

Two Pass Assembler for SIC/XE

This program is an assembler for SIC/XE. It consists of two passes.

WHAT DO EACH PASS DO?

Pass1 (define symbols):

1. Assign addresses to all statements in the program.
2. Save the values (addresses) assigned to all labels for use in Pass2.
3. Perform some processing of assembler directives. (This includes processing that affects address assignment, such as determining the length of data areas defined by BYTE, RESW, etc.)

Pass2 (assemble instructions and generate object program):

1. Assemble instructions (translating operation codes and looking up addresses).
2. Generate data values defined by BYTE, WORD, etc.
3. Perform processing of assembler directives not done during Pass 1.
4. Write the object program.

ALGORITHM (DEPTH 1)

Pass 1

Same with the Pass 1 for SIC, which is described in the text.

Pass 2

Variables:

- target address
- object code
- displacement

```
for line in intermediate-file
    search OPTAB for OPCODE
    if found then
        if there is a symbol in OPERAND field then
            if OPERAND starts with '#' then
                mark it is immediate addressing
                store the OPERAND as displacement
            else
                mark it is pc relative
                search SYMTAB for OPERAND
                store symbol value as target address
                calculate displacement
        else
            store 0 as displacement
            assemble the object code instruction
    else if OPCODE = 'WORD' then
        convert constant to object code
    else
        fill object code with zeros
        print address and object code.
```

ALGORITHM (DEPTH 2)

Assemble the object code instruction

Parameters

- `op_value` : the value corresponding opcode in OPTAB
- `is_immediate` : it is true if this line use immediate addressing
- `is_pc_relative` : it is true if this line use program counter relative addressing
- displacement

Variables

- part 1 (of object code)
 - opcode
 - flag bit – n, i
- part 2 (of object code)
 - flag bit – x, b, p, e
- part 3 (of object code)
 - displacement

if this line use immediate addressing **then**
 part 1 = `op_value` + 01(2)

else

 part 1 = `op_value` + 11(2)

if this line use pc relative addressing **then**
 part 2 = 0010(2)

else

 part 2 = 0000(2)

return part 1 + part 2 + displacement

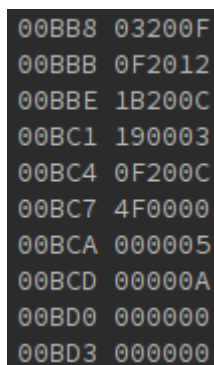
Calculate displacement

Parameters

- `target_address`
- program counter

return `target_address` – pc

SCREEN SHOT OF DISASSEMBLY



```
00BB8 03200F
00BBB 0F2012
00BBE 1B200C
00BC1 190003
00BC4 0F200C
00BC7 4F0000
00BCA 000005
00BCD 00000A
00BD0 000000
00BD3 000000
```