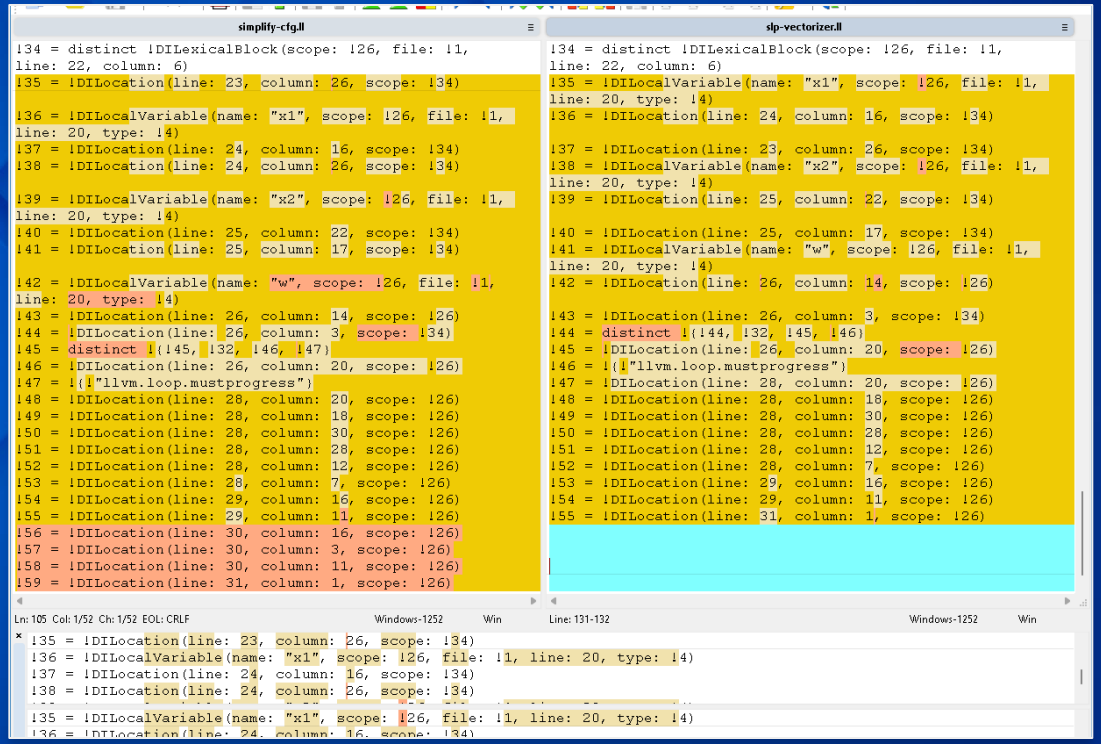
Why add an IR reader to llvm-debuginfo-analyzer tool

Carlos Alberto Enciso

Reduce the noisiness of comparing the debuginfo in LLVM IR

Reduce the noisiness of comparing the debuginfo in LLVM IR



IR changes: comparison tool

```
Reference: 'simplify-cfg.ll'
Target:    'slp-vectorizer.ll'

(2) Missing Symbols:
-    20     {Variable} 'x1' -> 'float'
-    20     {Variable} 'x2' -> 'float'

(5) Missing Lines:
-    24     {Line}
-    30     {Line}
-    30     {Line}
-    30     {Line}
-    24     {Line}
```

```
Reference: 'simplify-cfg.ll'
Target:    'slp-vectorizer.ll'

Logical View:
           {File} 'simplify-cfg.ll'

             {CompileUnit} 'test.cpp'
     13        {Function} extern not_inlined 'RandF32' -> 'float'
     14          {Variable} 'uRand' -> 'U32'
     15          {Variable} 'fRand' -> 'F32'
     19        {Function} extern not_inlined 'randGauss' -> 'void'
     19          {Parameter} 'work' -> '* float'
     20          {Variable} 'w' -> 'float'
-    20          {Variable} 'x1' -> 'float'
-    20          {Variable} 'x2' -> 'float'
```

IR changes: llvm-debuginfo-analyzer

Why add an IR reader to llvm-debuginfo-analyzer tool

LLVM and debug information

- Different formats, toolchain, tools
- Common problems

Analyzing optimizer IR output

- IR before/after an optimizer pass
- Test case: SLP Vectorizer pass drops debug information

