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

Lab01

```
1 #include<iostream>
   using namespace std;
3
   int main(){
5
       short r0 = 0, r1 = 0, r7 = 0;
       cin >> r0 >> r1;
6
7
       while(r0 != 0){
8
           r7 = r7 + r1;
9
           r0 -= 1;
10
11
       cout << r7;
12 }
```

Lab0p

循环展开版本

```
1 #include<iostream>
2 using namespace std;
3
4 int main(){
5
        short r0 = 0, r1 = 0, r3 = 0, r4 = 0, r7 = 0;
6
        cin >> r0 >> r1;
7
        r4 = r1 & 1;
8
        if(r4 != 0)
9
            r7 = r7 + r0;
        r0 <<= 1;
10
11
12
        r4 = r1 & 2;
```

```
13 if(r4 != 0)
14
           r7 = r7 + r0;
15
        r0 <<= 1;
16
17
        r4 = r1 & 4;
18
        if(r4 != 0)
19
          r7 = r7 + r0;
20
        r0 <<= 1;
21
22
        r3 = 8;
23
24
        r4 = r1 & r3;
25
        if(r4 != 0)
          r7 = r7 + r0;
26
27
        r0 <<= 1;
28
        r3 <<= 1;
29
30
        r4 = r1 & r3;
31
        if(r4 != 0)
32
          r7 = r7 + r0;
33
        r0 <<= 1;
34
        r3 <<= 1;
35
36
        r4 = r1 & r3;
37
        if(r4 != 0)
          r7 = r7 + r0;
38
39
        r0 <<= 1;
40
        r3 <<= 1;
41
42
        r4 = r1 & r3;
43
        if(r4 != 0)
44
         r7 = r7 + r0;
45
        r0 <<= 1;
46
        r3 <<= 1;
47
        r4 = r1 & r3;
48
49
        if(r4 != 0)
50
          r7 = r7 + r0;
51
        r0 <<= 1;
52
        r3 <<= 1;
53
54
        r4 = r1 & r3;
55
        if(r4 != 0)
56
           r7 = r7 + r0;
57
        r0 <<= 1;
58
        r3 <<= 1;
59
60
        r4 = r1 & r3;
61
        if(r4 != 0)
          r7 = r7 + r0;
62
63
        r0 <<= 1;
64
        r3 <<= 1;
```

```
65
66
         r4 = r1 & r3;
         if(r4 != 0)
67
68
            r7 = r7 + r0;
69
         r0 <<= 1;
70
         r3 <<= 1;
71
         r4 = r1 & r3;
72
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){
           scanf("%d", &n);
5
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;
```

```
10
            for(int i = 2; i < n; i++){
11
                r4 = r2;
                r2 = r3;
12
13
                r3 = (2*r1 + r3)&r5;
14
                r1 = r4;
15
                r7 = r3;
16
            }
            printf("N = %d Result: %d %4x\n",n, r7, r7);
17
18
            r1 = r2 = 1; r3 = 2;
19
        }
20
        return 0;
21 }
```

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>
 2
    int main(){
 3
        int n;
 4
        while(1){
 5
            scanf("%d", &n);
 6
                if(!n) break;
            short r1 = 1, r2 = 1, r7 = 2, r4 = 0;
 7
 8
            short r5 = 0x03ff;
 9
            if(n == 1 | | n == 2) r7 = 1;
            for(int i = 2; i < n; i++){
10
11
                r4 = r2;
12
                r2 = r7;
13
                r7 = (2*r1 + r7)&r5;
14
                r1 = r4;
15
            }
            printf("N = %d Result: %d %4x\n",n, r7, r7);
16
17
            r1 = r2 = 1; r3 = 2;
18
        }
19
        return 0;
20 }
```

rec

本题为模拟递归调用函数栈

```
1 #include<iostream>
2
   using namespace std;
 3
 4 void recur(int & dep){
 5
        if(--dep)
           recur(dep);
 6
7
   }
8
9
   int main(){
10
        int dep;
11
        cin >> dep;
        recur(dep);
12
13 }
```

mod

```
1 #include<iostream>
   using namespace std;
 3 short r4;
 4
5
   void mod(short r1){
        short r2 = 1, r3 = 8, r5;
 6
7
        r4 = 0;
8
        do{
9
            r5 = r1 & r3;
            if(r5 != 0)
10
11
                r4 = r2 + r4;
12
            r2 += r2;
13
            r3 += r3;
14
        }while(r3 != 0);
15
16
17
   int main(){
18
        short r0, r1, r2, r3, r5, r6, r7;
        r0 = r1 = r2 = r3 = r5 = r6 = r7 = 0;
19
20
        cin >> r1;
21
        do{
22
            mod (r1);
            r2 = r1 & 7;
23
24
            r1 = r2 + r4;
25
            r0 = r1 - 7;
26
        }while(r0 > 0);
27
        if(r1 == 7) cout << 0;
28
        else
29
           cout << r1;
30 }
```

prime

```
1 #include<iostream>
2 using namespace std;
3 int r0, r1;
 4 int judge(int r0) {
5
       int i = 2;
       r1 = 1;
 6
7
       while (i * i <= r0) {
8
            if (r0 % i == 0) {
9
               r1 = 0;
10
                break;
11
            }
12
           i++;
13
        }
14
        return r1;
   }
15
16
17
   int main(){
18
    cin >> r0;
19
     cout << judge(r0);</pre>
20 }
```

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。