Lab 8 report

PB22111711 陈昕琪

实验目的与内容

使用高级汇编语言（C或者C++）来实现前面实验所做过的内容

1. lab1：数输入的数字的二进制表示中的0
2. lab2：乒乓序列
3. lab3：比较字符串
4. lab4：九连环问题

逻辑设计

对于lc3来说，C语言是一种高级汇编语言，所以通过C语言的一些语句实现lc3语言。这里对每个实验都进行复习，下面将逐一说明程序。

程序代码分析

1. 实验一，输出输入的数字的二进制中有多少个0，并且需要加上自己学号的最后一位

> - 首先，初始化变量`cnt`为0，用于计数零的数量；初始化变量`rep`为16，表示循环的次数。
> - 如果`n`是偶数且小于0，则将`n`取反并加1，以确保`n`是奇数（odd）的形式。
> - 进入循环，循环执行16次。在每次循环中，判断`n`是否大于等于0，如果是，则计数器`cnt`加1；然后将`n`乘以2（相当于左移一位）。
> - 循环结束后，返回`cnt + STUDENT_ID_LAST_DIGIT`的值作为函数的返回值。

```
int16_t lab1(int16_t n) {
    // initialize
    int16_t cnt = 0;//count zeros
    int16_t rep = 16;
    if(!(n & 1) && n < 0) n = (~n) + 1;//check if n is odd, if yes then turn n
into its implement

    // calculation
    while(rep--){
        if(n >= 0) cnt++;
        n = n + n;
    }

    // return value
    return cnt + STUDENT_ID_LAST_DIGIT;
}
```

2. 实验二，输出乒乓序列

> - 初始化变量`rep`为 `n - 1`，`mod`为4095，`mul`为7，`sgn`为2，`ans`为3。

- 如果输入参数`n`等于1，则直接返回3。
- 否则，进入循环，循环执行`n - 1`次。在每次循环中，首先判断`ans`与`mul`的按位与运算结果是否为0，即判断`ans`是否为`mul`的倍数。如果是，则根据`sgn`的值进行特定操作，将`sgn`更新为-2或者取反加1。
- 如果`ans`不是`mul`的倍数，则判断`ans`的个位数是否为8。为了判断个位数是否为8，使用一个临时变量`tmp`对`ans`进行操作，通过循环将`tmp`减去10，直到`tmp`小于等于0为止。然后判断`tmp + 2`的结果是否为0，如果是，则根据`sgn`的值进行特定操作，将`sgn`更新为-2或者取反加1。
- 判断结束后，更新`ans`的值，将`ans`乘以2，然后加上`sgn`的值。使用按位与运算符`&`将`ans`与`mod`进行按位与运算，相当于对`ans`取模。再继续循环
- 循环结束后，返回`ans`作为函数的返回值。

```
int16_t lab2(int16_t n) {
    // initialize
    if(n == 1) return 3;
    int rep = n - 1, mod = 4095, mul = 7;
    int sgn = 2;
    int ans = 3;

    // calculation
    while(rep--){
        if(!(ans & mul)){ // if v_n is a multiple of 7
            if(sgn == 2) sgn = -2;
            else sgn = ~sgn + 1;
        }
        else{ // if the last digit of v_n is 8
            int tmp = ans;
            while(tmp > 0) tmp -= 10;
            if(!(tmp + 2)){
                if(sgn == 2) sgn = -2;
                else sgn = ~sgn + 1;
            }
        }
        ans = ans + ans + sgn;
        ans &= mod; // use & to do modulo
    }

    // return value
    return ans;
}
```

3. 实验三，比较两个字符串，并返回差值的位置

需要用到两个函数

1. `check`函数接受一个`char`类型的参数`a`作为输入。函数的功能是判断`a`是否为小写字母或者大写字母。若是则返回1，不是则返回0。
2. `lab3`函数接受两个`char`数组类型的参数`s1`和`s2`作为输入，返回字符串出现不同的位置的差值。
    - 先初始化指针`p1`和`p2`分别指向`s1`和`s2`的起始位置。初始化变量`cmp`为0。
    - 然后不断循环判断，只要有一个不是字母，则退出循环。

