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 `R0` 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

```cpp
#include <iostream>
int main()
{
  short n;
  std::cin >> n;
  int Fi[3];
  Fi[0] = 1;
  Fi[1] = 1;
  Fi[2] = 2;
  n -= 2;
  for (; n > 0; n--)
  {
      int temp = Fi[0] << 1;
      Fi[0] = Fi[1];
      Fi[1] = Fi[2];
      Fi[2] = (Fi[1] + temp) % 1024;
  }
  if (n < 0)
      std::cout << Fi[1] << std::endl;
  else
      std::cout << Fi[2] << std::endl;
  return 0;
}
```

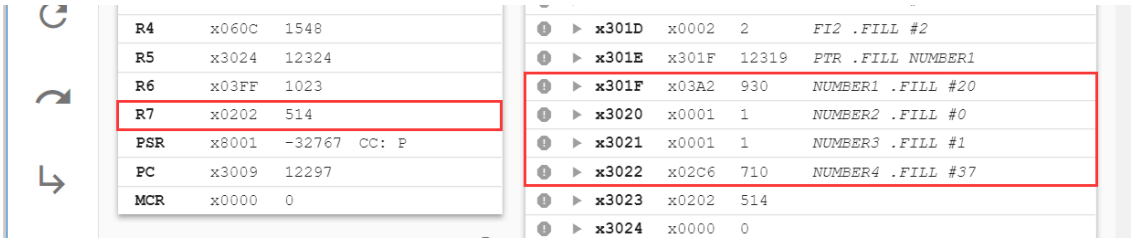
3. 初始版本代码

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

```
ADD R5, R5, #1          ;R5+=1
AND R0, R5, #4          ;R0=R5&100(这里利用偏移四次之后产生进位)
BRz return              ;没有进位就继续算
HALT
FUNCTION LD R1, FI1     ;初始化Fi(0)~Fi(2)
LD R2, FI1
LD R3, FI2
ADD R0, R0, #-1         ;R0-=1
START ADD R0, R0, #-1   ;R0-=1
BRnz OUTPUT
ADD R4, R1, R1          ;R1×2
ADD R1, R2 ,#0          ;向前
ADD R2, R3 ,#0          ;向前
ADD R3, R2 ,R4          ;计算
AND R3, R3 ,R6          ;取模
BRnzp START
OUTPUT BRz #2
ADD R7, R2 ,#0
BRnzp #1
ADD R7, R3 ,#0
BRnzp NN
MOD .FILL #1023
FI1 .FILL #1
FI2 .FILL #2
PTR .FILL NUMBER1
NUMBER1 .FILL #20
NUMBER2 .FILL #0
NUMBER3 .FILL #1
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

```
.ORIG x3000
LD R6, MOD              ;初始化R6为1023
ADD R1,R1,#1             ;初始化Fi(0)~Fi(2)
ADD R2,R2,#1
ADD R3,R3,#2
ADD R0, R0, #-1         ;R0-=1
START ADD R0, R0, #-1   ;R0-=1
BRnz OUTPUT
ADD R4, R1, R1          ;R1×2
ADD R1, R2 ,#0          ;向前
ADD R2, R3 ,#0          ;向前
ADD R3, R2 ,R4          ;计算
```

```
AND R3, R3 ,R6          ;取模
BRnzp START
OUTPUT BRz #2
ADD R7, R2 ,#0
BRnzp #1
ADD R7, R3 ,#0
HALT
MOD .FILL #1023
NUMBER1 .FILL #930
NUMBER2 .FILL #1
NUMBER3 .FILL #1
NUMBER4 .FILL #710
.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

```
.ORIG x3000
LD R6, MOD                ;初始化R6为1023
ADD R7,R7,#1              ;初始化Fi(0)~Fi(2)
ADD R2,R2,#1
ADD R3,R3,#2
START ADD R0, R0, #-1     ;R0-=1
BRzp #1
HALT
ADD R4, R7, R7            ;R7×2
ADD R7, R2, #0            ;向前
ADD R2, R3, #0            ;向前
ADD R3, R2 ,R4            ;计算
AND R3, R3 ,R6            ;取模
BRnzp START
MOD .FILL #1023
NUMBER1 .FILL #930
NUMBER2 .FILL #1
NUMBER3 .FILL #1
NUMBER4 .FILL #710
.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