

# Assignment 0 - EECS 211

Winter 2022

## Problem 0

In this problem, you need to follow the steps in the following tutorial: <https://justinmeiners.github.io/lc3-vm/index.html>. In the next assignments, we are going to edit this virtual processor, so make sure you can follow all the details in this tutorial.

## Problem 1

In this problem, you are going to compile/assemble code and run it on the virtual LC3 processor. First, you need to download the LC3 assembler which can be found on this website: [http://highereducation.com/sites/dl/free/0072467509/104652/lc3tools\\_v12.zip](http://highereducation.com/sites/dl/free/0072467509/104652/lc3tools_v12.zip). Follow the instructions in the README file (found inside the zip file) to install the LC3-tools. After installation, 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

```
1  ;; Set R3 to R1 ^ R2
2  ;;   i.e.      OR(   AND(NOT(R1),R2),   AND(R1,NOT(R2)))
3  ;;   i.e. NOT(AND(NOT(AND(NOT(R1),R2)),NOT(AND(R1,NOT(R2)))))
4  .ORIG    x3000
5  xor     NOT    R1,R1
6         AND    R3,R1,R2
7         NOT    R1,R1
8         NOT    R2,R2
9         AND    R4,R1,R2
10        NOT    R2,R2
11        NOT    R3,R3
12        NOT    R4,R4
13        AND    R3,R3,R4
14        NOT    R3,R3
15        HALT
```

Listing 2: Assembly Code # 2

```

1  ;; Reverse a string
2  .ORIG    x3000
3  rev     LEA     R0,FILE      ;; R0 is beginning of string
4          ADD     R1,R0,#-1
5  LOOP1   LDR     R3,R1,#1     ;; Note -- LDR "looks" at the word past R1
6          BRz     DONE1
7          ADD     R1,R1,#1
8          BR      LOOP1
9
10 DONE1   NOT     R2,R0
11          ADD     R2,R2,R1
12
13 ;; R0 == address of first character of string
14 ;; R1 == address of last character of string
15 ;; R2 == size of string - 2 (Think about it....)
16 LOOP2   ADD     R2,R2,#0
17          BRn     DONE2
18          LDR     R3,R0,#0     ;; Swap
19          LDR     R4,R1,#0
20          STR     R4,R0,#0
21          STR     R3,R1,#0
22          ADD     R0,R0,#1     ;; move pointers
23          ADD     R1,R1,#-1
24          ADD     R2,R2,#-2     ;; decrease R2 by 2
25          BR      LOOP2
26
27 DONE2   HALT
28
29 FILE     .STRINGZ "This is so much fun!"
30          .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):

```
1 M0: 542
2 M1: 12
3 M2: 0
4 M3: 11
5 M4: 345
6 R0: 12317
7 R1: 12316
8 R2: 65534
9 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.