lab2 实验报告

李远航 PB20000137

1. 实验内容

• 使用汇编语言实现修改后的 Fibonacci 数列, n 被储存在 RO 寄存器, 答案存储在 R7 寄存器

```
F(0) = 1
F(1) = 1
F(2) = 2
F(n) = (F(n) + 2 \times F(n-3)) \pmod{1024}
```

• 将学号PB20000137均匀分成四个相同的段,求出F(20), F(0), F(1), F(37)存在代码结尾

2. 设计思路

• 使用循环的方式求出答案,具体的实现思路见下 c++ 代码

```
1 #include <iostream>
2 int main()
   {
4
    short n;
 5
     std::cin >> n;
     int Fi[3];
 6
 7
     Fi[0] = 1;
 8
     Fi[1] = 1;
 9
     Fi[2] = 2;
10
     n -= 2;
     for (; n > 0; n--)
11
12
13
          int temp = Fi[0] << 1;</pre>
14
         Fi[0] = Fi[1];
          Fi[1] = Fi[2];
15
16
          Fi[2] = (Fi[1] + temp) \% 1024;
17
      }
    if (n < 0)
18
19
          std::cout << Fi[1] << std::endl;</pre>
20
          std::cout << Fi[2] << std::endl;</pre>
21
22
      return 0;
23 }
```

3. 初始版本代码

```
1 .ORIG x3000
2 LD R6, MOD ;初始化R6为1023
3 LD R5, PTR ;初始化数据指针
4 STR R0, R5, #4 ;memory[R5+4]=R0
5 return LDR R0, R5, #0 ;R0=memory[R5]
6 BRnzp FUNCTION ;跳转处理函数
7 NN STR R7, R5, #0 ;memory[R5]=R7
```

```
8 ADD R5, R5, #1 ;R5+=1
                          ;R0=R5&100(这里利用偏移四次之后产生进位)
 9
    AND RO, R5, #4
 10
    BRz return
                          ;没有进位就继续算
 11
    HALT
    FUNCTION LD R1, FI1 ;初始化Fi(0)~Fi(2)
 12
 13
    LD R2, FI1
 14
    LD R3, FI2
 15
    ADD RO, RO, #-1
                         ;R0-=1
 16 | START ADD RO, RO, #-1 ;RO-=1
 17
    BRnz OUTPUT
    ADD R4, R1, R1
 18
                          ;R1×2
 19
    ADD R1, R2 ,#0
                          ;向前
 20
    ADD R2, R3,#0
                          ;向前
 21 ADD R3, R2 ,R4
                          ;计算
 22
    AND R3, R3, R6
                          ;取模
 23 BRnzp START
 24
    OUTPUT BRZ #2
 25 ADD R7, R2 ,#0
 26 BRnzp #1
 27
    ADD R7, R3,#0
 28 BRnzp NN
 29
    MOD .FILL #1023
 30
    FI1 .FILL #1
 31 FI2 .FILL #2
    PTR .FILL NUMBER1
 33 NUMBER1 .FILL #20
 34 NUMBER2 .FILL #0
 35 NUMBER3 .FILL #1
 36 NUMBER4 .FILL #37
    .END
```

去掉开头结尾后 初始版本代码使用了 35 行代码

• 实例运行n = 16384



4. 修改后版本

第一版本理解错了题目的意思,修改之后可以得到以下所示代码

```
1 .ORIG x3000
2 LD R6, MOD
                         ;初始化R6为1023
3 ADD R1,R1,#1
                          ;初始化Fi(0)~Fi(2)
4
   ADD R2, R2, #1
5
   ADD R3, R3, #2
6
   ADD R0, R0, #-1
                          ;R0-=1
7
   START ADD R0, R0, \#-1; R0-=1
8
   BRnz OUTPUT
9
   ADD R4, R1, R1
                          ;R1×2
10 | ADD R1, R2 ,#0
                          ;向前
   ADD R2, R3,#0
11
                           ;向前
   ADD R3, R2 ,R4
                           ;计算
```

```
13 AND R3, R3, R6 ;取模
14 BRNZP START
15 OUTPUT BRZ #2
16 ADD R7, R2, #0
17 BRNZP #1
18 ADD R7, R3, #0
19 HALT
20 MOD .FILL #1023
21 NUMBER1 .FILL #930
22 NUMBER2 .FILL #1
23 NUMBER3 .FILL #1
24 NUMBER4 .FILL #710
25 .END
```

去掉开头结尾后 修改后版本代码使用了 23 行代码

• 代码解释

将 R0 , R1 , R2 寄存器分别初始化为F(0) , F(1) , F(2) ,接着每次计算得到F(n) 存在 R2 寄存器,并将F(n-1) , F(n-2) 移到 R1 , R2 寄存器,循环得到F(n) 的值,其中取模可以用&1023 替代

5. 最终版本

注意到可以通过计算到 F_{n+2} 的形式,直接取当时 R0 值来优化,同时如果更改 R0 为 R7 ,可以省略最后赋值的一行,最后版本在去掉开头结尾之后,使用了18行代码

```
1 .ORIG x3000
2 LD R6, MOD
                       ;初始化R6为1023
                      ;初始化Fi(0)~Fi(2)
3 ADD R7,R7,#1
4 ADD R2,R2,#1
5 ADD R3,R3,#2
6 START ADD RO, RO, #-1 ;RO-=1
7 BRzp #1
8 HALT
9 ADD R4, R7, R7
                    ;R7×2
10 ADD R7, R2, #0
                       ;向前
11 ADD R2, R3, #0
                       ;向前
12 ADD R3, R2 ,R4
                       ;计算
13 AND R3, R3, R6 ;取模
14 BRnzp START
15 MOD .FILL #1023
16 NUMBER1 .FILL #930
17 NUMBER2 .FILL #1
18 NUMBER3 .FILL #1
19 NUMBER4 .FILL #710
20 .END
```

6. 实验思考

- 更加深入了解了 LC-3 的指令
- 学会使用汇编语言实现简单的程序