Lab 7 report

PB22111711 陈昕琪

实验目的与内容

编写lc3汇编器，要求能编译出lc3语句，并且可以处理非法文件，识别字符，伪代码，标签等等。

逻辑设计

仿照lc3的编码过程，先扫描一遍，建立符号表，然后逐条语句分析，这里要注意进制转化和输出。由于实验代码过长，这里仅展示部分代码。

程序代码分析

1. 根据已有的代码框架，填写需要填写的地址，在`void assemble(char lines[][MAX_LINE_LENGTH], int num_lines, char *output[], int* num)`函数中，首先需要找到`.ORIG`和`.END`的位置，并建立符号表，这里类似lc3的编译过程，建立符号表之后再扫描一遍同时输出机器码。

```
//首先找到起始地址的位置
    int start = findins(lines, num_lines, ".ORIG ");
    int i = 0, j = 0, count = 0;
    int start_address= 0;
    while(lines[start][i] != 'x' && lines[start][i] != '#'){//判断是不是十六进制和十进制
        i++;
    }if(lines[start][i] == '#'){//十进制
        i++;
        for(; i < (int)strlen(lines[start]); i++){
            start_address = 10 * start_address;
            start_address += lines[start][i] - '0';
        }
    }else if(lines[start][i] == 'x'){//十六进制
        i++;
        for(; i < (int)strlen(lines[start]); i++){
            start_address = 16 * start_address;
            start_address += lines[start][i] - '0';
        }
    }
    //找到结尾的位置
    int end = findins(lines, num_lines, ".END");
    char labellist[100][MAX_LINE_LENGTH];//创建符号表
    int locate[100] = {0};
    //先遍历一遍，创建符号表
    for(i = start; i <= end; i++){
        if(is_label(lines[i])){
            j = strchr(lines[i], ' ') - lines[i] + 1;//发现label
```

```
            strncpy(labellist[count], lines[i], j);//进行标记
            labellist[count][j] = '\0';
            locate[count] = i + start_address - 1 - start;
            count++;
        }
    }
```

2. 判断语句并且逐条输出。指令集已经声明，需要根据寻找前缀的函数判断出指令并进行相应的机器码输出。在这里，`.BLKW`和`.STRING`需要特别判断，因为其占用多个位置。排除这两个指令后进行输出翻译。

```
for (int i = start; i < end; i++)
    {
        leftstring = lines[i];
        if(islabel(leftstring)){//判断label
            leftstring = strchr(leftstring, ' ') + 1;//跳过label
        }
        //.BLKW
        if(strncmp(leftstring, ".BLKW ", strlen(".BLKW ")) == 0){//判断出.BLKW
            leftstring = strchr(leftstring, ' ') + 1;
            if(leftstring[0] == 'x'){//十六进制
                for(int k = 1; leftstring[k] != '\0'; k++){
                    number *= 16;
                    if(leftstring[k] <= '9'){
                        number = leftstring[k] - '0';
                    }else if(leftstring[k] >= 'a'){
                        number = leftstring[k] - 'a' + 10;
                    }else{
                        number = leftstring[k] - 'A' + 10;
                    }
                }
                for(int j = 0; j < num; j++){
                    strcpy(output[i - start + count + j], "0000000000000000");
                    tempt++;
                    *num = *num + 1;
                }
            }
            else if(leftstring[0] == '#'){//十进制
                for(int k = 1; leftstring[k] != '\0'; k++){
                    number *= 10;
                    number += leftstring[k] - '0';
                }
                for(int j = 0; j < number; j++){
                    strcpy(output[i - start + count + j], "0000000000000000");
                    tempt++;
                    *(num)++;
                }
            }number = 0;
            count += (tempt - 1);
            tempt = 0;
            continue;
        }
        else if(strncmp(leftstring, ".STRINGZ ", strlen(".STRINGZ ")) == 0){//判
断出.STRINGZ
            leftstring = strchr(leftstring, ' ') + 2;//到达string
```

```
            for(int j = 0; leftstring[j] != '\0'; j++){
                number = leftstring[j];
                if(number == '"'){
                    strcpy(output[i - start + count + j],
"0000000000000000");//.STRING最后是结束
                    count++;
                    break;
                }for(int k = 0; k < 16; k++){
                    left = number % 2;
                    if(left == 0){
                        output[i - start + count + j][15 - k] = '0';
                    }else{
                        output[i - start + count + j][15 - k] = '1';
                    }number = number / 2;
                }output[i - start + count + j][16] = '\0';
                tempt++;
                *num = *num + 1;
            }count += (tempt - 1);
            number = 0;
            tempt = 0;
            continue;
        }
        translate_instruction(lines[i], output[i - start + count], locate,
labellist, i + start_address - start);
    }
```

