Lab6 实验报告

实验题目

使用高级语言完成前五次实验的七个程序

代码实现

lab0l

用累加的方法实现乘法，代码如下：

```
int main()
{
    cin>>R0>>R1;
    while(R1){
        R7=R7+R0;
        R1--;
    }
    cout<<R7<<endl;
    return 0;
}
```

lab0p

采用模拟列竖式的方式实现乘法，代码如下：

```
int main()
{
    cin>>R0>>R1;
    R2=1;
    while(R2){
        R3=R1&R2;
        if(R3) R7=R7+R0;
        R0=R0+R0;
        R2=R2+R2;
    }
    cout<<R7<<endl;
    return 0;
}
```

fib

使用递推的方式求解斐波那契值，c++ 中的 if-else 语句比汇编语言中的跳转设计更易于理解，代码如下：

```
int main()
{
    cin>>R0;
    R1=1,R2=1,R3=2;
    if(R0==0){
        R7=R1;
```

```
        cout<<R7<<endl;
        return 0;
    }
    R0=R0-1;
    if(R0==0){
        R7=R2;
        cout<<R7<<endl;
        return 0;
    }
    R0=R0-1;
    if(R0==0){
        R7=R3;
        cout<<R7<<endl;
        return 0;
    }
    while(R0){
        R7=(2*R1+R3)%1024;
        R1=R2;
        R2=R3;
        R3=R7;
        R0--;
    }
    cout<<R7<<endl;
    return 0;
}
```

fib-opt

根据找到的循环规律，进行打表，代码如下：

```
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()
{
    cin>>R0;
    if(R0<20) R7=fa[R0];
    else{
        R0=R0%128;
        R7=fb[R0];
    }
    cout<<R7<<endl;
    return 0;
}
```

rec

开辟一个数组，即为 mem，模拟 LC3 中的内存，然后根据机器码的逻辑完成代码：

```
int main()
{
    mem[0x3019]=5;
    R2=0x300f;
    R0=0;
    R7=0x3003;
    while(1){
        mem[R2]=R7;
        R2=R2+1;
        R0=R0+1;
        R1=mem[0x3019];
        R1--;
        mem[0x3019]=R1;
        if(R1==0){
            while(R7!=0x3003){
                R2=R2-1;
                R7=mem[R2];
            }
            break;
        }
        R7=0x300B;
    }
    cout<<R0<<" "<<R1<<" "<<R2<<" "<<R3<<" "<<R4<<" "<<R5<<" "<<R6<<" "<<R7<<" "
<<endl;
    return 0;
}
```

mod

对给定的数字进行模 7 运算，通过模 8 实现：

```
int main()
{
    cin>>R1;
    while(1){
        R2=1,R3=8,R4=0;
        while(R3){
            R5=R3&R1;
            if(R5){
                R4=R4+R2;
            }
            R2=R2+R2;
            R3=R3+R3;
        }
        R2=R1&7;
        R1=R2+R4;
        if(R1<7){
            cout<<R1<<endl;
            break;
        }
        else if(R1==7){
            R1=R1-7;
            cout<<R1<<endl;
```

```
            break;
        }
    }
    return 0;
}
```

prime

朴素的素数筛：

```
int main()
{
    cin>>R0;
    R1=1;
    for(int i=2;i<=sqrt(R0);i++){
        if(R0%i==0){
            R1=0;
            break;
        }
    }
    cout<<R1;
    return 0;
}
```

思考题

1. **how to evaluate the performance of your own high-level language programs**

- 调用库，可以记录程序运行时间，但是这种方法的求出的时间是毫秒级别的，许多程序单次运行的时间在 1ms 以下，应对办法是使用循环，将程序重复运行 10000 次，然后将总时间除以 10000，即可得到运行时间
- 通过上述方法可得，c++ 的运行速度是比较快的

2. **why is a high-level language easier to write than LC3 assembly**

- 首先，高级语言提供的操作较多，比如循环操作，子程序调用，if-else 语句。这些操作在 LC3 汇编语言中只能通过跳转指令来实现，使得在理解上缺少直观性
- 其二，高级语言提供的运算指令较多，比如取模运算、乘法运算、开根号运算等，且可以在一条语句中实现多条运算
- 其三，高级语言的代码实现顺序可以与代码书写顺序相契合，而 LC3 的汇编语言受制于指令的特点，代码实现的顺序与代码书写的顺序会不一致，给书写和阅读带了困难
- 其四，高级语言可以定义变量，而且可以自行命名变量名称，给书写和阅读带来便利。LC3 的中间变量只能借助八个通用寄存器来存储，需要代码编写者额外记忆不同寄存器存储的什么变量，给书写和阅读增加难度

3. **what instructions do you think need to be added to LC3? (You can think about the previous experiments and what instructions could be added to greatly simplify the previous programming)**

- 比较大小操作：可以定义为 CMP，它的汇编代码示例为：CMP,R1,R2，意为比较 R1 和 R2 两个寄存器中存储的数值的大小，根据比较结果置条件码：若 R1>R2，P=1,Z=0,N=0；若 R1 = R2, P=0,Z=1,N=0；若 R1 < R2, P=0,Z=0,N=1

  他的好处在于省去了对一个寄存器的数取反加一的操作，并且节省了一个寄存器。有利于根据两束大小关系进行跳转，或者取两数之差的绝对值

- 右移指令：可以定义为 MOR，它的汇编代码示例为：MOR R1,#1，即将存储在 R1 中的值右移1位，最高位补符号位，右移的位数由指令给出。

4. **is there anything you need to learn from LC3 for the high-level language you use?**

- 学习高级语言与汇编语言的对应，一句高级语言可能需要很多条汇编指令去实现，通过理解他们之间的对应关系，应该有助于更深入地理解和应用高级语言