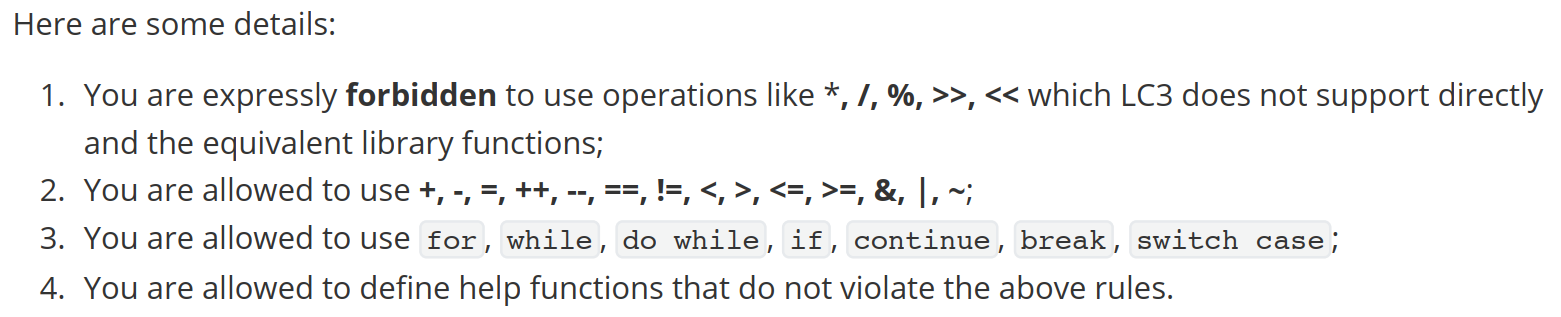
**Lab6 实验报告**

*PB21071416 张郑飞扬*

*2023.1.7*

一、实验目的（Purpose）

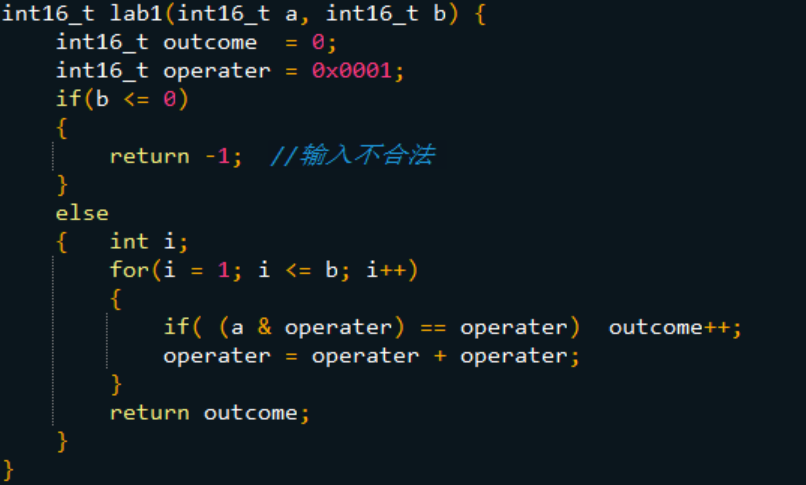
利用C/C++语言，对之前用机器码或汇编语言完成的五个实验重新编写一遍代码。要求该实验中用到的算法和之前实验中的算法必须一致。

