编译原理实验三报告

- 姓名：李润泽
- 211275011@smail.nju.edu.cn

一、程序功能

实现了c--语言的中间代码生成，选做了4.2数组相关的中间代码生成，Operand和interCode的结构体定义如下图所示，采用了链表式的IR

```
typedef struct _operand {
    enum {
        OP_VARIABLE,
        OP_CONSTANT,
        OP_ADDRESS,
        OP_LABEL,
        OP_FUNCTION,
        OP_RELOP,
    } kind;

    union {
        int value;
        char* name;
    } u;

} Operand;
```

```c
typedef struct _interCode {
    enum {
        IR_LABEL,
        IR_FUNCTION,
        IR_ASSIGN,
        IR_ADD,
        IR_SUB,
        IR_MUL,
        IR_DIV,
        IR_GET_ADDR,
        IR_READ_ADDR,
        IR_WRITE_ADDR,
        IR_GOTO,
        IR_IF_GOTO,
        IR_RETURN,
        IR_DEC,
        IR_ARG,
        IR_CALL,
        IR_PARAM,
        IR_READ,
        IR_WRITE,
    } kind;

    union {
        struct {
            pOperand op;
        } oneOp;
        struct {
            pOperand right, left;
        } assign;
        struct {
            pOperand result, op1, op2;
        } binOp;
        struct {
            pOperand x, relop, y, z;
        } ifGoto;
        struct {
            pOperand op;
            int size;
        } dec;
    } u;
} InterCode;
```

因为4.2虽然允许高维数组的定义与函数参数传递，但是并未允许数组的直接赋值，再根据示例可以看出c--的数组作为参数传递时是引用传递，需要注意。

另外对中间代码的生成没有做太多优化，主要对直接使用的符号和立即数进行了一步去掉一步创建新临时变量的优化。

因为翻译规则在讲义中已经明确给出，主要工作是数据结构的设计，相关接口的具体实现，中间代码的生成也是进入ExtDefList后的事，在这里选择不在语义分析时候生成IR，而是语义分析后不删除符号表再次遍历语法树生成IR，所以再次遍历语法树寻找ExtDefList节点然后开始处理

二、程序运行

因为不允许修改makefile，但本次实验中./parse相较而言需要多一个输出文件的参数，所以测试时是自己手动输入的，同样make编译，然后./parse [输入文件] [输出文件]. 执行程序