LC-3 汇编器

PB19051175 梁峻滔

1. 设计思路

该汇编器用 C 编写，使用一个二维字符数组存储一个字符串形式的汇编程序，此外还定义两个全局二维字符数组，一个作为符号表存储汇编程序出现的 Label，另一个存储 LC-3 汇编语言的关键字，以在创建符号表时辨别某一字符串是否是 Label。

采用两边扫描的方案，第一遍先识别程序中的 Label 并创建符号表，第二遍利用符号表将各条指令翻译成机器码形式。

整个汇编器 C 程序中包含的函数主要有：

① 一个判断指令类型的 assemble 函数，该函数的功能是传进一行汇编指令，通过字符串匹配辨别该指令是属于哪一种指令，返回对应的编号。

```
int assemble(char in[row][MAX_NUM],int r){
```

其中 in[row][MAX_NUM]为要翻译的汇编程序，r 为行号。

② 各种类型的指令的翻译函数，assemble 函数识别某一行汇编指令的类型后，通过返回的编号值，跳转至相应的指令翻译函数来将该行汇编指令翻译成机器码。每个翻译函数的基本操作都是字符串匹配来判别所需要翻译的寄存器、地址偏移等，并返回执行成功/失败信息。

```
case -1: break;
case 0 : {if(!BR(instr,number,r,label)) iserror = TRUE;break;}
case 1 : {if(!ADD(instr,number,r)) iserror = TRUE;break;}
case 2 : {if(!LD(instr,number,r,label)) iserror = TRUE;break;}
case 3 : {if(!ST(instr,number,r,label)) iserror = TRUE;break;}
case 4 : {if(!JSR(instr,number,r,label)) iserror = TRUE;break;}
case 5 : {if(!AND(instr,number,r)) iserror = TRUE;break;}
case 6 : {if(!LDR(instr,number,r)) iserror = TRUE;break;}
case 7 : {if(!STR(instr,number,r)) iserror = TRUE;break;}
case 8 : {if(!RTI(instr,number,r)) iserror = TRUE;break;}
case 9 : {if(!NOT(instr,number,r)) iserror = TRUE;break;}
case 10 : {if(!LDI(instr,number,r,label)) iserror = TRUE;break;}
case 11 : {if(!STI(instr,number,r,label)) iserror = TRUE;break;}
case 12 : {if(!JMP(instr,number,r)) iserror = TRUE;break;}
case 14 : {if(!LEA(instr,number,r,label)) iserror = TRUE;break;}
case 15 : {if(!TRAP(instr,number,r)) iserror = TRUE;break;}
case 16 : {if(!ORIG(instr,r)) iserror = TRUE;break;}
case 17 : {if(!HALT()) iserror = TRUE;break;}
case 18 : {END_visited = TRUE; finish = TRUE;}
```

③ 创建关键字表函数

```
Status CreateKeylist(char ke[key_amount][key_len]){
```

④ 创建符号表函数

```
Status CreateLabellist(char labe[row][MAX_NUM],char in[row][MAX_NUM],char ke[key_amount][key_len]){
```

⑤ 补码转换函数

```
//给定一个0-1字符串，将其转换成补码表示
Status complement(char bin[],int bit_num){
```

⑥ 二进制表示转换函数

```
//给定一个十进制数和指定bit数，输出二进制表示
Status PrintComplement(int R, int bit_num){
```

2. 功能

该汇编器能够识别某些指令的不同的寻址模式；能够创建符号表；能够识别 BR、JSR、LD、LDI、ST、STI 等指令中的 Label，并判断该 Label 是否是符号表中出现的符号，若是则计算相应的地址偏移量；通过识别.END 判断翻译停止。总而言之就是可以将 LC-3 的 15 种指令和.ORIG 的汇编形式翻译成对应的机器码

