Improving optimized-code line table quality

Orlando Cazalet-Hyams (speaker)

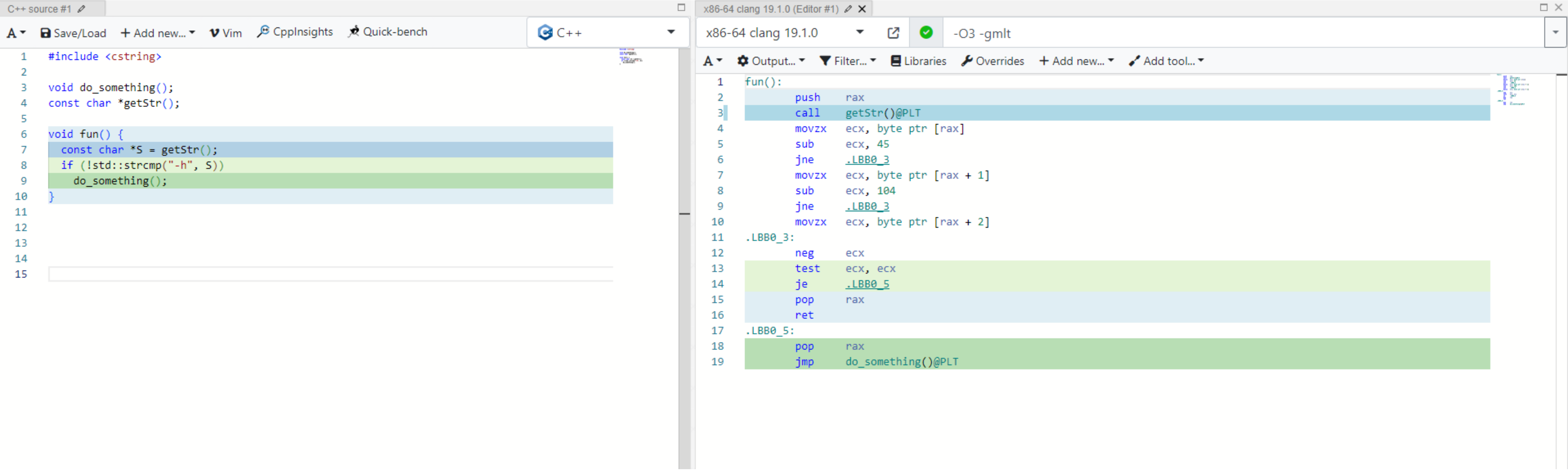
Stephen Livermore-Tozer

Jeremy Morse

Outline

- LLVM source location **defect finder** [almost done]
- Improved is_stmt placement for **better interactive debugging** [WIP]

Basics - Line table what and why



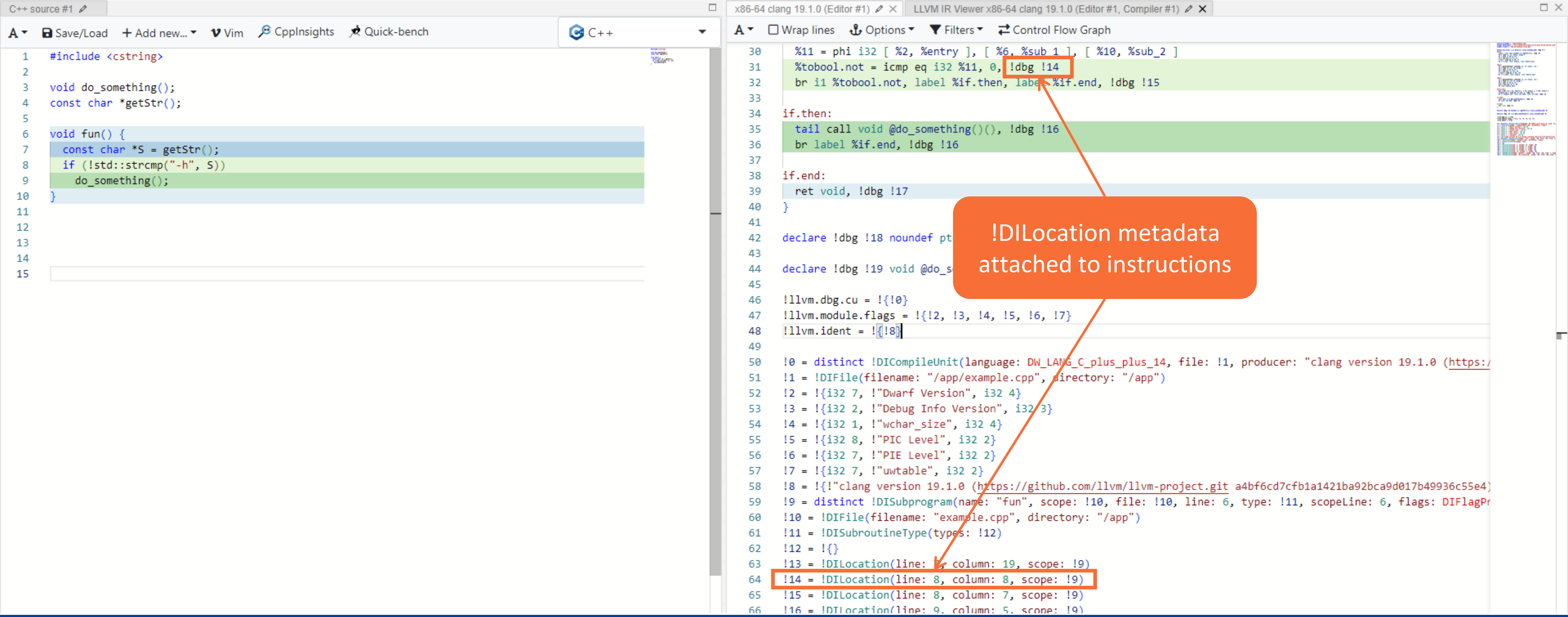
x86-64 clang 19.1.0

-O3 -gmlt

```cpp
#include <cstring>

void do_something();
const char *getStr();

void fun() {
  const char *S = getStr();
  if (!std::strcmp("-h", S))
    do_something();
}
```

```asm
fun():
        push    rax
        call    getStr()@PLT
        movzx   ecx, byte ptr [rax]
        sub     ecx, 45
        jne     .LBB0_3
        movzx   ecx, byte ptr [rax + 1]
        sub     ecx, 104
        jne     .LBB0_3
        movzx   ecx, byte ptr [rax + 2]
.LBB0_3:
        neg     ecx
        test    ecx, ecx
        je      .LBB0_5
        pop     rax
        ret
.LBB0_5:
        pop     rax
        jmp     do_something()@PLT
```

Basics – LLVM IR

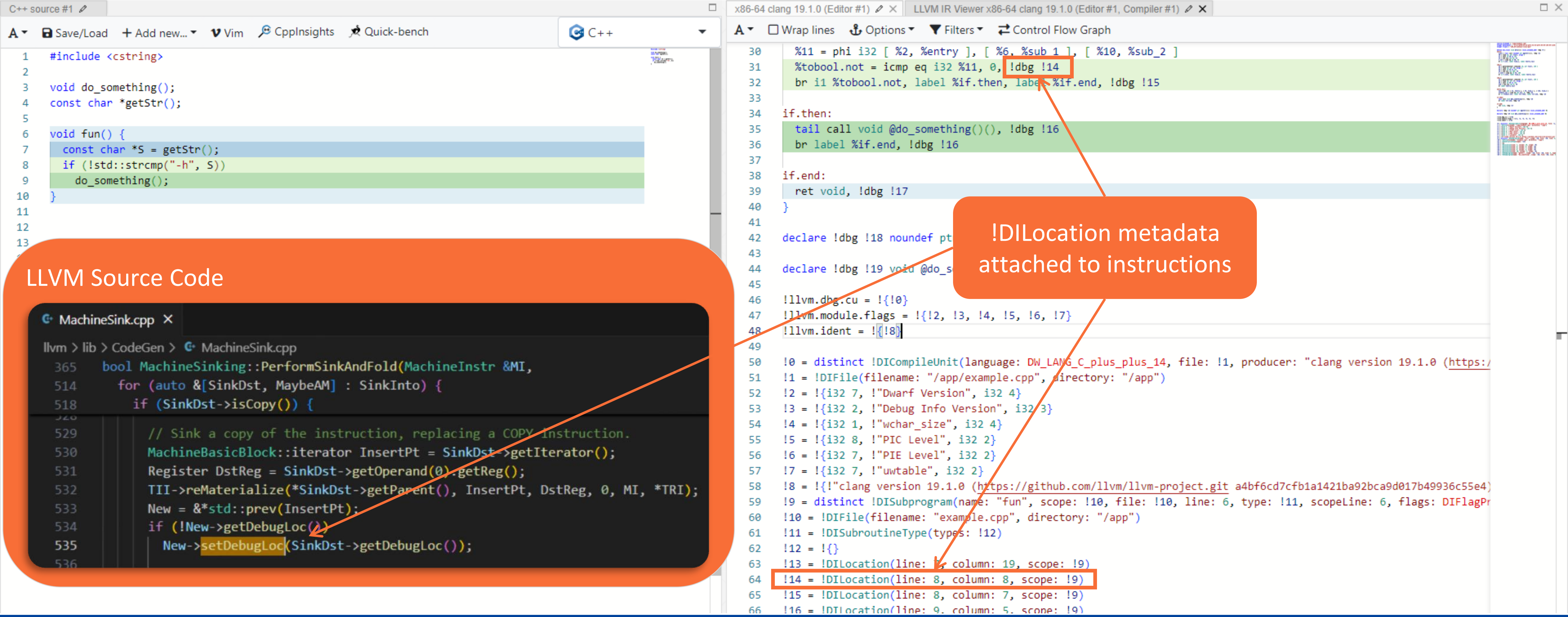


```cpp
#include <cstring>

void do_something();
const char *getStr();

void fun() {
  const char *S = getStr();
  if (!std::strcmp("-h", S))
    do_something();
}
```

```llvm
  %11 = phi i32 [ %2, %entry ], [ %6, %sub_1 ], [ %10, %sub_2 ]
  %tobool.not = icmp eq i32 %11, 0, !dbg !14
  br i1 %tobool.not, label %if.then, label %if.end, !dbg !15

if.then:
  tail call void @do_something()(), !dbg !16
  br label %if.end, !dbg !16

if.end:
  ret void, !dbg !17
}

declare !dbg !18 noundef pt

declare !dbg !19 void @do_s

!llvm.dbg.cu = !{!0}
!llvm.module.flags = !{!2, !3, !4, !5, !6, !7}
!llvm.ident = !{!8}

!0 = distinct !DICompileUnit(language: DW_LANG_C_plus_plus_14, file: !1, producer: "clang version 19.1.0 (https:/
!1 = !DIFile(filename: "/app/example.cpp", directory: "/app")
!2 = !{i32 7, !"Dwarf Version", i32 4}
!3 = !{i32 2, !"Debug Info Version", i32 3}
!4 = !{i32 1, !"wchar_size", i32 4}
!5 = !{i32 8, !"PIC Level", i32 2}
!6 = !{i32 7, !"PIE Level", i32 2}
!7 = !{i32 7, !"uwtable", i32 2}
!8 = !{!"clang version 19.1.0 (https://github.com/llvm/llvm-project.git a4bf6cd7cfb1a1421ba92bca9d017b49936c55e4)
!9 = distinct !DISubprogram(name: "fun", scope: !10, file: !10, line: 6, type: !11, scopeLine: 6, flags: DIFlagPr
!10 = !DIFile(filename: "example.cpp", directory: "/app")
!11 = !DISubroutineType(types: !12)
!12 = !{}
!13 = !DILocation(line: 7, column: 19, scope: !9)
!14 = !DILocation(line: 8, column: 8, scope: !9)
!15 = !DILocation(line: 8, column: 7, scope: !9)
!16 = !DILocation(line: 9, column: 5, scope: !9)
```

!DILocation metadata attached to instructions

Basics – LLVM IR


C++ source #1
A Save/Load + Add new... Vim CppInsights Quick-bench
C++
x86-64 clang 19.1.0 (Editor #1)
LLVM IR Viewer x86-64 clang 19.1.0 (Editor #1, Compiler #1)
A □ Wrap lines Options Filters Control Flow Graph
MachineSink.cpp
llvm > lib > CodeGen > MachineSink.cpp

```cpp
#include <cstring>

void do_something();
const char *getStr();

void fun() {
  const char *S = getStr();
  if (!std::strcmp("-h", S))
    do_something();
}
```

```llvm
  %11 = phi i32 [ %2, %entry ], [ %6, %sub_1 ], [ %10, %sub_2 ]
  %tobool.not = icmp eq i32 %11, 0, !dbg !14
  br i1 %tobool.not, label %if.then, label %if.end, !dbg !15

if.then:
  tail call void @do_something()(), !dbg !16
  br label %if.end, !dbg !16

if.end:
  ret void, !dbg !17
}

declare !dbg !18 noundef pt

declare !dbg !19 void @do_s

!llvm.dbg.cu = !{!0}
!llvm.module.flags = !{!2, !3, !4, !5, !6, !7}
!llvm.ident = !{!8}

!0 = distinct !DICompileUnit(language: DW_LANG_C_plus_plus_14, file: !1, producer: "clang version 19.1.0 (https:/
!1 = !DIFile(filename: "/app/example.cpp", directory: "/app")
!2 = !{i32 7, !"Dwarf Version", i32 4}
!3 = !{i32 2, !"Debug Info Version", i32 3}
!4 = !{i32 1, !"wchar_size", i32 4}
!5 = !{i32 8, !"PIC Level", i32 2}
!6 = !{i32 7, !"PIE Level", i32 2}
!7 = !{i32 7, !"uwtable", i32 2}
!8 = !{!"clang version 19.1.0 (https://github.com/llvm/llvm-project.git a4bf6cd7cfb1a1421ba92bca9d017b49936c55e4)
!9 = distinct !DISubprogram(name: "fun", scope: !10, file: !10, line: 6, type: !11, scopeLine: 6, flags: DIFlagPr
!10 = !DIFile(filename: "example.cpp", directory: "/app")
!11 = !DISubroutineType(types: !12)
!12 = !{}
!13 = !DILocation(line: 7, column: 19, scope: !9)
!14 = !DILocation(line: 8, column: 8, scope: !9)
!15 = !DILocation(line: 8, column: 7, scope: !9)
!16 = !DILocation(line: 9, column: 5, scope: !9)
```

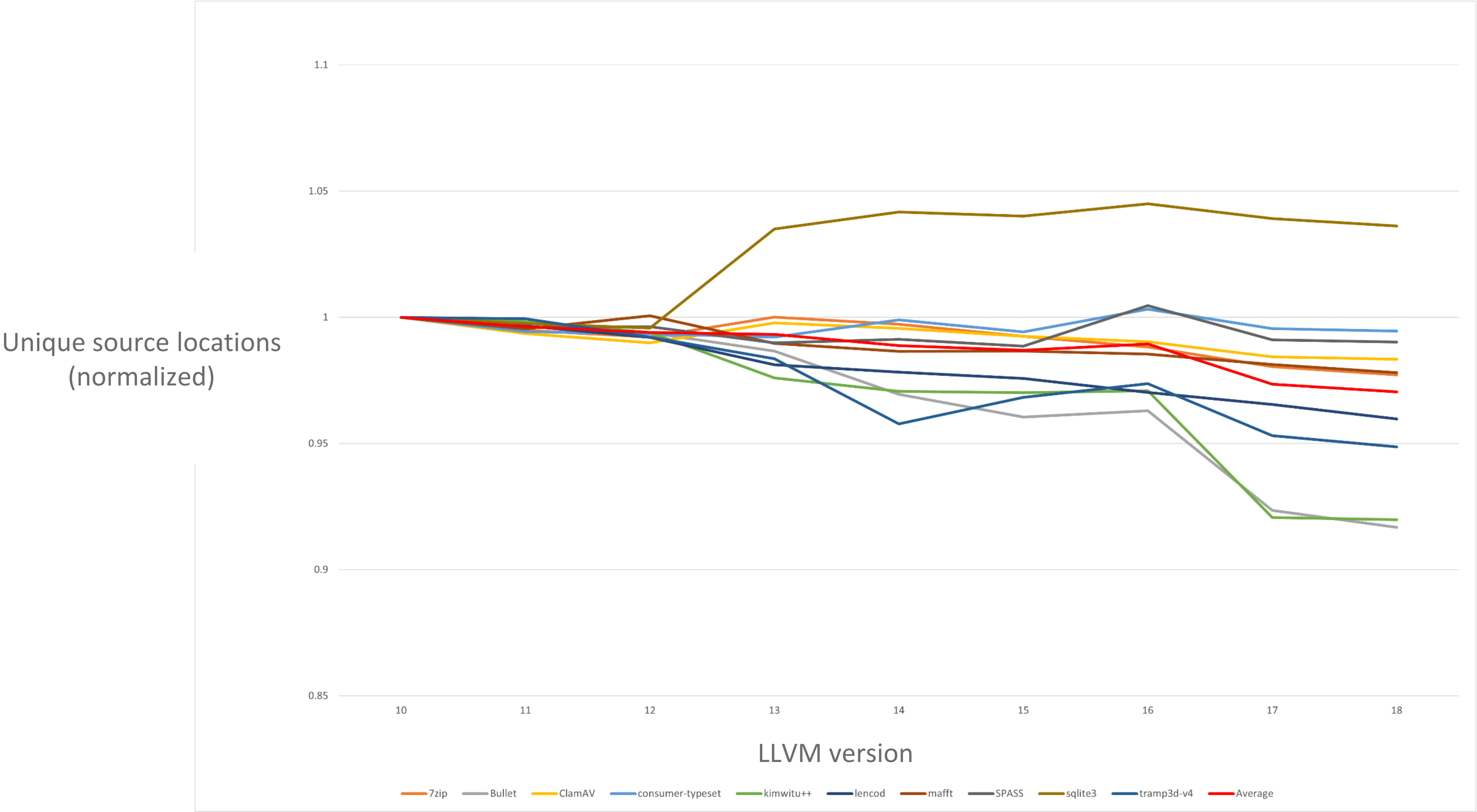
LLVM Source Code

```cpp
bool MachineSinking::PerformSinkAndFold(MachineInstr &MI,
  for (auto &[SinkDst, MaybeAM] : SinkInto) {
    if (SinkDst->isCopy()) {
      // Sink a copy of the instruction, replacing a COPY instruction.
      MachineBasicBlock::iterator InsertPt = SinkDst->getIterator();
      Register DstReg = SinkDst->getOperand(0).getReg();
      TII->reMaterialize(*SinkDst->getParent(), InsertPt, DstReg, 0, MI, *TRI);
      New = &*std::prev(InsertPt);
      if (!New->getDebugLoc())
        New->setDebugLoc(SinkDst->getDebugLoc());
```

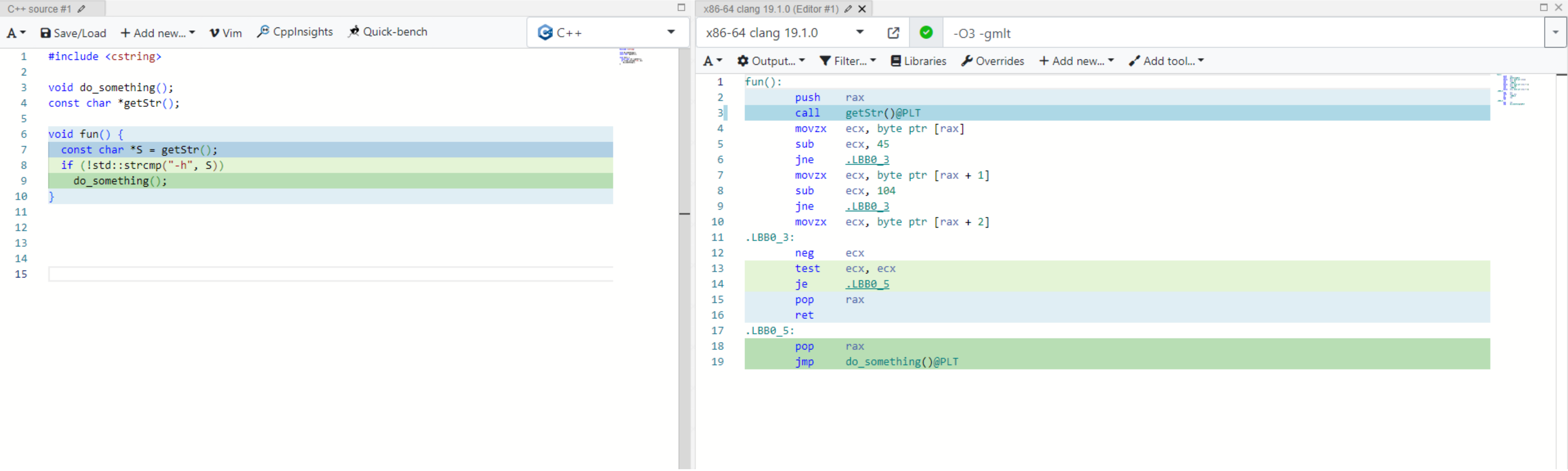
!DILocation metadata attached to instructions

Source location defect finder

https://discourse.llvm.org/t/rfc-proposed-update-to-handling-debug-locations-in-llvm


Unique source locations (normalized)
1.1
1.05
1
0.95
0.9
0.85
10
11
12
13
14
15
16
17
18
LLVM version
7zip
Bullet
ClamAV
consumer-typeset
kimwitu++
lencod
mafft
SPASS
sqlite3
tramp3d-v4
Average

Misattribution



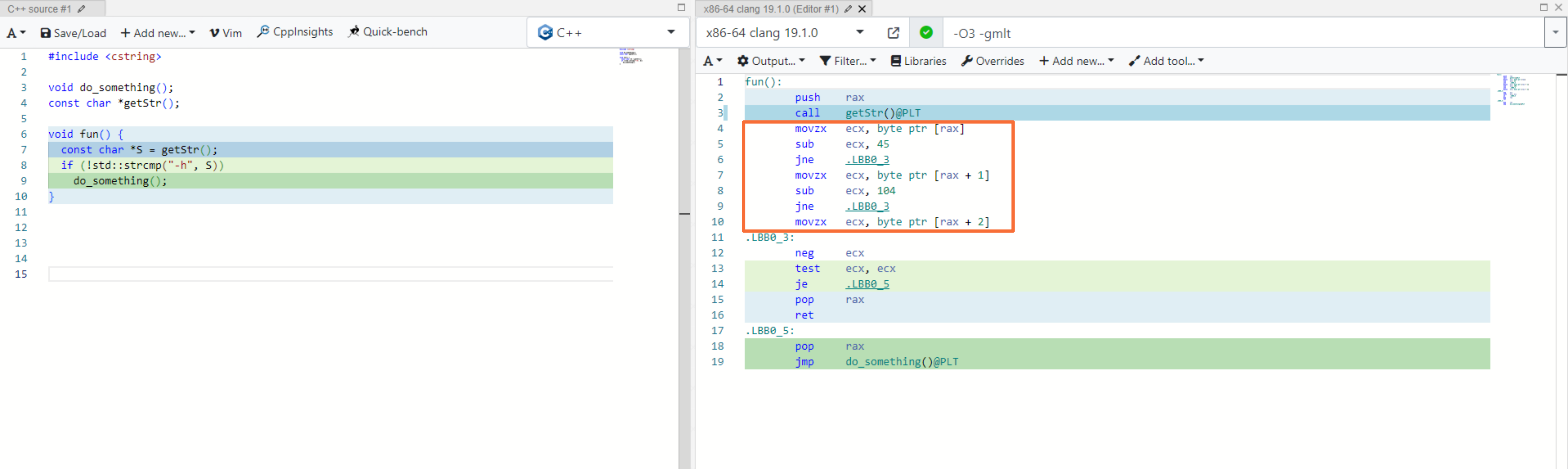
```cpp
#include <cstring>

void do_something();
const char *getStr();

void fun() {
  const char *S = getStr();
  if (!std::strcmp("-h", S))
    do_something();
}
```

x86-64 clang 19.1.0

-O3 -gmlt

```asm
fun():
        push    rax
        call    getStr()@PLT
        movzx   ecx, byte ptr [rax]
        sub     ecx, 45
        jne     .LBB0_3
        movzx   ecx, byte ptr [rax + 1]
        sub     ecx, 104
        jne     .LBB0_3
        movzx   ecx, byte ptr [rax + 2]
.LBB0_3:
        neg     ecx
        test    ecx, ecx
        je      .LBB0_5
        pop     rax
        ret
.LBB0_5:
        pop     rax
        jmp     do_something()@PLT
```

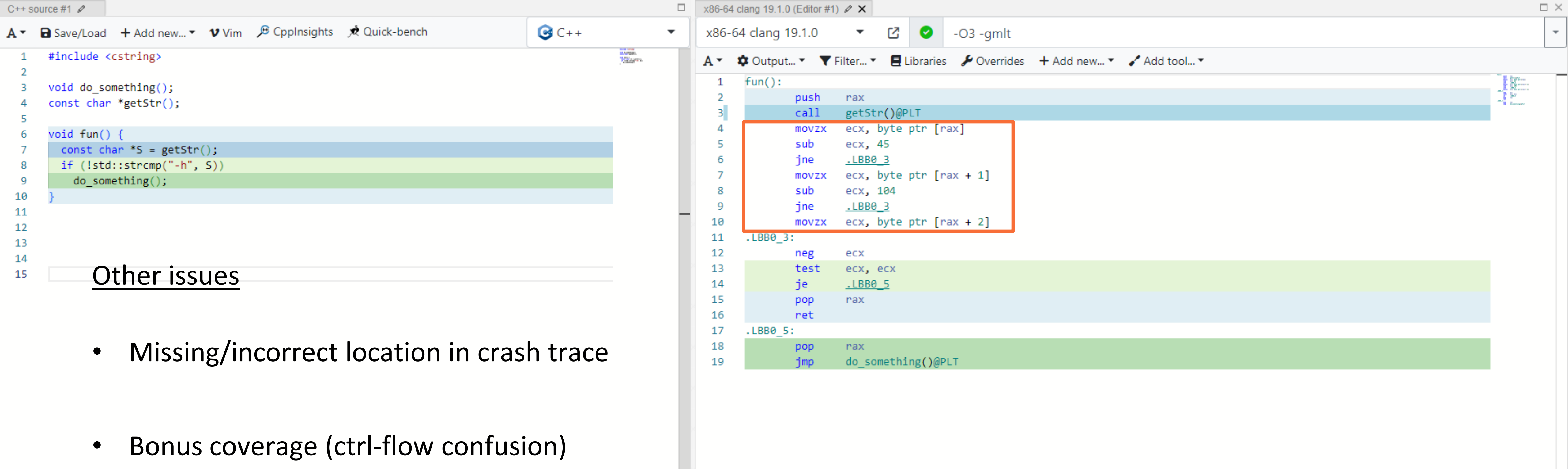
Misattribution



```cpp
#include <cstring>

void do_something();
const char *getStr();

void fun() {
  const char *S = getStr();
  if (!std::strcmp("-h", S))
    do_something();
}
```

```asm
fun():
        push    rax
        call    getStr()@PLT
        movzx   ecx, byte ptr [rax]
        sub     ecx, 45
        jne     .LBB0_3
        movzx   ecx, byte ptr [rax + 1]
        sub     ecx, 104
        jne     .LBB0_3
        movzx   ecx, byte ptr [rax + 2]
.LBB0_3:
        neg     ecx
        test    ecx, ecx
        je      .LBB0_5
        pop     rax
        ret
.LBB0_5:
        pop     rax
        jmp     do_something()@PLT
```

