Lab6

实验目的

- 使用高级语言，如C语言，编写前四次实验的程序
- 更加熟悉高级语言，熟悉汇编语言的特点
- 明白高级语言与汇编

实验原理

Lab1

- Lab1的实验较为简单，我们采取取模的方式判断各个位置是否为0，并且将非0的部分去掉，方便判断下一位置，同时我们注意到我们不能使用C语言中直接写好的取模的运算符，我们自己写了一个mod函数，方便后续使用，后面如遇到mod部分均为使用该函数，不再赘述

```
int16_t lab1(int16_t a, int16_t b) {
    // initialize
    int n = 1;
    int tem = a; //初始化tem=A，在每次取模后若结果不为零，则减去2^n，这样来实现第n位前的所有位均为0
    int result = 0;
    int loop = b;
    int t;
    // calculation
    while(loop){
        n = n + n; //考虑到汇编只能实现加法运算，这里的意思是在每次循环中都对n*2，以此来实现2^n操作
        t = tem;
        while(t >= 0){
            t = t - n; //这里的意思是在每次循环中都对tem-2^n，以此来实现第n位的取模操作
        }
        t = t + n;
        if(t) {
            result++; //如果取模不为零，将result加一
            tem = tem - t; //将第n位清零
        }
        loop--;
    }
```

```
int mod(int a, int b){
    int c = a;
    while(c >= 0){
        c = c - b;
    }
    c = c + b;
    return c;
}
```

Lab2

- 我们需要计算递归式

$$F(N) = F(N-2)\%p + F(N-1)\%q \tag{1}$$

- 我们只需要使用a_0和a_1两个变量来存储$F(N-2)$和$F(N-1)$的值，该程序用C语言来写非常简单，正确判断什么时候输出即可，并且我们的程序能很好地满足边界条件

```
int16_t lab2(int16_t p, int16_t q, int16_t n) {
    // initialize
    int a0 = 1;
    int a1 = 1;
    // calculation F(N)=F(N-2)%p+F(N-1)%q
    for(int i = 2; i <= n; i++){
        int temp = a1;
        a1 = mod(a0, p) + mod(a1, q);
        a0 = temp;
    }
    // return value
    return a1;
}
```

Lab3

- 本次实验需要找到最长的字符串，我们使用两个变量`char_pre`和`char_now`分别存储前一个字符和现在的字符，我们每次循环更新，并且我们判断前一个字符与现在的字符是否相同，如果相同则当前字符长度加一，如果不同那么我们就判断当前字符长度与最长字符相比的大小，如果需要替换，那么就替换

```
int16_t lab3(int16_t n, char s[]) {
    char char_now = s[0];
    char char_pre = s[0];
    int max = 0;
    int t;
    //int n;
    int i = 0;
```

```
    do{
        if(char_now != char_pre){
            if(t >= max){
                max = t;
                t = 0;
            }
            else{
                t = 0;
            }
        }
        t++;
        n--;
        i++;
        char_pre = char_now;
        char_now = s[i];
    }while(n>0);

    if(t >= max){
        return t;
    }
    return max;
}
```

Lab4

- 该程序我们分为三个部分，第一个部分是将传进来的程序进行排序

```
 // Sort Score from low to high
    int i;
    for(i = 0; i < 16; i++){
        for(int j = i + 1; j < 16; j++){
            if(score[i] > score[j]){
                int temp = score[i];
                score[i] = score[j];
                score[j] = temp;
            }
        }
    }
```

- 第二部分是计算成绩为A的学生，我们使用变量`count`计数

```
// Calculate top 25% and grade is more than 85, store in a
int16_t count = 0;
for(i = 15; i >= 12; i--){
  if(score[i] >= 85){
    count++;
  }
}
*a = count;
```

- 第三部分是计算成绩为B的学生，我们用`count2`计数，并且最后需要减去`count`的值

```
// Calculate top 50% and grade is more than 75, store in b
int count_b = 0;
for(i = 15; i >= 8; i--){
  if(score[i] >= 75){
    count_b++;
  }
}
*b = count_b - count;
```

实验过程

- 本次试验因为是用C语言复现前几次实验，试验难度很低，没有遇到太大的困难
- 并且我在之前每一次实验的时候都会给出一个C语言的版本用来解释我的汇编程序，所以本次试验较为容易

实验结果

本次实验采用老师的test.txt文件测试，测试结果如下

```
2
4
15
146
818
1219
3
4
3
0 10 20 25 30 35 40 45 50 55 60 80 85 90 95 100 41
0 10 15 20 25 35 40 45 50 65 70 75 80 90 95 100 32
9 10 11 21 22 33 44 53 55 57 66 77 88 97 98 99 41
```

注意的是我们使用g++进行编译，请依次执行下列的命令

```
g++ Lab6.cpp
./a.out
```

实验问题

- 高级语言我们拥有无限多的变量，不需要考虑寄存器是否够用。并且在高级语言中，我们的内存访问简单很多。高级语言我们对循环的判断也较为简单，并且程序更加简洁易懂
- 我认为需要添加减法指令，这很好实现，并且非常常用，如果我们采用ADD和NOT进行替代，我们需要执行三条指令，这大大降低了程序运行速度。同时我认为判断某一个寄存器中的值并进行跳转是非常重要的，不然我们将要浪费一个额外的指令进行这一步骤，在每次循环中都需要运行一遍，这非常浪费效率
- 是的，在高级语言的编写中，我们可以在某个变量变化多的情况下使用寄存器变量，这有助于提高我们的效率