刘雅迪

计26

学号：2021010521

step5: 局部变量和赋值

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。）

   答：

   ```
   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？）

   答：可以将相同变量名的定义变为一个栈。定义变量时，如果发现在一个作用域中该变量名已存在，则不报错，而是将新的定义压入栈中。查找变量时，返回栈顶的变量定义。

实验内容

主要修改内容涉及`frontend/typecheck/namer.py`文件中`Namer`类的`visitDeclaration`、`visitAssignment`和`visitIdentifier`函数，`frontend/tacgen/tacgen.py`文件中`TACGen`类的`visitDeclaration`、`visitAssignment`和`visitIdentifier`函数，以及在`backend/riscv/riscvasmemitter.py`文件的`RiscvInstrSelector`类中添加`visitAssign`函数。

1. frontend/typecheck/namer.py

这个文件里的三个函数基本按照注释写就不会有太大问题，唯一的问题是刚开始的时候不知道怎样设置变量的"symbol"属性，因为`tree.py`文件中的`Declaration`类和`Identifier`类都没有"symbol"属性。最后是询问了王伊荷同学，她告诉我分别用`setattr`函数和`getattr`函数设置和得到相关属性，我才能比较顺利地完成这部分内容。

`visitDeclaration`函数：这里的注释1写的不太清楚，需要lookup的是decl.indent的value，而不是decl.value。

```
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.
        """
        if ctx.lookup(decl.ident.value) is not None: raise
DecafUndefinedVarError(f"Variable {decl.ident.value} already declared")
        new_symbol = VarSymbol(decl.ident.value, decl.var_t)
        ctx.declare(new_symbol)
        # print(ctx.symbols)
        # print("decl:", decl)
        # print("decl.indent:", decl.ident)
        decl.setattr("symbol", new_symbol)
        # print(decl.ident.getattr("symbol"))
        if decl.init_expr is not None:
            # print("decl.init_expr:", decl.init_expr)
            decl.init_expr.accept(self, ctx)
        # raise NotImplementedError
```

`visitAssignment`函数：照搬`visitBinary`函数即可

```
def visitAssignment(self, expr: Assignment, ctx: Scope) -> None:
        """
        1. Refer to the implementation of visitBinary.
        """
        expr.lhs.accept(self, ctx)
        expr.rhs.accept(self, ctx)
        # raise NotImplementedError
```

`visitIdentifier`函数：

```
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.
        """
        symbol = ctx.lookup(ident.value)
        if symbol is None:
            raise DecafUndefinedVarError(ident.value)
        ident.setattr("symbol", symbol)
        # raise NotImplementedError
```

2. frontend/tacgen/tacgen.py

这里的三个函数我一开始搞不清楚怎样将symbol和temp联系起来，因为不清楚symbol的属性，结果原来很简单，直接symbol.temp即可。

写这里的三个函数的时候需要搞清楚decl和ident的"symbol"属性、"val"属性和temp之间的关系，不然很容易出错。因为涉及到寄存器的问题，所以左值和右值的属性处理不好很容易在tac代码中出现None，导致无法生成目的代码。比如在`visitAssignment`函数中左值的寄存器取"symbol"属性而右值的寄存器取"val"属性（至少在我的代码中是这样的），我一开始两者都取“val”属性，结果会有5个测例的tac码显示左值是None。

`visitDeclaration`函数：

```
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.
        """
        symbol = decl.getattr("symbol")
        new_temp = mv.freshTemp()
        symbol.temp = new_temp
        if decl.init_expr is not NULL:
            # print("new_temp:", new_temp)
            # print(decl.init_expr)
            # print(type(decl.init_expr))
            # print(decl)
            init_temp = decl.init_expr.accept(self, mv)
            mv.visitAssignment(new_temp, decl.init_expr.getattr("val")) #?

        # raise NotImplementedError
```

`visitAssignment`函数：

```
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)
        # breakpoint()
        lhs_symbol = expr.lhs.getattr("symbol").temp
        # print(lhs_symbol)
        rhs_symbol = expr.rhs.getattr("val")
        # print(lhs_symbol)
        rhs_temp = mv.visitAssignment(lhs_symbol, rhs_symbol)
        expr.setattr("val", rhs_temp)
        # raise NotImplementedError
```

`visitIdentifier`函数：

```
def visitIdentifier(self, ident: Identifier, mv: TACFuncEmitter) -> None:
        """
        1. Set the 'val' attribute of ident as the temp variable of the 'symbol' attribute
of ident.
        """
        symbol = ident.getattr("symbol")
        ident.setattr("val", symbol.temp)
        # print("symbol.temp:", symbol.temp)
        # raise NotImplementedError
```

3. backend/riscv/riscvasmemitter.py

在`RiscvInstrSelector`类中添加`visitAssign`函数：

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

调用Riscv类的Move函数，实现赋值运算的寄存器分配。

在这个类中增加这个函数的想法源自写完了从ast到tac的转化后出现的报错。

4. 其他

由于我写stage-2的时候使用实验指导上的例子进行测试的，而这个例子的type(decl.init_expr) == IntLiteral比较特殊，导致我的tac码中一直出现None。我一开始以为是因为`frontend/tacgen/tacgen.py`文件中的`visitDeclaration`函数需要分类讨论一下decl.init_expr的type，我分类讨论完后确实过的样例更多了，但是没有解决本质的左值寄存器一直是None的问题，后来才发现是"symbol"属性和"val"属性的选择出错。

而最后没有过的五个样例我以为是我backend部分代码的原因，经毛心怡同学和徐伊芃同学提醒后我才把debug的重心完全放到frontend代码上来，因为寄存器是frontend干的事。

此外，`decl.init_expr is None`和`decl.init_expr is NULL`也有区别。

Honor code

与王伊荷同学交流过stage-2的简要流程，但没有看过也没有参考过她的代码。

`frontend/tacgen/tacgen.py`的`visitAssignment`函数参考过徐伊芃同学的代码，用于debug。

此外，还与毛心怡同学交流过debug的问题，但没有借鉴她的代码。