Assembler Pseudocode

2 pass assembler for SIC/XE - Overview

Pass 1:

```
BEGIN {construction of symbol table}
    Skip over initial comment lines

    Process the START statement, if present,
       setting Locctr and ENDval to the operand's value

    Loop through the source lines until the END
      statement is reached or the source file runs out
      BEGIN
        Skip over source lines that are comment lines

        Extract Label, Opcode, & Operand parts

        IF there is a Label
           add it to the symbol table if it is not
              already there (otherwise it's an error)

        Increment Locctr:
           1. If Opcode is a storage directive, for
                 BYTE, RESB, or RESW, Operand determines
                 increment; for WORD it is 3
           2. If Opcode is an instruction, the increment
                 is 1,2,3, or 4 as per Opcode bits
           3. Increment = 0 for assembler directives

      END {of loop}

    If present, process END statement, and reset ENDval
       to the END statement's operand value, if present

END {of Pass 1}
```

Pass 2:

```
BEGIN {generation of object module}

    Write assembler report headings & any leading
      comment lines (Note: as each source line is
      processed, it is written to the assembler report)
    Process the START statement, if present, setting
    Locctr to the operand's value (default is 0)
    Initialize the object module:
      1. Locctr value is initial load point
      2. ENDval from Pass 1 is tentative "execute next"

    Loop through the source lines until the END
      statement is reached or source runs out
      BEGIN
        Skip over any comment lines (but write them
         to the assembler report)
        Extract Opcode, & Operand, increment Locctr, then
        if Opcode is
           1. RESW or RESB, start a new module:
                a. ! delimiter to end prior module
                b. loader address replaces ENDval in
                      prior module as "execute next"
                c. Locctr value is next load point
                d. ENDval from Pass 1 as this module's
                      a tentative "execute next"
           2. WORD or BYTE, Operand gives the storage
              value(s) to write to the object module
           3. an assembler directive, process as spec'd
           4. an instruction, build the object version
              utilizing nixbpe bits, Locctr, and
              Operand value from the symbol table
      END {of loop}

    Append the ! delimiter to end the final module
    Output the object module(s) as the object code file
      if no errors  were encountered in Pass 1 or 2
END {of Pass 2}
```