



RISC-V S-mode Physical Memory Protection for Hypervisor

Editor - Dong Du and Sandro Pinto

Version 0.1, 11/2023: This document is in development. Assume everything can change. See
<http://riscv.org/spec-state> for details.

Table of Contents

- Preamble..... 1
- Copyright and license information..... 2
- Contributors..... 3
- 1. Introduction..... 4
- 2. Specification 5
 - 2.1. Hypervisor-Extended SPMP (hSPMP) 5
 - 2.2. Guest/Virtual SPMP (vSPMP) 6

Preamble



This document is in the [Development state](#)

Assume everything can change. This draft specification will change before being accepted as standard, so implementations made to this draft specification will likely not conform to the future standard.

Copyright and license information

This specification is licensed under the Creative Commons Attribution 4.0 International License (CC-BY 4.0). The full license text is available at creativecommons.org/licenses/by/4.0/.

Copyright 2023 by RISC-V International.

Contributors

The proposed specifications (non-ratified, under discussion) has been contributed to directly or indirectly by:

- Dong Du, Editor <dd_nirvana@sjtu.edu.cn>
- Sandro Pinto, Editor <sandro.pinto@dei.uminho.pt>
- Bicheng Yang
- Jose Martins

Chapter 1. Introduction

RISC-V S-mode Physical Memory Protection (SPMP) is an extension to provide isolation when MMU is unavailable or disabled. This specification introduces the extension to support SPMP and hypervisor extension.

TODO: motivation for adding SPMP hypervisor support with example use cases from target application domains

Chapter 2. Specification

2.1. Hypervisor-Extended SPMP (hSPMP)

The Shmpmp extension enhances the behavior of the original SPMP so that when the Hypervisor extension is present the behavior of the SPMP is enhanced to an hypervisor-extended SPMP (hSPMP). When V=0, the hSPMP acts as the baseline SPMP. When V=1 and hgatp.mode=Bare the hSPMP checks all guest memory accesses, i.e., all accesses from VS or VU modes. The encoding permissions remain the same as for the baseline SPMP for when spmpcfg.S is clear, but enforcing the rule over VS/VU modes instead of U mode.

Note that the Hypervisor Virtual Machine Load and Store instructions (HLV/HLVX/HSV) executed from M, HS or HU (when hstatus.HU is set) modes are also subject to the same hSPMP checks as when V=1.

Table 1. hSPMP Encoding Table with spmpcfg.S=0 when V=1

spmpcfg	VS/VU-mode
R - -	Enforce
R - X	Enforce
- - X	Enforce
- - -	Enforce
R W -	Enforce
R W X	Enforce
- W X	RSVD
- W -	RSVD

When a guest access is denied by the hSPMP an exception is generated. Depending on the type of access (i.e., load, store/AMO, or instruction fetch) a different kind of fault is generated. The hSPMP reuses exception codes from the Hypervisor extension for guest page faults since SPMP protection and paged virtual memory are mutually exclusive and cannot be active at the same time as the SPMP is only enabled when satp.mode == Bare. Essentially this extension renames these exception codes as detailed in Table ?? seen next.

Interrupt	Exception Code	Description
0	20	Instruction guest-access/page fault
0	21	Load guest-access/page fault
0	23	Store/AMO guest-access/page fault

The hSPMP adds an extra hypervisor WARL CSR pair to the SPMP, hspmpswitch0-1. When V=1 these CSRs take the role of the original spmpswitch0-1. This means that, when V=1, hSPMP entry is active

only when both corresponding bits in `hspmpswitch` and `spmp[i]cfg,A` are set. When `V=0`, these registers have no effect.

2.2. Guest/Virtual SPMP (vSPMP)

This extension introduces a new PMP, the vSPMP, behaving exactly as the original SPMP, but under the control of VS-mode and checking accesses from VS and VU modes, therefore only active when `V=1`. Essentially, this extension introduces a two-stage SPMP before the PMP/ePMP (Figure ??). From a logical point of view, when both `vsatp.mode` and `hgap.mode` are set to Bare, guest accesses (i.e., originating from VS/VU) are first checked by the the vSPMP, and if allowed, then checked by the hSPMP. Meaning that if the access is denied both by the vSPMP and the hSPMP, an SPMP fault is generated (i.e., fault with exception code 12, 13 or 15). The implementation can perform vSPMP checks in parallel with hSPMP cheks.

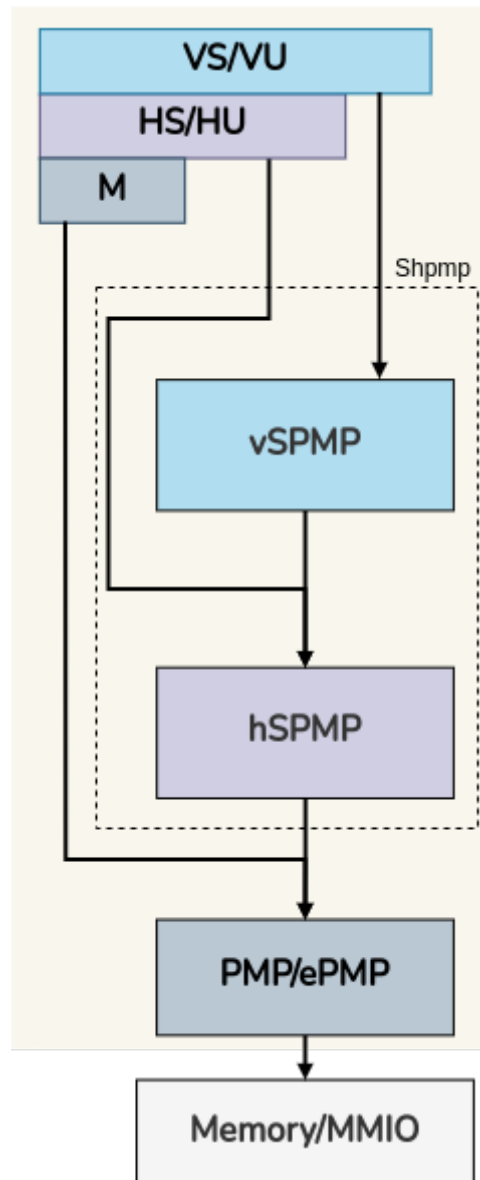


Figure 1. Dual-Stage SPMP

For the purposes of full alignment with the baseline Hypervisor specification, and from a logical point of view, addresses used before vSPMP checks are considered to be guest virtual addresses (GVA), and those checked by the hSPMP are guest physical addresses (GPA).

If `vsatp.mode` is set to Bare but `hgatp.mode` is not, the vSPMP check is logically followed by the usual G-stage translation. Implementations can implement them in parallel.

If `vsatp.mode` is not set to bare, but `hgatp.mode` is, the hSPMP happens logically after the VS-stage transaction. Implementations must use the guest physical address (GPA), resulting from the VS-stage translation, to perform the hSPMP check, meaning a parallel check is not feasible unless speculation is employed.

VS mode controls the vSPMP through a number of new virtual supervisors CSRs (VS CSRs), replicas of the normal SPMP supervisor CSRs. These include `vspmpcfg0-3`, `vspmpaddr0-64` CSRs, and `vspmpswitch0-1`. As with the virtual supervisors CSRs in the baseline Hypervisor specification, when `V=1`, accesses to original SPMP CSRs, are diverted to these VS CSRs.

The number of vSPMP entries can vary by implementation, and up to 64 SPMP entries are supported in the standard.