CDA 4205L Lab #3: ISA Assembler Design (Part 1)

University of South Florida

Lab Date: Jan. 29

Report due: One week after the lab section, see Canvas

***This is a group lab**. One submission per group is sufficient.*

Welcome to CDA 4205L Lab #3! This is Part 1 of 3. The purpose of Part 1 is to help you better understand some of the pre-processing steps used by an assembler. These steps will clean up the code, and make it easier to translate the code into machine code, which you will complete in Part 2 next week.

Prelab

An *assembler* is a program that converts low-level assembly code into machine code which can be understood by the target processor. It is one step in a larger toolchain that begins with a compiler and ends with a linker and object file generation (more on those later!). However, before the assembler can generate the machine code, it must prepare the assembly code by removing comments and transforming instructions to a standard format.

In this lab, you will design an assembler in Python programming language to pre-process an assembly code file. The provided assembly is written using a subset of the standard RISC-V ISA, with some small modifications to simplify the coding tasks. The assembly code will be provided to you. You will need to perform operations such as removing comments, removing blank lines, replacing labels in instructions with their corresponding address, replacing register names with their equivalent register values, etc.

Lab

1. Use the online version of JupyterLab which can be found here: https://jupyter.org/try-jupyter/lab/
2. Download 'Lab3_assembler_design.ipynb' file from Canvas (Files > Labs > Lab 3) and open it in JupyterLab.
3. Download 'example1.asm' and 'example2.asm' file from Canvas (Files > Labs > Lab 3) and upload it in the same directory as your 'Lab3_assembler_design.ipynb' in JupyterLab.
    a. 'example1.asm' is just for you to test and verify.
    b. **'example2.asm' is the code you should use in your report.**
4. Complete Tasks 1-9, as described in the 'Lab3_assembler_design.ipynb' file.

T1-9: Take screenshots of outputs from Tasks 1 - 9 using 'example2.asm' and include those in your report.

5. Example outputs from each task using ‘example1.asm’ *(only for reference)* -
    a. Input

```
# Implement a RISC-V program
start:
    lw   t2, 0(s1)
    addi a0, zero, 10 # store 10 in a0
    addi a1, zero, 20 # store 20 in a1
    bge  a0, a1, done
    add  a2, a1, zero
    jal  ra, exit
done:
    add  a2, a0, zero
    jal  ra, exit
exit:
    sw  a0, 0(s1)
```

    b. Output – Task 1

```
Assembly Instructions:

start:

    lw   t2, 0(s1)

    addi a0, zero, 10
    addi a1, zero, 20
    bge  a0, a1, done

    add  a2, a1, zero

    jal  ra, exit

done:

    add a2, a0, zero

    jal ra, exit

exit:

    sw  a0, 0(s1)
```

    c. Output – Task 2

```
Assembly Instructions:
[]
['start:']
['lw', 't2', '0', 's1']
['addi', 'a0', 'zero', '10']
['addi', 'a1', 'zero', '20']
['bge', 'a0', 'a1', 'done']
['add', 'a2', 'a1', 'zero']
['jal', 'ra', 'exit']
['done:']
['add', 'a2', 'a0', 'zero']
['jal', 'ra', 'exit']
['exit:']
['sw', 'a0', '0', 's1']
```

d. Output – Task 3

```
Assembly Instructions:
['start:']
['lw', 't2', '0', 's1']
['addi', 'a0', 'zero', '10']
['addi', 'a1', 'zero', '20']
['bge', 'a0', 'a1', 'done']
['add', 'a2', 'a1', 'zero']
['jal', 'ra', 'exit']
['done:']
['add', 'a2', 'a0', 'zero']
['jal', 'ra', 'exit']
['exit:']
['sw', 'a0', '0', 's1']
```

e. Output – Task 4

```
Assembly Instructions:
['start:']
['lw', 't2', 's1', '0']
['addi', 'a0', 'zero', '10']
['addi', 'a1', 'zero', '20']
['bge', 'a0', 'a1', 'done']
['add', 'a2', 'a1', 'zero']
['jal', 'ra', 'exit']
['done:']
['add', 'a2', 'a0', 'zero']
['jal', 'ra', 'exit']
['exit:']
['sw', 's1', 'a0', '0']
```

f. Output – Task 5

```
Assembly Instructions:
['lw', 't2', 's1', '0']
['addi', 'a0', 'zero', '10']
['addi', 'a1', 'zero', '20']
['bge', 'a0', 'a1', 'done']
['add', 'a2', 'a1', 'zero']
['jal', 'ra', 'exit']
['add', 'a2', 'a0', 'zero']
['jal', 'ra', 'exit']
['sw', 's1', 'a0', '0']
------------------------------
Assembly Labels:
LABEL | VALUE
start |     0
done  |    24
exit  |    32
```

g. Output – Task 6

```
Assembly Instructions:
['lw', 't2', 's1', 0]
['addi', 'a0', 'zero', 10]
['addi', 'a1', 'zero', 20]
['bge', 'a0', 'a1', 'done']
['add', 'a2', 'a1', 'zero']
['jal', 'ra', 'exit']
['add', 'a2', 'a0', 'zero']
['jal', 'ra', 'exit']
['sw', 's1', 'a0', 0]
------------------------------
Assembly Labels:
LABEL | VALUE
start |     0
done  |    24
exit  |    32
```

h. Output – Task 7

```
Assembly Instructions:
['lw', 't2', 's1', 0]
['addi', 'a0', 'zero', 10]
['addi', 'a1', 'zero', 20]
['bge', 'a0', 'a1', 12]
['add', 'a2', 'a1', 'zero']
['jal', 'ra', 12]
['add', 'a2', 'a0', 'zero']
['jal', 'ra', 4]
['sw', 's1', 'a0', 0]
------------------------------
Assembly Labels:
LABEL | VALUE
start |     0
done  |    24
exit  |    32
```

i. Output – Task 8

```
Assembly Instructions:
['lw', 7, 9, 0]
['addi', 10, 0, 10]
['addi', 11, 0, 20]
['bge', 10, 11, 12]
['add', 12, 11, 0]
['jal', 1, 12]
['add', 12, 10, 0]
['jal', 1, 4]
['sw', 9, 10, 0]
------------------------------
Assembly Labels:
LABEL | VALUE
start |     0
done  |    24
exit  |    32
```

j. Output – Task 9

```
Saved assembly code to: example1_out1.txt
```

Content inside "example1_output1.txt":

example1_out1.txt

```
1  lw 7 9 0
2  addi 10 0 10
3  addi 11 0 20
4  bge 10 11 12
5  add 12 11 0
6  jal 1 12
7  add 12 10 0
8  jal 1 4
9  sw 9 10 0
10
```

6. Answer the following questions.

T10: In your own words, explain the purpose of the 'replace_labels' function.

T11: In your own words, explain the purpose of 'replace_reg' function.

In-class work - Complete at least **T1-T5**, show them to the TA for review and submit them on Canvas before the lab session ends. Submit your in-class work as a Word document (.docx) format.

Final Report - Submit your report pdf with answers to all questions and screenshots. Also, submit the final jupyterlab file and the output '.txt' file for example2 only. Submit .zip file, one per group. Include these:

- the PDF report for lab 3
- the jupyter notebook (.ipynb) with your part 1 assembler
- "example2.asm" as we gave it to you
- "example2_out1.txt" generated by your code

No need to submit example1 or its .txt