实验报告—step6

2018011340 计 83 郭峥岩

一、 实验内容

1. 整体框架

本次实验中 step6 的整体架构与之前的框架相比最重要的是增加了一个 if-else 语句产生不同的跳转的指令，以及在遍历 AST 树生成相对应的 IR 指令序列的时候改如何进行生成。

我们采取的管理跳转指令的 label 的方式同管理局部变量的栈帧的偏移的方式，即在 ir 代码的生成遍历的类中增加一个管理者。

2. Step6 的功能实现

（1） 实现程序的语法规范的改动：

增加新的 if-else，条件表达式语句的对应语法。

（2） IR 类的更改和产生：

此次 step6 需要使用到的对应的 IR 类已经在 step5 中定义，因此不需要定义新的 IR 类。

重点在于跳转指令的管理和遍历 AST 树产生 IR 类的顺序。管理类也是在遍历过程中遇到 if-else 语句进行判断，根据不同的条件以及结果产生不同的跳转管理以及树的遍历顺序，此处的详细过程参考实验指导书上的内容进行实现，此处不再赘述.

（3） 汇编代码的产生：

汇编代码的产生即按照实验指导书的内容将其从 IR 指令逐步翻译成汇编指令即可。

为了方便之后的步骤的使用，此处可以进行代码结构的调整和存储。

二、 思考题

Rust 和 Go 语言中的 if-else 语法与 C 语言中略有不同，它们都要求两个分支必须用大括号包裹起来，而且条件表达式不需要用括号包裹起来：

```
if 条件表达式 {
  // 在条件为 true 时执行
} else {
  // 在条件为 false 时执行
}
```

请问相比 C 的语法，这两种语言的语法有什么优点？

解答：优点：

解决了悬挂 else 的问题，而且语义清晰，方便阅读；同时条件表达式不用括号进行包裹可以使得语法更加自由，比如 RUST 中有类似“if let”的语法。

同时用大括号可以明确确定 then 与 else 语句的范围,也可以避免写错。

三、 参考资料

助教所写的代码：md-dzy branch

四、 总结

本次实验过程遇到的问题：之前的 step 自定义的 errorListener 处理二义性报错的时候无法处理，查阅资料之后定义了新的 errorListener 来解决问题。

整体而言在 step5 的基础之上增加一些对 if，else 语句等的判断即可，大致原理同 step5，对代码的改动相对较多。