Lab 1 - EECS 211

Winter 2023

Problem 0

In this problem, first, you need to follow the steps in the following tutorial: https://justinmeiners.github.io/lc3-vm/index.html. In the next assignments, we will edit this virtual processor, so ensure you can follow all the details in this tutorial.

After this, you need to download the LC3 assembler, which can be found on this website: http://highered.mheducation.com/sites/dl/free/0072467509/104652/lc3tools_v12.zip. You will use this virtual LC3 processor in later problems to compile/assemble and run the codes.

The LC-3 tools package is designed to work as either a personal or administrative installation on various flavors of Unix, including Windows NT-based systems with appropriate support (e.g., Cygwin). If you are a Windows user, I recommend you install Ubuntu 22.04.1 LTS for free from the Microsoft store.

Run the following commands to install the required packages:

- sudo apt-get update
- sudo apt-get install flex wish make libncurses5-dev libncursesw5-dev

Finally, follow the instructions in the README file (zip folder) to finalize the installation.

Problem 1

In this problem, you will compile/assemble code and run it on the virtual LC3 processor. You should find an executable called "lc3as" which is the assembler you should use to convert the assembly codes to binary.

For each of the following assembly codes, convert them into a binary code (.bin file), open the symbol table file (the .sym file) to check the memory location of the data and labels, and then use the virtual processor to run the binary code on the LC3 processor.

Listing 1: Assembly Code # 1

```
;; Set R3 to R1 ^ R2
;;     i.e.       OR(    AND(NOT(R1),R2),      AND(R1,NOT(R2)))
;;     i.e. NOT(AND(NOT(AND(NOT(R1),R2)),NOT(AND(R1,NOT(R2)))))
          .ORIG   x3000
xor       NOT     R1,R1
          AND     R3,R1,R2
          NOT     R1,R1
          NOT     R2,R2
          AND     R4,R1,R2
          NOT     R2,R2
          NOT     R3,R3
          NOT     R4,R4
          AND     R3,R3,R4
          NOT     R3,R3
          HALT
          .END
```

Listing 2: Assembly Code # 2

```
;; Reverse a string
        .ORIG    x3000
rev     LEA      R0,FILE      ;; R0 is beginning of string
        ADD      R1,R0,#-1
LOOP1   LDR      R3,R1,#1     ;; Note -- LDR "looks" at the word past R1
        BRz      DONE1
        ADD      R1,R1,#1
        BR       LOOP1

DONE1   NOT      R2,R0
        ADD      R2,R2,R1

;; R0 == address of first character of string
;; R1 == address of last character of string
;; R2 == size of string - 2  (Think about it....)
LOOP2   ADD      R2,R2,#0
        BRn      DONE2
        LDR      R3,R0,#0     ;; Swap
        LDR      R4,R1,#0
        STR      R4,R0,#0
        STR      R3,R1,#0
        ADD      R0,R0,#1     ;; move pointers
        ADD      R1,R1,#-1
        ADD      R2,R2,#-2    ;; decrease R2 by 2
        BR       LOOP2

DONE2   HALT

FILE    .STRINGZ "This is so much fun!"
        .END
```

Problem 2

Edit the virtual processor code to dump the memory and registers in a text file, every time the code hits a "HALT". That is, you need to debug the virtual processor code to know what happens when the virtual processor tries to execute the "HALT" assembly command, and add a new function there that dumps the current content of the memory and the registers into a text file. Below is a snapshot of a file that dumps the first 5 locations of the memory and the first 4 registers (for this assignment, you need to dump the entire memory and registers, this is just a small example):

```
M0: 542
M1: 12
M2: 0
M3: 11
M4: 345
R0: 12317
R1: 12316
R2: 65534
R3: 111
```

Listing 1: Toy example for constraint satisfaction

Make sure that your file follows exactly the same format above, since we are going to automatically grade the assignment by comparing your files to the correct answer.

Deliverable

You need to upload two files named "memory_dump_1" and "memory_dump_2" that contains the memory/registers for the two assembly codes above.