Homework 13

Homework 13 is due on Wednesday, May 13, at the start of the lecture. Remember to include your *Discussions section* (e.g. ED1) and follow the complete Homework submission guidelines.

Please ask all questions about this assignment during the office hours, or post them on piazza.

LC-3 Programming in assembly

1. Code analysis

Assume a sequence of positive numbers is stored in consecutive memory locations, starting at memory address x3000. The sequence terminates with the value #-1 (xFFFF).

1. What does this program do? We will not read more than 30 words.

```
                        .ORIG x5000
                        LEA                R0, MESSAGE
                        LD                R1, TABLE
LOOP                LDR                R2, R1, #0
                        NOT                R3, R2
                        BRz                FINISH
                        ADD                R1, R1, #1
                        AND                R2, R2, #1
                        BRz                LOOP
                        PUTS
                        BRnzp        LOOP
FINISH                HALT
MESSAGE                .STRINGZ "Found!"
TABLE                .FILL        x3000
                        .END
```

2. Write a symbol table for the code above. Your symbol table should be similar in nature to that produced by the LC-3 assembler: for each label that appears in the code, your table should list the label and associate the label with an address in LC-3 memory.   For an example, see P&P Section 7.3.3, pp. 186-187.

2. String comparison

The following program compares two ASCII character strings of the same length. One string starts in memory location x5000, the other starts in memory location x6000. The characters are stored in a sequential series of memory addresses, and the last such address contains an ASCII NUL x00 (used as sentinel). If the strings are the same, the program terminates with the value +1 in R0, otherwise the program ends with the value -1 in R0.

1. Insert the missing instructions in the code below. You do not need to submit the program, only the missing instructions, referring to their respective number.

```
                        .ORIG x4000
                        ___________________        ; Insert instruction a.i) here
                        LD                R1, STRING1
                        LD                R2, STRING2
NEXTCHAR        LDR                R3, R1, #0
                        LDR                R4, R2, #0
                        BRz                EQUAL
                        ___________________        ; Insert instruction a.ii) here
                        ___________________        ; Insert instruction a.iii) here
                        NOT                R3, R3
                        ADD                R3, R3, #1
                        ADD                R4, R4, R3
                        BRz                NEXTCHAR
                        ADD                R0, R0, #-1
                        ___________________        ; Insert instruction a.iv) here
EQUAL                ADD                R0, R0, #1
STOP                HALT
STRING1                .FILL        x5000
STRING2                .FILL        x6000
                        .END
```

2. Write a symbol table for the code above. Your symbol table should be similar in nature to that produced by the LC-3 assembler: for each label that appears in the code, your table should list the label and associate the label with an address in LC-3 memory. For an example, see P&P Section 7.3.3, pp. 186-187.

3. Logical left shift

The following program is intended to do a logical left-shift of register R1 five times, but it has a bug.

```
                                .ORIG x3000
                                AND               R0, R0, #0
                                ADD               R0, R0, #5
SHIFT                        BRz               DONE
                                ADD               R0, R0, #-1
                                ADD               R1, R1, R1
                                BR               SHIFT
DONE                        TRAP        x25
                                .END
```

1. Identify the error and explain how to fix it. For your convenience, all lines have been numbered. We will not read more than 30 words.
2. Write a symbol table for the code above before you tried to fix it. Your symbol table should be similar in nature to that produced by the LC-3 assembler: for each label that appears in the code, your table should list the label and associate the label with an address in LC-3 memory. For an example, see P&P Section 7.3.3, pp. 186-187.

4. Equality test

The following code checks if the value in memory address x3025 is equal to 20, and if so, it prints a message to screen. However, it has a bug.

```
                                .ORIG x3000
                                LDI               R1, ADDRESS
                                ADD               R1, R1, #-20
                                BRnp        FINISH
                                LEA               R0, MESSAGE
                                PUTS
FINISH                        HALT
MESSAGE                        .STRINGZ        "M[x3025] is equal to twenty"
ADDRESS                        .FILL        x3025
                                .END
```

1. Identify the error and explain how to fix it. For your convenience, all lines have been numbered. We will not read more than 30 words.
2. State in which pass (first or second) the assembler identifies the bug.