形式，并通过打印出来的形式显示。

支持的错误检测：

① 指令中的立即数或偏移量是否越界；

② 指令中的 Label 是否是符号表中已有；

③ 汇编程序中是否缺少.END。

未能完善的部分：

① 存放在二维字符数组中的汇编程序不能有整行注释，即要求每一行都有一条有效指令，这是因为各翻译函数中计算地址偏移量时都使用了该指令的行号来计算；

② 未能实现.FILL、.BLKW、.STRINGZ 伪指令的相应翻译，所以要利用翻译出来的机器码在 LC-3 模拟器上运行的话，程序中.FILL、.BLKW、.STRINGZ 的功能需要预先手动设置。

3. 示例测试

采用课本的乘 6 的程序作为测试示例：


```
01  ;
02  ; Program to multiply an integer by the constant 6.
03  ; Before execution, an integer must be stored in NUMBER.
04  ;
05          .ORIG   x3050
06          LD      R1,SIX
07          LD      R2,NUMBER
08          AND     R3,R3,#0        ; Clear R3. It will
09                                  ; contain the product.
0A  ; The inner loop
0B  ;
0C  AGAIN   ADD     R3,R3,R2
0D          ADD     R1,R1,#-1       ; R1 keeps track of
0E          BRp     AGAIN           ; the iterations
0F  ;
10          HALT
11  ;
12  NUMBER  .BLKW   1
13  SIX     .FILL   x0006
14  ;
15          .END
```

Figure 7.1 An assembly language program.

```
char instr[row][MAX_NUM]={
".ORIG x3050",
"LD R1,SIX",
"LD R2,NUMBER",
"AND R3,R3,#0",
"AGAIN ADD R3,R3,R2",
"ADD R1,R1,#-1",
"BRp AGAIN",
"HALT",
"NUMBER .BLKW 1",
"SIX .FILL x0006 "
".END"
};
```

该程序是将最初存放在 NUMBER 的整数乘以 6，与教材一致，假设存放在 NUMBER 的整数为 123，则最后存放在 R3 的结果应是$123 \times 6 = 738$. 汇编器翻译的机器码如下所示：

```
PS C:\Users\Snowball> cd "d:\Software\VS code\Codes\"
 0011000001010000
 0010 001 000000111
 0010 010 000000101
 0101 011 011 1 00000
 0001 011 011 0 00 010
 0001 001 001 1 11111
 0000 001 111111101
 1111 0000 00100101
PS D:\Software\VS code\Codes>
```

将该翻译结果复制至 LC-3 模拟器，预先设置 NUMBER(x3057)处存放整数 123，SIX 处存放整数 6：

```
1   0011000001010000
2   0010 001 000000111
3   0010 010 000000101
4   0101 011 011 1 00000
5   0001 011 011 0 00 010
6   0001 001 001 1 11111
7   0000 001 111111101
8   1111 0000 00100101
```

!	x3050	x2207	8711	0010001000000111
!	x3051	x2405	9221	0010010000000101
!	x3052	x56E0	22240	0101011011100000
!	x3053	x16C2	5826	0001011011000010
!	x3054	x127F	4735	0001001001111111
!	x3055	x03FD	1021	0000001111111101
!	x3056	xF025	-4059	1111000000100101
!	x3057	x007B	123	
!	x3058	x0006	6	

R0	x0000	0
R1	x0000	0
R2	x0000	0
R3	x0000	0
R4	x0000	0
R5	x0000	0
R6	x0000	0
R7	x0000	0

运行后的结果为：

R0	x0000	0
R1	x0000	0
R2	x007B	123
R3	x02E2	738
R4	x0000	0
R5	x0000	0
R6	x0000	0
R7	x0000	0

与预期结果相符。

由于个人时间、精力、水平有限，加之面临其他各门课程的期末考试，且汇编器提交时间截止在即，目前的汇编器只能完成最基本的指令翻译功能，很多地方尚未完善，对于可能给助教们带来的检查上的麻烦，我深表歉意，感谢老师在本课程上的教导和助教们的巨大帮助。