- 在循环中，将`*p1`和`*p2`进行比较，计算差值赋给变量`cmp`。如果`cmp`不为0，说明两个字符不相等，直接返回`cmp`。否则，继续比较下一个字符，即将指针`p1`和`p2`分别向后移动一位。
- 循环结束后，如果两个字符串的字母都已经比较完毕，则返回`*p1 - *p2`的差值作为结果。
- 如果其中一个字符串已经比较完毕，而另一个字符串还有剩余字符，则直接返回`*p1 - *p2`的差值。

```
int check(char a){
    if((a - 'a' >= 0 && a - 'z' <= 0) || (a - 'A' >= 0 && a - 'Z' <= 0)) return
1;
    else return 0;
}

int16_t lab3(char s1[], char s2[]) {
    // initialize
    char *p1 = s1, *p2 = s2;
    int cmp = 0;

    // calculation
    while(check(*p1) && check(*p2)){
        cmp = *p1 - *p2;
        if(cmp) return cmp;
        p1++;
        p2++;
    }

    // return value
    return *p1 - *p2;
}
```

4. 实验四，九连环问题

九连环问题可以用递归调用的方法来解决。

`remove`函数和`put`函数分别表示将环从柱子上移走和放回到柱子上的操作。

`remove`函数先递归调用`remove`本身来移走除最后一个环外的其他环，然后计算出当前环需要移动的距离，并将这个距离保存在`memory`数组中。

`put`函数则先递归调用`put`本身来放置除最后一个环外的其他环，然后计算出当前环需要移动的距离，并将这个距离保存在`memory`数组中。

```
int16_t lab4(int16_t *memory, int16_t n, int16_t &state) { // stack, n rings,
stack index
    // initialize

    int16_t run = remove(memory, n, state);
    return run;
    // calculation

    // return value
}
```

```
int16_t remove(int16_t *memory, int16_t n, int16_t &state){

    if(n == 0){
        return 1;
    }
    else if(n == 1){
        memory[state + 1] = memory[state] + 1;
        state++;
        return 1;
    }
    int runR = 0;
    runR = remove(memory, n - 2, state);
    int shifter = n - 1, imp = 1;
    while(shifter--) imp += imp;
    memory[state + 1] = memory[state] + imp;
    state++;
    runR = put(memory, n - 2, state);
    runR = remove(memory, n - 1, state);

    return runR;
}

int16_t put(int16_t *memory, int16_t n, int16_t &state){
    if(n == 0){
        return 1;
    }
    else if(n == 1){
        memory[state + 1] = memory[state] - 1;
        state++;
        return 1;
    }
    int runP = 0;
    runP = put(memory, n - 1, state);
    runP = remove(memory, n - 2, state);
    int shifter = n - 1, imp = 1;
    while(shifter--) imp += imp;
    memory[state + 1] = memory[state] - imp;
    state++;
    runP = put(memory, n - 2, state);

    return runP;
}
```

测试结果图如下:

根据所给的`test.txt`文件运行出的结果如下:

txt文件:

```
5
9
DsTAs DsTA
3
```

输出：

```
===== lab1 =====
15
===== lab2 =====
786
===== lab3 =====
115
===== lab4 =====
0000000000000001
0000000000000101
0000000000000100
0000000000000110
0000000000000111
PS E:\Program_E\ics_lab\lab8>
```

判断分析结果正确

总结与反思

1. 本次实验，通过编写C语言，复习了前面写过的实验，对于lc3语言和C语言的共通点有了更加深入的了解。也建立了lc3和高级汇编语言的联系。
2. 之前已经了解过C语言所以本次实验遇到的困难不多，大多也是语法错误等产生的，可以通过自己编译解决。
3. 感谢助教和老师本学期的辛苦付出，助教辛苦啦！老师辛苦啦！完结撒花！！！