王建楠 2019013296

Step 5

实验内容

在语义分析中新增了 DeclStmt（局部变量定义的解析），新增了 LvalueExpr（左值表达式的解析，包含有赋值表达式和变量引用）。

完善了在建立符号表的过程中，对局部变量的变量名合法性判断及声明。

完成了左值表达式、赋值表达式、变量定义的中间代码生成。

将中间代码的赋值表达式和变量定义的赋值部分，生成为目标代码的 `move` 操作。

思考题

1

```
addi sp, sp, -26
```

2

定义的时候无需判断是否出现重名，直接进行声明，如果有重复的名字，将其覆盖即可。

查找时，按照原来的方式查找。

Step 6

实验内容

新增了 `?:` 三目运算符的词法语法解析。

参照 `IfStmt` 完成了 `?:` 的中间代码生成。

`?:` 与 `IfStmt` 的不同之处在于 `?:` 有返回值，所以在实现上与 `IfStmt` 不大相同。

```
void Translation::visit(ast::IfExpr *e){
    Label L1 = tr->getNewLabel();
    Label L2 = tr->getNewLabel();
    e->condition->accept(this);
    Temp temp = tr->getNewTempI4();
    tr->genJumpOnZero(L1, e->condition->ATTR(val));

    e->true_brch->accept(this);
    tr->genAssign(temp, e->true_brch->ATTR(val));
    tr->genJump(L2);

    tr->genMarkLabel(L1);
    e->false_brch->accept(this);
```

```
    tr->genAssign(temp, e->false_brch->ATTR(val));

    tr->genMarkLabel(L2);
    e->ATTR(val) = temp;
}
```

在 `IfExpr` 中创建一个 `Temp`，然后通过分支跳转，在不同条件下，`Temp` 的取值为 false branch 或 true brach。

思考题

1

我的语言框架采用的是：shift the `"else"`，即 `else` 优先和最接近的没有匹配 `else` 的 `if` 匹配。

虽然代码中并未显式的处理悬吊 else 问题（以下为我的代码）

```
IfStmt      : IF LPAREN Expr RPAREN Stmt
                { ... }
            | IF LPAREN Expr RPAREN Stmt ELSE Stmt
                { ... }
            ;
```

这样的代码在 Bison 中会报 Warning `Shift/Reduce Conflicts`，但是 Bison 会默认使用 `Shift` 的方式。

> Since the parser prefers to shift the `"else"`, the result is to attach the else-clause to the innermost if-statement. —— 参见 Bison 文档

即 `else` 优先和最接近的没有匹配 `else` 的 `if` 匹配。

2

将 true branch 和 false branch 分别执行完后，其返回值分别存储在 Temp 中。然后再根据条件来判断该条件表达式的值应为哪个。