# 编译原理实验三

组员一姓名：关宇聪 学号221900415 组员二姓名：蒋先威 学号：221900485

# 实现功能

1. 完成了必做内容和选做内容3.2（一维数组类型的变量可以作为函数参数，可以出现高维数组类型的变量）
    1. 在semantic.c文件中添加了add_read_write_func()，用于添加read和write函数
    2. 给FieldList结构体新增了一个**isArg**成员变量，用于标记一个变量是不是函数的形式参数。并且在语义分析时进行初始化，在VarList()中将形式参数标记
        1. 如果一个数组类型变量是函数的形式参数，则ID代表其数组的基地址，不需要取地址的操作
        2. 如果是全局或局部变量的数组，则需要取地址操作
    3. 关于处理高维数组：
        1. 当translate_Exp(Node*root, Operandplace)为Exp -> Exp1 LB Exp2 RB生成中间代码时，检测Exp1的类型是什么
            1. 如果Exp1是数组类型，则表示Exp是一维数组，需要取Exp1的基地址，然后再加上offset
            2. 如果Exp1是地址类型，则表示Exp是高维数组，之间将Exp1和offset相加，等到目标地址
    4. 关于数据结构
        1. 用于表示操作符的数据结构：

```
struct Operand_ {
    enum {
        OP_VARIABLE,
        OP_ADDRESS,
        OP_FUNCTION,
        OP_ARRAY,
        OP_STRUCTURE,
        OP_LABEL,
        OP_TEMP,
        OP_CONSTANT,
    } kind;
    union {
        char* var_name;
        char* array_name;
        int addr_no;
        char* func_name;
        int label_no;
        int temp_no;
        long long const_val;
    } u;
    Type type;
    int size;
};
```

1. 
2. 其中，var_name, array_name, func_name来源于程序原本的变量名， addr_no, label_no, temp_no分别表示地址名，标签名和临时变量的编号，采取自增的逻辑，方便给这些变量命名
3. 枚举类型变量kind枚举了操作数的不同类型。 生成和打印操作数时，会根据类型储存或检索不同成员变量的值，这在new_op() 和 print_op() 中有体现

2. 用于表示中间代码语句的数据结构：

1. 
```
struct InterCode_ {
    enum {…
    } kind;
    union {
        // IR_LABEL IR_FUNC IR_GOTO IR_RETURN IR_ARG IR_PARAM IR_
        struct {
            Operand op;
        } single_ir;

        // IR_ASSIGN IR_ADDR IR_LOAD IR_STORE IR_CALL
        struct {
            Operand right, left;
        } binary_ir;

        // IR_ADD IR_SUB IR_MUL IR_DIV
        struct {
            Operand res, op1, op2;
        } ternary_ir;

        // IR_DEC x [size]
        struct {
            Operand op;
            int size;
        } dec;

        // IR_IF_GOTO
        struct {
            Operand x, y, z;
            char relop[64];
        } if_goto;
    } u;
};
```
2. 其中，枚举类型变量kind枚举了中间代码语句的不同类型。 生成和打印中间代码时，会根据类型储存或检索不同成员变量的值，这在new_irCode() 和 print_irCode() 中有体现
3. 中间代码用双向链表储存，方便在尾部插入新的中间代码，打印时从头遍历即可