STAGE-2

计13 占海川 2021050009

实验内容

语义分析

`frontend/typecheck/namer.py`
节点新增`Identifier/Assignment/Declaration`

- `Declaration`

```
def visitDeclaration(self, decl: Declaration, ctx: Scope) -> None:
    # 检查声明是否冲突
    if ctx.lookup(decl.ident.value) == None:
        # 新建符号并声明
        var = VarSymbol(decl.ident.value, decl.var_t.type)
        ctx.declare(var)
        # 将符号设为声明的属性
        decl.setattr("symbol", var)
        # 若声明有初值(赋值), 则调用accept访问该节点
        if decl.init_expr != NULL:
            decl.init_expr.accept(self, ctx)
    else:
        raise DecafDeclConflictError(str(decl.ident.value))
```

- `Assignment`

```
def visitAssignment(self, expr: Assignment, ctx: Scope) -> None:
    # 此处的实现与Binary相同, 由accept访问两操作数
    expr.lhs.accept(self, ctx)
    expr.rhs.accept(self, ctx)
```

- `Identifier`

```
def visitIdentifier(self, ident: Identifier, ctx: Scope) -> None:
    # 若当前变量未声明, 报错
    if ctx.lookup(ident.value) == None:
        raise DecafUndefinedVarError(str(ident.value))
    # 将Declaration中新建的符号设为标识符的属性
    ident.setattr("symbol", ctx.get(ident.value))
```

中间代码生成

`utils/tac/tacop.py`

- 在`TacBinaryOp`类中加入`Assign`操作符
  `ASSIGN = auto()`

`frontend/tacgen/tacgen.py`
节点新增`Identifier/Assignment/Declaration`

- `Declaration`

```
def visitDeclaration(self, decl: Declaration, mv: TACFuncEmitter) -> None:
    # 获取声明在语义分析阶段新建的符号，并为其分配一个寄存器
    var = decl.getattr("symbol")
    var.temp = mv.freshTemp()
    # 若声明有初值，访问该节点并添加一条赋值指令
    if decl.init_expr != NULL:
        decl.init_expr.accept(self, mv)
        mv.visitAssignment(var.temp, decl.init_expr.getattr("val"))
```

- `Assignment`

```
def visitAssignment(self, expr: Assignment, mv: TACFuncEmitter) -> None:
    # 访问右操作数(语句/标识符/立即数)，获取其val属性
    expr.rhs.accept(self, mv)
    r_temp = expr.rhs.getattr("val")
    # 获取左操作数(标识符)的寄存器
    l_temp = expr.lhs.getattr("symbol").temp
    # 添加一条赋值语句，并将表达式的val属性设为右操作数的val
    mv.visitAssignment(l_temp, r_temp)
    expr.setattr("val", r_temp)
```

- `Identifier`

```
def visitIdentifier(self, ident: Identifier, mv: TACFuncEmitter) -> None:
    # 将标识符的val属性设为其符号的寄存器
    var = ident.getattr("symbol")
    ident.setattr("val", var.temp)
```

目标代码生成

`backend/riscv/riscvasmemitter.py`
节点新增`Assign`

- 将赋值操作翻译为`mv`

```
def visitAssign(self, instr: Assign) -> None:
    self.seq.append(Riscv.Move(instr.dst, instr.src))
```

思考题

1. 将栈顶指针`sp`向低地址移动16字节

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

2.
   - 当语义检查发现重复声明变量`a`时, 可新增符号`a_temp`(不设为新声明的`symbol`属性). 若新声明有初值, 首先访问赋值语句的右操作数, 获取其`symbol/val`属性, 随后互换`a`与`a_temp`值, 将`a`与新声明绑定, 访问左操作数.

代码参考

参考了以下repo中的`getattr`与`setattr`方法
`https://github.com/Btlmd/minidecaf-compiler/tree/9ee8a4442e33f0f1fd6815bd1c450bd4e1b88a46`