STAGE2 实验报告

致理-数理 1 赵宇峰 2021013376

1. 实验代码任务

编写代码如下：

符号表`frontend/typecheck/namer.py`

Declaration set symbol; Identifier read symbol;

```
def visitDeclaration(self, decl: Declaration, ctx: Scope) -> None:
        """
        1. Use ctx.lookup to find if a variable with the same name has been declared.
        2. If not, build a new VarSymbol, and put it into the current scope using ctx.declare.
        3. Set the 'symbol' attribute of decl.
        4. If there is an initial value, visit it.
        """
```

根据要求，依次执行如下代码：

```
def visitDeclaration(self, decl: Declaration, ctx: Scope) -> None:
        symbol = ctx.lookup(decl.ident.value)//1.寻找 lookup table 是否已经声明该项
        if symbol is None://2.如果没有
            newvar = VarSymbol(decl.ident.value, decl.var_t.type)//build 新的 varsymbol
            ctx.declare(newvar)//放入 declare
            decl.setattr('symbol', newvar)//3.设置 decl 的 symbol 属性
        if decl.init_expr is not NULL://4.如果有初始值，访问该初始值（使用 visitor 模式的 accept 功能）
            decl.init_expr.accept(self, ctx)
```

visitAssignment 则直接调用 visitbinary:

```
def visitAssignment(self, expr: Assignment, ctx: Scope) -> None:
        if not isinstance(expr.lhs, Identifier):
            raise DecafSyntaxError('Invalid Assigment cause lhs of the expression is not a Idenfifier')
        self.visitBinary(expr, ctx)
def visitIdentifier(self, ident: Identifier, ctx: Scope) -> None:
        """
        1. Use ctx.lookup to find the symbol corresponding to ident.
        2. If it has not been declared, raise a DecafUndefinedVarError.
        3. Set the 'symbol' attribute of ident.
        """
```

根据要求，依次执行如下代码：

```
        symbol = ctx.lookup(ident.value)//1.
        if symbol is None:
            raise DecafUndefinedFuncError('Invalid behavior because identifier is not defined')//2.
        ident.setattr('symbol',symbol)//3.
```

TAC 生成 frontend/tacgen/tacgen.py

根据提示 def visitDeclaration(self, decl: Declaration, mv: TACFuncEmitter) -> None:

"""

1. Get the 'symbol' attribute of decl.
2. Use mv.freshTemp to get a new temp variable for this symbol.
3. If the declaration has an initial value, use mv.visitAssignment to set it.

"""

def visitDeclaration(self, decl: Declaration, mv: TACFuncEmitter) -> None:

decl.getattr('symbol').temp = mv.freshTemp()//1. 获取 symbol 属性 2. 使用 mv.freshtemp 为其赋值

if decl.init_expr is not NULL://3.使用 mv.visitAssignment 设置初始值

decl.init_expr.accept(self, mv)//模仿上方代码，使用 accept 访问 initial value

mv.visitAssignment(decl.getattr('symbol').temp, decl.init_expr.getattr('val'))//使用 visitAssignment 识别

def visitAssignment(self, expr: Assignment, mv: TACFuncEmitter) -> None:

"""

1. Visit the right hand side of expr, and get the temp variable of left hand side.
2. Use mv.visitAssignment to emit an assignment instruction.
3. Set the 'val' attribute of expr as the value of assignment instruction.

"""

expr.rhs.accept(self, mv)//访问左右 expression

expr.lhs.accept(self, mv)

expr.setattr('val',mv.visitAssignment(expr.lhs.getattr('val'),expr.rhs.getattr('val')))// 发送赋值指令，并将 expression 的属性设置为'val'.

RISCV 指令生成 backend/riscv/riscvasmemitter.py

由于赋值语句 mv t0, t1 # 我们使用 mv 指令来翻译中间表示里的 ASSIGN 指令因此做 Assign 的 RISCV 命令如下：

def visitAssign(self, instr: Assign) -> None:

self.seq.append(Riscv.Move(instr.dst, instr.src))

思考题

1. 我们假定当前栈帧的栈顶地址存储在 sp 寄存器中，请写出一段 **risc-v 汇编代码**，将栈帧空间扩大 16 字节。（提示 1：栈帧由高地址向低地址延伸；提示 2：risc-v 汇编中 addi reg0, reg1, <立即数> 表示将 reg1 的值加上立即数存储到 reg0 中。）

   可以通过修改栈顶指针（sp）的值来分配恰当的栈帧空间。只需要使栈顶指针下移：addi sp sp -16

2. 有些语言允许在同一个作用域中多次定义同名的变量，例如这是一段合法的 Rust 代码（你不需要精确了解它的含义，大致理解即可）：

```
fn main() {
  let a = 0;
```

```
  let a = f(a);
  let a = g(a);
}
```

其中 `f(a)`中的 `a` 是上一行的 `let a = 0;`定义的，`g(a)`中的 `a` 是上一行的 `let a = f(a);`。如果 MiniDecaf 也允许多次定义同名变量，并规定新的定义会覆盖之前的同名定义，请问在你的实现中，需要对定义变量和查找变量的逻辑做怎样的修改？（提示：如何区分一个作用域中**不同位置**的变量定义？）

定义变量的逻辑:

若发现已有定义，覆盖之前定义，而无需报错（当然实现中也未设置报错）。

查找变量的逻辑:

可以在符号表中将多个定义分别存储。visitDeclaration 为每个声明做一个版本号:

```
def visitDeclaration(self, decl: Declaration, ctx: Scope) -> None:
    symbol = ctx.lookup(decl.ident.value)
    new_version = 1
    if symbol is not None:
        new_version = symbol.version + 1
    new_var = VarSymbol(decl.ident.value, decl.var_t.type, new_version)
    ctx.declare(new_var)
    decl.symbol = new_var
```

而在使用 visitIdentifier 时，维护其遇到的最新（最大）版本号，作为当前的唯一有效声明。

```
def visitIdentifier(self, ident: Identifier, ctx: Scope) -> None:
    symbol = ctx.lookup(ident.value)
    if symbol is not None:
        # 维护当前版本号
        current_version = max(current_version, symbol.version)
        ident.symbol = symbol
    else:
        raise DecafUndefinedVarError(f'Variable "{ident.value}" is not defined')
```