

Pseudocode for Two-Pass Assembler

Disclaimers

1. The following pseudocode IS NOT COMPLETE. There are several holes you need to fill in for it to be correct. It is, however, a good overview of the algorithm for the assembler. You should understand it thoroughly.
2. This is *not* the only way to do things. There are many equally good (and some better!) solutions. But before developing your solutions you need to understand the task at hand.

Data Structures

- Machine Op Table
- Pseudo Op Table
- Location Counter
- Symbol Table
- Literal Table

Pass 1

```
begin
  read in first non-comment line of source
  if (not an .ORIG line)
    return error
  endif
  store segment information
  initialize Location Counter

  read in next line of input
  while (not an .END line)
    if (line is not a comment line)

      if (there is a label)
        add symbol to Symbol Table:
```

```

        - report an error if it's already there
        - otherwise set its value (using Location Counter or operand of EQU)
endif

if (instruction in Machine Op Table or Pseudo Op Table)
    increment Location Counter appropriately
else
    error (invalid operation)
endif

if (operand contains a literal)
    add literal to Literal Table:
        - avoid duplicate entries
        - set the name and value
endif

    write line to intermediate file
endif
read in next line of input
end while

set addresses of literals
use Location Counter to calculate program length
end

```

Pass 2

```

begin
    output header record to object file
    read in first line from intermediate file
    while (not done with intermediate file)

        if (there are symbols in the operands)
            replace symbols with their value from Symbol Table
            - report an error if not present
        endif
        if (there are literals in the operands)
            replace literal with its address from Literal Table
        endif
        assemble line
        output line (text record) to object file
        output line to listing file

        read in next line from intermediate file
    end while
end

```