

Assembly Languages

What is an Assembly Language?

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

High-order language program (C, Java, Python) →

Compiler →

Binary machine language program (1111100001000000)

Assembly language program (LOAD 8 RA) →

Assembler →

Binary machine language program (1111100001000000)

- Binary programs (machine code) are tedious and very error-prone process.
- Assembly languages provide a user-friendly way to program.

- A sample assembly program: sum the first ten non-zero odd numbers

set up

MOVE 1 RA

MOVE 0 RB

MOVE 2 RC

MOVE 10 RD

loop

loop:

ADD RA RB RB

ADD RA RC RA

ADD -1 RD RD

BRZ breakout

BRA loop

breakout: #RB contains the sum

OPORT RB

- 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



- 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.