MiniDecaf Stage 3 Report

梁业升 2019010547（计 03）

2022 年 12 月 25 日

1 实验内容

对于 Step 7 的作用域与块，实验框架已经实现，在此不再赘述。

1.1 词法语法分析

While 语句增加 Do-while 支持：

```
  WhileStmt : WHILE LPAREN Expr RPAREN Stmt
-     { $$ = new ast::WhileStmt($3, $5, POS(@1)); }
+     { $$ = new ast::WhileStmt($3, $5, /*hasDo=*/false, POS(@1
    )); }
+ | DO Stmt WHILE LPAREN Expr RPAREN SEMICOLON
+     { $$ = new ast::WhileStmt($5, $2, /*hasDo=*/true, POS(@1)
    ); }
+ ;
```

相应地，`WhileStmt` 中增加一个成员 `bool hasDo`，表示是否是 Do-while 循环。

增加 For 循环：

```
+ ForStmt : FOR LPAREN Expr SEMICOLON Expr SEMICOLON Expr
    RPAREN Stmt
+     { $$ = new ast::ForStmt($3, $5, $7, $9, POS(@1)); }
+ | FOR LPAREN SEMICOLON Expr SEMICOLON Expr RPAREN Stmt
+     { $$ = new ast::ForStmt((ast::Expr*)nullptr, $4, $6, $8,
    POS(@1)); }
+ | FOR LPAREN Expr SEMICOLON SEMICOLON Expr RPAREN Stmt
+     { $$ = new ast::ForStmt($3, nullptr, $6, $8, POS(@1)); }
+ | FOR LPAREN Expr SEMICOLON Expr SEMICOLON RPAREN Stmt
+     { $$ = new ast::ForStmt($3, $5, nullptr, $8, POS(@1)); }
```

```
+ | FOR LPAREN SEMICOLON Expr SEMICOLON RPAREN Stmt
+     { $$ = new ast::ForStmt((ast::Expr*)nullptr, $4, nullptr,
     $7, POS(@1)); }
+ | FOR LPAREN Expr SEMICOLON SEMICOLON RPAREN Stmt
+     { $$ = new ast::ForStmt($3, nullptr, nullptr, $7, POS(@1)
    ); }
+ | FOR LPAREN SEMICOLON SEMICOLON Expr RPAREN Stmt
+     { $$ = new ast::ForStmt((ast::Expr*)nullptr, nullptr, $5,
     $7, POS(@1)); }
+ | FOR LPAREN SEMICOLON SEMICOLON RPAREN Stmt
+     { $$ = new ast::ForStmt((ast::Expr*)nullptr, nullptr,
    nullptr, $6, POS(@1)); }
+ | FOR LPAREN DeclStmt Expr SEMICOLON Expr RPAREN Stmt
+     { $$ = new ast::ForStmt($3, $4, $6, $8, POS(@1)); }
+ | FOR LPAREN DeclStmt Expr SEMICOLON RPAREN Stmt
+     { $$ = new ast::ForStmt($3, $4, nullptr, $7, POS(@1)); }
+ | FOR LPAREN DeclStmt SEMICOLON Expr RPAREN Stmt
+     { $$ = new ast::ForStmt($3, nullptr, $5, $7, POS(@1)); }
+ | FOR LPAREN DeclStmt SEMICOLON RPAREN Stmt
+     { $$ = new ast::ForStmt($3, nullptr, nullptr, $6, POS(@1)
    ); }
```

增加 Continue：

```
  Stmt : ReturnStmt {$$ = $1;} |
      ...
+     CONTINUE SEMICOLON
+       { $$ = new ast::ContStmt(POS(@1)); } |
      ...
```

1.2 符号表构建

在第一个 Pass，对于 `ForStmt` 节点，如在括号中声明变量，则为其创建一个本地作用域。需要注意的是此作用域在循环体内可见。

```
void SemPass1::visit(ast::ForStmt *s) {
    // Scope for the loop variable
    if (s->initDecl != nullptr) {
        Scope *declScope = new LocalScope();
        s->ATTR(decl_scope) = declScope;
        scopes->open(declScope);
        s->initDecl->accept(this);
    }
```

```
    if (s->cond)
        s->cond->accept(this);
    if (s->update)
        s->update->accept(this);
    s->body->accept(this);
    if (s->initDecl != nullptr)
        scopes->close();
}
```

