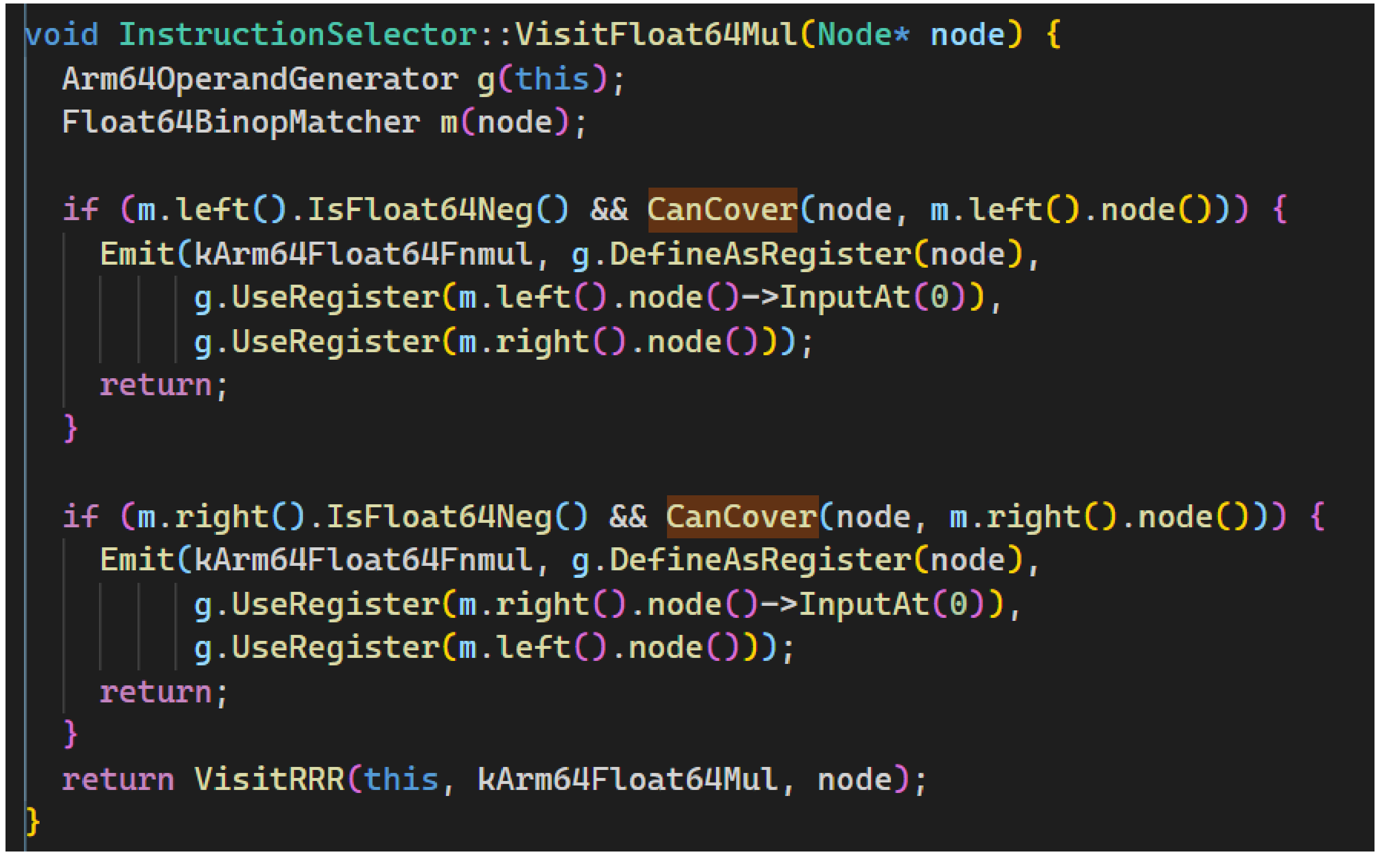
指令选择中的优化

软件所智能团研究中心PLCT实验室 陆亚涵

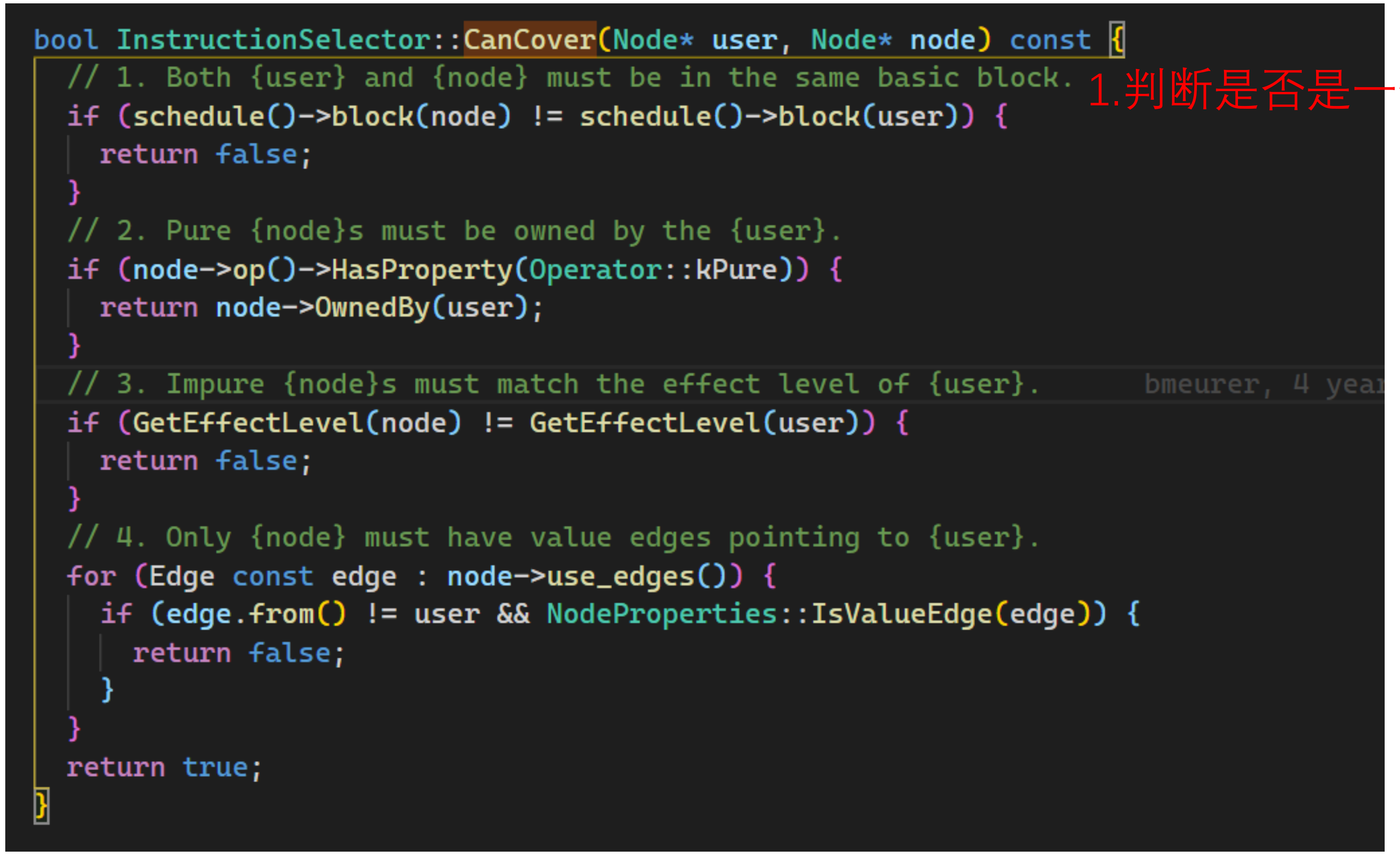
2020/5/26

指令选择中的优化



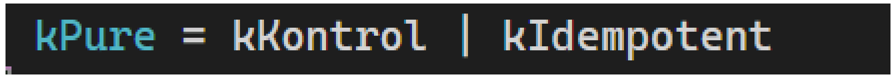
```
void InstructionSelector::VisitFloat64Mul(Node* node) {
  Arm64OperandGenerator g(this);
  Float64BinopMatcher m(node);

  if (m.left().IsFloat64Neg() && CanCover(node, m.left().node())) {
    Emit(kArm64Float64Fnmul, g.DefineAsRegister(node),
         g.UseRegister(m.left().node()->InputAt(0)),
         g.UseRegister(m.right().node()));
    return;
  }

  if (m.right().IsFloat64Neg() && CanCover(node, m.right().node())) {
    Emit(kArm64Float64Fnmul, g.DefineAsRegister(node),
         g.UseRegister(m.right().node()->InputAt(0)),
         g.UseRegister(m.left().node()));
    return;
  }
  return VisitRRR(this, kArm64Float64Mul, node);
