LAB04 实验报告

TASK1

原代码

```
1   1110010000001110    ;R2 = PC + 1 + 15
2   0101000000100000    ;R0 = 0
3   010010000000000x    ;R7 = PC + 1, PC = PC + 1 + X
4   1111000000100101    ;HALT
5   0111111010000000    ;MEM[R2] = R7
6   000101001010x001    ;R2 = R2 + X
7   0001000000100001    ;R0 = R0 + 1
8   0010001000010001    ;R1 = MEM[PC + 1 + 17]
9   0001001x01111111    ;R1 = RX - 1
10  0011001000001111    ;MEM[PC + 1 + 15] = R1
11  0000010000000001    ;IF Z, PC = PC + 2
12  0100111111111000    ;R7 = PC + 1, PC = PC + 1 - 8
13  0001010010111111    ;R2 = R2 - 1
14  01x0111010000000    ;UNKNOWM
15  1100000111000000    ;PC = R7
16  0000000000000000
17  0000000000000000
18  0000000000000000
19  0000000000000000
20  0000000000000000
21  0000000000000000
22  0000000000000000
23  0000000000000000
24  0000000000000000
25  0000000000000000
26  0000000000000101    ;X0005
```

- 首先对于第三行，显然程序不可能直接依序到达下一行然后终止，故必有

  ```
  3   0100 1 00000000001   ;R7 = PC + 1, PC = PC + 2
  ```

  其次对于8，9，10，11三行而言，若9行中的X不为1，则第8行的装载没有意义，故必有

  ```
  9   0001 001 001 1 11111     ;R1 = R1 - 1
  ```

  最后对于14行，01x0形式且后续具有14行形式的指令只有可能是LDR指令：

  ```
  14  0110 111 010 000000 ;R7 = MEM[R2]
  ```

  那么除开行6，我们可以对程序进行试运行。

- 假设PC从x3000开始，第一次跳转前（1-12）

PC	R0	R1	R2	R7	X300F	X3010	X3011	X3012	X3013
X3004	1	4	X300F + X	X300C	X3003	X0000	X0000	X0000	X0000

在此次之后，若R2 = R2 + 9,则在下下一次执行5行指令时，将会离开程序所占用的空间，显然是不能接受的，故必有

```
6   0001 010 010 1 00001     ;R2 = R2 + 1
```

我们可以继续执行指令以验证最终结果。

第二次跳转前（5-12）：

PC	R0	R1	R2	R7	X300F	X3010	X3011	X3012	X3013
X3004	2	3	X3010	X300C	X3003	X300C	X0000	X0000	X0000

第三次跳转前（5-12）：

PC	R0	R1	R2	R7	X300F	X3010	X3011	X3012	X3013
X3004	3	2	X3011	X300C	X3003	X300C	X300C	X0000	X0000

第四次跳转前（5-12）：

PC	R0	R1	R2	R7	X300F	X3010	X3011	X3012	X3013
X3004	4	1	X3012	X300C	X3003	X300C	X300C	X300C	X0000

第四次跳转，这一次R1 更新为0，执行11行，跳过12行到达 一个新的循环（13-15）：

PC	R0	R1	R2	R7	X300F	X3010	X3011	X3012	X3013
X300C	5	0	X3012	X300C	X3003	X300C	X300C	X300C	X300C

每一次循环R2都自减一次，最终达到X300F，此时：

PC	R0	R1	R2	R7	X300F	X3010	X3011	X3012	X3013
X3003	5	0	X300F	X3003	X3003	X300C	X300C	X300C	X300C

而X3003处所储存的正好是HALT指令，此时程序停止，且寄存器状态如下：

```
R0 = 5, R1 = 0, R2 = 300f, R3 = 0
R4 = 0, R5 = 0, R6 = 0, R7 = 3003
```

最终代码

```
1    1110010000001110     ;R2 = PC + 1 + 15
2    0101000000100000     ;R0 = 0
3    0100100000000001     ;R7 = PC + 1, PC = PC + 1 + X
4    1111000000100101     ;HALT
5    0111111010000000     ;MEM[R2] = R7
6    0001010010100001     ;R2 = R2 + X
```

```
7   0001000000100001    ;R0 = R0 + 1
8   0010001000010001    ;R1 = MEM[PC + 1 + 17]
9   0001001001111111    ;R1 = RX - 1
10  0011001000001111    ;MEM[PC + 1 + 15] = R1
11  0000010000000001    ;IF Z, PC = PC + 2
12  0100111111111000    ;R7 = PC + 1, PC = PC + 1 - 8
13  0001010010111111    ;R2 = R2 - 1
14  0110111010000000    ;UNKNOWM
15  1100000111000000    ;PC = R7
16  0000000000000000
17  0000000000000000
18  0000000000000000
19  0000000000000000
20  0000000000000000
21  0000000000000000
22  0000000000000000
23  0000000000000000
24  0000000000000000
25  0000000000000000
26  0000000000000101    ;X0005
```

