RISC-V S-mode Physical Memory Protection for Hypervisor

Editor - Bicheng Yang and José Martins

Version 0.2, 10/2025: This document is in development. Assume everything can change. See http://riscv.org/spec-state for details.

Table of Contents

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

Contributors

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

- Bicheng Yang, Editor <bichengyang@sjtu.edu.cn>
- José Martins, Editor <jose@osyx.tech>
- Dong Du
- Sandro Pinto
- Thomas Roecker
- Manuel Rodriguez
- Luís Cunha
- Kajetan Nürnberger

Chapter 1. Introduction

The RISC-V S-mode Physical Memory Protection (SPMP) for Hypervisor integrates the SPMP and Hypervisor extensions by providing a two-stage SPMP. The first stage, vSPMP, in control of VS-mode allowing a guest supervisor to specify memory protection policies within its guest physical address space, while the second stage, hSPMP, extends the baseline S-mode SPMP with support for the hypervisor to enforce physical memory access policies on guest memory accesses.

Chapter 2. "Sshspmp" Extension for Hypervisor-Extended SPMP

2.1. Extension Dependencies

1. The `Sshspmp` is dependent on the baseline SPMP extension, `Sspmp`.
2. The `Sshspmp` extension is dependent on `Ss1p13` and the Hypervisor Extension, with the following requirement relaxations:
   a. The RISC-V Privileged Architecture specification states that for the Hypervisor Extension to be implemented “standard page-based address translation must be supported, either Sv32 for RV32, or a minimum of Sv39 for RV64.”. When `Sshspmp` is implemented, this requirement is relaxed:
      - The only mandatory translation mode in `satp` is `Bare`.
   b. When `Sshspmp` is implemented, second-stage translation support is not required to implement the Hypervisor Extension:
      - The only mandatory mode in `hgatp` is `Bare`.

2.2. Hypervisor-Extended SPMP (hSPMP)

The `Shspmp` extension augments the baseline SPMP functionality to support virtualization into the hypervisor-extended SPMP (hSPMP). When V = 0, hSPMP behaves identically to the baseline SPMP. When V = 1 and `hgatp.MODE = Bare`, hSPMP enforces access checks on all memory accesses from VS and VU-modes, effectively applying SPMP protections to guest execution contexts. The permission encodings remain consistent with those of the baseline SPMP when `spmpcfg.U = 1`, but are applied to VS/VU rather than U-mode accesses. Additionally, the Hypervisor Virtual Machine Load and Store instructions (HLV, HLVX, HSV), when executed from M-mode, HS-mode, or U-mode (when `hstatus.HU = 1`), are also subject to hSPMP checks under the V = 1 condition.

When a VS/VU access is denied by hSPMP, an exception is raised. The type of exception depends on the nature of the access attempt, i.e., whether it is a load, store/AMO, or instruction fetch. The hSPMP reuses the guest page fault exception codes (20, 21, and 23, for instruction, load, and store/AMO accesses, respectively) defined by the Hypervisor extension.

Since, when V = 1, hSPMP is only active when `hgatp.MODE = Bare`, hSPMP protection and G-stage translation are mutually exclusive. Consequently, when hSPMP is active, paged virtual memory translations are disabled, and guest page fault exception codes can be repurposed to indicate guest hSPMP access violations.

Chapter 3. "Sshspmpsw" Extension for Optimizing Context Switching of hSPMP Entries

3.1. Extension Dependencies

1. The `Sshspmpsw` is dependent on `Sshspmp` and `Sspmpsw`.

3.2. Optimization for Context Switching of hSPMP Entries

The `Sshspmpsw` introduces additional CSRs:

- In RV64: one 64-bit WARL CSR called `hspmpswitch` is added.
- In RV32: in addition to `hspmpswitch`, a 32-bit WARL CSR called `hspmpswitchh` is added, which is an alias of the upper half of `hspmpswitch`.

When V = 1, these registers replace the function of `spmpswitch`. Specifically, when V = 1, an hSPMP entry is considered active only if both its `spmpcfg[i].A` is set and the corresponding bit in `hspmpswitch` is set. When V = 0, the `hspmpswitch` register has no effect, and the baseline SPMP behavior applies as defined by `spmpswitch` alone.

When an entry `i` is locked, as indicated by `spmpcfg[i].L == 1`, bit `i` in `hspmpswitch` becomes read-only.

