stage-3 实验报告

计 93 王哲凡 2019011200

实验内容

step7：作用域和块语句

首先，在符号表建立的阶段，需要通过 `ctx.open(Scope(ScopeKind.LOCAL))` 在 `visitBlock` 访问基本块时，开启一个新的局部作用域，并在结束时通过 `ctx.close()` 关闭。

在目标代码生成的寄存器分配过程中，对于无法到达的基本块，即 `graph.vis[i] == 0` 的第 i 个基本块（$i \neq 0$），需要跳过寄存器分配阶段（基本块不可达），而 `vis` 数组的判定可通过一个从 0 号基本块开始的深度优先搜索完成。

step8：循环语句

首先在词法语法分析阶段，需要在 `frontend/lexer/lex.py` 中添加对应的保留字：

```
reserved = {
    "return": "Return",
    "int": "Int",
    "if": "If",
    "else": "Else",
    "for": "For",
    "while": "While",
    "do": "Do",
    "break": "Break",
    "continue": "Continue",
}
```

其次，也需要在 `frontend/parser/ply_parser.py` 中添加对应的解析规则：

```
def p_for(p):
    """
    statement_matched : For LParen opt_expression Semi opt_expression Semi opt_expression RParen statement_matched
        | For LParen declaration Semi opt_expression Semi opt_expression RParen statement_matched
    statement_unmatched : For LParen opt_expression Semi opt_expression Semi opt_expression RParen statement_unmatched
        | For LParen declaration Semi opt_expression Semi opt_expression RParen statement_unmatched
    """
```

```
    p[0] = For(p[3], p[5], p[7], p[9])

def p_do_while(p):
    """
    statement_matched : Do statement_matched While LParen expression RParen Semi
    """
    p[0] = DoWhile(p[2], p[5])

def p_continue(p):
    """
    statement_matched : Continue Semi
    """
    p[0] = Continue()

```

具体规则参见任务概述。

为了配合语法词法分析，需要在 `frontend/ast/tree.py` 添加对应的抽象语法树节点类型 `For`、`DoWhile`、`Continue`，并在 `frontend/ast/visitor.py` 中实现对应的基类 `visit` 函数。

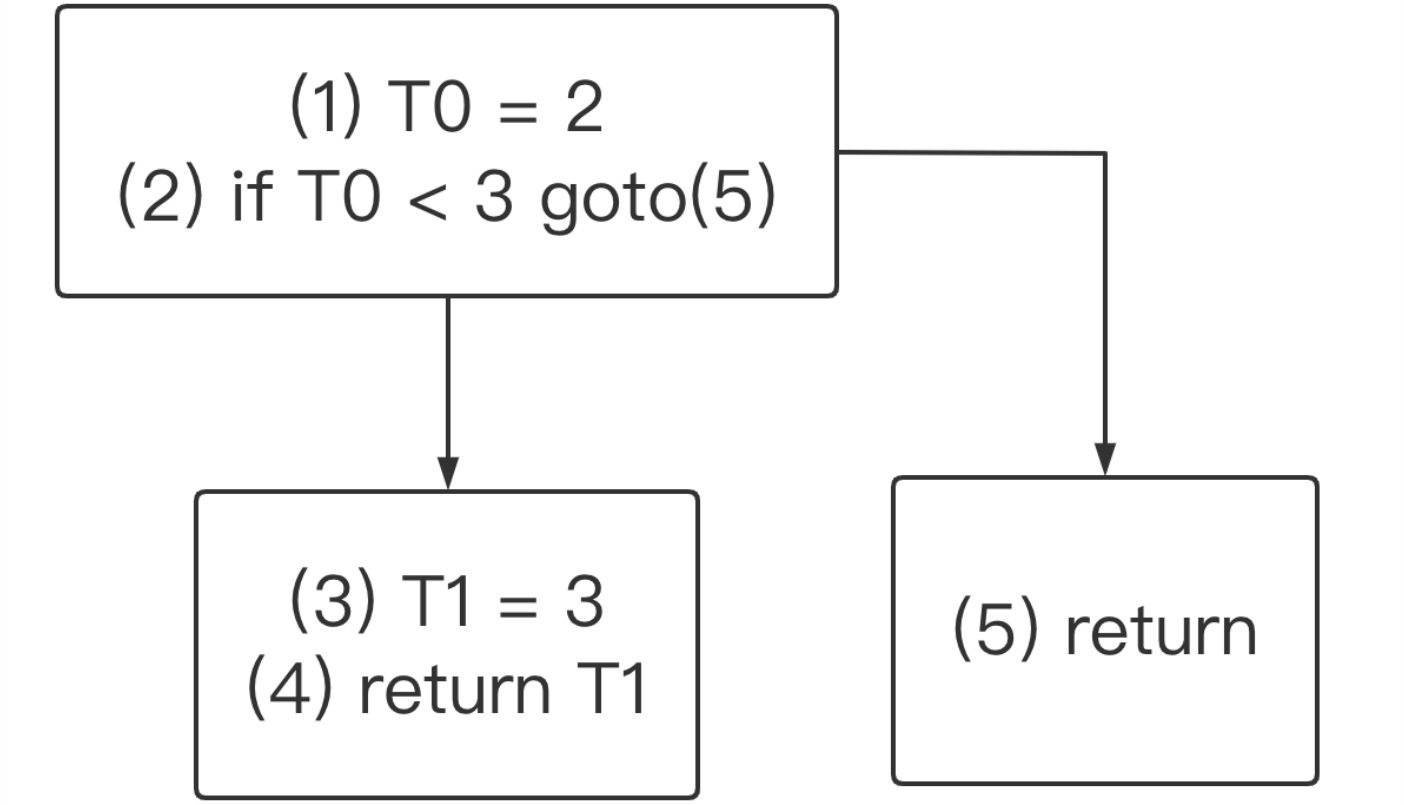
在 `frontend/typecheck/namer.py` 中原来注释的 `visitFor`、`visitDoWhile`、`visitContinue` 函数根据提示填充完毕，具体细节参照了 `visitWhile` 和 `visitBreak`。

最后，还需要在中间代码生成阶段完成对应的 `visit` 函数编写，具体来说，需要在 `frontend/tacgen/tacgen.py` 中添加上述 `visit` 函数，同样可参考 `visitWhile` 和 `visitBreak`。

特别注意的是，对于 `For` 的 `cond`，需要判断 `not isinstance(stmt.cond, node.NullType)` 才可进行相关的 `visit` 调用和跳转。

实验思考题

step7：作用域和块语句

1. 

step8：循环语句

1. 设 cond 对应 IR 的指令数为 n，body 对应 IR 的指令数为 m，假设进行了 $k \geq 1$ 次循环，通过 continue 或 break 减少了 body 内共 M 次指令的执行。

   对于第一种：

   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

   总共会执行：

   $$N_1 = (n+1+m+1) \times k + n + 1 - M = (k+1)(n+1) + k(m+1) - M$$

   对于第二种：

   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

总共会执行：

$$N_2 = n + 1 + (m + n + 1) \times k - M = (k + 1)(n + 1) + km - M$$

可知：

$$N_1 - N_2 = k \geq 1 \Rightarrow N_2 < N_1$$

即第二种翻译方式更好。