Lab3实验报告

赵铭南 161271033

一、完成的功能

- 中间代码生成：

  在词法分析、语法分析和语义分析的基础上，可对输入的*.cmm文件进行中间代码生成。但不支持结构体类型的变量，不支持高维数组的变量以及不支持一维数组作为函数参数出现。

二、实现方法

- 中间代码生成：

  在实验一词法语法分析和实验二语义分析的基础上，通过对语法树进行遍历完成对中间代码的生成。采用的做法另建文件intercode.c来完成中间代码的相关操作。

  中间代码的表示：

  每条中间代码都由Intercode_数据结构表示, 整个翻译的中间代码由所有的Intercode_数据结构组成的一条双向链表所表示：

```
typedef struct Intercode_* Intercode;
struct Intercode_
{
    enum {ASSIGN_, ADD_, SUB_, MUL_, DIV_, LABEL_, FUNCTION_, GOTO_,
CONDGOTO_, RETURN_, DEC_, ARG_, CALL_, PARAM_, READ_, WRITE_} kind;
    union
    {
        struct {Operand left, right;} assign;
        struct {Operand result, op1, op2;} binop;
        struct {char* funcname;} func;
        struct {int labelnum;} label;
        struct {int labelnum;} goto_;
        struct {Operand op1, op2; char* relop; int labelnum;} condgoto;
        struct {Operand op;} return_;
        struct {Operand op; int size;} dec;
        struct {Operand op;} arg;
        struct {Operand result; char* funcname;} call;
        struct {Operand op;} param;
        struct {Operand op;} read;
        struct {Operand op;} write;
    }u;
    Intercode prev;
    Intercode next;
```

```
};
```

中间代码的翻译：

对于每个语法单元X，都对应的有函数translate_X对其进行翻译，函数返还该语法单元以及语法树中其子树所对应的所有中间代码。

```
Intercode translate_args(struct treenode* args);
Intercode translate_cond(struct treenode* exp, int label_true, int
label_false);
Intercode translate_exp(struct treenode* exp, Operand op);
Intercode translate_paramdec(struct treenode* paramdec);
Intercode translate_varlist(struct treenode* varlist);
Intercode translate_fundec(struct treenode* fundec);
Intercode translate_vardec(struct treenode* vardec, int size);
Intercode translate_dec(struct treenode* dec);
Intercode translate_declist(struct treenode* declist);
Intercode translate_def(struct treenode* def);
Intercode translate_deflist(struct treenode* deflist);
Intercode translate_stmtlist(struct treenode* stmtlist);
Intercode translate_compst(struct treenode* compst);
Intercode translate_extdef(struct treenode* extdef);
Intercode translate_extdeflist(struct treenode* extdeflist);
Intercode translate_program(struct treenode* program);
```

中间代码的翻译过程由main.c里Intercode code = translate_program(root);开始，通过递归遍历语法树获得整个文件对应的中间代码。最后，将中间代码通过void print_code(Intercode code, FILE* f);存入文件。

三、编译运行方法

- 环境：

```
GNU Linux Release: Ubuntu 18.04, kernel version 4.15.0-47
GCC version 7.3.0
GNU Flex 2.6.4
GNU Bison 3.0.4
```

- 编译：

进入目录./Code，终端下输入：make，或：

```
flex lexical.l
bison -d syntax.y
```

```
gcc tree.c syntax.tab.c semantic.c intercode.c main.c -lfl -ly -o parser
```

- 运行：

  终端下输入：`./parser [*.cmm] [*.ir]`

四、实验总结

实验三完成了编译器的中间代码生成的功能，能够对C--源代码进行中间代码生成，加深了对编译过程中中间代码生成的理解。