ICS hw4

PB20000180 刘良宇

t1

```
0101 100 100 1 00000    ; set R4 0
1001 000 001 111111     ; R0 = ~R1
0001 000 000 1 00001    ; R0++
0001 000 000 000 010    ; R0 = R0 + R2
0000 011 000000001      ; if (R0 >= 0) goto halt;
0001 100 100 1 00001    ; R4 = R4 + 1
1111 0000 0010 0101     ; halt
```

由上述补充的注释可得：

R4 置为 0 ，R0 先变为自己的相反数，再加上 R2，如果结果是负数，R4 才加一

也就是说，相当于以下三目赋值语句：

R4 = (R2 - R0) < 0 ? 1 : 0

t2

```
0101 000 000 1 00000    ; set R0 0
0101 101 001 1 00001    ; R5 = R1 & 1
0000 010 000000001      ; if (!R5) goto halt
0001 000 000 1 00001    ; R0++
1111 0000 0010 0101     ; halt
```

由上述补充的注释可得：

R0 置为 0 ，R5 取 R1 的最低位，如果 R5 是 1，R0++

也就是说，R0 取 R1 的最低位

t3

观察可得，在执行了这条指令后，R5 的值从 x5555 变为 xFFF8

而 x1000 处是一个加法指令。

注意到可以让 R0 与立即数加法达到目的

则补全为：0001 101 000 1 11000

t4

```
x3000 0101 000 000 1 00000      ; R0 = 0
x3001 0001 000 000 1 00101      ; R0 += 5
x3002 0010 001 000000100        ; R1 = from x3007
x3003 0001 000 000 0 00 000     ; R0 *=2
x3004 0001 001 001 1 11111      ; R1--
x3005 0000 001 111111101        ; if (R1 > 0) goto x3003
x3006 1111 0000 00100101        ; halt
x3007 0000 000 000000100        ; data
```

a

所以是在计算 5×2^n 后存储到 R0

其中，n 是 x3007 地址存储的值

b

寄存器：R0 储存的是结果。R1 最后是 0，其他值不变

在 halt 前最后一条是 R1-- 且结果为 0。所以 nzp 为 010

c

查表可得：

$10 + 10 + 16 + 10 \times 4 + 10 \times 4 + 11 \times 3 + 10 = 159$

t5

The PC is initially at x3000.

```
x3000 0001 000 000 1 10000  ; R0 = xFFF0
x3001 0010 001 011111110    ; R1 = from x3100
x3002 0000 010 000000100    ; if (!R1) goto x3007
x3003 0000 011 000000001    ; if (R1 >= 0) goto x3005
x3004 0001 000 000 1 00001  ; R0++
x3005 0001 001 001 000 001  ; R1 << 1
x3006 0000 111 111111011    ; goto x3002
X3007 1001 000 000 111111   ; R0 = ~R0
X3008 0001 000 000 1 00001  ; R0++;
x3009 1111 0000 0010 0101   ; halt
```

从 R0为 -16 开始，遍历 x3100 处数值的每一位，如果是 1 ，就 R0++，最后 R0 取相反数。

故总结：本程序可以统计指定地址处数字 0 的个数，存到 R0

t6

```
x3000 1110 000 011111111    ; R0 = x3100
x3001
x3002
x3003 0001 000 000 1 00001  ; R0++
x3004 0000 111 111111100    ; goto x3001
x3005 0011 000 001001010    ; store x3050 = R0
x3006 1111 0000 0010 0101   ; halt
```

根据语义填写指令：这里应该是取 R0 地址处对应的值（不妨存入 R1），如果是负数，就跳出循环

x3001: 0110 001 000 000000 ; R1 = [R0]

x3002: 0000 100 000000010 ; goto x3005

t7

LDI 指令实际包含了两次内存读取，

STI 指令包含了一次读和一次写。

这就可以解释 “矛盾”

t8

a

R1 = R1 + 0 。缺点是会更新 nzp

b

不论如何，PC++，会直接跳过下一个指令

c

可以。

t9

实现寄存器 1 和寄存器 2 的逻辑或运算，结果储存到寄存器 3

$R3 = R1 \mid R2$

$\overline{R3} = \overline{R1}\ \overline{R2}$

$R3 = \overline{\overline{R1}\ \overline{R2}}$

故可以据此写出代码

```
1001 100 001 111111     ; R4 = ~R1
1001 101 010 111111     ; R5 = ~R2
0101 110 100 0 00 101   ; R6 = R4 & R5
1001 011 110 111111     ; R3 = ~R6
```