3. 逐条分析语句

- `NOT`：直接输出，这里需要判断寄存器的数字并输出

```
//NOT
    if(strncmp(ins_string, "NOT ", strlen("NOT ")) == 0){
        strcpy(machine_code, "1001");
        ins_string = strchr(ins_string, 'R') + 1;
        strcat(machine_code, get_register(ins_string[0]));//DR
        ins_string = strchr(ins_string, 'R') + 1;
        strcat(machine_code, get_register(ins_string[0]));//SR
        strcat(machine_code, "111111");
        return ;
    }//endNOT
```

- `ADD`：需要判断是立即数还是寄存器，根据相应的条件输出结果。

```
//ADD
    if(strncmp(ins_string, "ADD ", strlen("ADD ")) == 0){
        strcpy(machine_code, "0001");//条件码
        ins_string = strchr(ins_string, 'R') + 1;
        strcat(machine_code, get_register(ins_string[0]));//DR
        ins_string = strchr(ins_string, 'R') + 1;
        strcat(machine_code, get_register(ins_string[0]));//SR1
        ins_string = strchr(ins_string, ' ') + 1;
        if(ins_string[0] == 'R'){//寄存器
            strcat(machine_code, "000");
            strcat(machine_code, get_register(ins_string[1]));
            return ;
```

```
        }//立即数
        else if(ins_string[0] == 'x'){//十六进制
            strcat(machine_code, "1");
            char* hex = gethx_5(ins_string + 1);//五位补码
            strcat(machine_code, hex);
            free(hex);
            return ;
        }else{//十进制
            strcat(machine_code, "1");
            char* dec = getde_5(ins_string + 1);
            strcat(machine_code, dec);
            free(dec);
            return ;
        }
    }//endadd
```

- `AND`：与`ADD`类似，判断出是寄存器还是立即数并进行输出。

```
//AND，与ADD类似
    else if(strncmp(ins_string, "AND ", strlen("AND ")) == 0){
        strcpy(machine_code, "0101");//条件码
        ins_string = strchr(ins_string, 'R') + 1;
        strcat(machine_code, get_register(ins_string[0]));//DR
        ins_string = strchr(ins_string, 'R') + 1;
        strcat(machine_code, get_register(ins_string[0]));//SR1
        ins_string = strchr(ins_string, ' ') + 1;
        if(ins_string[0] == 'R'){//SR2
            strcat(machine_code, "000");
            strcat(machine_code, get_register(ins_string[1]));
            return ;
        }else if(ins_string[0] == 'x'){
            strcat(machine_code, "1");
            char* hex = gethx_5(ins_string + 1);
            strcat(machine_code, hex);
            free(hex);
            return ;
        }else{
            strcat(machine_code, "1");
            char* dec = getde_5(ins_string + 1);
            strcat(machine_code, dec);
            free(dec);
            return ;
        }
    }//endAND
```