WhileStmt 的符号表构建过程沿用框架给出的实现即可。

1.3 类型检查

针对新增的表达式 ForStmt 增加类型检查（其余新增的表达式在框架中已给出），需要注意括号内表达式为空的情形：

```
void SemPass2::visit(ast::ForStmt *s) {
    if (s->initExpr) {
        s->initExpr->accept(this);
        if (!s->initExpr->ATTR(type)->equal(BaseType::Int))
            issue(s->initExpr->getLocation(), new
    BadTestExprError());
    }
    if (s->initDecl) {
        scopes->open(s->ATTR(decl_scope));
        s->initDecl->accept(this);
    }
    if (s->cond) {
        s->cond->accept(this);
        if (!s->cond->ATTR(type)->equal(BaseType::Int))
            issue(s->cond->getLocation(), new BadTestExprError
    ());
    }
    if (s->update) {
        s->update->accept(this);
        if (!s->update->ATTR(type)->equal(BaseType::Int))
            issue(s->update->getLocation(), new
    BadTestExprError());
    }
    s->body->accept(this);
    if (s->initDecl)
        scopes->close();
}
```

1.4 翻译为中间代码

For 循环对应的三地址码如下：

```
    ... # 初始化语句/表达式
L1:
    ... # 条件判断
    JZERO xx, L3
    ... # 循环体语句
L2:
    ... # 更新语句
    JUMP L1
L3:
```

对应的翻译代码如下，注意需要将当前循环体内的 L3、L2 分别“入栈”break_label、continue_label：

```
void Translation::visit(ast::ForStmt *s) {
    Label l1 = tr->getNewLabel();
    Label l2 = tr->getNewLabel();
    Label l3 = tr->getNewLabel();

    // Visit init statement / expression
    if (s->initDecl)
        s->initDecl->accept(this);
    else if (s->initExpr)
        s->initExpr->accept(this);

    Label old_break = current_break_label;
    Label old_continue = current_continue_label;
    current_break_label = l3;
    current_continue_label = l2;

    tr->genMarkLabel(l1);

    if (s->cond) {
        s->cond->accept(this);
        tr->genJumpOnZero(l3, s->cond->ATTR(val));
    }
    if (s->body)
        s->body->accept(this);
    tr->genMarkLabel(l2);
    if (s->update)
        s->update->accept(this);
```

```
    tr->genJump(l1);
    tr->genMarkLabel(l3);

    current_break_label = old_break;
    current_continue_label = old_continue;
}
```

对于 `WhileStmt`，由于增加了 Do-while，需要根据 `s->hasDo` 进行相应的调整（即将条件判断的 TAC 置于不同位置）：

```
void Translation::visit(ast::WhileStmt *s) {
    Label L1 = tr->getNewLabel();
    Label L2 = tr->getNewLabel();

    Label old_break = current_break_label;
    Label old_continue = current_continue_label;
    current_break_label = L2;
    current_continue_label = L1;

    tr->genMarkLabel(L1);
    s->condition->accept(this);
    tr->genJumpOnZero(L2, s->condition->ATTR(val));
    if (!s->hasDo) {
        s->condition->accept(this);
        tr->genJumpOnZero(L2, s->condition->ATTR(val));
    }
    s->loop_body->accept(this);
    if (s->hasDo) {
        s->condition->accept(this);
        tr->genJumpOnZero(L2, s->condition->ATTR(val));
    }
    tr->genJump(L1);
    tr->genMarkLabel(L2);

    current_break_label = old_break;
    current_continue_label = old_continue;
}
```

2 思考题

1. **Step 7**:

由 Mind 生成的汇编代码经精简后如下：

```
---------------------- A
main:
    sw    ra, -4(sp)
    sw    fp, -8(sp)
    mv    fp, sp
    addi  sp, sp, -8
    li    t0, 2
    add   t1, zero, t0
    li    t2, 3
    slt   s1, t1, t2
    beqz  s1, __LL1
---------------------- B
    li    t0, 3
    add   t1, zero, t0
    mv    a0, t1
    mv    sp, fp
    lw    ra, -4(fp)
    lw    fp, -8(fp)
    ret
---------------------- C
__LL1:
    li    t0, 0
    mv    a0, t0
    mv    sp, fp
    lw    ra, -4(fp)
    lw    fp, -8(fp)
    ret
----------------------
```

控制流图如下：

```
A --> B
 \
  -> C
```

2. **Step 8**：第一种更好。假设执行一次循环体，循环体、条件判断均只有 1 条指令，则第一种执行 6 条指令，第二种执行 7 条指令。