stage-2 实验报告

秦若愚 2019011115

实验内容

step-5

实验目标

- 增加变量的声明
- 增加变量的使用（读取/赋值）
- 增加语义检查：变量不能重复声明，不能使用未声明的变量

实验内容

首先在 frontend/typecheck/namer.py 中处理 Declaration，Assignment 和 Identifier，用于建立符号表。处理 Declaration 时，首先查询符号是否被声明，若已被声明则报错，否则在符号表中声明该符号，当 Declaration 右侧是一个表达式时需要递归处理。处理 Assignment 时，若表达式左侧不是左值则报错，否则调用与 VisitBinary 相同的逻辑。在处理 Identifier 时，从符号表中查询该 ident 对应的符号，若存在则赋给该 ident，否则报错。

其次在 frontend/tacgen/tacgen.py 中处理 Declaration，Assignment 和 Identifier，用于生成 TAC。处理 Declaration 时，为 ident 对应的符号创建一个新寄存器，当 Declaration 右侧是一个表达式时需要为寄存器赋初始值。处理 Assignment 时，生成赋值中间语句。处理 Identifier 时，将 ident 对应的符号的寄存器分配给该 ident。

最后在 backend/riscv/riscvasmemitter.py 中添加 VisitAssign() 函数，用于生成赋值语句对应的指令，注意右侧表达式为 None 时不需要生成这条指令。

step-6

实验目标

- 支持 If 表达式
- 支持条件表达式

实验内容

因为已经提供了 If 语句的实现，所以只需在 frontend/typecheck/namer.py 和 frontend/tacgen/tacgen.py 中完善 visitCondExpr() 函数即可。在 namer.py 中，需要检查 cond，then 和 otherwise 三部分。在 tacgen.py 中，需生成条件表达式的 TAC，具体实现是设置 skipLabel 和 exitLabel 各一个，skipLabel 置于 otherwise 块之前、then 块之后，根据 cond 选择是否跳转。exitLabel 置于 otherwise 块之后，在 then 块完成后跳转。

思考题

step-5

1. 答：

```
addi sp, sp, -16
```

2. 答：

Scope 类的成员变量 symbols 应改为如下类型：`{str: list[Symbol]}`，即每一个 name 对应一个 Symbol 的列表，同时每个 name 还对应一个计数器 count（从 0 开始），`symbols[name][count]` 表示第 count + 1 次声明的名字为 name 的变量对应的符号。

在定义变量和查找变量时，需要按照从上至下、每个 Declaration 先访问初始化部分再访问声明变量的顺序，且每次访问一个声明变量之后，对应 name 的计数器加 1。如此，可以将多次定义的同名变量区分开。

step-6

1. 答：

Python 框架中关于 If 语句的 parser 设置如下：

```
def p_if_else(p):
    """
    statement_matched : If LParen expression RParen statement_matched Else
statement_matched
    statement_unmatched : If LParen expression RParen statement_matched Else
statement_unmatched
    """
    p[0] = If(p[3], p[5], p[7])


def p_if(p):
    """
    statement_unmatched : If LParen expression RParen statement
    """
    p[0] = If(p[3], p[5])
```

可以看到，对于 `if(a) if(b) c=0; else d=0;` 语句，若第一个 if 与 else 结合，则按照 p_if_else 设置 `if(b) c=0;` 只能对应 statement_matched，而 p_if 设置中没有 statement_matched，所以这种情况不会发生。由此便解决了悬吊 else 问题。

2. 答：

需要先顺序访问 cond、then、otherwise，再根据 cond 的值选择将 then 或 otherwise 的结果作为最终返回结果。具体代码如下：

```
def visitCondExpr(self, expr: ConditionExpression, mv: FuncVisitor) -> None:
    expr.cond.accept(self, mv)
    expr.then.accept(self, mv)
    expr.otherwise.accept(self, mv)
    skipLabel = mv.freshLabel()
    exitLabel = mv.freshLabel()
    temp = mv.freshTemp()
    mv.visitCondBranch(
        tacop.CondBranchOp.BEQ, expr.cond.getattr("val"), skipLabel
    )
    mv.visitAssignment(temp, expr.then.getattr("val"))
    mv.visitBranch(exitLabel)
    mv.visitLabel(skipLabel)
    mv.visitAssignment(temp, expr.otherwise.getattr("val"))
    mv.visitLabel(exitLabel)
    expr.setattr("val", temp)
```