The rationale for providing a separate `hspmpswitch` register, rather than reusing `spmpswitch`, is that HS/U and VS/VU contexts conceptually occupy distinct address spaces, not merely different permission domains. This mirrors the baseline Hypervisor extension for page-based translation, where HS/U accesses use the page tables pointed to by `satp`, while VS/VU accesses are translated through a separate set of page tables defined by `hgatp`. Similarly, `spmpswitch` governs SPMP entries for the hypervisor's own address space, while `hspmpswitch` controls the guest address space enforced by hSPMP. This separation allows the hypervisor to manage its own and guest memory protection configurations independently.

Chapter 4. "Ssvspmp" Extension for Virtual SPMP

4.1. Extension Dependencies

1. The `Ssvspmp` is dependent on `Ss1p13` and the Hypervisor Extension, with the following requirement relaxations:
   - The only mandatory mode in `vsatp` is `Bare`.
2. The `Ssvspmp` is dependent on the Hypervisor-Extended SPMP extension, `Sshspmp`.

4.2. Virtual SPMP (vSPMP)

The `Ssvspmp` extension introduces a new memory protection layer called Virtual SPMP (vSPMP). The vSPMP behaves analogously to the baseline SPMP but is under the control of VS-mode and applies to memory accesses originating from VS and VU modes. The vSPMP is active only when V = 1. Conceptually, this extension establishes a two-stage SPMP hierarchy, preceding any checks by PMP or ePMP (see Figure 1). From a logical perspective, when both `vsatp.MODE` and `hgatp.MODE` are set to Bare, memory accesses initiated by guest contexts (VS/VU) are subject to the following enforcement sequence:

1. vSPMP check: If the access is permitted by vSPMP, then
2. hSPMP check: The access must also be permitted by hSPMP.

If the vSPMP check or both the vSPMP and hSPMP checks deny a VS/VU access, a standard SPMP page-fault directed at VS-mode is triggered using page fault exception codes 12 (instruction), 13 (load), or 15 (store/AMO). If only the hSPMP denies a VS/VU access, a fault directed at HS-mode is raised using guest page fault exception codes as defined by `Sshspmp`. Implementations may perform vSPMP and hSPMP checks in parallel, provided the overall enforcement behavior matches the sequential logic described above.

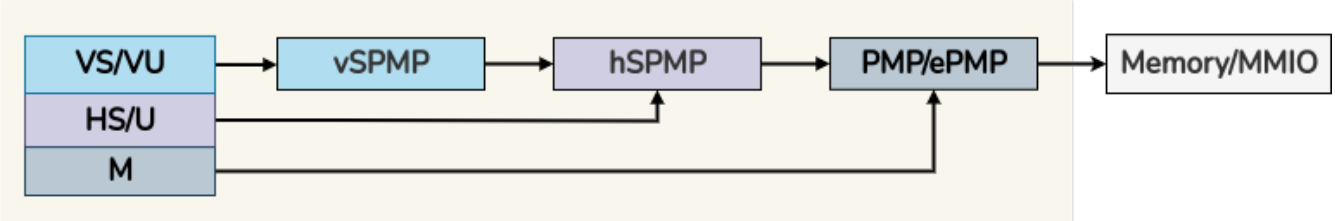

Figure 1. Dual-Stage SPMP

When `vsatp.MODE` is set to `Bare` but `hgatp.MODE` is not, the vSPMP check is logically followed by G-stage address translation. Implementations may perform the vSPMP check and G-stage translation in parallel.

When `vsatp.MODE` is not set to `Bare`, but `hgatp.MODE` is, the hSPMP check logically occurs after the VS-stage address translation. In this scenario, the Guest Physical Address (GPA) produced by the VS-stage translation must be used as the input to the hSPMP check. Consequently, parallel evaluation of VS-stage translation and hSPMP enforcement is not possible unless speculative techniques are

employed.

The vSPMP is configured through a set of virtual supervisor indirect-access registers, which are architectural replicas of the standard SPMP registers. These include vspmpaddr0–vspmpaddr63, vspmpcfg9–vspmpcfg63. See Section 4.3 for details on how these registers are accessed.

Furthermore, if Sspmpsw is implemented, the vSPMP must implement additional virtual supervisor CSRs which control the active vSPMP entries, analogously to sspmpswitch for the SPMP:

- In RV64: one 64-bit WARL CSR called vspmpswitch is added.
- In RV32: in addition to vspmpswitch, a 32-bit WARL CSR called vspmpswitchh is added, which is an alias of the upper half of vspmpswitch.

4.3. Access Method for vSPMP registers

Consistent with the behavior defined in the Hypervisor and Indirect CSR Access extensions, vSPMP registers can be accessed from M and HS-modes via the vsiselect/vsiregX and accesses to the original SPMP registers from VS-mode via siselect/siregX CSRs are transparently redirected to the corresponding virtual supervisor vSPMP registers.

