ICS Lab6 实验报告

张艺耀 PB2011630

请思考以下问题：

1. 如何评估自己用用高级语言编写的程序的性能
2. 为什么高级语言比汇编代码编写更为简单
3. 你认为LC-3应增加什么指令
4. 学习LC-3对你用高级语言编程有什么启发

Lab0l

```
#include<iostream>
using namespace std;

int main(){
    short r0 = 0, r1 = 0, r7 = 0;
    cin >> r0 >> r1;
    while(r0 != 0){
        r7 = r7 + r1;
        r0 -= 1;
    }
    cout << r7;
}
```

Lab0p

循环展开版本

```
#include<iostream>
using namespace std;

int main(){
    short r0 = 0, r1 = 0, r3 = 0, r4 = 0, r7 = 0;
    cin >> r0 >> r1;
    r4 = r1 & 1;
    if(r4 != 0)
        r7 = r7 + r0;
    r0 <<= 1;

    r4 = r1 & 2;
```

```
    if(r4 != 0)
        r7 = r7 + r0;
    r0  <<= 1;

    r4 = r1 & 4;
    if(r4 != 0)
        r7 = r7 + r0;
    r0  <<= 1;

    r3 = 8;

    r4 = r1 & r3;
    if(r4 != 0)
        r7 = r7 + r0;
    r0 <<= 1;
    r3 <<= 1;

    r4 = r1 & r3;
    if(r4 != 0)
        r7 = r7 + r0;
    r0 <<= 1;
    r3 <<= 1;

    r4 = r1 & r3;
    if(r4 != 0)
        r7 = r7 + r0;
    r0 <<= 1;
    r3 <<= 1;

    r4 = r1 & r3;
    if(r4 != 0)
        r7 = r7 + r0;
    r0 <<= 1;
    r3 <<= 1;

    r4 = r1 & r3;
    if(r4 != 0)
        r7 = r7 + r0;
    r0 <<= 1;
    r3 <<= 1;

    r4 = r1 & r3;
    if(r4 != 0)
        r7 = r7 + r0;
    r0 <<= 1;
    r3 <<= 1;

    r4 = r1 & r3;
    if(r4 != 0)
        r7 = r7 + r0;
    r0 <<= 1;
    r3 <<= 1;
```

```
65
66      r4 = r1 & r3;
67      if(r4 != 0)
68          r7 = r7 + r0;
69      r0 <<= 1;
70      r3 <<= 1;
71
72      r4 = r1 & r3;
73      if(r4 != 0)
74          r7 = r7 + r0;
75      r0 <<= 1;
76      r3 <<= 1;
77
78      r4 = r1 & r3;
79      if(r4 != 0)
80          r7 = r7 + r0;
81      r0 <<= 1;
82      r3 <<= 1;
83
84      r4 = r1 & r3;
85      if(r4 != 0)
86          r7 = r7 + r0;
87      r0 <<= 1;
88      r3 <<= 1;
89
90      r4 = r1 & r3;
91      if(r4 != 0)
92          r7 = r7 + r0;
93      r0 <<= 1;
94      r3 <<= 1;
95
96      r4 = r1 & r3;
97      if(r4 != 0)
98          r7 = r7 + r0;
99
100     cout << r7;
101     return 0;
102 }
```

fib

```
1 #include<stdio.h>
2 int main(){
3     int n;
4     while(1){
5         scanf("%d", &n);
6             if(!n) break;
7         short r1 = 1, r2 = 1, r3 = 2, r4, r7 = 0;
8         short r5 = 0x03ff;
9         if(n == 1||n == 2) r7 = 1;
```

```
        for(int i = 2; i < n; i++){
            r4 = r2;
            r2 = r3;
            r3 = (2*r1 + r3)&r5;
            r1 = r4;
            r7 = r3;
        }
        printf("N = %d Result: %d %4x\n",n, r7, r7);
        r1 = r2 = 1; r3 = 2;
    }
    return 0;
}
```

这里接下来的优化就是直接全部将R3替换成R7存储结果以节省行数且略微减少指令数。

fib-opt

```
#include<stdio.h>
int main(){
    int n;
    while(1){
        scanf("%d", &n);
            if(!n) break;
        short r1 = 1, r2 = 1, r7 = 2, r4 = 0;
        short r5 = 0x03ff;
        if(n == 1||n == 2) r7 = 1;
        for(int i = 2; i < n; i++){
            r4 = r2;
            r2 = r7;
            r7 = (2*r1 + r7)&r5;
            r1 = r4;
        }
        printf("N = %d Result: %d %4x\n",n, r7, r7);
        r1 = r2 = 1; r3 = 2;
    }
    return 0;
}
```

rec

本题为模拟递归调用函数栈

```cpp
#include<iostream>
using namespace std;

void recur(int & dep){
    if(--dep)
        recur(dep);
}

int main(){
    int dep;
    cin >> dep;
    recur(dep);
}
```

mod

```cpp
#include<iostream>
using namespace std;
short r4;

void mod(short r1){
    short r2 = 1, r3 = 8, r5;
    r4 = 0;
    do{
        r5 = r1 & r3;
        if(r5 != 0)
            r4 = r2 + r4;
        r2 += r2;
        r3 += r3;
    }while(r3 != 0);
}

int main(){
    short r0, r1, r2, r3, r5, r6, r7;
    r0 = r1 = r2 = r3 = r5 = r6 = r7 = 0;
    cin >> r1;
    do{
        mod (r1);
        r2 = r1 & 7;
        r1 = r2 + r4;
        r0 = r1 - 7;
    }while(r0 > 0);
    if(r1 == 7) cout << 0;
    else
        cout << r1;
}
```

prime

```cpp
#include<iostream>
using namespace std;
int r0, r1;
int judge(int r0) {
    int i = 2;
    r1 = 1;
    while (i * i <= r0) {
        if (r0 % i == 0) {
            r1 = 0;
            break;
        }
        i++;
    }
    return r1;
}

int main(){
  cin >> r0;
  cout << judge(r0);
}
```

总结与思考

性能：

只需考虑时间复杂度即汇编程序执行指令的次数，为此可将程序分块，逐块分析各个块的时间复杂度，再去取最高值。比如素数程序的时间复杂度，乘法部分为O(n)，取模部分为O(nlogn)，我们取O(nlogn)作为整个程序的时间复杂度。

高级语言的便利：

因为高级语言为我们省去了繁杂对寄存器的操作过程和地址的计算，因为寄存器的作用在代码中不加注释就难以分辨。如在C++中我们通过简单的push pop等操作就可以完成栈入栈出，而这在汇编语言中需要很多条语句，涉及到很多寄存器，操作繁杂。

LC-3的改进：

应增加基本的运算指令如乘法，移位指令，这样可以为运算提供很多便利。

启发：

明白了高级语言是如何转化为最基本的01串被机器所识别和运行的，了解到了高级语言的底层实现。