}
```

指令选择中的优化



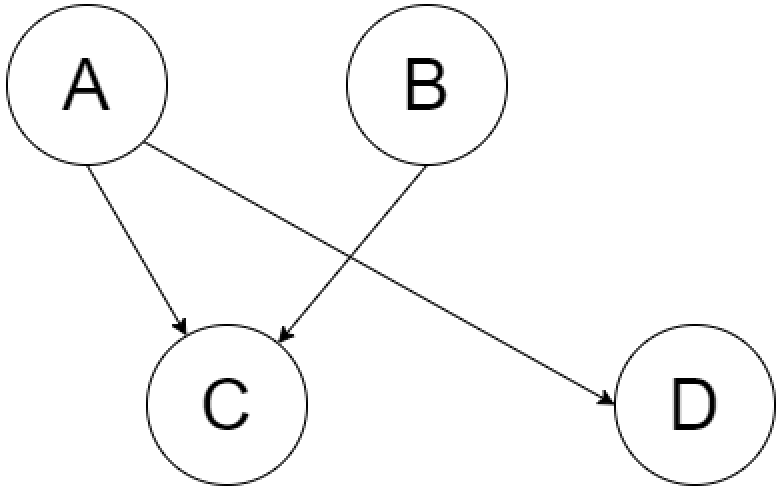
```
bool InstructionSelector::CanCover(Node* user, Node* node) const {
  // 1. Both {user} and {node} must be in the same basic block.
  if (schedule()->block(node) != schedule()->block(user)) {
    return false;
  }
  // 2. Pure {node}s must be owned by the {user}.
  if (node->op()->HasProperty(Operator::kPure)) {
    return node->OwnedBy(user);
  }
  // 3. Impure {node}s must match the effect level of {user}.        bmeurer, 4 year
  if (GetEffectLevel(node) != GetEffectLevel(user)) {
    return false;
  }
  // 4. Only {node} must have value edges pointing to {user}.
  for (Edge const edge : node->use_edges()) {
    if (edge.from() != user && NodeProperties::IsValueEdge(edge)) {
      return false;
    }
  }
  return true;
}
```

