

Assembly Languages (II)

Two-Pass Assembler

- The two-pass assembler parses the input file containing the assembly program twice. Before parse the assembly program, the first step is to tokenize the program.
 - During the tokenization, the whitespace and comments in the program are discarded.
 - The outcome of tokenization is a list of lists. Each sublist contains the elements on a line of code. For example, [['move', '1', 'RA'], ['move', '0', 'RB'], ...] .
- 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 (aka the list of lists after tokenization), determine the length of the corresponding assembly program, and therefore determine the line number for the labels.
 - The outcome of the first pass is a symbol table that associate the line number with each label.
- Second pass read the program (the list of lists) again from the beginning. Each instruction is translated into the appropriate binary machine code. The translation includes:
 - Translate the mnemonic (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 the symbol table generated by the first pass.