实验报告—step8

2018011340 计 83 郭峥岩

一、 实验内容

1. 整体框架

本次实验中 step8 的整体架构与之前的框架相比没有发生变化。

重点是增加了对循环语句，以及对 break，continue 的解析，并且对同一个非终结符进行重命名产生不同节点。

2. Step8 的功能实现

（1） 实现程序的语法规范的改动：

新增几种 for 循环，while 循环以及 break，continue 的语义解析。

（2） IR 类的更改和产生：

IR 类不需要进行更改。

遍历 AST 树产生相应 IR 中间码时增加对循环节点的解析。同时对同一的循环节点，参照助教的实现方式，实现了一个从循环节点生成 IR 的函数。将循环的初始化条件，终止条件，迭代条件都传入即可产生相应的 IR 代码。

（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。

3. 实现的思想和细节

其实本次实现循环语句的思想沿用 step7 中对于符号表的思想分析，新增循环类来维护标号的栈，每次遇到循环就将标号进行相应的迭代。然后把产生的循环标志压入对应的栈中。如果遇到 break 语句，其跳转目标就是 break 标号栈的栈顶，如果栈为空就报错。离开 Loop 的时候弹栈。

二、 思考题

1. 将循环语句翻译成 IR 有许多可行的翻译方法，例如 while 循环可以有以下两种翻译方式：

   第一种（即实验指导中的翻译方式）：

   1. `label BEGINLOOP_LABEL`：开始下一轮迭代
   2. `cond 的 IR`
   3. `beqz BREAK_LABEL`：条件不满足就终止循环
   4. `body 的 IR`
   5. `label CONTINUE_LABEL`：continue 跳到这
   6. `br BEGINLOOP_LABEL`：本轮迭代完成
   7. `label BREAK_LABEL`：条件不满足，或者 break 语句都会跳到这儿

   第二种：

   1. `cond 的 IR`
   2. `beqz BREAK_LABEL`：条件不满足就终止循环
   3. `label BEGINLOOP_LABEL`：开始下一轮迭代
   4. `body 的 IR`
   5. `label CONTINUE_LABEL`：continue 跳到这
   6. `cond 的 IR`
   7. `bnez BEGINLOOP_LABEL`：本轮迭代完成，条件满足时进行下一次迭代
   8. `label BREAK_LABEL`：条件不满足，或者 break 语句都会跳到这儿

   从执行的指令的条数这个角度（`label` 指令不计算在内，假设循环体至少执行了一次），请评价这两种翻译方式哪一种更好？

   解答：我认为第二种方式会更好。

   原因：执行一次循环体：

   第一种方式：cond , deqz, body , br , cond , beqz六个部分；

   第二种方式：cond, beqz , body, cond, bnez五个部分。

   并且每次执行多循环一次，第一种方式则需要执行body, br, cond, deqz四个部分，而第二种方式执行body, cond, bnez三个部分。

   综上所述：执行体执行至少一次的情况下，第二种方式执行的指令的条数会更少，效果更好。

三、 参考资料

助教所写的代码：md-dzy branch

四、 总结

本次的实验过程即增加几条语义分析，根据相应的语义在遍历 AST 树产生 IR 代码的时候进行了一些增加，相对而言比较容易，按照实验指导书上的正常顺序进行并未遇到什么太大的问题。