编译原理期末作业展示

卢冠泽

题目: pandaC

```c
void visit_if_stmt(Node *node)
{
  /* FIXME: 实现如下代码生成片段，开始 */
  char *l_else = new_label();
  char *l_next = new_label();
  ExprResult cond = visit_expr(node->if_stmt.cond);

  if(node->if_stmt.else_stmt == NULL){
    emit_insn(OPC_BEQ, cond.value.reg, REG_ZERO, l_next);
    visit(node->if_stmt.then_stmt);
    emit_label(l_next);
  }else{
    emit_insn(OPC_BEQ, cond.value.reg, REG_ZERO, l_else);
    visit(node->if_stmt.then_stmt);
    emit_insn(OPC_J, l_next);
    emit_label(l_else);
    visit(node->if_stmt.else_stmt);
    emit_label(l_next);
  }

  free(l_else);
  free(l_next);
  free_reg(cond.value.reg);
  /* FIXME: 实现如上代码生产片段，结束 */
}
```

```c
Node *parse_for_statement()
{
#if DEBUG
  printf("%s\n", __func__);
#endif
  // fixme
  consume(TOK_FOR);
  consume('(');

  Node *init = NULL;
  if (token.id != ';') {
    init = parse_expression();
  }
  consume(';');

  Node *cond = NULL;
  if (token.id != ';') {
    cond = parse_expression();
  }
  consume(';');

  Node *incr = NULL;
  if (token.id != ')') {
    incr = parse_expression();
  }
  consume(')');

  Node *body = parse_statement();

  return create_for_stmt(init, cond, incr, body);
}
```

```
19  void init_scanner()
20  {
21      ENTRY e;
22      ENTRY *ep;
23      static char *keywords[] = {
24          "else", "for", "if", "int", "return", "void", "while", "constexpr"
25      };
26      static int token_id[] = {
27          TOK_ELSE, TOK_FOR, TOK_IF, TOK_INT, TOK_RETURN, TOK_VOID, TOK_WHILE, TOK_IFCONSTEXPR,
28      };
```

支持if constexpr表达式

```c
Node *parse_if_statement()
{
#if DEBUG
    printf("%s\n", __func__);
#endif

    consume(TOK_IF);
    bool is_ifconstexpr = try_to_consume(TOK_IFCONSTEXPR);
    if(!is_ifconstexpr){
        consume('(');
        Node *cond = parse_expression();
        consume(')');
        Node *then_stmt = parse_statement();
        Node *else_stmt = NULL;
        if (token.id == TOK_ELSE) {
            consume(TOK_ELSE);
            else_stmt = parse_statement();
        }
        return create_if_stmt(cond, then_stmt, else_stmt);
    }else{
        consume('(');
        Node *cond = parse_expression();
        consume(')');
        bool result = !!compile_calc(cond);
        Node *then_stmt = NULL;
        Node *else_stmt = NULL;
        if(result){
            then_stmt = parse_statement();
            if(try_to_consume(TOK_ELSE)){
                if(token.id == '{'){
                    while(token.id !='}')
                        consume(token.id);
                    consume('}');
                }else{
                    while(token.id !=';')
                        consume(token.id);
                    consume(';');
                }
            }
        }else{
            parse_statement();
            if(try_to_consume(TOK_ELSE)){
                then_stmt = parse_statement();
            }
        }
        free(cond);
        cond = NULL;
        return create_if_stmt(cond, then_stmt, else_stmt);
    }
}
```

```c
int do_unary_op(Node*, int);
int do_binary_op(Node*, int, int);
int compile_calc(Node* cond){
    switch(cond->kind){
        case BINARY_OP:{
            int left = compile_calc(cond->binary_op.left);
            int right = compile_calc(cond->binary_op.left);
            return do_binary_op(cond, left, right);
        }
        case UNARY_OP:{
            int val = compile_calc(cond->unary_op.operand);
            val = do_unary_op(cond, val);
            return val;
        }
        case CONST_EXPR:{
            return cond->const_expr.value;
        }
        default:{
            error("unimplemented kind of compile time calc");
            exit(-1);
        }
    }
}

int do_unary_op(Node* cond, int value){
    switch (cond->unary_op.op)
    {
    case OP_PLUS:
        return + value;
    case OP_MINUS:
        return - value;
    default:
        break;
    }
}
int do_binary_op(Node* cond, int left, int right){
    switch (cond->binary_op.op)
    {
    case OP_PLUS:
        return left + right;
    case OP_MINUS:
        return left - right;
    default:
        // OP_AND,
```

没支持const, 直接使用字面量

```
629  void visit_if_stmt(Node *node)
630  {
631      /* FIXME: 实现如下代码生成片段，开始 */
632      if(node->if_stmt.cond == NULL){
633          if(node->if_stmt.then_stmt != NULL){
634              visit(node->if_stmt.then_stmt);
635          }
636      }else{
637          char *l_else = new_label();
638          char *l_next = new_label();
639          ExprResult cond = visit_expr(node->if_stmt.cond);
640
641          if(node->if_stmt.else_stmt == NULL){
642              emit_insn(OPC_BEQ, cond.value.reg, REG_ZERO, l_next);
643              visit(node->if_stmt.then_stmt);
644              emit_label(l_next);
645          }else{
646              emit_insn(OPC_BEQ, cond.value.reg, REG_ZERO, l_else);
647              visit(node->if_stmt.then_stmt);
648              emit_insn(OPC_J, l_next);
649              emit_label(l_else);
650              visit(node->if_stmt.else_stmt);
651              emit_label(l_next);
652          }
653
654          free(l_else);
655          free(l_next);
656          free_reg(cond.value.reg);
657          /* FIXME: 实现如上代码生产片段，结束 */
658      }
```

只编译命中的分支

panda-c > if_constexpr.s

```
    .file   "test/if_constexpr.c"
    .text
    .globl  if_cond
    .type   if_cond, @function
if_cond:
    addi    sp, sp, -20
    sw  ra, 16(sp)
    sw  fp, 12(sp)
    addi    fp, sp, 20
    sw  a0, -20(fp)
    sw  a1, -16(fp)
    sw  a2, -12(fp)
    lw  t0, -16(fp)
    mv  a0, t0
    j   .Lif_cond_exit
.Lif_cond_exit:
    lw  ra, 16(sp)
    lw  fp, 12(sp)
    addi    sp, sp, 20
    ret
```

```
int if_cond(int x, int y, int z)
{
  if constexpr (1)
    return y;
  else
    whatever;
}
```

只编译命中的分支

```
121  int if_constexpr(int x, int y)
122  {
123      if constexpr(1+1-1==1)
124          return x+y;
125      else{
126          whatever ...
127      }
128  }
```

```
+2 = 2
-2 = -2
2 + 5 = 7
2 - 5 = -3
2 * 5 = 10
2 / 5 = 0
2 || 5 = 1
2 && 5 = 1
2 == 5 = 0
2 != 5 = 1
2 < 5 = 1
2 > 5 = 0
2 <= 5 = 1
2 >= 5 = 0
2 < 5 ? 2 : 5 = 2
2 > 5 ? 2 : 5 = 5
1 + ... + 2 = 3
1 + ... + 5 = 15
2 + 1 + 1 = 4
gcd(2, 5) = 1
if_constexpr(2, 5) = 7
```