Department of Electrical and Computer Engineering

The University of Texas at Austin

EE 306, Fall 2021
Problem Set 4
Due: **November 1st**, before class
Yale N. Patt, Instructor
TAs: Sabee Grewal, Ali Fakhrzadehgan, Ying-Wei Wu, Michael Chen, Jason Math, Adeel Rehman

Instructions: You are encouraged to work on the problem set in groups and turn in one problem set for the entire group. **The problem sets are to be submitted on Gradescope**. Only one student should submit the problem set on behalf of the group, but everyone should create a gradescope account and be tagged on the homework.

1. (Adapted from 5.31) The following diagram shows a snapshot of the 8 registers of the LC-3 before and after the instruction at location x1000 is executed. Fill in the bits of the instruction at location x1000.

Register	**Before**	**After**
R0	x0000	x0000
R1	x1111	x1111
R2	x2222	x2222
R3	x3333	x3333
R4	x4444	x4444
R5	x5555	xFFF8
R6	x6666	x6666
R7	x7777	x7777

Memory Location	**Value**
x1000	0001 **101 000 1 11000**

2. [Updated 10/28] The memory locations x3000 to x3007 contain the values as shown in the table below. Assume the memory contents below are loaded into the simulator and the PC has been set to point to location x3000. Assume that a breakpoint has been placed to the left of the HALT instruction (i.e. at location x3006 which contains 1111 0000 0010 0101). Assume that before the program is run, each of the 8 registers has the value x0000 and the NZP bits are 010.

Memory Location	Value
x3000	0101000000100000
x3001	0001000000100101
x3002	0010001000000100
x3003	0001000000000000
x3004	0001001001111111
x3005	0000001111111101
x3006	1111000000100101
x3007	0000000000000100

a. In no more than 15 words, summarize what this program will do when the Run button is pushed in the simulator.

   Hint: What relationship is there between the value loaded from memory and the final value in R0 after the program has completed?

   5 is put in R0 and shifted left the value at location x3007 times

b. What are the contents of the PC, the 8 general purpose registers (R0-R7), and the N, Z, and P condition code registers after the program completes?

PC	x3006
R0	x0050
R1	x0000
R2	x0000
R3	x0000
R4	x0000
R5	x0000
R6	x0000
R7	x0000
N	0
Z	1
P	0

c. What is the total number of CPU clock cycles that this program will take to execute until it reaches the breakpoint?

[Updated 10/28] Note: You should refer to the state machine in the Handouts section of the website to determine how many cycles an instruction takes. Assume each state that accesses memory takes 5 cycles to complete and every other state takes 1 cycle to execute. (This is the same state machine we gave you during the exam.)

Memory Location	Value	Instruction	Cycles to Execute	Number of times Executed	Total Cycles
x3000	0101000000100000	AND	9	1	9
x3001	0001000000100101	ADD	9	1	9
x3002	0010001000000100	LD	15	1	15
x3003	0001000000000000	ADD	9	4	36
x3004	0001001001111111	ADD	9	4	36
x3005	0000001111111101	Branch	10 taken 9 not taken	3 taken 1 not taken	39

Total Cycle: 9 + 9 + 15 + 36 + 36 + 39 = 144

3. Below is a segment of LC-3 assembly language program.

```
        ADD R2, R1, #0
HERE    ADD R3, R2, #-1
        AND R3, R3, R2
        BRz END
        ADD R2, R2, #1
        BRnzp HERE
END     HALT
```

If the data in R1 is an unsigned integer larger than 1, what does the program do?

(Hint: what is the relationship between the resulting integer in R2 and the original integer in R1?)

The program finds out the smallest power of 2 which is larger than or equal to the unsigned integer in R1.