1.判断是否是一个代码块



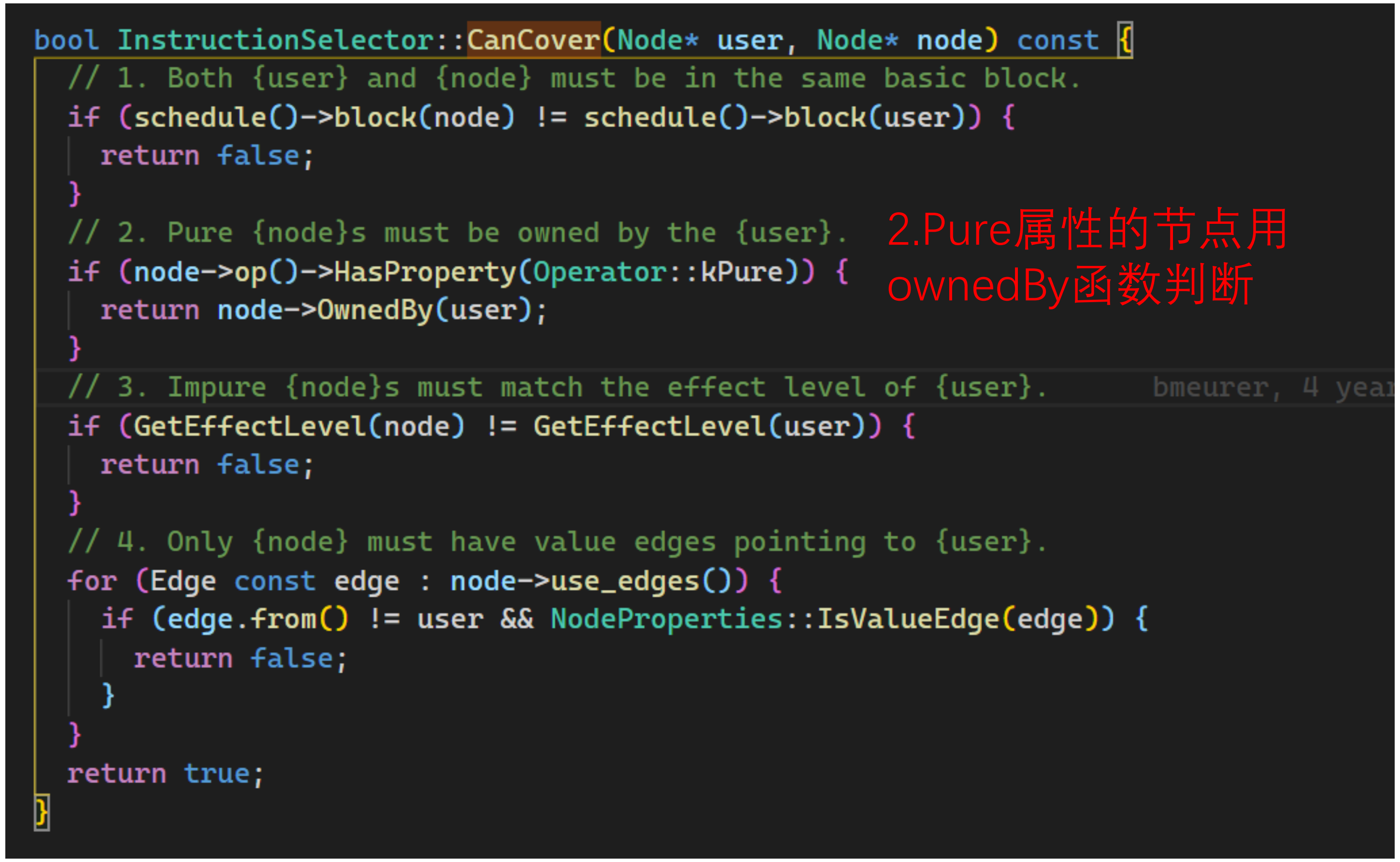
```
kPure = kKontrol | kIdempotent
```

幂等



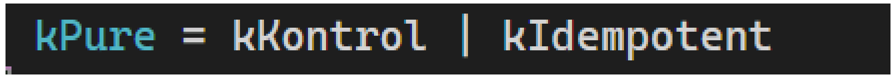
B、C结点有合并成一个节点的必要条件

指令选择中的优化

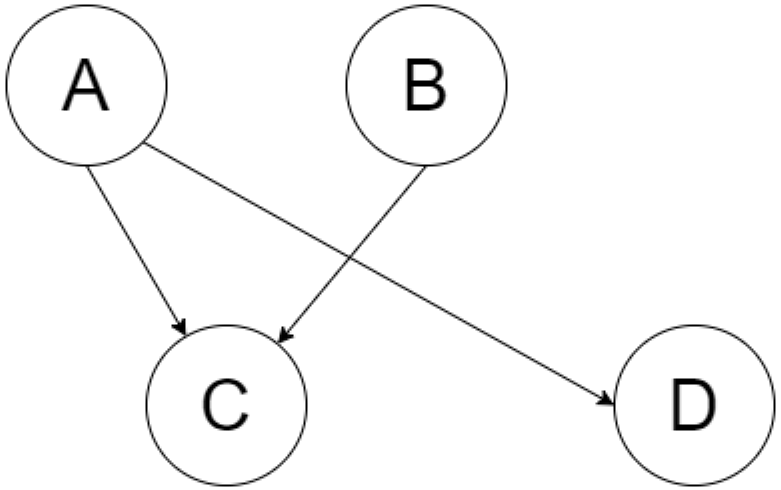


```
bool InstructionSelector::CanCover(Node* user, Node* node) const {
  // 1. Both {user} and {node} must be in the same basic block.
  if (schedule()->block(node) != schedule()->block(user)) {
    return false;
  }
  // 2. Pure {node}s must be owned by the {user}.
  if (node->op()->HasProperty(Operator::kPure)) {
    return node->OwnedBy(user);
  }
  // 3. Impure {node}s must match the effect level of {user}.      bmeurer, 4 year
  if (GetEffectLevel(node) != GetEffectLevel(user)) {
    return false;
  }
  // 4. Only {node} must have value edges pointing to {user}.
  for (Edge const edge : node->use_edges()) {
    if (edge.from() != user && NodeProperties::IsValueEdge(edge)) {
      return false;
    }
  }
  return true;
}
```

