Lab3 实验报告

161220117 唐诗美

一、功能

在实验要求的 7 个假设下实现了 C--源代码的翻译；

进行了一些中间代码生成的优化；

二、实现方法

本次实验的主要实现是在 ir.c 与 ir.h 两个文件中。有 Operand 和 InterCodes 两个主要结构体，分别是中间代码的操作数和中间代码：

```
1.  struct Operand_ {
2.      op_kind kind;   //seven types
3.      union {
4.          int var_no;
5.          int label_no;
6.          int tmp_no;
7.          int value_int;
8.          float value_float;
9.          char f_name[60];
10.     } u;
11.     char v_name[60];
12.     int is_address;     //for tmp -1: nothing 0: address_content 1: address

13.     struct Operand_* next;
14. };
```

其中记录变量的名字是为了后面简化代码，is_address 变量是为了对于区分临时变量是否是地址还是地址中的值或是被赋了普通变量的值。

中间代码是用双向链表的方式连接起来的，在连接两个 ir 部分的时候，是使用 concat(InterCodes, InterCodes)函数实现的。

Operand 也是利用链表连接的，主要是为了处理已经声明过一个变量之后，就不用再次用新的变量去进行运算了，并且对于已经存在于 Operand 链表中的变量，也不需要先用临时变量取出它的值，可以直接利用在 Operand 链表中找到的这个变量直接带入进行计算，从而可以节省一条语句。

主要功能是通过 translate_XXX(...)函数实现的，和语义分析相似，都是重新遍历一遍这棵语法树，基本上是按照讲义上的来写的。但在立即数处理的时候就不需要一个新的临时变量。在判断为是立即数或是已经存在的变量后就把当前创建的临时变量直接变为当前对应的操作数，并且把临时变量计数减一。

本次实验在条件语句 label 与 goto 语句的生成部分利用了作业和书上写的含有 fall 的翻译规则，label 类型的 Operand 的 label_no 变为-1 即代表 fall。主要是在 translate_cond 中还有 translate_stmt 中的 Stmt_IfLpExpRpStmt 、 Stmt_IfLpExpRpStmtElseStmt 、 Stmt_WhileLpExpRpStmt 中进行修改。一开始没有注意到 translate_cond 的默认情况同样需要

修改，直接进行运行结果错的很离谱，发现了之后加了 label_no=-1 的时候不生成中间代码就正确了。利用这些翻译语句可以大大简化中间代码的数量。

打印部分是使用 print_ir(InterCodes)进行打印，其中调用了 print_op(Operand)函数。

三、编译运行方法

本次实验是在 ubuntu16.04 中进行的。

通过在 Code 文件夹中输入“make”指令，得到 parser，运行 parser 进行解析即可。3project_test 中的测试是讲义中的 test，mytest 是写的另外的一些测试，包括了生成一定范围的质数、gcd、哥德巴赫猜想数的判断、对于优化后的代码的 if、while 等跳转条件语句的测试，还有一些简单的函数调用。

四、实验总结

通过本次实验，对于中间代码的生成有了进一步认识，特别是之前学的时候 stmt 翻译时候的 S.next，在这次实验中也得到了充分的理解。

本次实验的很多 bug 还是由于没有对一些指针进行判断就使用，导致空指针赋值会出现段错误；同时对于指针一开始没有赋值的时候要初始化成 NULL，不然可能会指向一些奇怪的地方，从而使得代码报错；还有一些 bug 是因为之前在写的时候一些规则没有完全覆盖到，导致对应的那条规则没有正确的行为从而产生报错；对于书上规则 if false 的处理，一开始也没有处理正确，直接’>’的 false 就是’<’了，导致进行哥德巴赫猜想测试的时候输出完全不正确；调用函数之前传入参数我是利用一个 Operand 的数组实现的，但是一开始实现的时候压栈没有考虑到是反着压，前面几个测试都只有一个参数所以也没有发现这个问题，到了有多个参数的测试的时候才发现存在这个问题。

最后由于之前实现数组记录维数的顺序刚好和这次实验在访问数组元素大小相反，如果要实现选做改动较大就没有实现。但是可以实现结构体和多维数组声明的中间代码翻译。