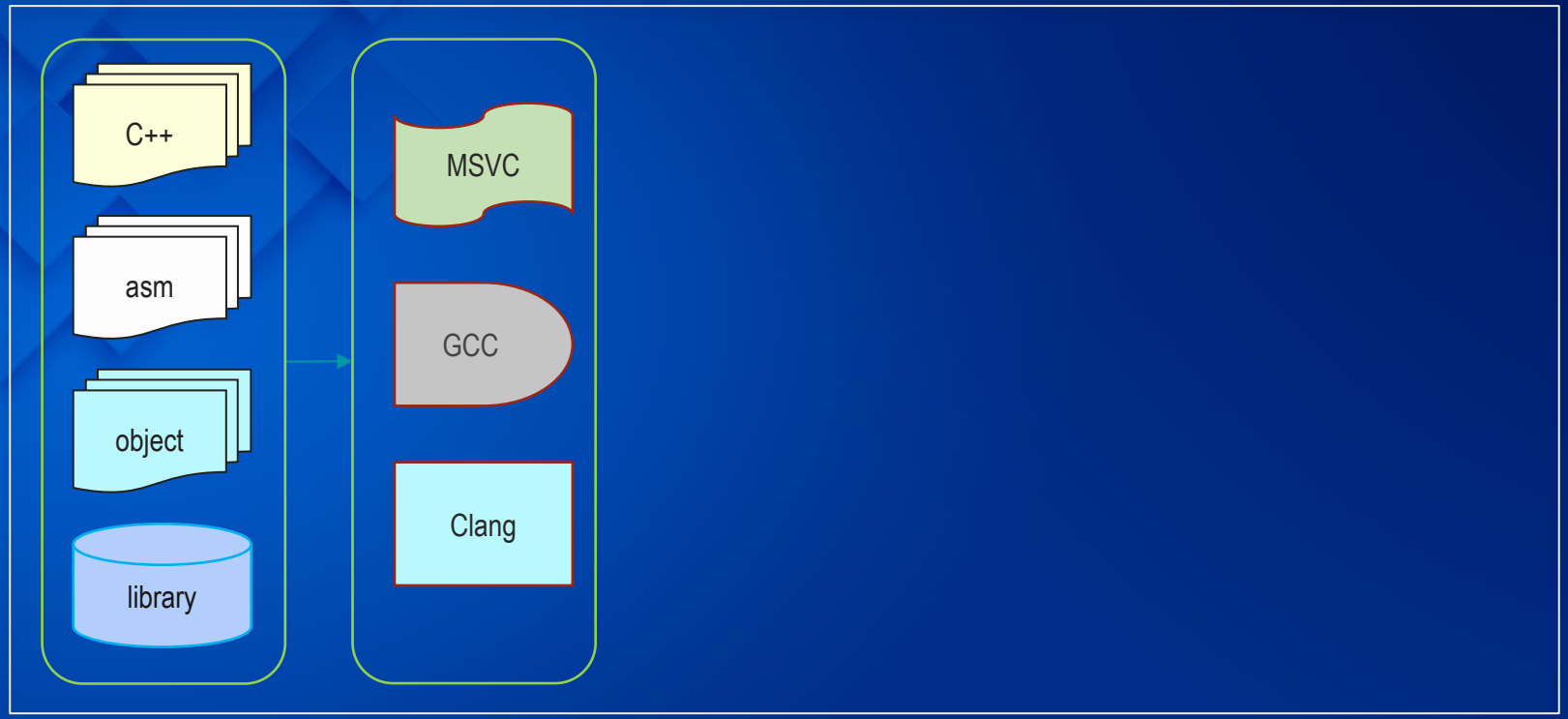
llvm-debuginfo-analyzer

- Basic introduction
- Print logical view
- Compare logical view

LLVM and debug information - inputs

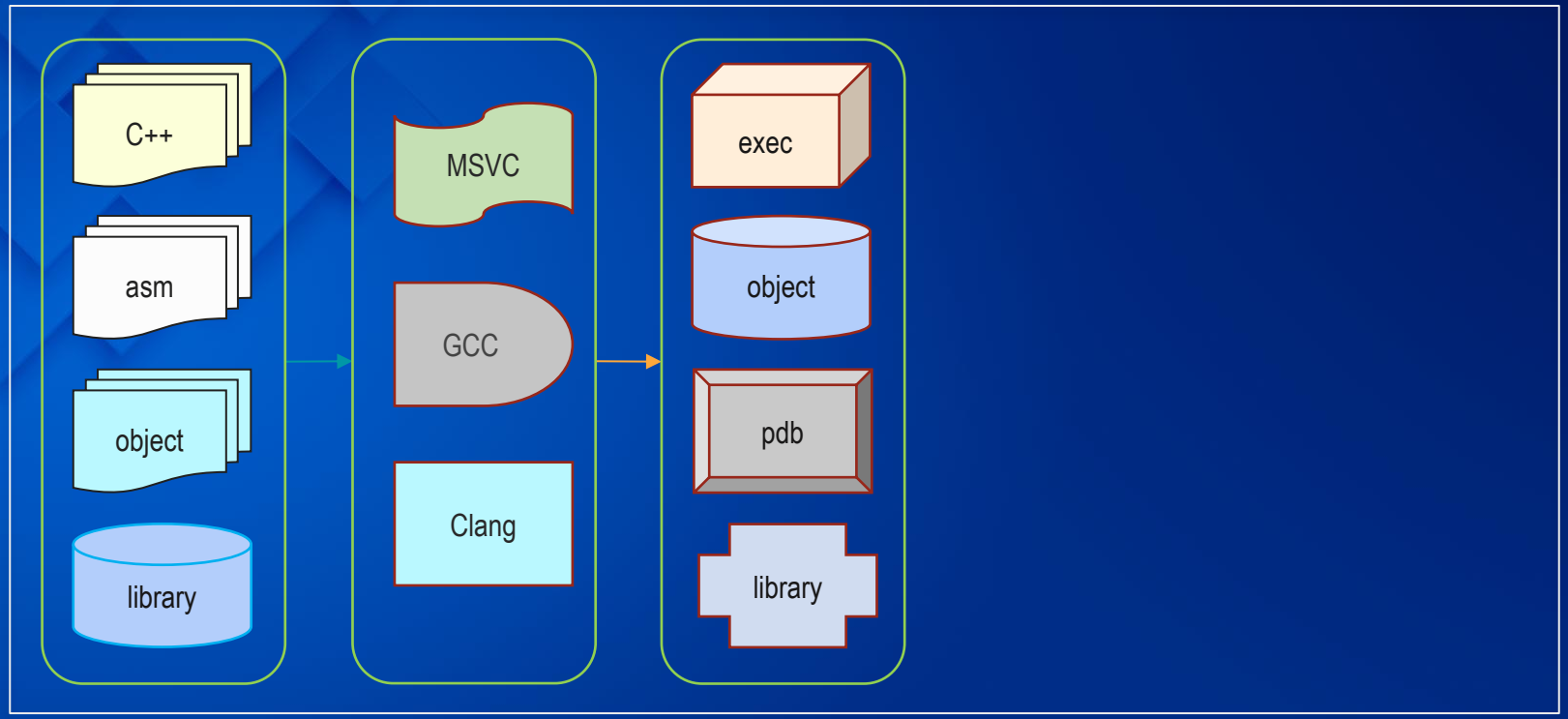
Different inputs

LLVM and debug information - toolchains



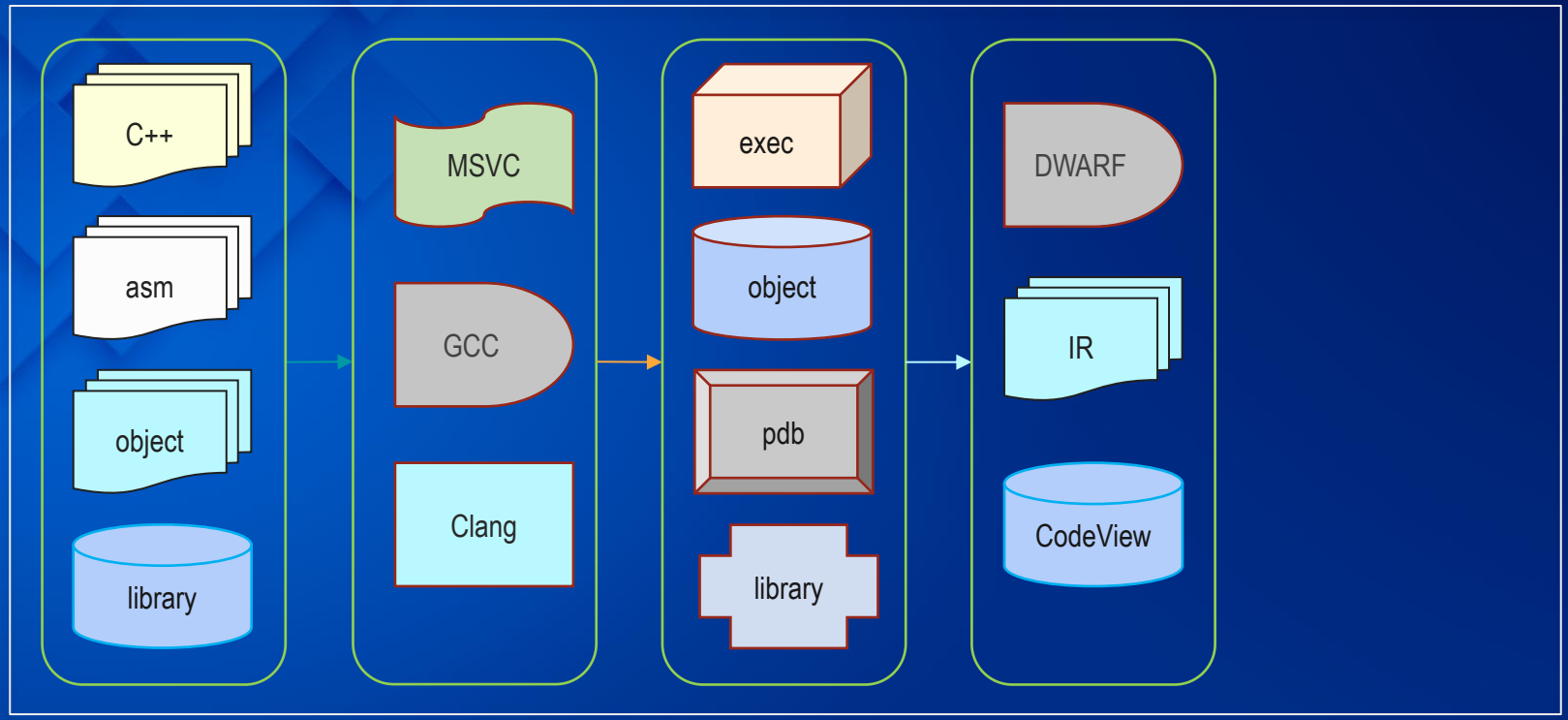
Different toolchains

LLVM and debug information - binary file formats



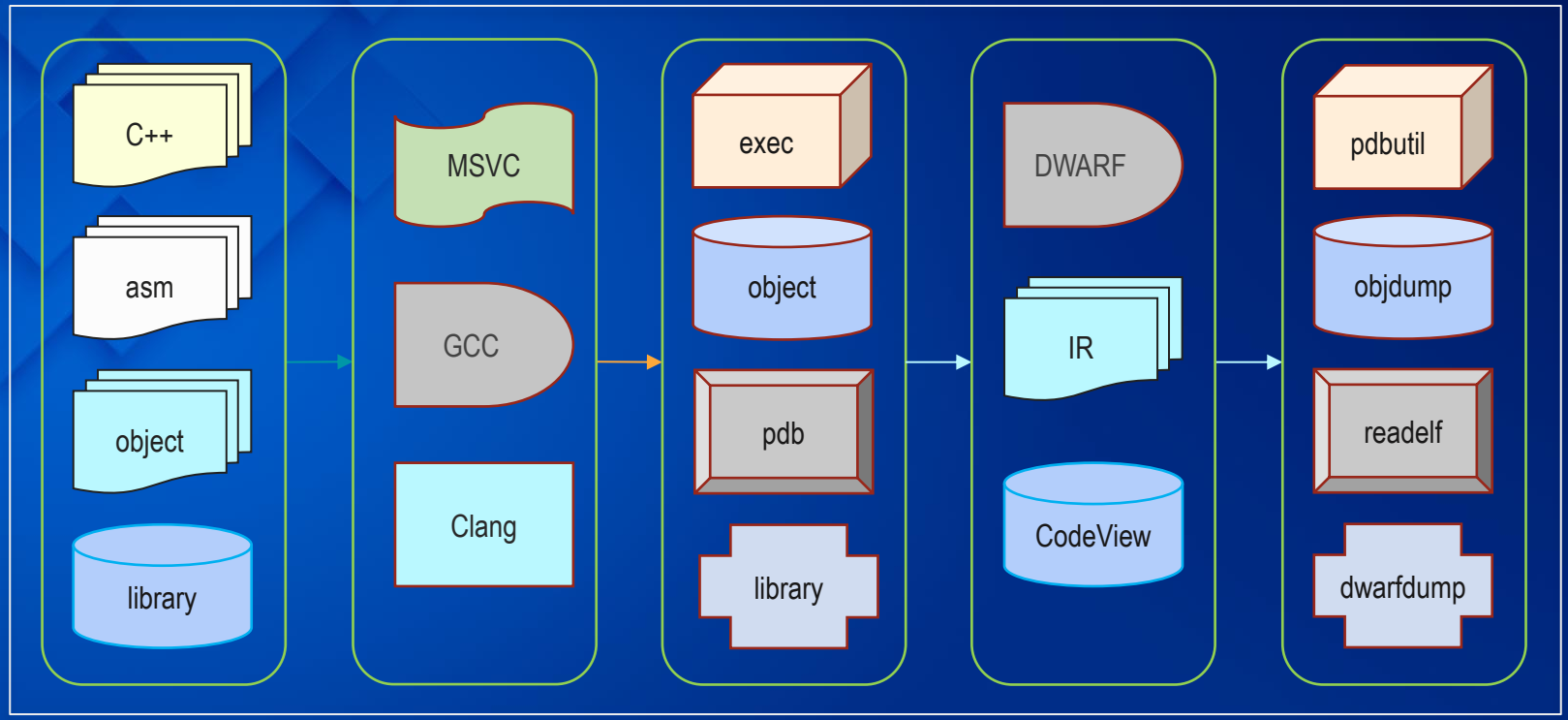
Different binary formats

LLVM and debug information - debug information formats



Different debug information formats

LLVM and debug information - debug information tools



Different debug information tools

Common problems with debug information

Does the debug information represent the original source

- Which variables are dropped due to optimization
- Why I cannot stop at a particular line
- Which lines are associated to a specific code range
- Size changes due to toolchain features

Common problems with debug information

Does the debug information represent the original source
- Which variables are dropped due to optimization
- Why I cannot stop at a particular line
- Which lines are associated to a specific code range
- Size changes due to toolchain features

Semantic differences in the generated debug information
- By different toolchain versions (same platform)
- Same source code compiled in different platforms

Analyzing optimizer IR output

IR before/after an optimizer pass

The IR output is very rich and noisy

- The metadata identifiers changes between passes
- Difficult to see the changes using a comparison tool
- Requires knowledge about the IR and passes
- By bisection, we can pin down which pass causes the change

Analyzing optimizer IR output

IR before/after an optimizer pass

The IR output is very rich and noisy

- The metadata identifiers changes between passes
- Difficult to see the changes using a comparison tool
- Requires knowledge about the IR and passes
- By bisection, we can pin down which pass causes the change

The optimizer includes a wide set of debug printing options

- --print-before, --print-before-all, --print-after, --print-after-all, etc.

Analyzing optimizer IR output

IR before/after an optimizer pass

The IR output is very rich and noisy
- The metadata identifiers changes between passes
- Difficult to see the changes using a comparison tool
- Requires knowledge about the IR and passes
- By bisection, we can pin down which pass causes the change

The optimizer includes a wide set of debug printing options
- --print-before, --print-before-all, --print-after, --print-after-all, etc.

Test case: SLP Vectorizer pass drops debug information

- [DebugInfo@O2] https://github.com/llvm/llvm-project/issues/45507
- Drops location information for local variables x1 and x2
- Generate IR after Simplify CFG and SLP Vectorizer passes

IR metadata - Simplify CFG and SLP Vectorizer passes

```
 1 // Compile with clang -g -O2.
 2 // The SLP Vectorizer pass drops location
 3 // information for the local variables:
 4 // x1 and x2.
 5
 6 typedef unsigned int U32;
 7 typedef float F32;
 8 extern "C" double log(float);
 9 extern "C" double sqrt(float);
10
11 extern unsigned RandU32();
12
13 float RandF32() {
14   U32 uRand = RandU32();
15   F32 fRand = ((F32)uRand / 4294967810.0f);
16   return (fRand);
17 }
18
19 void randGauss(float work[2]) {
20   float x1, x2, w;
21
22   do {
23     x1 = 2.f * RandF32() - 1.f;
24     x2 = 2.f * RandF32() - 1.f;
25     w = x1 * x1 + x2 * x2;
26   } while (w >= 1.f);
27
28   w = sqrt((-2.f * log(w)) / w);
29   work[0] = x1 * w;
30   work[1] = x2 * w;
31 }
```

IR metadata - Simplify CFG and SLP Vectorizer passes

```
 1 // Compile with clang -g -O2.
 2 // The SLP Vectorizer pass drops location
 3 // information for the local variables:
 4 // x1 and x2.
 5
 6 typedef unsigned int U32;
 7 typedef float F32;
 8 extern "C" double log(float);
 9 extern "C" double sqrt(float);
10
11 extern unsigned RandU32();
12
13 float RandF32() {
14   U32 uRand = RandU32();
15   F32 fRand = ((F32)uRand / 4294967810.0f);
16   return (fRand);
17 }
18
19 void randGauss(float work[2]) {
20   float x1, x2, w;
21
22   do {
23     x1 = 2.f * RandF32() - 1.f;
24     x2 = 2.f * RandF32() - 1.f;
25     w = x1 * x1 + x2 * x2;
26   } while (w >= 1.f);
27
28   w = sqrt((-2.f * log(w)) / w);
29   work[0] = x1 * w;
30   work[1] = x2 * w;
31 }
```

```
!35 = !DILocation(line: 23, column: 26, scope: !34)
!36 = !DILocalVariable(name: "x1", scope: !26, file:
!1, line: 20, type: !4)
!37 = !DILocation(line: 24, column: 16, scope: !34)
!38 = !DILocation(line: 24, column: 26, scope: !34)
!39 = !DILocalVariable(name: "x2", scope: !26, file:
!1, line: 20, type: !4)
!40 = !DILocation(line: 25, column: 22, scope: !34)
!41 = !DILocation(line: 25, column: 17, scope: !34)
!42 = !DILocalVariable(name: "w", scope: !26, file:
!1, line: 20, type: !4)
!43 = !DILocation(line: 26, column: 14, scope: !26)
!44 = !DILocation(line: 26, column: 3, scope: !34)
!45 = distinct !{!45, !32, !46, !47}
!46 = !DILocation(line: 26, column: 20, scope: !26)
!47 = !{!"llvm.loop.mustprogress"}
!48 = !DILocation(line: 28, column: 20, scope: !26)
!49 = !DILocation(line: 28, column: 18, scope: !26)
!50 = !DILocation(line: 28, column: 30, scope: !26)
!51 = !DILocation(line: 28, column: 28, scope: !26)
!52 = !DILocation(line: 28, column: 12, scope: !26)
!53 = !DILocation(line: 28, column: 7, scope: !26)
!54 = !DILocation(line: 29, column: 16, scope: !26)
!55 = !DILocation(line: 29, column: 11, scope: !26)
!56 = !DILocation(line: 30, column: 16, scope: !26)
!57 = !DILocation(line: 30, column: 3, scope: !26)
!58 = !DILocation(line: 30, column: 11, scope: !26)
!59 = !DILocation(line: 31, column: 1, scope: !26)
```

IR metadata after Simplify CFG

IR metadata - Simplify CFG and SLP Vectorizer passes

```
 1 // Compile with clang -g -O2.
 2 // The SLP Vectorizer pass drops location
 3 // information for the local variables:
 4 // x1 and x2.
 5
 6 typedef unsigned int U32;
 7 typedef float F32;
 8 extern "C" double log(float);
 9 extern "C" double sqrt(float);
10
11 extern unsigned RandU32();
12
13 float RandF32() {
14   U32 uRand = RandU32();
15   F32 fRand = ((F32)uRand / 4294967810.0f);
16   return (fRand);
17 }
18
19 void randGauss(float work[2]) {
20   float x1, x2, w;
21
22   do {
23     x1 = 2.f * RandF32() - 1.f;
24     x2 = 2.f * RandF32() - 1.f;
25     w = x1 * x1 + x2 * x2;
26   } while (w >= 1.f);
27
28   w = sqrt((-2.f * log(w)) / w);
29   work[0] = x1 * w;
30   work[1] = x2 * w;
31 }
```