2.Pure属性的节点用ownedBy函数判断



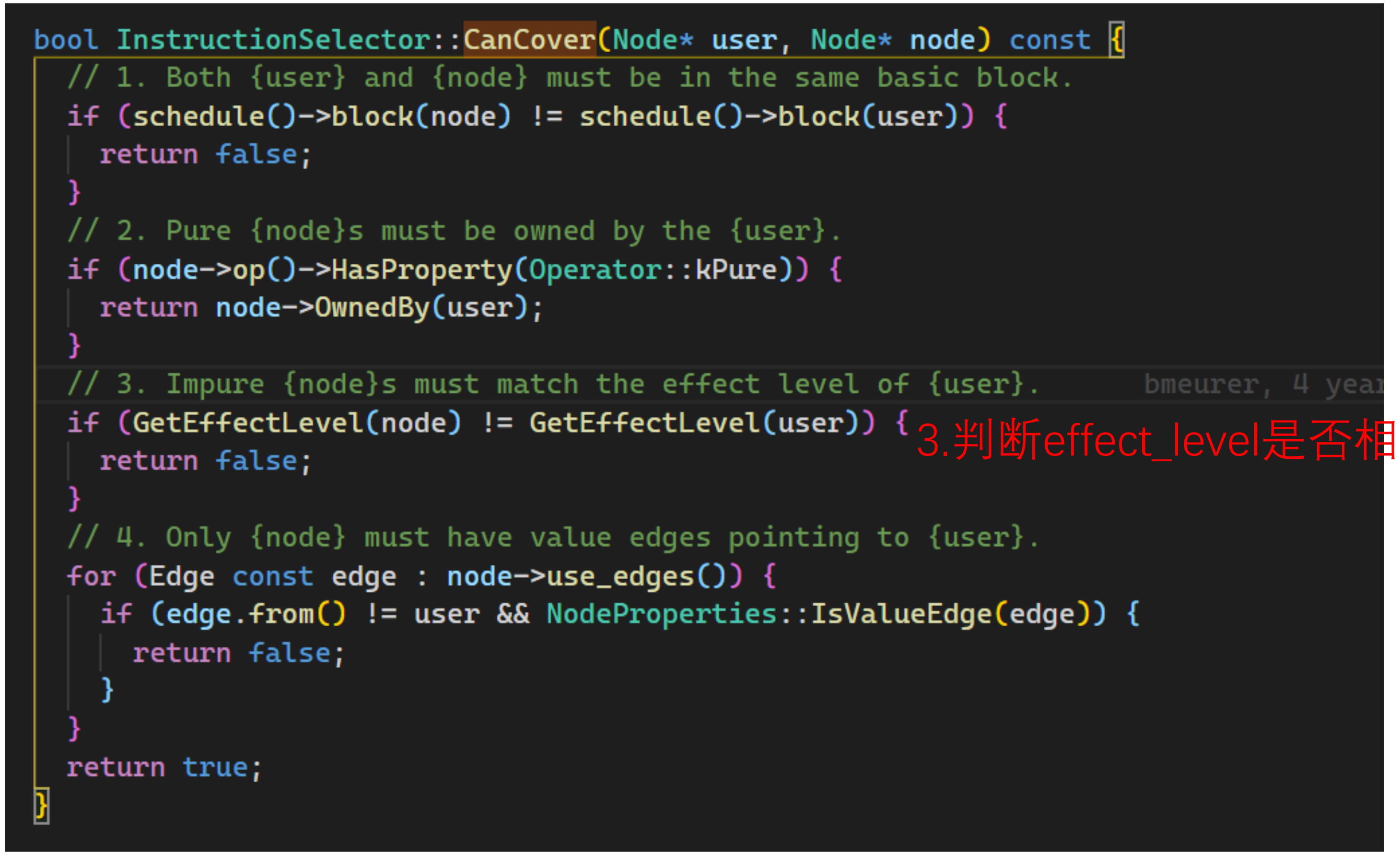
```
kPure = kKontrol | kIdempotent
```



$$\begin{tikzcd}
A \arrow[d] \arrow[drr] & B \arrow[dl] & \\
C & & D
\end{tikzcd}$$

B、C结点有合并成一个节点的必要条件

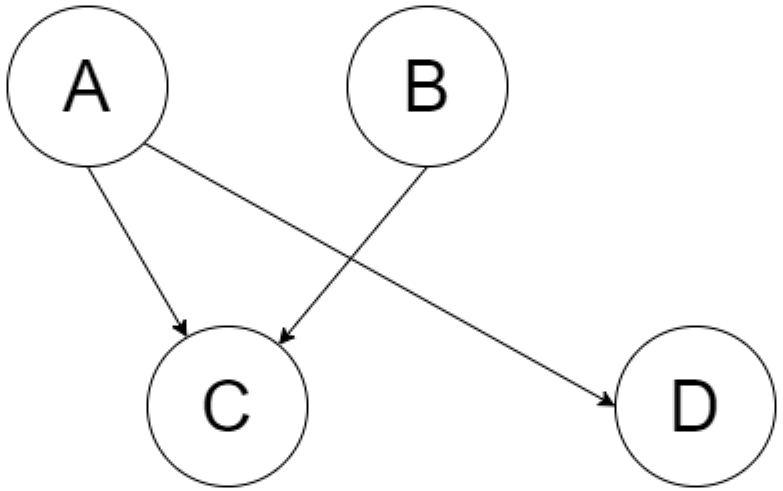
指令选择中的优化



```
bool InstructionSelector::CanCover(Node* user, Node* node) const {
  // 1. Both {user} and {node} must be in the same basic block.
  if (schedule()->block(node) != schedule()->block(user)) {
    return false;
  }
  // 2. Pure {node}s must be owned by the {user}.
  if (node->op()->HasProperty(Operator::kPure)) {
    return node->OwnedBy(user);
  }
  // 3. Impure {node}s must match the effect level of {user}.      bmeurer, 4 yea
  if (GetEffectLevel(node) != GetEffectLevel(user)) {
    return false;
  }
  // 4. Only {node} must have value edges pointing to {user}.
  for (Edge const edge : node->use_edges()) {
    if (edge.from() != user && NodeProperties::IsValueEdge(edge)) {
      return false;
    }
  }
  return true;
}
```

