

# STAGE2: 变量和语句 实验报告

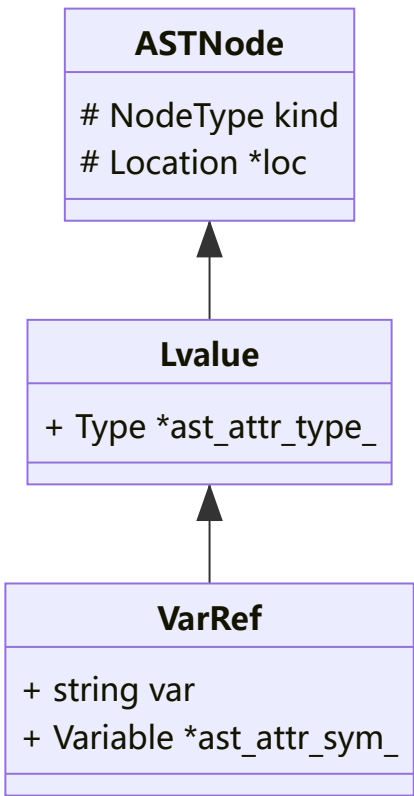
邢竞择 2020012890

## 1 Step 5

### 1.1 实验内容

支持变量的声明、读取、赋值

#### 1.1.1 框架分析



#### 1.1.2 前端

在 Expr 中添加

```
1 | IDENTIFIER
2 | { $$ = new ast::LvalueExpr(new ast::VarRef($1, POS(@1)), POS(@1)); }
3 | IDENTIFIER ASSIGN Expr
4 | { $$ = new ast::AssignExpr(new ast::VarRef($1, POS(@1)), $3, POS(@2)); }
  | }
```

新建 Decl，并添加

```

1 Decl      : Type IDENTIFIER SEMICOLON
2           { $$ = new ast::VarDecl($2, $1, POS(@2)); }
3           | Type IDENTIFIER ASSIGN Expr SEMICOLON
4           { $$ = new ast::VarDecl($2, $1, $4, POS(@2)); }

```

### 1.1.3 中端

在 SemPass1 的 visit 函数中，添加对变量定义冲突的检查，即：在 scope 中查找变量名，若结果非空则有冲突

```

1 void SemPass1::visit(ast::VarDecl *vdecl) {
2     Type *t = NULL;
3
4     vdecl->type->accept(this);
5     t = vdecl->type->ATTR(type);
6
7     Variable *v = new Variable(vdecl->name, t, vdecl->getLocation());
8     Symbol *sym = scopes->lookup(vdecl->name, vdecl->getLocation(), false);
9     if (sym != NULL)
10         issue(vdecl->getLocation(), new DeclConflictError(vdecl->name, sym));
11     else {
12         scopes->declare(v);
13         vdecl->ATTR(sym) = v;
14         if (vdecl->init != NULL)
15             vdecl->init->accept(this);
16     }
17 }

```

在 SemPass2 中，已经包含了从变量定义到诸表达式的类型推断

在 Translation 中，添加生成三地址码的 visit 函数

```

1 void Translation::visit(ast::AssignExpr *s) {
2     // TODO
3     s->left->accept(this);
4     s->e->accept(this);
5     tr->genAssign(((ast::VarRef *)s->left)->ATTR(sym)->getTemp(), s->e->ATTR(val));
6     s->ATTR(val) = ((ast::VarRef *)s->left)->ATTR(sym)->getTemp();
7 }
8
9 void Translation::visit(ast::LvalueExpr *e) {
10    // TODO
11    e->lvalue->accept(this);
12    e->ATTR(val) = ((ast::VarRef *)e->lvalue)->ATTR(sym)->getTemp();
13 }
14
15 void Translation::visit(ast::VarRef *ref) {
16     switch (ref->ATTR(lv_kind)) {

```

```

17     case ast::Lvalue::SIMPLE_VAR:
18         break;
19     default:
20         mind_assert(false);
21 }
22 }
23
24 void Translation::visit(ast::VarDecl *decl) {
25     decl->ATTR(sym)->attachTemp(tr->getNewTempI4());
26     // the `init` Expr is allowed to use the new-declared symbol
27     if (decl->init != NULL)
28         decl->init->accept(this);
29     if (decl->init != NULL)
30         tr->genAssign(decl->ATTR(sym)->getTemp(), decl->init->ATTR(val));
31 }

```

### 1.1.4 后端

对于 Tac::ASSIGN，添加相应的翻译函数

```

1 void RiscvDesc::emitTac(Tac *t) {
2     ...
3     case Tac::ASSIGN:
4         emitAssignTac(t);
5         break;
6     ...
7 }
8
9 void RiscvDesc::emitAssignTac(Tac *t) {
10     // eliminates useless assignments
11     if (!t->LiveOut->contains(t->op0.var))
12         return;
13     int r1 = getRegForRead(t->op1.var, 0, t->LiveOut);
14     int r0 = getRegForWrite(t->op0.var, r1, 0, t->LiveOut);
15     addInstr(RiscvInstr::MOVE, _reg[r0], _reg[r1], NULL, 0, EMPTY_STR, NULL);
16 }

```

## 1.2 思考题

### 1.2.1 1

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

### 1.2.2 2

- 支持在 scope 中修改一个变量名对应的 Symbol 类
- 在 SemPass2 中，在 VarDecl 节点动态地更新变量名对应的 Symbol

## 2 Step 6

### 2.1 实验内容

#### 2.1.1 中端

在类型检查中，要求 IfExpr 两侧的表达式类型相同

```
1 void SemPass2::visit(ast::IfExpr *s) {
2     s->condition->accept(this);
3     if (!s->condition->ATTR(type)->equal(BaseType::Int)) {
4         issue(s->condition->getLocation(), new BadTestExprError());
5     };
6 }
7 s->true_brch->accept(this);
8 s->>false_brch->accept(this);
9 if (!s->true_brch->ATTR(type)->equal(s->>false_brch->ATTR(type)))
10     issue(s->true_brch->getLocation(), new BadTestExprError());
11 s->ATTR(type) = s->true_brch->ATTR(type);
12 }
```

在三地址码翻译中添加访问 IfExpr 的函数

```
1 void Translation::visit(ast::IfExpr *e) {
2     Label L1 = tr->getNewLabel(); // entry of the false branch
3     Label L2 = tr->getNewLabel(); // exit
4     e->ATTR(val) = tr->getNewTempI4();
5
6     e->condition->accept(this);
7     tr->genJumpOnZero(L1, e->condition->ATTR(val));
8
9     e->true_brch->accept(this);
10    tr->genAssign(e->ATTR(val), e->true_brch->ATTR(val));
11    tr->genJump(L2);
12
13    tr->genMarkLabel(L1);
14    e->>false_brch->accept(this);
15    tr->genAssign(e->ATTR(val), e->>false_brch->ATTR(val));
16
17    tr->genMarkLabel(L2);
18 }
```

### 2.2 思考题

#### 2.2.1 1

根据实验指导书，bison 默认在 shift-reduce conflict 的时候选择shift，优先使用 If Else 匹配，从而对悬挂else进行就近匹配。

### 2.2.2 2

新建两个临时寄存器，将条件表达式的两支的计算结果分别存储在其中，最后再根据 `condition` 的结果，将其中一个寄存器的值赋给表达式左值。