Principles of Compilers
Experiment: Stage-4

Chenghua Liu*
liuch18@mails.tsinghua.edu.cn
Department of Computer Science
Tsinghua University

目录

*刘程华，学号 2018011687

1 实验内容及过程

1.1 step9

step9 开始，我们要支持多函数了。我们需要支持函数的声明和定义，我们还需要支持函数调用。语义检查部分，我们需要检查函数的重复定义、检查调用函数的实参（argment）和形参（parameter）的个数类型一致。我们不支持 void 返回值，直接忽略 int 返回值即可。

借助实验指导书，回顾整个框架流程，主要在以下部分进行修改：

- 词法

  增加逗号。

- 语法

  在 tree.py 中新增 Function/Parameter/Call 节点，分别用来表示函数定义、参数和函数调用，同时要在 visitor.py 中为这些节点提供 Visitor 的默认函数。在 ply_parse.py 中，实现相应的语法：允许多个函数、允许函数带有参数列表、允许函数调用。

- 语义

  Namer 和 Typer 里的 visitProgram 函数不再仅是访问 main 函数了，我们需要依次遍历所有函数。首先新建一个函数符号 FuncSymbol，Scope(ScopeKind.NORMAL) 函数作用域，这个作用域既包括函数体内部的变量，也包括函数的所有参数。在访问函数体之前，需要先扫描参数列表，新增 visitParameter 函数为所有参数建立符号，并存入符号表。最后，再访问函数体。对于函数调用，类似的，我们需要增加 visitCall 函数，并进行语义检查。

- TAC

  TAC 阶段，我们需要在 tac/tacinstr.py 中新增 PARAM 指令和 CALL 指令，用来设置函数调用参数和函数调用。在 funcvisitor.py 新增对应的 visitFunction、visitParameter 函数。为了支持函数的定义声明分离，我们还在 tac/tacinstr.py 定义了参数设置函数。

  在 TACGen 中，修改 transform 函数对每个函数体分别进行 TAC 生成，并新增 visitCall 函数，依次访问所有调用参数，并将它们的返回值（getattr('val')）依次执行 PARAM 指令，然后执行 CALL 指令，并新建临时变量为函数设置返回值。

- 目标代码

  目标代码阶段，我们需要在 riscv/riscvasmemitter.py 中的 RiscvInstrSelector 中新增 visitParam、visitCall 这两个方法用于保存函数调用的参数和调用函数。后者用于函数调用前后活跃寄存器的保存和恢复，函数调用以及对函数返回值的处理。我们还需要在 riscvasmemitter.py 中的 RiscvSubroutineEmitter 中实现把 Temp 保存到栈上和从栈上把 Temp 读出来以及实现 ra 寄存器的保存以及活跃寄存器的保存。（我的思路是把所有参数存栈

上。）在解析函数声明时，已经为各个参数符号设置了固定的 offset，因此寄存器分配算法可以在使用时自动读取。

1.2 step10

step10 我们要支持的是全局变量。全局变量和局部变量不同，它不是分配在栈上，而是放在某个固定地址，写在汇编的.bss 段或.data 段里。访问它也不能通过 fp 加偏移量，而是需要通过它的符号加载它的地址，通过它的地址访问它。

借助实验指导书，回顾整个框架流程，主要在以下部分进行修改：

- 词法

  无

- 语法

  实现全局变量时，无需新增 AST 节点，但需要修改 Program 节点，允许变量定义 Declaration 节点作为它的孩子。

- 语义

  修改 namer.py 中的 visitDeclaration 函数，变量定义时，判断当前作用域。如果处于全局作用域中，说明这是一个全局变量，我们将它的符号放入全局符号表中。全局变量只支持常量初始化，我们要对对初值进行语义检查。

- TAC

  需要在 utils/tac/tacinstr.py 中新增 LOAD/STORE/LOAD_ADDRESS 指令，表示设置读取/写入指定地址的内存量，并在同目录的 funcvisitor.py 等处新增对应的 visit 函数。我们在 VarSymbol 类中新增 isGlobal 和 addr，分别表示是否为全局变量和全局变量对应的地址。如果发现在使用全局变量时 addr 为空，则先通过 LOAD_ADDRESS 加载地址。还应注意在每个函数生成结束后，都应当清除各个全局变量的 addr 信息，因为 TAC Temp 不可以跨函数使用。

- 目标代码

  在 backend/riscv/riscvasmemitter.py 中的 RiscvAsmEmitter 的构造函数中处理全局变量的声明。在 riscvasmemitter.py 中的 RiscvInstrSelector 中进行判断，Temp 是否为全局变量。如果是，读取和赋值都要先执行 la 指令获取地址后通过 lw/sw 指令实现。

  还应遍历全局符号表，为全局变量生成.data 段汇编信息。因为我们规定全局 int 类型变量会默认初始化为 0，所以目前都放到.data 段中。

2 思考题

2.1 step9

1. MiniDecaf 的函数调用时参数求值的顺序是未定义行为。试写出一段 MiniDecaf 代码，使得不同的参数求值顺序会导致不同的返回结果。

   答：

```
int add(int a, int b){
    return  a+b;
}
int main(){
    int x = 0;
    return add(x, x=1)
}
```

   显然，如果从左向右依次传参，传递的参数为 0，1；反之，从右向左依次传参，传递的参数是 1，1。

2.2 step10

1. 写出 la v0, a 这一 RiscV 伪指令可能会被转换成哪些 RiscV 指令的组合（说出两种可能即可）。

   答：

   以下为两种可能的转换：

   (a) addi

   (b) lui

   也可能转换成上面两种的组合。