Lab3 实验报告

实验题目

1. 阅读收到的代码，理解其实现过程
2. 根据收到代码的末尾四行，猜测代码编写者的学号
3. 优化收到的代码
4. 将对代码的建议反馈给代码编写同学

Task read and understand

该同学的实现思路是递推求解，不断运用递推公式更新寄存器内容，最终求得 n 对应的斐波那契值。

Task guess

使用以下程序由斐波那契值求出原数：

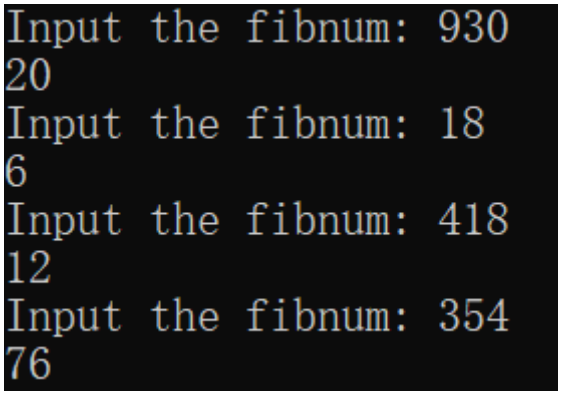
```
#include<iostream>
#include<cstdio>

using namespace std;
short int R0,R1,R2,R3,R4,R5,R6,R7;
short int fa[100]={1,1, 2, 4, 6, 10, 18, 30, 50, 86, 146, 246, 418, 710, 178,
1014, 386, 742, 722, 470};
short int fb[200]={514, 614, 594, 598, 802, 966, 114, 694, 578, 806, 146, 278,
866, 134, 690, 374, 642, 998, 722, 982, 930, 326, 242, 54, 706, 166, 274, 662,
994, 518, 818, 758, 770, 358, 850, 342, 34, 710, 370, 438, 834, 550, 402, 22
,98, 902, 946, 118, 898, 742, 978, 726, 162, 70 ,498, 822, 962, 934, 530, 406,
226, 262, 50, 502, 2, 102, 82, 86, 290, 454, 626, 182, 66, 294, 658, 790, 354,
646, 178, 886, 130, 486, 210, 470, 418, 838, 754, 566, 194, 678, 786, 150, 482,
6, 306, 246, 258, 870, 338, 854, 546, 198, 882, 950, 322, 38, 914, 534, 610,
390, 434, 630, 386, 230, 466, 214, 674, 582, 1010, 310, 450, 422, 18, 918, 738,
774, 562, 1014};

int main()
{
    int check;
    for(int k=1;k<=4;k++){
        cout<<"Input the fibnum: ";
        cin>>check;
        for(int i=0;i<=100;i++){
            R0=i;
            if(R0<20) R7=fa[R0];
            else{
                R0=R0%128;
                R7=fb[R0];
            }
            if(R7==check){
                cout<<i<<endl;
                break;
            }
        }
    }
```

```
    return 0;
}
```

可由 fib.txt 的后四行反推出这位同学的学号：



学号为：PB20061276

Task optimize

通过打表可以发现，此斐波那契数列存在循环，在第 21 个数字开始，每 128 个数循环一次

所以可以将可能出现的数字 .FILL 进内容，先判断原数是否大于 20，如果小于 20，进行特殊处理。如果大于 20，通过对原数模 128，再根据已经存入的值，即可求出其斐波那契值

优化：因为 128 用二进制表示为：10000000. 所以对 128 取模，即为保留低七位，故可用原数和二进制数 1111111 进行与运算，实现模 128.

代码为：

```
.ORIG x3000
    LD R4,NUM1
    ADD R1,R0,R4
    BRnz Spe

    LD R2,NUM2
    LD R4,NUM4

MOD AND R0,R0,R4

Spe ADD R2,R2,R0

LDD LEA R4,#3
    ADD R2,R2,R4
    LDR R7,R2,#0

    TRAP X25

.FILL    #1
.FILL    #1
.FILL    #2
.FILL    #4
.FILL    #6
.FILL    #10
.FILL    #18
.FILL    #30
.FILL    #50
.FILL    #86
```

```
.FILL    #146
.FILL    #246
.FILL    #418
.FILL    #710
.FILL    #178
.FILL    #1014
.FILL    #386
.FILL    #742
.FILL    #722
.FILL    #470
.FILL    #514
.FILL    #614
.FILL    #594
.FILL    #598
.FILL    #802
.FILL    #966
.FILL    #114
.FILL    #694
.FILL    #578
.FILL    #806
.FILL    #146
.FILL    #278
.FILL    #866
.FILL    #134
.FILL    #690
.FILL    #374
.FILL    #642
.FILL    #998
.FILL    #722
.FILL    #982
.FILL    #930
.FILL    #326
.FILL    #242
.FILL    #54
.FILL    #706
.FILL    #166
.FILL    #274
.FILL    #662
.FILL    #994
.FILL    #518
.FILL    #818
.FILL    #758
.FILL    #770
.FILL    #358
.FILL    #850
.FILL    #342
.FILL    #34
.FILL    #710
.FILL    #370
.FILL    #438
.FILL    #834
.FILL    #550
.FILL    #402
.FILL    #22
.FILL    #98
.FILL    #902
.FILL    #946
.FILL    #118
```