4. What does the following program do (in 15 words or fewer)? The PC is initially at x3000.

Memory Location	**Value**
x3000	0101 000 000 1 00000
x3001	0010 001 011111110
x3002	0000 010 000000100
x3003	0000 011 000000001
x3004	0001 000 000 1 00001
x3005	0001 001 001 000 001
x3006	0000 111 111111011
x3007	1111 0000 0010 0101

Counts the number of bits that are set to 1 in the word at x3100

5. Prior to executing the following program, memory locations x3100 through x4000 are initialized to random values, exactly one of which is negative. The following program finds the address of the negative value, and stores that address into memory location x3050. Two instructions are missing. Fill in the missing instructions to complete the program. The PC is initially at x3000.

Memory Location	Value
x3000	1110 000 011111111
x3001	**0110 001 000 000000**
x3002	**0000 100 000000010**
x3003	0001 000 000 1 00001
x3004	0000 111 111111100
x3005	0011 000 001001010
x3006	1111 0000 0010 0101

6. The LC-3 has just finished executing a large program. A careful examination of each clock cycle reveals that the number of executed store instructions (ST, STR, and STI) is greater than the number of executed load instructions (LD, LDR, and LDI). However, the number of memory write accesses is less than the number of memory read accesses, *excluding instruction fetches*. How can that be? Be sure to specify which instructions may account for the discrepancy.

   A large number of LDI instructions (two read accesses) and STI instructions (one read access and one write access) could account for this discrepancy.

7. (7.2) An LC-3 assembly language program contains the instruction:

```
ASCII       LD R1, ASCII
```

   The label ASCII corresponds to the address x4F08. If this instruction is executed during the running of the program, what will be contained in R1 immediately after the instruction is executed?

   Since the LD instruction is loading itself, the PC offset is #-1

   R1 = 0010 001 1 1111 1111

8. (Adapted from 7.10) The following program fragment has an error in it. Identify the error and explain how to fix it.

```
        ADD R3, R3, #30

        ST R3, A

        HALT

A       .BLKW 1
```

   Will this error be detected when this code is assembled or when this code is run on the LC-3?

   The immediate value is too large for `ADD R3, R3, #30.`

   We can split the instruction into two `ADD R3, R3, #15.`

9. (Adapted from 6.14) Consider the following assembly language program:

```
        AND R2, R2, #0
LOOP    ADD R1, R1, #-3
        BRn END
        ADD R2, R2, #1
        BRnzp LOOP
END     HALT
```

What are the possible initial values of R1 that cause the final value in R2 to be 3?

For R2 to contain the value 3, we must have looped the loop body 3 times. This means `BRn` must not have initiated a branch for 3 consecutive times. Therefore, `R1` wasn't negative after the instruction `ADD R1, R1, #-3` was executed 3 times and then was negative after the 4th execution of the instruction. That is:

$$R1 - (3 \times 3) \geq 0 \text{ and } R1 - (3 \times 4) < 0$$

Solving the inequalities yields, $9 \leq R1 < 12$. Since a register contains integers, R1 could have been 9, 10, or 11.

10. (Adapted from 7.16) Assume a sequence of nonnegative integers is stored in consecutive memory locations, one integer per memory location, starting at location x4000. Each integer has a value between 0 and 30,000 (decimal). The sequence terminates with the value -1 (i.e., xFFFF).

    a. Create the symbol table entries generated by the assembler when translating the following routine into machine code:

```
        .ORIG x3000
        AND R4, R4, #0
        AND R3, R3, #0
        LD R0, NUMBERS
LOOP    LDR R1, R0, #0
```

```
            NOT R2, R1
            BRz DONE
            AND R2, R1, #1
            BRz L1
            ADD R4, R4, #1
            BRnzp NEXT
L1          ADD R3, R3, #1
NEXT        ADD R0, R0, #1
            BRnzp LOOP
DONE        TRAP x25
NUMBERS     .FILL x4000
            .END
```

Label	**Address**
LOOP	x3003
L1	x300A
NEXT	x300B
DONE	x300D
NUMBER	x300E

b. What does the above program do?

