

ICS 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 JSR F
5 ...
6 HALT
7 ; R3 and R4 are input.
8 ; Modifies R0, R3, R4, R5, and R6
9 ; R0 is the output
10 F
11 JSR SaveRegisters
12 ...
13 JSR RestoreRegisters
14 RET
15 .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.

h5 (a)

```
1 SaveRegisters
2 ST R5, SaveR5
3 ST R6, SaveR6
4 RET
```

```
1 RestoreRegisters
2 LD R5, SaveR5
3 LD R6, SaveR6
4 RET
```

h5 (b)

JSR 套 JSR，原 R7 的存储的返回地址会被新的覆盖掉，无法正确返回；

在 `JSR SaveRegisters` 之前加入语句 `ST R7, SaveR7`，保存返回地址，在 `RET` 之前加入语句 `LD R7, SaveR7`，恢复返回地址；

```
1 F
2 ST R7, SaveR7
3 JSR SaveRegisters
4 ...
5 JSR RestoreRegisters
6 LD R7, SaveR7
7 RET
```

T2

“

Assume that you have the following table in your program:

```
1 MASKS
2 .FILL x0001
3 .FILL x0002
4 .FILL x0004
5 .FILL x0008
6 .FILL x0010
7 .FILL x0020
8 .FILL x0040
9 .FILL x0080
10 .FILL x0100
11 .FILL x0200
12 .FILL x0400
13 .FILL x0800
14 .FILL x1000
15 .FILL x2000
16 .FILL x4000
17 .FILL x8000
```

(a) Write a subroutine `CLEAR` in LC-3 assembly language that clears a bit in R0 using the table above. The index of the bit to clear is specified in R1. R0 and R1 are inputs to the subroutine.

(b) Write a similar subroutine `SET` that sets the specified bit instead of clearing it.

h5 (a)

在表中找除了目标位置为 1 外，其余位置都为 0 的数，NOT 取反，再 AND R0

```
1 CLEAR
2 ST R2, SaveR2
3 LEA R2, MASKS
4 AGAIN ADD R2, R2, #1
5 ADD R1, R1, #-1
6 BRp AGAIN
7 LDR R2, R2, #0
8 NOT R2, R2
9 AND R0, R0, R2
10 LD R2, SaveR2
11 RET
```

h5 (b)

先 **CLEAR**，再加上除了目标位置为 1 外，其余位置都为 0 的数

```
1 SET
2 ST R7, SaveR7
3 JSR CLEAR
4 LEA R2, MASKS
5 AGAIN ADD R2, R2, #1
6 ADD R1, R1, #-1
7 BRp AGAIN
8 LDR R2, R2, #0
9 ADD R0, R0, R2
10 LD R7, SaveR7
11 RET
```

T3

“

The following program needs to be assembled and stored in LC-3 memory

```
1 .ORIG x4000
2 AND R0,R0,#0
3 ADD R1,R0,#0
4 ADD R0,R0,#4
5 LD R2,B
6 A LDR R3,R2,#0
7 ADD R1,R1,R3
8 ADD R2,R2,#1
9 ADD R0,R0,#-1
10 BRnp A
11 JSR SHIFTR
12 ADD R1,R4,#0
```

```

13 JSR SHIFTR
14 ST R4,C
15 TRAP x25
16 B .BLKW 1
17 C .BLKW 1
18 .END

```

- (a) How many memory locations are required to store the assembled program?
- (b) What is the address of the location labeled C?
- (c) Before the program can execute, the location labeled B must be loaded by some external means. You can assume that happens before this program starts executing. You can also assume that the subroutine starting at location SHIFTR is available for this program to use. SHIFTR takes the value in R1, shifts it right one bit, and stores the result in R4.
- After the program executes, what is in location C?

h5 (a)

需要 16 行存储

h5 (b)

x4015

h5 (c)