Misattribution

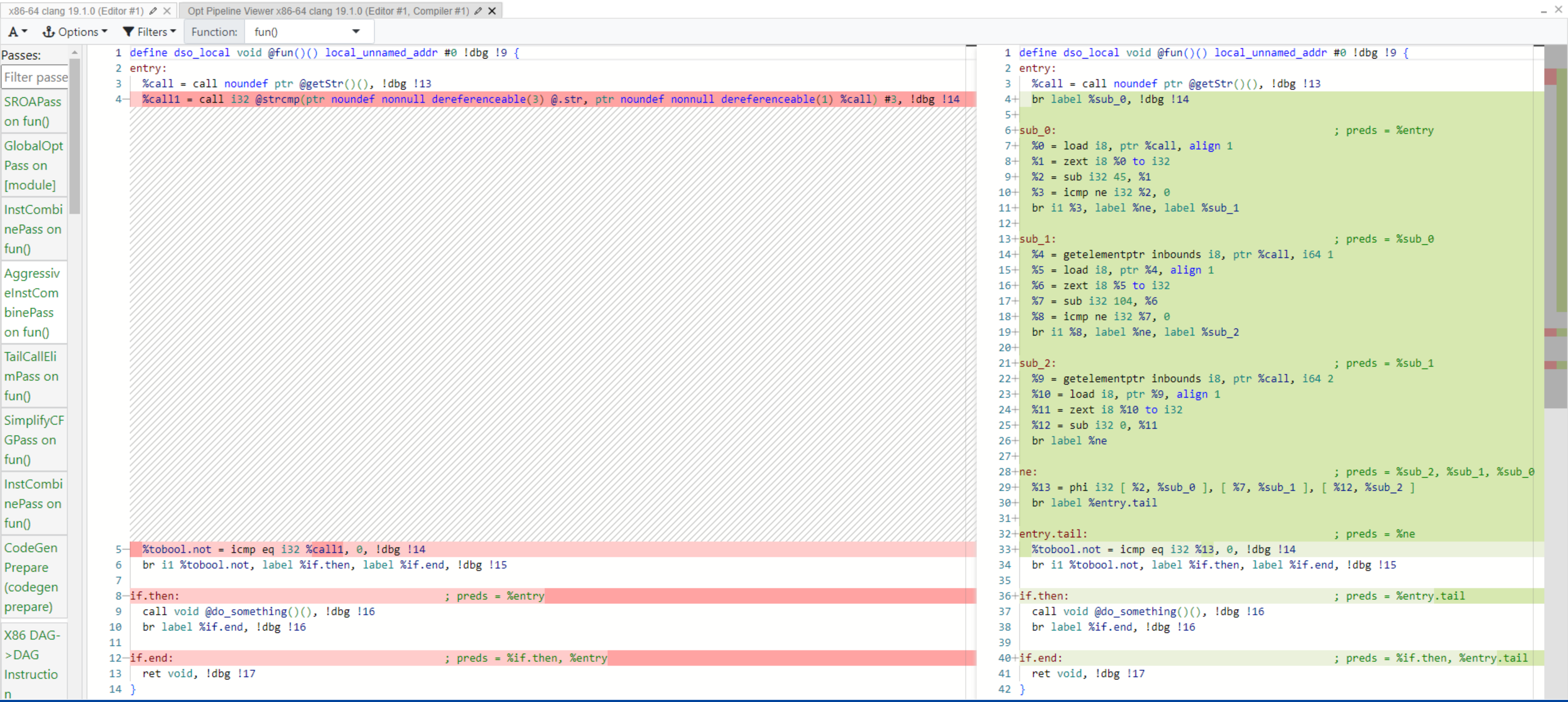


```cpp
#include <cstring>

void do_something();
const char *getStr();

void fun() {
  const char *S = getStr();
  if (!std::strcmp("-h", S))
    do_something();
}
```

```asm
fun():
        push    rax
        call    getStr()@PLT
        movzx   ecx, byte ptr [rax]
        sub     ecx, 45
        jne     .LBB0_3
        movzx   ecx, byte ptr [rax + 1]
        sub     ecx, 104
        jne     .LBB0_3
        movzx   ecx, byte ptr [rax + 2]
.LBB0_3:
        neg     ecx
        test    ecx, ecx
        je      .LBB0_5
        pop     rax
        ret
.LBB0_5:
        pop     rax
        jmp     do_something()@PLT
```

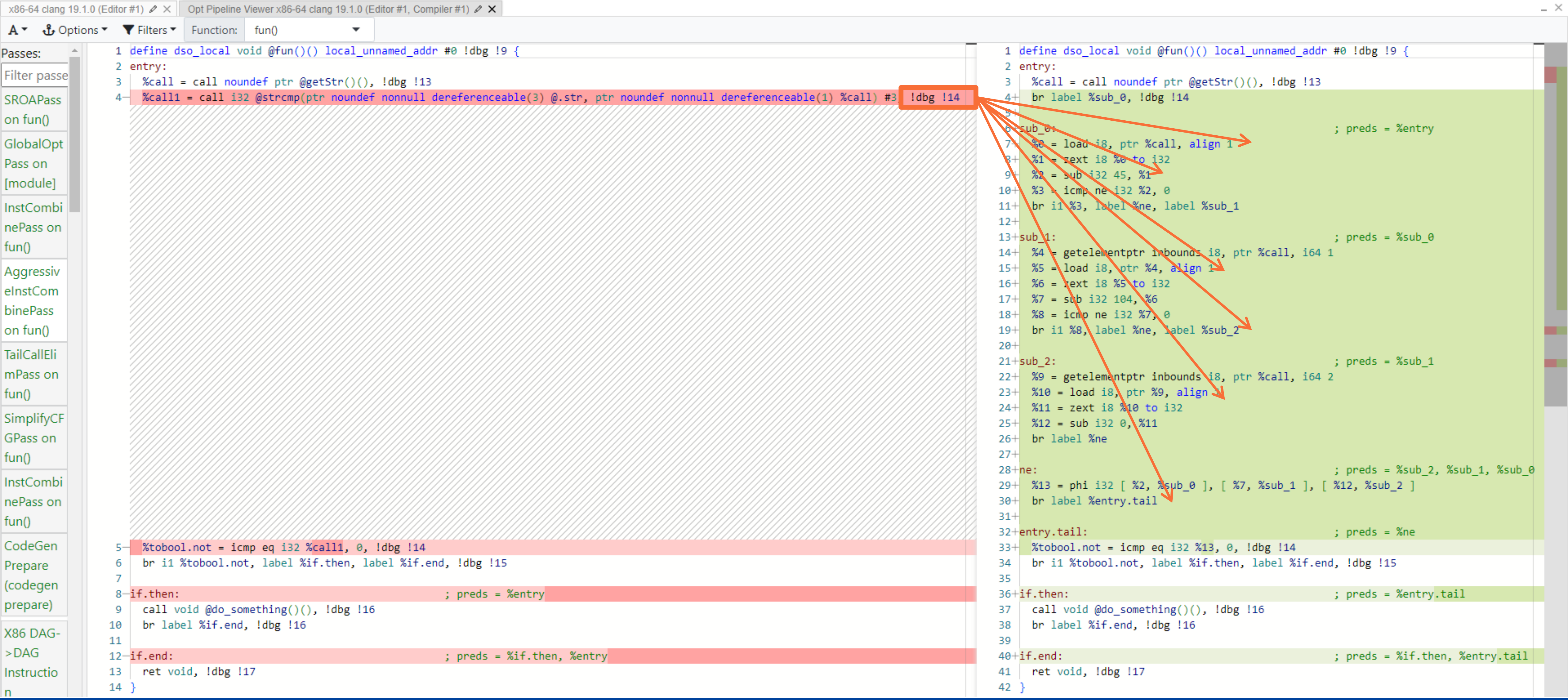
Other issues

- Missing/incorrect location in crash trace
- Bonus coverage (ctrl-flow confusion)
- Cover unreachable code


x86-64 clang 19.1.0 (Editor #1)
Opt Pipeline Viewer x86-64 clang 19.1.0 (Editor #1, Compiler #1)
A
Options
Filters
Function:
fun()
Passes:
Filter passes
SROAPass on fun()
GlobalOptPass on [module]
InstCombinePass on fun()
AggressiveInstCombinePass on fun()
TailCallElimPass on fun()
SimplifyCFGPass on fun()
InstCombinePass on fun()
CodeGenPrepare (codegenprepare)
X86 DAG->DAG Instruction

```
define dso_local void @fun()() local_unnamed_addr #0 !dbg !9 {
entry:
  %call = call noundef ptr @getStr()(), !dbg !13
  %call1 = call i32 @strcmp(ptr noundef nonnull dereferenceable(3) @.str, ptr noundef nonnull dereferenceable(1) %call) #3, !dbg !14
  %tobool.not = icmp eq i32 %call1, 0, !dbg !14
  br i1 %tobool.not, label %if.then, label %if.end, !dbg !15

if.then:                                          ; preds = %entry
  call void @do_something()(), !dbg !16
  br label %if.end, !dbg !16

if.end:                                           ; preds = %if.then, %entry
  ret void, !dbg !17
}
```

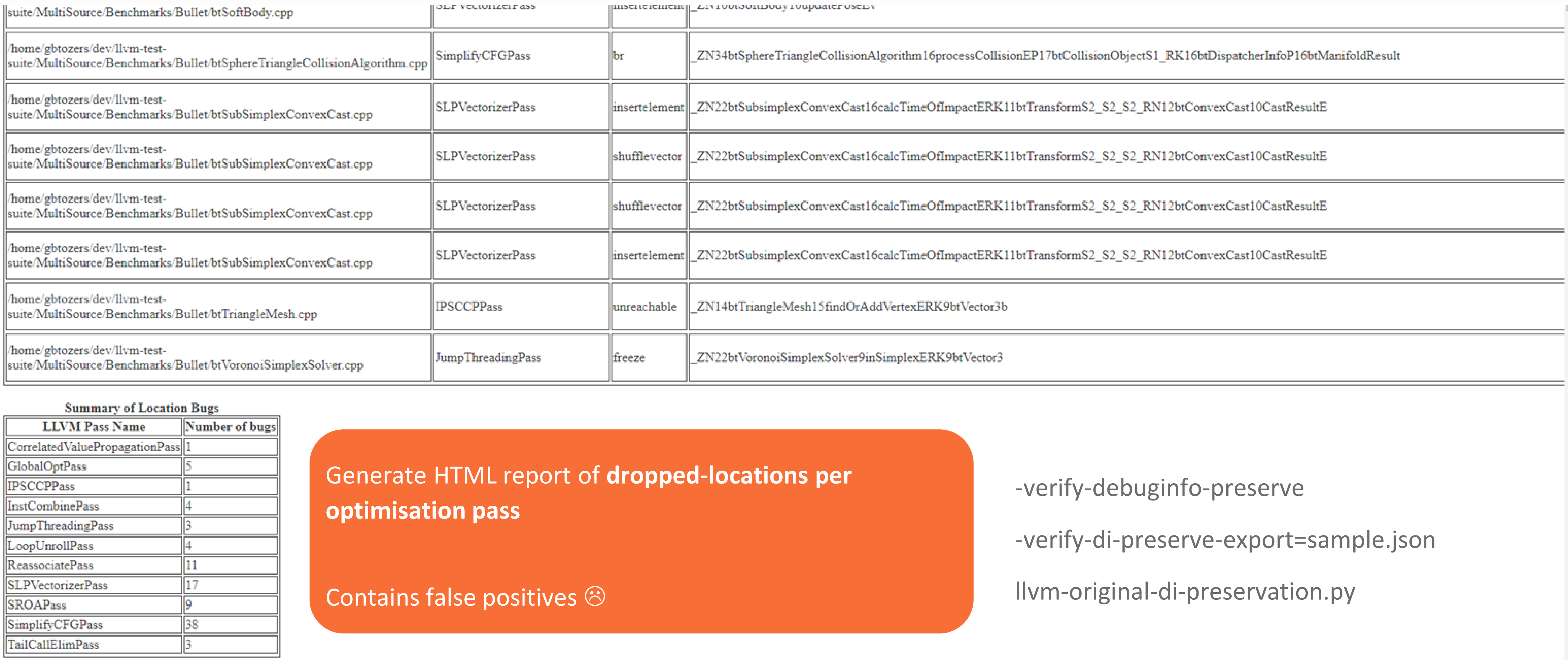
```
define dso_local void @fun()() local_unnamed_addr #0 !dbg !9 {
entry:
  %call = call noundef ptr @getStr()(), !dbg !13
  br label %sub_0, !dbg !14

sub_0:                                            ; preds = %entry
  %0 = load i8, ptr %call, align 1
  %1 = zext i8 %0 to i32
  %2 = sub i32 45, %1
  %3 = icmp ne i32 %2, 0
  br i1 %3, label %ne, label %sub_1

sub_1:                                            ; preds = %sub_0
  %4 = getelementptr inbounds i8, ptr %call, i64 1
  %5 = load i8, ptr %4, align 1
  %6 = zext i8 %5 to i32
  %7 = sub i32 104, %6
  %8 = icmp ne i32 %7, 0
  br i1 %8, label %ne, label %sub_2

sub_2:                                            ; preds = %sub_1
  %9 = getelementptr inbounds i8, ptr %call, i64 2
  %10 = load i8, ptr %9, align 1
  %11 = zext i8 %10 to i32
  %12 = sub i32 0, %11
  br label %ne

ne:                                               ; preds = %sub_2, %sub_1, %sub_0
  %13 = phi i32 [ %2, %sub_0 ], [ %7, %sub_1 ], [ %12, %sub_2 ]
  br label %entry.tail

entry.tail:                                       ; preds = %ne
  %tobool.not = icmp eq i32 %13, 0, !dbg !14
  br i1 %tobool.not, label %if.then, label %if.end, !dbg !15

if.then:                                          ; preds = %entry.tail
  call void @do_something()(), !dbg !16
  br label %if.end, !dbg !16

if.end:                                           ; preds = %if.then, %entry.tail
  ret void, !dbg !17
}
```


x86-64 clang 19.1.0 (Editor #1)
Opt Pipeline Viewer x86-64 clang 19.1.0 (Editor #1, Compiler #1)
A
Options
Filters
Function:
fun()
Passes:
Filter passes
SROAPass on fun()
GlobalOptPass on [module]
InstCombinePass on fun()
AggressiveInstCombinePass on fun()
TailCallElimPass on fun()
SimplifyCFGPass on fun()
InstCombinePass on fun()
CodeGenPrepare (codegen prepare)
X86 DAG->DAG Instruction

```
define dso_local void @fun()() local_unnamed_addr #0 !dbg !9 {
entry:
  %call = call noundef ptr @getStr()(), !dbg !13
  %call1 = call i32 @strcmp(ptr noundef nonnull dereferenceable(3) @.str, ptr noundef nonnull dereferenceable(1) %call) #3 !dbg !14
  %tobool.not = icmp eq i32 %call1, 0, !dbg !14
  br i1 %tobool.not, label %if.then, label %if.end, !dbg !15

if.then:                                          ; preds = %entry
  call void @do_something()(), !dbg !16
  br label %if.end, !dbg !16

if.end:                                           ; preds = %if.then, %entry
  ret void, !dbg !17
}
```

```
define dso_local void @fun()() local_unnamed_addr #0 !dbg !9 {
entry:
  %call = call noundef ptr @getStr()(), !dbg !13
  br label %sub_0, !dbg !14

sub_0:                                            ; preds = %entry
  %0 = load i8, ptr %call, align 1
  %1 = zext i8 %0 to i32
  %2 = sub i32 45, %1
  %3 = icmp ne i32 %2, 0
  br i1 %3, label %ne, label %sub_1

sub_1:                                            ; preds = %sub_0
  %4 = getelementptr inbounds i8, ptr %call, i64 1
  %5 = load i8, ptr %4, align 1
  %6 = zext i8 %5 to i32
  %7 = sub i32 104, %6
  %8 = icmp ne i32 %7, 0
  br i1 %8, label %ne, label %sub_2

sub_2:                                            ; preds = %sub_1
  %9 = getelementptr inbounds i8, ptr %call, i64 2
  %10 = load i8, ptr %9, align 1
  %11 = zext i8 %10 to i32
  %12 = sub i32 0, %11
  br label %ne

ne:                                               ; preds = %sub_2, %sub_1, %sub_0
  %13 = phi i32 [ %2, %sub_0 ], [ %7, %sub_1 ], [ %12, %sub_2 ]
  br label %entry.tail

entry.tail:                                       ; preds = %ne
  %tobool.not = icmp eq i32 %13, 0, !dbg !14
  br i1 %tobool.not, label %if.then, label %if.end, !dbg !15

if.then:                                          ; preds = %entry.tail
  call void @do_something()(), !dbg !16
  br label %if.end, !dbg !16

if.end:                                           ; preds = %if.then, %entry.tail
  ret void, !dbg !17
}
```

Existing tooling - Debugify



suite/MultiSource/Benchmarks/Bullet/btSoftBody.cpp	SLPVectorizerPass	insertelement	_ZN10btSoftBody10updatePoseEv
/home/gbtozers/dev/llvm-test-suite/MultiSource/Benchmarks/Bullet/btSphereTriangleCollisionAlgorithm.cpp	SimplifyCFGPass	br	_ZN34btSphereTriangleCollisionAlgorithm16processCollisionEP17btCollisionObjectS1_RK16btDispatcherInfoP16btManifoldResult
/home/gbtozers/dev/llvm-test-suite/MultiSource/Benchmarks/Bullet/btSubSimplexConvexCast.cpp	SLPVectorizerPass	insertelement	_ZN22btSubsimplexConvexCast16calcTimeOfImpactERK11btTransformS2_S2_S2_RN12btConvexCast10CastResultE
/home/gbtozers/dev/llvm-test-suite/MultiSource/Benchmarks/Bullet/btSubSimplexConvexCast.cpp	SLPVectorizerPass	shufflevector	_ZN22btSubsimplexConvexCast16calcTimeOfImpactERK11btTransformS2_S2_S2_RN12btConvexCast10CastResultE
/home/gbtozers/dev/llvm-test-suite/MultiSource/Benchmarks/Bullet/btSubSimplexConvexCast.cpp	SLPVectorizerPass	shufflevector	_ZN22btSubsimplexConvexCast16calcTimeOfImpactERK11btTransformS2_S2_S2_RN12btConvexCast10CastResultE
/home/gbtozers/dev/llvm-test-suite/MultiSource/Benchmarks/Bullet/btSubSimplexConvexCast.cpp	SLPVectorizerPass	insertelement	_ZN22btSubsimplexConvexCast16calcTimeOfImpactERK11btTransformS2_S2_S2_RN12btConvexCast10CastResultE
/home/gbtozers/dev/llvm-test-suite/MultiSource/Benchmarks/Bullet/btTriangleMesh.cpp	IPSCCPPass	unreachable	_ZN14btTriangleMesh15findOrAddVertexERK9btVector3b
/home/gbtozers/dev/llvm-test-suite/MultiSource/Benchmarks/Bullet/btVoronoiSimplexSolver.cpp	JumpThreadingPass	freeze	_ZN22btVoronoiSimplexSolver9inSimplexERK9btVector3

Summary of Location Bugs

LLVM Pass Name	Number of bugs
CorrelatedValuePropagationPass	1
GlobalOptPass	5
IPSCCPPass	1
InstCombinePass	4
JumpThreadingPass	3
LoopUnrollPass	4
ReassociatePass	11
SLPVectorizerPass	17
SROAPass	9
SimplifyCFGPass	38
TailCallElimPass	3

Generate HTML report of **dropped-locations per optimisation pass**

Contains false positives ☹

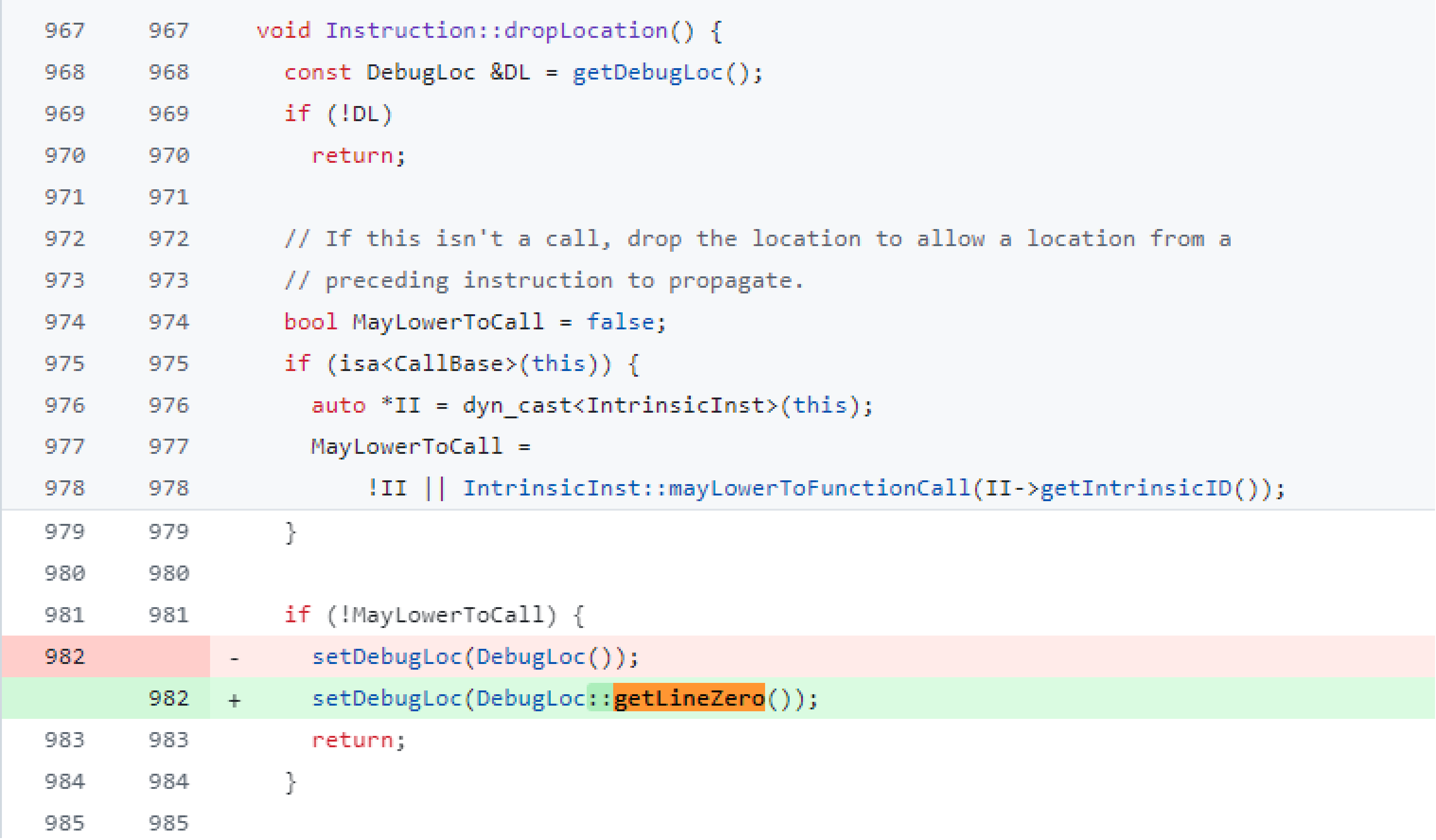
-verify-debuginfo-preserve

-verify-di-preserve-export=sample.json

llvm-original-di-preservation.py

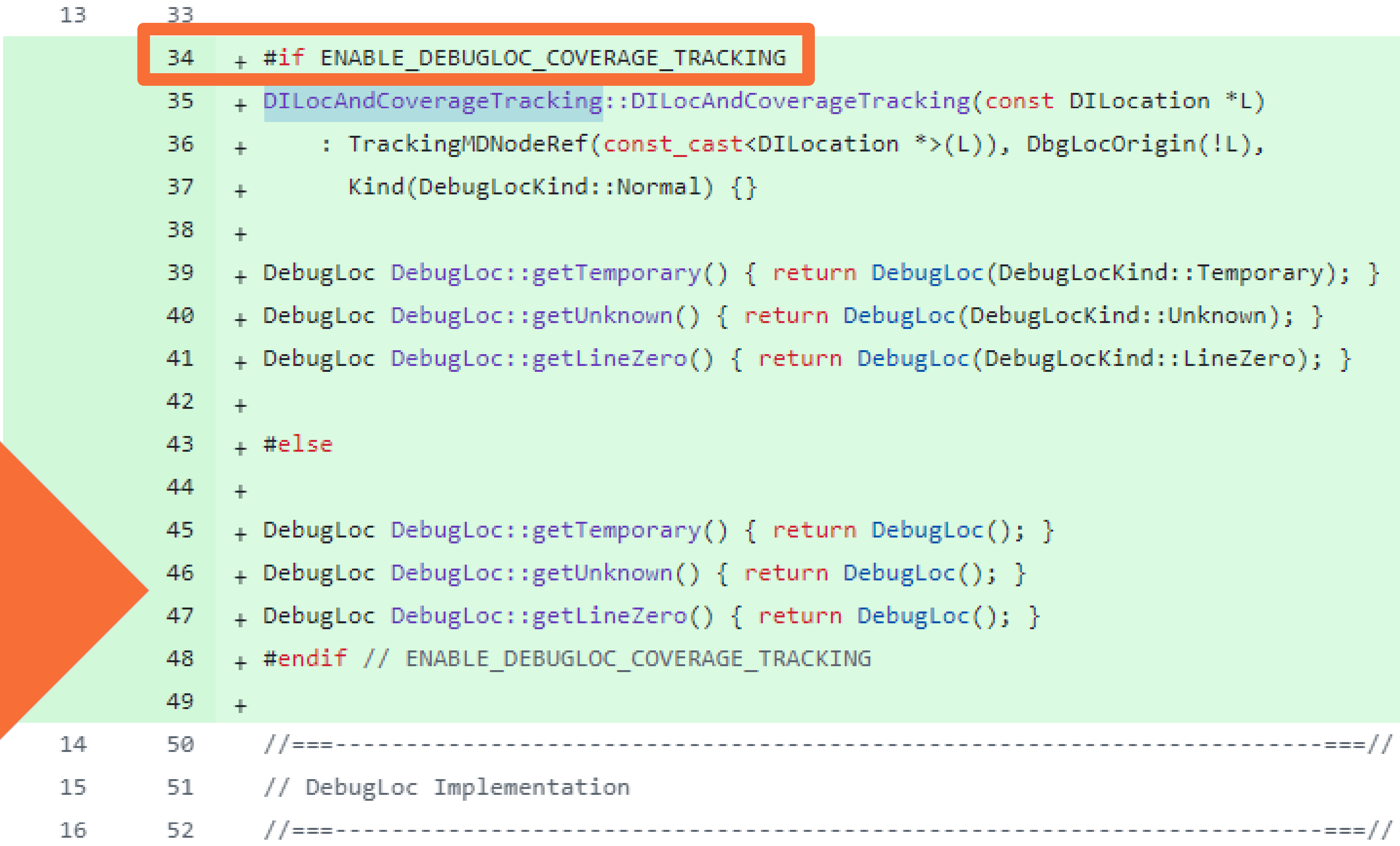
Solution

Declare intent using new API



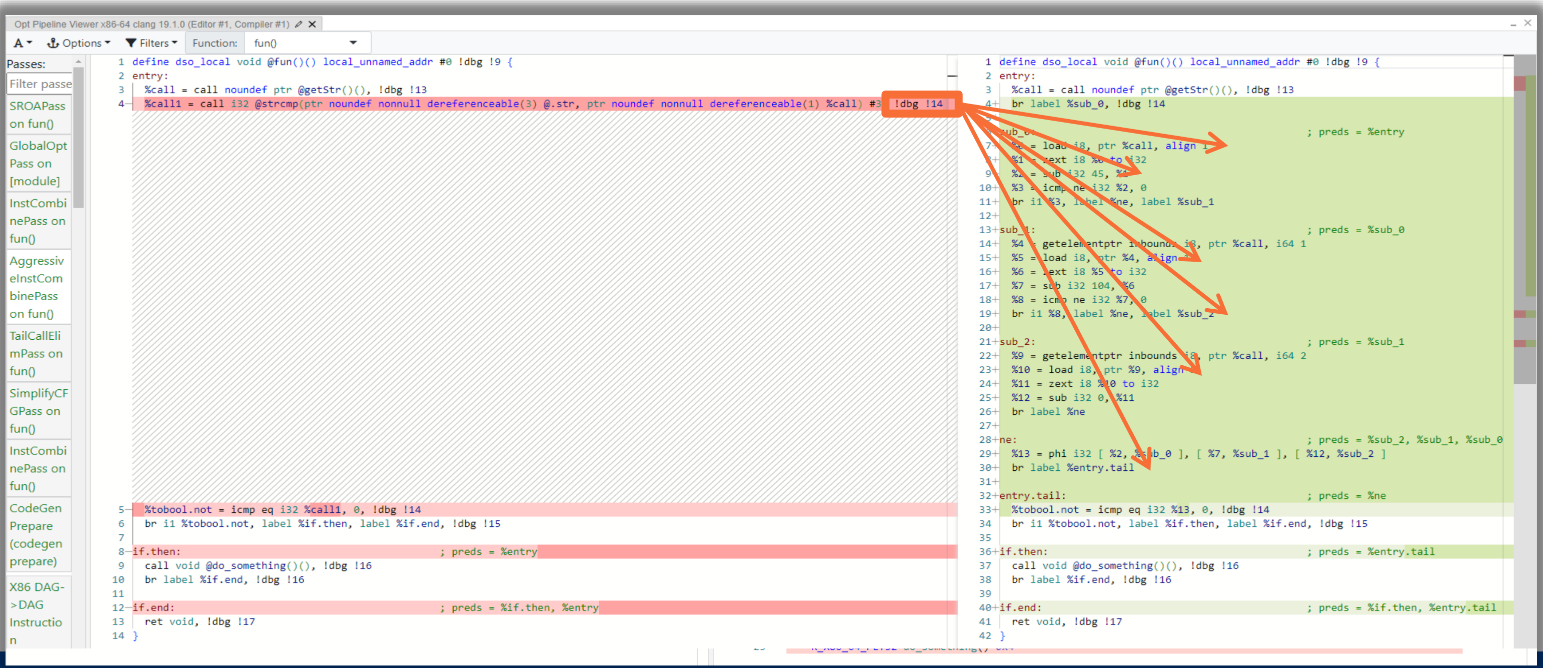
```
  void Instruction::dropLocation() {
    const DebugLoc &DL = getDebugLoc();
    if (!DL)
      return;

    // If this isn't a call, drop the location to allow a location from a
    // preceding instruction to propagate.
    bool MayLowerToCall = false;
    if (isa<CallBase>(this)) {
      auto *II = dyn_cast<IntrinsicInst>(this);
      MayLowerToCall =
          !II || IntrinsicInst::mayLowerToFunctionCall(II->getIntrinsicID());
    }

    if (!MayLowerToCall) {
-     setDebugLoc(DebugLoc());
+     setDebugLoc(DebugLoc::getLineZero());
      return;
    }

