编译原理（3）实验报告

张语婷-181860140-普通班-1051570722@qq.com

一、 实现功能 基础+选做 3.1

1. Operand 及中间代码语句:

```
struct Operand_{
    enum{FROM_ARG,VARIABLE,TEMP,CONSTANT,ADDRESS,WADDRESS,FUNCTION,LABEL,RELOP}kind;
    int u_int;//t1t2
    char* u_char;
    Type type;//用于结构体和数组变量的Offset查询
};
```

```
struct InterCode_{
    enum{ILABEL,IFUNCTION,ASSIGN,
        ADD,SUB,MUL,DIV,
        ADDRASS1,ADDRASS2,ADDRASS3,
        GOTO,IF,RETURN,DEC,ARG,
        CALL,PARAM,READ,WRITE}kind;
    union{
        //LABEL,FUNCTION,GOTO,RETURN,ARG
        //PARAM,READ,WRITE
        struct{Operand op;}ulabel;
        //ASSIGN,CALL
        //ADDRASS1,ADDRASS2,ADDRASS3
        struct{Operand op1,op2;}uassign;
        //ADD,SUB,MUL,DIV
        struct{Operand result,op1,op2;}ubinop;
        //IF
        struct{Operand x,relop,y,z;}uif;
        //DEC
        struct{Operand op;int size;}udec;
    }u;
    InterCode before;
    InterCode next;
};
```

2. 每一个语法单元设置相应 translate 函数，根据产生式进行调用
   下面列举部分需要自己翻译的翻译模式:

   1）Exp → Exp_1 DOT ID
   - code1= translate_Exp (Exp_1, t1)
     取出结构体变量的首地址
   - code2=[place =: t1 + offset]
     通过 t1 中的 Type 类型（LAB2 中定义结构体时生成，LAB3 中从符号表中取出）以及 ID 中的域名信息，计算出该域名在结构体中的偏移量
   - 记录 place->type=ID.type，通过在符号表中查找 ID 取出

   2）Exp → Exp_1 LB Exp_2 RB
   - code1= translate_Exp (Exp_1, t1)
     取出数组变量的首地址
   - code2= translate_Exp (Exp_2, t2)
     计算数组元素的序号，存储在 t2->u_int 中

- code3=[offset=: t2 * size]
  size 为该数组单个元素大小，通过 t1->type 进行计算
- code4=[place =: t1 + offset]
  加法获得所求数组元素的具体地址
- 记录 place->type=Exp_1.type->u.array.elem

3）数组与结构体相互嵌套

- Exp->ID 返回的 place 的 type 为符号表中记录的 ID 的类型
- 数组返回的 place 的 type 为该数组的定义类型
- 结构体返回的 place 的 type 为调用的域的类型
- 以上三个 type 的返回值保证了不论多少重嵌套，都能找到正确的 type，用于下次 offset 的计算

4）Exp → ID LP Args RP

- 判断是否为 write 函数，是:
  code1=translate_Exp(Args->child, t1)
  code2=[WRITE t1]
- 不是 write 函数:
  code1=translate_Args(Args->child, t1)
  code2=[place := CALL ID.name]

5）Exp → ID LP RP

- 判断是否为 read 函数，是:
  code1=[READ place]
- 不是 read 函数:
  code1=[place := CALL ID.name]

6）translate_Args: Args → Exp COMMA Args| Exp

- 正向翻译 Exp 时，利用双向链表记录中间代码;
- 所有 Exp 语句翻译完成，从链表尾部向前将中间代码加入总的 Intercode 中

7）数组与结构体定义

- 生成语句：VarDec → ID | $VarDec_1$ LB INT RB ($VarDec_1$→ID) 或 StructSpecifier → STRUCT Tag (Tag →ID)
- 在符号表中找到 ID 对应的 TABLE (LAB2 中记录) ,通过其 field->type 计算出大小 size 即可
- code1=[DEC ID size]

8）函数定义

- 生成语句：FunDec → ID LP VarList RP| ID LP RP
- code1=[FUNCTION ID :]
- 在函数表中找到 ID 对应的 FUNCTION(LAB2 中记录) ,通过之前记录的函数参数列表，对于每一个参数都生成一个 PARAM 语句即可

9）其他

- 根据产生式调用下层 translate 函数，直至 translate_FunDec\Stmt\Exp\VarDec\Args\StructSpecifier
- 根据讲义以及上述自行翻译的语句生成对应的中间代码，最后回到 Pragram 中后进行中间代码的打印

二、 编译语句

make
./parser name.cmm out.ir（name.cmm 为 cmm 文件名字）