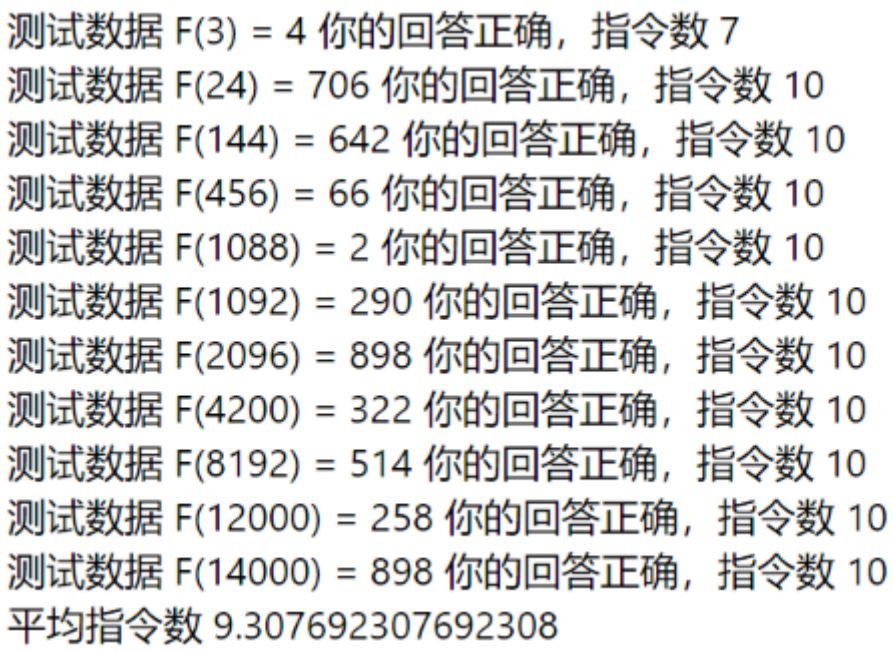
```
.FILL    #898
.FILL    #742
.FILL    #978
.FILL    #726
.FILL    #162
.FILL    #70
.FILL    #498
.FILL    #822
.FILL    #962
.FILL    #934
.FILL    #530
.FILL    #406
.FILL    #226
.FILL    #262
.FILL    #50
.FILL    #502
.FILL    #2
.FILL    #102
.FILL    #82
.FILL    #86
.FILL    #290
.FILL    #454
.FILL    #626
.FILL    #182
.FILL    #66
.FILL    #294
.FILL    #658
.FILL    #790
.FILL    #354
.FILL    #646
.FILL    #178
.FILL    #886
.FILL    #130
.FILL    #486
.FILL    #210
.FILL    #470
.FILL    #418
.FILL    #838
.FILL    #754
.FILL    #566
.FILL    #194
.FILL    #678
.FILL    #786
.FILL    #150
.FILL    #482
.FILL    #6
.FILL    #306
.FILL    #246
.FILL    #258
.FILL    #870
.FILL    #338
.FILL    #854
.FILL    #546
.FILL    #198
.FILL    #882
.FILL    #950
.FILL    #322
.FILL    #38
```

```
.FILL    #914
.FILL    #534
.FILL    #610
.FILL    #390
.FILL    #434
.FILL    #630
.FILL    #386
.FILL    #230
.FILL    #466
.FILL    #214
.FILL    #674
.FILL    #582
.FILL    #1010
.FILL    #310
.FILL    #450
.FILL    #422
.FILL    #18
.FILL    #918
.FILL    #738
.FILL    #774
.FILL    #562
.FILL    #1014

NUM1 .FILL #-19
NUM2 .FILL #20
NUM3 .FILL #-128
NUM4 .FILL #127

.END
```

运行效果为：



运行的指令数为：当原数小于 20 时，需要 7 条指令；当原数大于等于 20 时，需要 10 条指令

优化过程

- 取模运算进行了优化，在之前使用减法实现取模，在原数较大时会运行较多数量的指令。通过观察模数 128 的特性，使用与运算实现取模，可以达到优化效果

  优化前代码为（省略 .FILL 部分）

```
.ORIG x3000
    LD R4,NUM1
    ADD R1,R0,R4
    BRnz Spe
    ADD R3,R0,#0

    LD R4,NUM2
    ADD R2,R2,R4
    LD R4,NUM3

MOD ADD R0,R3,#0
    ADD R3,R0,R4
    BRzp MOD

Spe ADD R1,R0,#0
    BRz LDD

Loop    ADD R2,R2,#1
    ADD R1,R1,#-1
    BRp Loop

LDD LEA R4,#3
    ADD R2,R2,R4
    LDR R7,R2,#0

    TRAP X25
```

- 对空间的优化：最初打表的设想是将所有可能的数字全部 .FILL 进内存，这需要 $16 \text{ x } 10^4$ bits 内存。而通过找规律，发现数据存在循环，除起始的 20 个数字，每 128 个数字循环一次。可根据这条规律，运用取模运算，节省大量的内存空间。

实验总结

1. 本次实验阅读了另一位同学的代码，发现在没有注释的情况下理解一份汇编语言不是一件易事，即使已经知道了这份代码的目的
2. 打表找规律是一项技能，在本次实验中得到了练习。如果没有通过规律，那么本次实验的 .FILL 会达到上万行，会非常占用空间。而通过规律，虽然多执行了了 4 次指令，但只需要 148 条空间即可存储打表数据，很大程度地节省了空间