需要注意的细节限制：

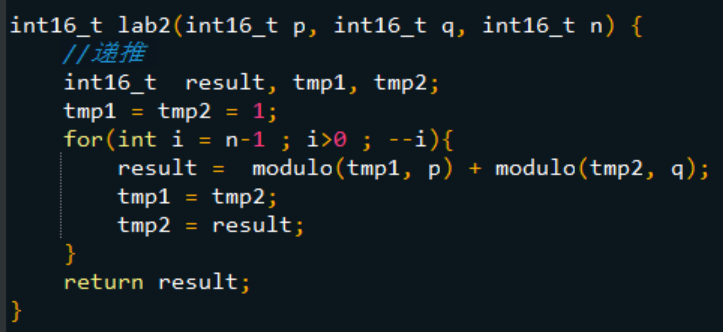
二、实验原理（Principles）

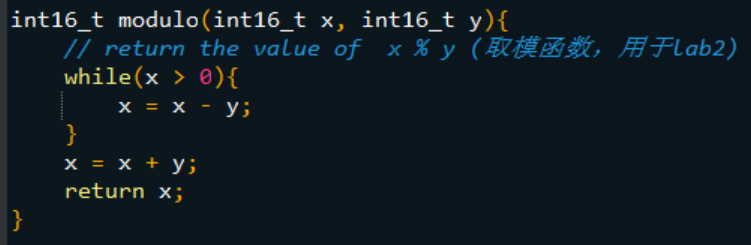
lab1：计算给定的16位二进制数a中，后b位有多少个1

由于规定不能使用移位等LC-3不直接支持的操作，所以我们考虑仅用按位与（&）操作来实现。算法和使用机器码时一样。

即设置一个操作数（operater），初始化为0x0001，与a进行按位与运算，如果a的最后一位是1，那么结果为0x0001。用代码表达即为 （a & operater == operater）。这样，我们完成了对最后一位的判断。如有必要，接下来进行对倒数第二位的判断。此时我们只需要将operater翻倍，即 operater \* 2，再重复进行上述操作即可。另外，设置一个计数器，记录有多少个1，作为返回值即可。具体代码如下：

lab2：一个斐波那契数列的变式

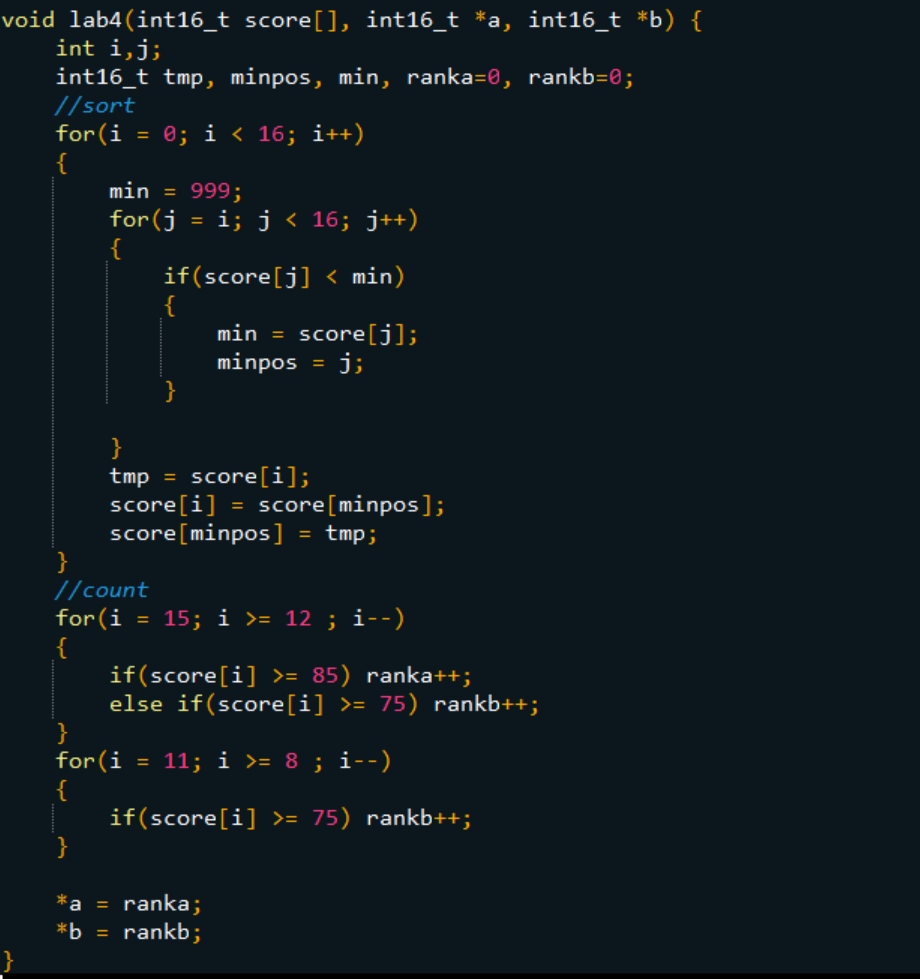
在用汇编语言编写该程序时，本人采用了递推的方法。故此处亦采用递推方法，这种方法也是比较高效的，在数据比较大时得到结果的效率比递归要优秀。由于已经给出了递推式，直接按照递推式来编写代码即可。具体代码如下：

