Lab5

201250203 张若皓

实验思路

我个人觉得和Lab3的思路还是差不多的。（可能因为都在visitor里进行实现）

总体来说，我将Lab3中的scope进行简化之后放入了这次的项目框架内部，不存在什么很复杂的继承关系，就是作为一个变量放在visitor中间帮助实现符号表。

在每一次定义一个函数或者一个变量的时候都把对应的LLVMValueRef放到当前scope里的符号表中。在找的时候就再取出，整体而言和Lab3非常像。

剩下的内容几乎都是照葫芦画瓢，先声明变量空间，然后如果有初始值就存进去，如果没有的话就不存。

此外对于函数的参数，我觉得与局部变量并没有什么不同，在函数定义时先把参数固定下来，然后在访问参数的时候就将其空间分配好，随后将在函数定义时确定下来的参数BuildStore给这些空间。

精巧设计

我在写visitLval的时候，返回的都是已经BuildLoad之后的内容所生成的LLVMValueRef，导致我在assign的时候会出问题。我的解决办法是，在assign的时候，对于等号左边的值进行特殊化处理，我不使用visitLval得到对应LLVMValueRef，而是将visitLval的除了BuildLoad的内容搬到assign那边。（选择这么做的原因是：在我写lab4的时候，我默认里面的值都是buildload之后的内容，修改起来会有点麻烦。）

你遇到的困难及解决办法(遇到的奇怪bug)

遇到的困难还是不太了解LLVM到底怎么使用。当时写Lab4的时候，有点囫囵吞枣地看完了助教的api教程，导致刚开始走了不少弯路。然后根据生成出来的IR代码感觉非常直观地能够看懂。

还有一个困难就是我的精巧设计。这里就不再多复述了。

最后是oj上报的错为 **error: expected instruction opcode }**，后来看到群里有人在讨论这个问题提到了有可能void类型的函数没有return语句。我的解决方案也算有点**精妙（？）**，我在visitStmt的return语句的时候，将一个全局变量flag改成true,那么在返回到funcdef的时候对这个flag进行检查，如果是true，那么就不需要buildret。如果需要buildret，那么也就说明这个函数一定是void类型函数，那么就直接buildretvoid即可。