MiniDecaf Stage 1 Report

梁业升 2019010547（计 03）

2022 年 10 月 2 日

1 实验内容

1.1 词法语法分析

词法分析：在 scanner.l 中添加 BNOT、LNOT 等运算符的词法：

```
// scanner.l
"~"          { return yy::parser::make_BNOT(loc); }
"!"          { return yy::parser::make_LNOT(loc); }
"*"          { return yy::parser::make_TIMES(loc); }
"/"          { return yy::parser::make_SLASH(loc); }
"%"          { return yy::parser::make_MOD(loc); }
"=="         { return yy::parser::make_EQU(loc); }
"!="         { return yy::parser::make_NEQ(loc); }
"<"          { return yy::parser::make_LT(loc); }
"<="         { return yy::parser::make_LEQ(loc); }
">"          { return yy::parser::make_GT(loc); }
">="         { return yy::parser::make_GEQ(loc); }
"&&"         { return yy::parser::make_AND(loc); }
"||"         { return yy::parser::make_OR(loc); }
```

语法分析：在 parser.y 中，分别参照一元运算符 MINUS 和二元运算符 PLUS 的声明添加所实现运算符的语法：

```
// parser.y
Expr: ...
    /* Unary */
    | MINUS Expr    %prec NEG
        { $$ = new ast::NegExpr($2, POS(@1)); }
    | BNOT Expr     %prec BNOT
        { $$ = new ast::BitNotExpr($2, POS(@1)); }
    | LNOT Expr     %prec LNOT
```

```
        { $$ = new ast::NotExpr($2, POS(@1)); }
    /* Binary */
    | Expr PLUS Expr
        { $$ = new ast::AddExpr($1, $3, POS(@2)); }
    | Expr MINUS Expr
        { $$ = new ast::SubExpr($1, $3, POS(@2)); }
    | Expr TIMES Expr
        { $$ = new ast::MulExpr($1, $3, POS(@2)); }
    | Expr SLASH Expr
        { $$ = new ast::DivExpr($1, $3, POS(@2)); }
    | Expr MOD Expr
        { $$ = new ast::ModExpr($1, $3, POS(@2)); }
    | Expr EQU Expr
        { $$ = new ast::EquExpr($1, $3, POS(@2)); }
    | Expr NEQ Expr
        { $$ = new ast::NeqExpr($1, $3, POS(@2)); }
    | Expr LT Expr
        { $$ = new ast::LesExpr($1, $3, POS(@2)); }
    | Expr GT Expr
        { $$ = new ast::GrtExpr($1, $3, POS(@2)); }
    | Expr LEQ Expr
        { $$ = new ast::LeqExpr($1, $3, POS(@2)); }
    | Expr GEQ Expr
        { $$ = new ast::GeqExpr($1, $3, POS(@2)); }
    | Expr AND Expr
        { $$ = new ast::AndExpr($1, $3, POS(@2)); }
    | Expr OR Expr
        { $$ = new ast::OrExpr($1, $3, POS(@2)); }
```

顺便一提，框架给出的的三元运算符的语法 `Expr QUESTION Expr COLON Expr` 存在 shift/reduce 冲突:

```
frontend/parser.y: warning: shift/reduce conflict on token "+"
    [-Wcounterexamples]
  Example: Expr "?" Expr ":" Expr • "+" Expr
  Shift derivation
    Expr
      41: Expr "?" Expr ":" Expr
                              28: Expr • "+" Expr
  Reduce derivation
    Expr
      28: Expr                              "+" Expr
          41: Expr "?" Expr ":" Expr •
```

使用 %prec 显式指出此表达式的优先级即可解决：

```
Expr QUESTION Expr COLON Expr %prec QUESTION
```

2 中间代码生成

与上面类似，一元/二元运算符参考一元运算符 MINUS 和二元运算符 PLUS，在各个涉及的 Visitor（Translation 和 SemPass2）中添加对应的 visit 函数即可：

```
// Translation.cpp
/* Unary */
void Translation::visit(ast::BitNotExpr *e) {
    e->e->accept(this);
    e->ATTR(val) = tr->genBNot(e->e->ATTR(val));
}
...
/* Binary */
void Translation::visit(ast::SubExpr *e) {
    e->e1->accept(this);
    e->e2->accept(this);
    e->ATTR(val) = tr->genSub(e->e1->ATTR(val), e->e2->ATTR(val
    ));
}
...

// type_check.cpp
/* Unary */
void SemPass2::visit(ast::BitNotExpr *e) {
    e->e->accept(this);
    expect(e->e, BaseType::Int);
    e->ATTR(type) = BaseType::Int;
}
...
/* Binary */
void SemPass2::visit(ast::SubExpr *e) {
    e->e1->accept(this);
    expect(e->e1, BaseType::Int);
    e->e2->accept(this);
    expect(e->e2, BaseType::Int);
    e->ATTR(type) = BaseType::Int;
}
...
```

3 目标代码生成

部分中间代码有直接对应的的（伪）汇编代码，直接转换即可：

- `BNOT: not`
- `LNOT: seqz`
- `SUB: sub`
- `MUL: mul`
- `DIV: div`
- `MOD: rem`
- `LES: slt`
- `GTR: sgt`
- `LAND: and`
- `LOR: or`

其余中间代码没有对应的汇编，需要组合多条汇编代码实现：

- `EQU: sub + seqz`
- `NEQ: sub + snez`
- `LEQ: sgt + seqz`
- `GEQ: slt + seqz`

4 思考题

1. **Step 2**：`--2147483647`
2. **Step 3**：
   (a) **x86-64**：`floating point exception (core dumped)`
   (b) **riscv32**：`-2047483648`
3. **Step 4**：短路时第二个表达式无需计算，可以给程序编写提供便利，如第二个表达式依赖于第一个表达式所满足的某些条件。