实验报告

1.实现功能

本次实验是在词法分析、语法分析和语义分析程序的基础上，将C--源代

码翻译为中间代码。然后在虚拟机进行测试判断正确性。

2.实现思路

本次实验中，我使用双向链表来实现线性IR，然后对链表遍历来输出相应的中间代码。

数据结构使用的是实验指导中的Operand InterCode的实现。

```
struct _Operand{
    enum{
        Em_VARIABLE, // 变量（var）
        Em_CONSTANT, // 常量（#1）
        Em_ADDRESS, // 地址（&var）
        Em_LABEL, // 标签(LABEL label1:)
        Em_ARR, // 数组（arr[2]）
        Em_STRUCT, // 结构体（struct Point p）
        Em_TEMP, // 临时变量（t1）
        Em_RELOP,
        Em_FUNC,
    } kind;
    union{
        char*name;
        int varno; // 变量定义的序号
        int labelno; // 标签序号
        int val; // 操作数的值
        int tempno; // 临时变量序号（唯一性）
    } u;
    Type type; // 计算数组、结构体占用size
    int para; // 标识函数参数
};

struct _InterCode {
    enum {
        IC_ASSIGN,
        IC_LABEL,
        IC_PLUS,
        IC_SUB,
        IC_MUL,
        IC_DIV,
        IC_DEC,
        IC_FUNC,
        IC_CALL,
        IC_PARAM,
        IC_READ,
        IC_WRITE,
        IC_RETURN,
        IC_ARG,
```

```
        IC_GOTO,
        IC_IF_GOTO,
        IC_GET_ADDR,
        IC_READ_ADDR,
        IC_WRITE_ADDR,
        // ...
    } kind;

    union {
        Operand op;
        char* func;
        struct { Operand right, left; } assign;
        struct { Operand result, op1, op2; } binop; // 三地址代码
        struct { Operand x, y, z;char*relop} if_goto;
        struct { Operand result; char* func; } call;
        struct { Operand op; int size;} dec;
    } u;
};
```

在实验中，使用Operand类型的变量和genintercode函数以及操作类型来生成相应的Codelist。在实现过程中需要注意是传值还是传地址，当前operand位置是要地址还是变量，这一点所体现的需求在数组部分尤为明显。

3 印象深刻的点

- 在实现数组的过程中，需要注意取地址，声明的时候只是变量的值。
- 当两个数组相互赋值时，在本次实验中不能单纯的传地址，还需要遍历每一个元素。
- 选做3.2部分时，数组作为函数参数传递的话就是传地址，相应的其中使用数组元素的话就不同于main函数中的要取地址。
- 同时，面对高维数组，要进行十分复杂的分类讨论和计算offset值。要注意的是offset值的计算，相应的第i维对应的宽度是不一定相等的。要找好对应关系

4.运行

后台makefile编译即可