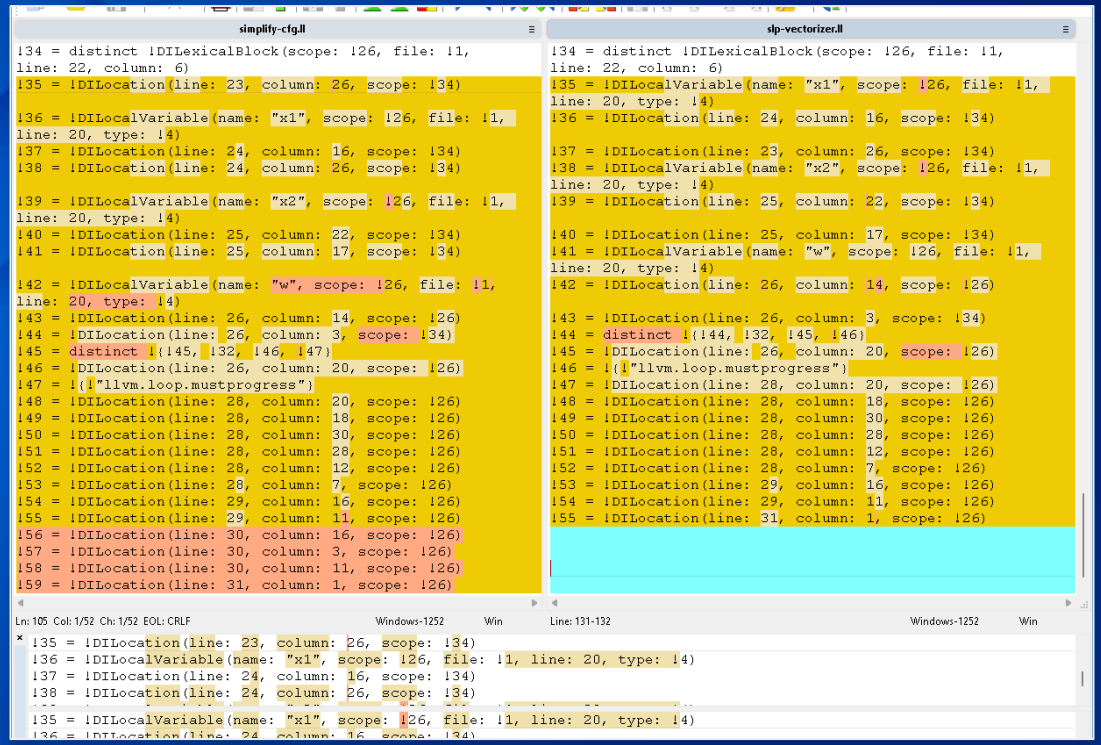
```
!35 = !DILocation(line: 23, column: 26, scope: !34)
!36 = !DILocalVariable(name: "x1", scope: !26, file:
!1, line: 20, type: !4)
!37 = !DILocation(line: 24, column: 16, scope: !34)
!38 = !DILocation(line: 24, column: 26, scope: !34)
!39 = !DILocalVariable(name: "x2", scope: !26, file:
!1, line: 20, type: !4)
!40 = !DILocation(line: 25, column: 22, scope: !34)
!41 = !DILocation(line: 25, column: 17, scope: !34)
!42 = !DILocalVariable(name: "w", scope: !26, file:
!1, line: 20, type: !4)
!43 = !DILocation(line: 26, column: 14, scope: !26)
!44 = !DILocation(line: 26, column: 3, scope: !34)
!45 = distinct !{!45, !32, !46, !47}
!46 = !DILocation(line: 26, column: 20, scope: !26)
!47 = !{!"llvm.loop.mustprogress"}
!48 = !DILocation(line: 28, column: 20, scope: !26)
!49 = !DILocation(line: 28, column: 18, scope: !26)
!50 = !DILocation(line: 28, column: 30, scope: !26)
!51 = !DILocation(line: 28, column: 28, scope: !26)
!52 = !DILocation(line: 28, column: 12, scope: !26)
!53 = !DILocation(line: 28, column: 7, scope: !26)
!54 = !DILocation(line: 29, column: 16, scope: !26)
!55 = !DILocation(line: 29, column: 11, scope: !26)
!56 = !DILocation(line: 30, column: 16, scope: !26)
!57 = !DILocation(line: 30, column: 3, scope: !26)
!58 = !DILocation(line: 30, column: 11, scope: !26)
!59 = !DILocation(line: 31, column: 1, scope: !26)
```

IR metadata after Simplify CFG

```
!35 = !DILocalVariable(name: "x1", scope: !26, file:
!1, line: 20, type: !4)
!36 = !DILocation(line: 24, column: 16, scope: !34)
!37 = !DILocation(line: 23, column: 26, scope: !34)
!38 = !DILocalVariable(name: "x2", scope: !26, file:
!1, line: 20, type: !4)
!39 = !DILocation(line: 25, column: 22, scope: !34)
!40 = !DILocation(line: 25, column: 17, scope: !34)
!41 = !DILocalVariable(name: "w", scope: !26, file:
!1, line: 20, type: !4)
!42 = !DILocation(line: 26, column: 14, scope: !26)
!43 = !DILocation(line: 26, column: 3, scope: !34)
!44 = distinct !{!44, !32, !45, !46}
!45 = !DILocation(line: 26, column: 20, scope: !26)
!46 = !{!"llvm.loop.mustprogress"}
!47 = !DILocation(line: 28, column: 20, scope: !26)
!48 = !DILocation(line: 28, column: 18, scope: !26)
!49 = !DILocation(line: 28, column: 30, scope: !26)
!50 = !DILocation(line: 28, column: 28, scope: !26)
!51 = !DILocation(line: 28, column: 12, scope: !26)
!52 = !DILocation(line: 28, column: 7, scope: !26)
!53 = !DILocation(line: 29, column: 16, scope: !26)
!54 = !DILocation(line: 29, column: 11, scope: !26)
!55 = !DILocation(line: 31, column: 1, scope: !26)
```

