HW6 solution

T1

(a)

```
SaveRegisters
ST R0, SAVER0
ST R3, SAVER3
ST R4, SAVER4
ST R5, SAVER5
ST R6, SAVER6
RET
;
RestoreRegisters
LD R0, SAVER0
LD R3, SAVER3
LD R4, SAVER4
LD R5, SAVER5
LD R6, SAVER6
RET
;
SAVER0 .BLKW x1
SAVER3 .BLKW x1
SAVER4 .BLKW x1
SAVER5 .BLKW x1
SAVER6 .BLKW x1
```

(b)

原因：`F` 里调用了其他 subroutine，之前 main 里 `JSR F` 时 R7 里存的 PC 早已被修改，在 `F` 最后 `RET` 时将会不断地循环。

修改：调用 `SaveRegisters` 前保存 PC，最后 `RET` 前恢复即可。

T2

(a)

```
CLEAR
ST  R2, TEMP
LEA R2, MASKS
ADD R2, R1, R2
LDR R2, R2, #0
NOT R2, R2
AND R0, R2, R0
LD  R2, TEMP
RET
TEMP .BLKW #1
```

(b)

```
SET
ST  R2, TEMP
LEA R2, MASKS
ADD R2, R1, R2
LDR R2, R2, #0
NOT R2, R2
NOT R0, R0
AND R0, R2, R0
NOT R0, R0
LD  R2, TEMP
RET
TEMP .BLKW #1
```

T3

(a) 16

(b) C 的地址是 **x400F**

(c) 程序执行完成后，C 中包含从 B 中指定的内存位置开始的连续四个数字的**平均值**。

T4

```
FACT    ST    R1,SAVE_R1
        ADD   R1,R0,#0
        BRnp  SKIP
        ADD   R1,R1,#1
        BRnzp DONE
SKIP    ADD   R0,R0,#-1
        BRz   DONE
AGAIN   MUL   R1,R1,R0
        ADD   R0,R0,#-1     ; R0 gets next integer for MUL
        BRnp  AGAIN
DONE    ADD   R0,R1,#0      ; Move n! to R0
        LD    R1,SAVE_R1
        RET
SAVE_R1 .BLKW 1
```

在 `FACT` 中增加对于 0 的判断，从而可以处理 0!。

T5

(a) 程序可能会反复读取同一个字符。

(b) 当 KBSR[15] 为 1 时，KBDR 存的数据还没有被读取。因此，当键盘在写入 KBDR 前没有检查 KBSR 时，可能会造成用户输入的字符丢失。

(c) 问题 (a) 更容易发生，因为相比程序执行指令的速度，用户输入新字符的速度要慢很多。

T6

当 KBSR 和 DSR 是两个独立的寄存器时，由其最高位来表示状态。编写程序时，只需要读取寄存器的值，并判断符号即可。例如：

```
START LDI R1, A  ; Test for
      BRzp START ; character input
      LDI R0, B
      BRnzp NEXT_TASK ; Go to the next task
A .FILL xFE00 ; Address of KBSR
B .FILL xFE02 ; Address of KBDR
```

而当 KBSR 和 DSR 合成同一个寄存器时，我们必须用掩码和 IOSR 做“与”运算，获得第 14 位和 15 位的值，再进行判断。

因此这种方案虽然减少了寄存器个数，但是增加了编写程序的复杂度，不是一个好的方案。

T7

(a) 256。因为 `TRAP` 向量有 8 位。

(b) 4。

第一次：取指。

第二次：PSR 入系统栈

第三次：PC 入系统栈

第四次：取 `TRAP` 向量对应的内容。

T8

x22	**PUTS**	**Write a string of ASCII characters to the console display. The characters are contained in consecutive memory locations, one character per memory location,starting with the address specified in R0. Writing terminates with the occurrence of x0000 in a memory location.**

`PUTS` 在遇上 x0000 时会停止，因此答案为：FUN

T9

如果 A 中值为素数，就在 RESULT 存 1，否则存 0。

T10

(a) `ADD R1, R1, #1`

(b) `TRAP x25`

(c) `ADD R0, R0, #5`

(d) `BRzp K`

T11

(a)

1. 允许键盘中断

2. 重复输出”2“在屏幕上

(b)

重复输出字符两次

(c)

先输出若干个”2“，再输出键盘输入的字符 2 次或 3 次，再循环输出”2“

(d)

2 次或 3 次。

当键盘中断发生在 `LD R0, B` 和 `TRAP x21` 之间时（即 `LD R0, B` 执行完后立即发生），会输出 3 次，其他情况输出 2 次。

T12

1. privilege mode 和 condition codes 没有被恢复
2. `RET` 是将 R7 的值赋给 PC，但此时 R7 里存的不是正确的 PC 值，会返回到一个错误的地址
3. 栈没有弹出数据，栈指针没有被移动