编译原理实验三 实验报告

分工

小组由三位同学组成,分工如下:

1. 表达式部分：丁楚阳
2. 函数部分：付安胜

后续代码调试与优化：丁楚阳，王文昊

文档撰写：王文昊

协作方式

小组建立了GitHub仓库,成员将本地测试成功的文件上传并与主分支merge,让每个成员都能时刻同步最新的代码,保证成员之间的有效交流,GitHub地址如下:

https://github.com/KaoriMiyazonoApril/NJU_Compiler_Design

编译方法

小组沿用了助教团队提供的makefile 文件,只需要在makefile文件目录下make即可得到parser可执行文件

功能和实现方法

小组实现了中间代码生成模块,从而将经过语义分析的语法树转化成三地址码形式的中间表示,该中间代码可被虚拟机解释执行

IR数据结构构建

采用结构化的设计方式定义IR所需的核心数据结构，包括操作数、中间代码指令和代码列表等

IR操作和管理函数

实现了完整的IR操作和管理函数，包括创建、删除和打印等功能

```
// 创建新的操作数
pOperand newOperand(OperandKind kind, ...);

// 删除操作数
void deleteOperand(pOperand p);

// 打印操作数
void printOp(FILE *fp, pOperand op);

// 创建新的中间代码
pInterCode newInterCode(InterCodeKind kind, ...);

// 删除中间代码
void deleteInterCode(pInterCode p);

// 打印中间代码
```

```
void printInterCode(FILE *fp, pInterCodeList list);

// 创建新的中间代码列表
pInterCodeList newInterCodeList();

// 删除中间代码列表
void deleteInterCodeList(pInterCodeList p);

// 添加中间代码到列表
void addInterCodes(pInterCodeList list, pInterCodes code);
```

核心翻译逻辑实现

遍历语法树，完成对应的中间代码生成，相应函数的作用如下注释所示：

核心思路：generateIRFromAST()为入口函数，各translate*函数根据语法规则处理对应的语法结构，最终生成三地址码形式的中间表示。

中间代码生成功能说明：

1. 表达式翻译：
    - 支持变量引用、常量和算术运算的翻译
    - 实现常量折叠优化，在编译期计算常量表达式的值
    - 采用translateToOperand辅助函数避免不必要的临时变量
2. 数组处理：
    - 区分参数数组和局部数组的不同地址获取方式
    - 实现多维数组元素访问的偏移量计算
    - 支持数组元素的读取和赋值操作
3. 控制流翻译：
    - 使用短路求值优化逻辑运算（AND/OR）
    - 将关系表达式转换为条件跳转指令
    - 正确处理循环和分支结构的标签生成
4. 函数调用机制：
    - 参数从右到左的压栈顺序
    - 特殊处理内置函数read/write
    - 数组参数传递时自动获取地址
5. 内存管理：
    - 自动生成唯一的临时变量名和标签名
    - 实现基本的资源释放机制
    - 避免内存泄漏和重复分配

```
// 主程序入口，进入语法树到IR的翻译
void generateIRFromAST(pNode root);

// 翻译表达式节点
static void translateExpNode(Node *node, pOperand place);
```

```
// 翻译条件表达式节点
static void translateCondNode(Node *node, pOperand labelTrue, pOperand
labelFalse);

// 翻译函数调用参数
static void translateArgsNode(Node *node);

// 获取二元运算对应的IR指令类型
static InterCodeKind getBinaryOpKind(const char *typeName);

// 将表达式翻译为操作数
static pOperand translateToOperand(Node *node);

// 计算多维数组偏移量
static ArrayOffsetResult computeArrayOffset(pOperand baseAddr, Node **idxNodes,
int idxCount, Type *arrayType);

// 判断是否为参数数组名称
static bool isParameterArrayName(const char *name);
```