```

DebugLoc API



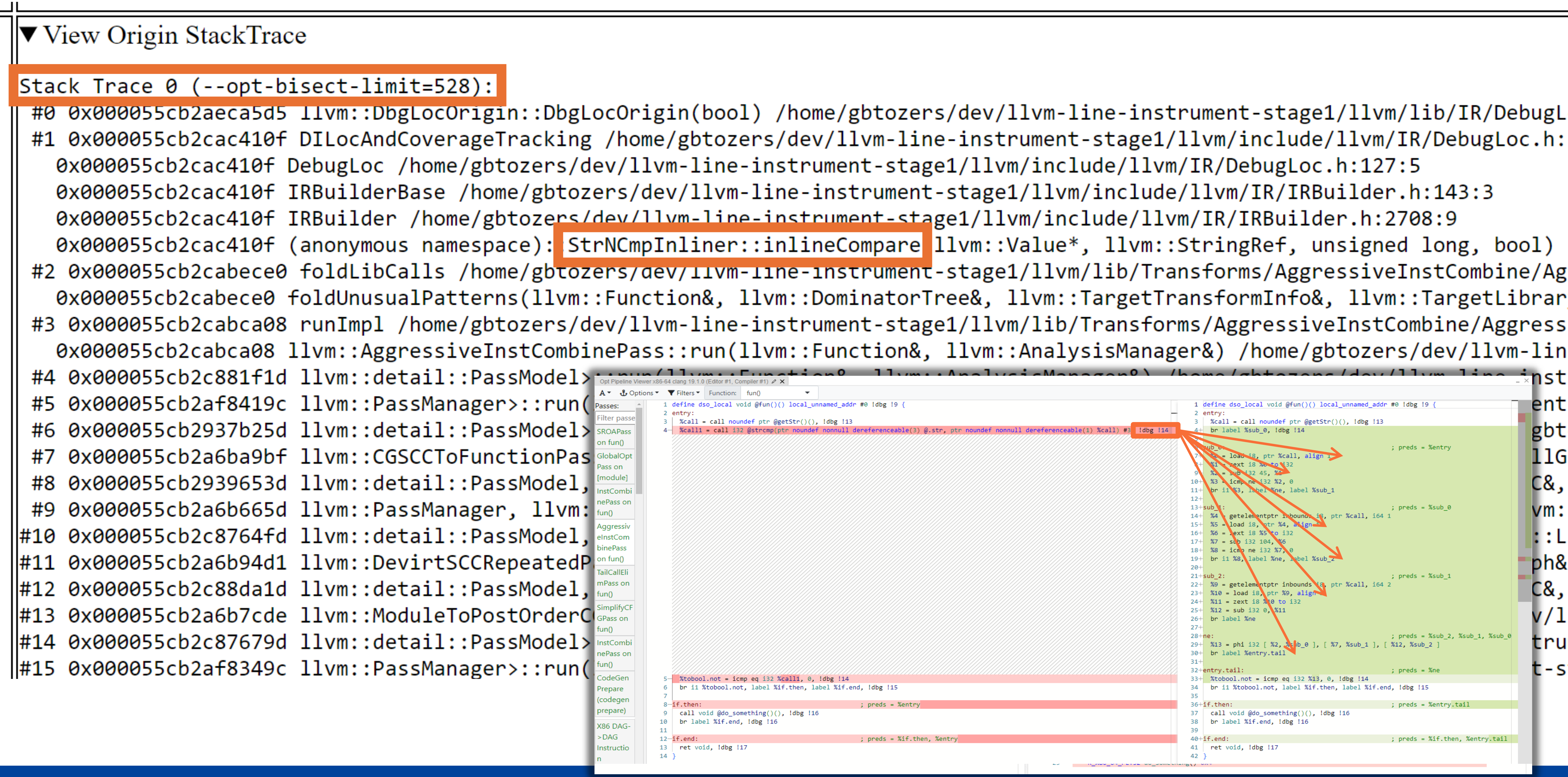
```
 
+#if ENABLE_DEBUGLOC_COVERAGE_TRACKING
+DILocAndCoverageTracking::DILocAndCoverageTracking(const DILocation *L)
+    : TrackingMDNodeRef(const_cast<DILocation *>(L)), DbgLocOrigin(!L),
+      Kind(DebugLocKind::Normal) {}
+
+DebugLoc DebugLoc::getTemporary() { return DebugLoc(DebugLocKind::Temporary); }
+DebugLoc DebugLoc::getUnknown() { return DebugLoc(DebugLocKind::Unknown); }
+DebugLoc DebugLoc::getLineZero() { return DebugLoc(DebugLocKind::LineZero); }
+
+#else
+
+DebugLoc DebugLoc::getTemporary() { return DebugLoc(); }
+DebugLoc DebugLoc::getUnknown() { return DebugLoc(); }
+DebugLoc DebugLoc::getLineZero() { return DebugLoc(); }
+#endif // ENABLE_DEBUGLOC_COVERAGE_TRACKING
+
 //===----------------------------------------------------------------------===//
 // DebugLoc Implementation
 //===----------------------------------------------------------------------===//
```

Does nothing by default


Opt Pipeline Viewer x86-64 clang 19.1.0 (Editor #1, Compiler #1)
A
Options
Filters
Function:
fun()
Passes:
Filter passe
SROAPass on fun()
GlobalOptPass on [module]
InstCombinePass on fun()
AggressiveInstCombinePass on fun()
TailCallElimPass on fun()
SimplifyCFGPass on fun()
InstCombinePass on fun()
CodeGenPrepare (codegen prepare)
X86 DAG->DAG Instruction

```
define dso_local void @fun()() local_unnamed_addr #0 !dbg !9 {
entry:
  %call = call noundef ptr @getStr()(), !dbg !13
  %call1 = call i32 @strcmp(ptr noundef nonnull dereferenceable(3) @.str, ptr noundef nonnull dereferenceable(1) %call) #3 !dbg !14
  %tobool.not = icmp eq i32 %call1, 0, !dbg !14
  br i1 %tobool.not, label %if.then, label %if.end, !dbg !15

if.then:                                          ; preds = %entry
  call void @do_something()(), !dbg !16
  br label %if.end, !dbg !16

if.end:                                           ; preds = %if.then, %entry
  ret void, !dbg !17
}
```

```
define dso_local void @fun()() local_unnamed_addr #0 !dbg !9 {
entry:
  %call = call noundef ptr @getStr()(), !dbg !13
  br label %sub_0, !dbg !14

sub_0:                                            ; preds = %entry
  %0 = load i8, ptr %call, align 1
  %1 = zext i8 %0 to i32
  %2 = sub i32 45, %1
  %3 = icmp ne i32 %2, 0
  br i1 %3, label %ne, label %sub_1

sub_1:                                            ; preds = %sub_0
  %4 = getelementptr inbounds i8, ptr %call, i64 1
  %5 = load i8, ptr %4, align 1
  %6 = zext i8 %5 to i32
  %7 = sub i32 104, %6
  %8 = icmp ne i32 %7, 0
  br i1 %8, label %ne, label %sub_2

sub_2:                                            ; preds = %sub_1
  %9 = getelementptr inbounds i8, ptr %call, i64 2
  %10 = load i8, ptr %9, align 1
  %11 = zext i8 %10 to i32
  %12 = sub i32 0, %11
  br label %ne

ne:                                               ; preds = %sub_2, %sub_1, %sub_0
  %13 = phi i32 [ %2, %sub_0 ], [ %7, %sub_1 ], [ %12, %sub_2 ]
  br label %entry.tail

entry.tail:                                       ; preds = %ne
  %tobool.not = icmp eq i32 %13, 0, !dbg !14
  br i1 %tobool.not, label %if.then, label %if.end, !dbg !15

if.then:                                          ; preds = %entry.tail
  call void @do_something()(), !dbg !16
  br label %if.end, !dbg !16

if.end:                                           ; preds = %if.then, %entry.tail
  ret void, !dbg !17
}
```



▼ View Origin StackTrace

```
Stack Trace 0 (--opt-bisect-limit=528):
 #0 0x000055cb2aeca5d5 llvm::DbgLocOrigin::DbgLocOrigin(bool) /home/gbtozers/dev/llvm-line-instrument-stage1/llvm/lib/IR/DebugL
 #1 0x000055cb2cac410f DILocAndCoverageTracking /home/gbtozers/dev/llvm-line-instrument-stage1/llvm/include/llvm/IR/DebugLoc.h:
   0x000055cb2cac410f DebugLoc /home/gbtozers/dev/llvm-line-instrument-stage1/llvm/include/llvm/IR/DebugLoc.h:127:5
   0x000055cb2cac410f IRBuilderBase /home/gbtozers/dev/llvm-line-instrument-stage1/llvm/include/llvm/IR/IRBuilder.h:143:3
   0x000055cb2cac410f IRBuilder /home/gbtozers/dev/llvm-line-instrument-stage1/llvm/include/llvm/IR/IRBuilder.h:2708:9
   0x000055cb2cac410f (anonymous namespace): StrNCmpInliner::inlineCompare llvm::Value*, llvm::StringRef, unsigned long, bool)
 #2 0x000055cb2cabece0 foldLibCalls /home/gbtozers/dev/llvm-line-instrument-stage1/llvm/lib/Transforms/AggressiveInstCombine/Ag
   0x000055cb2cabece0 foldUnusualPatterns(llvm::Function&, llvm::DominatorTree&, llvm::TargetTransformInfo&, llvm::TargetLibrar
 #3 0x000055cb2cabca08 runImpl /home/gbtozers/dev/llvm-line-instrument-stage1/llvm/lib/Transforms/AggressiveInstCombine/Aggress
   0x000055cb2cabca08 llvm::AggressiveInstCombinePass::run(llvm::Function&, llvm::AnalysisManager&) /home/gbtozers/dev/llvm-lin
 #4 0x000055cb2c881f1d llvm::detail::PassModel<
 #5 0x000055cb2af8419c llvm::PassManager>::run
 #6 0x000055cb2937b25d llvm::detail::PassModel<
 #7 0x000055cb2a6ba9bf llvm::CGSCCToFunctionPas
 #8 0x000055cb2939653d llvm::detail::PassModel,
 #9 0x000055cb2a6b665d llvm::PassManager, llvm:
#10 0x000055cb2c8764fd llvm::detail::PassModel,
#11 0x000055cb2a6b94d1 llvm::DevirtSCCRepeatedP
#12 0x000055cb2c88da1d llvm::detail::PassModel,
#13 0x000055cb2a6b7cde llvm::ModuleToPostOrderC
#14 0x000055cb2c87679d llvm::detail::PassModel<
#15 0x000055cb2af8349c llvm::PassManager>::run(
```

Opt Pipeline Viewer x86-64 clang 19.1.0 (Editor #1, Compiler #1)
A
Options
Filters
Function:
fun()
Passes:
Filter passe
SROAPass on fun()
GlobalOptPass on [module]
InstCombinePass on fun()
AggressiveInstCombinePass on fun()
TailCallElimPass on fun()
SimplifyCFGPass on fun()
InstCombinePass on fun()
CodeGenPrepare (codegen prepare)
X86 DAG->DAG Instruction

```
define dso_local void @fun()() local_unnamed_addr #0 !dbg !9 {
entry:
  %call = call noundef ptr @getStr()(), !dbg !13
  %call1 = call i32 @strcmp(ptr noundef nonnull dereferenceable(3) @.str, ptr noundef nonnull dereferenceable(1) %call) #3 !dbg !14
  %tobool.not = icmp eq i32 %call1, 0, !dbg !14
  br i1 %tobool.not, label %if.then, label %if.end, !dbg !15

if.then:                                          ; preds = %entry
  call void @do_something()(), !dbg !16
  br label %if.end, !dbg !16

if.end:                                           ; preds = %if.then, %entry
  ret void, !dbg !17
}
```

```
define dso_local void @fun()() local_unnamed_addr #0 !dbg !9 {
entry:
  %call = call noundef ptr @getStr()(), !dbg !13
  br label %sub_0, !dbg !14

sub_0:                                            ; preds = %entry
  %0 = load i8, ptr %call, align 1
  %1 = zext i8 %0 to i32
  %2 = sub i32 45, %1
  %3 = icmp ne i32 %2, 0
  br i1 %3, label %ne, label %sub_1

sub_1:                                            ; preds = %sub_0
  %4 = getelementptr inbounds i8, ptr %call, i64 1
  %5 = load i8, ptr %4, align 1
  %6 = zext i8 %5 to i32
  %7 = sub i32 104, %6
  %8 = icmp ne i32 %7, 0
  br i1 %8, label %ne, label %sub_2

sub_2:                                            ; preds = %sub_1
  %9 = getelementptr inbounds i8, ptr %call, i64 2
  %10 = load i8, ptr %9, align 1
  %11 = zext i8 %10 to i32
  %12 = sub i32 0, %11
  br label %ne

ne:                                               ; preds = %sub_2, %sub_1, %sub_0
  %13 = phi i32 [ %2, %sub_0 ], [ %7, %sub_1 ], [ %12, %sub_2 ]
  br label %entry.tail

entry.tail:                                       ; preds = %ne
  %tobool.not = icmp eq i32 %13, 0, !dbg !14
  br i1 %tobool.not, label %if.then, label %if.end, !dbg !15

if.then:                                          ; preds = %entry.tail
  call void @do_something()(), !dbg !16
  br label %if.end, !dbg !16

if.end:                                           ; preds = %if.then, %entry.tail
  ret void, !dbg !17
}
```

Defect finder summary

- New defect-finder based on debugify, no false positives

  -DLLVM_ENABLE_DEBUGLOC_COVERAGE_TRACKING = COVERAGE_AND_ORIGIN

- Pass authors encode intent with new API

- We’re fixing existing issues it found

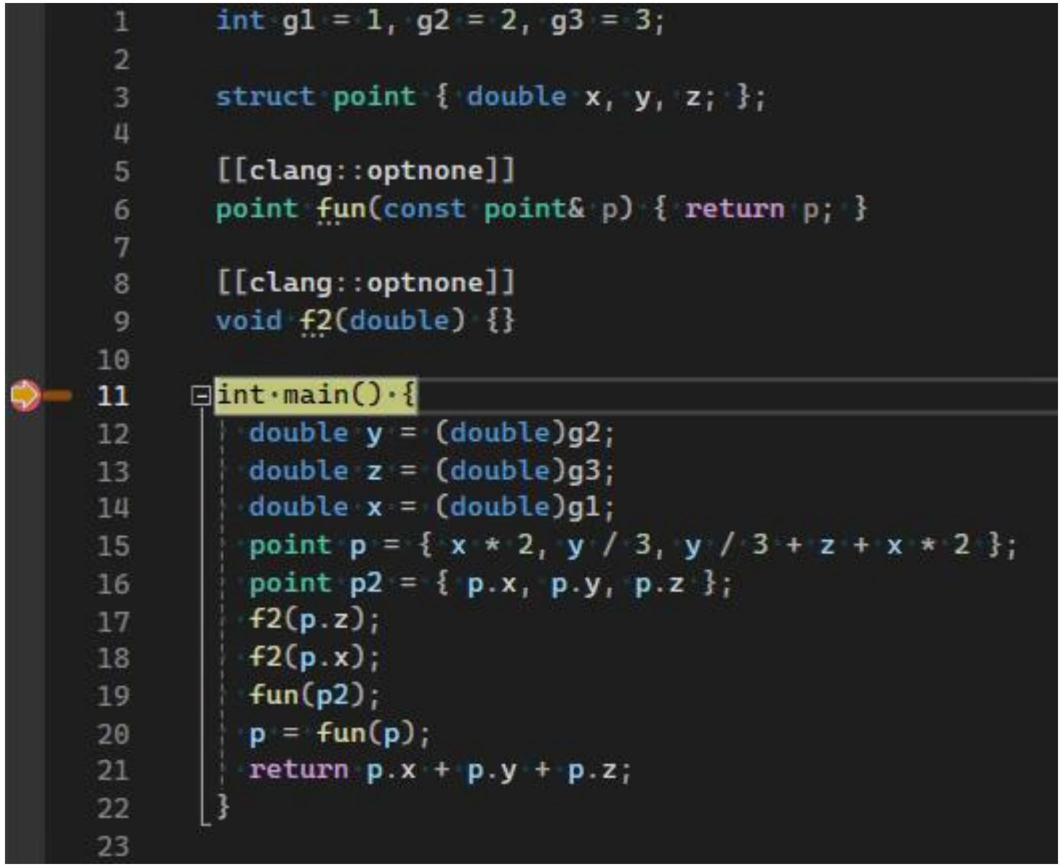
- Then let’s put this on a buildbot

Key Instructions

https://discourse.llvm.org/t/rfc-improving-is-stmt-placement-for-better-interactive-debugging

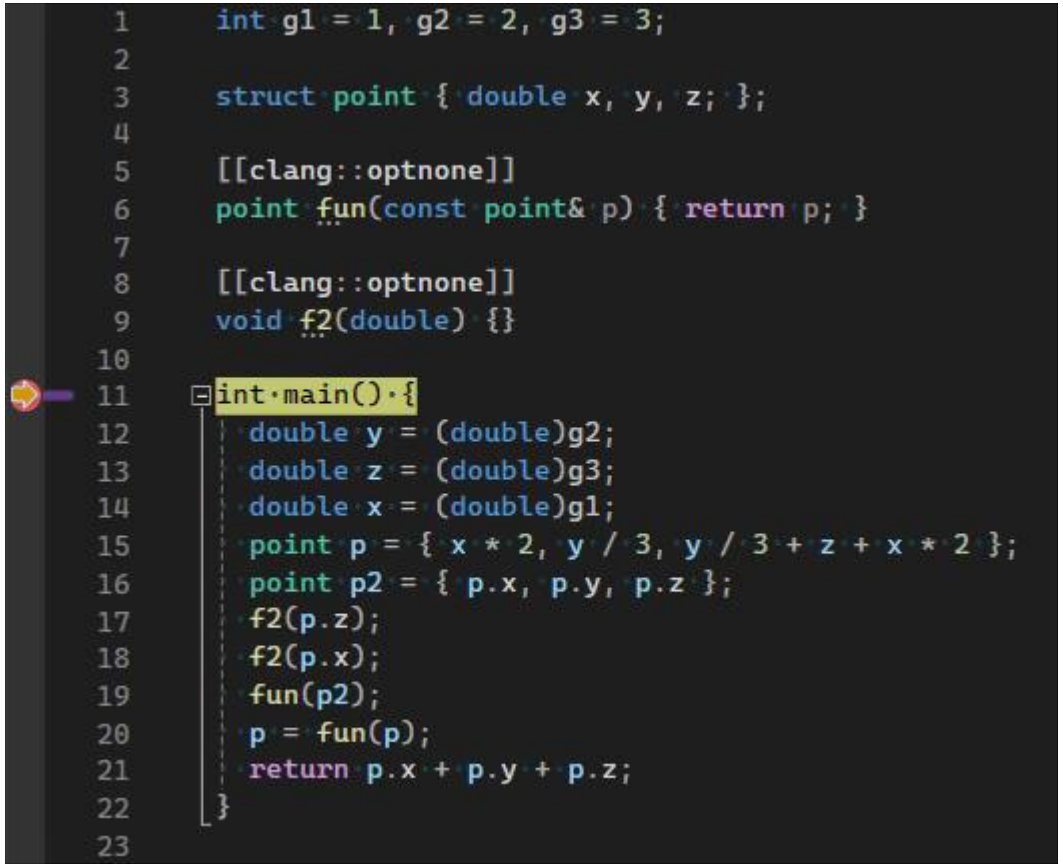
Optimised code debugging

-O2 -g



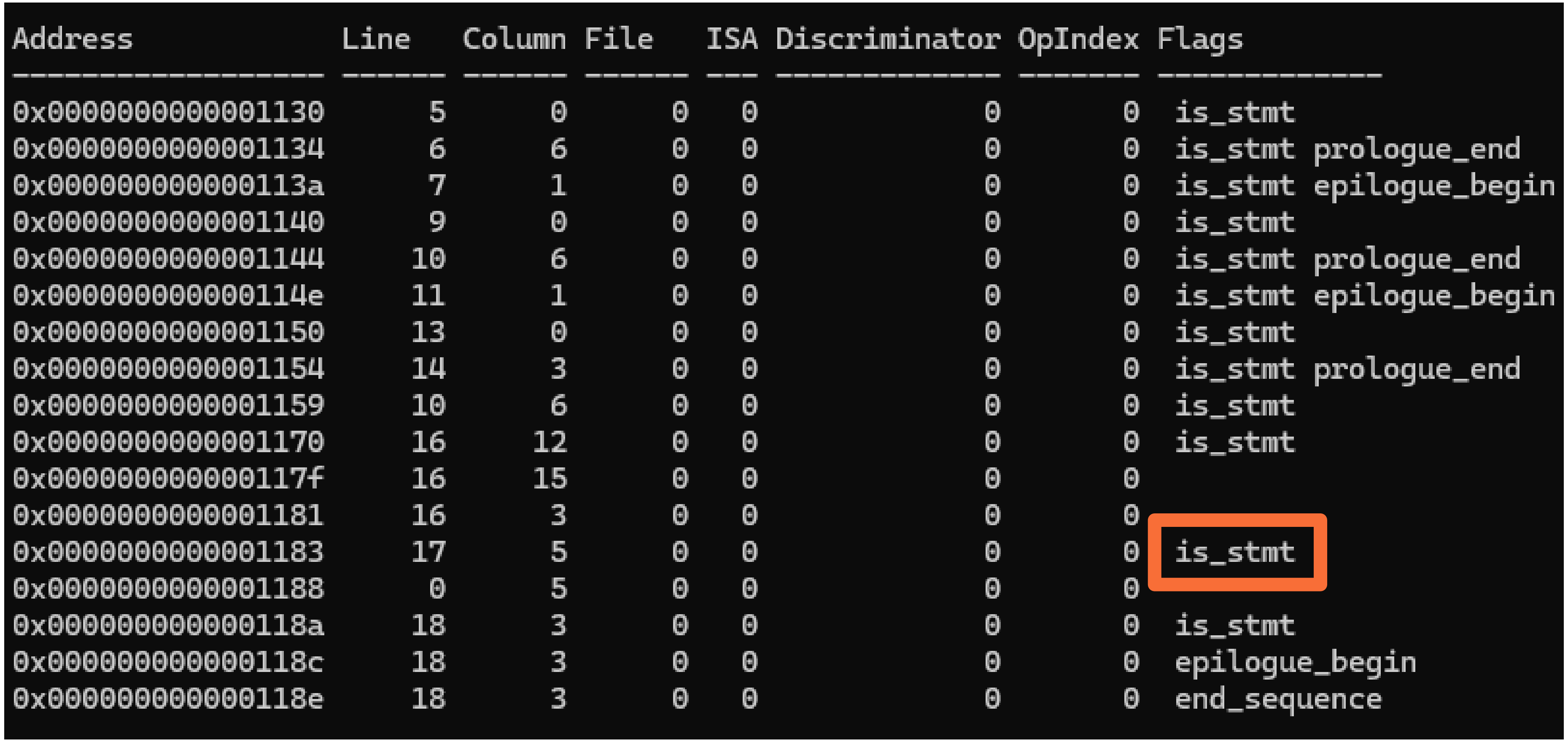
```
int g1 = 1, g2 = 2, g3 = 3;

struct point { double x, y, z; };

[[clang::optnone]]
point fun(const point& p) { return p; }

[[clang::optnone]]
void f2(double) {}

int main() {
    double y = (double)g2;
    double z = (double)g3;
    double x = (double)g1;
    point p = { x * 2, y / 3, y / 3 + z + x * 2 };
    point p2 = { p.x, p.y, p.z };
    f2(p.z);
    f2(p.x);
    fun(p2);
    p = fun(p);
    return p.x + p.y + p.z;
}
```

-O2 –g -???