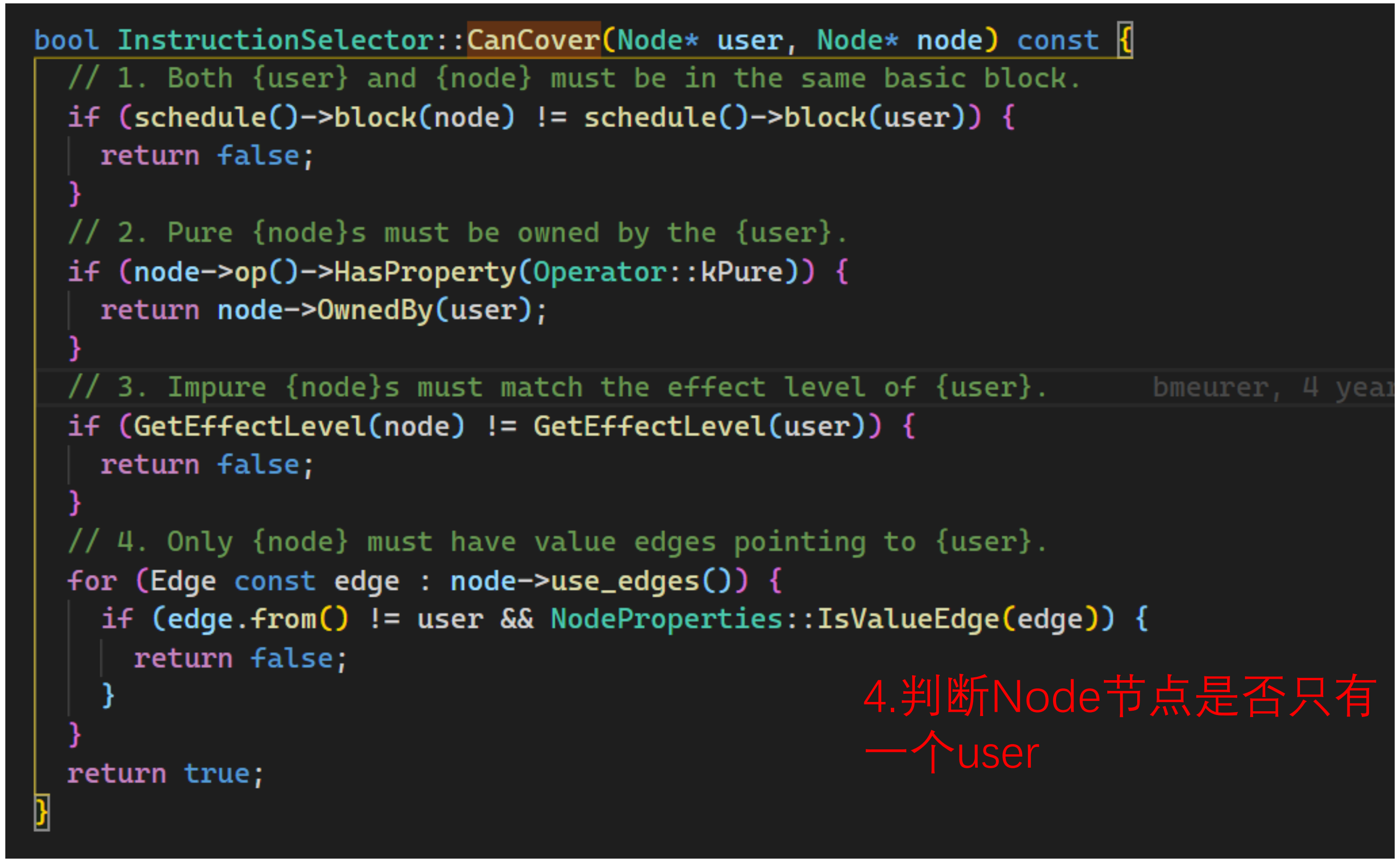
3.判断effect_level是否相同

```
kPure = kKontrol | kIdempotent
```



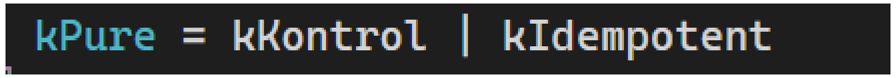
B、C结点有合并成一个节点的必要条件

指令选择中的优化

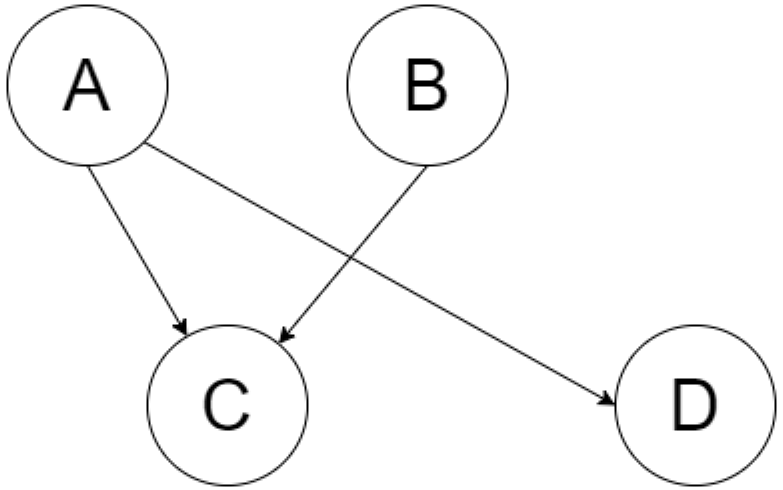


```
bool InstructionSelector::CanCover(Node* user, Node* node) const {
  // 1. Both {user} and {node} must be in the same basic block.
  if (schedule()->block(node) != schedule()->block(user)) {
    return false;
  }
  // 2. Pure {node}s must be owned by the {user}.
  if (node->op()->HasProperty(Operator::kPure)) {
    return node->OwnedBy(user);
  }
  // 3. Impure {node}s must match the effect level of {user}.      bmeurer, 4 year
  if (GetEffectLevel(node) != GetEffectLevel(user)) {
    return false;
  }
  // 4. Only {node} must have value edges pointing to {user}.
  for (Edge const edge : node->use_edges()) {
    if (edge.from() != user && NodeProperties::IsValueEdge(edge)) {
      return false;
    }
  }
  return true;
}
```