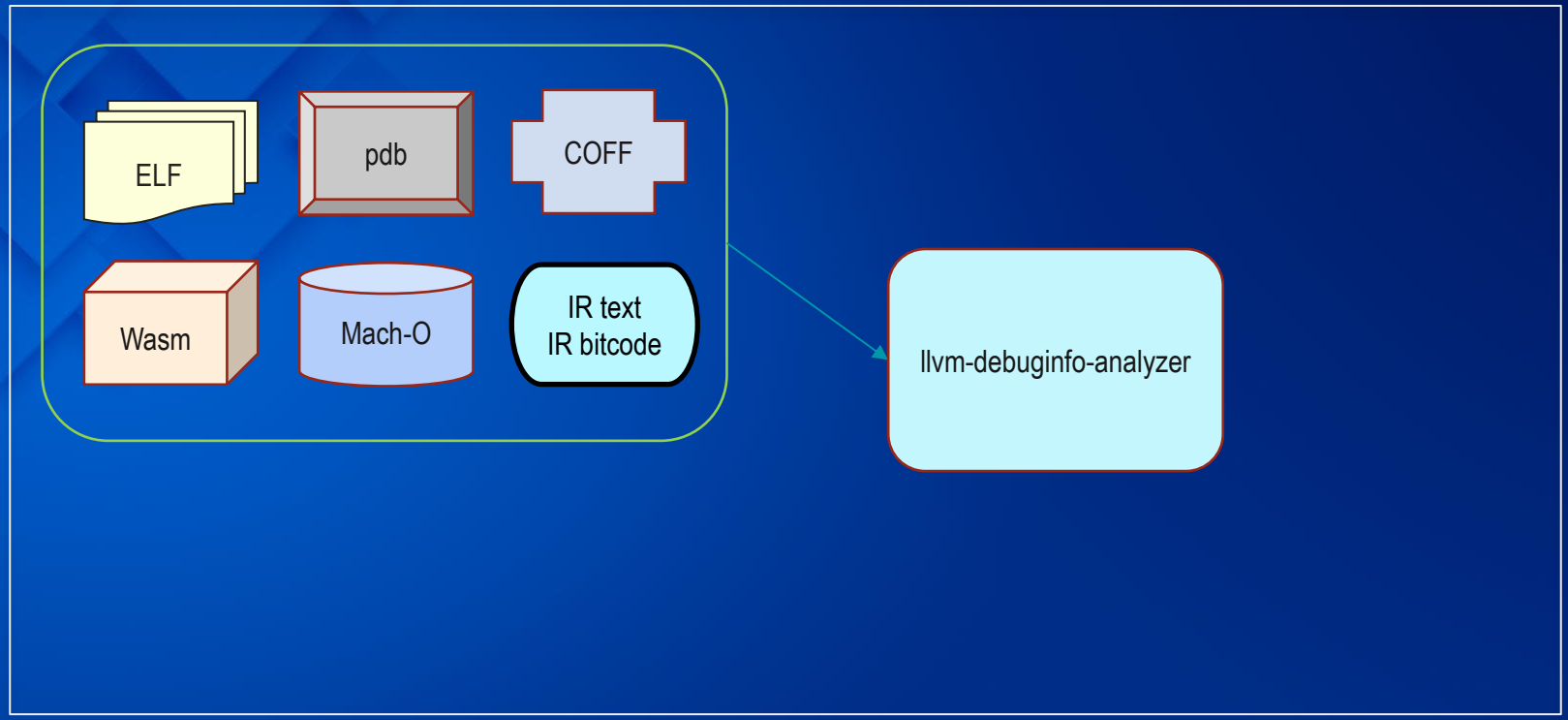
IR metadata after SLP Vectorizer

IR metadata comparison - comparison tool



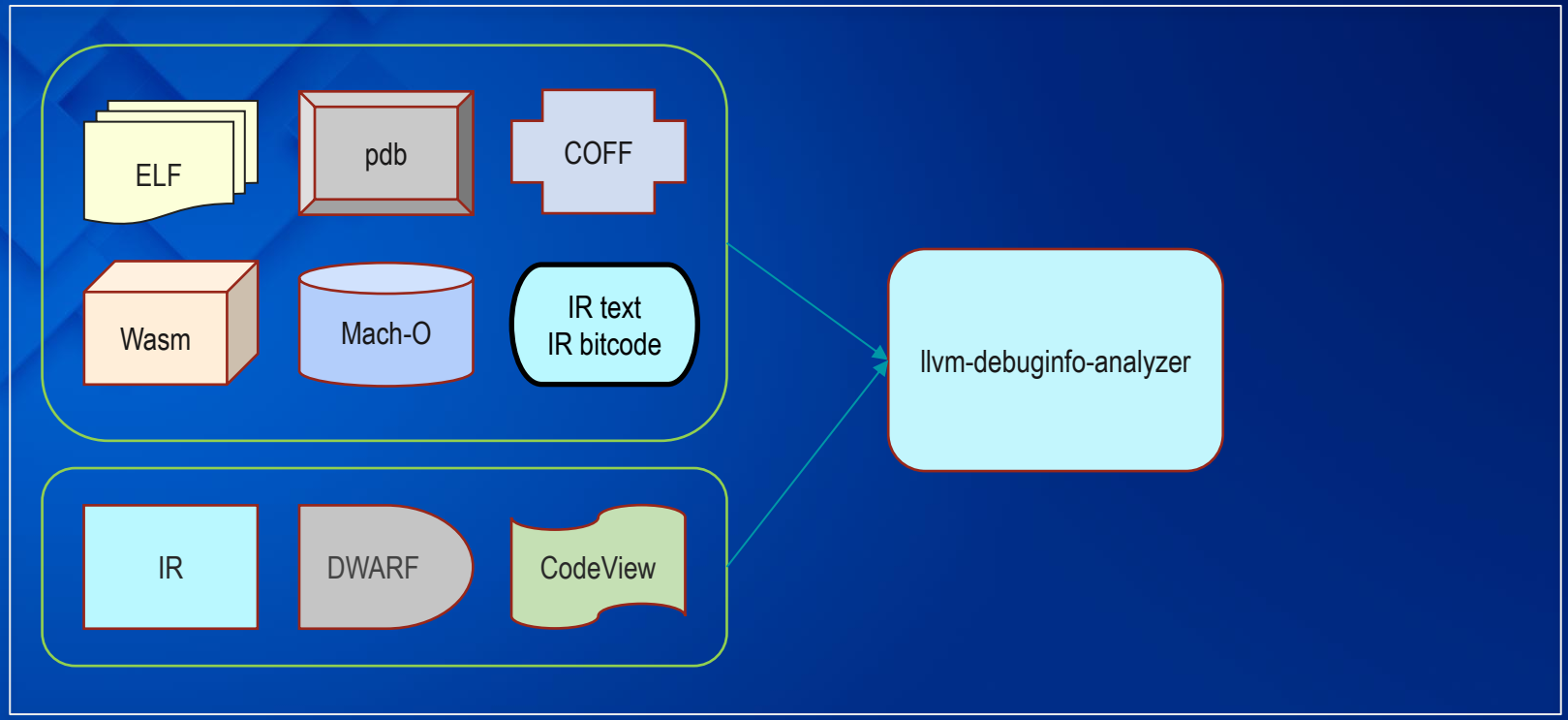
IR changes: comparison tool

llvm-debuginfo-analyzer - binary file formats



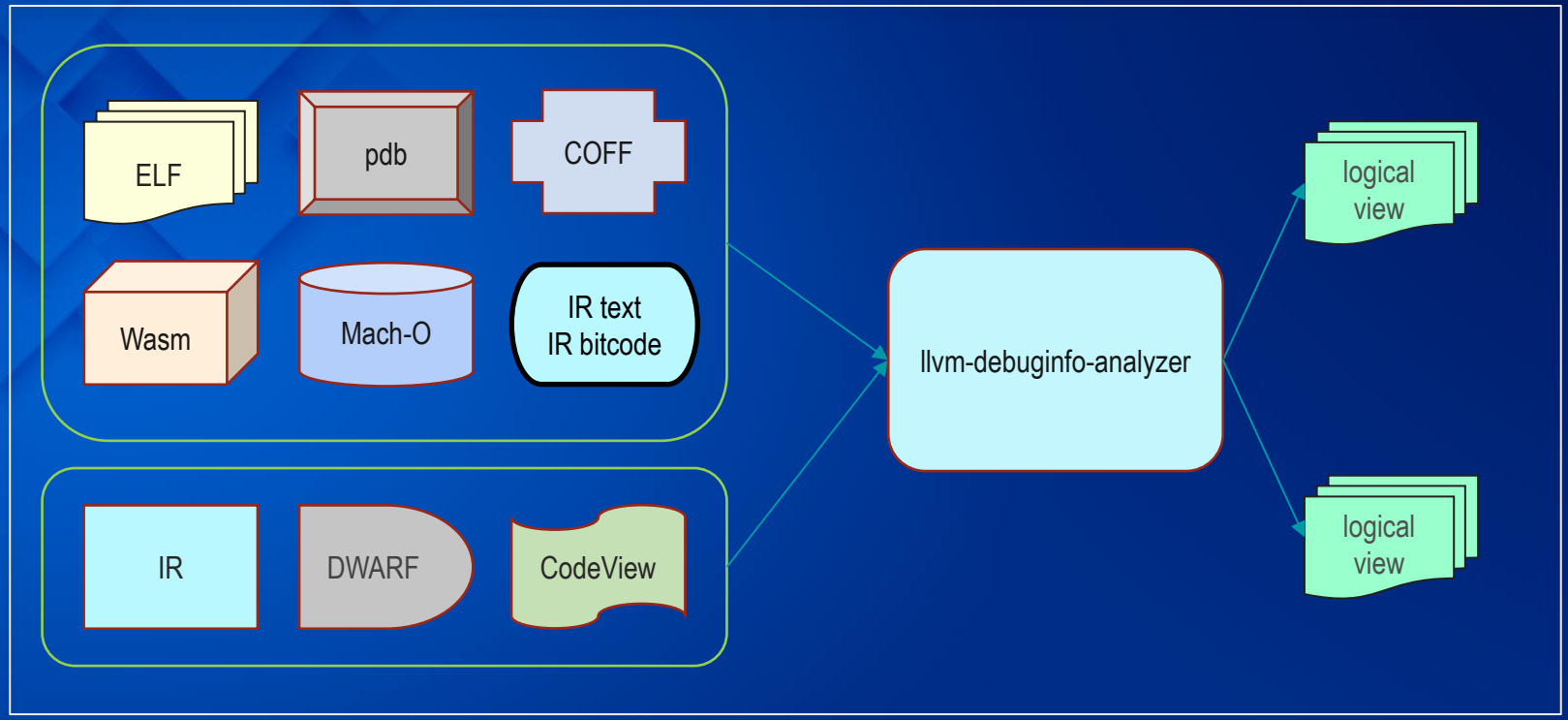
Supported binary file formats

llvm-debuginfo-analyzer - debug information formats



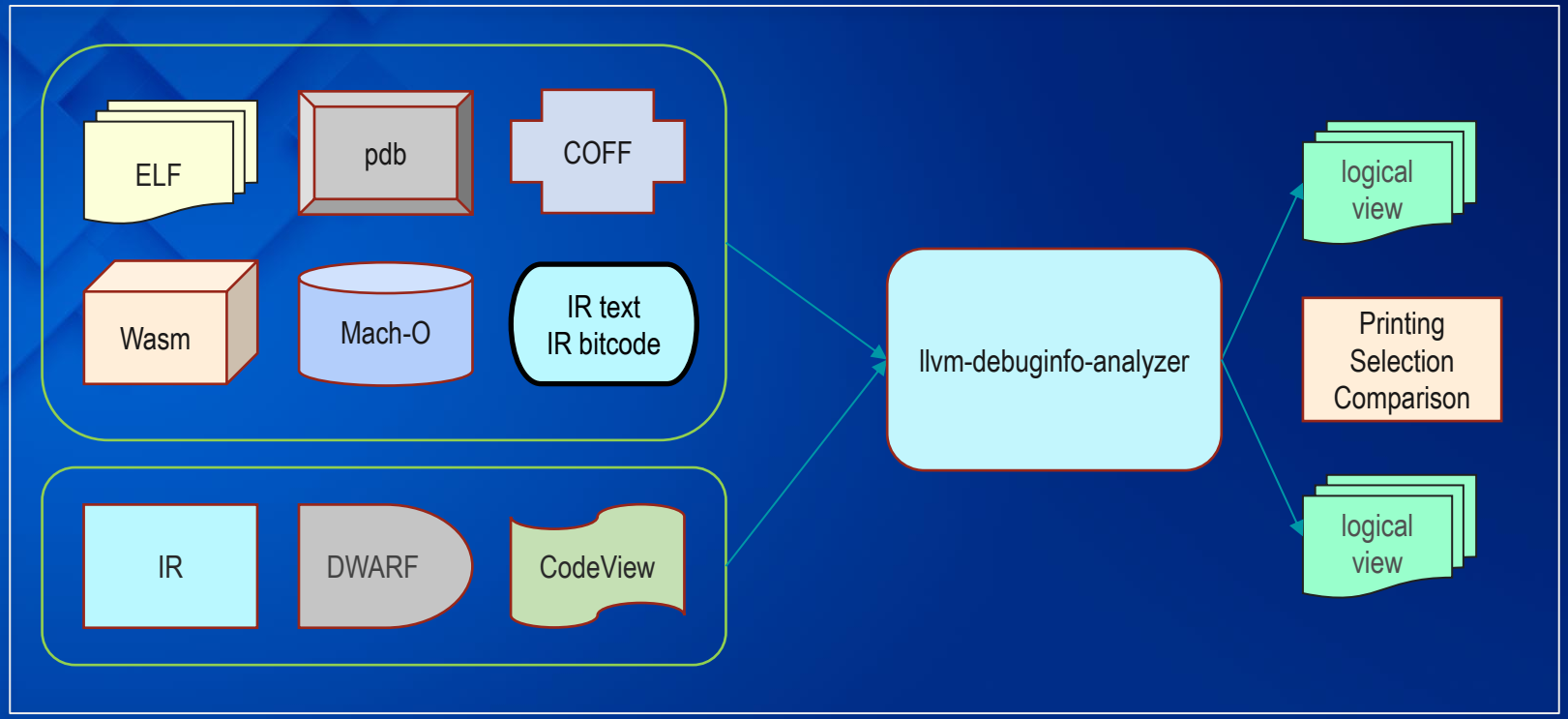
Supported debug information formats

llvm-debuginfo-analyzer - canonical logical views



Logical view is a high-level representation of the debug information

llvm-debuginfo-analyzer - printing, selection, comparison



Logical views can be printed, selected and compared

DWARF debug information (llvm-dwarfdump)

```
DW_TAG_compile_unit
  DW_AT_producer   ("clang")
  DW_AT_language   (DW_LANG_C_plus_plus_14)
  DW_AT_name       ("hello-world.cpp")
  DW_AT_low_pc     (0x0000000000000000)
  DW_AT_high_pc    (0x0000000000000014)
  DW_AT_addr_base  (0x00000008)

  DW_TAG_variable
    DW_AT_type      (0x0000002d "const char[13]")
    DW_AT_decl_line (4)
    DW_AT_location  (DW_OP_addrx 0x0)

  DW_TAG_array_type
    DW_AT_type     (0x00000039 "const char")

    DW_TAG_subrange_type
      DW_AT_type   (0x00000042 "__ARRAY_SIZE_TYPE__")
      DW_AT_count  (0x0d)

  DW_TAG_subprogram
    DW_AT_low_pc        (0x0000000000000000)
    DW_AT_high_pc       (0x0000000000000014)
    DW_AT_frame_base    (DW_OP_reg6 RBP)
    DW_AT_linkage_name  ("_Z3foov")
    DW_AT_name          ("foo")
    DW_AT_decl_line     (3)
    DW_AT_external      (true)
```

DWARF debug information

CodeView debug information (llvm-pdbutil)

```
DW_TAG_compile_unit
  DW_AT_producer   ("clang")
  DW_AT_language   (DW_LANG_C_plus_plus_14)
  DW_AT_name       ("hello-world.cpp")
  DW_AT_low_pc     (0x0000000000000000)
  DW_AT_high_pc    (0x0000000000000014)
  DW_AT_addr_base  (0x00000008)

  DW_TAG_variable
    DW_AT_type      (0x0000002d "const char[13]")
    DW_AT_decl_line (4)
    DW_AT_location  (DW_OP_addrx 0x0)

  DW_TAG_array_type
    DW_AT_type     (0x00000039 "const char")

    DW_TAG_subrange_type
      DW_AT_type   (0x00000042 "__ARRAY_SIZE_TYPE__")
      DW_AT_count  (0x0d)

  DW_TAG_subprogram
    DW_AT_low_pc        (0x0000000000000000)
    DW_AT_high_pc       (0x0000000000000014)
    DW_AT_frame_base    (DW_OP_reg6 RBP)
    DW_AT_linkage_name  ("_Z3foov")
    DW_AT_name          ("foo")
    DW_AT_decl_line     (3)
    DW_AT_external      (true)
```

DWARF debug information

```
Types (.debug$T)
================
0x1000 | LF_ARGLIST [size = 8]
0x1001 | LF_PROCEDURE [size = 16]
         return type = 0x0003 (void), # args = 0,
param list = 0x1000
         calling conv = cdecl, options = None
0x1002 | LF_FUNC_ID [size = 16]
         name = foo, type = 0x1001, parent scope =
<no type>
0x1004 | LF_STRING_ID [size = 24] ID: <no type>,
String: hello-world.cpp

Symbols
=======
Mod 0000 | `.debug$S`:
       0 | S_OBJNAME [size = 64] sig=0, `hello-
world-clang-cv.o`
       0 | S_COMPILE3 [size = 156]
           machine = intel x86-x64, Ver = clang
version 21.0.0, language = c++
           frontend = 21.0.0.0  flags = none
       0 | S_GPROC32_ID [size = 44] `foo`
           parent = 0, end = 0, addr = 0000:0000
           type = `0x1002 (foo)`, debug start = 0,
debug end = 0, flags = noinline | opt debuginfo
       0 | S_FRAMEPROC [size = 32]
           size = 40, padding size = 0 padding = 0
           bytes of callee saved registers = 0,
exception handler addr = 0000:0000
           local fp reg = RSP, param fp reg = RSP
           flags = safe buffers
       0 | S_PROC_ID_END [size = 4]
       0 | S_BUILDINFO [size = 8] BuildId = `0x1008`
```

CodeView debug information

DWARF & CodeView canonical logical view

```
DW_TAG_compile_unit
  DW_AT_producer   ("clang")
  DW_AT_language   (DW_LANG_C_plus_plus_14)
  DW_AT_name       ("hello-world.cpp")
  DW_AT_low_pc     (0x0000000000000000)
  DW_AT_high_pc    (0x0000000000000014)
  DW_AT_addr_base  (0x00000008)

  DW_TAG_variable
    DW_AT_type      (0x0000002d "const char[13]")
    DW_AT_decl_line (4)
    DW_AT_location  (DW_OP_addrx 0x0)

  DW_TAG_array_type
    DW_AT_type     (0x00000039 "const char")

    DW_TAG_subrange_type
      DW_AT_type   (0x00000042 "__ARRAY_SIZE_TYPE__")
      DW_AT_count  (0x0d)

  DW_TAG_subprogram
    DW_AT_low_pc        (0x0000000000000000)
    DW_AT_high_pc       (0x0000000000000014)
    DW_AT_frame_base    (DW_OP_reg6 RBP)
    DW_AT_linkage_name  ("_Z3foov")
    DW_AT_name          ("foo")
    DW_AT_decl_line     (3)
    DW_AT_external      (true)
```

