RT - RK
COMPUTER BASED SYSTEMS

Debug info in optimized code – how far can we go?

Improving LLVM debug info with function entry values

RT-RK LLC
Speakers:
Djordje Todorovic djordje.todorovic@rt-rk.com
Nikola Prica nikola.prica@rt-rk.com

RT-RK (in brief)

- Software company specialized in system software and embedded systems
- Part of a group working on compilers and tools (GCC, binutils, LLVM, LuaJIT, Valgrind, v8, libART, Go, ...)
- Working on LLVM since 2010
- Working on debug related issues for the last two years

RT-RK
COMPUTER BASED SYSTEMS

- Debugging software release products
- Finding parameters values in parent's frame
- DWARF 5 extensions
- IR call site representation
- Transition from IR to MIR
- Backend call site representation
- Measurements
- Conclusions

RT-RK HEADQUARTER

RT-RK
COMPUTER BASED SYSTEMS

Debugging software release products

Finding parameters values in parent's frame

DWARF 5 extensions

IR call site representation

Transition from IR to MIR

Backend call site representation

Measurements

Conclusions

RT-RK HEADQUARTER

Debugging of software release products

RT-RK
COMPUTER BASED SYSTEMS

- Release mode
- Optimizations levels
- Debug info due to optimizations
- Variable’s life

RT-RK
COMPUTER BASED SYSTEMS

Debugging software release products

Finding parameters values in parent's frame

DWARF 5 extensions

IR call site representation

Transition from IR to MIR

Backend call site representation

Measurements

Conclusions

RT-RK HEADQUARTER

Finding parameters values in parent's frame

Example

- Example with no entry values

```
(gdb) b fn2
Breakpoint 1 at 0x400563: file ./ex.h, line 8.
(gdb) r
Starting program: /nobackup/djtodoro/llvm_trunk/tests/a.out
7 7

Breakpoint 1, fn2 (a=7) at ./ex.h:8
8           printf("The num is %d\n", a);
(gdb) bt
#0  fn2 (a=7) at ./ex.h:8
#1  0x000000000040058b in fn1 (x=<optimized out>, y=7) at new.c:22
#2  0x00000000004005b1 in main () at new.c:33
(gdb) f 1
#1  0x000000000040058b in fn1 (x=<optimized out>, y=7) at new.c:22
22          fn2 (a);
(gdb) p x
$1 = <optimized out>
(gdb) f 2
#2  0x00000000004005b1 in main () at new.c:33
33          fn1 (1, k);
(gdb) p 1
$2 = 7
```

Finding parameters values in parent's frame

Example

- Automate the process in debagger
- Example with entry values

```
(gdb) b fn2
Breakpoint 1 at 0x400563: file ./ex.h, line 8.
(gdb) r
Starting program: /nobackup/djtodoro/llvm_trunk/tests/a.out
7 7

Breakpoint 1, fn2 (a=a@entry=7) at ./ex.h:8
8           printf("The num is %d\n", a);
(gdb) bt
#0  fn2 (a=a@entry=7) at ./ex.h:8
#1  0x000000000040058b in fn1 (x=x@entry=7, y=y@entry=7) at new.c:22
#2  0x00000000004005b1 in main () at new.c:33
```

Finding parameters values in parent's frame

Motivation

RT-RK
COMPUTER BASED SYSTEMS

- By implementing this feature in LLVM/Clang we have noticed that the number of function parameters with fully covered debug location range within its scope grows up to 15%

RT-RK
COMPUTER BASED SYSTEMS

Debugging software release products

Finding parameters values in parent's frame

DWARF 5 extensions

IR call site representation

Transition from IR to MIR

Backend call site representation

Measurements

Conclusions

RT-RK HEADQUARTER

DWARF 5 extensions

- DWARF tags:
  - DW_TAG_call_site
  - DW_TAG_call_site_parameter
  - …
- DWARF attributes:
  - DW_AT_call_pc
  - DW_AT_call_origin
  - DW_AT_call_target
  - DW_AT_call_value
  - …
- DWARF operand:
  - DW_OP_entry_value
- Implemented in GCC and GNU GDB since 2011 (it was GNU extension initially).

