CS520 FALL 2019 DUE DATE: OCT 16, 2019

PROGRAM 3

Description

The goal of this assignment is to write a two-pass assembler for the assembly language for the vm520 virtual machine.

Your work must be done in the context of our assembler "front-end" and conform to the interfaces I have defined for your code and our code to utilize to interact. The code base you are to work within is available in *~cs520/public/program3*.

You must implement the following three functions:

- *initAssemble*: This function will be called once at program start-up to allow you to initialize any data structures, such as the symbol table.
- *assemble*: This function is called on both the first pass and the second pass for every line of the input that contains either a label definition or an instruction (or both). The first parameter to this function is the label being defined, if there is one. The second parameter contains the opcode and operands for the instruction, if there is one. The details of how instructions are represented are contained in the file *defs.h*.
- *betweenPasses*: This function is called once, after the first pass completes and before the second pass begins. Its single parameter is the file pointer (FILE*) where you should write the output, the object file that will be written during the second pass. This function should return the count of the number of errors detected during the first pass. Our part of the code will not execute the second pass if errors were detected during the first pass. The goal for full credit on this assignment is to detect all errors during the first pass or the call to the *betweenPasses* function.

Stubs for these three functions are present in *assemble.c*. All your code for this assignment should be placed in this single file. **Do not edit any of the other distributed files!**

Understand that a vm520 object file is not an ASCII file. You will want to use *od* to examine output files to be sure they are correct. I recommend using the flag *-tx4* to show the file in hex as a sequence of 32-bit words, since an object file is a sequence of 32-bit words.

Points will be awarded for this assignment in the following manner:

- 50 points: Determining addresses for labels in a correct input file. Have the function *betweenPasses* write the defined labels and their addresses to stdout, one label per lane, with the label name being first on the line, followed by one space and the address printed in decimal. The labels should be printed in ascending order of label names. You should implement the symbol table using the hash table you constructed for Lab 5.
- 15 points: Detecting all input errors (except errors concerning the importing or exporting of symbols), during the first pass or in the *betweenPasses* function. Call the function *error* (in *message.c*) to report all errors*. This function will ensure a common format for all error messages, including reporting the line number of the error.
- 25 points: Generating the object file for correct input files.
- 10 points: Properly handling import and export directives, including detecting errors and generating entries in the insymbol and outsymbol sections of the output file.

Your program will be graded primarily by testing it for correct functionality. In addition, however, you may lose points if your program is not properly structured and documented. Decompose sub-problems appropriately into functions and do incremental testing. Leave your debugging output in your code, but disabled, when you do your final assignment submission.

You will only submit **assemble.c** and **symtab.c**. This, of course, means you should not edit any of the other distributed files.

*example of a .asm file with errors and the error messages printed by the assembler

```
#
# this file contains a few errors
#
top:
     load r0,r1  # bad format for opcode

     lood r0,x   # bad opcode

     ldimm r0, 0x1FFFFF # bad constant

     ldind r0, 0x1FFFFF(r0) # bad offset

top:  # duplicate label

     word r0   # bad format

     halt 100 # another bad format
```

```
[error] line 5:  opcode does not match the given operands
[error] line 7:  unknown opcode
[error] line 9:  constant 2097151 will not fit in 20 bits
[error] line 11:  offset 2097151 will not fit in 16 bits
[error] line 13:  label top already defined
[error] line 15:  opcode does not match the given operands
[error] line 17:  opcode does not match the given operands
[error] line 18:  assembler terminating after first pass with 7 error(s)
```