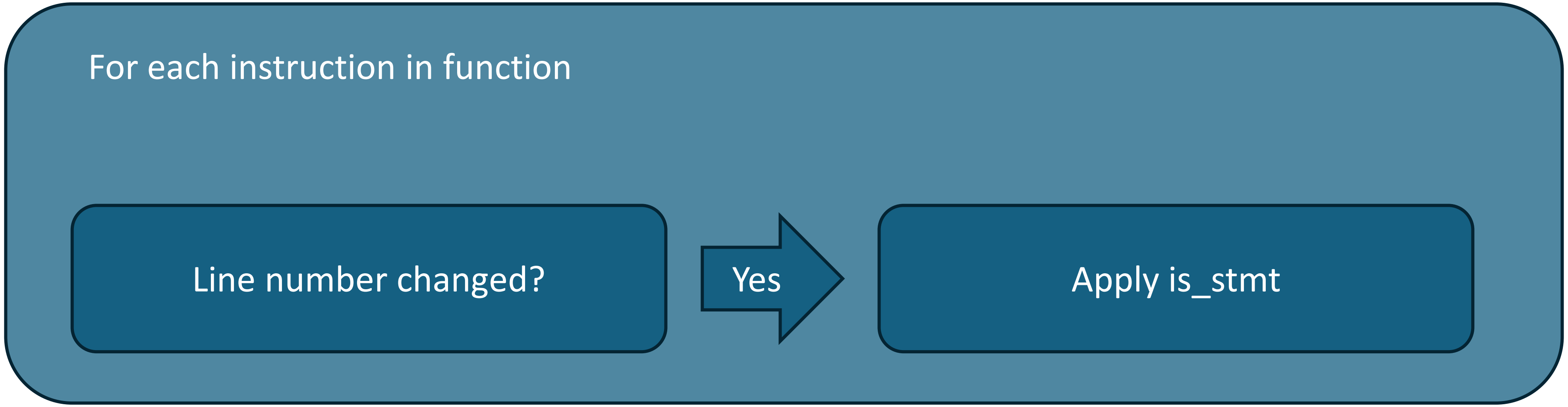
```
int g1 = 1, g2 = 2, g3 = 3;

struct point { double x, y, z; };

[[clang::optnone]]
point fun(const point& p) { return p; }

[[clang::optnone]]
void f2(double) {}

int main() {
    double y = (double)g2;
    double z = (double)g3;
    double x = (double)g1;
    point p = { x * 2, y / 3, y / 3 + z + x * 2 };
    point p2 = { p.x, p.y, p.z };
    f2(p.z);
    f2(p.x);
    fun(p2);
    p = fun(p);
    return p.x + p.y + p.z;
}
```

DWARF line table



```
Address            Line   Column File   ISA Discriminator OpIndex Flags
------------------ ------ ------ ------ --- ------------- ------- -------------
0x0000000000001130      5      0      0   0             0       0  is_stmt
0x0000000000001134      6      6      0   0             0       0  is_stmt prologue_end
0x000000000000113a      7      1      0   0             0       0  is_stmt epilogue_begin
0x0000000000001140      9      0      0   0             0       0  is_stmt
0x0000000000001144     10      6      0   0             0       0  is_stmt prologue_end
0x000000000000114e     11      1      0   0             0       0  is_stmt epilogue_begin
0x0000000000001150     13      0      0   0             0       0  is_stmt
0x0000000000001154     14      3      0   0             0       0  is_stmt prologue_end
0x0000000000001159     10      6      0   0             0       0  is_stmt
0x0000000000001170     16     12      0   0             0       0  is_stmt
0x000000000000117f     16     15      0   0             0       0
0x0000000000001181     16      3      0   0             0       0
0x0000000000001183     17      5      0   0             0       0  is_stmt
0x0000000000001188      0      5      0   0             0       0
0x000000000000118a     18      3      0   0             0       0  is_stmt
0x000000000000118c     18      3      0   0             0       0  epilogue_begin
0x000000000000118e     18      3      0   0             0       0  end_sequence
```

LLVM’s is_stmt strategy



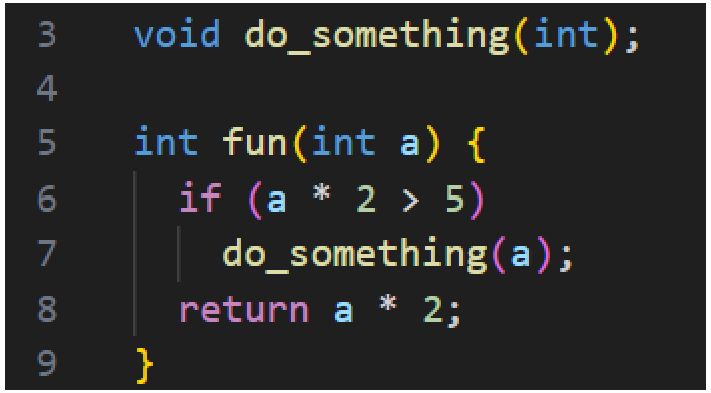
For each instruction in function

Line number changed?

Yes

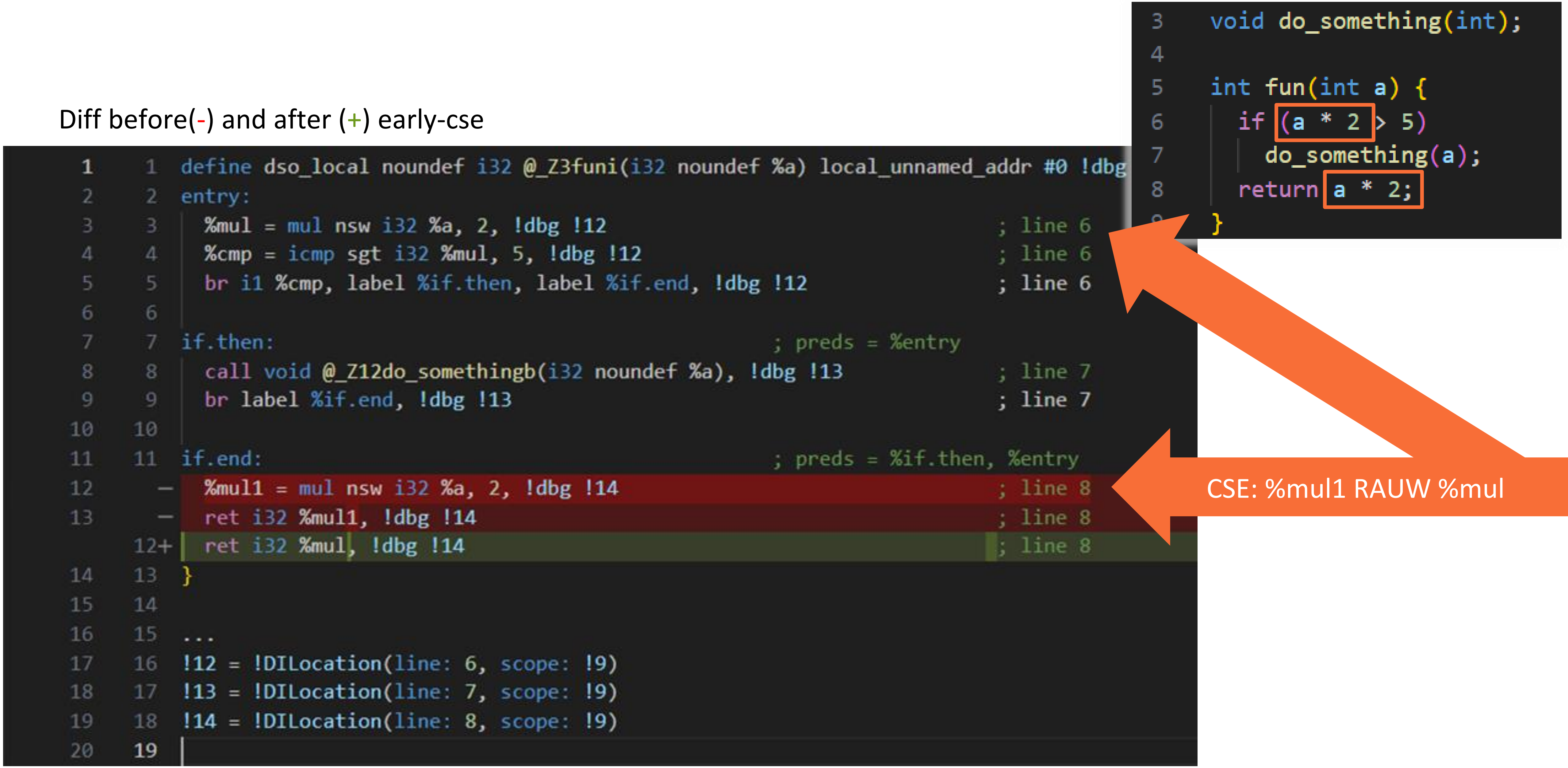
Apply is_stmt

This “works” but results in excessively jumpy stepping



```c
void do_something(int);

int fun(int a) {
  if (a * 2 > 5)
    do_something(a);
  return a * 2;
}
```

Diff before(-) and after (+) early-cse

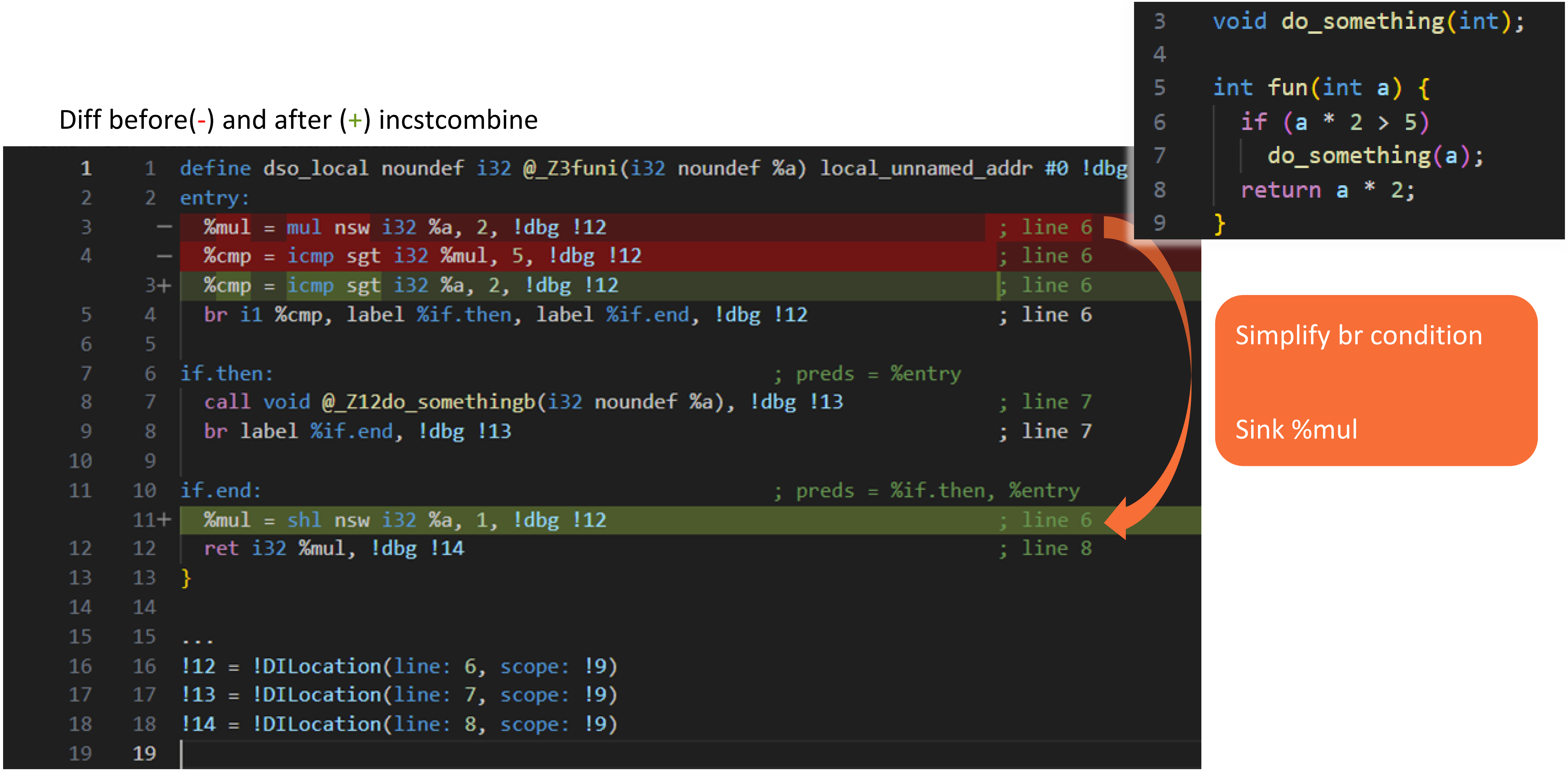


```
void do_something(int);

int fun(int a) {
  if (a * 2 > 5)
    do_something(a);
  return a * 2;
}
```

```
 define dso_local noundef i32 @_Z3funi(i32 noundef %a) local_unnamed_addr #0 !dbg
 entry:
   %mul = mul nsw i32 %a, 2, !dbg !12                                  ; line 6
   %cmp = icmp sgt i32 %mul, 5, !dbg !12                               ; line 6
   br i1 %cmp, label %if.then, label %if.end, !dbg !12                 ; line 6

 if.then:                                         ; preds = %entry
   call void @_Z12do_somethingb(i32 noundef %a), !dbg !13              ; line 7
   br label %if.end, !dbg !13                                          ; line 7

 if.end:                                          ; preds = %if.then, %entry
-  %mul1 = mul nsw i32 %a, 2, !dbg !14                                 ; line 8
-  ret i32 %mul1, !dbg !14                                             ; line 8
+  ret i32 %mul, !dbg !14                                              ; line 8
 }

 ...
 !12 = !DILocation(line: 6, scope: !9)
 !13 = !DILocation(line: 7, scope: !9)
 !14 = !DILocation(line: 8, scope: !9)
```

CSE: %mul1 RAUW %mul

Diff before(-) and after (+) incstcombine



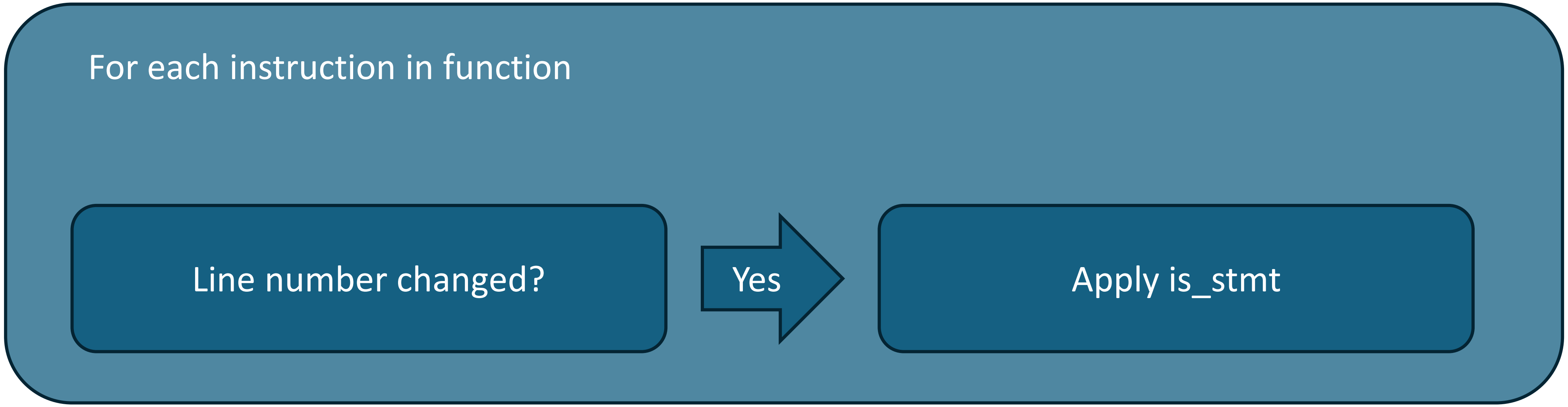
```
void do_something(int);

int fun(int a) {
  if (a * 2 > 5)
    do_something(a);
  return a * 2;
}
```

```
3  void do_something(int);
4
5  int fun(int a) {
6    if (a * 2 > 5)
7      do_something(a);
8    return a * 2;
9  }
```

```
 define dso_local noundef i32 @_Z3funi(i32 noundef %a) local_unnamed_addr #0 !dbg
 entry:
-  %mul = mul nsw i32 %a, 2, !dbg !12                                  ; line 6
-  %cmp = icmp sgt i32 %mul, 5, !dbg !12                               ; line 6
+  %cmp = icmp sgt i32 %a, 2, !dbg !12                                 ; line 6
   br i1 %cmp, label %if.then, label %if.end, !dbg !12                 ; line 6

 if.then:                                          ; preds = %entry
   call void @_Z12do_somethingb(i32 noundef %a), !dbg !13              ; line 7
   br label %if.end, !dbg !13                                          ; line 7

 if.end:                                           ; preds = %if.then, %entry
+  %mul = shl nsw i32 %a, 1, !dbg !12                                  ; line 6
   ret i32 %mul, !dbg !14                                              ; line 8
 }

 ...
 !12 = !DILocation(line: 6, scope: !9)
 !13 = !DILocation(line: 7, scope: !9)
 !14 = !DILocation(line: 8, scope: !9)
```

Simplify br condition

Sink %mul

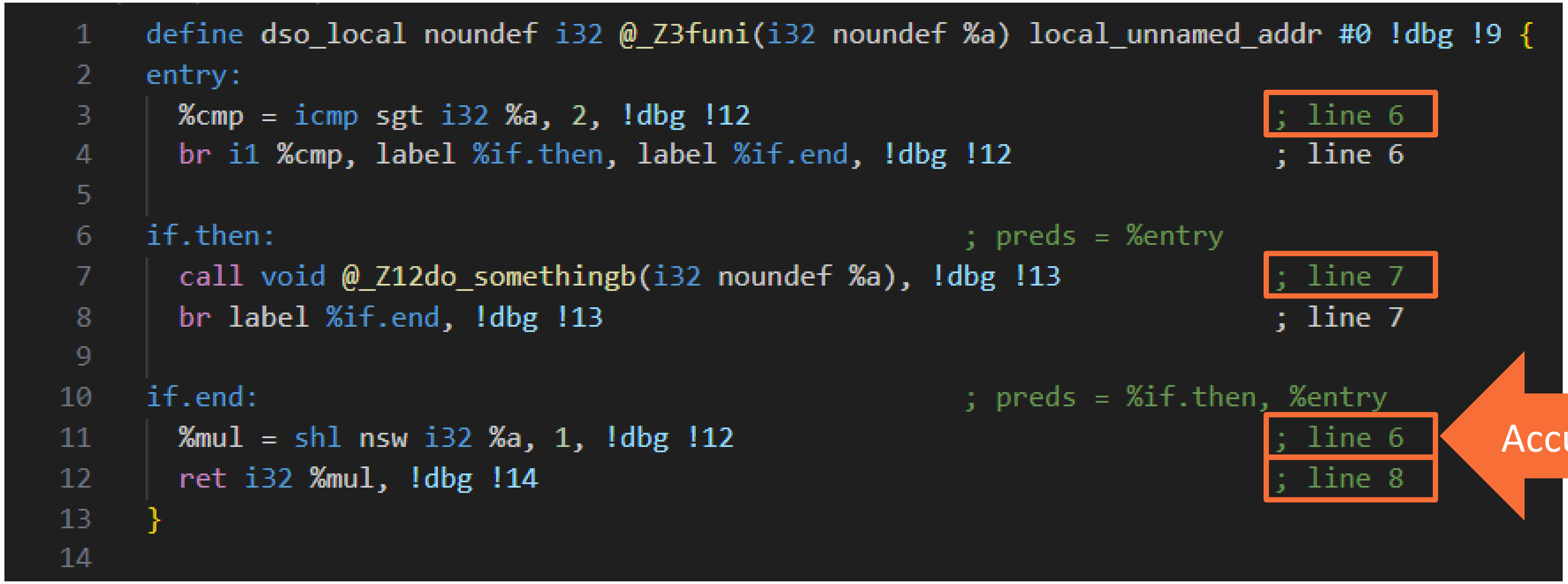


For each instruction in function

Line number changed?

Yes

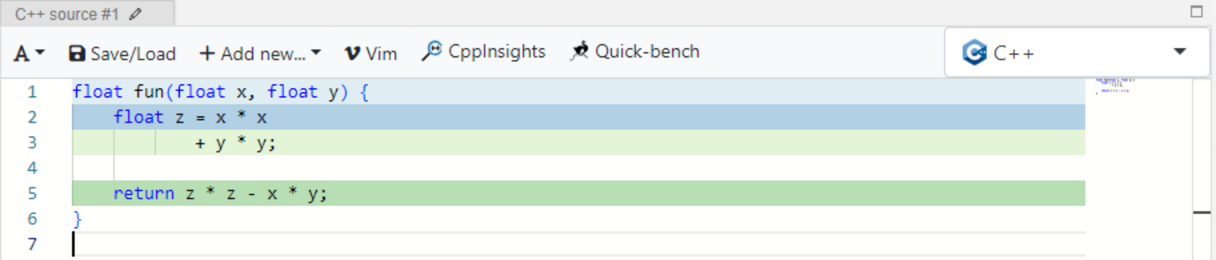
Apply is_stmt



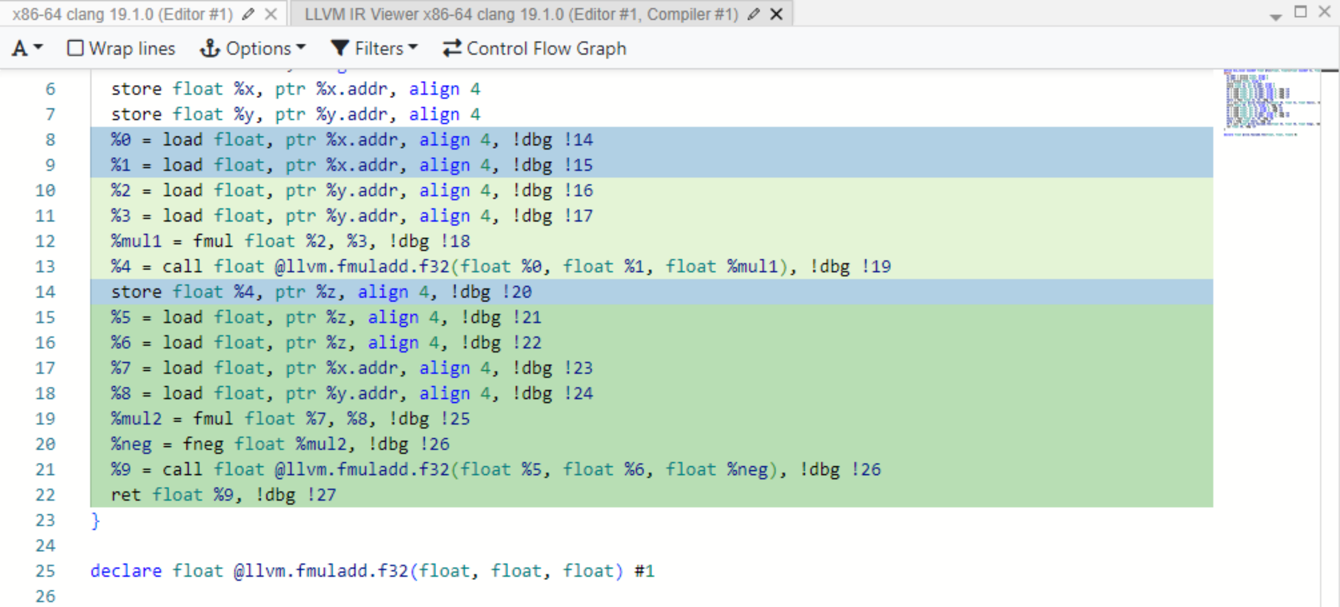
```
define dso_local noundef i32 @_Z3funi(i32 noundef %a) local_unnamed_addr #0 !dbg !9 {
entry:
  %cmp = icmp sgt i32 %a, 2, !dbg !12                               ; line 6
  br i1 %cmp, label %if.then, label %if.end, !dbg !12               ; line 6

if.then:                                         ; preds = %entry
  call void @_Z12do_somethingb(i32 noundef %a), !dbg !13            ; line 7
  br label %if.end, !dbg !13                                        ; line 7

if.end:                                          ; preds = %if.then, %entry
  %mul = shl nsw i32 %a, 1, !dbg !12                                ; line 6
  ret i32 %mul, !dbg !14                                            ; line 8
}

```

Accurate **attribution** but undesirable **step**

Solution – Key Instructions



```
float fun(float x, float y) {
    float z = x * x
            + y * y;

    return z * z - x * y;
}

```

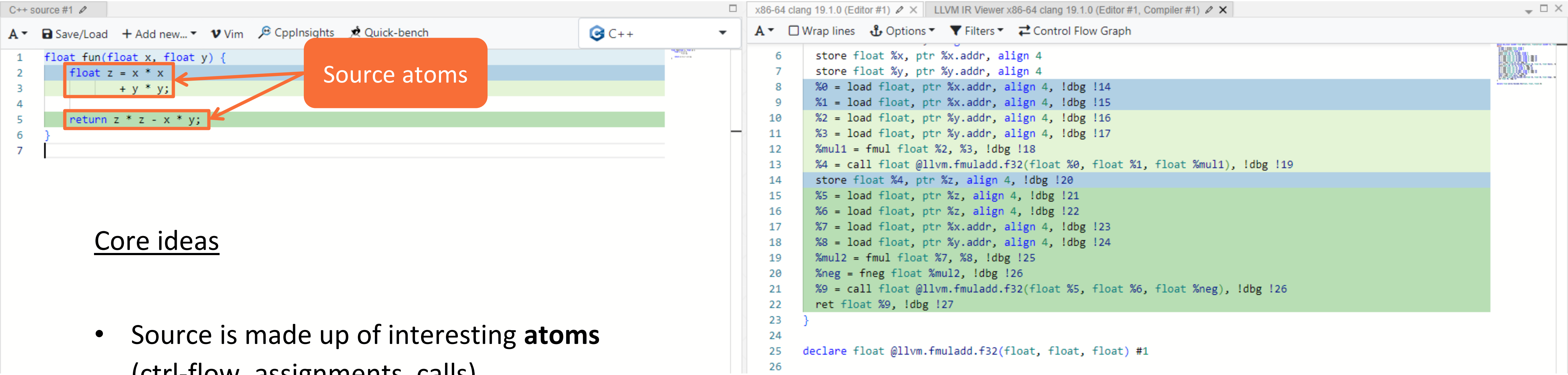

x86-64 clang 19.1.0 (Editor #1)
LLVM IR Viewer x86-64 clang 19.1.0 (Editor #1, Compiler #1)
A
Wrap lines
Options
Filters
Control Flow Graph

```
  store float %x, ptr %x.addr, align 4
  store float %y, ptr %y.addr, align 4
  %0 = load float, ptr %x.addr, align 4, !dbg !14
  %1 = load float, ptr %x.addr, align 4, !dbg !15
  %2 = load float, ptr %y.addr, align 4, !dbg !16
  %3 = load float, ptr %y.addr, align 4, !dbg !17
  %mul1 = fmul float %2, %3, !dbg !18
  %4 = call float @llvm.fmuladd.f32(float %0, float %1, float %mul1), !dbg !19
  store float %4, ptr %z, align 4, !dbg !20
  %5 = load float, ptr %z, align 4, !dbg !21
  %6 = load float, ptr %z, align 4, !dbg !22
  %7 = load float, ptr %x.addr, align 4, !dbg !23
  %8 = load float, ptr %y.addr, align 4, !dbg !24
  %mul2 = fmul float %7, %8, !dbg !25
  %neg = fneg float %mul2, !dbg !26
  %9 = call float @llvm.fmuladd.f32(float %5, float %6, float %neg), !dbg !26
  ret float %9, !dbg !27
}

declare float @llvm.fmuladd.f32(float, float, float) #1