DWARF 5 extensions

Example of DWARF DIE

```
…
<2><b8>: Abbrev Number: 8 (DW_TAG_call_site)
   <b9>   DW_AT_call_pc      : 0x40051b
   <c1>   DW_AT_call_origin: <0x2a>
 <3><c5>: Abbrev Number: 9 (DW_TAG_call_site_parameter)
   <c6>   DW_AT_location    : 1 byte block: 55        (DW_OP_reg5 (rdi))
   <c8>   DW_AT_call_value: 1 byte block: 37  (DW_OP_lit7)
…
```

DWARF 5 extensions

Example of an entry in .debug_loc section

- Entry value can be used even for reporting value of the variable in the places it is not live, if the function parameter has unchanged value thought the course of the function
- Example of the entry in .debug_loc section

…

xx yy (DW_OP_reg5 (rdi))

yy zz (DW_OP_entry_value: (DW_OP_reg5 (rdi)); DW_OP_stack_value)

…

RT - RK
COMPUTER BASED SYSTEMS

- Debugging software release products
- Finding parameters values in parent's frame
- DWARF 5 extensions
- IR call site representation
- Transition from IR to MIR
- Backend call site representation
- Measurements
- Conclusions

RT-RK HEADQUARTER

IR call site representation

- DIMetadata resemblance to DWARF Tags
- DICallSite and DICallSiteParam
- Variable constness tracking

```
extern int foo(int);
int baa(int a) {
  if (a > 0)
    return foo(a + 3);
  return foo (a * 10);
}
```

```
if.then:                                          ; preds = %entry
  %1 = load i32, i32* %a.addr, align 4, !dbg !28, !tbaa !20
  %add = add nsw i32 %1, 3, !dbg !29
  %call = call i32 @foo(i32 %add), !dbg !30, !call_site !12
  store i32 %call, i32* %retval, align 4, !dbg !31
  br label %return, !dbg !31

if.end:                                           ; preds = %entry
  %2 = load i32, i32* %a.addr, align 4, !dbg !32, !tbaa !20
  %mul = mul nsw i32 %2, 10, !dbg !33
  %call1 = call i32 @foo(i32 %mul), !dbg !34, !call_site !17
  store i32 %call1, i32* %retval, align 4, !dbg !35
  br label %return, !dbg !35
```

```
!12 = !DICallSite(scope: !13, file: !1, parameters: !14, line: 4, calledSubprogram: !16)
!14 = !{!15}
!15 = !DICallSiteParam(argno: 1, variable: !11, expr: !DIExpression(DW_OP_lit3, DW_OP_plus))
!16 = !DISubprogram(name: "foo", scope: !1, file: !1, line: 1, isLocal: false, isDefinition: false, flags:
DIFlagPrototyped, isOptimized: true, elements: !2)
!17 = !DICallSite(scope: !6, file: !1, parameters: !18, line: 5, calledSubprogram: !16)
!18 = !{!19}
!19 = !DICallSiteParam(argno: 1, variable: !11, expr: !DIExpression(DW_OP_lit10, DW_OP_mul))
```

IR call site representation

- Benefits:
  - Resemblance to DWARF tags
  - Additional backup representation of call site parameters
  - DW_TAG_call_site_param can have DW_OP_entry_value expression
- Limitations:
  - Need for representing multiple variable expression
  - No support for parameter that is function call
  - No easy way to represent address of a variable
  - Change of variables creation interface

RT-RK
COMPUTER BASED SYSTEMS

- Debugging software release products
- Finding parameters values in parent's frame
- DWARF 5 extensions
- IR call site representation
- Transition from IR to MIR
- Backend call site representation
- Measurements
- Conclusions

RT-RK HEADQUARTER

Transition from IR to MIR

- Collecting of call site information after call lowering.

```
$physreg1 = COPY %vreg1
$physreg2 = COPY %vreg2
call foo
```

- General parameter matching algorithm over Call lowering chain nodes:
  - CALLSEQ_START and CALLSEQ_END
  - CopyToReg
  - Target depended issues for call sequence
  - Matching argument nodes to CopyToReg source nodes
- Emitting DBG_CALLSITE and DBG_CALLSITEPARAM