DWARF debug information

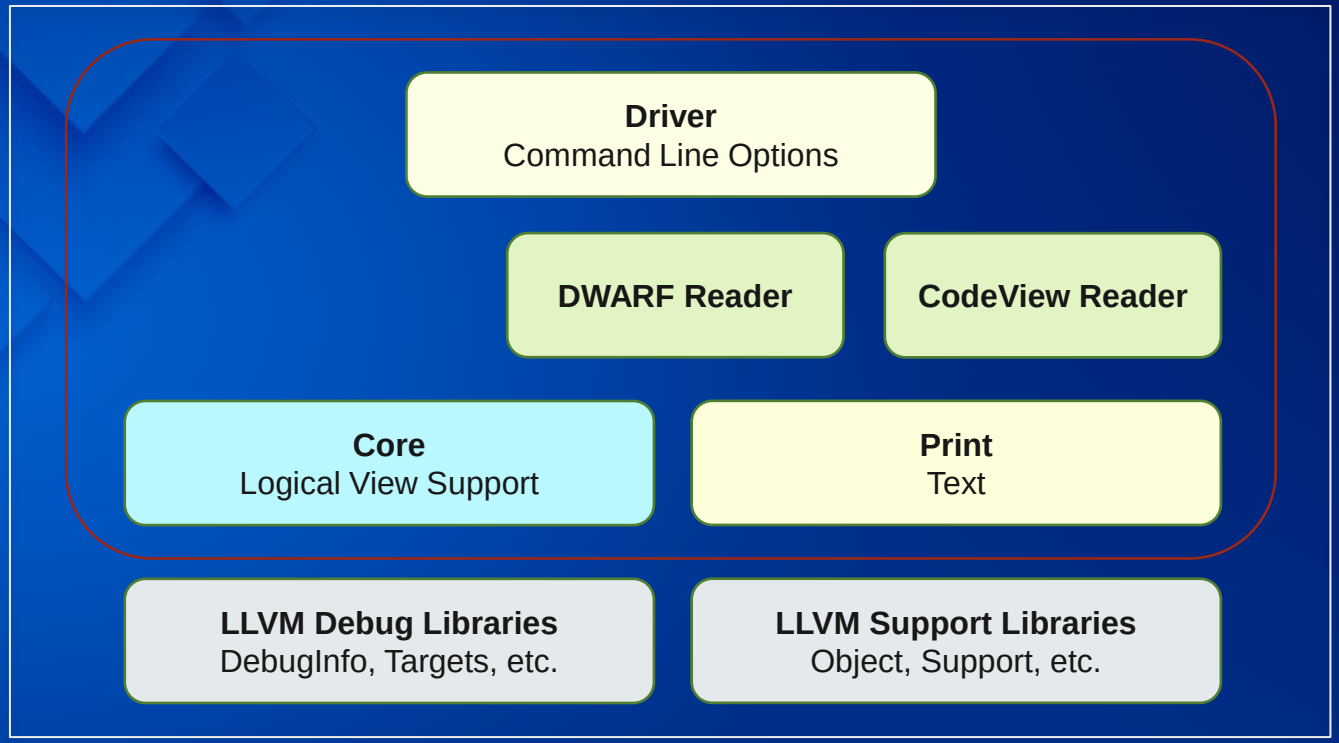
```
Logical View:
    {File} 'hello-world-clang.o'

      {CompileUnit} 'hello-world.cpp'
        {Array} 'const char [13]'
 3      {Function} not_inlined 'foo' -> 'void'
 3        {Line}
          {Code} 'pushq               %rbp'
          {Code} 'movq                %rsp, %rbp'
 4        {Line}
          {Code} 'leaq                (%rip), %rdi'
          {Code} 'movb                $0x0, %al'
          {Code} 'callq               0x0'
 5        {Line}
          {Code} 'popq                %rbp'
          {Code} 'retq'
 5        {Line}
 4      {Variable} '' -> 'const char [13]'
```

Canonical logical view

```
Types (.debug$T)
================
0x1000 | LF_ARGLIST [size = 8]
0x1001 | LF_PROCEDURE [size = 16]
         return type = 0x0003 (void), # args = 0,
param list = 0x1000
         calling conv = cdecl, options = None
0x1002 | LF_FUNC_ID [size = 16]
         name = foo, type = 0x1001, parent scope =
<no type>
0x1004 | LF_STRING_ID [size = 24] ID: <no type>,
String: hello-world.cpp

Symbols
=======
Mod 0000 | `.debug$S`:
       0 | S_OBJNAME [size = 64] sig=0, `hello-
world-clang-cv.o`
       0 | S_COMPILE3 [size = 156]
           machine = intel x86-x64, Ver = clang
version 21.0.0, language = c++
           frontend = 21.0.0.0  flags = none
       0 | S_GPROC32_ID [size = 44] `foo`
           parent = 0, end = 0, addr = 0000:0000
           type = `0x1002 (foo)`, debug start = 0,
debug end = 0, flags = noinline | opt debuginfo
       0 | S_FRAMEPROC [size = 32]
           size = 40, padding size = 0 padding = 0
           bytes of callee saved registers = 0,
exception handler addr = 0000:0000
           local fp reg = RSP, param fp reg = RSP
           flags = safe buffers
       0 | S_PROC_ID_END [size = 4]
       0 | S_BUILDINFO [size = 8] BuildId = `0x1008`
```

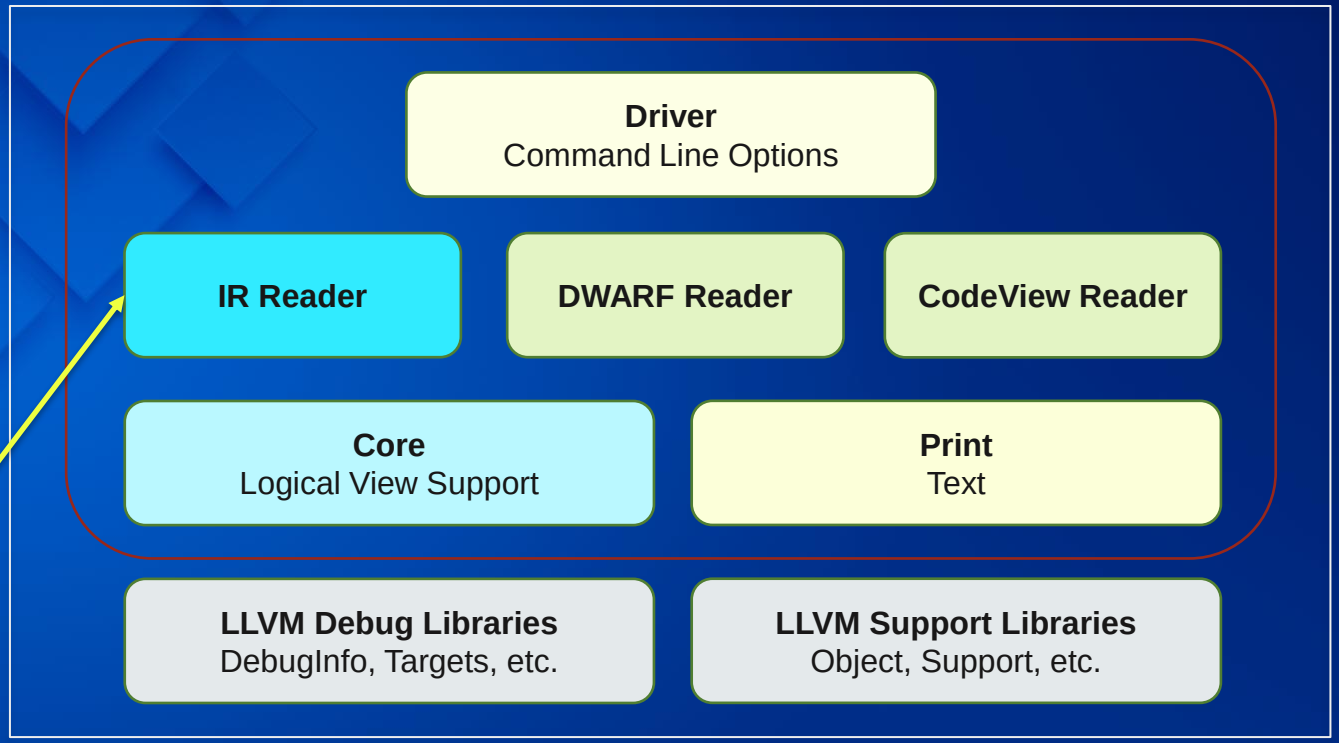
CodeView debug information

llvm-debuginfo-analyzer - current components



Current components

llvm-debuginfo-analyzer - IR reader component



IR Reader component

Print logical views

Common options to print logical views when dealing with IR

- --attribute=level
- --print=scopes,types,symbols,lines

Print logical views

Common options to print logical views when dealing with IR

- --attribute=level
- --print=scopes,types,symbols,lines

IR tests

- After Simplify CFG pass: simplify-cfg.ll
- After SLP Vectorizer pass: slp-vectorizer.ll

Print logical views

Common options to print logical views when dealing with IR

- --attribute=level
- --print=scopes,types,symbols,lines

IR tests

- After Simplify CFG pass: simplify-cfg.ll
- After SLP Vectorizer pass: slp-vectorizer.ll

llvm-debuginfo-analyzer command line

- --attribute=level --print=scopes,symbols,types simplify-cfg.ll
- --attribute=level --print=scopes,symbols,types slp-vectorizer.ll

Logical views for Simplify CFG and SLP Vectorizer passes

```
!llvm.dbg.cu = !{!0}
!llvm.module.flags = !{!5, !6, !7, !8, !9, !10, !11}
!llvm.ident = !{!12}
!0 = distinct !DICompileUnit(language: DW_LANG_C_plus_plus_14, file: !1, producer: "clang", isOptimized: false, runtimeVersion: 0,
emissionKind: FullDebug, retainedTypes: !2, splitDebugInlining: false, nameTableKind: None)
!1 = !DIFile(filename: "test.cpp", directory: "")
!2 = !{!3}
!3 = !DIDerivedType(tag: DW_TAG_typedef, name: "F32", file: !1, line: 7, baseType: !4)
!4 = !DIBasicType(name: "float", size: 32, encoding: DW_ATE_float)
!5 = !{i32 7, !"Dwarf Version", i32 5}
!6 = !{i32 2, !"Debug Info Version", i32 3}
!12 = !{!"clang"}
!13 = distinct !DISubprogram(name: "RandF32", linkageName: "_Z7RandF32v", scope: !1, file: !1, line: 13, type: !14, scopeLine: 13, flags:
DIFlagPrototyped, spFlags: DISPFlagDefinition, unit: !0, retainedNodes: !16)
!14 = !DISubroutineType(types: !15)
!15 = !{!4}
!16 = !{}
!17 = !DILocation(line: 14, column: 15, scope: !13)
!18 = !DILocalVariable(name: "uRand", scope: !13, file: !1, line: 14, type: !19)
!19 = !DIDerivedType(tag: DW_TAG_typedef, name: "U32", file: !1, line: 6, baseType: !20)
!20 = !DIBasicType(name: "unsigned int", size: 32, encoding: DW_ATE_unsigned)
!21 = !DILocation(line: 0, scope: !13)
!22 = !DILocation(line: 15, column: 21, scope: !13)
!23 = !DILocation(line: 15, column: 27, scope: !13)
!24 = !DILocalVariable(name: "fRand", scope: !13, file: !1, line: 15, type: !3)
!25 = !DILocation(line: 16, column: 3, scope: !13)
!26 = distinct !DISubprogram(name: "randGauss", linkageName: "_Z9randGaussPf", scope: !1, file: !1, line: 19, type: !27, scopeLine: 19,
flags: DIFlagPrototyped, spFlags: DISPFlagDefinition, unit: !0, retainedNodes: !16)
!27 = !DISubroutineType(types: !28)
!28 = !{null, !29}
!29 = !DIDerivedType(tag: DW_TAG_pointer_type, baseType: !4, size: 64)
!30 = !DILocalVariable(name: "work", arg: 1, scope: !26, file: !1, line: 19, type: !29)
!31 = !DILocation(line: 0, scope: !26)
!32 = !DILocation(line: 22, column: 3, scope: !26)
!33 = !DILocation(line: 23, column: 16, scope: !34)
!34 = distinct !DILexicalBlock(scope: !26, file: !1, line: 22, column: 6)
!35 = !DILocation(line: 23, column: 26, scope: !34)
!36 = !DILocalVariable(name: "x1", scope: !26, file: !1, line: 20, type: !4)
!37 = !DILocation(line: 24, column: 16, scope: !34)
!38 = !DILocation(line: 24, column: 26, scope: !34)
!39 = !DILocalVariable(name: "x2", scope: !26, file: !1, line: 20, type: !4)
!40 = !DILocation(line: 25, column: 22, scope: !34)
!41 = !DILocation(line: 25, column: 17, scope: !34)
!42 = !DILocalVariable(name: "w", scope: !26, file: !1, line: 20, type: !4)
!43 = !DILocation(line: 26, column: 14, scope: !26)
!44 = !DILocation(line: 26, column: 3, scope: !34)
!45 = distinct !{!45, !32, !46, !47}
!46 = !DILocation(line: 26, column: 20, scope: !26)
!47 = !{!"llvm.loop.mustprogress"}
!48 = !DILocation(line: 28, column: 20, scope: !26)
!49 = !DILocation(line: 28, column: 18, scope: !26)
!50 = !DILocation(line: 28, column: 30, scope: !26)
!51 = !DILocation(line: 28, column: 28, scope: !26)
!52 = !DILocation(line: 28, column: 12, scope: !26)
!53 = !DILocation(line: 28, column: 7, scope: !26)
!54 = !DILocation(line: 29, column: 16, scope: !26)
!55 = !DILocation(line: 29, column: 11, scope: !26)
!56 = !DILocation(line: 30, column: 16, scope: !26)
!57 = !DILocation(line: 30, column: 3, scope: !26)
!58 = !DILocation(line: 30, column: 11, scope: !26)
!59 = !DILocation(line: 31, column: 1, scope: !26)
```