```

Prior work + inspiration

- **Key Instructions: Solving the Code Location Problem for Optimized Code** (C. Tice, S. L. Graham, 2000)
- **Debugging Optimized Code: Concepts and Implementation on DIGITAL Alpha Systems** (R. F. Brender et al)

Key Instructions overview



Source atoms

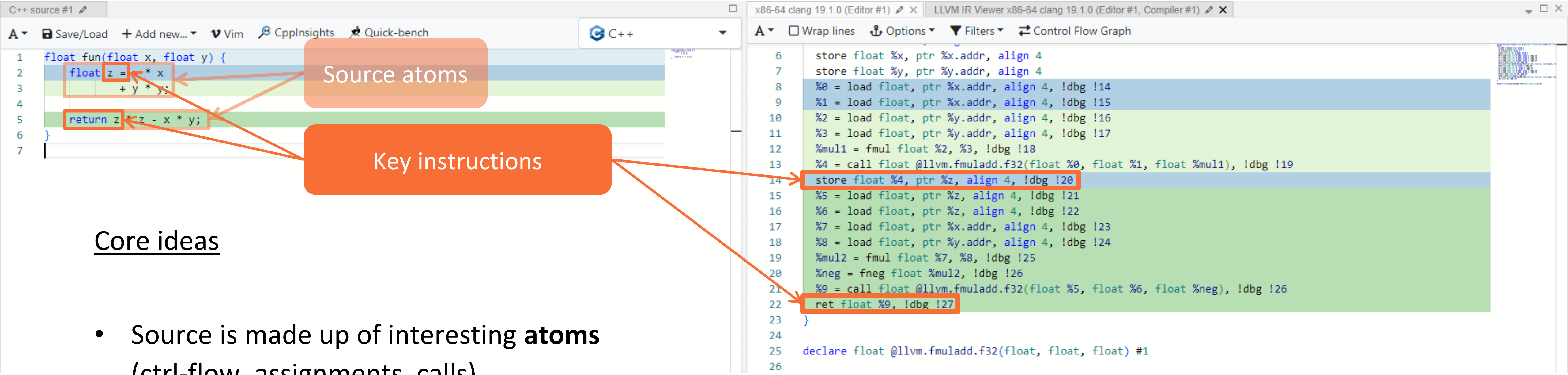
```cpp
float fun(float x, float y) {
    float z = x * x
            + y * y;

    return z * z - x * y;
}
```

```llvm
  store float %x, ptr %x.addr, align 4
  store float %y, ptr %y.addr, align 4
  %0 = load float, ptr %x.addr, align 4, !dbg !14
  %1 = load float, ptr %x.addr, align 4, !dbg !15
  %2 = load float, ptr %y.addr, align 4, !dbg !16
  %3 = load float, ptr %y.addr, align 4, !dbg !17
  %mul1 = fmul float %2, %3, !dbg !18
  %4 = call float @llvm.fmuladd.f32(float %0, float %1, float %mul1), !dbg !19
  store float %4, ptr %z, align 4, !dbg !20
  %5 = load float, ptr %z, align 4, !dbg !21
  %6 = load float, ptr %z, align 4, !dbg !22
  %7 = load float, ptr %x.addr, align 4, !dbg !23
  %8 = load float, ptr %y.addr, align 4, !dbg !24
  %mul2 = fmul float %7, %8, !dbg !25
  %neg = fneg float %mul2, !dbg !26
  %9 = call float @llvm.fmuladd.f32(float %5, float %6, float %neg), !dbg !26
  ret float %9, !dbg !27
}

declare float @llvm.fmuladd.f32(float, float, float) #1
```

Core ideas

- Source is made up of interesting **atoms** (ctrl-flow, assignments, calls)

Key Instructions overview


C++ source #1
A Save/Load + Add new... Vim CppInsights Quick-bench
C++
x86-64 clang 19.1.0 (Editor #1)
LLVM IR Viewer x86-64 clang 19.1.0 (Editor #1, Compiler #1)
A Wrap lines Options Filters Control Flow Graph
Source atoms
Key instructions

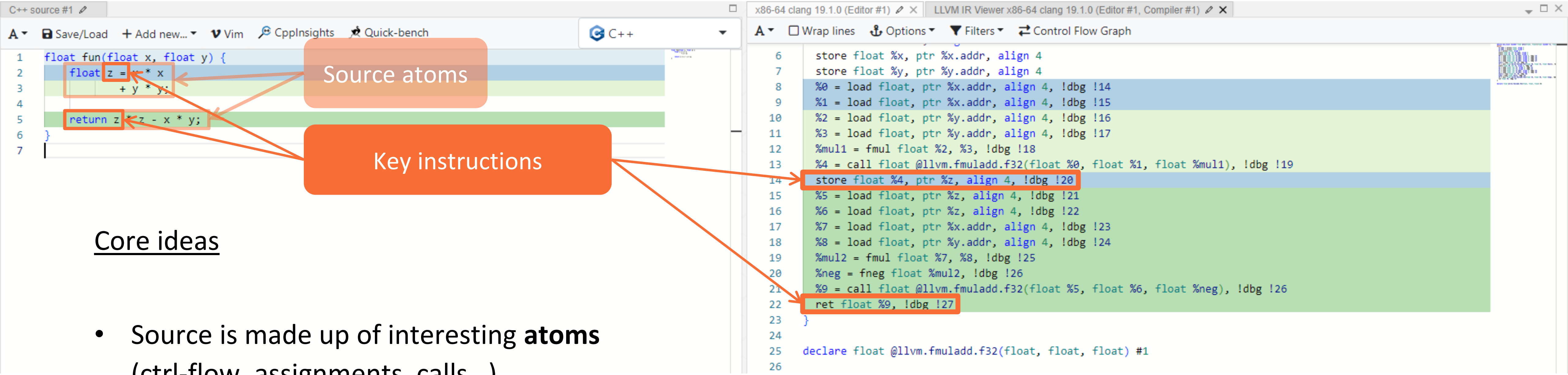
```cpp
float fun(float x, float y) {
    float z = x * x
            + y * y;

    return z * z - x * y;
}
```

```llvm
  store float %x, ptr %x.addr, align 4
  store float %y, ptr %y.addr, align 4
  %0 = load float, ptr %x.addr, align 4, !dbg !14
  %1 = load float, ptr %x.addr, align 4, !dbg !15
  %2 = load float, ptr %y.addr, align 4, !dbg !16
  %3 = load float, ptr %y.addr, align 4, !dbg !17
  %mul1 = fmul float %2, %3, !dbg !18
  %4 = call float @llvm.fmuladd.f32(float %0, float %1, float %mul1), !dbg !19
  store float %4, ptr %z, align 4, !dbg !20
  %5 = load float, ptr %z, align 4, !dbg !21
  %6 = load float, ptr %z, align 4, !dbg !22
  %7 = load float, ptr %x.addr, align 4, !dbg !23
  %8 = load float, ptr %y.addr, align 4, !dbg !24
  %mul2 = fmul float %7, %8, !dbg !25
  %neg = fneg float %mul2, !dbg !26
  %9 = call float @llvm.fmuladd.f32(float %5, float %6, float %neg), !dbg !26
  ret float %9, !dbg !27
}

declare float @llvm.fmuladd.f32(float, float, float) #1
```

Core ideas

- Source is made up of interesting **atoms** (ctrl-flow, assignments, calls)
- Atoms typically have one “**key instruction**”

Key Instructions overview



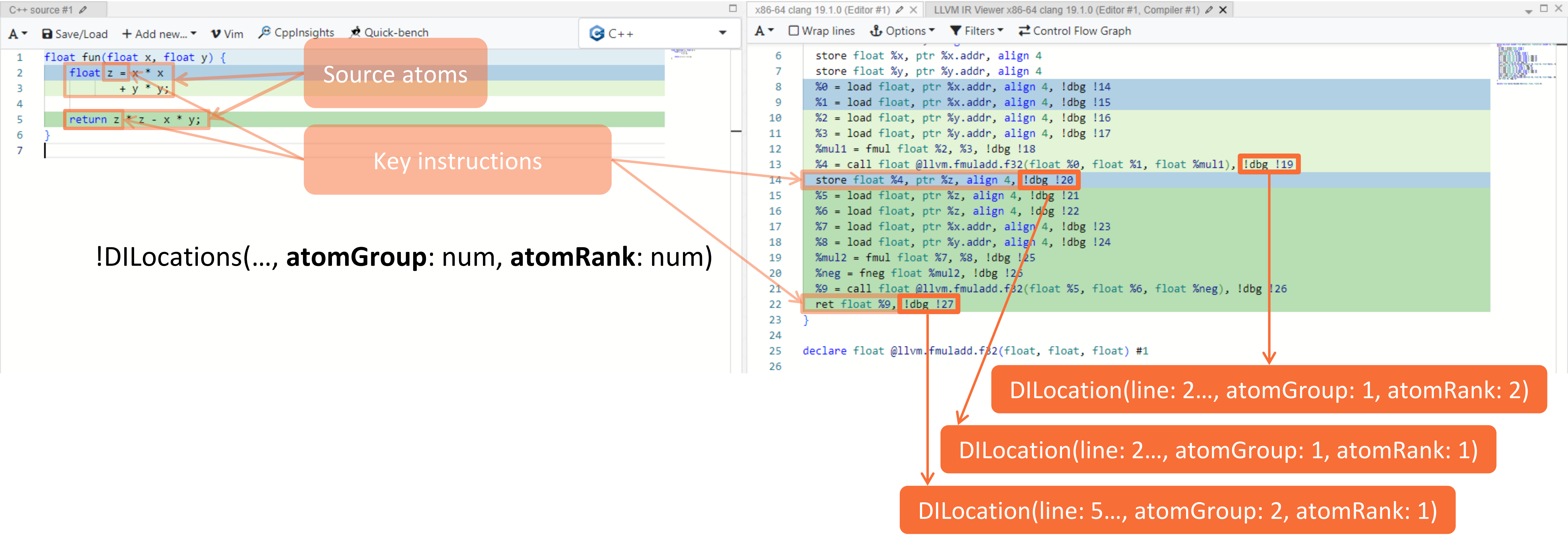
```
float fun(float x, float y) {
    float z = x * x
            + y * y;

    return z * z - x * y;
}
```

```
  store float %x, ptr %x.addr, align 4
  store float %y, ptr %y.addr, align 4
  %0 = load float, ptr %x.addr, align 4, !dbg !14
  %1 = load float, ptr %x.addr, align 4, !dbg !15
  %2 = load float, ptr %y.addr, align 4, !dbg !16
  %3 = load float, ptr %y.addr, align 4, !dbg !17
  %mul1 = fmul float %2, %3, !dbg !18
  %4 = call float @llvm.fmuladd.f32(float %0, float %1, float %mul1), !dbg !19
  store float %4, ptr %z, align 4, !dbg !20
  %5 = load float, ptr %z, align 4, !dbg !21
  %6 = load float, ptr %z, align 4, !dbg !22
  %7 = load float, ptr %x.addr, align 4, !dbg !23
  %8 = load float, ptr %y.addr, align 4, !dbg !24
  %mul2 = fmul float %7, %8, !dbg !25
  %neg = fneg float %mul2, !dbg !26
  %9 = call float @llvm.fmuladd.f32(float %5, float %6, float %neg), !dbg !26
  ret float %9, !dbg !27
}

declare float @llvm.fmuladd.f32(float, float, float) #1
```

Core ideas

- Source is made up of interesting **atoms** (ctrl-flow, assignments, calls…)
- Atoms typically have one “**key instruction**”
- Only apply **is_stmt** to key instructions

Prototype implementation



```
float fun(float x, float y) {
    float z = x * x
            + y * y;

    return z * z - x * y;
}
```

```
  store float %x, ptr %x.addr, align 4
  store float %y, ptr %y.addr, align 4
  %0 = load float, ptr %x.addr, align 4, !dbg !14
  %1 = load float, ptr %x.addr, align 4, !dbg !15
  %2 = load float, ptr %y.addr, align 4, !dbg !16
  %3 = load float, ptr %y.addr, align 4, !dbg !17
  %mul1 = fmul float %2, %3, !dbg !18
  %4 = call float @llvm.fmuladd.f32(float %0, float %1, float %mul1), !dbg !19
  store float %4, ptr %z, align 4, !dbg !20
  %5 = load float, ptr %z, align 4, !dbg !21
  %6 = load float, ptr %z, align 4, !dbg !22
  %7 = load float, ptr %x.addr, align 4, !dbg !23
  %8 = load float, ptr %y.addr, align 4, !dbg !24
  %mul2 = fmul float %7, %8, !dbg !25
  %neg = fneg float %mul2, !dbg !26
  %9 = call float @llvm.fmuladd.f32(float %5, float %6, float %neg), !dbg !26
  ret float %9, !dbg !27
}

declare float @llvm.fmuladd.f32(float, float, float) #1
```

Source atoms

Key instructions

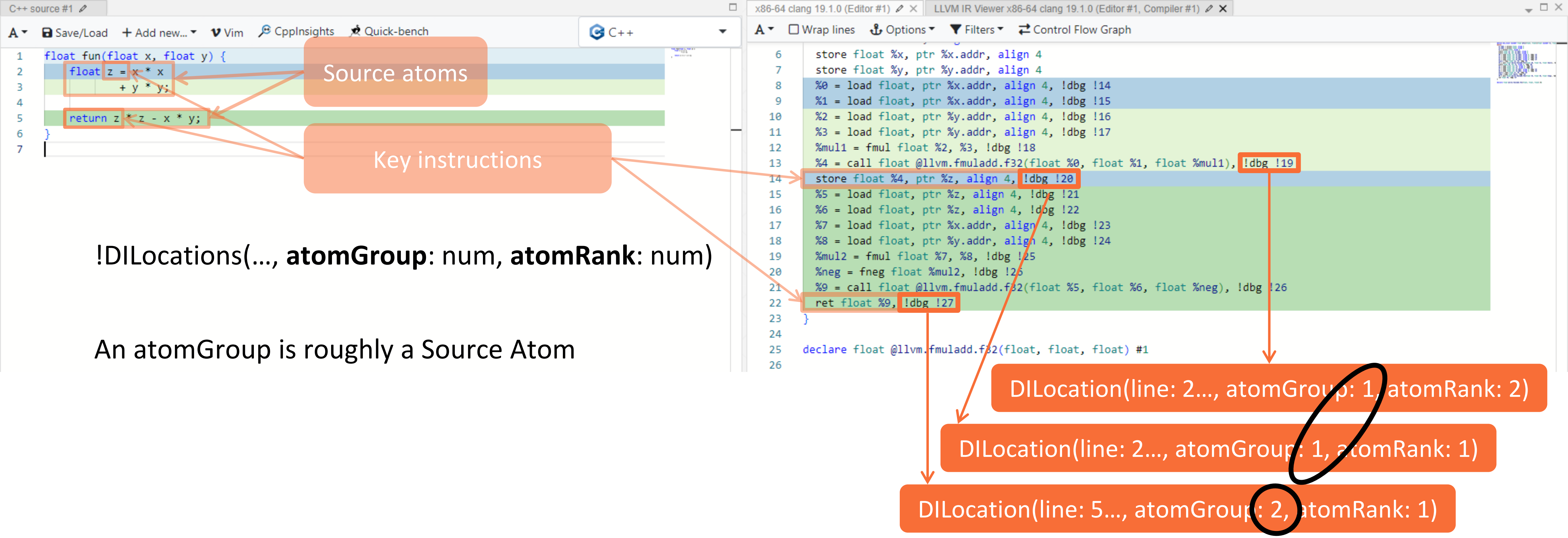
!DILocations(…, **atomGroup**: num, **atomRank**: num)

DILocation(line: 2…, atomGroup: 1, atomRank: 2)

DILocation(line: 2…, atomGroup: 1, atomRank: 1)

DILocation(line: 5…, atomGroup: 2, atomRank: 1)

Prototype implementation



```
float fun(float x, float y) {
    float z = x * x
            + y * y;

    return z * z - x * y;
}
```

```
  store float %x, ptr %x.addr, align 4
  store float %y, ptr %y.addr, align 4
  %0 = load float, ptr %x.addr, align 4, !dbg !14
  %1 = load float, ptr %x.addr, align 4, !dbg !15
  %2 = load float, ptr %y.addr, align 4, !dbg !16
  %3 = load float, ptr %y.addr, align 4, !dbg !17
  %mul1 = fmul float %2, %3, !dbg !18
  %4 = call float @llvm.fmuladd.f32(float %0, float %1, float %mul1), !dbg !19
  store float %4, ptr %z, align 4, !dbg !20
  %5 = load float, ptr %z, align 4, !dbg !21
  %6 = load float, ptr %z, align 4, !dbg !22
  %7 = load float, ptr %x.addr, align 4, !dbg !23
  %8 = load float, ptr %y.addr, align 4, !dbg !24
  %mul2 = fmul float %7, %8, !dbg !25
  %neg = fneg float %mul2, !dbg !26
  %9 = call float @llvm.fmuladd.f32(float %5, float %6, float %neg), !dbg !26
  ret float %9, !dbg !27
}

declare float @llvm.fmuladd.f32(float, float, float) #1
```

!DILocations(…, **atomGroup**: num, **atomRank**: num)

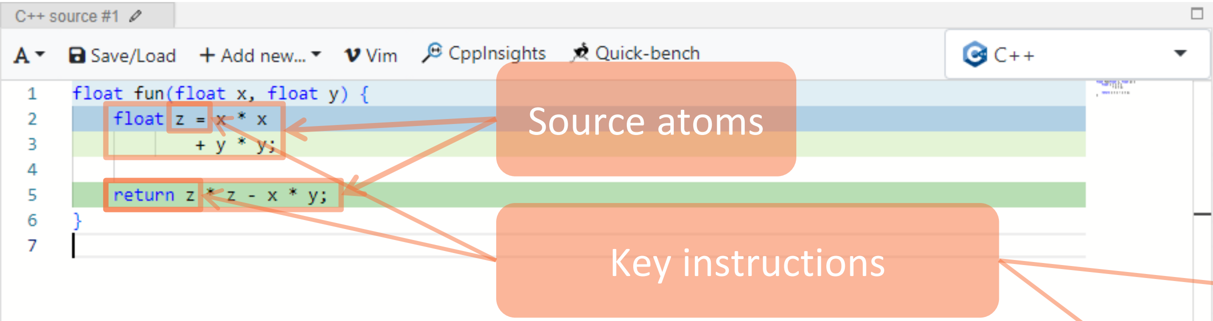
An atomGroup is roughly a Source Atom

DILocation(line: 2…, atomGroup: 1, atomRank: 2)

DILocation(line: 2…, atomGroup: 1, atomRank: 1)

DILocation(line: 5…, atomGroup: 2, atomRank: 1)

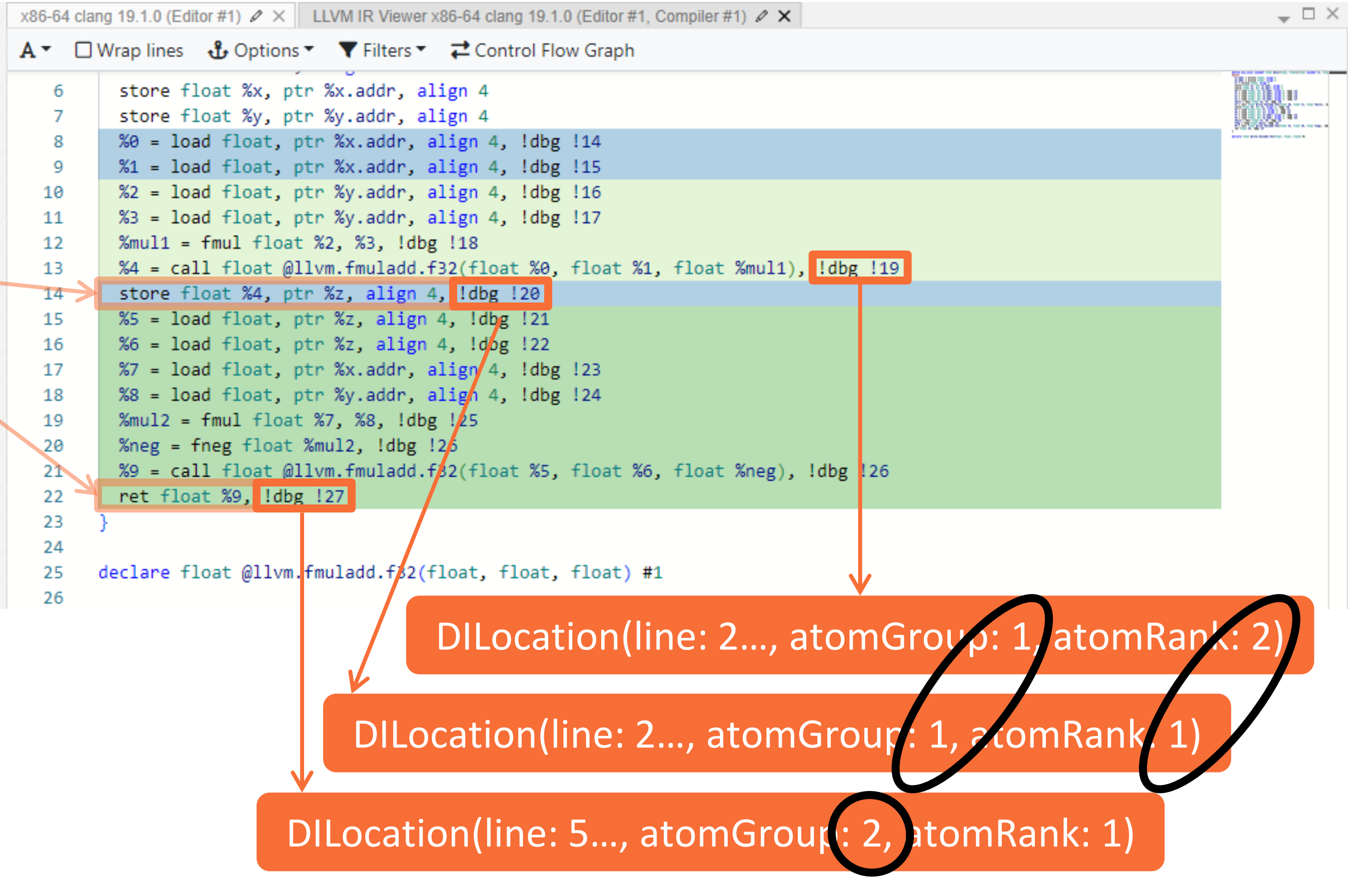
Prototype implementation



```
float fun(float x, float y) {
    float z = x * x
            + y * y;

    return z * z - x * y;
}
```

Source atoms

Key instructions


x86-64 clang 19.1.0 (Editor #1)
LLVM IR Viewer x86-64 clang 19.1.0 (Editor #1, Compiler #1)
A
Wrap lines
Options
Filters
Control Flow Graph

```
  store float %x, ptr %x.addr, align 4
  store float %y, ptr %y.addr, align 4
  %0 = load float, ptr %x.addr, align 4, !dbg !14
  %1 = load float, ptr %x.addr, align 4, !dbg !15
  %2 = load float, ptr %y.addr, align 4, !dbg !16
  %3 = load float, ptr %y.addr, align 4, !dbg !17
  %mul1 = fmul float %2, %3, !dbg !18
  %4 = call float @llvm.fmuladd.f32(float %0, float %1, float %mul1), !dbg !19
  store float %4, ptr %z, align 4, !dbg !20
  %5 = load float, ptr %z, align 4, !dbg !21
  %6 = load float, ptr %z, align 4, !dbg !22
  %7 = load float, ptr %x.addr, align 4, !dbg !23
  %8 = load float, ptr %y.addr, align 4, !dbg !24
  %mul2 = fmul float %7, %8, !dbg !25
  %neg = fneg float %mul2, !dbg !26
  %9 = call float @llvm.fmuladd.f32(float %5, float %6, float %neg), !dbg !26
  ret float %9, !dbg !27
}

declare float @llvm.fmuladd.f32(float, float, float) #1
```

DILocation(line: 2..., atomGroup: 1, atomRank: 2)

DILocation(line: 2..., atomGroup: 1, atomRank: 1)

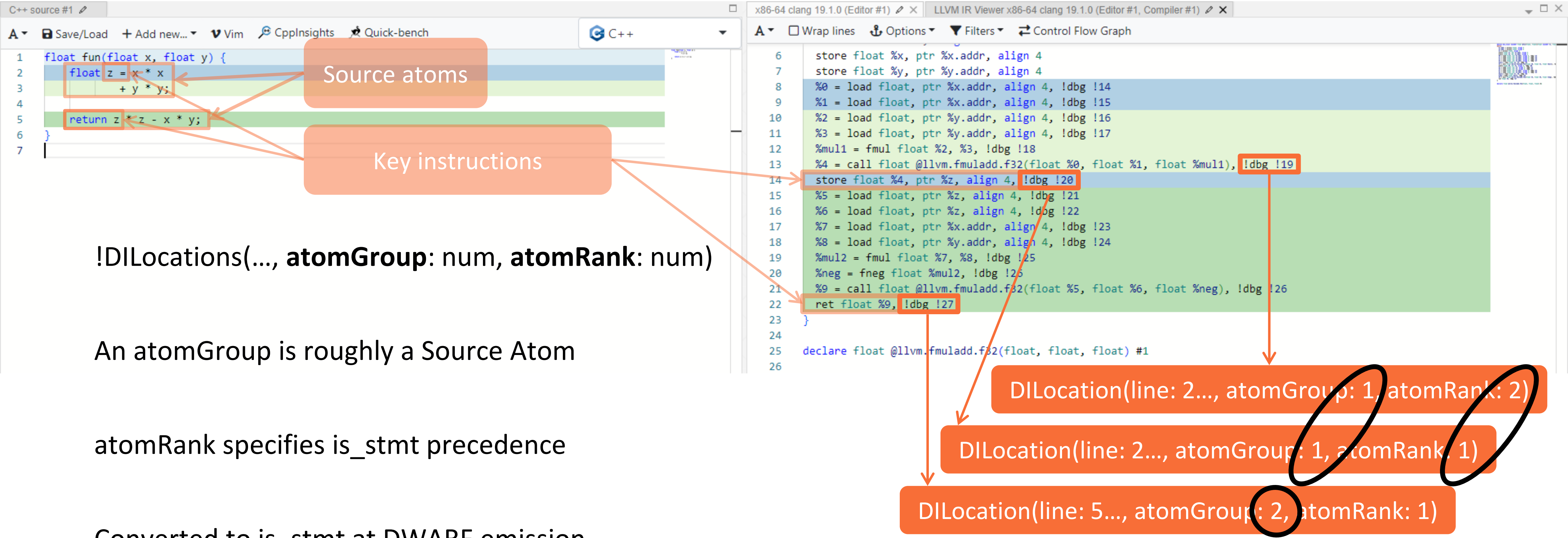
DILocation(line: 5..., atomGroup: 2, atomRank: 1)

