RISC-V Semihosting

Krste Asanovic, Palmer Dabbelt, Liviu Ionescu, Keith Packard, Megan Wachs

Version 0.2, 4/2020: This document is under development. Expect potential changes.

Table of Contents

Preamble

This document is in the Development state

Expect potential changes. This draft specification is likely to evolve before it is accepted as a standard. Implementations based on this draft may not conform to the future standard.

Copyright and license information

Contributors

This RISC-V specification has been contributed to directly or indirectly by:

- Krste Asanovic <krste@sifive.com>
- Palmer Dabbelt <palmer@dabbelt.com>
- Liviu Ionescu <ilg@livius.net>
- Keith Packard <keith.packard@sifive.com>
- Megan Wachs <megan@sifive.com>

Chapter 1. Introduction

Semihosting is a technique where an application running in a debug or simulation environment can access elements of the system hosting the debugger or simulator including console, file system, time and other functions. This allows for diagnostics, interaction and measurement of a target system without requiring significant infrastructure to exist in that target environment.

The RISC-V semihosting borrows from the design of other publicly available and open source semihosting mechanisms to minimize the development effort required. This specification only defines the binary interface between RISC-V software and the system hosting the debugger or simulator.

Chapter 2. RISC-V Semihosting Binary Interface

The RISC-V semihosting binary interface consist of a trap instruction sequence and mechanism to pass parameters which are defined by the following sub-sections.

2.1. Semihosting Trap Instruction Sequence

Semihosting operations are requested using a sequence of instructions including `ebreak`. Because the RISC-V base ISA does not provide more than one `ebreak` instruction, RISC-V semihosting uses a special sequence of instructions to distinguish a semihosting `ebreak` from a debugger inserted `ebreak`. The Listing 1 shows the instruction sequence used to invoke a Semihosting operation.

Listing 1. RISC-V Semihosting Trap Sequence

```
slli x0, x0, 0x1f   # 0x01f01013   Entry NOP
ebreak              # 0x00100073   Break to debugger
srai x0, x0, 7      # 0x40705013   NOP encoding the semihosting call number 7
```

These three instructions must be 32-bit-wide instructions, they may not be compressed 16-bit instructions. This same sequence is used on all RISC-V architectures. If paging is in use in the current mode, this sequence must not cross a page boundary as the semihosting system must be able to check for the semihosting sequence without needing data from potentially missing pages. The Listing 2 shows how this can be done by placing the sequence in a separate function and aligning that to prevent that from spanning a page boundary.

Listing 2. RISC-V Semihosting Trap Function

```
        .option norvc
        .text
        .balign 16
        .global sys_semihost
        .type sys_semihost @function
sys_semihost:
        slli zero, zero, 0x1f
        ebreak
        srai zero, zero, 0x7
        ret
```

2.2. Semihosting Parameters

The type of semihosting operation and it's parameter are specified using general purpose registers. The Table 1 shows the specific registers that are used, and the size of the fields in the data block, which depend on whether the caller is 32-bit or 64-bit.

	32-bit	64-bit
OPERATION NUMBER REGISTER	A0	A0
PARAMETER REGISTER	A1	A1
RETURN REGISTER	A0	A0
Data block field size	32 bits	64 bits

Table 1. RISC-V Registers and field size

2.3. Semihosting Execution Environment

Any configuration required to support semihosting on RISC-V systems must be defined by the execution environment. Of particular interest is the RISC-V external debug specification, which allows M-mode (and optionally S/U modes) `ebreak` instructions to be treated as semihosting traps.

Bibliography