编译原理 Lab 6 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 lab 5 的基础上，扩充了 SysY 语言的 IR 生成程序 FunctionAndVarIRVisitor.java，实现了全局变量、常量，全局变量数组、常量数组的声明、定义和初始化，实现了 if 语句的翻译。

我是这么实现的

本次实验直接沿用 Lab 1 到 Lab 5 所写的 SysYLexer.g4、 SysYParser.g4 、TypeCheckListener.java、FunctionAndVarIRVisitor.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。

本次处理的核心是全局常量的定义与初始化、全局变量的定义与初始化、if 语句和条件表达式的翻译。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

我设计了一个变量 isReturn 用于处理 if 语句里面有 return 语句导致报错 “error: instruction expected to be numbered”，在函数定义的时候置 false，在 return 语句的时候置 true。

在处理 if 语句的时候，如果 isReturn 是 true，说明该 if 语句内除了嵌套的 if 语句内，还有返回语句，故此时不需要跳转到 exit block，从而避免 llvm 编译器命名错误。

过程中有趣现象和印象很深的 bug

这次 Lab 我遇到一个很有意思的 Bug。

我所有完成提交后发现有 4 个测试用例报 “error: instruction expected to be numbered”，查了 stackoverflow 后得知是在 return 语句后如果紧接着 br 语句会隐式转化成另一个 basic block，这个 basic block 会占用一个默认编号，如果后续的变量都是用默认编号的话就会导致编号偏移产生错误。

对此我想到两个解决方法。一个是所有的 Build 语句都不用空串，另一个就是采用我的精巧设计，判断 if 语句里面有没有 return 语句。