实验报告（三）

- 曹洋笛 161220004 2904428882@qq.com

一、实现功能

- 翻译中间代码
- 进行了少许的优化
- 实现选做1、2；数组和结构体可以被较妥善地处理

二、亮点

1. 双向链表保存中间代码：
2. 简单的小优化：

   对于赋值语句，如果检测到等号右侧是一个ID（基本变量）或者Int，则不必再使用 translateExp() 计算并创建一个临时变量存储它，直接把右侧值赋给左侧，也就是把很大一部分赋值语句的中间代码减少一行。

3. 良好的代码结构：最主要的一些函数（translate...()函数）全放在文件“IR.h/.c”中，采用自顶向下的顺序；将较为常用的过程封装在文件“IRUtils.h/.c”中，以供translate函数调用，且每一个函数都有注释，极大提升了代码可读性。

三、实现

1. 数据结构

- InterCode：表示一行中间代码

```
typedef enum { IR_ASSIGN, IR_ADD, IR_SUB, IR_MUL, IR_DIV, IR_LABEL,
IR_FUNCTION, IR_RETURN, IR_PARAM, IR_ARG, IR_CALL, IR_READ, IR_WRITE,
IR_IF, IR_GOTO, IR_DEC } IRKind; // 操作类型，即多个操作数的连接方式

struct InterCode {
  IRKind kind;
  union {
    struct { Operand* op; } one; // 包括：RETURN, PARAM, ARG, READ,
WRITE
    struct { Operand* op1; Operand* op2; } two; // 包括：ASSIGN, DEC
    struct { Operand* op1; Operand* op2; Operand* op3; } three; // 包
括：ADD, SUB, MUL, DIV
    struct { Operand* op; char* funcName; } call; // 包括：CALL
    struct { Operand* op1; Relop relop; Operand* op2; char* label; }
ifcode; // 包括：IF
    char* name; // 包括：LABEL, FUNCTION, GOTO
```

```
  };
  InterCode* prev;
  InterCode* next;
};
```

- Operand：某些中间代码的组成部分

```
typedef enum { OP_VAR, OP_CONST, OP_TEMP, OP_ADDR, OP_GETADDR,
OP_GETCONT } OpKind;

struct Operand {
  OpKind kind;
  union {
    char* name; // 变量名
    int val; // Lab3只考虑整数
  };
};
```

- 实现思路：

  首先定义并在语义分析过程中调用一个函数负责把“read()”和“write()”函数写入函数符号表。

  生成中间代码时，遍历一遍语法树，每遇到一个节点Node就调用其对应的“translateNode()”函数，被调用的函数会返回该子节点下的全部中间代码的双向链表，而父节点也就是调用者负责把它所有子节点返回的几段中间代码连接起来，直到root节点获得了全部的中间代码。

  对于数组和结构体，添加了函数“translateStructAddr(Node* expNode, ..., Operand* place)”以及“translateArrayAddr(Node* expNode, ..., Operand* place)”。当一个Exp节点发现它是一个数组访问，则调用translateArrayAddr，结构体则调用translateStructAddr，这两个函数都是能够计算出访问的地址，并把这个地址填写进place中。

  对于实参，由于数组和结构体传递的是**地址**，将传递的Operand的kind置为**OP_ADDR**；当且仅当打印PARAM时，OP_ADDR将打印出“**&**”，且对于函数之中类型为数组或结构体的参数，在该PARAM相关的translateExp中，不使用**取址(&)**符号。

四、编译

使用助教的Makefile可以成功编译。

运行时，可以选择：

- `./parser xxx.cmm`：翻译成中间代码并打印在控制台
- `./parser xxx.cmm out.ir`：翻译成中间代码并保存在.ir 文件中

五、注意事项（一些私设）

- 如果出现语法/词法错误，将不进行中间代码分析
- 如果出现作用域导致的语义错误（因为本人实验二实现了作用域），不报错，且不影响中间代码的生成（因为中间代码都是全局）