在x86环境上使用gdb调试qemu-user上运行的RVV程序

邱吉

2025/05/23

准备工作

- 待测程序使用交叉编译器riscv64-unknown-linux-gnu-gcc 加上-g选项编译
- 系统上安装好riscv64-unknown-linux-gnu-gdb
- riscv64-unknown-linux-gnu-gcc和riscv64-unknown-linux-gnu-gdb的版本最好是匹配的，可以选择从 https://github.com/riscv/riscv-gnu-toolchain 下载并构建整个套件

一个例子rvv_addvx8.c

- 第一步：编译源文件：
  - riscv64-unknown-linux-gnu-gcc -march=rv64gcv_zfh_zvfh -mabi=lp64d -static -Wall -Wextra -Wno-sign-conversion -O3 rvv_addvx8.c -g -o rvv_addvx8.elf
- 第二步：启动交叉gdb
  - qemu-riscv64 -g 1234 rvv_addvx8.elf -s –S

In order to use gdb, launch QEMU with the -s and -S options. The -s option will make QEMU listen for an incoming connection from gdb on TCP port 1234, and -S will make QEMU not start the guest until you tell it to from gdb. (If you want to specify which TCP port to use or to use something other than TCP for the gdbstub connection, use the -gdb dev option instead of -s. See Using unix sockets for an example.)

一个例子rvv_addvx8.c

- 第三步：在另一个窗口里启动riscv64-unknown-linux-gnu-gdb
  - riscv64-unknown-linux-gnu-gdb rvv_addvx8.elf
  - 在gdb环境中输入“target remote localhost:1234” 去通过socket连接qemu
  - 链接上以后，就可以按照常规的gdb使用方法来设置断点和查看变量/寄存器了
  - 比如将断点设置到c文件的vmseq语句上观察v0寄存器如何被设置

```
qjivy@plct-c7:~/work/rvvqemugdb$ riscv64-unknown-linux-gnu-gdb rvv_addvx8.elf
GNU gdb (GDB) 15.2
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=x86_64-pc-linux-gnu --target=riscv64-unknown-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from rvv_addvx8.elf...
(gdb) target remote localhost:1234
Remote debugging using localhost:1234
0x000000000001052c in _start ()
(gdb) b rvv_addvx8.c:44
Breakpoint 1 at 0x10492: rvv_addvx8.c:44. (2 locations)
```

gdb如何查看RVV相关的寄存器

- 查看vl和vtype寄存器

```
(gdb) p /x $vl
$18 = 0x5
(gdb) p /x $vtype
$19 = 0xc0
```

- 执行单步一条指令si，并查看v0寄存器的前后值变化

```
(gdb) p /x $v0
$24 = {q = {0x1010101}, l = {0x1010101, 0x0}, w = {0x1010101, 0x0, 0x0, 0x0}, s = {0x101, 0x101, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, b = {0x1, 0x1, 0x1, 0x1,
    0x0 <repeats 12 times>}}
(gdb) si
49      vuint8m1_t vec_value_3_0 = __riscv_vadd_vx_u8m1_m (mask_value_0_0, vec_value_1_0,       placeholder2, vl);
(gdb) p /x $v0
$25 = {q = {0x101010f}, l = {0x101010f, 0x0}, w = {0x101010f, 0x0, 0x0, 0x0}, s = {0x10f, 0x101, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0}, b = {0xf, 0x1, 0x1, 0x1,
    0x0 <repeats 12 times>}}
```

可以看到最后用到的mask实际上就是0xf
(了解vector masking用法请看上一个视频)