- `BR`：这里需要根据`label`的值判断`pcoffset9`,并进行输出

```
//BR跳转指令
    else if(strncmp(ins_string, "BR ", strlen("BR ")) == 0 ||
            strncmp(ins_string, "BRN ", strlen("BRN ")) == 0 ||
            strncmp(ins_string, "BRZ ", strlen("BRZ ")) == 0 ||
            strncmp(ins_string, "BRP ", strlen("BRP ")) == 0 ||
            strncmp(ins_string, "BRZP ", strlen("BRZP ")) == 0 ||
            strncmp(ins_string, "BRNP ", strlen("BRNP ")) == 0 ||
```

```
            strncmp(ins_string, "BRNZ ", strlen("BRNZ ")) == 0 ||
            strncmp(ins_string, "BRNZP ", strlen("BRNZP ")) == 0){
        strcpy(machine_code, "0000");//条件码
        ins_string = strchr(ins_string, 'R') + 1;
        if(ins_string[0] == 'n' || ins_string[0] == 'N'){//N
            strcat(machine_code, "1");
            ins_string = ins_string + 1;
        }else{
            strcat(machine_code, "0");
        }if(ins_string[0] == 'z' || ins_string[0] == 'Z'){//Z
            strcat(machine_code, "1");
            ins_string = ins_string + 1;
        }else{
            strcat(machine_code, "0");
        }if(ins_string[0] == 'p' || ins_string[0] == 'P'){//P
            strcat(machine_code, "1");
            ins_string = ins_string + 1;
        }else{
            strcat(machine_code, "0");
        }if(strncmp(machine_code, "0000000", 7) == 0){//判断BR
            strcpy(machine_code, "0000111");
        }ins_string++;
        char* dis = getPCoffset9(pc, ins_string, labellist, locate);
        strcat(machine_code, dis);
        free(dis);
        return ;
    }//endBR
```

- `JMP`, `RET`,`JSRR`,`JSR`,`RTI`：与其他语句相比较简单，判断出条件之后可以直接输出。

```
//JMP
    else if(strncmp(ins_string, "JMP ", strlen("JMP ")) == 0){
        strcpy(machine_code, "1100000");//条件码
        ins_string = strchr(ins_string, 'R') + 1;
        strcat(machine_code, get_register(ins_string[0]));
        strcat(machine_code, "000000");
        return ;
    }//endJMP
    //RET,直接输出
    else if(strncmp(ins_string, "RET ", strlen("RET ")) == 0){
        strcpy(machine_code, "1100000111000000");
        return ;
    }//endRET
    //JSRR
    else if(strncmp(ins_string, "JSRR ", strlen("JSRR ")) == 0){
        strcpy(machine_code, "0100000");
        ins_string = strchr(ins_string, ' ') + 2;
        strcat(machine_code, get_register(ins_string[0]));
        strcat(machine_code, "000000");
        return ;
    }//endJSRR
    //JSR,判断完JSRR才能判断JSR
    else if(strncmp(ins_string, "JSR ", strlen("JSR ")) == 0){
        strcpy(machine_code, "01001");
        ins_string = strchr(ins_string, ' ') + 1;
```

```
        char* dis = getPCoffset11(pc, ins_string, labellist, locate);
        strcat(machine_code, dis);
        free(dis);
        return ;
    }//endJSRR
```