4.判断Node节点是否只有一个user

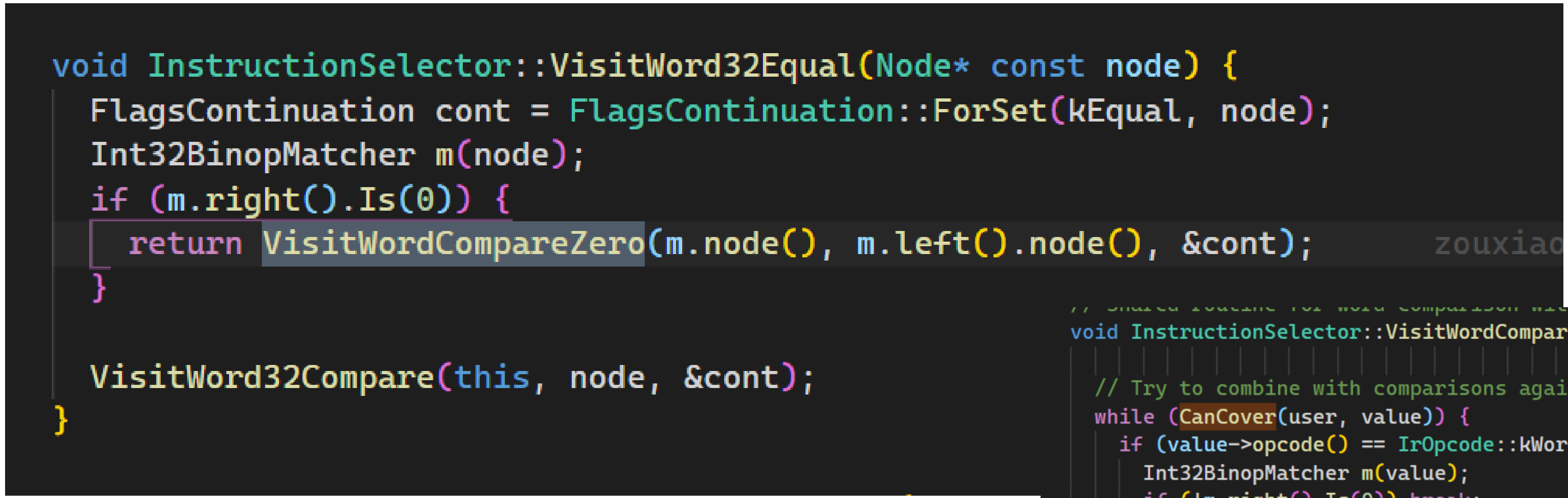


```
kPure = kKontrol | kIdempotent
```

