Lab Report - 02

PB20111686　Ruixuan Huang

Lab Goal

Giving the defination of a sequence

$$F(0)=1, F(1)=1, F(2)=2, F(n)=[F(n-1)+2F(n-3)] \bmod 1024$$

In the beginning, n is stored in `R0`. All other registers' initial values are 0. The lab goal is to store $F(n)$ in register `R7`. And divide your student number into four equal segments, labelling them with a, b, c and d. Store value of $F(a)$, $F(b)$, $F(c)$ and $F(d)$ at the end of your code with `.FILL` pseudo command.

Code

Note that n varies in 1 and 16384. If I store $F(0), F(1), F(2)$ in registers as initial state, I have to examine whether the input is $1, 2$ or not. So I caculated $F(-2), F(-1)$ using the recurrence formula (ignoring range of n temporarily).

$$F(2)=F(1)+2F(-1) \Rightarrow F(-1)=1/2$$
$$F(1)=F(0)+2F(-2) \Rightarrow F(-2)=0$$

However, LC-3 cannot represent $1/2$. We have to caculate $G(n)$ as alternative, and get $F(n)$ through $G(n)$.

$$G(-2)=0, G(-1)=1, G(0)=2, G(n)=[G(n-1)+2G(n-3)] \bmod 2048$$
$$F(n)=G(n)/2$$

```
.ORIG x3000
;Initial
    ADD R4, R4, #0  ; G(-2)
    ADD R5, R5, #1  ; G(-1)
    ADD R6, R6, #2  ; G(0)
    LD  R1, LEB     ; R1 = 2047

LOP ADD R3, R4, R4  ; R3 = 2R4
    ADD R3, R3, R6  ; R3 = 2R4 + R6
    AND R3, R3, R1  ; R3 = R3 % 2048
    ADD R4, R5, #0  ; R5 = R6
    ADD R5, R6, #0  ; R6 = R7
    ADD R6, R3, #0  ; R7 = R4
    ADD R0, R0, #-1 ; n = n - 1
    BRp LOP         ; if z, R6 now is G(n)

;To get F(n): R7 = R6 >> 1
    AND R1, R1, #0
    ADD R1, R1, #2  ; R1 = 'b10
    ADD R2, R2, #1  ; R2 = 'b01
DOP AND R4, R1, R6  ; R4 = R1 & R6
    BRz IFU
    ADD R7, R7, R2  ; R7 += R2
IFU ADD R2, R2, R2  ; R2 << 1
    ADD R1, R1, R1  ; R1 << 1
    BRp DOP

    TRAP x25

LEB .FILL x7ff ; 2047
```

```
a   .FILL #930
b   .FILL #246
c   .FILL #386
d   .FILL #754
.END
```

Not counting the a, b, c, d costs, the line number of my program is 22.

Optimization

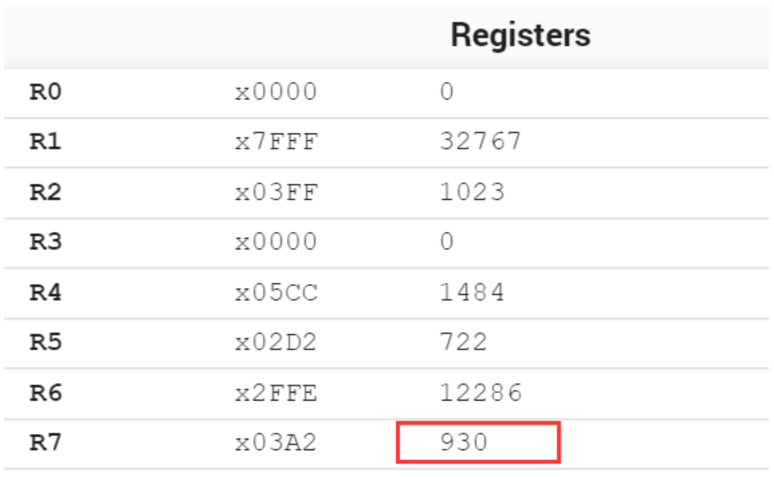
Although the first version doesn't need to judge the special case of n, it needs 9 extra lines to realize the right shift operation. That's not good. Using special judging may makes the number of line less.

```
.ORIG x3000
;Initial
    ADD R7, R7, #1
    ADD R0, R0, #-1
    BRz END         ;if(n==1)R7=1
    ADD R7, R7, #1
    ADD R0, R0, #-1
    BRz END         ;if(n==2)R7=2
    LD  R2, LEB
;Start Loop
    ADD R6, R6, #1
    ADD R5, R5, #1
LOP ADD R4, R5, R5
    ADD R5, R6, #0
    ADD R6, R7, #0
    ADD R7, R4, R7
    AND R7, R7, R2
    ADD R0, R0, #-1
    BRp LOP

END TRAP x25

LEB .FILL #1023

a   .FILL #930
b   .FILL #246
c   .FILL #386
d   .FILL #754

.END
```

Not counting the a, b, c, d costs, the line number of my program is 18.

Correctness

Using LC-3 tools, input `R0` as 20. The output is the same with C++ programming.



	Registers	
R0	x0000	0
R1	x7FFF	32767
R2	x03FF	1023
R3	x0000	0
R4	x05CC	1484
R5	x02D2	722
R6	x2FFE	12286
R7	x03A2	930