IR metadata after Simplify CFG

Logical views for Simplify CFG and SLP Vectorizer passes

```
!llvm.dbg.cu = !{!0}
!llvm.module.flags = !{!5, !6, !7, !8, !9, !10, !11}
!llvm.ident = !{!12}
!0 = distinct !DICompileUnit(language: DW_LANG_C_plus_plus_14, file: !1, producer: "clang", isOptimized: false, runtimeVersion: 0,
emissionKind: FullDebug, retainedTypes: !2, splitDebugInlining: false, nameTableKind: None)
!1 = !DIFile(filename: "test.cpp", directory: "")
!2 = !{!3}
!3 = !DIDerivedType(tag: DW_TAG_typedef, name: "F32", file: !1, line: 7, baseType: !4)
!4 = !DIBasicType(name: "float", size: 32, encoding: DW_ATE_float)
!5 = !{i32 7, !"Dwarf Version", i32 5}
!6 = !{i32 2, !"Debug Info Version", i32 3}
!12 = !{!"clang"}
!13 = distinct !DISubprogram(name: "RandF32", linkageName: "_Z7RandF32v", scope: !1, file: !1, line: 13, type: !14, scopeLine: 13, flags:
DIFlagPrototyped, spFlags: DISPFlagDefinition, unit: !0, retainedNodes: !16)
!14 = !DISubroutineType(types: !15)
!15 = !{!4}
!16 = !{}
!17 = !DILocation(line: 14, column: 15, scope: !13)
!18 = !DILocalVariable(name: "uRand", scope: !13, file: !1, line: 14, type: !19)
!19 = !DIDerivedType(tag: DW_TAG_typedef, name: "U32", file: !1, line: 6, baseType: !20)
!20 = !DIBasicType(name: "unsigned int", size: 32, encoding: DW_ATE_unsigned)
!21 = !DILocation(line: 0, scope: !13)
!22 = !DILocation(line: 15, column: 21, scope: !13)
!23 = !DILocation(line: 15, column: 27, scope: !13)
!24 = !DILocalVariable(name: "fRand", scope: !13, file: !1, line: 15, type: !3)
!25 = !DILocation(line: 16, column: 3, scope: !13)
!26 = distinct !DISubprogram(name: "randGauss", linkageName: "_Z9randGaussPf", scope: !1, file: !1, line: 19, type: !27, scopeLine: 19,
flags: DIFlagPrototyped, spFlags: DISPFlagDefinition, unit: !0, retainedNodes: !16)
!27 = !DISubroutineType(types: !28)
!28 = !{null, !29}
!29 = !DIDerivedType(tag: DW_TAG_pointer_type, baseType: !4, size: 64)
!30 = !DILocalVariable(name: "work", arg: 1, scope: !26, file: !1, line: 19, type: !29)
!31 = !DILocation(line: 0, scope: !26)
!32 = !DILocation(line: 22, column: 3, scope: !26)
!33 = !DILocation(line: 23, column: 16, scope: !34)
!34 = distinct !DILexicalBlock(scope: !26, file: !1, line: 22, column: 6)
!35 = !DILocation(line: 23, column: 26, scope: !34)
!36 = !DILocalVariable(name: "x1", scope: !26, file: !1, line: 20, type: !4)
!37 = !DILocation(line: 24, column: 16, scope: !34)
!38 = !DILocation(line: 24, column: 26, scope: !34)
!39 = !DILocalVariable(name: "x2", scope: !26, file: !1, line: 20, type: !4)
!40 = !DILocation(line: 25, column: 22, scope: !34)
!41 = !DILocation(line: 25, column: 17, scope: !34)
!42 = !DILocalVariable(name: "w", scope: !26, file: !1, line: 20, type: !4)
!43 = !DILocation(line: 26, column: 14, scope: !26)
!44 = !DILocation(line: 26, column: 3, scope: !34)
!45 = distinct !{!45, !32, !46, !47}
!46 = !DILocation(line: 26, column: 20, scope: !26)
!47 = !{!"llvm.loop.mustprogress"}
!48 = !DILocation(line: 28, column: 20, scope: !26)
!49 = !DILocation(line: 28, column: 18, scope: !26)
!50 = !DILocation(line: 28, column: 30, scope: !26)
!51 = !DILocation(line: 28, column: 28, scope: !26)
!52 = !DILocation(line: 28, column: 12, scope: !26)
!53 = !DILocation(line: 28, column: 7, scope: !26)
!54 = !DILocation(line: 29, column: 16, scope: !26)
!55 = !DILocation(line: 29, column: 11, scope: !26)
!56 = !DILocation(line: 30, column: 16, scope: !26)
!57 = !DILocation(line: 30, column: 3, scope: !26)
!58 = !DILocation(line: 30, column: 11, scope: !26)
!59 = !DILocation(line: 31, column: 1, scope: !26)
```

IR metadata after Simplify CFG

```
Logical View:
[000]       {File} 'simplify-cfg.ll'

[001]         {CompileUnit} 'test.cpp'
[002]   6       {TypeAlias} 'U32' -> 'unsigned int'
[002]   7       {TypeAlias} 'F32' -> 'float'
[002]  13       {Function} 'RandF32' -> 'float'
[003]  14         {Variable} 'uRand' -> 'U32'
[003]  15         {Variable} 'fRand' -> 'F32'
[003]  14         {Line}
[003]  15         {Line}
[003]  15         {Line}
[003]  16         {Line}
[002]  19       {Function} 'randGauss' -> 'void'
[003]  19         {Parameter} 'work' -> '* float'
[003]  20         {Variable} 'w' -> 'float'
[003]  20         {Variable} 'x1' -> 'float'
[003]  20         {Variable} 'x2' -> 'float'
[003]  28         {Line}
[003]  29         {Line}
[003]  30         {Line}
[003]  30         {Line}
[003]  30         {Line}
[003]  31         {Line}
[003]  23         {Line}
[003]  24         {Line}
[003]  25         {Line}
```

Logical view after Simplify CFG

Logical views for Simplify CFG and SLP Vectorizer passes

```
!llvm.dbg.cu = !{!0}
!llvm.module.flags = !{!5, !6, !7, !8, !9, !10, !11}
!llvm.ident = !{!12}
!0 = distinct !DICompileUnit(language: DW_LANG_C_plus_plus_14, file: !1, producer: "clang", isOptimized: false, runtimeVersion: 0,
emissionKind: FullDebug, retainedTypes: !2, splitDebugInlining: false, nameTableKind: None)
!1 = !DIFile(filename: "test.cpp", directory: "")
!2 = !{!3}
!3 = !DIDerivedType(tag: DW_TAG_typedef, name: "F32", file: !1, line: 7, baseType: !4)
!4 = !DIBasicType(name: "float", size: 32, encoding: DW_ATE_float)
!5 = !{i32 7, !"Dwarf Version", i32 5}
!6 = !{i32 2, !"Debug Info Version", i32 3}
!12 = !{!"clang"}
!13 = distinct !DISubprogram(name: "RandF32", linkageName: "_Z7RandF32v", scope: !1, file: !1, line: 13, type: !14, scopeLine: 13, flags:
DIFlagPrototyped, spFlags: DISPFlagDefinition, unit: !0, retainedNodes: !16)
!14 = !DISubroutineType(types: !15)
!15 = !{!4}
!16 = !{}
!17 = !DILocation(line: 14, column: 15, scope: !13)
!18 = !DILocalVariable(name: "uRand", scope: !13, file: !1, line: 14, type: !19)
!19 = !DIDerivedType(tag: DW_TAG_typedef, name: "U32", file: !1, line: 6, baseType: !20)
!20 = !DIBasicType(name: "unsigned int", size: 32, encoding: DW_ATE_unsigned)
!21 = !DILocation(line: 0, scope: !13)
!22 = !DILocation(line: 15, column: 21, scope: !13)
!23 = !DILocation(line: 15, column: 27, scope: !13)
!24 = !DILocalVariable(name: "fRand", scope: !13, file: !1, line: 15, type: !3)
!25 = !DILocation(line: 16, column: 3, scope: !13)
!26 = distinct !DISubprogram(name: "randGauss", linkageName: "_Z9randGaussPf", scope: !1, file: !1, line: 19, type: !27, scopeLine: 19,
flags: DIFlagPrototyped, spFlags: DISPFlagDefinition, unit: !0, retainedNodes: !16)
!27 = !DISubroutineType(types: !28)
!28 = !{null, !29}
!29 = !DIDerivedType(tag: DW_TAG_pointer_type, baseType: !4, size: 64)
!30 = !DILocalVariable(name: "work", arg: 1, scope: !26, file: !1, line: 19, type: !29)
!31 = !DILocation(line: 0, scope: !26)
!32 = !DILocation(line: 22, column: 3, scope: !26)
!33 = !DILocation(line: 23, column: 16, scope: !34)
!34 = distinct !DILexicalBlock(scope: !26, file: !1, line: 22, column: 6)
!35 = !DILocation(line: 23, column: 26, scope: !34)
!36 = !DILocalVariable(name: "x1", scope: !26, file: !1, line: 20, type: !4)
!37 = !DILocation(line: 24, column: 16, scope: !34)
!38 = !DILocation(line: 24, column: 26, scope: !34)
!39 = !DILocalVariable(name: "x2", scope: !26, file: !1, line: 20, type: !4)
!40 = !DILocation(line: 25, column: 22, scope: !34)
!41 = !DILocation(line: 25, column: 17, scope: !34)
!42 = !DILocalVariable(name: "w", scope: !26, file: !1, line: 20, type: !4)
!43 = !DILocation(line: 26, column: 14, scope: !26)
!44 = !DILocation(line: 26, column: 3, scope: !34)
!45 = distinct !{!45, !32, !46, !47}
!46 = !DILocation(line: 26, column: 20, scope: !26)
!47 = !{!"llvm.loop.mustprogress"}
!48 = !DILocation(line: 28, column: 20, scope: !26)
!49 = !DILocation(line: 28, column: 18, scope: !26)
!50 = !DILocation(line: 28, column: 30, scope: !26)
!51 = !DILocation(line: 28, column: 28, scope: !26)
!52 = !DILocation(line: 28, column: 12, scope: !26)
!53 = !DILocation(line: 28, column: 7, scope: !26)
!54 = !DILocation(line: 29, column: 16, scope: !26)
!55 = !DILocation(line: 29, column: 11, scope: !26)
!56 = !DILocation(line: 30, column: 16, scope: !26)
!57 = !DILocation(line: 30, column: 3, scope: !26)
!58 = !DILocation(line: 30, column: 11, scope: !26)
!59 = !DILocation(line: 31, column: 1, scope: !26)
```

IR metadata after Simplify CFG

```
Logical View:
[000]       {File} ‘simplify-cfg.ll'

[001]         {CompileUnit} ‘test.cpp'
[002]   6       {TypeAlias} 'U32' -> 'unsigned int'
[002]   7       {TypeAlias} 'F32' -> 'float'
[002]  13       {Function} 'RandF32' -> 'float'
[003]  14         {Variable} 'uRand' -> 'U32'
[003]  15         {Variable} 'fRand' -> 'F32'
[003]  14         {Line}
[003]  15         {Line}
[003]  15         {Line}
[003]  16         {Line}
[002]  19       {Function} 'randGauss' -> 'void'
[003]  19         {Parameter} 'work' -> '* float'
[003]  20         {Variable} 'w' -> 'float'
[003]  20         {Variable} 'x1' -> 'float'
[003]  20         {Variable} 'x2' -> 'float'
[003]  28         {Line}
[003]  29         {Line}
[003]  30         {Line}
[003]  30         {Line}
[003]  30         {Line}
[003]  31         {Line}
[003]  23         {Line}
[003]  24         {Line}
[003]  25         {Line}
```

Logical view after Simplify CFG

```
Logical View:
[000]       {File} ‘slp-vectorizer.ll'

[001]         {CompileUnit} ‘test.cpp'
[002]   6       {TypeAlias} 'U32' -> 'unsigned int'
[002]   7       {TypeAlias} 'F32' -> 'float'
[002]  13       {Function} 'RandF32' -> 'float'
[003]  14         {Variable} 'uRand' -> 'U32'
[003]  15         {Variable} 'fRand' -> 'F32'
[003]  14         {Line}
[003]  15         {Line}
[003]  15         {Line}
[003]  16         {Line}
[002]  19       {Function} 'randGauss' -> 'void'
[003]  19         {Parameter} 'work' -> '* float'
[003]  20         {Variable} 'w' -> 'float'
[003]  28         {Line}
[003]  29         {Line}
[003]  29         {Line}
[003]  29         {Line}
[003]  31         {Line}
[003]  23         {Line}
[003]  23         {Line}
[003]  23         {Line}
[003]  25         {Line}
[003]  25         {Line}
```

Logical view after SLP Vectorizer

Compare logical views

Common options to compare logical views when dealing with IR

- --report=list --report=view
- --print=scopes,types,symbols,lines

Compare logical views

Common options to compare logical views when dealing with IR

- --report=list --report=view
- --print=scopes,types,symbols,lines

IR tests

- After Simplify CFG pass: simplify-cfg.ll
- After SLP Vectorizer pass: slp-vectorizer.ll