RT-RK
COMPUTER BASED SYSTEMS

- Debugging software release products
- Finding parameters values in parent's frame
- DWARF 5 extensions
- IR call site representation
- Transition from IR to MIR
- Backend call site representation
- Measurements
- Conclusions

RT-RK HEADQUARTER

Backend call site representation

```
extern int foo(int ,int , int);
extern int gaa(int);
int baa(int a, int b) {
  int c = 4;
  int d = 33;
  c = gaa(c);
  d = gaa(d);
  return foo(c, d + 3, -31);
}
```

```
%EDI<def> = MOV32ri 4;
CALL64pcrel32 <ga:@gaa>, <regmask ...>, %RSP, %EDI, %RSP, %EAX;
DBG_CALLSITE 0, %noreg, <!15>;
  * DBG_CALLSITEPARAM %EDI, "c", 4, %noreg;
%EBX<def> = MOV32rr %EAX;
%EDI<def> = MOV32ri 33;
CALL64pcrel32 <ga:@gaa>, <regmask ...>, %RSP, %EAX;
DBG_CALLSITE 0, %noreg, <!19>;
  * DBG_CALLSITEPARAM %EDI, "d", 33, %noreg;
%EAX<def> = KILL %EAX, %RAX;
%ESI<def> = LEA64_32r %RAX<kill>, 1, %noreg, 3, %noreg;
%EDX<def> = MOV32ri -31;
%EDI<def> = MOV32rr %EBX<kill>;
%RBX<def> = POP64r %RSP<imp-def>, %RSP<imp-use>; flags: FrameDestroy
TAILJMPd64 <ga:@foo>, <regmask ...>, %RSP, %RSP, %EDI, %ESI, %EDX;
DBG_CALLSITE 1, %noreg, <!22>;
  * DBG_CALLSITEPARAM %EDI, "c", %EBX, %noreg;
  * DBG_CALLSITEPARAM %EDX,  !DIExpression(DW_OP_lit31, DW_OP_neg) , 4294967265,
%noreg;
  * DBG_CALLSITEPARAM %ESI, "d" !DIExpression(DW_OP_lit3, DW_OP_plus) ,
                                  %RAX, 0, <!DIExpression(DW_OP_constu, 3, DW_OP_plus)>;
```

Backend call site representation

RT-RK
COMPUTER BASED SYSTEMS

- PrologueEpilogueInsrter, RegisterAllocation, SplitKit, Virutal Register Rewriter, LiveDebugValues…
- Emitting of DW_OP_entry_value in LiveDebugValues
- Adjustment of DBG_CALLSITEPARAM in LiveDebugValues
- Producing dwarf tags is handled similarly as DBG_VALUES

