编译原理 lab6

一、项目结构

本次实验项目结构较上次实验基本没有变化。主要还是在 LLVMvisitor 类上进行函数的重载。

二、实验思路

1、对全局变量进行处理：

首先在 visitVarDecl 中，先判断当前作用域是否是 global，之后再通过对 constExpContext 是否为 null 的判断，来判断是否为数组。之后创建数组操作和局部变量差别不大，只不过使用 LLVMAddGlobal 来建立 LLVMValueRef。全局数组就算没有初始化也要将所有值都置为 0，由于没有 builder，故不能使用 GEP 指令，只能建立一个 LLVMConstArray 存放数组初始化的值，之后再通过 LLVMSetInitializer 将值传入数组进行初始化。

全局变量则更为简单，直接使用 LLVMAddGlobal 来建立 LLVMValueRef，之后判断是否要初始化，需要则使用 LLVMSetInitializer 进行初始化。

上述全局变量和数组也都要再 currentScope 中进行 define。

2、对 if else 语句处理：

首先新创建三个 block，分别为 true，false，entry，分别代表 if{}, else{}以及之后的函数的其他部分。

首先要对 if(cond)中的 cond 部分进行判断，即 visit(cond)，若结果为 i1 则 LLVMBuildCondBr 进行跳转，否则(为 i32，即 cond 部分是 exp)则通过 LLVMBuildICmp (builder, LLVMIntNE, condRe, zero,"");判断是否为 0，将结果传入 LLVMBuildCondBr 进行跳转。

之后按顺序使用 LLVMPositionBuilderAtEnd 将 visit(stmt(0))和 visit(stmt(1))得到的 IR 语句分别加到 true 和 false block 中，并使得二者最后都通过 LLVMBuildCondBr 跳转到 entry block 中。

之后则是对 cond 语句的判断的函数的重载，主要重载的函数都是下面这些语句相关

```
cond
    : exp #expCond
    | cond (LT | GT | LE | GE) cond #ltCond
    | cond (EQ | NEQ) cond #eqCond
    | cond AND cond #andCond
    | cond OR cond #orCond
    ;
```

ExpCond 不需要重载

LtCond 和 EqCond 需要使用 LLVMBuildICmp，且为了防止出现 a<b<c 这种，在通过 visit 得到两边的操作数后要先对其进行判断，类型是否是 i1.若是，则通过 LLVMBuildZExt 转为 i32 再参与处理。

And 和 or 本次实验并未考察，但仍进行了相应处理，不再赘述。

三、遇见的问题

1、关于大小比较，我的第一反应是从 LLVMValue 中得到 int 值再进行比较，但关于全局变量和数组使用 LLVMConstIntGetSExtValue 得到的 int 值是乱码，最后在仔细阅读 LLVM 文档后以及在 lli 编译帮助下使用 LLVM 相应 api 解决。

2、LLVMBuildIcmp 按理来说返回的应该是个 i1 类的 LLVMConstInt，故我定义了两个分别为 0 和 1 的 i1 类的 LLVMConstInt，来和 LLVMBuildIcmp 返回值通过.equal 进行比较，但结果都是 false。最后则使用其他方法，规避了对 LLVMBuildIcmp 返回值究竟是 true 还是 false 的判断。