Qualcomm

“Hey, do you want a RISC-V debugger?”

Enabling RISC-V support in LLDB

Ted Woodward

Senior Staff Engineer

Qualcomm Innovation Center, Inc

@qualcomm

Getting started

- RISC-V compiler discussed with management
- How do you run RISC-V programs?
  - QEMU
    - LLDB can talk to QEMU!
- “Hey, do you want a RISC-V debugger?”
  - “Yes! That would be great!”
- Time to build LLDB for RISC-V

Problems debugging RISC-V programs – ABI plugin needed

- Stopped at a bad address
- Can’t read the PC
- QEMU publishes the registers, so what’s the problem?
- LLDB can’t figure out which register is the PC
- Other architectures get that from their ABI plugin
- No RISC-V ABI plugin, so we need to write one

Terminal

File Edit View Search Terminal Help

```
}=hu-tedwood-lv=2.41=: /local/mnt/ted/8.8/riscv -> bin/lldb
(lldb) platform select qemu-user
  Platform: qemu-user
    Triple: x86_64-*-linux-gnu
OS Version: 5.4.0 (5.4.0-187-generic)
  Hostname: 127.0.0.1
WorkingDir: /local/mnt/ted/8.8/riscv
    Kernel: #207~18.04.1-Ubuntu SMP Thu Jun 13 15:01:12 UTC 2024
(lldb) settings set platform.plugin.qemu-user.architecture riscv32
(lldb) file ~/lldb_test/factrv32
Current executable set to '/usr2/tedwood/lldb_test/factrv32' (riscv32).
(lldb) b main
Breakpoint 1: where = factrv32`main + 28 at factorial.c:32:8, address = 0x000104
ea
(lldb) r
Process 1 launched: '/usr2/tedwood/lldb_test/factrv32' (riscv32)
Process 1 stopped
* thread #1, stop reason = signal SIGTRAP
    frame #0: 0xffffffffffffffff
(lldb) re r pc
error: Command requires a process, which is currently stopped.
(lldb)
```

Problems debugging RISC-V programs – bad disassembly

- Many instructions don't disassemble correctly
- Turn on "a" and "m" extensions by default

Terminal

File Edit View Search Terminal Help

```
}=hu-tedwood-lv=2.44=: /local/mnt/ted/8.8/riscv -> bin/lldb
(lldb) file ~/lldb_test/factrv32
Current executable set to '/usr2/tedwood/lldb_test/factrv32' (riscv32).
(lldb) dis -s main
factrv32`main:
factrv32[0x104ce] <+0>:
factrv32[0x104d0] <+2>:
factrv32[0x104d2] <+4>:
factrv32[0x104d4] <+6>:
factrv32[0x104d6] <+8>:
factrv32[0x104d8] <+10>: sw     a2, -28(s0)
factrv32[0x104dc] <+14>: sw     a2, -12(s0)
factrv32[0x104e0] <+18>: sw     a0, -16(s0)
factrv32[0x104e4] <+22>: sw     a1, -20(s0)
factrv32[0x104e8] <+26>:
factrv32[0x104ea] <+28>: sw     a0, -24(s0)
(lldb)
```

Problems debugging RISC-V programs – single step issue

- Source “step in” sometimes does a “step over”
- Jump instructions not marked IsBranch
- LLDB will run past them
- Worked with upstream to get this fixed

```
Terminal
File  Edit  View  Search  Terminal  Help
(lldb) file ~/lldb_test/factrv32
Current executable set to '/usr2/tedwood/lldb_test/factrv32' (riscv32).
(lldb) dis -b -s main -e 0x10502
factrv32`main:
factrv32[0x104ce] <+0>:  01 11        addi   sp, sp, -0x20
factrv32[0x104d0] <+2>:  06 ce        sw     ra, 0x1c(sp)
factrv32[0x104d2] <+4>:  22 cc        sw     s0, 0x18(sp)
factrv32[0x104d4] <+6>:  00 10        addi   s0, sp, 0x20
factrv32[0x104d6] <+8>:  01 46        li     a2, 0x0
factrv32[0x104d8] <+10>: 23 22 c4 fe  sw     a2, -0x1c(s0)
factrv32[0x104dc] <+14>: 23 2a c4 fe  sw     a2, -0xc(s0)
factrv32[0x104e0] <+18>: 23 28 a4 fe  sw     a0, -0x10(s0)
factrv32[0x104e4] <+22>: 23 26 b4 fe  sw     a1, -0x14(s0)
factrv32[0x104e8] <+26>: 29 45        li     a0, 0xa
factrv32[0x104ea] <+28>: 23 24 a4 fe  sw     a0, -0x18(s0)
factrv32[0x104ee] <+32>: 03 25 84 fe  lw     a0, -0x18(s0)
factrv32[0x104f2] <+36>: 23 20 a4 fe  sw     a0, -0x20(s0)
factrv32[0x104f6] <+40>: 59 37        jal    0x1047c
factrv32[0x104f8] <+42>: 83 25 04 fe  lw     a1, -0x20(s0)
factrv32[0x104fc] <+46>: 2a 86        mv     a2, a0
factrv32[0x104fe] <+48>: 17 d5 03 00  auipc  a0, 0x3d
(lldb)
```

Upstream the ABI plugin

- Someone upstreamed one, but no activity for a year
- I took some ideas from it, then upstreamed mine
- Many review comments addressed, and merged it a month later
- Now LLDB can debug RISC-V programs!
- We took the upstream code and released a product based on it

Thanks to:

- Ana Pazos (Qualcomm Innovation Center) and her RISC-V compiler team
  - Inspiring me to get LLDB working with RISC-V
  - Answering RISC-V questions as I learned about it
- Jason Molenda and David Spickett
  - Reviewing my ABI plugin, and helping make it better
- Pavel Labath
  - Writing the qemu-user platform plugin

Thank you

Follow us on:

For more information, visit us at qualcomm.com & qualcomm.com/blog