C 中存储的是 R1 右移两位的结果, $R1 = M[B] + M[B + 1] + M[B + 2] + M[B + 3]$

T4

“

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 ; R0 gets next integer for MUL
7 BRnp AGAIN
8 DONE ADD R0,R1,#0 ; Move n! to R0
9 LD R1,SAVE_R1
10 RET
11 SAVE_R1 .BLKW 1

```

```

1  FACT ST R1,SAVE_R1
2      ADD R1,R0,#0
3      ADD R0,R0, #-1
4      BRnz DONE
5  AGAIN MUL R1,R1,R0
6      ADD R0,R0,#-1 ; R0 gets next integer for MUL
7      BRnp AGAIN
8  DONE ADD R0,R1,#0 ; Move n! to R0
9      BRp SKIP
10     ADD R0, R0, #1
11  SKIP LD R1,SAVE_R1
12     RET
13  SAVE_R1 .BLKW 1

```

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

h5 (a)

可能读取到上次读入的字符

h5 (b)

可能覆盖掉上一次未被读入的字符

h5 (c)

第一个问题更可能发生，因为键盘是中断驱动的 I/O，当有新的输入时会自动调用中断服务程序完成读取操作

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?

Input 时像 LC-3 一样，直接检测 IOSR 的是否是负数即可；**Output** 时需要检测 IOSR[14] 是否为 1，可以先左移一位，再检测是否为负数；

理论上说，这样的设计多消耗了一个临时寄存器用来存储 IOSR 左移的结果，又多了一个 ADD 指令，使 I/O 不方便不统一，is a poor decision

T7

“

- (a) How many **TRAP** service routines can be implemented in the LC-3? Why?
- (b) How many accesses to memory are made during the processing of a **TRAP** instruction?

h5 (a)

最多 256 个服务程序，因为 8bit 只能表示 256 个地址

h5 (b)

两次，先读取陷入向量表中对应位置存储的服务程序地址，再读取相应地址存储的程序内容

T8

“

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, #3
4 TRAP x22
5 TRAP x25
6 LABEL .STRINGZ "FUNKY"
7 LABEL2 .STRINGZ "HELLO WORLD"
8 .END
```

输出结果为: **FUN**

T9

“

Assume that an integer greater than 2 and less than 32,768 is deposited in memory location A by another module before the program below is executed.

```
1 .ORIG x3000
2 AND R4, R4, #0
```

```

3      LD R0, A
4      NOT R5, R0
5      ADD R5, R5, #2
6      ADD R1, R4, #2
7      ;
8      REMOD JSR MOD
9      BRz STORE0
10     ;
11     ADD R7, R1, R5
12     BRz STORE1
13     ADD R1, R1, #1
14     BR REMOD
15     ;
16     STORE1 ADD R4, R4, #1
17     STORE0 ST R4, RESULT
18     TRAP x25
19     ;
20     MOD ADD R2, R0, #0
21     NOT R3, R1
22     ADD R3, R3, #1
23     DEC ADD R2, R2, R3
24     BRp DEC
25     RET
26     ;
27     A .BLKW 1
28     RESULT .BLKW 1
29     .END

```

In 25 words or fewer, what does the above program do?

判断 A 是否是素数，如果是，通过 RESULT 输出 1，否则输出 0

T10

“

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

```

- ☐ (a): `ADD R1, R1, #1`
- ☐ (b): `HALT`
- ☐ (c): `ADD R0, R0, #5`
- ☐ (d): `BRzp K`

T11

“

nterrupt-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."

h5 (a)

循环打印字符 ‘2’

h5 (b)

回显两次键盘输入

h5 (c)

先回显两次键盘输入，然后继续循环打印 ‘2’

h5 (d)

两次或三次，因为在中断服务程序中没有保存和回复 R0，的值，如果是在 `LD R0, B` 之后 `TRAP x21` 之前中断的，那么会输出三次键盘输入字符；否则只会输出两次

T12

“

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.

PSR 寄存器无法恢复；PC 寄存器无法恢复；

