Lab 2 实验报告

- 姓名：谭旭
- 学号：PB20030775

实验内容

使用 LC3 实现对一个类斐波那契数列的计算。

数列递推公式为

$$F(0) = 1$$

$$F(1) = 1$$

$$F(2) = 2$$

$$F(n) = (F(n-1) + 2F(n-3)) \mod 1024, (1 \leq n \leq 16384)$$

初始时， n 存放于 R0 ， 要求将结果存放到 R7 。

实验过程

第一版

Code:

```
        .ORIG   x3000
        LD      R1,     NUMBERA
        ADD     R0,     R0,     #-1
        BRp     A
        ADD     R7,     R7,     #1
        HALT
A       ADD     R0,     R0,     #-1
        BRp     B
        ADD     R7,     R7,     #2
        HALT
B       ADD     R2,     R2,     #1
        ADD     R3,     R3,     #1
        ADD     R7,     R7,     #2
C       ADD     R4,     R7,     #0
        ADD     R7,     R7,     R2
        ADD     R7,     R7,     R2
        ADD     R2,     R3,     #0
        ADD     R3,     R4,     #0
        ADD     R0,     R0,     #-1
        BRp     C
        AND     R7,     R7,     R1
        HALT
NUMBERA .FILL   #1023
```

```
NUMBER1 .FILL   #930
NUMBER2 .FILL   #4
NUMBER3 .FILL   #30
NUMBER4 .FILL   #790
        .END
```

共 28 行，删除始末语句、学号后剩余 22 行。

对于 $n=0,1,2$ ，由于无法通过通项公式得出，故使用特判来处理。

若 $n-1\leq 0$ ，则程序将不会有任何跳转，直接将答案 1 赋值给 R7 后结束。

否则，程序将会继续计算是否 $n-2=0$ ，若相等，则将答案 2 赋值给 R7 后结束。

对于其他情况，程序将使用三个寄存器 R2 、 R3 、 R4 作为中继，计算数列中的下一个值。

三个寄存器的所储存的数据如下：

当完成了对 R7 的计算时，即 R7 中存储了 $F(n)$ 时，

有 R2 中存储了 $F(n-3)$ ， R3 中存储了 $F(n-2)$ ， R4 中存储了 $F(n-1)$ 。

并且，在计算下一个 $F(n)$ 时，直接使用 R7 中的值作为 $F(n-1)$ 。

第二版

分析上述第一版的代码，不难发现其中特判的部分占据了 8 行，其长度与代码核心循环部分相近。

因此考虑优化特判部分。

由于对于 $n=0,1$ ， $F(n)$ 的值都相同，可以考虑直接计算 $n-2$ 的值，

当 $n-2<0$ 时，将答案 1 存放到 R7 ，

当 $n-2=0$ 时，将答案 2 存放到 R7 ，

这样一来，特判部分只需要做一次减法，即可分两种情况输出答案。

并且，由于当 $n>2$ 时，需要对寄存器 R7 初始化为 2 ，

可以考虑在特判之前直接将寄存器 R7 初始化为 2 ，如果得到 $n-2=0$ ，则直接停止程序。

因此第二版代码比第一版代码减少了 2 行。

Code:

```
        .ORIG   x3000
        LD      R1,     NUMBERA
        ADD     R2,     R2,     #1
        ADD     R3,     R3,     #1
        ADD     R7,     R7,     #2
        ADD     R0,     R0,     #-2
        BRn     A
```

```
        BRp     B
        HALT
A       ADD     R0,     R0,     #-1
        HALT
B       ADD     R4,     R7,     #0
        ADD     R7,     R7,     R2
        ADD     R7,     R7,     R2
        ADD     R2,     R3,     #0
        ADD     R3,     R4,     #0
        ADD     R0,     R0,     #-1
        BRp     B
        AND     R7,     R7,     R1
        HALT
NUMBERA .FILL   #1023
NUMBER1 .FILL   #930
NUMBER2 .FILL   #4
NUMBER3 .FILL   #30
NUMBER4 .FILL   #790
        .END
```

共 26 行，删除始末语句、学号后剩余 20 行。

第三版

继续考虑优化特判部分，对于 $1 \leq n \leq 16384$ ，显然有两个特殊情况不得不考虑，

并且由于 LC3 中不能使用浮点数，并不能通过假设 $F(-2), F(-1), F(0)$ 的方式计算得到特殊情况

而当 $4 \leq n \leq 16387$ 即 $1 \leq n-3 \leq 16384$ 时，显然对于所有的 n ，都可以使用递推公式得到 $F(n)$ 。

而在第一版的算法描述中，已经说明：

当完成了对 R7 的计算时，即 R7 中存储了 $F(n)$ 时，

有 R2 中存储了 $F(n-3)$ ， R3 中存储了 $F(n-2)$ ， R4 中存储了 $F(n-1)$ 。

因此可以对于任意的 n ，先计算出 $F(n+3)$ ，那么 R2 中就存储了 $F(n)$ 。

再将这个值赋值到 R7 输出即可。

Code:

```
        .ORIG   x3000
        LD      R1,     NUMBERA
        ADD     R2,     R2,     #1
        ADD     R3,     R3,     #1
        ADD     R7,     R7,     #2
B       ADD     R4,     R7,     #0
        ADD     R7,     R7,     R2
        ADD     R7,     R7,     R2
        ADD     R2,     R3,     #0
        ADD     R3,     R4,     #0
```

```
        ADD     R0,     R0,     #-1
        BRp     B
        ADD     R7,     R2,     #0
        AND     R7,     R7,     R1
        HALT
NUMBERA .FILL   #1023
NUMBER1 .FILL   #930
NUMBER2 .FILL   #4
NUMBER3 .FILL   #30
NUMBER4 .FILL   #790
        .END
```

共 21 行，删除始末语句、学号后剩余 15 行。

第四版

在第三版的基础上，考虑从一开始就在 R7 中存放 $F(n-3)$ ，就可以减少一次赋值操作。

Code:

```
        .ORIG   x3000
        LD      R1,     NUMBERA
        ADD     R7,     R7,     #1
        ADD     R6,     R6,     #1
        ADD     R4,     R4,     #2
A       ADD     R5,     R4,     #0
        ADD     R4,     R4,     R7
        ADD     R4,     R4,     R7
        ADD     R7,     R6,     #0
        ADD     R6,     R5,     #0
        ADD     R0,     R0,     #-1
        BRp     A
        AND     R7,     R7,     R1
        HALT
NUMBERA .FILL   #1023
NUMBER1 .FILL   #930
NUMBER2 .FILL   #4
NUMBER3 .FILL   #30
NUMBER4 .FILL   #790
        .END
```

共 20 行，删除始末语句、学号后剩余 14 行。

当完成了对 R4 的计算时，即 R4 中存储了 $F(n+3)$ 时，

有 R7 中存储了 $F(n)$ ， R6 中存储了 $F(n+1)$ ， R5 中存储了 $F(n+2)$ 。