lab4:Reveal Yourseslf

PB20000248　郑子涵

实验要求

- 根据程序运行后寄存器的状态来补全task1.txt的汇编代码
- 根据程序的功能（对七取模）补全task2.txt的汇编代码

实验过程

Task1

- **第一处的X=1**,因为第一处x的下一句为HALT指令，若x=0，则相当于下一句执行HALT，程序结束，不符合结果，所以一定为x=1，跳过HALT语句。
- 综合分析接下来的程序，R2相当于是一个栈空间的指针储存R7(即程序返回的地址值)，R1初始化为5，即递归5次直到R1=0，每次递归R0++,R2每次 push R7，然后依次return程序，R2 pop 返回地址到R7，最后 pop 第一个返回值X3003到HALT，程序结束。
- 第二处x代表这push时栈指针的递增，即R2=R2+1，所以**第二处X=0**。
- 第三处的x代表R1递减，若x=1代表R1=R5-1，不符合最后寄存器的状态，所以应为R1=R1-1,即递归条件为R1!=0，递归5次，所以**第三处X=0**。
- 第四处x代表，把返回地址pop到R7中，即指令应为LDR指令,把M[R2]处的的值load到R7作为返回地址，所以**第四处X=1**。

```
1.  1110 010 000001110      ;LEA  R2,x300f
2.  0101 000 000 1 00000    ;R0=0
3.  0100 1 0000000000x      ;JSR TO "x3004"  R7=X3003   (x=1)
4.  1111000000100101        ;HALT
5.  0111 111 010 000000     ;STR M[R2]=R7
6.  0001 010 010 1 0x001    ;R2=R2+1      (x=0)
7.  0001 000 000 1 00001    ;R0=R0+1
8.  0010 001 000010001      ;R1=M[X3019]
9.  0001 001 x01 1 11111    ;R1=R1-1     (x=0)
10. 0011 001 000001111      ;M[X3019]=R1
11. 0000 010 000000001      ;IF R1=0 BR TO "X300C"
12. 0100 1 11111111000      ;JSR TO "X3004"  R7=X300B
13. 0001 010 010 111111     ;R2=R2-1
14. 01x0 111 010 000000     ;R7=M[R2]       (x=1)
15. 1100 000 111 000000     ;JMP->R7
16. 0000000000000000
17. 0000000000000000
18. 0000000000000000
19. 0000000000000000
20. 0000000000000000
21. 0000000000000000
22. 0000000000000000
23. 0000000000000000
24. 0000000000000000
25. 0000000000000000
26. 0000000000000101
```

Tast2

- Tast2中的程序的目的是算mod7的余数,对7的余数可以用以下方法

> 整数 X
> $X = (7+1) * n + m$
> $X\%7 = (n+m)\%7$

- 第一部分，下面程序中第11到22行就是算出n把其存储到R4中，其中R2代表这各个位对于8的权值，R3用于依次取R1第四位以上的各个位，所以第20行处，代表着R3的左移，所以有(XXX=011)，第21行处是用于判断是否统计完R1所有位，即判断R3是否为0，所以(xxx=101)。
- 第二部分，就是先取R1的后三位在加上第一部分算出的n存到R1，便可使得变换前后$R1\%7$的值不变，之后一直重复上述过程，直到R1<=7，最后处理R1=7的情况，把其变为0即可。

```
1.  0010 001 000010101       ;LD,R1,NUM
2.  0100 1 00000001000       ;POSE3 JSR "X300A"  R7=X3002
3.  0101 010 001 1 00111     ;R2=R1 AND X0007
4.  0001 001 010 000 100     ;R4+R2->R1
5.  0001 000 0xx x 11001     ;R0=R1-7   (XXX=011)
6.  0000 001 1xxx11011       ;BR IF R0>0  TO POSE3  (xxx=111)
7.  0001 000 0xx x 11001     ;R0=R1-7  (XXX=011)
8.  0000 100 000000001       ;BR to HALT IF R0<0
9.  0001 001 001 1 11001     ;R1=R1-7
10. 1111000000100101         ;HALT
11. 0101 010 010 1 00000     ;R2=0
12. 0101 011 011 1 00000     ;R3=0
13. 0101 100 100 1 00000     ;R4=0
14. 0001 010 010 1 00001     ;R2=R2+1
15. 0001 011 011 1 01000     ;R3=R3+8
16. 0101 101 011 000 001     ;POSE2 R1 AND R3 -> R5
17. 0000 010 000000001       ;IF R5=0 JUMP->POSE1
18. 0001 100 010 000 100     ;R4=R2+R4
19. 0001 010 010 000 010     ;POSE1    R2=R2+R2
20. 0001 xxx 011 000 011     ;R3=R3+R3    (XXX=011)
21. 0000 xxx 111111010       ;BRnp     JUMP->POSE2   (XXX=101)
22. 1100000111000000         ;RET
23. 0000000100100000         ;NUM  #288
```

实验总结

- Tast1中的程序主要是理解代码的运行过程
- Tast2中的程序主要是要理解算法，即算余数的方法，但发现Tast2中的程序对负数取余并不有有效，所以对于最后一个BR语句的条件码，既可以为001，也可以为101。若为101是则是把第二十三行看成是一个16bits的无符号整数。