编译原理 Project3 实验报告

姓名:万子文

学号:151220102

邮箱:540304594@qq.com

如果测试过程中出现问题，希望助教可以通过邮箱联系我，避免因为gcc 版本或者其他原因产生问题。

一．实验完成情况

1. 完成了实验要求的在实验 2 基础上对代码进行翻译，产生了实验指导中规定格式的中间代码。
2. 定义了中间代码存储的格式，用列表的方式管理中间代码，建立了一系列维护链表的函数以及维护中间代码中各种操作数的函数(包括全局符号表中到中间代码变量的映射，以及中间代码中临时变量的管理)
3. 实现了实验中的基本要求，包括对一维数组的支持。通过了所有的必做测试样例以及自己创建的对一维数组的测试

二．实验说明

由于本次实验产生的代码文件并不多，所以采用手动逐条输入命令的方式。编译步骤(*在工程文件下的 lexical 子目录下)

1. flex lexical.l(产生词法分析代码)
2. bison –d syntax.y (产生语法分析代码)
3. gcc main.c syntax.tab.c parsertree.c ../semantic/symbolTable.c ../semantic/covert.c ../intercode/intercode.h ../intercode/translate.h -lfl -ly -o parser
   将上述所有代码编译，链接，最终产生的目标文件为 parser
4. 执行: ./parser example.c,我才用 Linux 下>操作导向到文件输出
   比如 ./parser example.c >result.IR

三．实验细节

1. 中间代码的维护

中间代码的结构体如下所示，：

```
struct InterCode_{
    int kind;
    union{
        struct {int LIndex;} LABELDEC;
        struct {char* funcname;} FUN;
        struct {Operand right;Operand left;} ASSIGN;
        struct {Operand result; Operand op1; Operand op2;} BINOP;
        struct {int LIndex;} GOTO;
        struct {Cond cond; int LIndex;} IFGOTO;
        struct {Operand ret;} RETURN;
        struct {Operand address; int size;} DEC;
        struct {Operand argument;} ARG;
        struct {Operand ret;char* funName;} CALL;
        struct {Operand parameter;} PARAM;
        struct {Operand input;} READ;
        struct {Operand output;} WRITE;
    };
};
```

```
struct Operand_{
    int kind;
    int attr;
    union{
        int VIndex;
        int TIndex;
        int ICons;
    };
};
```

用 Union 来维护不同类型中间代码的操作数等信息。其中 Operand 定义为 struct Operand_的指针。 该部分的定义在(intercode.h)中。此外为了维护中间代码，建立了双向链表，维护了对链表的合并等操作。

2. 中间代码翻译:

由于翻译的策略在实验指导中已经给出，所以这里举一个例子，表示我的实现部分。

IF LP Exp RP $Stmt_1$	label1 = new_label() label2 = new_label() code1 = translate_Cond(Exp, label1, label2, sym_table) code2 = translate_Stmt($Stmt_1$, sym_table) return code1 + [LABEL label1] + code2 + [LABEL label2]

```
int l1=NewLabel();
int l2=NewLabel();
ICEntry list1=transCond(GetithChild(x,3),l1,l2);
NewLabelIC(l1,list1);
ICEntry list2=transStmt(GetithChild(x,5));
list1=MergeInterCode(list1,list2);
NewLabelIC(l2,list1);
return list1;
```

上述是对于 IF LP Exp RP Stmt 的翻译，可以很清楚的看到代码和上文的处理策略的对应关系.相应的翻译函数在 translate.c 中