!DILocations(..., **atomGroup**: num, **atomRank**: num)

An atomGroup is roughly a Source Atom

atomRank specifies is_stmt precedence

Prototype implementation



```
float fun(float x, float y) {
  float z = x * x
          + y * y;

  return z * z - x * y;
}
```

```
  store float %x, ptr %x.addr, align 4
  store float %y, ptr %y.addr, align 4
  %0 = load float, ptr %x.addr, align 4, !dbg !14
  %1 = load float, ptr %x.addr, align 4, !dbg !15
  %2 = load float, ptr %y.addr, align 4, !dbg !16
  %3 = load float, ptr %y.addr, align 4, !dbg !17
  %mul1 = fmul float %2, %3, !dbg !18
  %4 = call float @llvm.fmuladd.f32(float %0, float %1, float %mul1), !dbg !19
  store float %4, ptr %z, align 4, !dbg !20
  %5 = load float, ptr %z, align 4, !dbg !21
  %6 = load float, ptr %z, align 4, !dbg !22
  %7 = load float, ptr %x.addr, align 4, !dbg !23
  %8 = load float, ptr %y.addr, align 4, !dbg !24
  %mul2 = fmul float %7, %8, !dbg !25
  %neg = fneg float %mul2, !dbg !26
  %9 = call float @llvm.fmuladd.f32(float %5, float %6, float %neg), !dbg !26
  ret float %9, !dbg !27
}

declare float @llvm.fmuladd.f32(float, float, float) #1
```

DILocation(line: 2..., atomGroup: 1, atomRank: 2)

DILocation(line: 2..., atomGroup: 1, atomRank: 1)

DILocation(line: 5..., atomGroup: 2, atomRank: 1)

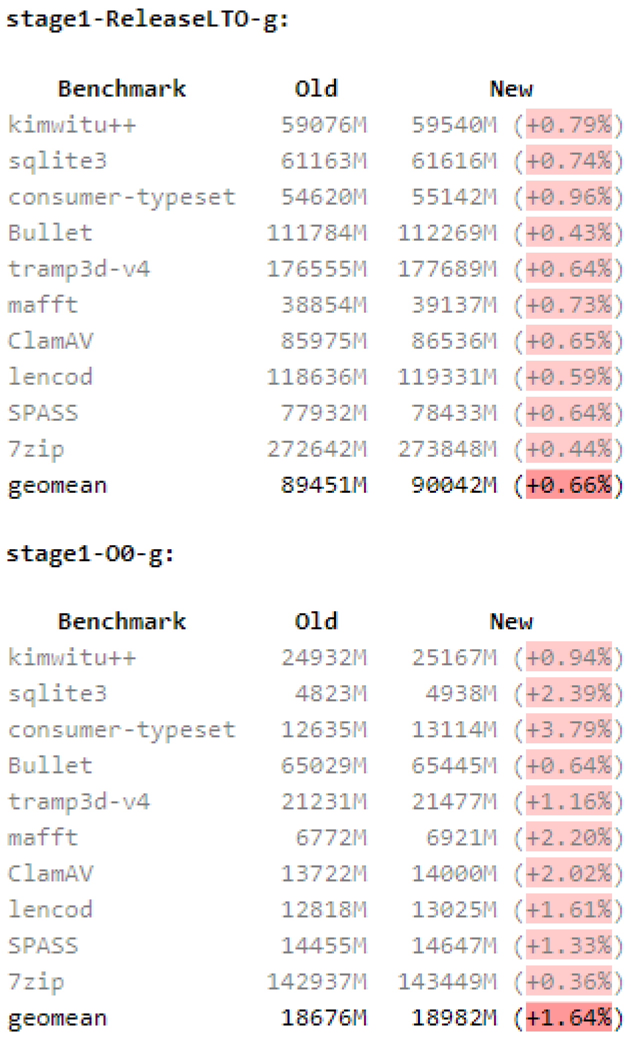
!DILocations(..., **atomGroup**: num, **atomRank**: num)

An atomGroup is roughly a Source Atom

atomRank specifies is_stmt precedence

Converted to is_stmt at DWARF emission

Costs (prototype)



```
stage1-ReleaseLTO-g:

      Benchmark           Old           New
kimwitu++               59076M    59540M (+0.79%)
sqlite3                 61163M    61616M (+0.74%)
consumer-typeset        54620M    55142M (+0.96%)
Bullet                 111784M   112269M (+0.43%)
tramp3d-v4             176555M   177689M (+0.64%)
mafft                   38854M    39137M (+0.73%)
ClamAV                  85975M    86536M (+0.65%)
lencod                 118636M   119331M (+0.59%)
SPASS                   77932M    78433M (+0.64%)
7zip                   272642M   273848M (+0.44%)
geomean                 89451M    90042M (+0.66%)

stage1-O0-g:

      Benchmark           Old           New
kimwitu++               24932M    25167M (+0.94%)
sqlite3                  4823M     4938M (+2.39%)
consumer-typeset        12635M    13114M (+3.79%)
Bullet                  65029M    65445M (+0.64%)
tramp3d-v4              21231M    21477M (+1.16%)
mafft                    6772M     6921M (+2.20%)
ClamAV                  13722M    14000M (+2.02%)
lencod                  12818M    13025M (+1.61%)
SPASS                   14455M    14647M (+1.33%)
7zip                   142937M   143449M (+0.36%)
geomean                 18676M    18982M (+1.64%)
```

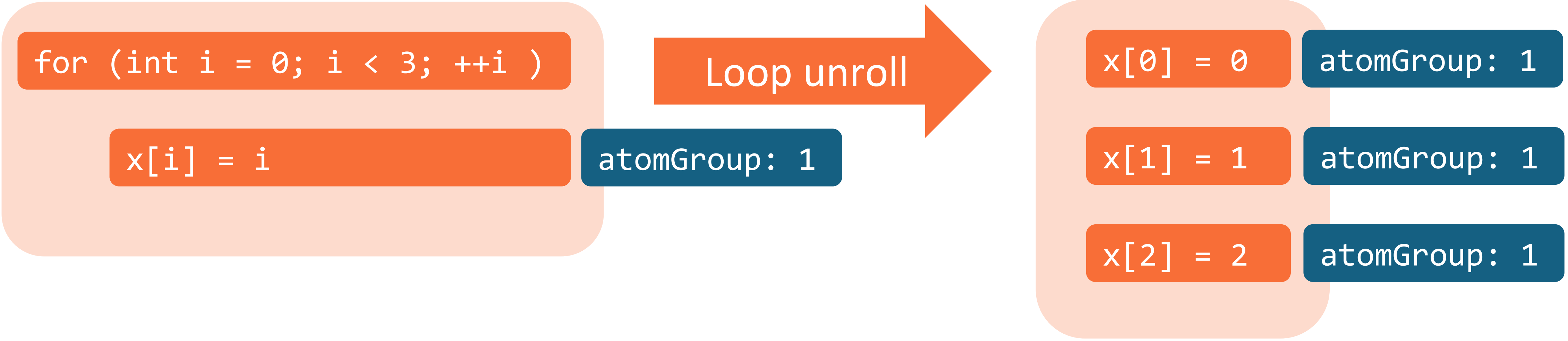
Prototype compile-time cost:

- *compile-time-tracker, CTMark, instructions:u*
- +0.66% to LTO builds
- +1.64% to unoptimized builds

Dev-time cost:

- More info for some optimisations

Costs - middle end work



```
for (int i = 0; i < 3; ++i )
    x[i] = i
```

atomGroup: 1

Loop unroll

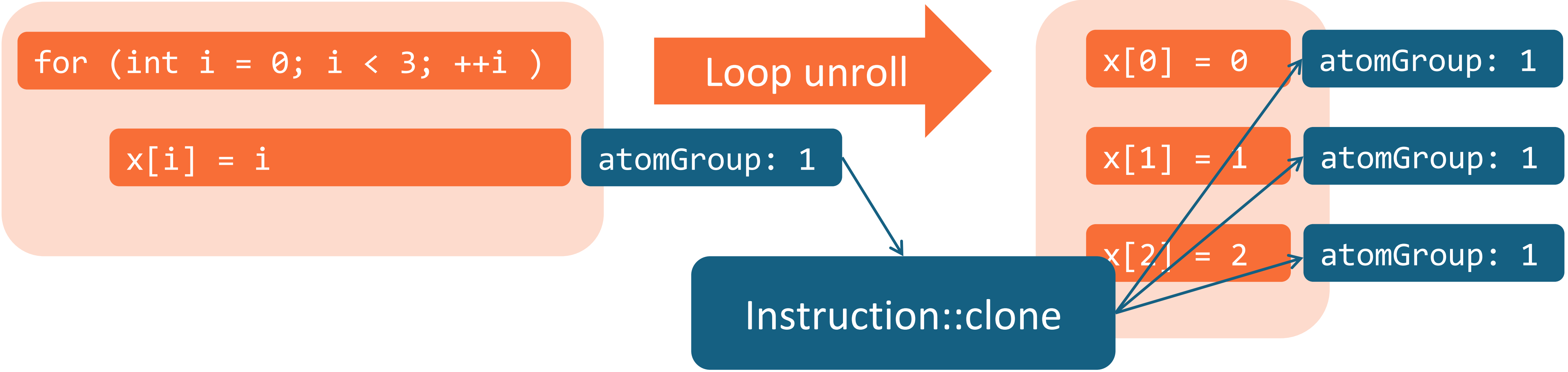
```
x[0] = 0
x[1] = 1
x[2] = 2
```

atomGroup: 1

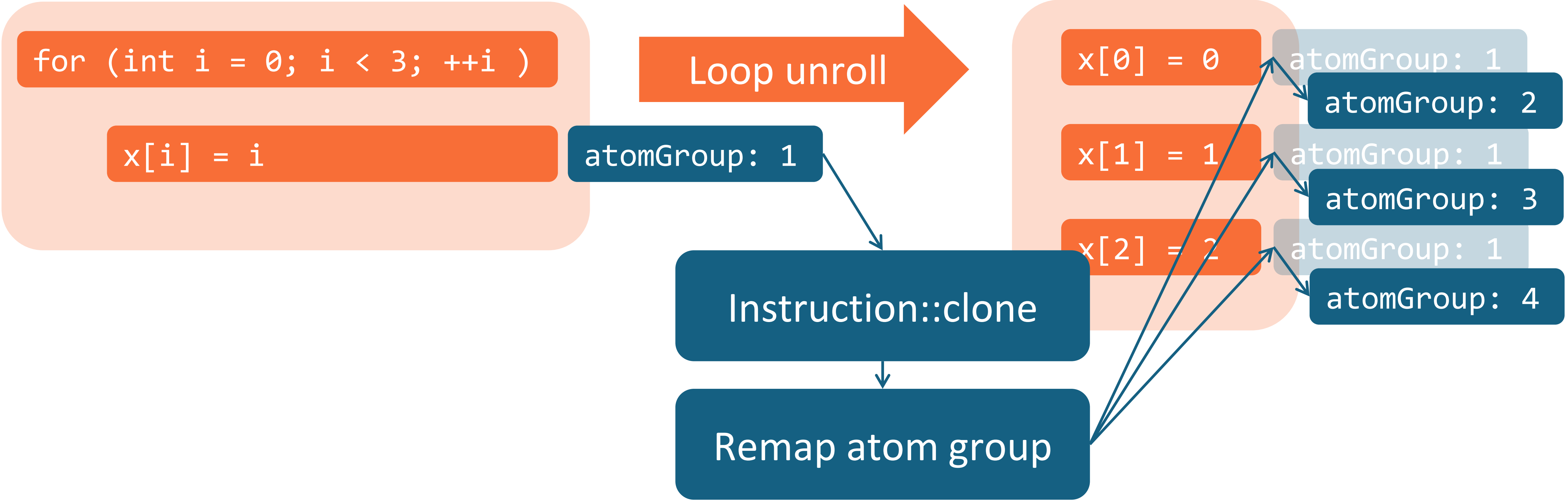
atomGroup: 1

atomGroup: 1

Costs - middle end work

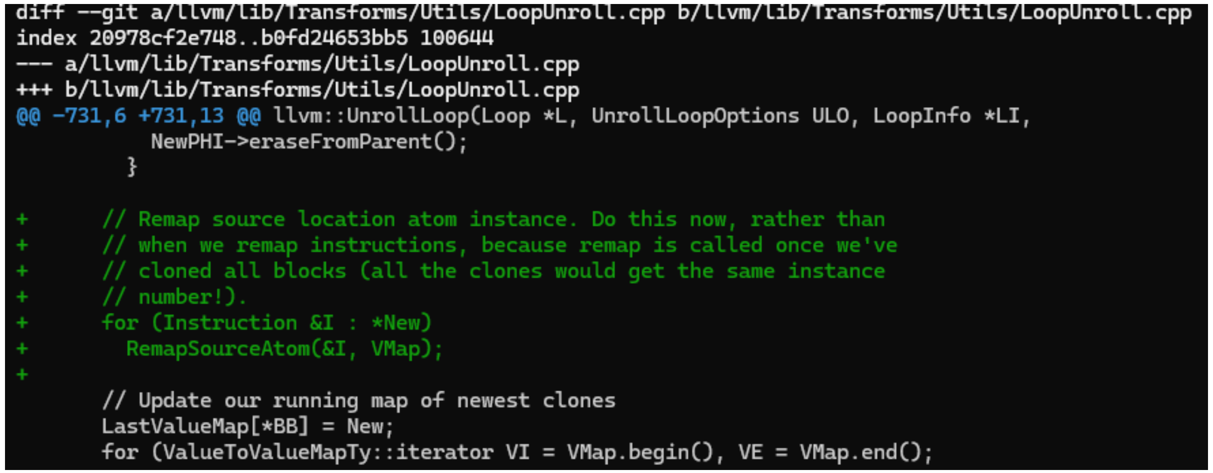


Remap atoms for duplicated control flow



```
for (int i = 0; i < 3; ++i )
    x[i] = i
```

Remap atoms for duplicated control flow



```diff
diff --git a/llvm/lib/Transforms/Utils/LoopUnroll.cpp b/llvm/lib/Transforms/Utils/LoopUnroll.cpp
index 20978cf2e748..b0fd24653bb5 100644
--- a/llvm/lib/Transforms/Utils/LoopUnroll.cpp
+++ b/llvm/lib/Transforms/Utils/LoopUnroll.cpp
@@ -731,6 +731,13 @@ llvm::UnrollLoop(Loop *L, UnrollLoopOptions ULO, LoopInfo *LI,
           NewPHI->eraseFromParent();
         }

+      // Remap source location atom instance. Do this now, rather than
+      // when we remap instructions, because remap is called once we've
+      // cloned all blocks (all the clones would get the same instance
+      // number!).
+      for (Instruction &I : *New)
+        RemapSourceAtom(&I, VMap);
+
       // Update our running map of newest clones
       LastValueMap[*BB] = New;
       for (ValueToValueMapTy::iterator VI = VMap.begin(), VE = VMap.end();
```

llvm/lib/Transforms/IPO/IROutliner.cpp

llvm/lib/Transforms/Scalar/JumpThreading.cpp

llvm/lib/Transforms/Scalar/SimpleLoopUnswitch.cpp

llvm/lib/Transforms/Utils/BreakCriticalEdges.cpp

llvm/lib/Transforms/Utils/CloneFunction.cpp

llvm/lib/Transforms/Utils/InlineFunction.cpp

llvm/lib/Transforms/Utils/LoopRotationUtils.cpp

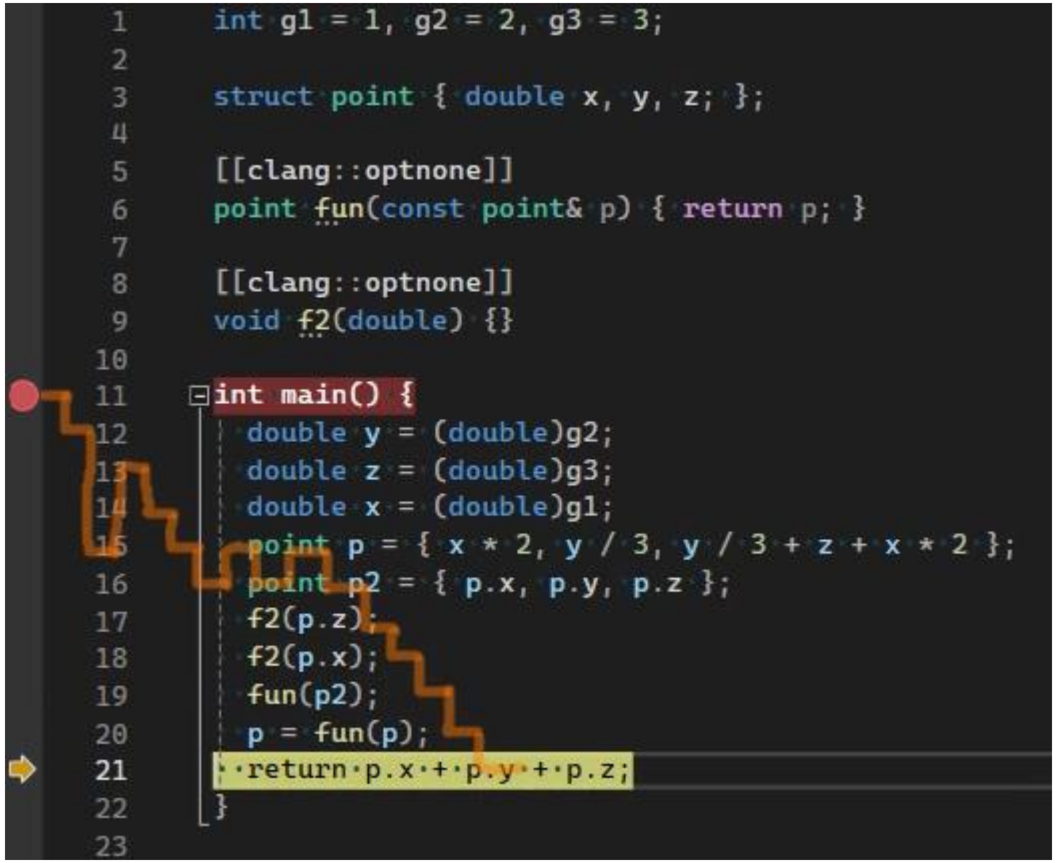
llvm/lib/Transforms/Utils/LoopUnroll.cpp

llvm/lib/Transforms/Utils/SimplifyCFG.cpp

Risks / evaluation

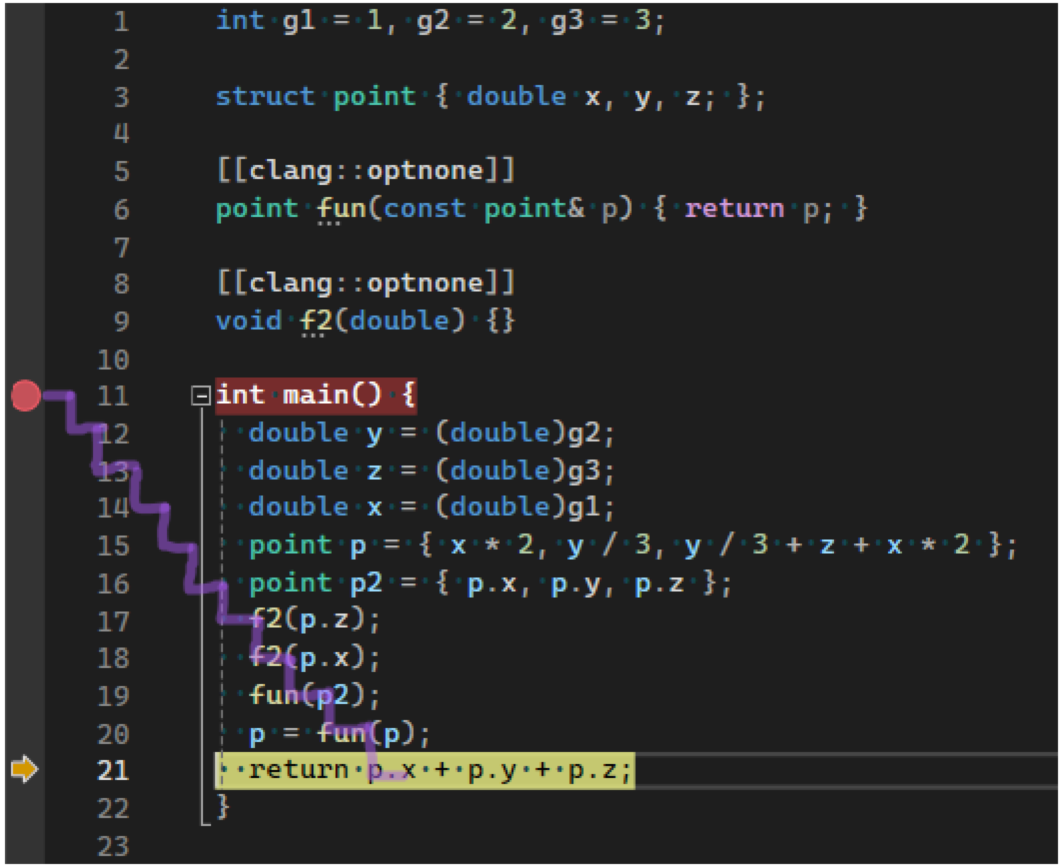
Risks / evaluation

-O2 -g

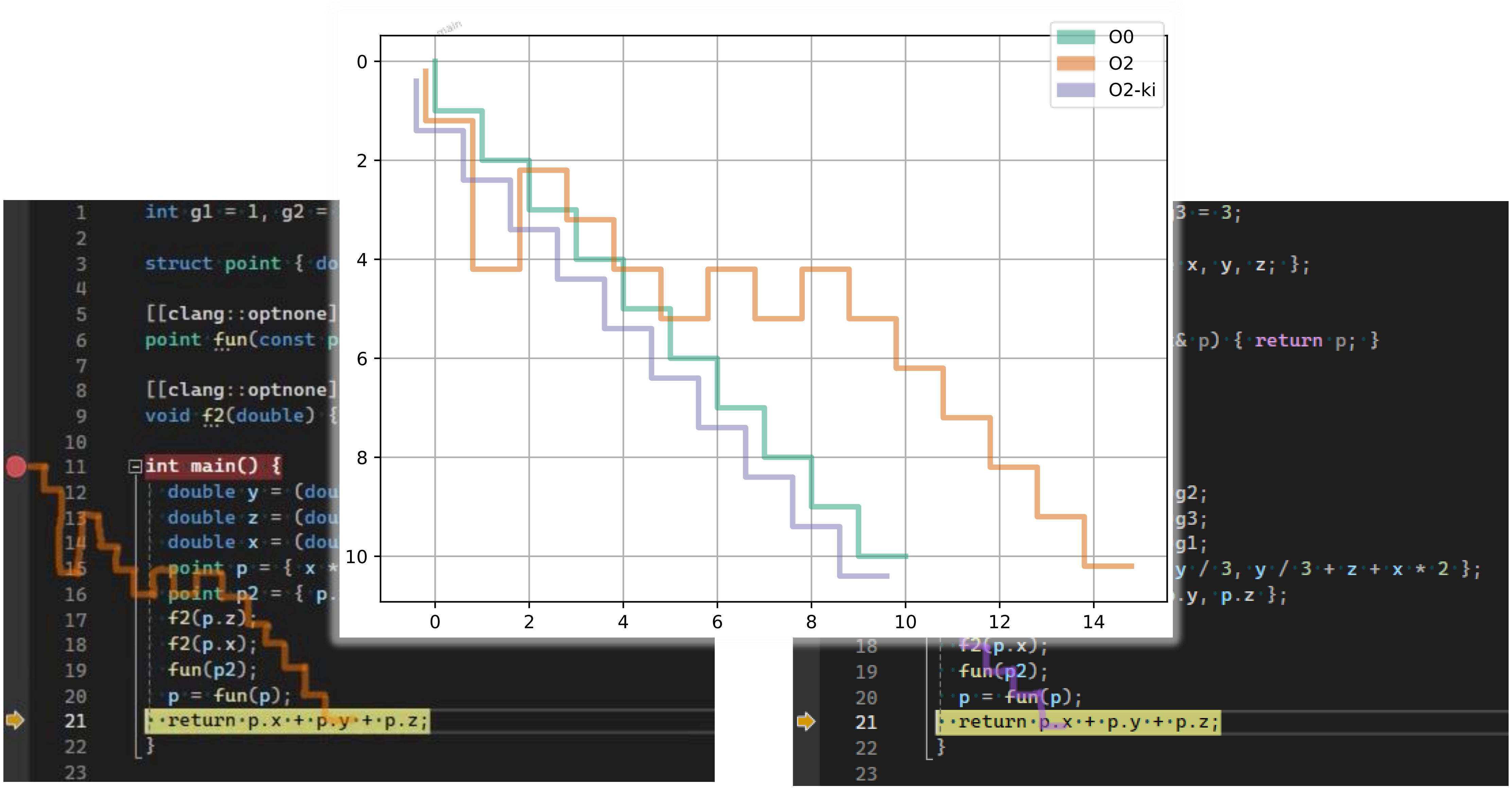


```
int g1 = 1, g2 = 2, g3 = 3;

struct point { double x, y, z; };

[[clang::optnone]]
point fun(const point& p) { return p; }

[[clang::optnone]]
void f2(double) {}

int main() {
  double y = (double)g2;
  double z = (double)g3;
  double x = (double)g1;
  point p = { x * 2, y / 3, y / 3 + z + x * 2 };
  point p2 = { p.x, p.y, p.z };
  f2(p.z);
  f2(p.x);
  fun(p2);
  p = fun(p);
  return p.x + p.y + p.z;
}
```

-O2 –g –fkey-instructions



```
int g1 = 1, g2 = 2, g3 = 3;

struct point { double x, y, z; };

[[clang::optnone]]
point fun(const point& p) { return p; }

[[clang::optnone]]
void f2(double) {}

int main() {
  double y = (double)g2;
  double z = (double)g3;
  double x = (double)g1;
  point p = { x * 2, y / 3, y / 3 + z + x * 2 };
  point p2 = { p.x, p.y, p.z };
  f2(p.z);
  f2(p.x);
  fun(p2);
  p = fun(p);
  return p.x + p.y + p.z;
}
```

