lab6 实验报告

李远航
PB20000137

1. 实验内容

将以往每一次的实验的机器码或汇编码程序转化成高级语言程序

2. 设计思路及代码

(1)lab1 L version

使用机器码处理乘法的思路是先判断正负，然后通过把加法转换成乘法来计算，具体代码如下:

```
#include <iostream>
short times(short a, short b)
{
    short flag = -1;
    if (b < 0)
        flag = 1;
    short answer = 0;
    while (b != 0)
    {
        answer += a;
        b += flag;
    }
    return (-flag) * answer;
}
int main()
{
    short a, b;
    std::cin >> a >> b;
    std::cout << times(a, b) << std::endl;
    return 0;
}
```

(2)lab1 P version

该版本的思路是使用移位运算符进行运算，由于高级语言有移位运算符，因此可以大大简化代码

```
#include <iostream>
short times(short a, short b)
{
    if (b == 0)
        return 0;
    int flag = 0;
    if (b < 0)
    {
        flag = 1;
        b = -b;
    }
    short answer = 0;
    while (b)
    {
```

```
        if (b & 1)
            answer += a;
        a = a << 1;
        b = b >> 1;
    }
    if (flag == 1)
        answer = -answer;
    return answer;
}
```

(3)lab2 fibonacci

使用循环的方式，逐步向后计算答案

```
#include <iostream>
int main()
{
    short n;
    std::cin >> n;
    short Fi[3];
    Fi[0] = 1;
    Fi[1] = 1;
    Fi[2] = 2;
    n -= 2;
    for (; n > 0; n--)
    {
        short 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;
}
```

(4)lab3 fibonacci

优化程序直接使用打表的方式

```
#include <bits/stdc++.h>
int main()
{
   short F[16385];
    F[0] = 1;
    F[1] = 1;
    //……
    short n;
    std::cin >> n;
    std::cout << F[n] << std::endl;
    return 0;
}
```

(5)lab4 task1 rec

程序目的是让寄存器变成定好的状态(本质上是一个递归,通过栈来存地址返回)，高级语言可以直接赋值

```
#include <bits/stdc++.h>
short reg[8];
void init()
{
    reg[0] = 0x5;
    reg[1] = 0x0;
    reg[2] = 0x300f;
    reg[3] = 0x0;
    reg[4] = 0x0;
    reg[5] = 0x0;
    reg[6] = 0x0;
    reg[7] = 0x3003;
}
int main()
{
    init();
    return 0;
}
```

(6)lab4 task2 mod

取模 7 的余数的思路是$X \equiv 8 \times x + R \equiv x + R \pmod 7$，据此可得到代码

```
#include <bits/stdc++.h>
int main()
{
    short num;
    std::cin >> num;
    while (num >= 7)
    {
        num = num / 8 + num % 8;
        if (num == 7)
            num -= 7;
    }
    std::cout << num << std::endl;
    return 0;
}
```

(7)lab5 prime

判断一个数字是不是素数，即判断比它小的数字(除 1)是否是他的因子

```
#include <bits/stdc++.h>
short judge(int r0)
{
    short i = 2;
    short r1 = 1;
    while (i * i <= r0)
    {
        if (r0 % i == 0)
        {
            r1 = 0;
            break;
```

```
        }
        i++;
    }
    return r1;
}
int main()
{
    short num;
    std::cin >> num;
    std::cout << judge(num) << std::endl;
    return 0;
}
```

3. 实验收获及感想

- 可以通过代码长度以及`clock()`获取程序执行时间(执行多次取时间)的方式来判断代码的表现
- 高级语言拥有更多的关键字，函数，分支结构，因此比 LC-3 汇编更加容易编写
- 我认为 LC-3 应该加入移位指令，乘法指令，除法指令
- 在使用高级语言的同时，要思考相关运算的底层实现，尽管高级语言便利了我们的编写，但我们同时也要掌握其底层的原理