RISC-V Semihosting

Version 0.95, 21st January 2025: This document is Frozen.

Table of Contents

Preamble

This document is in the Frozen state

Change is extremely unlikely. A high threshold will be used, and a change will only occur because of some truly critical issue being identified during the public review cycle. Any other desired or needed changes can be the subject of a follow-on new extension.

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>
- Anup Patel <apatel@ventanamicro.com>
- Ved Shanbhogue <ved@rivosinc.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 specification adopts the design of the ARM semihosting specification [1] to minimize the development effort. The services defined by the ARM semihosting specification [1] are portable across different architectures, and only the mechanism of invoking a semihosting service (aka semihosting binary interface) is archicture specific. The Figure 1 below shows an architecture independent high-level view of semihosting usage.

The RISC-V semihosting specification only defines the semihosting binary interface for RISC-V platforms and all other aspects of semihosting are defined by the ARM semihosting specification [1].

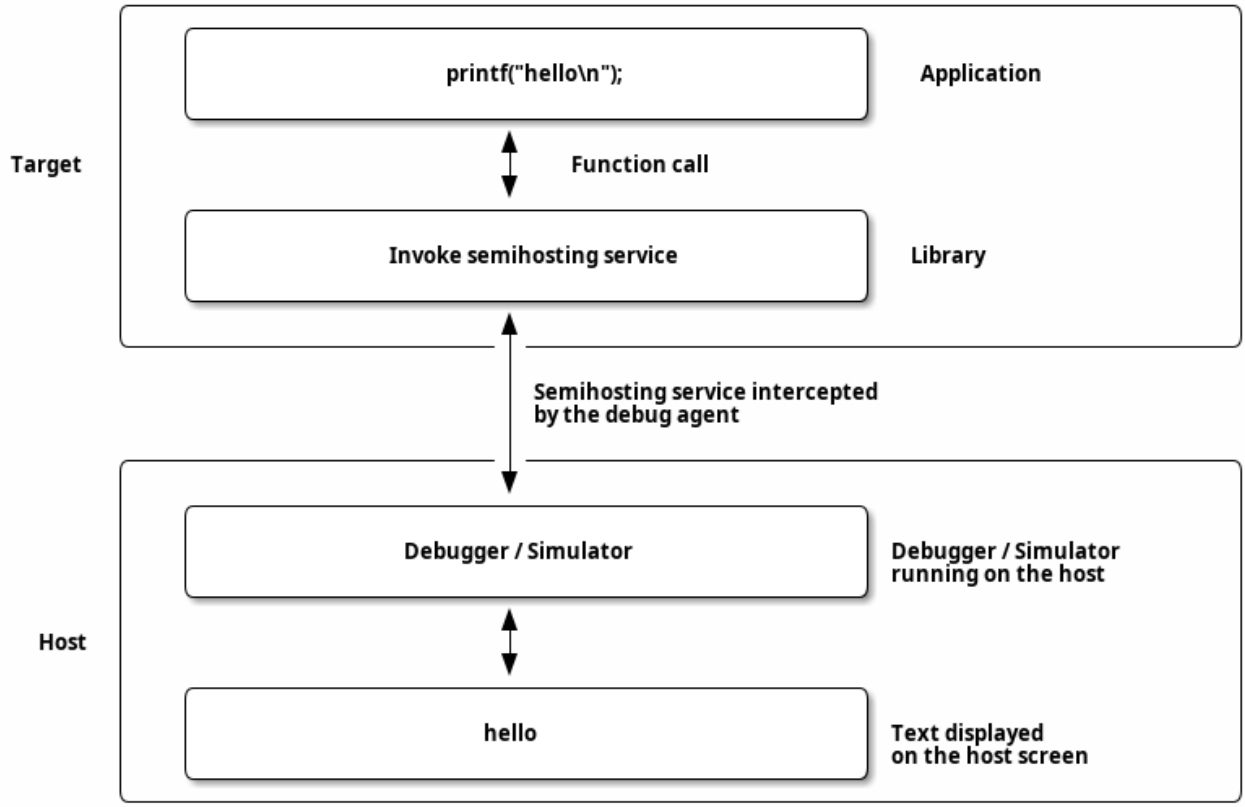

Figure 1. Generic Semihosting Usage Flow

Chapter 2. RISC-V Semihosting Binary Interface

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

2.1. Semihosting Breakpoint 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.

```
slli x0, x0, 0x1f   # 0x01f01013   Entry NOP
ebreak              # 0x00100073   Break to debugger
srai x0, x0, 7      # 0x40705013   Exit NOP
```

Listing 1. RISC-V Semihosting Breakpoint Sequence

These three instructions must be 32-bit wide instructions. This sequence is applicable to all RISC-V base ISAs. If address translation and protection is enabled for the semihosting caller then the semihosting instruction sequence and data passed via memory must be paged in else the behavior of the semihosting call is UNSPECIFIED.

The `SLLI`, `EBREAK`, and `SRAI` instructions are part of the ratified RV32E, RV32I, RV64E and RV64I (aka Base Integer Instruction Set) specifications [2] hence these instructions are present on almost all RISC-V platforms.

The `SLLI` and `SRAI` instruction based NOPs which serve as semihosting marker have been randomly selected from the Base Integer Instruction Set since these are designated for custom use and unlikely to appear in real life code.

2.2. Semihosting Parameters

The type of semihosting operation and its parameters are specified using general purpose registers. The OPERATION NUMBER is specified in the `a0` register, and the PARAMETER is specified in the `a1` register, whereas the RETURN VALUE is available in the `a0` register. All registers and data block fields are XLEN wide.

Bibliography

[1] “Semihosting for AArch32 and AArch64 2023Q3.” 2023, [Online]. Available: github.com/ARM-software/abi-aa/releases/download/2023Q3/semihosting.pdf.

[2] “The RISC-V Instruction Set Manual Volume I: Unprivileged Architecture.” 2024, [Online]. Available: github.com/riscv/riscv-isa-manual/releases.