TASK2

原代码

```
1   0010001000010101    ;R1 = MEM[PC + 1 + 21]
2   0100100000001000    ;R7 = PC + 1, PC = PC + 1 + 8
3   0101010001100111    ;R2 = R1 AND X0007
4   0001001010000100    ;R1 = R2 + R4
5   00010000xxx11001    ;R0 = RX + X
6   00000011xxx11011    ;IF P, PC = PC - X
7   00010000xxx11001    ;R0 = RX + X
8   0000100000000001    ;IF N, PC = PC + 2
9   0001001001111001    ;R1 = R1 - 7
10  1111000000100101    ;HALT
11  0101010010100000    ;R2 = R2 AND X0000
12  0101011011100000    ;R3 = R3 AND X0000
13  0101100100100000    ;R4 = R4 AND X0000
14  0001010010100001    ;R2 = R2 + 1
15  0001011011101000    ;R3 = R3 + 8
16  0101101011000001    ;R5 = R3 AND R1
17  0000010000000001    ;IF Z, PC = PC + 2
18  0001100010000100    ;R4 = R2 + R4
19  0001010010000010    ;R2 = R2 + R2
20  0001xxx011000011    ;RX = R3 + R3
21  0000xxx111111010    ;IF X, PC = PC - 6
22  1100000111000000    ;PC = R7
23  0000000100100000    ;X0120
```

观察2行，其跳转位置为11行，观察21行，其跳转位置为16行，即若不满足21行的条件，则会一直处在该循环中。根据hint，我们可以猜测此处循环的作用是将被除数除以8。

- 猜想
    - 初始化（11-15），此后寄存器的状态：

```
R2 = 0000 0000 0000 0001;
R3 = 0000 0000 0000 1000;
```

- 执行循环，即令R3与R1相与，检查R1[3]是否为1。

  若有：

  ```
  R4 = 0000 0000 0000 0001;
  R2 = 0000 0000 0000 0010;
  R3 = 0000 0000 0001 0000;
  ```

  若无：

  ```
  R2 = 0000 0000 0000 0010;
  R3 = 0000 0000 0001 0000;
  ```

  即若此处有值，直接将其右移3位，并将其值累加储存于R4，若无值，则集体右移，进行下一次循环。最终的结果即为：

  ```
  R1 = AAAA AAAA AAAA AXXX => R4 = 000A AAAA AAAA AAAA
  ```

  浅显的说，若设

  ```
  R1 = 8 * m + n
  ```

  则该出循环结果为最终R4储存m，然后PC跳转回3行。

- 当然上面的仍是猜想，我们不妨带着这样的结果继续执行程序，在达到丢失的代码前寄存器状态会更新为：

  ```
  R2 = n, R1 = m + n
  ```

  我们发现事实上原来的R1可以写为：

  ```
  R1 = 7 * m + m + n
  ```

  即在执行完除以8的循环后得到的m + n理应就是余数。

  若m + n > 7，则重新令R1 = m + n，再次利用相同方法进行计算；

  若m + n <= 7，则直接输出。

  而原代码正好可以写成符合如此判断的形式：

  ```
  5    0001000011111001    ;R0 = m + n - 7
  6    0000001111111011    ;else m + n > 7, update R1 and then divide it by 8
  7    00010000xxx11001    ;R0 = m + n - 7
  8    0000100000000001    ;else m + n < 7, halt
  9    0001001001111001    ;else m + n = 0
  10   1111000000100101    ;HALT
  ```

  最终得到的余数储存在R1中。

- 验证猜想。在上述的猜想中我们可以得到一个完整的可执行的且逻辑清晰的代码。最后我们可以通过实际执行来验证猜想的正确性：
  - 初始状态

R1	R2	R4
X0120	X0000	X0000

  - 除以8

R1	R2	R4
X0120	X2000	X0024

  - 判断R1 = R2 + R4

R1	R2	R4
X0024	X0000	X0024

  - 显然R1 > 7，再次执行循环

R1	R2	R4
X0024	X0004	X0004

  - 判断R1 = R2 + R4

R1	R2	R4
X0008	X0004	X0004

  - 显然R1 > 7, 再次执行循环

R1	R2	R4
X0008	X0000	X0001

  - 判断R1 = R2 + R4

R1	R2	R4
X0001	X0000	X0001

    此时R1 < 7，直接输出。而事实上X0120所代表的288除以7所得余数确实为1。至此，我们完成了猜想的验证与代码的完善。

完整代码

```
0010001000010101    ;R1 = MEM[PC + 1 + 21]
0100100000001000    ;R7 = PC + 1, PC = PC + 1 + 8
0101010001100111    ;R2 = R1 AND X0007
0001001010000100    ;R1 = R2 + R4
0001000011111001    ;R0 = R1 - 7
0000001111111011    ;IF P, PC = PC - 4
0001000011111001    ;R0 = R1 - 7
```

```
0000100000000001    ;IF N, PC = PC + 2
0001001001111001    ;R1 = R1 - 7
1111000000100101    ;HALT
0101010010100000    ;R2 = R2 AND X0000
0101011011100000    ;R3 = R3 AND X0000
0101100100100000    ;R4 = R4 AND X0000
0001010010100001    ;R2 = R2 + 1
0001011011101000    ;R3 = R3 + 8
0101101011000001    ;R5 = R3 AND R1
0000010000000001    ;IF Z, PC = PC + 2
0001100010000100    ;R4 = R2 + R4
0001010010000010    ;R2 = R2 + R2
0001011011000011    ;R3 = R3 + R3
0000101111111010    ;IF NOT Z, PC = PC - 6
1100000111000000    ;PC = R7
0000000100100000    ;X0120
```