The instruction `AND R2, R1, #1` performs a bit mask (`x0001`) to decide whether the least significant bit of the value is 0 or 1. The LSB of a number is used to determine whether the integer was even or odd. For example, numbers with a zero LSB are: 0000 (#0), 0010 (#2), 0100 (#4), 0110 (#6), which are all even.

Hence, R3 counts the amount of even numbers in the list and R4 counts the amount of odd numbers.

11. [Updated 10/30] (Adapted from 7.18) The following LC-3 program compares two character strings of the same length. The source strings are in the .STRINGZ form. The first string starts at memory location x4000, and the second string starts at memory location x4100. If the strings are the same, the program terminates with the value 1 in R5; otherwise the program terminates with the value 0 in R5. Insert one instruction each at (a), (b), and (c) that will complete the program. (*Note: The memory location immediately following each string contains x0000.)*

```
        .ORIG x3000
        LD R1, FIRST
        LD R2, SECOND
        AND R0, R0, #0
LOOP    LDR R3, R1, #0              ; (a)
        LDR R4, R2, #0
        BRz NEXT
        ADD R1, R1, #1
        ADD R2, R2, #1
        NOT R4, R4                  ; (b)
        ADD R4, R4, #1              ; (c)
```

```
            ADD R3, R3, R4
            BRz LOOP
            AND R5, R5, #0
            BRnzp DONE
NEXT        AND R5, R5, #0
            ADD R5, R5, #1
DONE        TRAP x25
FIRST       .FILL x4000
SECOND      .FILL x4100
            .END
```

12. The data at memory address x3500 is a bit vector with each bit representing whether a certain power plant in the area is generating electricity (bit = 1) or not (bit = 0). The program counts the number of power plants that generate electricity and stores the result at x3501. However, the program contains a mistake which prevents it from correctly counting the number of electricity generating (operational) power plants. Identify it and explain how to fix it.

```
            .ORIG x3000
            AND R0, R0, #0
            LD R1, NUMBITS
            LDI R2, VECTOR
            ADD R3, R0, #1
CHECK       AND R4, R2, R3
            BRz NOTOPER
            ADD R0, R0, #1
NOTOPER     ADD R3, R3, R3
            ADD R1, R1, #-1
```

```
                BRp CHECK
                LD R2, VECTOR
                STR R0, R2, #1
                TRAP x25
NUMBITS         .FILL #16
 VECTOR         .FILL x3500
                .END
```

`R2` contains the bit vector, and not the address at which the bit vector is contained. The instruction `LDI R2, VECTOR` loaded the value at `x3500` into `R2` since the value at the memory address labeled as `VECTOR` was used as the address from which to load. The store instruction `STR R0, R2, #1` uses the value of `R2` to evaluate an address. However, `R2` must be modified to contain an address for an `STR` instruction to work. Thus, `LD R2, VECTOR` is an additional required instruction.

13. The following program does not do anything useful. However, being an *electronic idiot*, the LC-3 will still execute it.

```
.ORIG x3000
LD R0, Addr1
LEA R1, Addr1
LDI R2, Addr1
LDR R3, R0, #-6
LDR R4, R1, #0
ADD R1, R1, #3
ST R2, #5
STR R1, R0, #3
STI R4, Addr4
HALT
```

```
Addr1     .FILL x300B

Addr2     .FILL x000A

Addr3     .BLKW 1

Addr4     .FILL x300D

Addr5     .FILL x300C

          .END
```

Without using the simulator, answer the following questions:

a. What will the values of registers R0 through R4 be after the LC-3 finishes executing the ADD instruction?

R0	**x300B**
R1	**x300D**
R2	**x000A**
R3	**x1263**
R4	**x300B**

b. What will the values of memory locations Addr1 through Addr5 be after the LC-3 finishes executing the HALT instruction?

Addr1	**x300B**
Addr2	**x000A**
Addr3	**x000A**
Addr4	**x300B**
Addr5	**x300D**