HW6

T1

The following operations are performed on a **stack**:

PUSH A, PUSH B, POP, PUSH C, PUSH D, POP, PUSH E, POP, POP, PUSH F

(a) What does the stack contain after the PUSH F?
(b) At which point does the stack contain the most elements?
(c) In addition to the stack data structure, we also know a data structure called a **queue**, which ensures that the first element enqueued is the first to be dequeued (FIFO—First In, First Out). If we replace the stack in this problem with a queue, re-answer the above questions.

T2

Two students wrote interrupt service routines for an assignment. Both service routines did exactly the same work, but the first student accidentally used `RET` at the end of his routine, while the second student correctly used `RTI` .There are three errors that arose in the first student' s program due to his mistake. Describe any two of them.

T3

(a) What problem could occur if a program does not check the ready bit of the `KBSR` before reading the `KBDR`?
(b) What problem is likely to occur if the keyborad hardware does not check the `KBSR` before writing to the `KBDR`?
(c) What problem could occur if the display hardware does not check the `DSR` before writing to the `DDR`?

T4

Assemble the following LC-3 assembly language program

```
1           .ORIG x3000
2           AND R0, R0, #0
3           ADD R2, R0, #10
4           LD  R1, MASK
5           LD  R3, PTR1
6   LOOP    LDR R4, R3, #0
7           AND R4, R4, R1
8           BRz NEXT
9           ADD R0, R0, #1
10  NEXT    ADD R3, R3, #1
```

```
11          ADD R2, R2, #-1
12          BRp LOOP
13          STI R0, PTR2
14          HALT
15  MASK    .FILL x8000
16  PTR1    .FILL x4000
17  PTR2    .FILL x5000
18          .END
```

What does the program do?

T5

The following code implements the function of reversing a string. Please complete the missing parts.

```
1               .ORIG x3000
2               LEA   R6, STACKBASE
3               LEA   R0, PROMPT
4               TRAP  x22
5               AND   R1, R1, #0
6   LOOP        TRAP  x20
7               TRAP  x21
8               ADD   R3, R0, #-10
9               BRz   INPUTDONE
10              JSR   PUSH
11              ADD   R1, R1, #1
12              BRnzp LOOP
13  INPUTDONE   ADD   R1, R1, #0
14              BRz   DONE
15  LOOP2       ---
16              TRAP  x21
17              ADD   R1, R1, #-1
18              BRp   LOOP2
19  DONE        TRAP  x25
20
21  PUSH        ---
22              ---
23              ---
24  POP         LDR   R0, R6, #-2
25              ADD   R6, R6, #2
26              RET
27  PROMPT      .STRINGZ "Please enter a sentence:"
28              STACKSPAC .BLKW #50
29              STACKBASE .FILL #0
30              .END
```

(a) Complete the code for the subroutine `PUSH`

(b) What instruction should be filled in line 15 ? Also, explain the role of the R1 register in the code.

T6

In this question, **we modify how `JSR` / `JSRR` and `RET` works in LC3.**
`JSR` / `JSRR` will use the stack as the linkage instead of R7 by pushing the return PC onto the stack. Similarly, `RET` will load the PC with the value popped from the top of the stack.

In memory, there are 10 subroutines to print the 10 digits. For example, the subroutine to print the number "5" consists of the following instructions:

```
Print5      LD R0 Lable5
            OUT
            RET
Lable5      .FILL x35
```

Since each subroutine requires 4 memory locations, the starting address of each subroutine is always 4 greater than the starting address of the previous subroutine. The starting address of the subroutine Print0 is x5000. Therefore, the starting address of the subroutine Print1 is x5004, the starting address of the subroutine Print2 is x5008, and so on.

A student wishes print number "306" with a program that does not use any `JSR` / `JSRR` instructions nor I/O trap service routines in the main program.

Your Job: Fill in the blanks in the main program, so that executing this program will result in printing the digits "306" on the console, and then halt the machine.

Hint: What happens when a `RET` is executed without a JSR being executed beforehand ?

```
          .ORIG x3000
          LD  R6, ______
          LD  R0, ______
          ADD __________
          STR R0, R6, #0
          LD  R0, ______
          ADD __________
          STR R0, R6, #0
          LD  R0, ______
          ADD __________
          STR R0, R6, #0
          LD  R0, ______
          ADD __________
          STR R0, R6, #0
          RET
  ____________________
  ____________________
  ____________________
  ____________________
          HALT
  SP_INIT .FILL xFE00
          .END
```

T7

The following program copies a specified number(SIZE) of memory values from a source(SRC) to a destination(DST).

```
1           .ORIG x300
2           LD  R2, DST
3           LD  R1, SRC
4           LD  R0, SIZE
5   LOOP    BRz EXIT
6           ADD R0, R0, #-1
7           ADD R4, R1, R0
8           LDR R5, R4, #0
9           ADD R4, R2, R0
10          STR R5, R4, #0
11          ADD R0, R0, #0
12          BR LOOP
13  EXIT    HALT
14  SRC     .FILL x4000
15  DST     .FILL x5000
16  SIZE    .FILL x10
17          .END
```

Do not count the HALT instruction when answering the questions.

(a) How many instruction are processed during the runtime period ?

(b) Assuming one memory access takes 10 clock cycles, how many clock cycles will the program take ?