Assembly Language

- The development of assembly language was a major milestone in the evolution of computer technology. First step to the high-level language.

  High-level language program (C, Java, Python) →
  Compiler →
  Assembly language program (LOAD 8 RA) →
  Assembler →
  Binary machine language program (1111100001000000)

- Binary programs are tedious and very error-prone process.

- Assembly program:
  • use **symbolic name of each instruction**
  • use **symbolic address** (label)
  • **assembly language is hardware dependent**, with a different assembly language for each type of processor. (reference to specific registers, opcodes supported by the processor, bit length of registers and operand of machine languages)
  • Four elements of a statement in a typical assembly language

  ![Assembly Language Format](image)

  • The first element is the symbolic address; some lines have no symbolic address, implying that the address of that line is one more than the address of the previous line; If the operand(s) is(are) for memory-referencing instruction, it contains the symbolic address.

- Why use label?
  • **make a program location easier to find and remember**
  • **instructions can easily be moved to correct the program**. The assembler will automatically change the address in all instructions that use the label when the program is reassembled.
  • **programmers do not have to calculate relative or absolute memory address**, but just uses labels as needed.
Two-Pass Assembler

- First pass is to construct a symbol table that contains a list of all labels and their associated line number (address).
  - The labels are usually used for loop.
  - Examine the assembly program line by line, determine the length of the corresponding machine instructions, and therefore determine the line number for the labels.

- Second pass read the program again from the beginning. Each instruction is translated into the appropriate binary machine code. The translation includes:
  - Translate the opcode into binary
  - Use the opcode to determine the format of instruction and length of the various fields in the instruction
  - Translate each operand name into appropriate register or memory code.
  - Translate each immediate value into binary string
  - Translate any references to labels into the appropriate line number using symbol table