

# Hw5

## T1

- **问题：**程序缺少 **HALT** 指令；
- **可能会发生的：**在完成主逻辑后，会执行内存中的未知指令。

- ```
1 ; 解法1
2 .ORIG x3000
3     AND R1, R1, #0
4 PROGRAM ADD R1, R1, #1
5     TRAP x23      ; IN
6     LEA R0, MESSAGE
7     TRAP x22      ; PUTS
8 NEW     TRAP x25      ; HALT
9 MESSAGE .STRINGZ "Program reporting"
10 .END             ; HALT
```

- ```
1 ; 解法2
2 .ORIG x3000
3     AND R1, R1, #0
4 PROGRAM ADD R1, R1, #1
5     TRAP x23      ; IN
6     LEA R0, MESSAGE
7     TRAP x22      ; PUTS
8 NEW     BRnzp PROGRAM ; 实现程序循环
9 MESSAGE .STRINGZ "Program reporting"
10 .END             ; HALT
```

## T2

1. 因为在执行 **HALT** 指令时，操作系统跳转到 **HALT** 操作历程，在该历程中暂时借用 **R0**、**R1**、**R6** 进行相关操作，但在 **MCR** 最高位置为 0 后，没有机会再将寄存器恢复到原来的值（也没必要恢复），于是部分寄存器中的值被修改。
2.
  - 不会执行。
  - 作为安全措施，保证在运行发生意外时，重新尝试 **HALT** 操作，而不是退出该服务例程继续执行操作。

## T3

1. **触发机制不同：**
  - **TRAP:** 由程序内部的 **TRAP** 指令触发；
  - **中断:** 外部硬件设备触发；
2. **同步性不同：**
  - **TRAP:** 是时钟同步，发生位置可以被预测的；
  - **中断:** 是异步的，在程序执行指令的任何时刻都可能发生；
3. **优先级不同：**
  - **TRAP:** 不需要进行优先级检查；

- **中断**: 需要进行优先级检查, 只有外部设备优先级高于 **CPU** 当前优先级, **CPU** 才会响应中断, 否则中断被挂起。

## T4

1. **系统空间**: 操作系统所在的特权内存区域 (x0000-x2FFF), 包含 OS 代码和数据。
2. **用户空间**: 用户程序运行的内存区域 (x3000-xFDFF)。
3. **用户模式下不可访问的地址**: 系统空间地址 (x0000-x2FFF) 和 I/O 设备寄存器地址(xFE00-xFFFF)。
4. **ACV**: Access Control Violation (访问控制违规)。指在用户模式下尝试访问系统空间地址时触发的硬件异常。
5. **特权模式确定**: 处理器状态寄存器最高位确定 (**PSR[15]** 的值)。
6. **切换到特权模式的方法**:
  - **TRAP (System Call)** ;
  - **Interrupt (中断)** ;
  - **Exception (异常)** 。

## T5

- **MEM[A] = 0010 0000 0000 1010**
- **MEM[E] = 0010 0011 1111 0100**
- **B -> ADD R0, R0, #1**
- **MEM[C] = x0021 + x1021 = x1042 -> ADD R0, R1, R2**
- **R0 = MEM[E] + x0001 + x0021 + MEM[A] = x4420**

## T6

1.	Cycle	State	Bus	Important Control Signals for This Cycle
	T	18	x3010	LD.MAR=1, LD.PC=1, PCmux=PC+1, GatePC=1
	T+4	30	xA202	GateMDR=1, LD.IR=1
	T+6	10	x3013	ADDR1MUX=PC, ADDR2MUX=10, MARMUX=1, GateMARMUX=1
	T+10	26	x4567	GateMDR=1, LD.MAR=1
	T+14	27	x0000	LD.REG=1, LD.CC=1, GateMDR=1, DR=001

2. **LDI R1, #2**
3. **x3010;**
4. 内存访问需要 2 个时钟周期;
5.
  - **M[x3010] = xA202**
  - **M[x3013] = x4567**
  - **M[x4567] = x0000**

## T7

```

1      .ORIG x3000
2      LD  R0, A
3      LD  R1, B
4  AGAIN BRnzp DONE
5      ADD R0, R0, R0      ; (a)
6      ADD R1, R1, #-1    ; (b)
7      BRnzp AGAIN
8  DONE ST  R0, A
9      HALT
10     A      .FILL x0c00      ; (c)
11     B      .FILL x0001
12     .END

```

Cycle Number	State Number	Control Signals
1	18	LD.MAR : 1, LD.REG : 0, GateMDR : 0, LD.PC : 1, PCmux : PC+1, GatePC : 1
44	0	LD.MAR : 0, LD.REG : 0, BEN : 0, LD.PC : 0, LD.CC : 0
54	1	LD.REG : 1, DR : 000, GateMDR : 0, GateALU : 1, GateMARMux : 0
64	1	LD.MAR : 0, ALUK : 00, GateALU : 1, LD.REG : 1, DR : 001, GatePC : 0
86	22	ADDR1MUX : 0, ADDR2MUX : 10, LD.PC : 1, LD.MAR : 0, PCMUX : 10
112	15	-

- 将 MEM[A] 左移 MEM[B], 并存回 MEM[A]。

## T8

```

1  ADD R2, R0, #0 ; R2 = ADDRESS
2  LDR R0, R2, #0 ; R0 = MEM[ADDRESS] = value
3  TRAP x21      ; 输出 R0
4  ADD R0, R2, #0 ; R0 = ADDRESS
5  LDR R1, R0, #1 ; R1 = next_addr
6  LDR R2, R1, #1 ; R2 = next_addr->next
7  LDR R1, R1, #0 ; R1 = next_value
8  STR R1, R0, #1
9  STR R2, R0, #0
10 RET

```