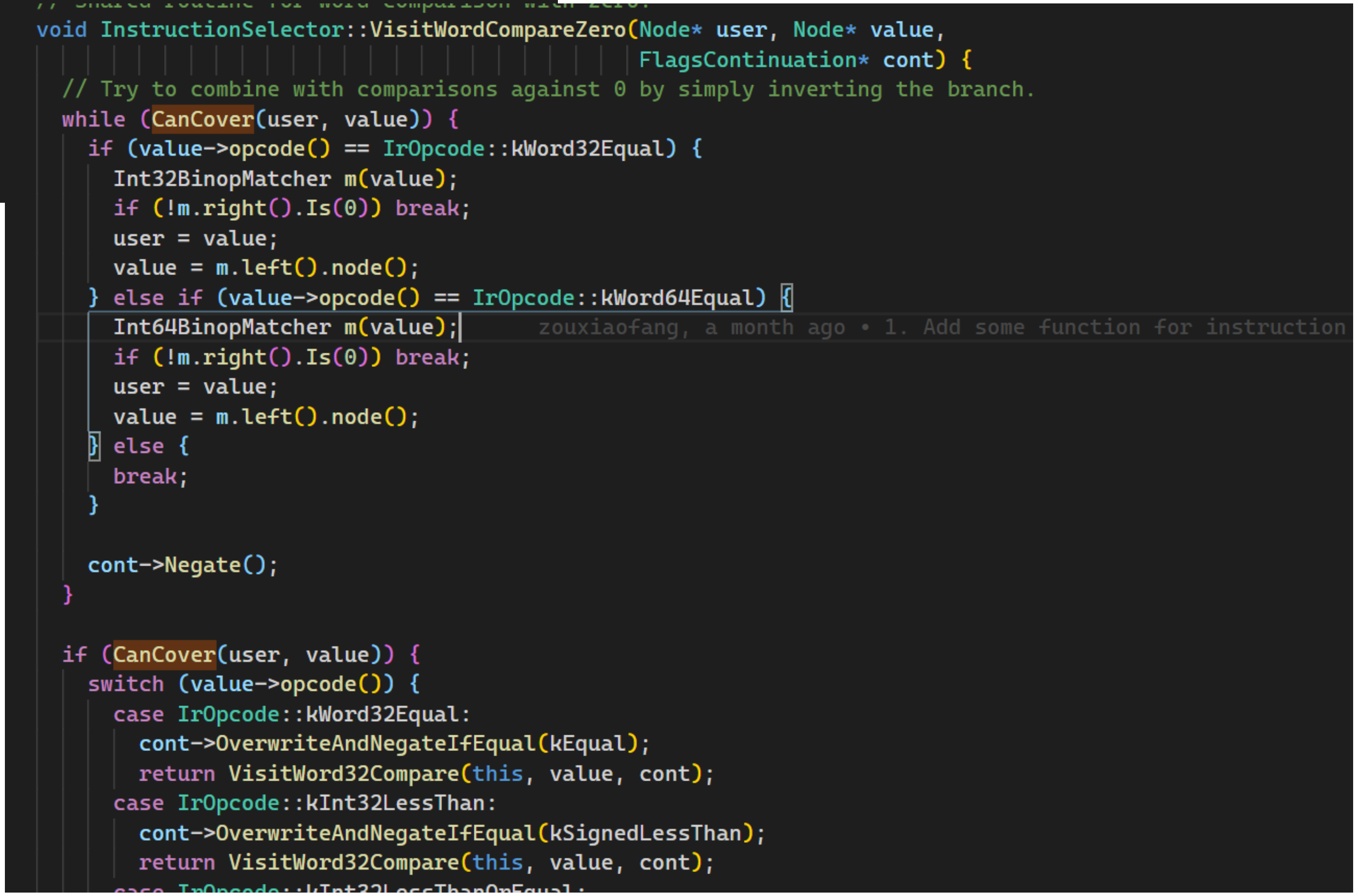


B、C结点有合并成一个节点的必要条件

指令选择中的优化

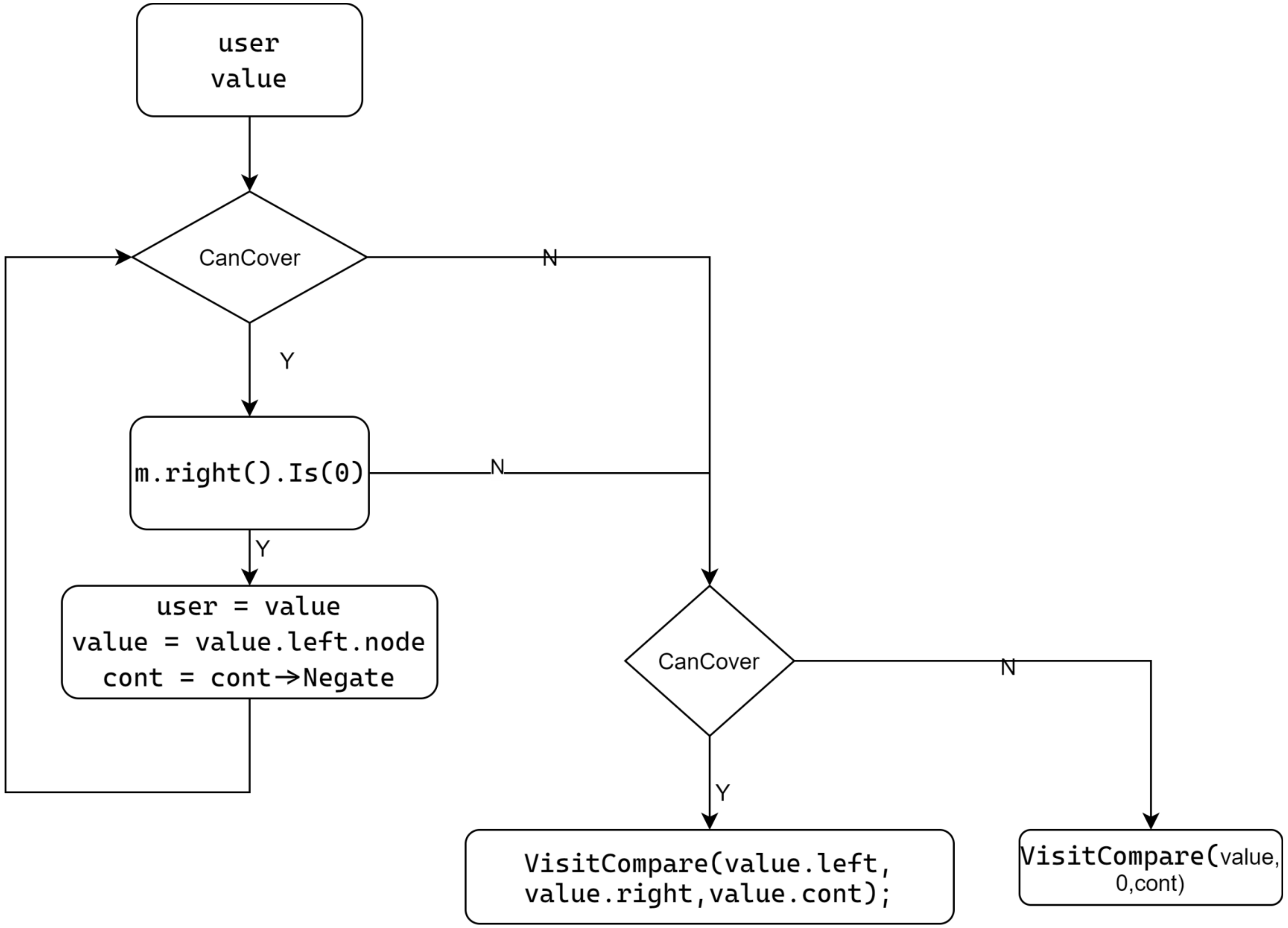


```
void InstructionSelector::VisitWord32Equal(Node* const node) {
  FlagsContinuation cont = FlagsContinuation::ForSet(kEqual, node);
  Int32BinopMatcher m(node);
  if (m.right().Is(0)) {
    return VisitWordCompareZero(m.node(), m.left().node(), &cont);      zouxiao
  }

  VisitWord32Compare(this, node, &cont);
}
```



```
// Shared routine for word comparison with zero.
void InstructionSelector::VisitWordCompareZero(Node* user, Node* value,
                                               FlagsContinuation* cont) {
  // Try to combine with comparisons against 0 by simply inverting the branch.
  while (CanCover(user, value)) {
    if (value->opcode() == IrOpcode::kWord32Equal) {
      Int32BinopMatcher m(value);
      if (!m.right().Is(0)) break;
      user = value;
      value = m.left().node();
    } else if (value->opcode() == IrOpcode::kWord64Equal) {
      Int64BinopMatcher m(value);      zouxiaofang, a month ago • 1. Add some function for instruction
      if (!m.right().Is(0)) break;
      user = value;
      value = m.left().node();
    } else {
      break;
    }

    cont->Negate();
  }

  if (CanCover(user, value)) {
    switch (value->opcode()) {
      case IrOpcode::kWord32Equal:
        cont->OverwriteAndNegateIfEqual(kEqual);
        return VisitWord32Compare(this, value, cont);
      case IrOpcode::kInt32LessThan:
        cont->OverwriteAndNegateIfEqual(kSignedLessThan);
        return VisitWord32Compare(this, value, cont);
      case IrOpcode::kInt32LessThanOrEqual:
```

