        HLT         /Halt computer
MIN,    DEC 83      /Minuend
SUB,    DEC -23     /Subtrahend
DIF,    HEX 0       /Difference stored here
        END         /End of symbolic program
```

Assembler:

An assembler is a program that accepts a symbolic language program and produces its binary machine language equivalent.

- The input symbolic program is called the source program and the resulting binary program is called the object program.

The assembler is a program that operates on character strings and produces an equivalent binary interpretation.

Prior to starting the assembly process, the symbolic program must be stored in memory. The user types the symbolic program on a terminal. A loader program is used to input the characters of the symbolic program into memory. Since the program consists of symbols, its representation in memory must use an alphanumeric character code.

Working of Assembler:

A two pass assembler scans the entire symbolic program twice:

- During the first pass:
  - It generates a table that correlates all user-defined address symbols with their binary equivalent value.
    - the binary translation is done during the second pass.
  - Keep track of location counter(to keep track of the location of instructions).
  - Process pseudo-operations.
- During the second pass:
  - Machine instructions are translated during the second pass by means of table lookup procedures.
  - Generate object code by converting symbolic code into respective machine code.
  - During this pass, assembler uses four tables:
    - Pseudo instruction table
    - MRI table
    - Non-MRI table
    - Address Symbol table