Stage-2 : 变量和语句

陈炫中　2019011236

- **Step 5**

实验内容

增加变量的声明和使用（读取/赋值）。

`namer.py` 中的 `ctx` 是一个 `Scope` 对象，通过其来维护 `main` 函数中所有出现的变量符号。

例如在变量声明时，通过调用 `findConflict` 来检查该变量是否已经存在，不存在时新建变量符号 `VarSymbol` 并放入当前 `ctx`，当该声明的变量有初值时，对其值进行访问：

```
if ctx.findConflict(decl.ident.value) is None :
    symbol = VarSymbol(decl.ident.value, decl.var_t)
    ctx.declare(symbol)
    decl.setattr('symbol', symbol)
    if decl.init_expr != NULL:
        decl.init_expr.accept(self, ctx)
```

在变量的使用时，通过调用 `lookup` 在 `ctx` 中寻找当前的变量符号是否存在，存在使返回该变量符号的属性，否则报 `undefinedVarError`：

```
symbol = ctx.lookup(ident.value)
if symbol is None:
    raise DecafUndefinedVarError(ident.value)
else:
    ident.setattr('symbol', symbol)
```

此外，`visitAssignmet` 可参考 `visitBinary` 类似实现，不再赘述。

三地址码生成阶段，在访问标识符时，通过如下方式将 `ident` 的 `val` 属性设置为 `ident` 的 `symbol` 属性的临时变量：

```
ident.setattr('val', ident.getattr('symbol').temp)
```

声明变量时，若变量有初值，通过 `visitAssignment` 进行赋值：

```
if not decl.init_expr is NULL:
    decl.init_expr.accept(self, mv)
    mv.visitAssignment(symbol.temp, decl.init_expr.getattr('val'))
```

思考题

1.

risc-v 汇编代码编写如下：

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

2.

在定义变量时，不再需要去判断该变量是否已经被定义；

对于查找变量的逻辑不需要进行修改。

- **Step 6**

实验内容

要支持 if 语句和条件表达式。

借鉴 if 语句的实现，将条件表达式 `visitCondExpr` 实现如下：

```
def visitCondExpr(self, expr: ConditionExpression, mv: FuncVisitor) -> None:
    """
    1. Refer to the implementation of visitIf and visitBinary.
    """
    expr.cond.accept(self, mv)
    skipLabel = mv.freshLabel()
    exitLabel = mv.freshLabel()
    mv.visitCondBranch(
        tacop.CondBranchOp.BEQ, expr.cond.getattr("val"), skipLabel
    )
    expr.then.accept(self, mv)
    mv.visitBranch(exitLabel)
    mv.visitLabel(skipLabel)          # visit skip label
    expr.otherwise.accept(self, mv)  # otherwise expression（与 visit skip label
交换位置后，条件表达式不短路）
    mv.visitAssignment(expr.then.getattr("val"),expr.otherwise.getattr("val")) #
与 if 语句不同，条件表达式具有返回值
    mv.visitLabel(exitLabel)
    expr.setattr("val", expr.then.getattr("val"))
```

在这一部分，条件表达式和 `if/else` 分支语句生成汇编代码的方式大体相同，需要注意的是，与 if 语句不同的是，条件表达式具有返回值，因此，我通过赋值语句 `mv.visitAssignment(expr.then.getattr("val"),expr.otherwise.getattr("val"))` 来实现为其分配临时变量的过程。

思考题

1.

在 `python` 框架中，悬吊 else 问题的处理是通过设置产生式的优先级，优先选择没有 else 的 if 来进行解决的。

在语法分析程序中，有如下规则：

```
Rule 9     statement -> statement_matched
Rule 10    statement -> statement_unmatched
Rule 11    statement_matched -> If LParen expression RParen statement_matched
Else statement_matched
Rule 12    statement_unmatched -> If LParen expression RParen statement_matched
Else statement_unmatched
Rule 13    statement_unmatched -> If LParen expression RParen statement
```

当遇到 `if(a) if(b) c=0; else d=0;` 时，运用以上文法规则：

对于第一个`if`，在使用`Rule 9 & 11`后，第二个`if`会被推导为`statement_matched`，但之后对这个`statement_matched`非终结符无法使用任何一条规则来继续向下推导。对于第一个`if`使用`Rule 10 & 12`后也会遇到同样的情况，因此对第一个`if`只能通过使用`Rule 10 & 13`，即选择其为没有`else`的`if`，之后对于第二个`if`时则将其选择为有`else`的`if`。

2.

在我的实现中，要求条件表达式不短路，可以通过将上述代码注释中`then expression`和`otherwise expression`中的位置调整到`mv.visitCondBranch(...)`之前，来使得条件表达式的`then`和`otherwise`部分的表达式一定被执行，而在`skipLabel`和`exitLabel`分支中只进行返回值的处理。

修改后的`visitCondExpr`如下所示：

```
def visitCondExpr(self, expr: ConditionExpression, mv: FuncVisitor) -> None:
    """
    1. Refer to the implementation of visitIf and visitBinary.
    """
    expr.cond.accept(self, mv)
    skipLabel = mv.freshLabel()
    exitLabel = mv.freshLabel()
    expr.then.accept(self, mv)       # then expression（将其放到mv.visitCondBranch后，条件表达式不短路)
    expr.otherwise.accept(self, mv)  # otherwise expression（将其放到mv.visitCondBranch后，条件表达式不短路)
    mv.visitCondBranch(
    tacop.CondBranchOp.BEQ, expr.cond.getattr("val"), skipLabel
    )

    mv.visitBranch(exitLabel)
    mv.visitLabel(skipLabel)         # visit skip label

    mv.visitAssignment(expr.then.getattr("val"),expr.otherwise.getattr("val")) # 与 if 语句不同，条件表达式具有返回值
    mv.visitLabel(exitLabel)
    expr.setattr("val", expr.then.getattr("val"))
```