Each vsiselect represents an access to the corresponding SPMP virtual supervisor CSRs. The vsireg accesses vspmpaddr, and the sireg2 accesses vspmpcfg.vsireg3, vsireg4, vsireg5, and vsireg6 are read-only 0.

vsiselect **number**	**indirect CSR access of** vsireg
vsiselect#0	vsireg → vspmpaddr[0], vsireg2 → vspmpcfg[0]
vsiselect#1	vsireg → vspmpaddr[1], vsireg2 → vspmpcfg[1]
…	…
vsiselect#63	vsireg → vspmpaddr[63], vsireg2 → vspmpcfg[63]

Both M and HS-mode accesses via vsireg2 CSRs, and VS-mode accesses via sireg2 CSRs to the vSPMP vspmpcfg resgisters, can set vspmpcfg[i].L to lock a vSPMP entry. When vspmpcfg[i].L is set, writes to that entry are ignored. Only M and HS-mode writes via vsireg2 can reset vspmpcfg[i].L.

Chapter 5. "Sshspmpdeleg" Extension for Sharing Hardware Resources between hSPMP and vSPMP

5.1. Extension Dependencies

1. The `Sshspmpdeleg` is dependent on `Ssvpmp` and `Smpmpdeleg`.

5.2. Resource Sharing between hSPMP and vSPMP

With this extension the hypervisor can delegate entries from hSPMP to vSPMP. For this, it introduces a new WARL `hspmpdeleg` CSR with the same format and functionality of `mpmpdeleg`, but for delegating hSPMP entries to vSPMP. Any PMP entry with an index greater than or equal to `mpmpdeleg.pmpnum` + `hspmpdeleg.pmpnum` is delegated to the vSPMP. For example, if the number of implemented PMP entries is 48, `mpmpdeleg.pmpnum` is set o 8, and `hspmpdeleg.pmpnum` is set to 16, PMP entries 24 to 47 are delegated to the vSPMP.

If `hspmpdeleg.pmpnum` is written with a value such that `mpmpdeleg.pmpnum` + `hspmpdeleg.pmpnum` is equal to or greater than the number of implemented PMP entries, the field will read back as the number of remaining hSPMP entries. For example, if the number of implemented PMP entries is 32, `mpmpdeleg.pmpnum` is set to 16, and `hspmpdeleg.pmpnum` is written a value of 32, the field will read back as 16.

`hspmpdeleg.pmpnum` must be set to a value larger than the index of any locked hSPMP entry. For example, if `hSPMP[7]` is locked, `hspmpdeleg.pmpnum` must be no less than 8.

`hspmpdeleg.pmpnum` can be hardwired to allow an implementation to fix the hSPMP/vSPMP split only if `mpmpdeleg.pmpnum` is also hardwired.

This extension allows for the number of implemented PMP entries to be greater than 64 and up to a maximum of 192. If the number of implemented entries is greater than 64, unless hardwired, `mpmpdeleg.pmpnum` reset value is 64, and `hspmpdeleg.pmpnum` is set to the value of the remaining entries available to the hSPMP. For example, if the total number of entries is 96, `hspmpdeleg.pmpnum` reset value is 32.

The limit of 192 for implemented PMP entries stems from the fact that this is the maximum number of addressable entries: 64 addressable for the vSPMP, 64 for the hSPMP, plus 64 for PMP. If more PMP entries were implemented, software would have no way to address and access such entries.

This extension is mandatory.

Addressing:

vSPMP entries will be supported contiguously and begin with the lowest vSPMP indirect CSR number. For instance, given an implementation with a total of 48 PMP entries, if `mpmpdeleg.pmpnum` is

set to 16, and hspmpdeleg.pmpnum to 16, PMP entry[32] to PMP entry[47] would map to vSPMP[0] to vSPMP[15]. A read of an out-of-index vSPMP entry (in the previous example, starting with vSPMP[16]) will return 0, and a write to such a vSPMP entry will be ignored.

If mpmpdeleg.pmpnum + hspmpdeleg.pmpnum is configured such that more than 64 entries are delegated to vSPMP, only the lowest 64 delegated entries are addressable through the vSPMP CSRs. Any additional delegated entries are architecturally inaccessible and have no effect on memory protection. For example, if the number of implemented PMP entries is 128, mpmpdeleg.pmpnum is set to 16, and hspmpdeleg.pmpnum is set to 16, PMP entry[32] to PMP entry[95] would map to vSPMP[0] to vSPMP[63], and PMP entry[96] to PMP entry[127] fall outside the vSPMP addressable range and therefore and have no effect on memory protection.