RISC-V S-mode Physical Memory Protection (SPMP)

Editor - Dong Du, RISC-V SPMP Task Group

Version 0.8.0, 10/2022: 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 SPMP specifications (non-ratified, under discussion) has been contributed to directly or indirectly by:

- Dong Du, Editor <dd_nirvana@sjtu.edu.cn>
- Bicheng Yang
- Nick Kossifidis
- Andy Dellow
- Manuel Offenberg
- Allen Baum
- Bill Huffman
- Xu Lu
- Wenhao Li
- Yubin Xia
- Joe Xie
- Paul Ku
- Jonathan Behrens
- Robin Zheng
- Zeyu Mi

Chapter 1. Introduction

This document describes RISC-V S-mode Physical Memory Protection (SPMP) proposal to provide isolation when MMU is unavailable. RISC-V based processors recently stimulated great interest in the emerging internet of things (IoT). However, as page-based virtual memory (MMU) is usually unavailable on IoT devices, it is hard to isolate the S-mode OSes (e.g., RTOS) and user-mode applications. To support secure processing and isolate faults of U-mode software, the SPMP is desirable to enable S-mode OS to limit the physical addresses accessible by U-mode software on a hart.

Chapter 2. S-mode Physical Memory Protection (SPMP)

An optional RISC-V S-mode Physical Memory Protection (SPMP) provides per-hart supervisor-mode control registers to allow physical memory access privileges (read, write, execute) to be specified for each physical memory region. The SPMP is checked before the PMA and PMP checks, the same as paged virtual memory.

Like PMP, the granularity of SPMP access control settings is platform-specific and, within a platform, may vary by physical memory region. However, the standard SPMP encoding should support regions as small as four bytes.

SPMP checks will be applied to all accesses for U mode and S mode, depending on the values in the configuration registers. M-mode accesses are not affected and always pass SPMP permission checks. SPMP registers can always be modified by M-mode and S-mode software. SPMP registers can grant permissions to U-mode, which has none by default and revoke permissions from S-mode, which has all permissions allowed through PMP/ePMP by default.

2.1. Requirements

1) S mode should be implemented

2.2. S-mode Physical Memory Protection CSRs

Like PMP, SPMP entries are described by an 8-bit configuration register and one XLEN-bit address register. Some SPMP settings additionally use the address register associated with the preceding SPMP entry. The number of SPMP entries can vary by implementation, and up to 64 SPMP entries are supported in the standard.

The terms, entry and rule, are similar to ePMP.

The SPMP configuration registers are packed into CSRs the same way as PMP. For RV32, 16 CSRs, spmpcfg0-spmpcfg15, hold the configurations spmp0cfg-spmp63cfg for the 64 SPMP entries. For RV64, even numbered CSRs (i.e., spmpcfg0, spmpcfg2, ..., spmpcfg14) hold the configurations for the 64 SPMP entries; odd numbered CSRs (e.g., spmpcfg1) are illegal. Figures 1 and 2 demonstrate the first 16 entries of SPMP. The layout of the rest entries is similar.

