HW6

T1

The main program below calls a subroutine `F` . The `F` subroutine uses R3 and R4 as input, and produces an output which is placed in R0. The subroutine modifies registers R0, R3, R4, R5, and R6 in order to complete its task. `F` calls two other subroutines, `SaveRegisters` and `RestoreRegisters` , that are intended handle the saving and restoring of the modified registers (although we will see in question (b) that this may not be the best idea!)

```
1   ; Main Program;
2   .ORIG x3000
3   ...
4   ...
5   JSR F
6   ...
7   ...
8   HALT
9   ; R3 and R4 are input.
10  ; Modifies R0, R3, R4, R5, and R6
11  ; R0 is the output
12  ;
13  F
14  JSR SaveRegisters
15  ...
16  ...
17  ...
18  JSR RestoreRegisters
19  RET
20  .END
```

(a) Write the two subroutines `SaveRegisters` and `RestoreRegisters` .
(b) When we run the code we notice there is an infinite loop. Why? What small change can we make to our program to correct this error. Please specify both the correction and the subroutine that is being corrected.

T2

Memory locations x5000 to x5FFF contain 2's complement integers. What does the following program do?

```
1           .ORIG x3000
2           LD R1, ARRAY
3           LD R2, LENGTH
4           AND R3, R3, #0
5   AGAIN   LDR R0, R1, #0
6           AND R0, R0, #1
7           BRz SKIP
8           ADD R3, R3, #1
9   SKIP    ADD R1, R1, #1
10          ADD R2, R2, #-1
```

```
11          BRp AGAIN
12          HALT
13          ARRAY .FILL x5000
14          LENGTH .FILL x1000
15          .END
```

T3

Our code to compute n factorial worked for all positive integers n. Augment the iterative solution to `FACT` to also work for 0!

```
1   FACT    ST R1,SAVE_R1
2           ADD R1,R0,#0
3           ADD R0,R0,#-1
4           BRz DONE
5   AGAIN   MUL R1,R1,R0
6           ADD R0,R0,#-1
7           BRnp AGAIN
8   DONE    ADD R0,R1,#0
9           LD R1,SAVE_R1
10          RET
11  SAVE_R1 .BLKW 1
```

T4

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?

Without removing the elements left on the stack from the previous operations, we perform:

PUSH G, PUSH H, PUSH I, PUSH J, POP, PUSH K, POP, POP, POP, PUSH L, POP, POP, PUSH M

(c) What does the stack contain now?

T5

(a) What problem could occur if a program does not check the Ready bit of the KBSR before reading the KBDR?

(b) What problem could occur if the keyboard hardware does not check the KBSR before writing to the KBDR?

(c) Which of the above two problems is more likely to occur? Give your reason.

T6

Some computer engineering students decided to revise the LC-3 for their senior project.

In designing the LC-4, they decided to conserve on device registers by combining the KBSR and the DSR into one status register: the IOSR (the input/output status register). IOSR[15] is the keyboard device ready bit and IOSR[14] is the display device ready bit.

What are the implications for programs wishing to do I/O? Is this a poor design decision?

T7

The following LC-3 program is assembled and then executed. There are no assemble time or run-time errors.

What is the output of this program? Assume all registers are initialized to 0 before the program executes.

```
1           .ORIG x3000
2           LEA R0, LABEL
3           STR R1, R0, #4
4           TRAP x22
5           TRAP x25
6   LABEL   .STRINGZ "FUNKY"
7   LABEL2  .STRINGZ "HELLO WORLD"
8           .END
```

T8

The program below, when complete, should print the following to the monitor:

```
1           ABCFGH
```

Insert instructions at (a)–(d) that will complete the program.

```
1           .ORIG x3000
2           LEA R1, TESTOUT
3   BACK_1  LDR R0, R1, #0
4           BRz NEXT_1
5           TRAP x21
6           ------------ (a)
7           BRnzp BACK_1
8           ;
9   NEXT_1  LEA R1, TESTOUT
10  BACK_2  LDR R0, R1, #0
11          BRz NEXT_2
12          JSR SUB_1
13          ADD R1, R1, #1
14          BRnzp BACK_2
15          ;
16  NEXT_2  ------------ (b)
17          ;
18  SUB_1   ------------ (c)
19  K       LDI R2, DSR
20          ------------ (d)
```

```
21          STI R0, DDR
22          RET
23  DSR     .FILL xFE04
24  DDR     .FILL xFE06
25  TESTOUT .STRINGZ "ABC"
26    .     END
```

T9

Interrupt-driven I/O:

(a) What does the following LC-3 program do?

```
1           .ORIG x3000
2           LD R3, A
3           STI R3, KBSR
4           AGAIN LD R0, B
5           TRAP x21
6           BRnzp AGAIN
7   A       .FILL x4000
8   B       .FILL x0032
9           KBSR .FILL xFE00
10          .END
```

(b) If someone strikes a key, the program will be interrupted and the keyboard interrupt service routine will be executed as shown below. What does the keyboard interrupt service routine do?

```
1           .ORIG x1000
2           LDI R0, KBDR
3           TRAP x21
4           TRAP x21
5           RTI
6   KBDR    .FILL xFE02
7           .END
```

(c) Finally, suppose the program of part a started executing, and someone sitting at the keyboard struck a key. What would you see on the screen?

(d) In part c, how many times is the digit typed shown on the screen? Why is the correct answer: "I cannot say for sure."

T10

What does the following LC-3 program do?

```
1               .ORIG x3000
2               LEA R6, STACKBASE
3               LEA R0, PROMPT
4               TRAP x22 ; PUTS
```

```
5               AND R1, R1, #0
6   LOOP        TRAP x20 ; IN
7               TRAP x21
8               ADD R3, R0, #-10 ; Check for newline
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              JSR POP
17              TRAP x21
18              ADD R1, R1, #-1
19              BRp LOOP2
20  DONE        TRAP x25 ; HALT
21
22  PUSH        ADD R6, R6, #-2
23              STR R0, R6, #0
24              RET
25  POP         LDR R0, R6, #0
26              ADD R6, R6, #2
27              RET
28  PROMPT      .STRINGZ "Please enter a sentence:"
29              STACKSPAC .BLKW #50
30              STACKBASE .FILL #0
31             .END
```