**编译原理研讨课PR002实验报告**

小组成员：

杜旭蕾 2022K8009929003 王锦如 2022K8009929022

一. 实验任务

完成中间代码的生成，生成LLVM IR。

二. 设计思路

在生成中间代码之前，需要完成语法分析，具体更改如下：

//消除左递归

mulExp : unaryExp ( (Star | Div | Mod) unaryExp )\* ;

addExp : mulExp ( (Plus | Minus) mulExp )\* ;

relExp : addExp ( (Less | Greater | LessEqual | GreaterEqual) addExp )\* ;

eqExp : relExp ( (Equal | NotEqual) relExp )\* ;

lAndExp: eqExp ( AndAnd eqExp )\* ;

lOrExp : lAndExp ( OrOr lAndExp )\* ;

stmt : exp [Assign exp] Semi

...

在完成语法分析对g4文件的修改之后，按照以下思路设计生成IR的代码：

1. 最基本cact代码生成IR

首先针对以下代码内容设计IR的生成代码：

int main(){  
 int a = 0;

return 0;

}

运行实验1的代码，获得如下AST：

(compUnit (funcDef (funcType int) main ( funcFParams ) (block { (blockItem (decl (vardecl (bType int) (varDef a = (constInitVal (constExp (number 0)))) ;))) (blockItem (stmt return (exp (addExp (mulExp (unaryExp (primaryExp (number 0)))))) ;)) })) <EOF>)

针对这个代码，我们首先设计一个简单的生成IR的代码。这个代码没有实现符号表和函数表，其visit函数只包括以上AST中出现的节点名。主要实现非数组变量的vardecl以及stmt模块return语生成IR。运行并调试这个简单代码得到如下.ll文件，使用clang将其编译为可执行代码，获取其程序退出码，值为0，因此这个IR代码没有问题。

define i32 @main() {

entry:

%a = alloca i32, align 4

store i32 0, i32\* %a, align 4

ret i32 0

}

1. 扩展代码

逐步使用更加复杂的代码，发现当前代码运行出现问题，则为当前生成IR的代码逐步添加并完善其他节点的处理、加入符号表和函数表的实现、加入6个内置函数的声明并实现它们的运行时库。重复以上过程直到实现一个可以处理所有cact测试代码的程序。

三. 代码实现

本次实验代码具有如下结构：

cact/

│── build/

│── build.sh # 编译生成compiler

│── libcact\_rt.a # 运行时库

│── run\_code.sh # 将.ll文件链接运行时库生成可执行文件

│── grammar/

│── Hello.g4

│── src/

│── io/ # 生成运行时库的文件

│── main.cpp # compiler的main函数

│── Analysis.cpp # 生成IR的函数（主要为visit节点函数）

│── include/

│── Analysis.h # Analysis.cpp的头函数

│── SymbolTable.h # 符号表及其相关函数实现

│── CMakeLists.txt # 添加安装的llvm为path

llvm\_install/

│── clang+llvm-11.1.0-x86\_64-linux-gnu-ubuntu-16.04/ # 安装的llvm和clang

其中，本次实验生成IR的核心函数集中于Analysis.cpp文件和SymbolTable.h文件。

Analysis.cpp 文件中主要实现了所有直接使用 ANTLR 生成的访问器接口函数，这些接口继承自 HelloBaseVisitor。这类函数的命名规则为 "visit" + AST 节点名称，比如：visitCompUnit、visitFuncDef。每个函数对应语法树中的一个节点类型，其作用是定义访问该节点时应执行的逻辑，在语法树的遍历过程中可以实现IR的生成。

SymbolTable.h文件主要实现了符号表及其相关函数的实现。

由于Analysis.cpp代码很长，实验报告代码实现部分主要选取这两个文件中重点部分进行分析，内容如下：

1. 符号表实现（包含写入、作用域、查找…）
2. 变量和常量声明与初始化
   1. 常量声明与初始化
   2. 变量声明与初始化
3. 函数声明
   1. 函数表实现
   2. 函数参数处理方法
4. Stmt模块实现
   1. 表达式：exp [Assign exp] Semi
   2. 返回语句：RETURN\_KW exp? Semi
   3. Block块：block
   4. IF语句：IF\_KW LeftParen cond RightParen stmt ( ELSE\_KW stmt )?
   5. WHILE语句：WHILE\_KW LeftParen cond RightParen stmt
   6. BREAK语句：BREAK\_KW Semi
   7. CONTINUE 语句：CONTINUE\_KW Semi ;
5. 短路求值实现
6. Lval模块实现
   1. Lval是常量或无下下标数组访问
   2. Lval是有下标数组访问
7. Number模块实现
8. 内置函数实现方法（包括静态链接库生成、run\_code.sh）

四. Debug过程

1. 函数数组形参的存储和使用（isArrayinFunc字段）

2. 函数数组形参符号表的插入（dims字段）

3. IF、WHILE条件的短路求值

五. 实验结果

六. 实验总结