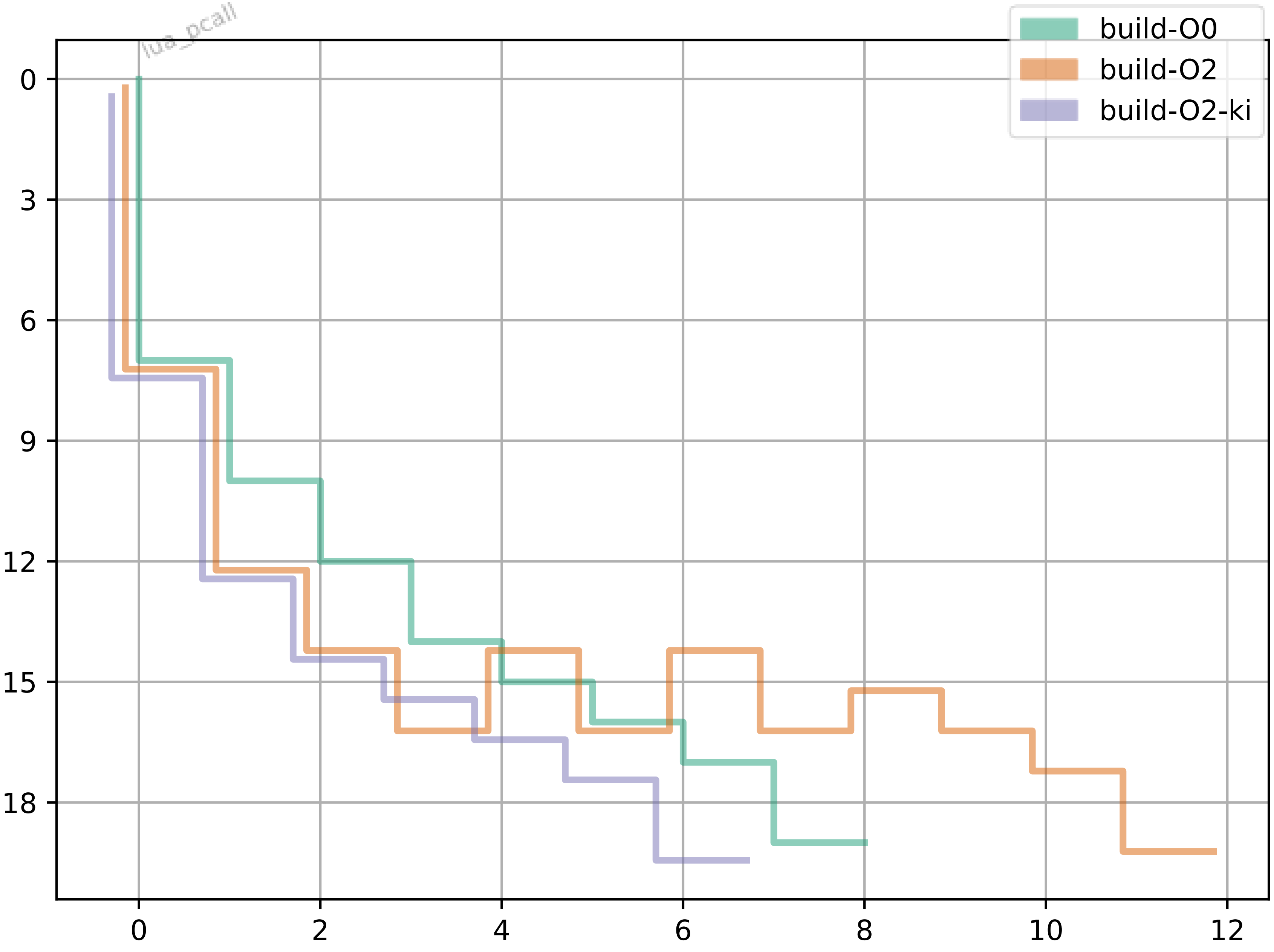


```
int g1 = 1, g2 =

struct point { do

[[clang::optnone]
point fun(const p

[[clang::optnone]
void f2(double) {

int main() {
  double y = (dou
  double z = (dou
  double x = (dou
  point p = { x *
  point p2 = { p.
  f2(p.z);
  f2(p.x);
  fun(p2);
  p = fun(p);
  return p.x + p.y + p.z;
}
```

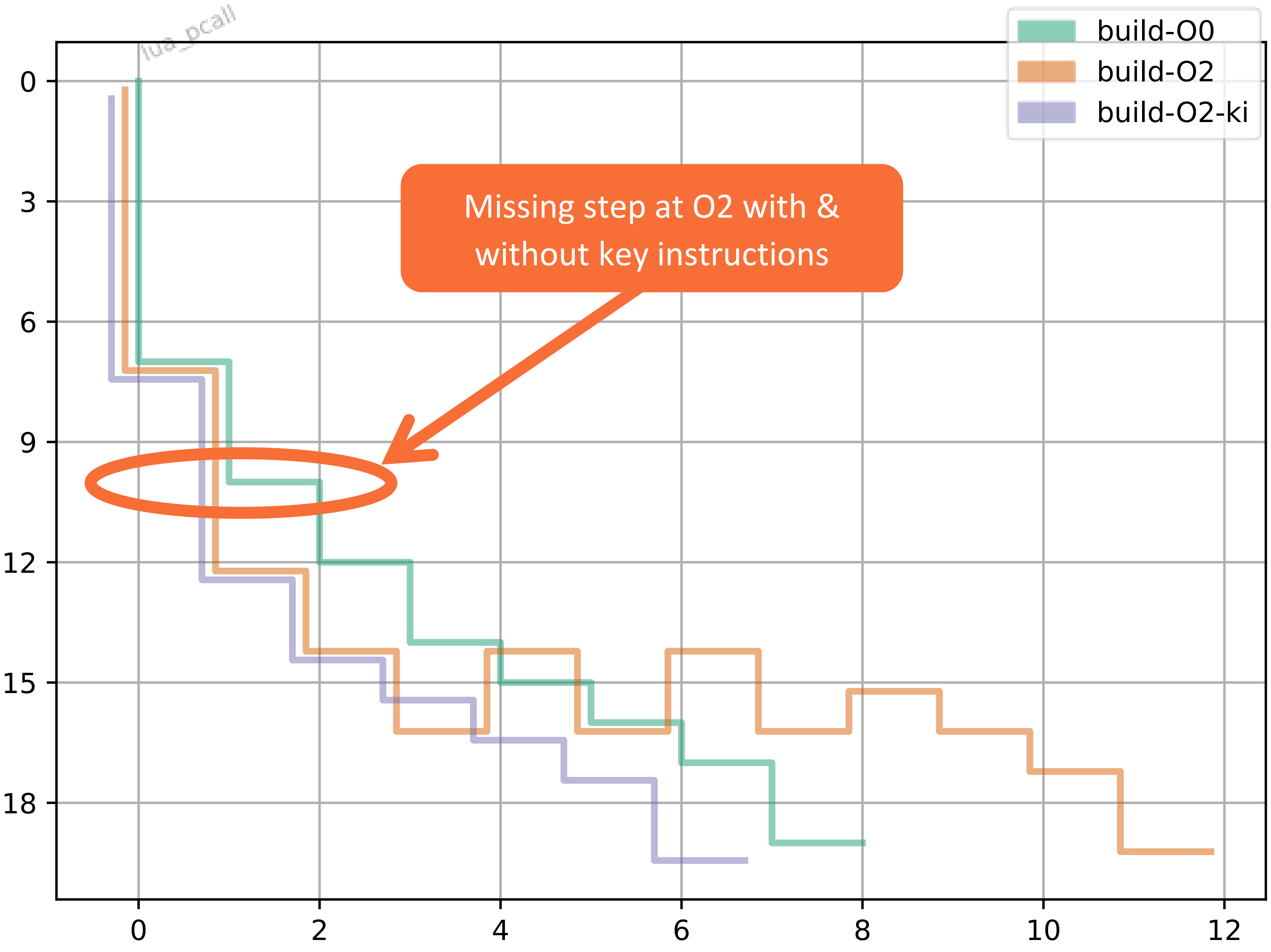
```
3 = 3;

x, y, z; };

& p) { return p; }

g2;
g3;
g1;
y / 3, y / 3 + z + x * 2 };
.y, p.z };
```

```
  f2(p.x);
  fun(p2);
  p = fun(p);
  return p.x + p.y + p.z;
}
```

A debugging trace from Lua (bytecode interpreter, C)

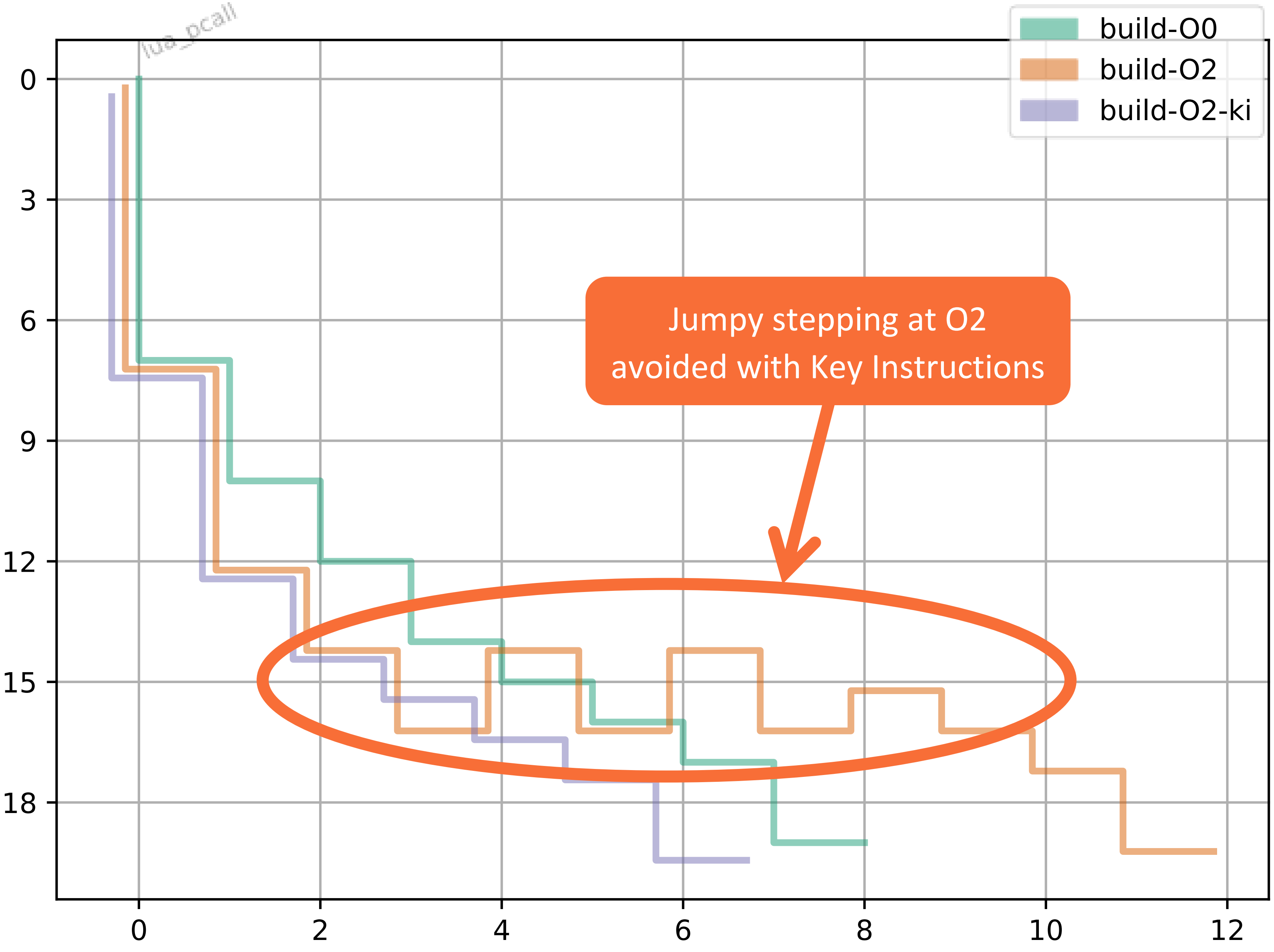


A debugging trace from Lua (bytecode interpreter, C)



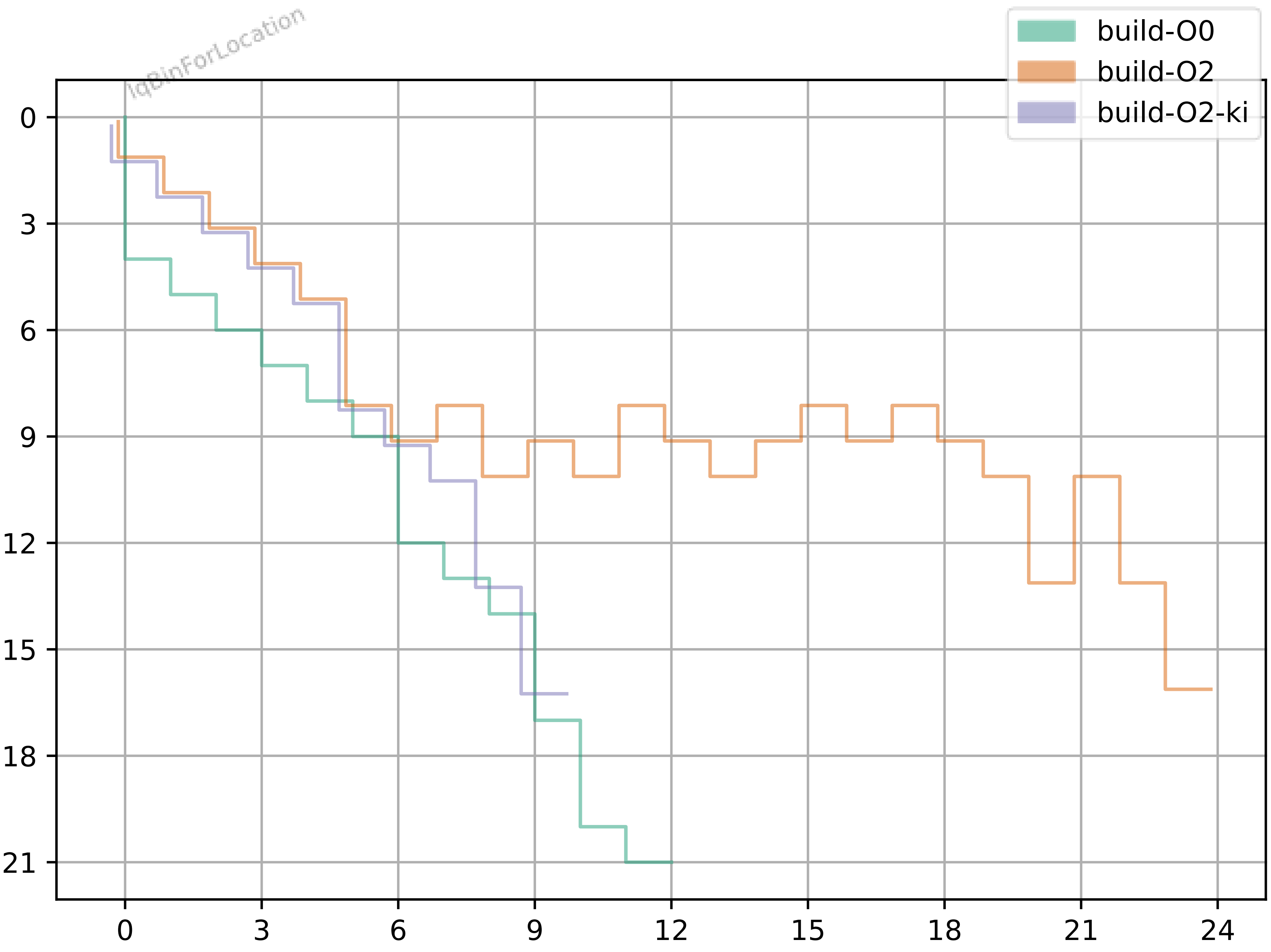
Missing step at O2 with & without key instructions

A debugging trace from Lua (bytecode interpreter, C)

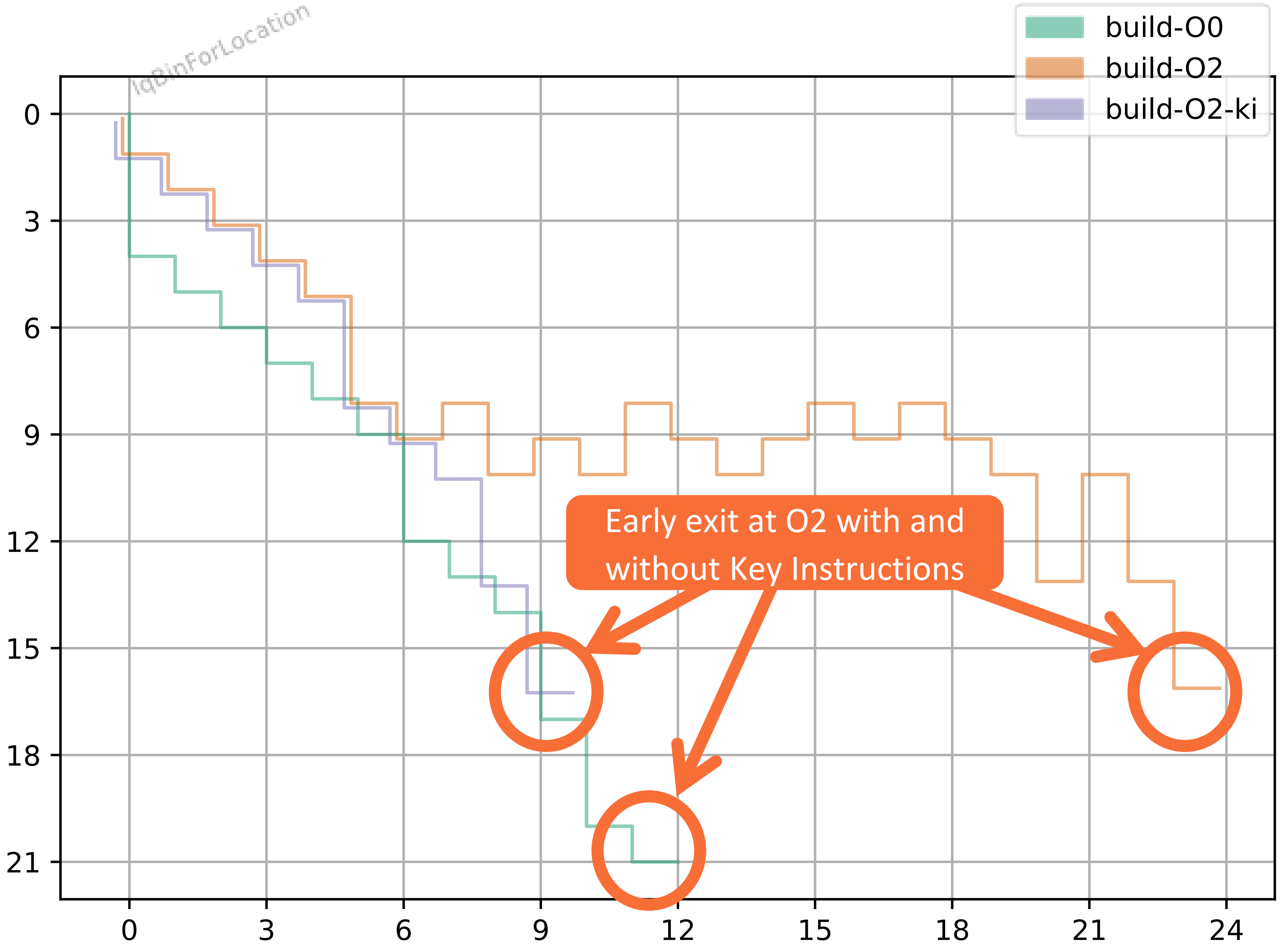


Jumpy stepping at O2 avoided with Key Instructions

A debugging trace from OpenSteer (agent steering library, C++)



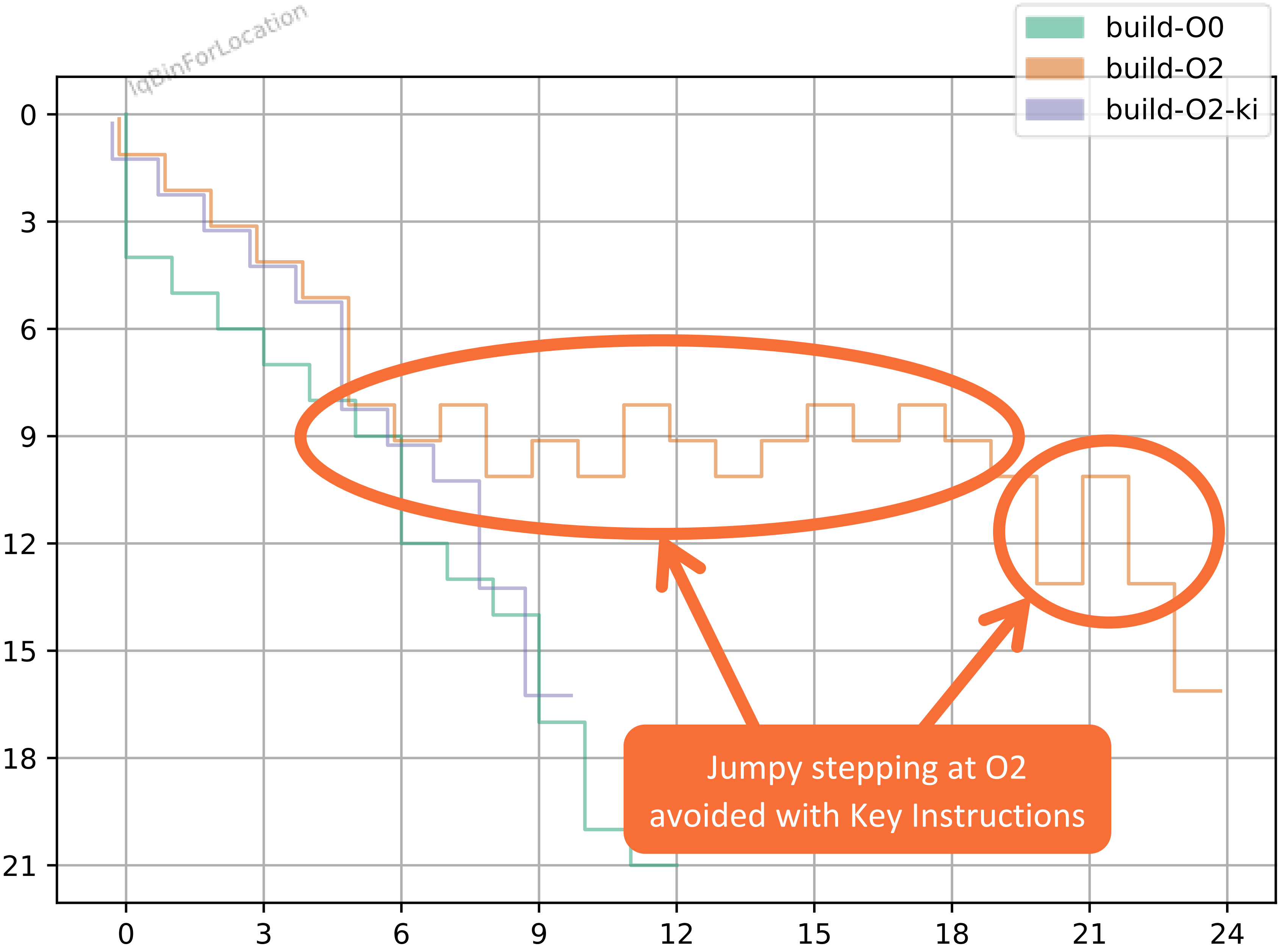
A debugging trace from OpenSteer (agent steering library, C++)



Early exit at O2 with and without Key Instructions

A debugging trace from OpenSteer

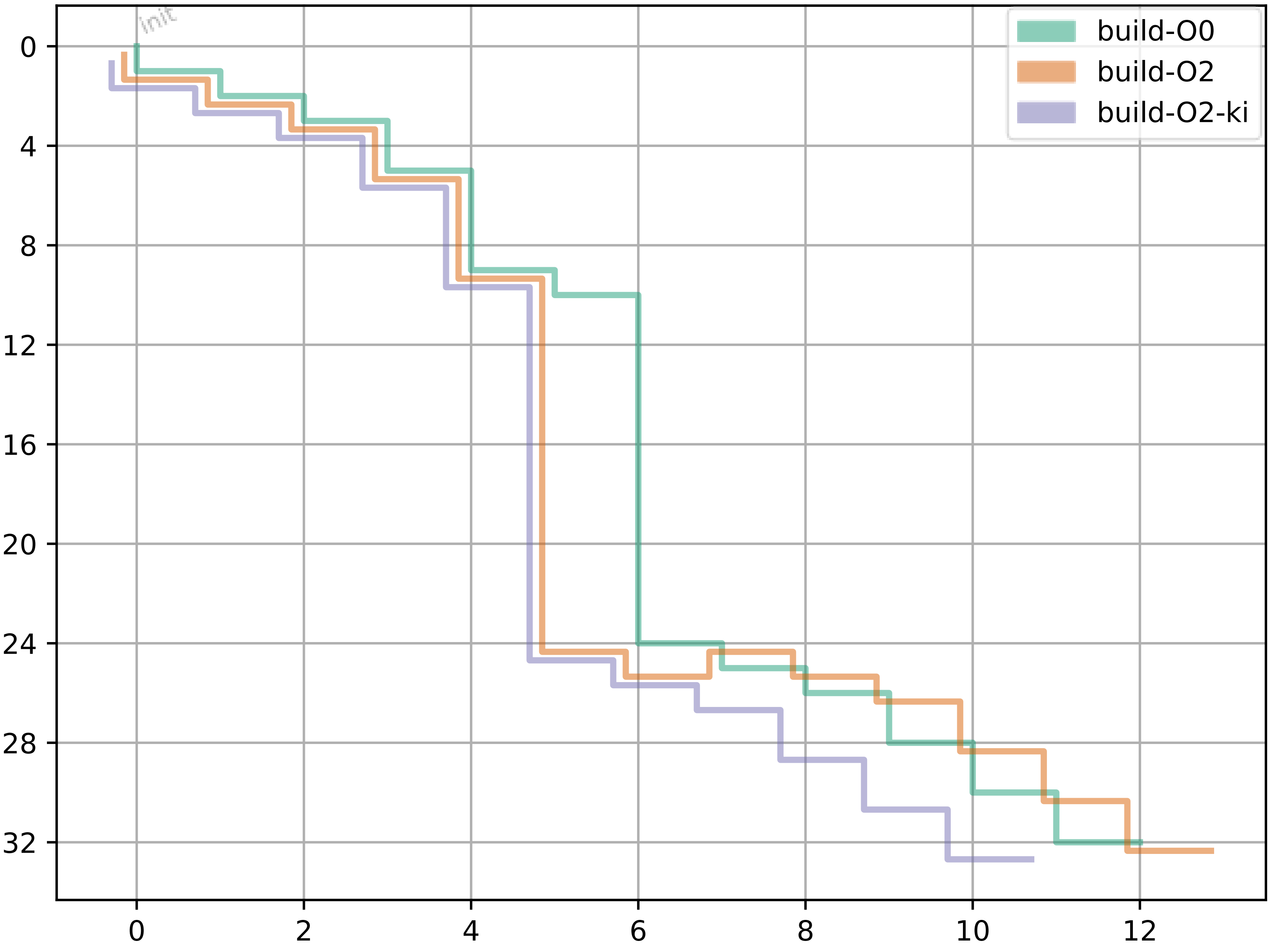
(agent steering library, C++)