- `LDI`,`LDR`,`LD`,`LEA`：都需要对地址进行判断，不同的语句相应的有不同的判断方式，进行对应的输出

```
//LDI
    else if(strncmp(ins_string, "LDI ", strlen("LDI ")) == 0){
        strcpy(machine_code, "1010");
        ins_string = strchr(ins_string, 'R') + 1;
        strcat(machine_code, get_register(ins_string[0]));
        ins_string = strchr(ins_string, ' ') + 1;
        char* dis = getPCoffset9(pc, ins_string, labellist, locate);
        strcat(machine_code, dis);
        free(dis);
        return ;
    }//endLDI
    //LDR
    else if(strncmp(ins_string, "LDR ", strlen("LDR ")) == 0){
        strcpy(machine_code, "0110");
        ins_string = strchr(ins_string, ' ') + 2;
        strcat(machine_code, get_register(ins_string[0]));//DR
        ins_string = strchr(ins_string, 'R') + 1;
        strcat(machine_code, get_register(ins_string[0]));//BaseR
        ins_string = strchr(ins_string, ' ') + 1;//offset6
        char* distance = getoff6(ins_string);
        strcat(machine_code, distance);
        free(distance);
        return ;
    }//endLDR
    //LD
    else if(strncmp(ins_string, "LD ", strlen("LD ")) == 0){
        strcpy(machine_code, "0010");
        ins_string = strchr(ins_string, 'R') + 1;
        strcat(machine_code, get_register(ins_string[0]));
        ins_string = strchr(ins_string, ' ') + 1;
        char* distance = getPCoffset9(pc, ins_string, labellist, locate);
        strcat(machine_code, distance);
        free(distance);
        return ;
    }//endLD
    //LEA
    else if(strncmp(ins_string, "LEA ", strlen("LEA ")) == 0){
        strcpy(machine_code, "1110");
        ins_string = strchr(ins_string, 'R') + 1;
        strcat(machine_code, get_register(ins_string[0]));//DR
        ins_string = strchr(ins_string, ' ') + 1;//label
        char* distance = getPCoffset9(pc, ins_string, labellist, locate);
        strcat(machine_code, distance);
        free(distance);
        return ;
    }//endLEA
```

> `STI`，`STR`，`ST`语句与上面语句类似，这里不过多阐述

- `TRAP`：判断出矢量的值并且输出二进制

```
//TRAP
    else if(strncmp(ins_string, "TRAP ", strlen("TRAP ")) == 0){
        strcpy(machine_code, "11110000");
        ins_string = strchr(ins_string, 'x') + 1;
        char* b_number;
        b_number = getbinary(ins_string[0]);
        strcat(machine_code, b_number);
        free(b_number);
        b_number = getbinary(ins_string[1]);
        strcat(machine_code, b_number);
        free(b_number);
        return ;
    }//endTRAP
```

- `.ORIG`：需要输出后面的起始地址，起始地址需要转化成二进制输出

```
//.ORIG
    else if(strncmp(ins_string, ".ORIG ", strlen(".ORIG ")) == 0){
        ins_string = strchr(ins_string, ' ') + 1;
        int start_address = 0, left;
        if(ins_string[0] == 'x'){
            for(int i = 1; ins_string[i] != '\0'; i++){
                start_address *= 16;
                start_address += ins_string[i] - '0';
            }
        }else{
            for(int i = 1; ins_string[i] != '\0'; i++){
                start_address *=  10;
                start_address += ins_string[i] - '0';
            }
        }for(int i = 0; i < 16; i++){
            left = start_address % 2;
            if(left == 0){
                machine_code[15 - i] = '0';
            }else{
                machine_code[15 - i] = '1';
            }start_address = start_address / 2;
        }machine_code[16] = '\0';
        return ;
    }//end.ORIG
```

- `END`：伪操作结束指令，输出回车

```
//.END
    else if(strncmp(ins_string, ".END ", strlen(".END ")) == 0){//.END伪操作
        strcpy(machine_code, "\0");
        return ;
    }
```

- `.FILL`：填充，需要看后面的内容并转化成二进制

```
else if(strncmp(ins_string, ".FILL ", strlen(".FILL ")) == 0){//.FILL伪操作
        ins_string = strchr(ins_string, ' ') + 1;
        char* distance;
        if(ins_string[0] == 'x'){//十六进制
            int count = 0;
            for(int i = 1; ins_string[i] != '\0'; i++){
                count++;
                distance = getbinary(ins_string[i]);
                strcat(machine_code, distance);
            }for(int i = 0; i + count <= 4; i++){
                strcat(machine_code, "0000");
            }
        }else if(ins_string[0] == '#'){
            int number = 0, left;
            for(int i = 1; ins_string[i] != '\0'; i++){
                number *= 10;
                number += ins_string[i] - '0';
            }for(int i = 0; i < 16; i++){
                left = number % 2;
                if(left == 0){
                    machine_code[15 - i] = '0';
                }else{
                    machine_code[15 - i] = '1';
                }number = number / 2;
            }
        }machine_code[16] = '\0';
        return ;
    }
```

