Homework 5

T1

What is the purpose of the `.END` pseudo-op? How does it differ from the `HALT` instruction?

T2

What are the definitions of a *queue*?

T3

The following program has an error in it. What is the error? How would you fix it?

```
    .ORIG x3000
A   .FILL xDEAD
B   .FILL xBEEF
    LD R0, A
    ST R0, B
    HALT
    .END
```

T4

Suppose you write two separate assembly language modules that you expect to be combined by the linker. Each module uses the label `AGAIN`, and neither module contains the pseudo-op `.EXTERNAL AGAIN`. Is there a problem using the label `AGAIN` in both modules? Why or why not?

T5

Your friend has just written a simple program intended to calculate complements, which is as follows:

```
    .ORIG x3000
    ; Simple program that should calculate
    ; complement of DATA and store the result back
    LD R2, DATA
    NOT R2, R2
    ADD R2, R2, #1
    ST R2, DATA
DATA .FILL xF001
    .END
```

However, it does not seem to be reliable for some reason...

Questions:

1. What's the 2's complement of `xF001` in hex?
2. Will the program store the complement to `DATA`?
3. What will happen afterwards? Why?

> Open questions (Answer if you like, but it **WILL NOT** be graded): What's the root cause of this phenomenon? How can we prevent this from happening?

T6

What's the difference between pseudo-ops `.FILL`, `.BLKW` and `.STRINGZ` in LC3?

T7

It is often useful to find the midpoint between two values. **For this problem, assume A and B are both even numbers, and A is less than B.** For example, if A = 2 and B = 8, the midpoint is 5. The following program finds the midpoint of two even numbers A and B by continually incrementing the smaller number and decrementing the larger number. You can assume that `A` and `B` have been loaded with values before this program starts execution.

Your job: Insert the missing instructions.

```
    .ORIG x3000
    LD R0, A
    LD R1, B
X   _________________ (a)
    _________________ (b)
    ADD R2, R2, R1
    _________________ (c)
    ADD R1, R1, #-1
    _________________ (d)
    BRnzp X
DONE ST R1,C
    TRAP x25
A   .BLKW 1
B   .BLKW 1
C   .BLKW 1
    .END
```

T8

We all know that we can achieve left-shift by adding the number to itself. For example, `ADD R0, R0, R0` will left-shift `R0` by 1 bit. However, **right-shift** is not that easy. Complete the following LC3 program so that it will right-shift `R0` by 1 bit. Note that some comments have been deleted.

```
    .ORIG x3000
    ; Suppose R0 is already loaded with the target number
    ; Initialize
    AND R1, R1, #0      ; Result
    ADD R2, R1, #15     ; Loop var i
    ADD R3, R1, #__ (a) ; 1 << (**DELETED**)
    ADD R4, R1, #1      ; 1 << (15 - i)
    AND R5, R5, #0      ; Temp result
    ; Main Loop
L   AND R5, R3, R0      ; Test bit
    BR___ (b) N         ; **DELETED**
    ADD R1, R1, R4      ; Add to result
N   ADD R3, __, __ (c)  ; **DELETED**
    ADD R4, R4, R4      ; L-shift R4
    ADD __, __, __ (d)  ; **DELETED**
    BRp L
    ; End
    HALT
    .END
```

T9

The following operations are performed on a stack:

```
PUSH A
PUSH B
POP
PUSH C
POP
PUSH D
PUSH E
PUSH F
POP
PUSH G
POP
POP
POP
PUSH H
```

1. What dose the stack contain after the `PUSH H` ?
2. At which point does the stack contain the most element?

Without removing the element left in the stack from the previous operations, we change this stack to a queue (the front of queue is the top of stack), and perform

```
ENQUEUE I
DEQUEUE
ENQUEUE J
ENQUEUE K
DEQUEUE
ENQUEUE L
DEQUEUE
DEQUEUE
DEQUEUE
DEQUEUE
ENQUEUE M
DEQUEUE
```

3. What does the stack contain now?

T10

Write a function that implements another stack function, `PEEK`. `PEEK` returns the value of the top element of the stack without removing the element from the stack. The return value is stored in `R0`, so you don't need to save `R0`. `PEEK` should also do underflow error checking: if an underflow occurs, you should output the string "Stack underflow error" and halt. (Suppose the pointer of top of the stack is in `R6`, and the stack can only take up the memory space from `x3FFF` to `x3FF0`)