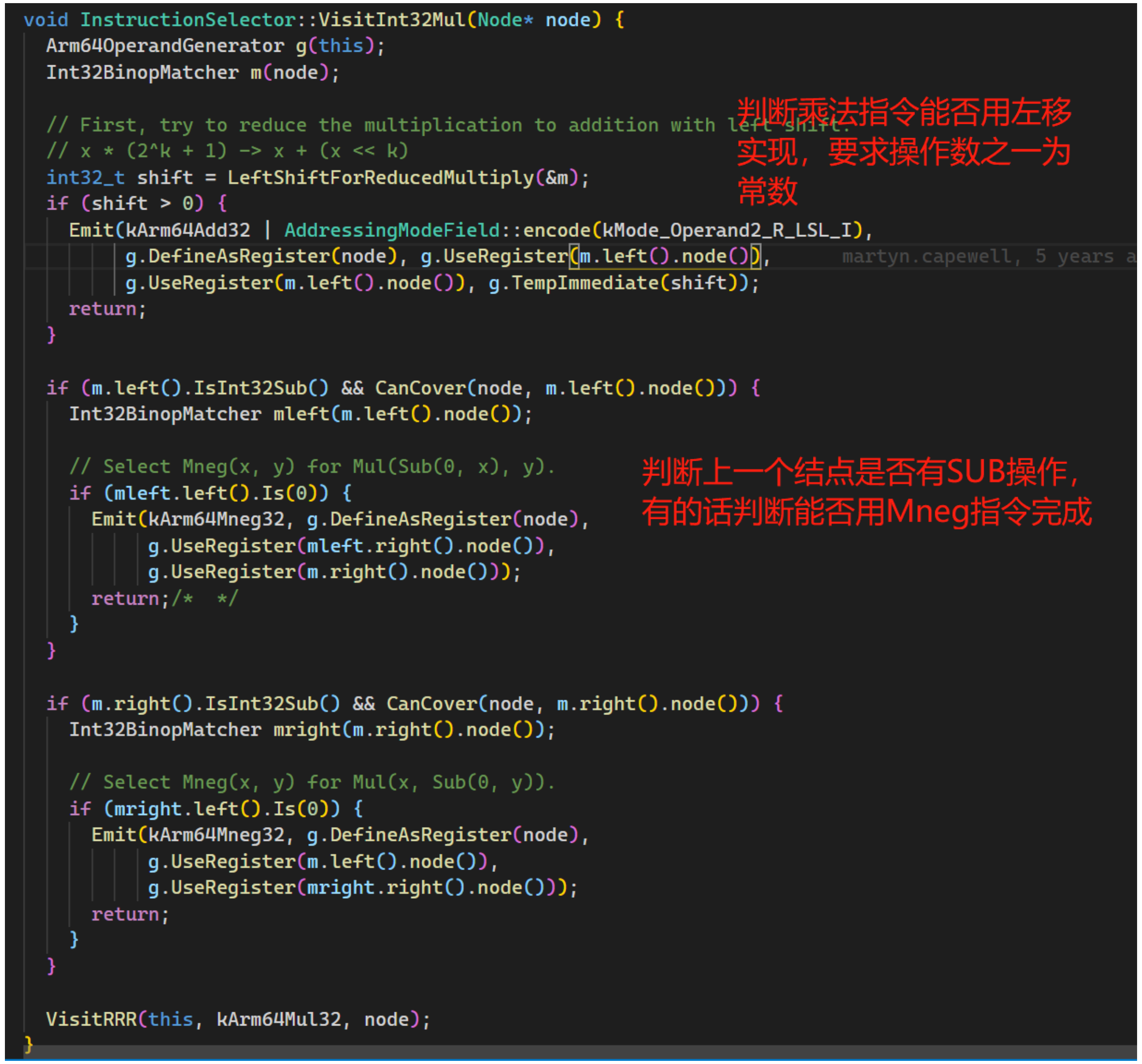
指令选择中的优化


user
value
CanCover
N
Y
m.right().Is(0)
N
Y
user = value
value = value.left.node
cont = cont→Negate
CanCover
N
Y
VisitCompare(value.left,
value.right,value.cont);
VisitCompare(value,
0,cont)

指令选择中的优化

ISCAS 中国科学院软件研究所 Institute of Software Chinese Academy of Sciences

智能软件研究中心 Intelligent Software Research Center



```
void InstructionSelector::VisitInt32Mul(Node* node) {
  Arm64OperandGenerator g(this);
  Int32BinopMatcher m(node);

  // First, try to reduce the multiplication to addition with left shift.
  // x * (2^k + 1) -> x + (x << k)
  int32_t shift = LeftShiftForReducedMultiply(&m);
  if (shift > 0) {
    Emit(kArm64Add32 | AddressingModeField::encode(kMode_Operand2_R_LSL_I),
         g.DefineAsRegister(node), g.UseRegister(m.left().node()),       martyn.capewell, 5 years a
         g.UseRegister(m.left().node()), g.TempImmediate(shift));
    return;
  }

  if (m.left().IsInt32Sub() && CanCover(node, m.left().node())) {
    Int32BinopMatcher mleft(m.left().node());

    // Select Mneg(x, y) for Mul(Sub(0, x), y).
    if (mleft.left().Is(0)) {
      Emit(kArm64Mneg32, g.DefineAsRegister(node),
           g.UseRegister(mleft.right().node()),
           g.UseRegister(m.right().node()));
      return;/*  */
    }
  }

  if (m.right().IsInt32Sub() && CanCover(node, m.right().node())) {
    Int32BinopMatcher mright(m.right().node());

    // Select Mneg(x, y) for Mul(x, Sub(0, y)).
    if (mright.left().Is(0)) {
      Emit(kArm64Mneg32, g.DefineAsRegister(node),
           g.UseRegister(m.left().node()),
           g.UseRegister(mright.right().node()));
      return;
    }
  }

  VisitRRR(this, kArm64Mul32, node);
}
```

判断乘法指令能否用左移实现，要求操作数之一为常数

判断上一个结点是否有SUB操作，有的话判断能否用Mneg指令完成

谢 谢

欢迎交流合作

2019/02/25