4. 辅助函数，用于给语句判断提供机器码

主要实现的函数如下，在这里不多加赘述：

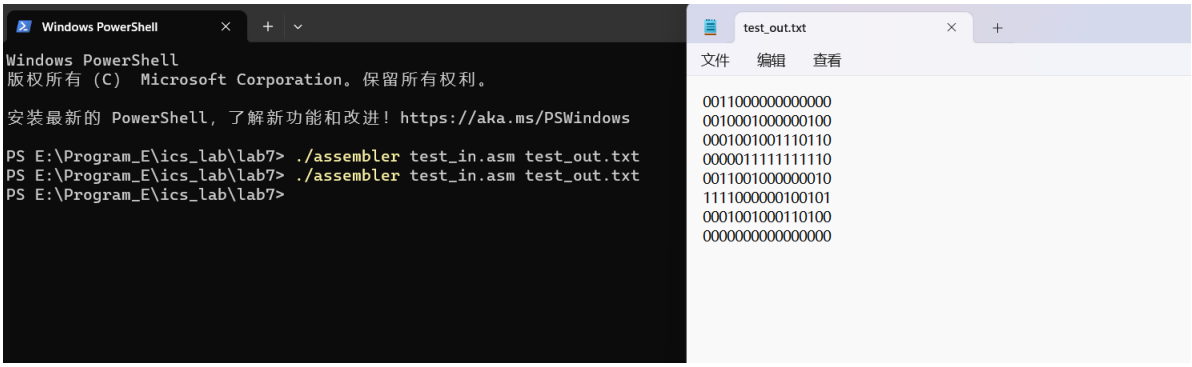
```
int findins(char lines[][MAX_LINE_LENGTH], int num_lines, char* string);//寻找前缀和对应的字符匹配的语句
char* getPCoffset9(int pc, const char* label, char labellist[][MAX_LINE_LENGTH], int* locate);//返回应当输出的PCoffset9的值
char* getPCoffset11(int pc, const char* label, char labellist[][MAX_LINE_LENGTH], int* locate);//返回应当输出的PCoffset11的值
int is_label(const char* string);//判断是不是label
char* get_register(char number);//获取寄存器的编码
char* getoff6(const char* number);//获取10进制或16进制转换成对应的6位补码
char* getbinary(char number);//将16进制转换成二进制
char* gethx_5(const char* string);//返回16进制对应的5位补码
char* getde_5(const char* string);返回10进制对应的5位补码
```

测试结果图如下:

在`asm`文件中输入指令，在终端中运行，测试结果如下：

（这里用的是实验文档中的asm文件）

输出：



判断分析结果正确

总结与反思

1. 通过编写lc3汇编器，更加深入的了解了lc3识别语句的过程，并将此过程代码化，实现机器码输出，同时也加深了自己对lc3语言的了解与认识。
2. 临近期末周，代码难度很大，本次实验中遇到了很多困难。在写代码时是一边写一边补充函数的，应该在写代码之前规划好，这点浪费了很多时间而且使得编译代码的过程非常麻烦。编写代码时遇到了很多问题（自己C语言没学好导致的），包括代码的连续性，函数调用的参数是否正确（C语言指针没学好）。以及对于`labellist`的建立、使用，包括不同进制间的转化，都需要细致的调试，真的写的很破防qwq，但是能写出代码还是有成就感的。
3. 未来可以考虑优化代码结构和提高程序的健壮性，增加更多的测试用例以验证程序的准确性。