但需要注意的是，要求不能直接使用取模运算符“%”，所以需要自己编写一个取模运算函数（modulo），这也比较简单，假如我们要得到x%y的值，只需要将x循环减去y，直到x变为负数，再加上一次y即可得到结果。具体代码如下：

lab3：最长连续子串长度

该程序需要求最长连续子串的长度，我们只需要设置两个变量tmp和max，从头开始，每次比较字符串的两个相邻字符，如果相同，那么tmp++，如果不同，那么tmp和max需要进行比较，若tmp > max，则用tmp刷新max的值。注意每次出现相邻两个字符不同时，tmp都需要重置为1。另外，需要注意遍历完字符串后，还要进行一次tmp和max的比较，因为如果最长连续子串在末尾，是无法在遍历字符串时更新到max值上的。具体代码如下：

lab4：对16个学生的分数排序并计算A/B等人数

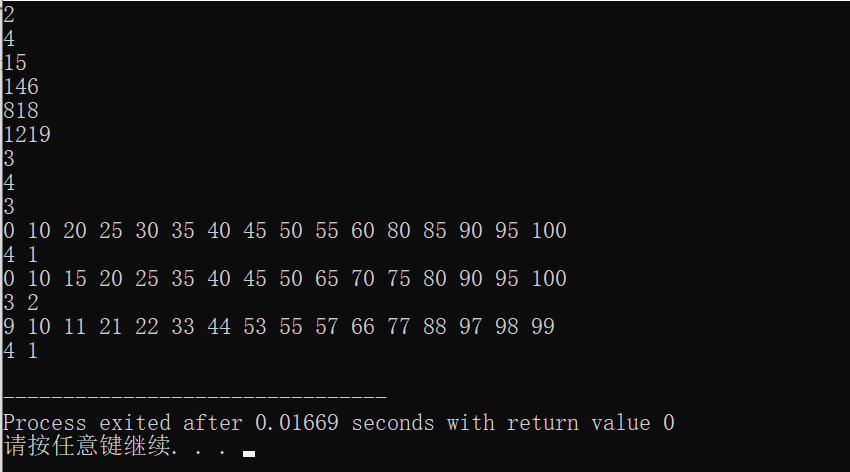
该程序考察排序和计数，在使用汇编语言写该程序时，要麻烦一些，而使用C语言时，可以更方便地完成该程序。和写汇编程序时使用同样的选择排序算法对分数进行排序，并遍历前八个同学的分数，找出A、B等同学的人数。具体代码如下：

三、实验过程（Procedure）

在编写代码时，并没有遇到太多阻碍，因为用C语言来重写这些程序相比用汇编语言，复杂度降低了很多。  
 但在对16位整型变量进行操作时，开始遇到了一些问题。一开始本人并未使用实验手册给出的代码模板，而尝试自己用C语言写，这导致printf、fprintf等函数无法正确输出16位的整型变量，而想要正确输出需要比较复杂的操作。于是本人改为采用实验手册的模板，用C++的标准输出来操作，这个问题就被顺利解决了。

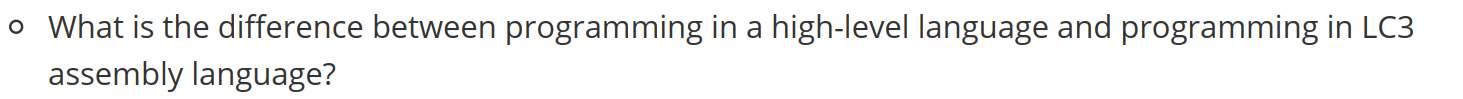
四、测试结果（Result of the test）

使用给定的测试文件，得到测试结果如下：

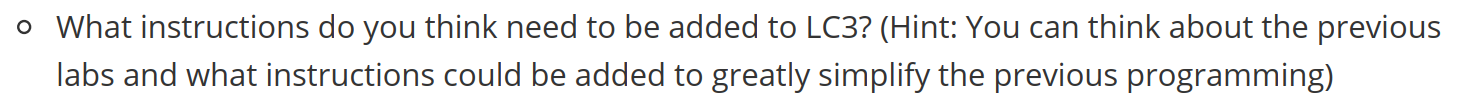


这与正确结果相符。

五、问题回答（Answer to the questions）



高级语言有更强的表达能力，可以更方便地控制数据运算和程序的控制结构，更好地描述各种算法，所以编写起来更加省时省力了。而汇编语言则需要更关注机器底层的各种细节，比如要关注寄存器值保存等问题，比较麻烦。



可以加一条除法指令，简化乘除法和取模运算。

通过LC-3，我对机器的底层运行原理有了更深刻的了解，从而在使用高级语言时知其然，也知其所以然，有时候透过一些现象看到本质，优化程序设计，提高运行效率。