## 实验感想

本次实验在上次语法分析产生的抽象语法图的基础上实现语义分析，生成LLVM的IR，也就是LLVM的中间表示。由于是在助教给出的框架上完成的任务，所以需要趁热打铁，不然忘记了`asg.hpp`里面的各种结构就很麻烦。

1. 首先需要处理一些经常出现的节点，比如`ImplicitCastExpr`语句非常常见，它常用于**将数组转成指针、将左值转成右值**，这应该是出现次数最多的节点了
2. 接下来处理一些数组操作，这也是笔者花费时间最多的地方，理解GEP指令以及对指针的操作还是比较耗时的
3. 再就是分支跳转的实现了，这部分用栈实现比较容易，但是有很多需要注意的点，**稍不留神就会导致基本块没有终结指令**，另外还有`break, continue`的实现也会用到栈
4. 最后剩下的都是一些简单的中间代码翻译了，像函数调用、变量声明、全局和局部变量的管理、函数返回等等，这些都很简单

通过本次实验我不仅学到语义分析的相关知识，还学到了很多实际操作，比如LLVM的IR的生成、基本块之间的跳转等等，特别是在看到添加修改代码带来的$PASS$，这感觉真的很棒。语义分析这次实验不仅考验API的使用，还考验算法能力（虽然笔者的代码疏于管理看着超级难受就是了~~），需要分析答案IR去得到如何生成它们的思路。

## 实验改进建议

1. 可以让实验者知道更多的分析方式，比如用`clang -Xclang -ast-dump -fsyntax-only`命令输出语法分析树，这真的很方便，**因为输出的数据结构名称含义和助教给出的框架的基本一致，这对于测试案例的调试很有帮助**
2. 可以给出更多GEP指令和一些关键点比如短路求值、跳转语句翻译的提示，比如用栈实现跳转基本块的生成，不然还是比较难上手的
3. 测试案例`if-combine1.sysu.c`中出现了变量`m`的重定义，不过因为两次定义是在不同作用域所以不会报错，**但我认为还是需要提醒以下同学们**，因为在处理`VarDecl`的时候如果没有正确对待这个重定义变量，程序可以正确转换生成中间表示不会报错，**但是输出值会不一样**，**这会导致同学直接去检查中间代码逻辑而忽略检查源程序的特点，会浪费一些不必要的时间**