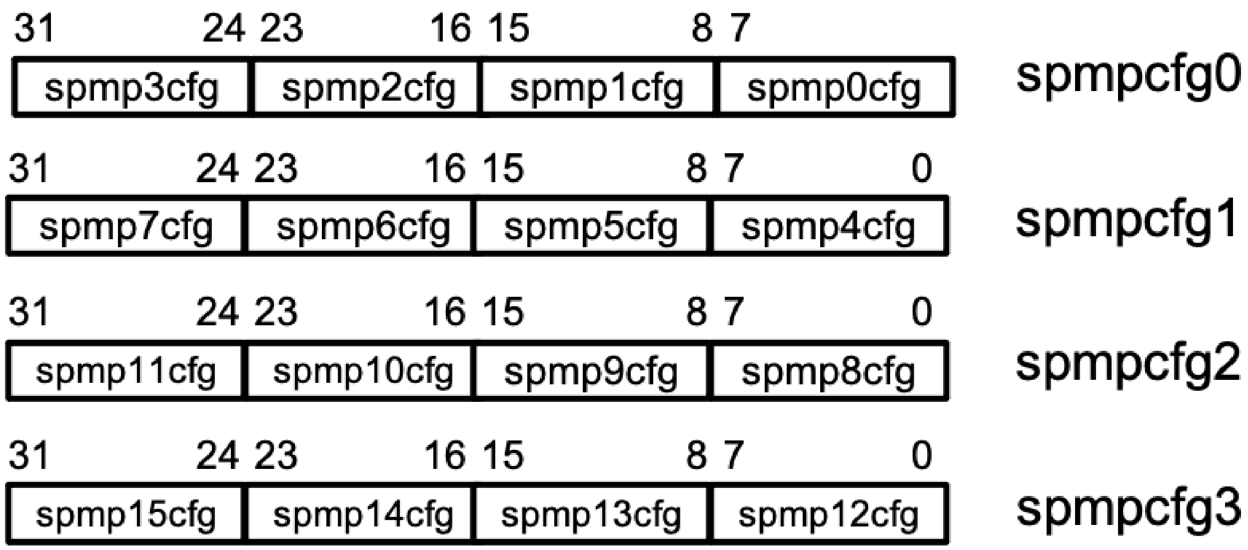

Figure 1. RV32 SPMP configuration CSR layout

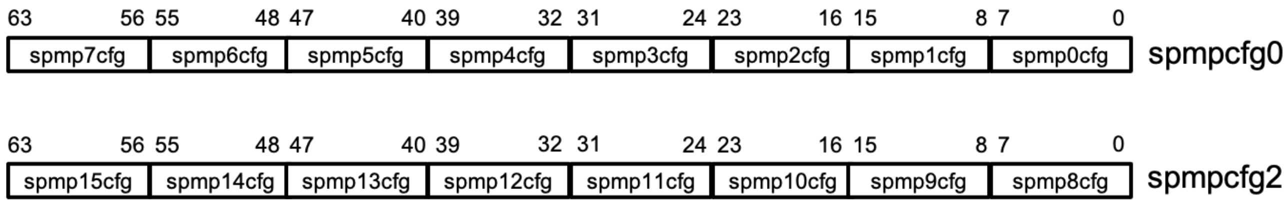

Figure 2. RV64 SPMP configuration CSR layout

The SPMP address registers are CSRs named spmpaddr0-spmpaddr63. Each SPMP address register encodes bits 33-2 of 34-bit physical address for RV32, as shown in Figure 4. For RV64, each SPMP address encodes bits 55–2 of a 56-bit physical address, as shown in Figure 5. Fewer address bits may be implemented for specific reasons, e.g., systems with smaller physical address space. Implemented address bits must be contiguous and go from lower to higher bits.

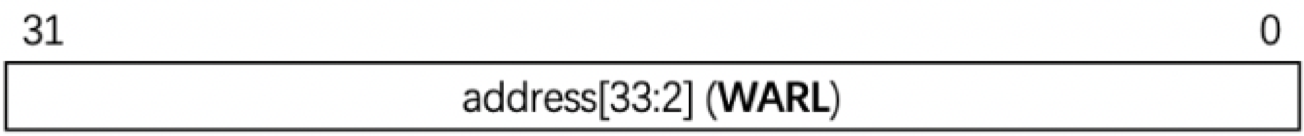

Figure 3. SPMP address register format, RV32

Figure 4. SPMP address register format, RV64

The layout of SPMP configuration registers is the same as PMP configuration registers, as shown in Figure 6. The register is WARL.

1. The S bit marks a rule as **S-mode-only** when set and **U-mode-only** when unset. The encoding of `spmpcfg.RW=01`, and the encoding `spmpcfg.SRWX=1111`, now encode a Shared-Region. The rules and encodings for permission are explained in section 2.4, which resembles the encoding of ePMP (except SPMP does not use locked rules).
2. Bit 5 and 6 are reserved for future use.

3. The A bit will be described in the following sections (2.3).
4. The R/W/X bits control read, write, and instruction execution permissions.

7	6 5	4 3	2	1	0
S	**Reserved**	A	X	W	R
1	2	2	1	1	1

Figure 5. SPMP configuration register format

The number of SPMP entries: The proposal allows 64 SPMP entries, providing 64 isolated regions concurrently. The software in S-mode (usually an OS) can virtualize more isolated regions and schedule them by switching the values in SPMP entries to provide more isolation regions.

The reset state: On system reset, the A field of spmp[i]cfg should be zero.

SPMP CSRs should be allocated contiguously starting with the lowest CSR number.

2.3. Address Matching

The A field in an SPMP entry's configuration register encodes the address-matching mode of the associated SPMP address register. It is the same as PMP/ePMP.

2.4. Encoding of Permissions

SPMP has three kinds of rules: **U-mode-only**, **S-mode-only**, and **Shared-Region** rules. The S bit marks a rule as **S-mode-only** when set and **U-mode-only** when unset. The encoding `spmpcfg.RW=01` encodes a Shared-Region and `spmpcfg.SRWX=1000` is reserved for future standard use.