RT-RK
COMPUTER BASED SYSTEMS

Debugging software release products

Finding parameters values in parent's frame

DWARF 5 extensions

IR call site representation

Transition from IR to MIR

Backend call site representation

Measurements

Conclusions

RT-RK HEADQUARTER

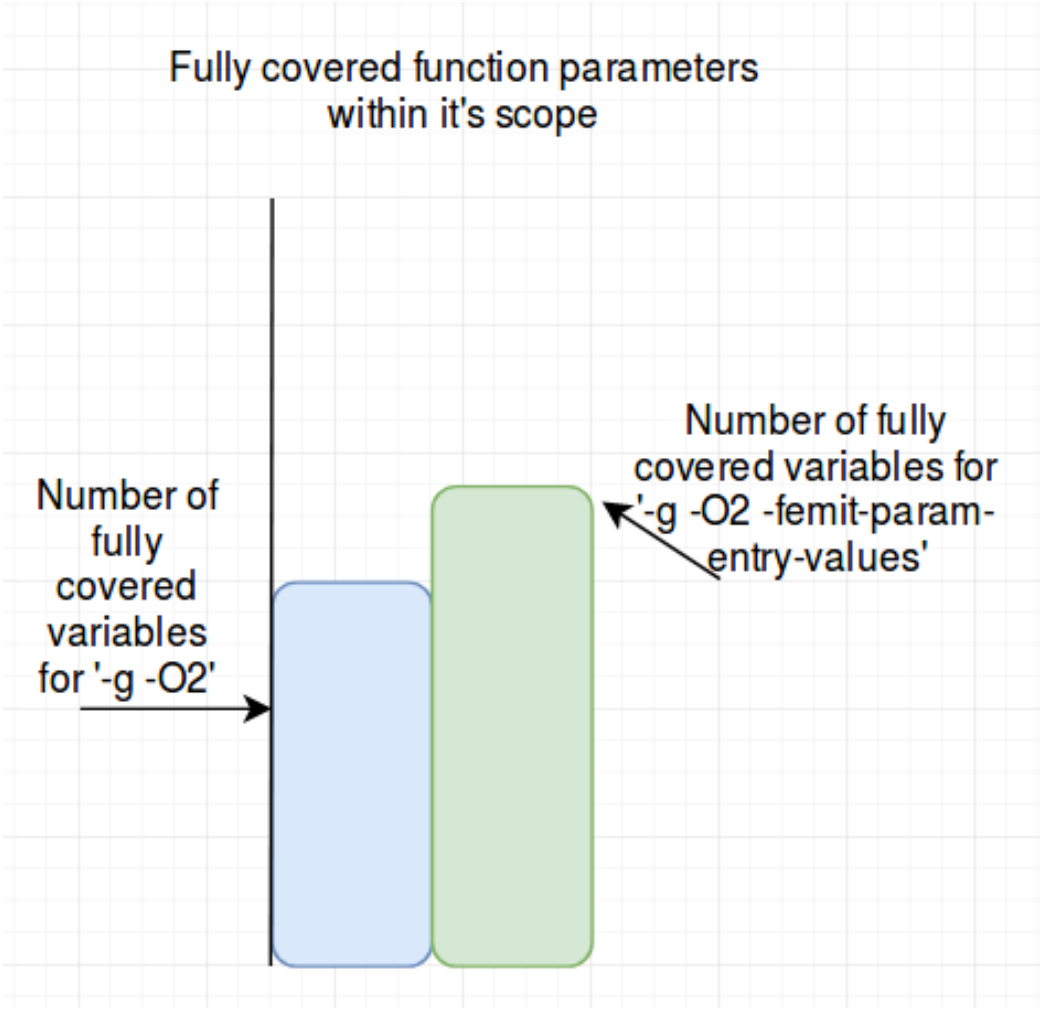
Measurements

- Built gdb-7.11 and SPEC 2006 benchmark for X86
- Used locstats tool from elfutils
- Increased debug location coverage
- Increased size of .debug sections
- No change in .text, .data, .bss
- -O2 and -O3 level have very similar results