Jumpy stepping at O2 avoided with Key Instructions

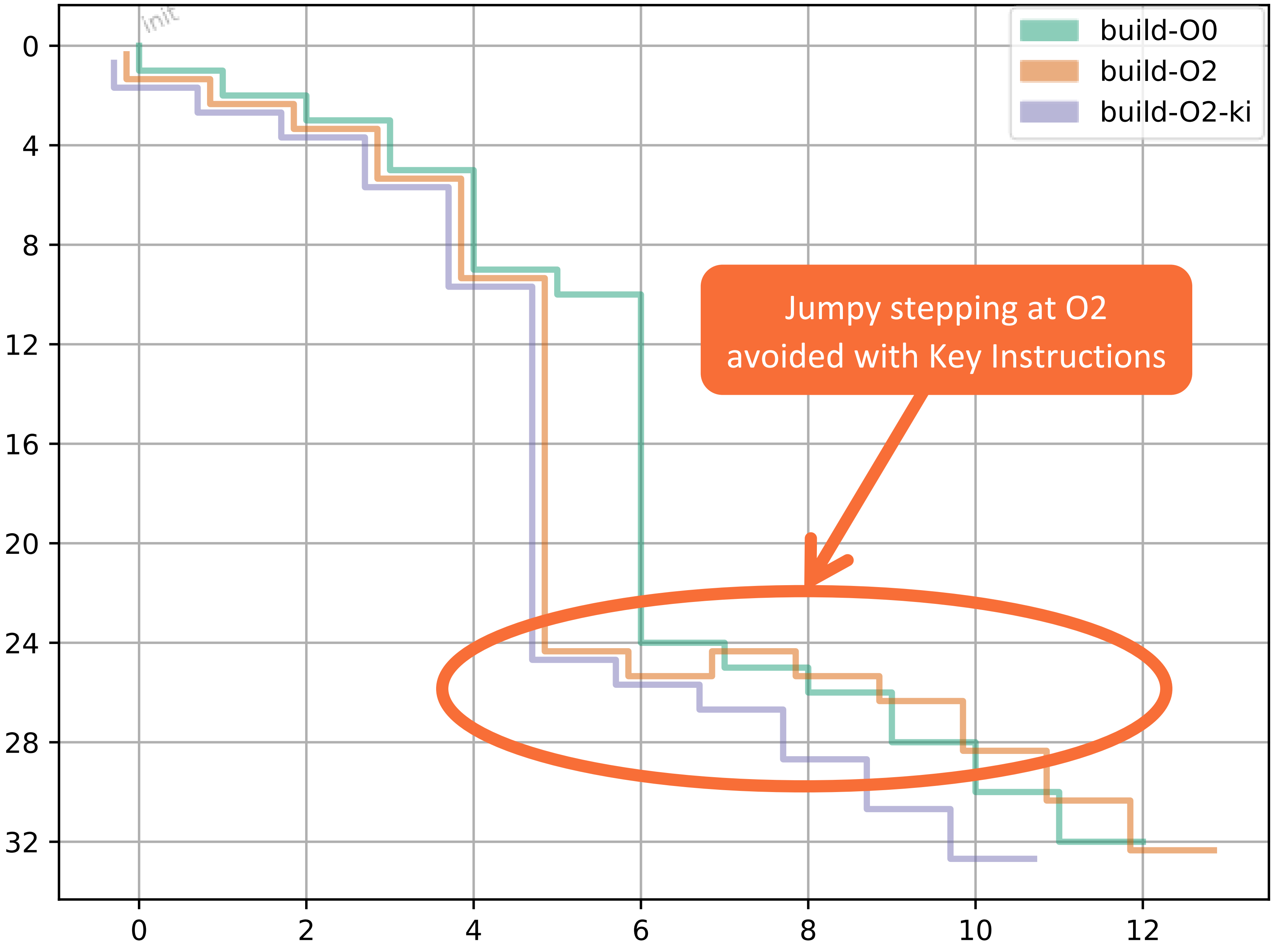
A debugging trace from AGG aa_demo

(vector graphics library, C++)



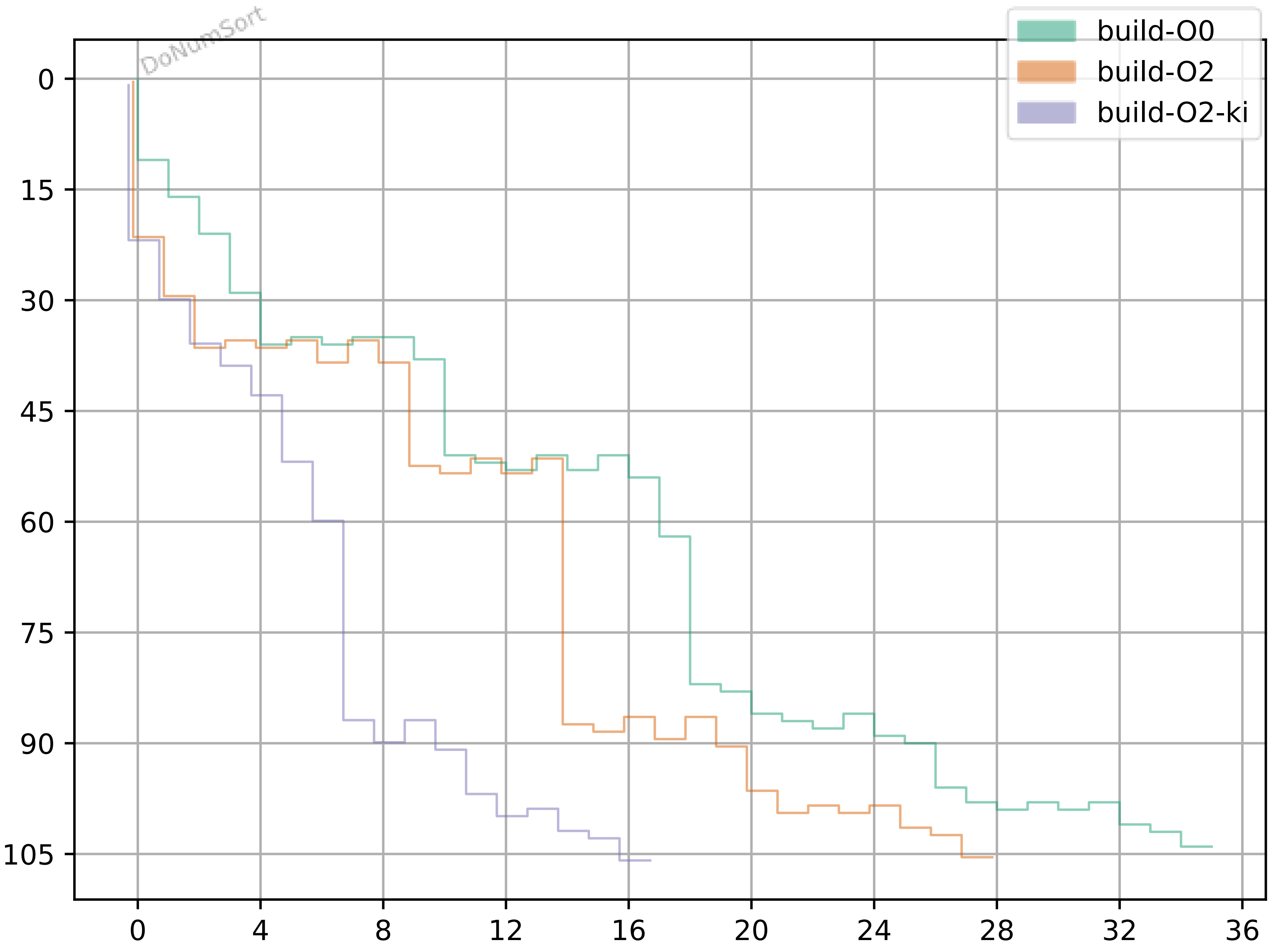
A debugging trace from AGG aa_demo

(vector graphics library, C++)

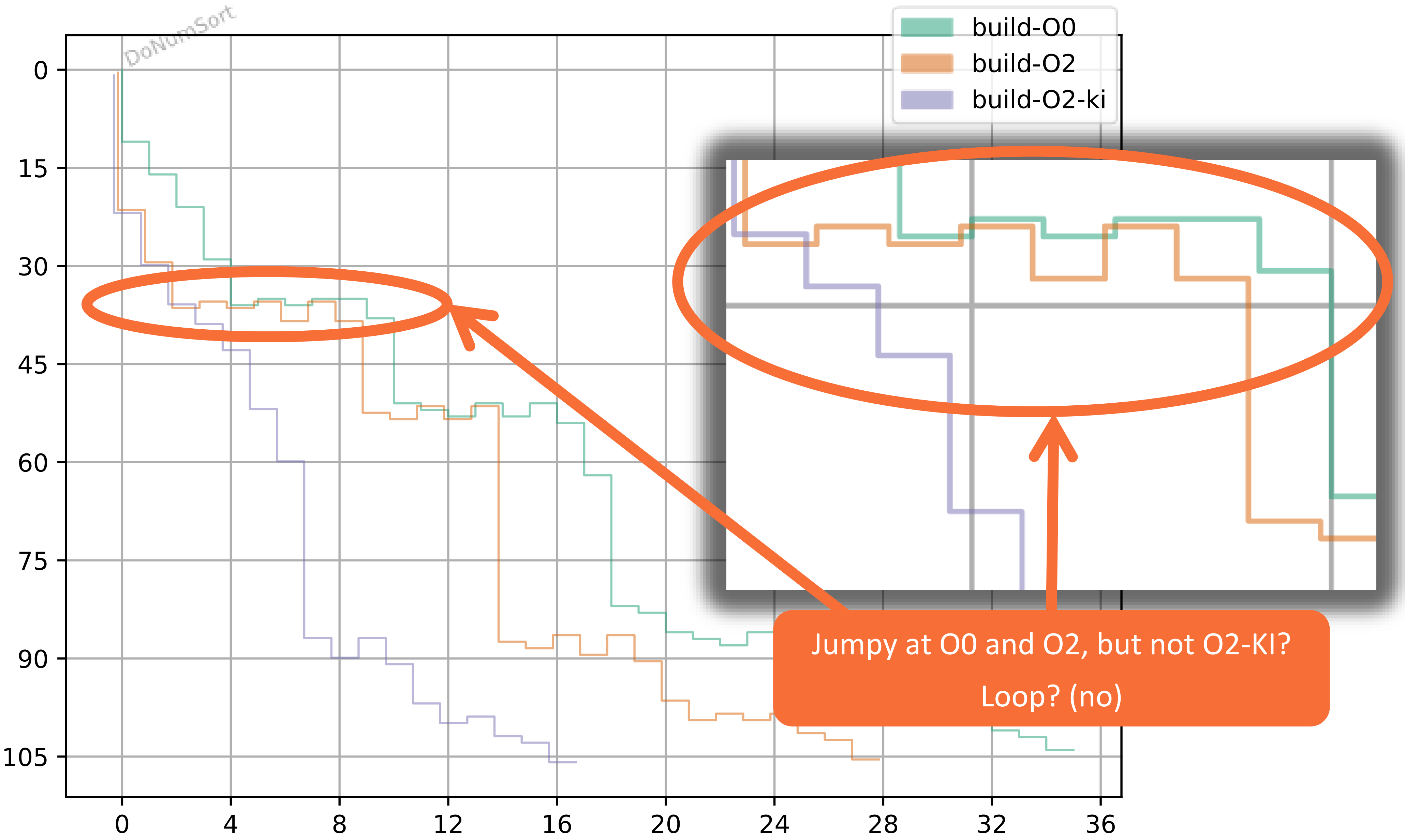


Jumpy stepping at O2 avoided with Key Instructions

A debugging trace from BYTEmark (benchmarking, C)



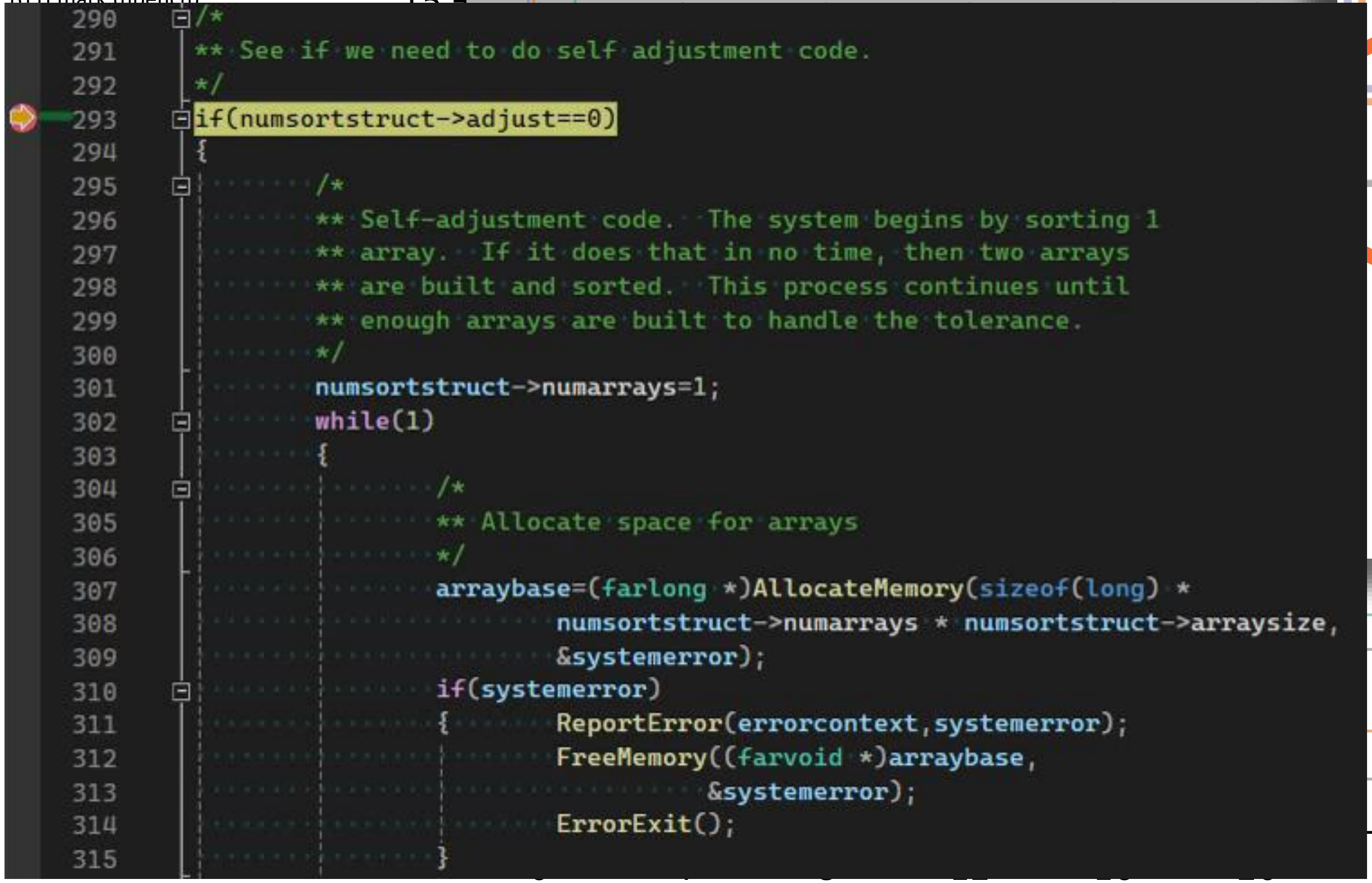
A debugging trace from BYTEmark (benchmarking, C)



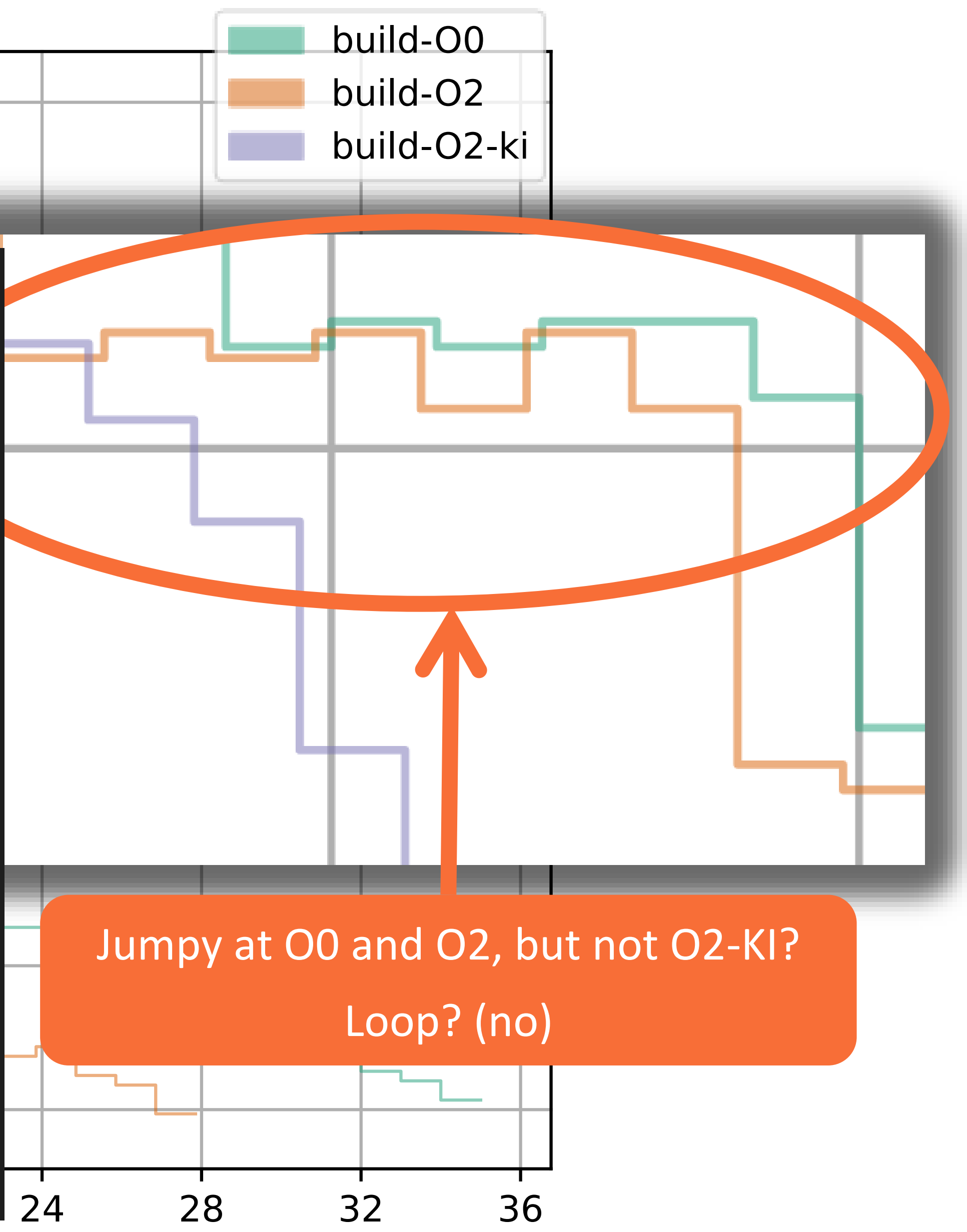
Jumpy at O0 and O2, but not O2-KI? Loop? (no)

Single step through

BYTEmark (nbench)



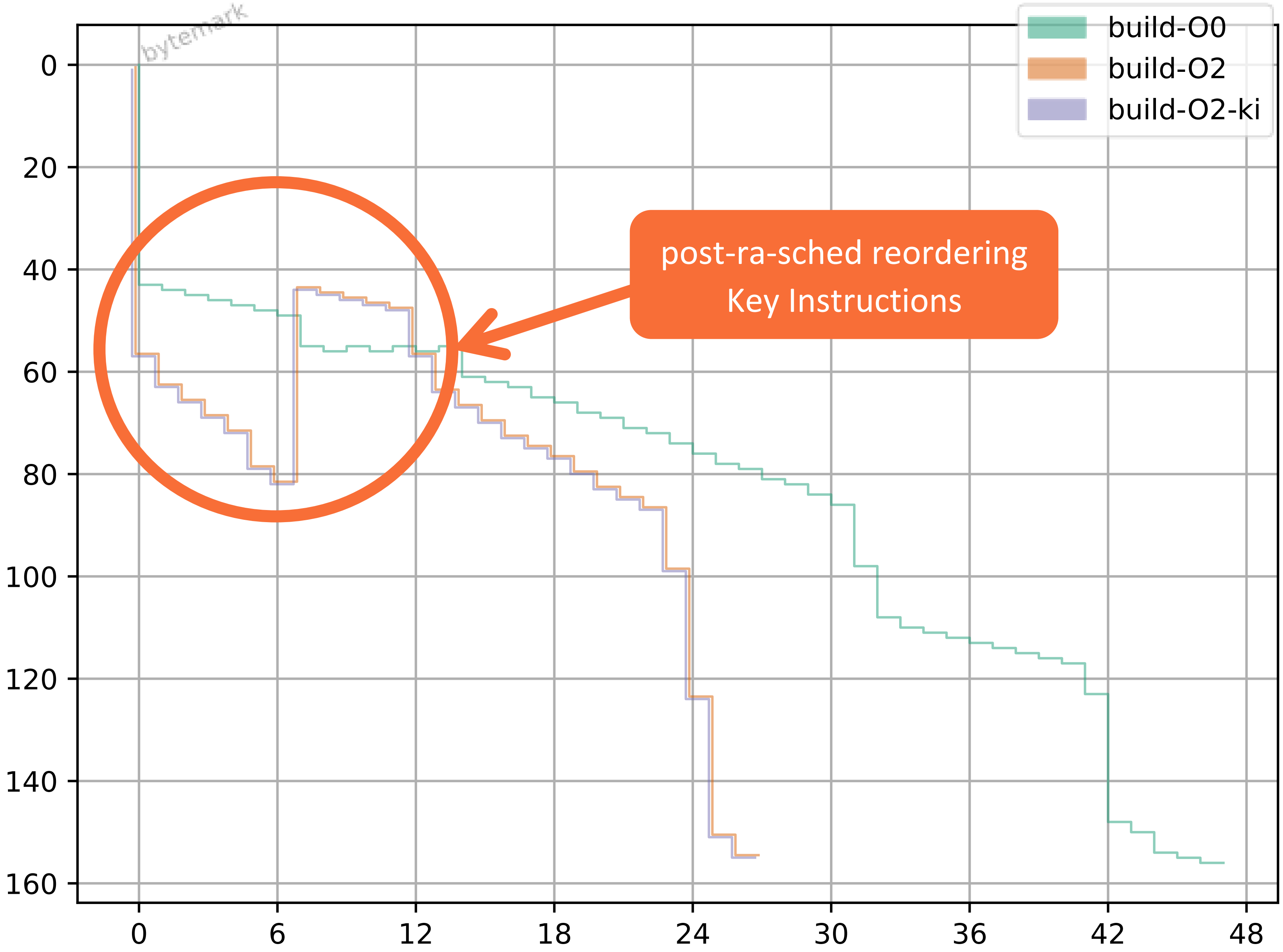
```
/*
** See if we need to do self adjustment code.
*/
if(numsortstruct->adjust==0)
{
    /*
    ** Self-adjustment code.  The system begins by sorting 1
    ** array.  If it does that in no time, then two arrays
    ** are built and sorted.  This process continues until
    ** enough arrays are built to handle the tolerance.
    */
    numsortstruct->numarrays=1;
    while(1)
    {
        /*
        ** Allocate space for arrays
        */
        arraybase=(farlong *)AllocateMemory(sizeof(long) *
                numsortstruct->numarrays * numsortstruct->arraysize,
                &systemerror);
        if(systemerror)
        {       ReportError(errorcontext,systemerror);
                FreeMemory((farvoid *)arraybase,
                          &systemerror);
                ErrorExit();
        }
```



Jumpy at O0 and O2, but not O2-KI?

Loop? (no)

A debugging trace from BYTEmark (benchmarking, C)



post-ra-sched reordering
Key Instructions

Key Instructions summary

- Code motion and scheduling causes a lot of jumpiness
- Smarter is_stmt placement can greatly reduce impact
- Pass authors encode intent with new API
- See RFC for open question(s)

Thanks for listening

- LLVM’s optimized-code line tables can be improved

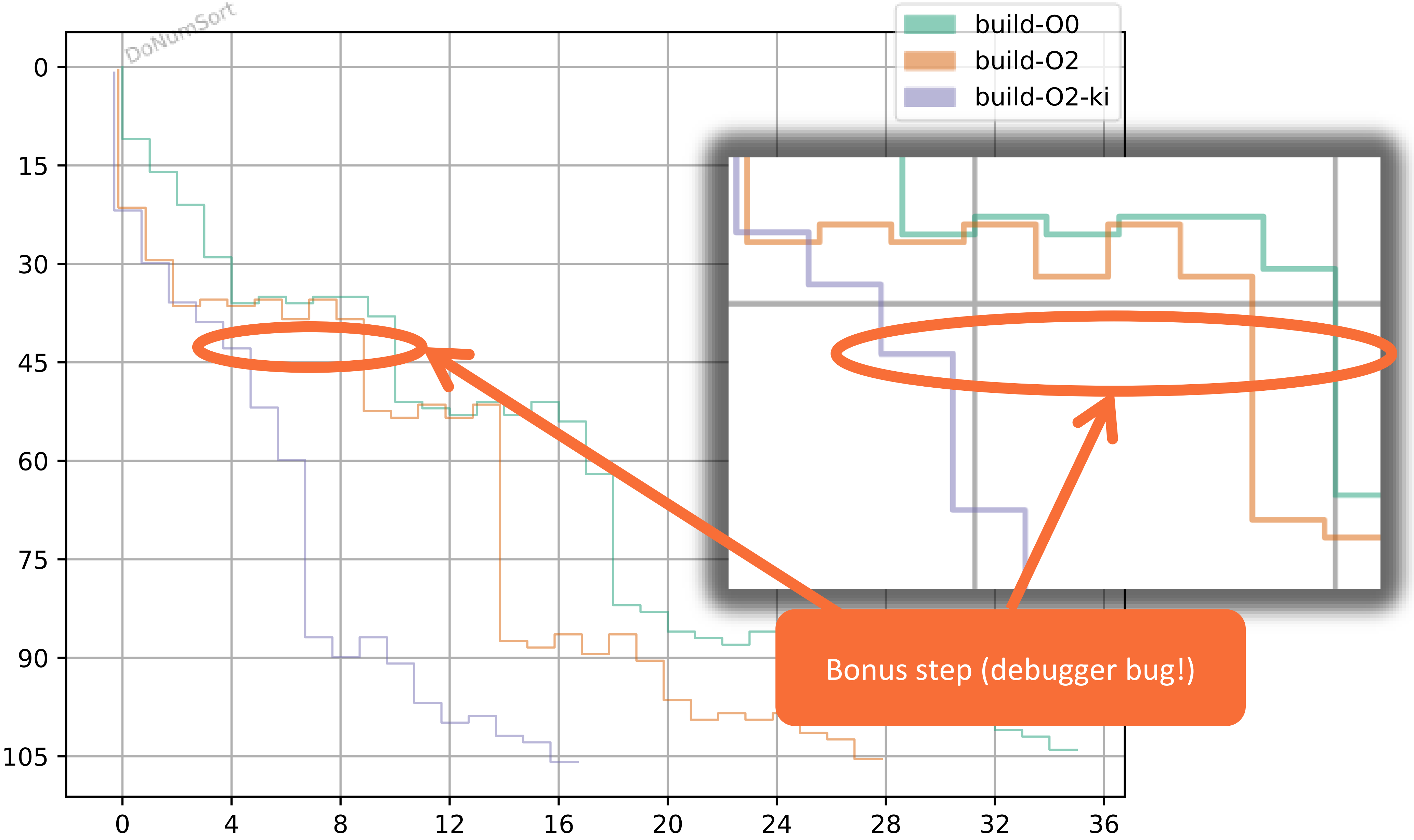
- Use new APIs to encode info about optimisations for better debug info handling

- Source location defect finder
  - Improves attribution
  - https://discourse.llvm.org/t/**rfc-proposed-update-to-handling-debug-locations-in-llvm**

- Key instructions
  - Improves stepping
  - https://discourse.llvm.org/t/**rfc-improving-is-stmt-placement-for-better-interactive-debugging**

A debugging trace from BYTEmark

(benchmarking, C)



Bonus step (debugger bug!)