Homework 06

1.

- How many TRAP service routines can be implemented in the LC-3? Why?
- Why must a RET instruction be used to return from a TRAP routine? Why won't a BRnzp (unconditional BR) instruction work instead?
- How many accesses to memory are made during the processing of a TRAP instruction?

2. What are the defining characteristics of a stack? Give two implementations of a stack and describe their differences.

3. The input stream of a stack is a list of all the elements we pushed onto the stack, in the order that we pushed them. The output stream is a list of all the elements that are popped off the stack in the order that they are popped off.
a. If the input stream is ZYXWVUTSR, create a sequence of pushes and pops such that the output stream is YXVUWZSRT.
b. If the input stream is ZYXW, how many different output streams can be created.

4. Rewrite the PUSH and POP routines such that the stack on which they operate holds elements that take up two memory locations each. Assume we are writing a program to simulate a stack machine that manipulates 32-bit integers with the LC-3. We would need PUSH and POP routines that operate with a stack that holds elements which take up two memory locations each. Rewrite the PUSH and POP routines for this to be possible.

5.Figure out what the following program does.

```
        .ORIG X3000
        LEA R2, C
        LDR R1, R2, #0
        LDI R6, C
        LDR R5, R1, #-3
        ST R5, C
        LDR R5, R1, #-4
        LDR R0, R2, #1
        JSRR R5
        AND R3, R3, #0
        ADD R3, R3, #7
        LEA R4, B
A       STR R4, R1, #0
        ADD R4, R4, #2
        ADD R1, R1, #1
        ADD R3, R3, #-1
        BRP A
        HALT
B       ADD R2, R2, #1
        LDR R0, R2, #0
        JSRR R5
        TRAP X29
        ADD R2, R2, #15
        ADD R0, R2, #3
        LD R5, C
        TRAP X2B
        ADD R2, R2, #5
        LDR R0, R2, #0
        JSRR R5
        TRAP X27
        JSRR R5
        JSRR R6
C       .FILL X25
        .STRINGZ "EE306 and tests are awesome"
        .END
```

6. Jane Computer (Bob's adoring wife), not to be outdone by her husband, decided to rewrite the TRAP x22 handler at a different place in memory. Consider her implementation below. If a user writes a program that uses this TRAP handler to output an array of characters, how many times is the ADD instruction at the location with label A executed? Assume that the user only calls this "new" TRAP x22 once. What is wrong with this TRAP handler? Now add the necessary instructions so the TRAP handler executes properly.

Hint: RET uses R7 as linkage back to the caller (RET is equivalent to JMP R7).

```
; TRAP handler
; Outputs ASCII characters stored in consecutive memory locations.
; R0 points to the first ASCII character before the new TRAP x22 is called.
; The null character (x00) provides a sentinel that terminates the output sequence.

        .ORIG x020F
START   LDR R1, R0, #0
        BRz DONE
        ST R0, SAVER0
        ADD R0, R1, #0
        TRAP x21
        LD R0, SAVER0
A       ADD R0, R0, #1
        BRnzp START
DONE    RET
```

```
SAVER0    .BLKW #1
          .END
```

7. A zero-address machine is a stack-based machine where all operations are done by using values stored on the operand stack. For this problem, you may assume that the ISA allows the following operations:

PUSH M - pushes the value stored at memory location M onto the operand stack.

POP M - pops the operand stack and stores the value into memory location M.

OP - Pops two values off the operand stack and performs the binary operation OP on the two values. The result is pushed back onto the operand stack.

Note: OP can be ADD, SUB, MUL, or DIV for parts a and b of this problem.

a. Draw a picture of the stack after each of the instructions below are executed. What is the minimum number of memory locations that have to be used on the stack for the purposes of this program? Also write an arithmetic equation expressing u in terms of v, w, x, y, and z. The values u, v, w, x, y, and z are stored in memory locations U, V, W, X, Y, and Z.

```
PUSH V
PUSH W
PUSH X
PUSH Y
MUL
ADD
PUSH Z
SUB
DIV
POP U
```

b. Write the assembly language code for a zero-address machine (using the same type of instructions from part a) for calculating the expression below. The values a, b, c, d, and e are stored in memory locations A, B, C, D, and E.

e = ((a * ((b - c) + d))/(a + c))