编译原理 Lab 5 实验报告

学号：201250185

姓名：王福森

我实现的功能

本次实验我通过 antlr 工具和 llvm工具，在 SysYLexer.g4 和 SysYParser.g4 文件基础上，加之对 lab 3 实现的符号表稍微修改（即为每个 Symbol 具体类增加一个 LLVMValueRef 字段，为Symbol 接口增加对 LLVMValueRef 的get 和 set 方法），对 lab 4 生成的 IR 进行了扩充，实现了函数定义与调用，局部变量的声明、定义、使用的 IR 生成。

我是这么实现的

本次实验直接沿用 Lab 1、 Lab 2、Lab 3 所写的 SysYLexer.g4、 SysYParser.g4 、TypeCheckListener.java文件。

在 main 函数中，我先遍历一遍语法分析树，构建基本的符号表，再遍历一遍语法分析树，实现符合要求的 IR 的生成。其中，构建基本的符号表我沿用的是 lab 3 的 TypeCheckListener；为了生成符合要求的 IR， 我设计了一个继承 SysYParserBaseVisitor<LLVMValueRef> 的 FunctionAndVarIRVisitor，在遍历语法分析树的过程中构建逐步向符号表中各个符号填充 LLVMValueRef 字段，同时向 module 中加入函数、基本块和指令，用以生成符合文档要求格式的输入文件的 IR。

本次处理的核心是函数的定义、常量的定义与初始化、变量的定义与初始化、各种表达式的翻译。对于各个语句，只需要按照 sysY 语言的语义生成对应的指令即可。

我的精巧设计

本次实验可操作性较少，基本都是对着 sysY 语言的每条语义规则逐条翻译指令，没有什么精巧的设计。

过程中有趣现象和印象很深的 bug

这次 Lab 我做的还算顺利，总共遇到 5 个 bug，每个 bug 解决的时间可以接受。

有其中的 3 个 bug 是我漏掉了实验文档的部分要求，这一些虽然难发现 bug 在哪，但是一发现可以很快解决，庆幸的是我没有花很多时间来发现这 3 个 bug。比较头疼的是另外两个 bug，这两个 bug 依赖于我对于 LLVM api 的不熟悉以及 LLVM 文件格式的不充分理解。一个是我在处理函数调用表达式的时候，需要将每个实参加到 PointerPointer 中再传入 LLVMBuildCall 中，我在逐个加入 PointerPointer 的时候调用了错误的 put 方法，导致出错。这个 bug 是最耗时间的，我一度怀疑我 LLVMBuildCall 的传参用错了，查了好多资料，最后终于搞定。最后一个 bug 是 void 函数省略的 return 语句我应该增加对应的指令，这个 bug 我是在同学的帮助下发现的。