CPE 315
Spring 2019
Seng
Laboratory #2
Complete by Friday night 4/26/19 (11:59pm)

Updates:

- none

Objectives:

- To build a MIPS simulator front-end assembler/parser

Description:

For this lab, you will write a 2-pass assembler in Java. This assembler will load in MIPS assembly files and output the corresponding machine code (to the screen). The input to your assembler will be MIPS assembly files with comments, labels, and whitespace (spaces and tabs). Assume the following regarding the input:

1. **Comments** - Comments start with '#'. A comment will start at the beginning of a line or may appear after an instruction (on the same line). If a comment appears after an instruction, there may be whitespace before the '#' symbol.

2. **Labels** - Labels will only contain alphanumeric characters (0-9, a-z, and A-Z). There will be no other symbols in the labels. A colon ':' will appear after a label. There may be whitespace before a label and the label may be followed by an instruction. A label may appear on a line without an instruction.

3. Blank lines may contain a mixture of whitespace characters.

4. There may be some whitespace between operands.

5. Immediate values may be negative.

6. Your assembler must support the following instructions: and, or, add, addi, sll, sub, slt, beq, bne, lw, sw, j, jr, and jal.

7. You do NOT need to support the following registers: $at, $k0, $k1, $gp, $fp.

8. On the same line, whitespace may appear between a label and an instruction. There will always be whitespace between j/jal instructions and the target label.

9. Your program should error check for invalid instructions. This assembler will be used in future lab assignments, so you may wish to create data structures to store the instructions.

Implementation:

This assembler is called a 2-pass assembler because it takes 2 passes through the file in order to generate the machine code. The first pass should run through all the lines of the file to compute the address of each label. During the second pass, all the instructions are converted to machine code.

For this lab, your assembler will output machine code to the screen. In future labs, the machine code will be stored in data structures.

Here are some sample files which you can use as test input: test1.asm test2.asm test3.asm test4.asm

The corresponding output: test1.output test2.output test3.output test4.output

---

Assignment submission and Grading:

- I will grade this lab using automated scripts to test compilation and output correctness (I test with 'diff -w -B').
- Your lab must compile and run on unix1.csc.calpoly.edu.
- Code which does not compile will receive a score of 0.
- You must have a Makefile which should compile the lab assignment using a 'make lab2' command. Here is a sample Makefile
- Your code should run with the command: **java lab2 file.asm** (where file.asm is the input file)

Turning in the assignment

Handin electronically:

- Java source files (do not .zip files)
- Makefile

At the unix prompt, type 'handin jseng' to see what assignments I am accepting. In order to handin a file for Lab 2, type 'handin jseng 315_lab2_x files_to_handin'.