Compare logical views

Common options to compare logical views when dealing with IR

- --report=list --report=view
- --print=scopes,types,symbols,lines

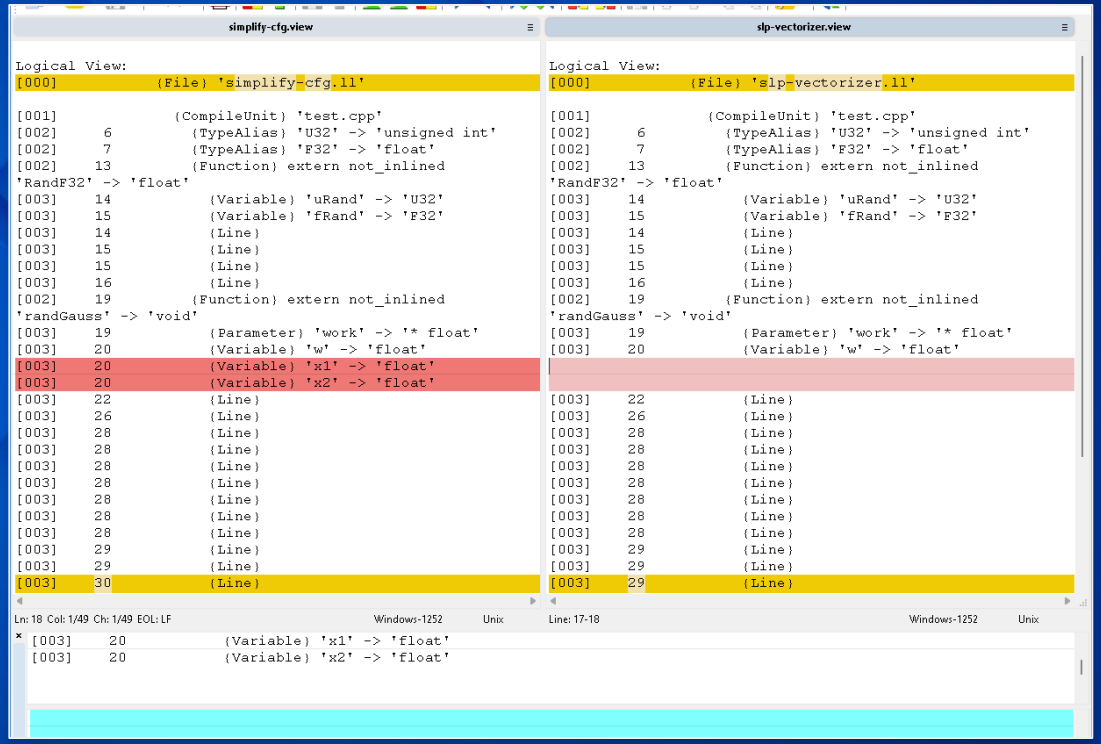
IR tests

- After Simplify CFG pass: simplify-cfg.ll
- After SLP Vectorizer pass: slp-vectorizer.ll

llvm-debuginfo-analyzer command line

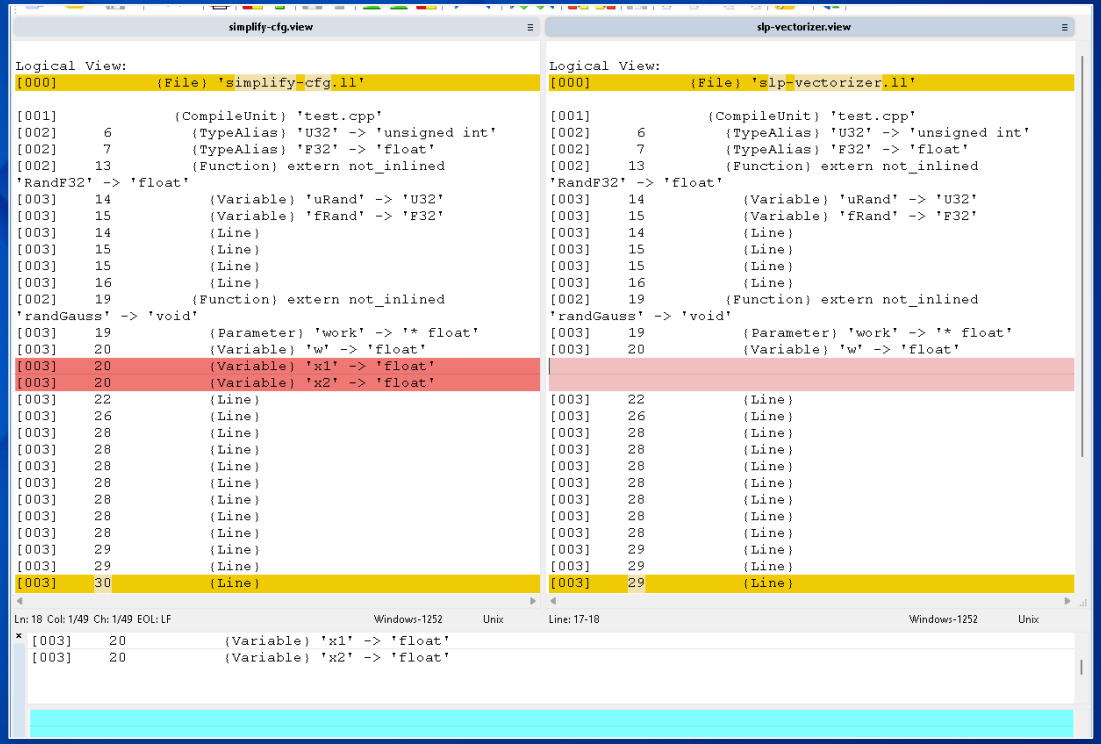
- --compare=symbols,lines --report=list --print=symbols,lines simplify-cfg.ll slp-vectorizer.ll
- --compare=symbols,lines --report=view --print=symbols simplify-cfg.ll slp-vectorizer.ll

Logical view changes - comparison tool



IR changes: comparison tool

Logical view changes - built-in compare (report mode)

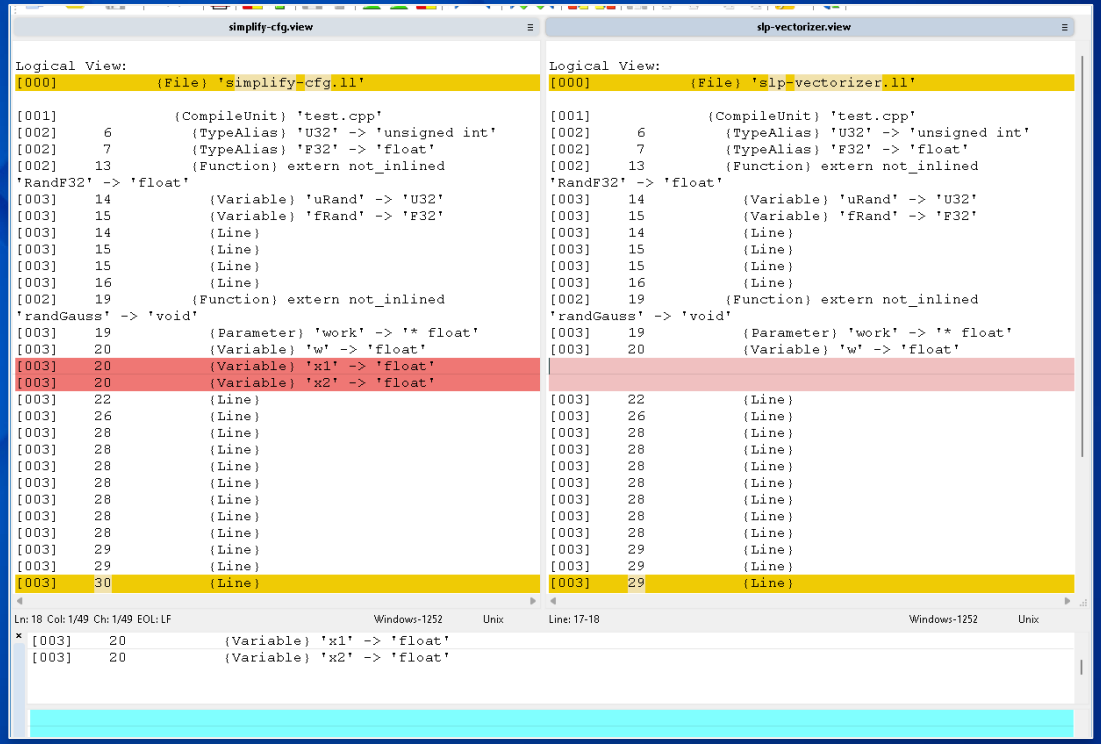


IR changes: comparison tool

```
Reference: ‘simplify-cfg.ll'
Target:    ‘slp-vectorizer.ll'
```

IR changes: llvm-debuginfo-analyzer

Logical view changes - built-in compare (report mode)

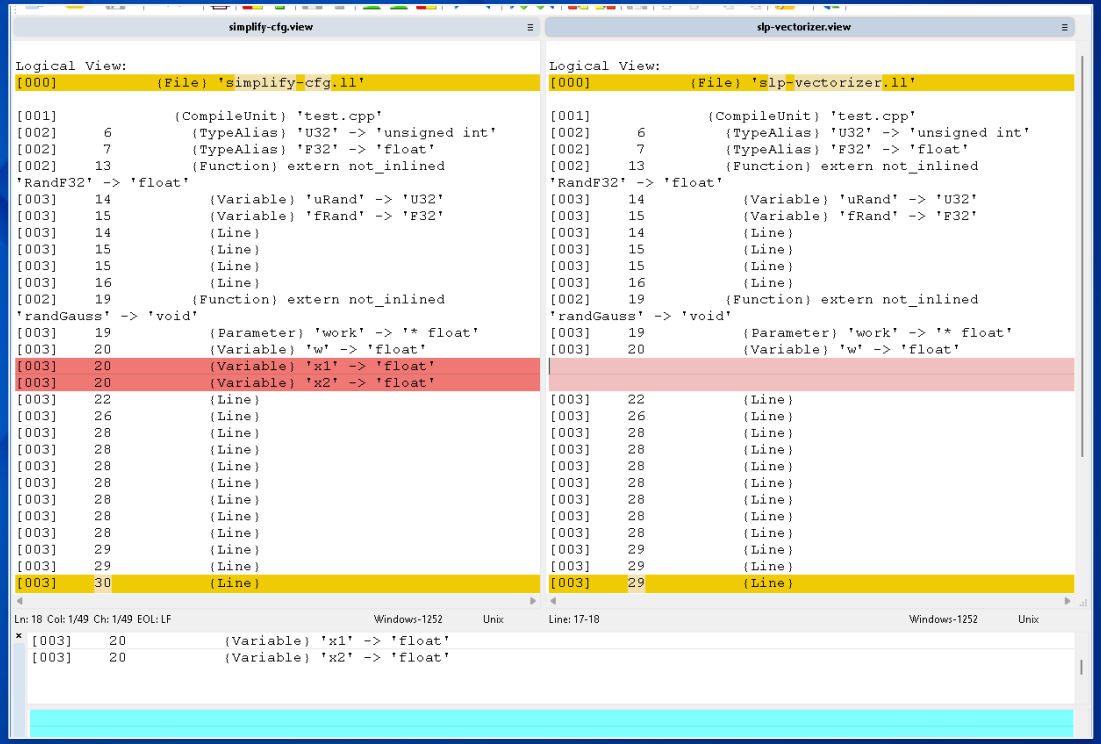


IR changes: comparison tool

```
Reference: ‘simplify-cfg.ll'
Target:    ‘slp-vectorizer.ll'

(2) Missing Symbols:
-    20     {Variable} 'x1' -> 'float'
-    20     {Variable} 'x2' -> 'float'
```

IR changes: llvm-debuginfo-analyzer

Logical view changes - built-in compare (report mode)

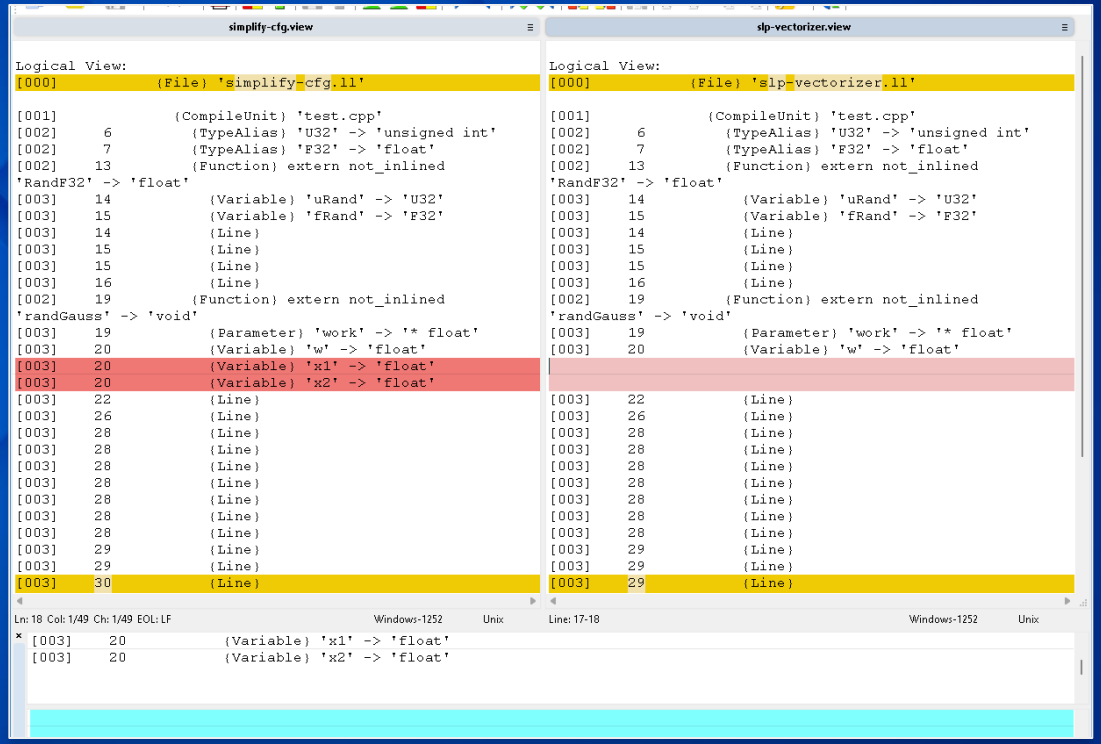


IR changes: comparison tool

```
Reference: ‘simplify-cfg.ll'
Target:    ‘slp-vectorizer.ll'

(2) Missing Symbols:
-    20     {Variable} 'x1' -> 'float'
-    20     {Variable} 'x2' -> 'float'

(5) Missing Lines:
-    24     {Line}
-    30     {Line}
-    30     {Line}
-    30     {Line}
-    24     {Line}
```