Measurements

GDB 7.11

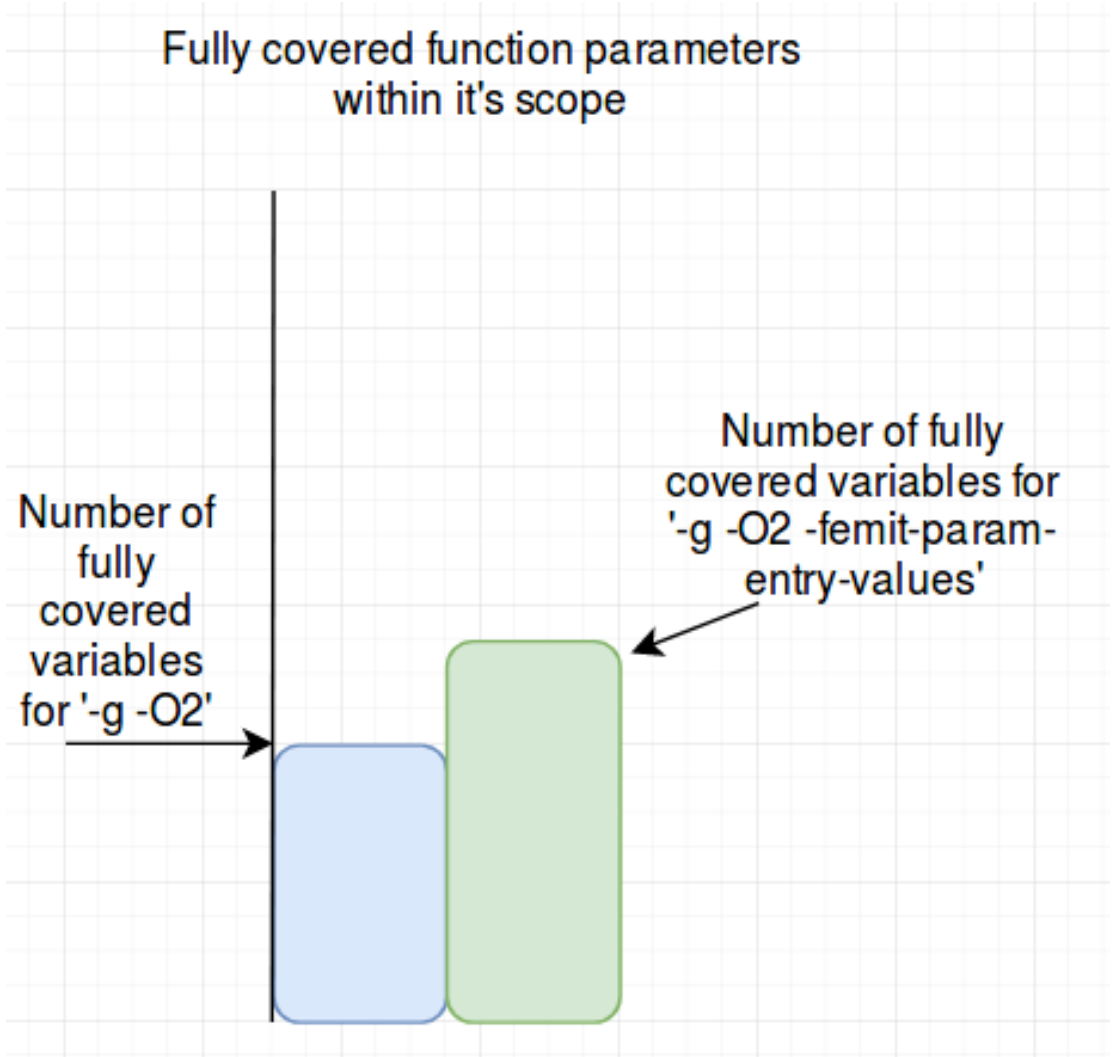
- Released version of GNU GDB 7.11
- Increment of 15% for function parameters (about 17000 variables more)
  with fully covered debug location of its scope.
- Average coverage per variable increased from 51.5% up to 61.3%
- Size incremented from 34.6M to 39.4M which is about 14%
- Increment only in .debug sections
- Build time increased for about 2%



Measurements

SPEC 2006 benchmark (483.xalancbmk package)

- SPEC 2006 benchmark is industry-standardized benchmark suite that stressing a system's processor, memory subsystem and compiler
- Increment of 8% for function parameters (about 12000 variables more) with fully covered debug location of its scope
- Average coverage per variable increased from 47.2% up to 51.4%
- Size incremented about 15%
- Increment only in .debug sections
- Build time increased for about 1%



RT-RK
COMPUTER BASED SYSTEMS

Debugging software release products

Finding parameters values in parent's frame

DWARF 5 extensions

IR call site representation

Transition from IR to MIR

Backend call site representation

Measurements

Conclusions

RT-RK HEADQUARTER

Conclusions

RT-RK
COMPUTER BASED SYSTEMS

- After finishing this feature in LLVM we have identified two main spots in our implementation that can be subject for further discussion and re-implementation:
  - Usefulness of DICallSitePram
  - Collecting and processing call site information in SelectionDAGBuilder
- Implementation provides:
  - Correct debug data
  - Untouched code generation process
  - Backbone for further improvements

Collaboration

RT-RK
COMPUTER BASED SYSTEMS

CISCO Systems

Ananthakrishna Sowda asowda@cisco.com

Ivan Baev ibaev@cisco.com

“If Debugging is the process of removing software bugs, then programming must be the process of putting them in …”

EDSGER DIJKSTRA.

RT - RK
COMPUTER BASED SYSTEMS

Contact us

RT-RK Institute for Computer Based Systems
Narodnog fronta 23a
21000 Novi Sad
Serbia

www.rt-rk.com
info@rt-rk.com