《编译原理》实验三

卢志超（131220017、497183006@qq.com）

(南京大学 计算机科学与技术系，南京 210093)

1 实现的功能：

在词法语法分析及语义分析和类型检查均通过的情况下，对代码进行了中间代码的生成，实现了所有试验必做内容.

1.1 必做部分

亮点：

1. 将中间代码生成与语义分析部分分离开来，先进行语义分析后根据语法树进行中间代码生成（新增加的文件 intercode.c 与 intercode.h 中存放中间代码生成的一些接口代码，gencode.c 与 gencode.h 为遍历语法树并生成中间代码的代码）。
2. 维护了一张双向链表，如下：

```
void insertCode(InterCode c);
void deleteCode(InterCode c);
void printCode(char *path);
void printOp(Operand op,FILE *fp);
extern InterCode head;
extern InterCode tail;
extern int varcount;
extern int labcount;
```

insertCode()与 deleteCode()为对双向表的操作接口函数，
head 与 tail 指针分别指向双向链表的头与尾。

3. 采用语法制导的方法，自顶向下的生成中间代码，在 genCode.c 中 有详细内容。虽然这样生成效率比较低，但代码的模块性比较好，便于以后的修改。
4. 将一些操作封装使用，提升了代码之间的模块性，如下：

```
Operand newtemp(){
    Operand op=(Operand)malloc(sizeof(struct Operand_));
    op->kind=TEMPVAR;
    op->u.var_no=varcount++;
    return op;
}

Operand newlabel(){
    Operand op=(Operand)malloc(sizeof(struct Operand_));
    op->kind=LAB;
    op->u.var_no=labcount++;
    return op;
```

Newtemp 为生成一个新的临时变量的 Oprand 变量

5. 定义了一些测试函数用以 debug:

```
25 void printO(Operand o){
26     printf("kind:%d\n",o->kind);
27     switch(o->kind){
28         case LABEL:case TEMPVAR:
29         printf("no:%d\n",o->u.var_no);
30         break;
31         case CONSTANT:case VARIABLE:
32         printf("value:%s\n",o->u.value);
33         break;
34         case TADDRESS:case VADDRESS:
35        // printO(o->u.name);
36         break;
37     }
38
39 }
40
```

思想：

从程序开始的 Program 节点开始，为所有节点定义函数，这个函数的参数是这个节点，为所有需要生成代码的节点，根据相应的翻译模式，调用相应的规则来生成中间代码。比如：

ID	variable = lookup(sym_table, ID) return [place := variable.name]

中间代码生成的语法规则

```
else if(strcmp(p->type,"ID")==0){
    struct typeNode *q=p->brotherNode;
    if(q==NULL){
        FieldList f=find_var(p->text);
        place->kind=VARIABLE;
        place->u.value=f->name;
```

实际生成代码

如图，上图为 ID 节点的生成规则，下图为实际生成的代码。Find_var()函数从符号表寻找与 ID 节点对应的变量的信息，即 lookup()，其中，place 变量为上层已经分配的 oprand 类型变量，可以看做是上层为这一层准备好的“信息”。

当然，这只是所有节点中的一个节点 tExp()的一小部分而已，对每一个节点，不仅需要考虑分配给它何种类型的 Oprand 与 Intercode,更重要的是必须知道中间代码的翻译模式，根据翻译模式来生成中间代码。

2 实验体会

本次实验让我掌握了中间代码的生成，了解了中间代码生成过程的具体过程。老师给的实验讲义写的相当详细且有帮助，助教在群里的回答也很有帮助。我觉得本次实验相较于实验二简单一些，这可能是因为我没写选做和优化代码 QAQ。总的来说，我觉得这次实验还是挺有趣的。

3 如何编译

在 Code 文件目录下 make 即可。运行方法与实验二类似，需要指令最后加文件路径用来存放 ir 文件，比如：**./parser ~/Desktop/Test/e1.cmm /home/lzc/Desktop/out.ir**

最后感谢老师和助教，祝好！