IR changes: llvm-debuginfo-analyzer

Logical view changes - built-in compare (view mode)



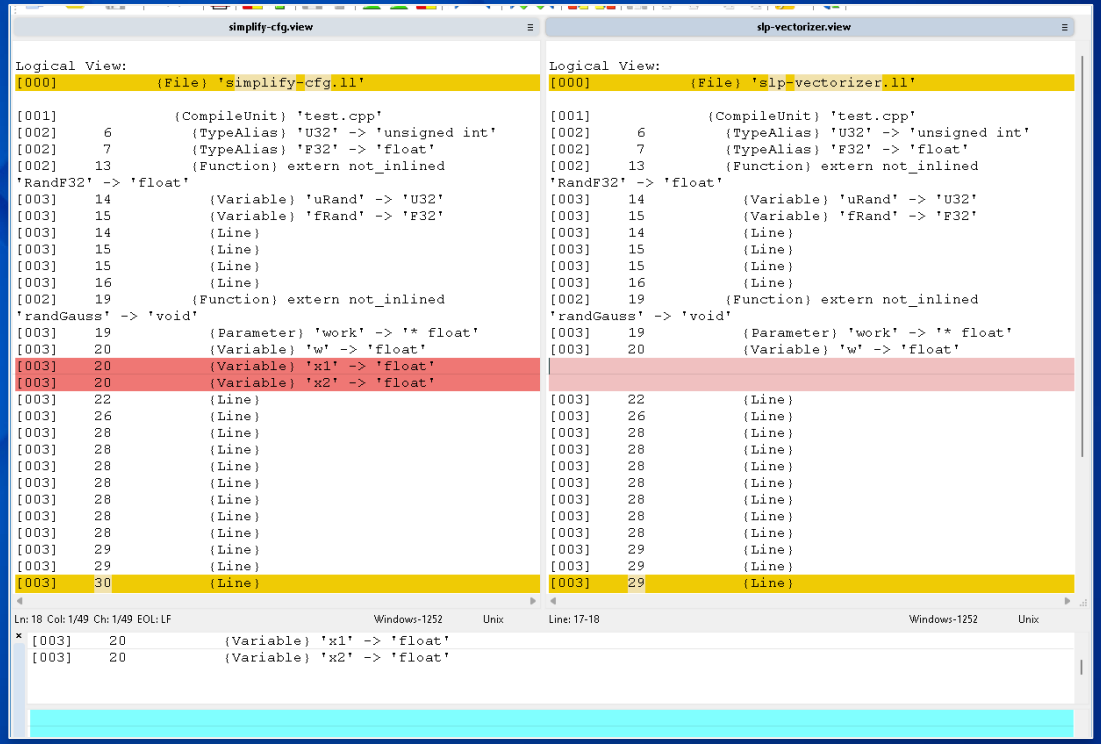
IR changes: comparison tool

```
Reference: ‘simplify-cfg.ll'
Target:    ‘slp-vectorizer.ll'

(2) Missing Symbols:
-    20     {Variable} 'x1' -> 'float'
-    20     {Variable} 'x2' -> 'float'

(5) Missing Lines:
-    24     {Line}
-    30     {Line}
-    30     {Line}
-    30     {Line}
-    24     {Line}
```

```
Reference: ‘simplify-cfg.ll'
Target:    ‘slp-vectorizer.ll'
```

IR changes: llvm-debuginfo-analyzer

Logical view changes - built-in compare (view mode)



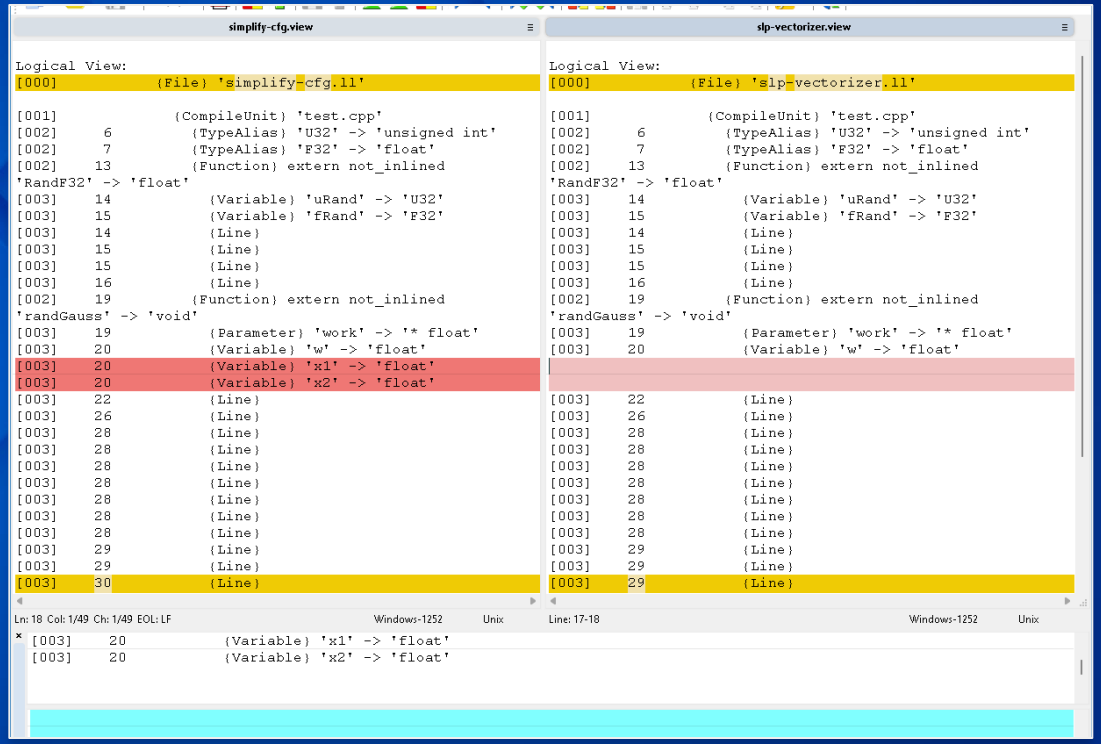
IR changes: comparison tool

```
Reference: ‘simplify-cfg.ll'
Target:    ‘slp-vectorizer.ll'

(2) Missing Symbols:
-    20     {Variable} 'x1' -> 'float'
-    20     {Variable} 'x2' -> 'float'

(5) Missing Lines:
-    24     {Line}
-    30     {Line}
-    30     {Line}
-    30     {Line}
-    24     {Line}
```

```
Reference: ‘simplify-cfg.ll'
Target:    ‘slp-vectorizer.ll'

Logical View:
           {File} ‘simplify-cfg.ll'

             {CompileUnit} ‘test.cpp'
     13        {Function} extern not_inlined 'RandF32' -> 'float'
     14          {Variable} 'uRand' -> 'U32'
     15          {Variable} 'fRand' -> 'F32'
     19        {Function} extern not_inlined 'randGauss' -> 'void'
     19          {Parameter} 'work' -> '* float'
     20          {Variable} 'w' -> 'float'
```

IR changes: llvm-debuginfo-analyzer

Logical view changes - built-in compare (view mode)



IR changes: comparison tool

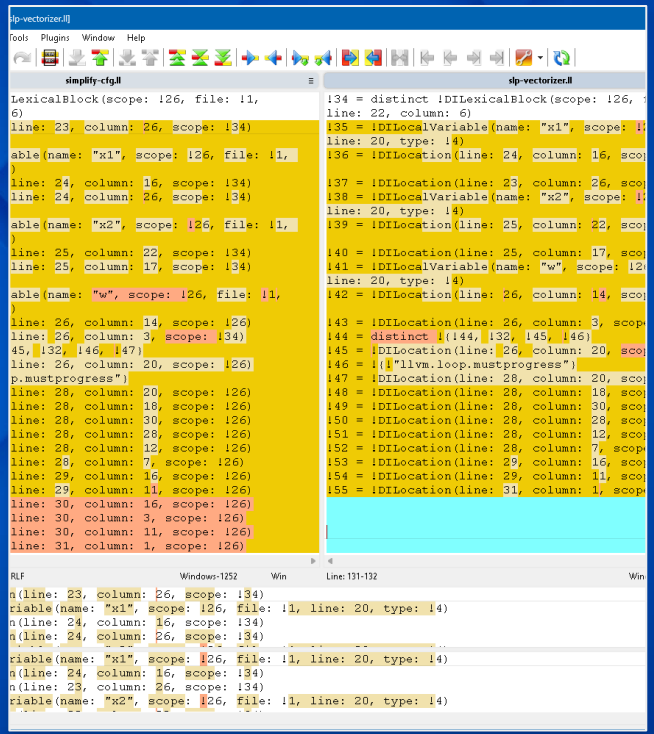
```
Reference: ‘simplify-cfg.ll'
Target:    ‘slp-vectorizer.ll'

(2) Missing Symbols:
-    20     {Variable} 'x1' -> 'float'
-    20     {Variable} 'x2' -> 'float'

(5) Missing Lines:
-    24     {Line}
-    30     {Line}
-    30     {Line}
-    30     {Line}
-    24     {Line}
```

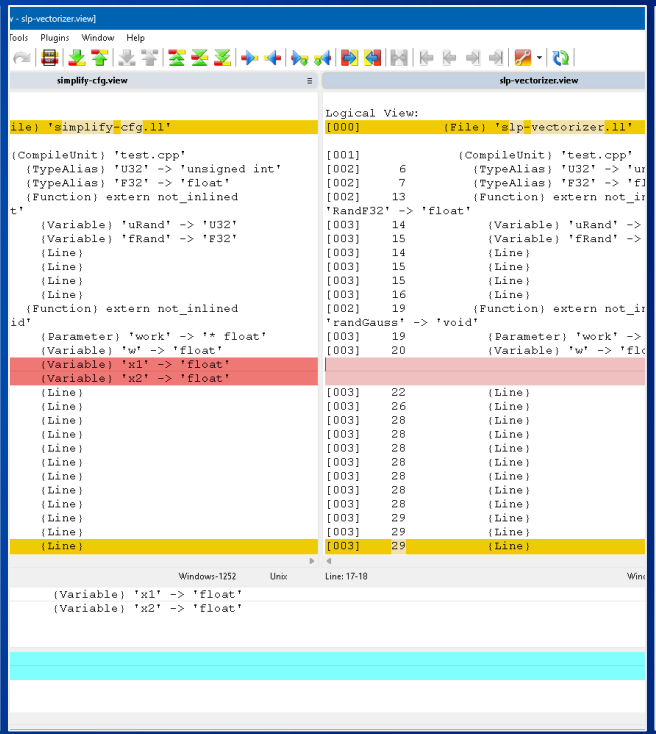
```
Reference: ‘simplify-cfg.ll'
Target:    ‘slp-vectorizer.ll'

Logical View:
           {File} ‘simplify-cfg.ll'

             {CompileUnit} ‘test.cpp'
     13        {Function} extern not_inlined 'RandF32' -> 'float'
     14          {Variable} 'uRand' -> 'U32'
     15          {Variable} 'fRand' -> 'F32'
     19        {Function} extern not_inlined 'randGauss' -> 'void'
     19          {Parameter} 'work' -> '* float'
     20          {Variable} 'w' -> 'float'
-    20          {Variable} 'x1' -> 'float'
-    20          {Variable} 'x2' -> 'float'
```

IR changes: llvm-debuginfo-analyzer

Conclusion

Reduce the noisiness of comparing the debuginfo in LLVM IR





IR changes: comparison tool

```
Reference: ‘simplify-cfg.ll'
Target:    ‘slp-vectorizer.ll'

(2) Missing Symbols:
-    20      {Variable} 'x1' -> 'float'
-    20      {Variable} 'x2' -> 'float'

(5) Missing Lines:
-    24      {Line}
-    30      {Line}
-    30      {Line}
-    30      {Line}
-    24      {Line}
```

```
Reference: ‘simplify-cfg.ll'
Target:    ‘slp-vectorizer.ll'

Logical View:
            {File} ‘simplify-cfg.ll'

              {CompileUnit} ‘test.cpp'
      13        {Function}  'RandF32' -> 'float'
      14          {Variable} 'uRand' -> 'U32'
      15          {Variable} 'fRand' -> 'F32'
      19        {Function} 'randGauss' -> 'void'
      19          {Parameter} 'work' -> '* float'
      20          {Variable} 'w' -> 'float'
-     20          {Variable} 'x1' -> 'float'
-     20          {Variable} 'x2' -> 'float'
```

IR changes: llvm-debuginfo-analyzer

Sony Interactive Entertainment | PlayStation

Thank you!