1. An *S-mode-only* rule is **enforced** on Supervisor mode and **denied** on User mode.
2. A *U-mode-only* rule is **enforced** on User modes and **denied/enforced** on Supervisor mode depending on the value of `sstatus.SUM` bit:
   - If `sstatus.SUM` is set, a U-mode-only rule is enforced without code execution permission on Supervisor mode to ensure SMEP.
   - If `sstatus.SUM` is unset, a U-mode-only rule is denied on Supervisor mode.
3. A *Shared-Region* rule is enforced on both Supervisor and User modes, with restrictions depending on the `spmpcfg.S` and `spmpcfg.X` bits:
   - If `spmpcfg.S` is not set, the region can be used for sharing data between S-mode and U-mode, yet not executable. S-mode has RW permission to that region, and U-mode has read-only permission if `spmpcfg.X` is not set or RW permission if `spmpcfg.X` is set.
   - If `spmpcfg.S` is set, the region can be used for sharing code between S-mode and U-mode, yet not writeable. S-mode and U-mode have execute permission to the region, and S-mode may also have read permission if `spmpcfg.X` is set.
   - The encoding `spmpcfg.SRWX=1111` can be used for sharing data between S-mode and U-mode, where both modes only have read-only permission to the region.

The encoding and results are shown in the table:

	S=0			S=1		
spmpcfg	Smode	Smode	Umode	Smode	Smode	Umode
RWX	Sum=0	Sum=1	Sum=x	Sum=0	Sum=1	Sum=x
R - -	Deny	EnforceNoX	Enforce	Enforce	Enforce	Deny
R - X	Deny	EnforceNoX	Enforce	Enforce	Enforce	Deny
- - X	Deny	EnforceNoX	Enforce	Enforce	Enforce	Deny
- - -	Deny	EnforceNoX	Enforce	RSVD		
R W -	Deny	EnforceNoX	Enforce	Enforce	Enforce	Deny
R W X	Deny	EnforceNoX	Enforce	shr RO		
- W X	shr RW			shr RX		shr X
- W -	shr RW		shr RO	shr X		

Figure 6. SPMP Encoding Table

Deny: Access not allowed.

Enforce: The R/W/X permissions are enforced on accesses.

EnforceNoX: The R/W permissions are enforced on accesses, while the X bit is forced to be zero.

SHR: It is shared between S/U modes with X, RX, RW, or ReadOnly privileges.

RSVD: It is reserved for future use.

SUM bit: We re-use the sstatus.SUM (allow Supervisor User Memory access) bit of modifying the privilege with which S-mode loads and stores access to physical memory. The semantics of SUM in SPMP is consistent with those in Sv.

2.5. Priority and Matching Logic

M-mode accesses are always considered to pass SPMP checks. If PMP/ePMP is implemented, accesses succeed only if both PMP/ePMP and SPMP permission checks pass.

Like PMP entries, SPMP entries are also statically prioritized. The lowest-numbered SPMP entry that matches any byte of access (indicated by an address and the accessed length) determines whether that access is allowed or fails. The SPMP entry must match all bytes of access, or the access fails, irrespective of the S, R, W, and X bits.

1. If the privilege mode of the access is M, the access is allowed;
2. If the privilege mode of the access is S and no SPMP entry matches, the access is allowed;
3. If the privilege mode of the access is U and no SPMP entry matches, but at least one SPMP entry is implemented, the access fails;
4. Otherwise, the access is checked according to the permission bits in the matching SPMP entry. It is allowed if it satisfies the permission checking with the S, R, W, or X bit corresponding to the access type.

2.6. SPMP and Paging

The table below shows which mechanism to use. (Assume both paged virtual memory and SPMP are implemented.)

satp	Isolation mechanism
satp.mode == Bare	SPMP only
satp.mode != Bare	Paged Virtual Memory only

We do not allow both SPMP and paged virtual memory permissions to be actived at the same time now because: (1) It will introduce one more layer to check permission for each memory access. This issue will be more serious for a guest OS that may have host SPMP and guest SPMP. (2) Paged virtual memory can provide sufficient protection.

That means SPMP is enabled when `satp.mode==Bare` and SPMP is implemented.

If page-based virtual memory is not implemented, or when it is disabled, memory accesses check the SPMP settings synchronously, so no fence is needed.

2.7. Exceptions

Failed accesses generate an exception. SPMP follows the strategy that uses different exception codes for different cases, i.e., load, store/AMO, instruction faults for memory load, memory store/AMO and instruction fetch, respectively.

The SPMP reuses exception codes of page fault for SPMP fault. The SPMP reuses exception codes of page fault for SPMP fault. Because page fault is typically delegated to S-mode, so does SPMP fault, we can benefit from reusing page fault. S-mode software(i.e., OS) can distinguish page fault from SPMP fault by checking satp.mode (as mentioned in 2.6, SPMP and paged virtual memory will not be activated simultaneously). **SPMP proposes to rename page fault to SPMP/page fault for clarity**.

Note that a single instruction may generate multiple accesses, which may not be mutually atomic.

Table of renamed exception codes:

Interrupt	Exception Code	Description
0	12	Instruction SPMP/page fault
0	13	Load SPMP/page fault
0	15	Store/AMO SPMP/page fault

You can refer to Table 3.6 in riscv-privileged spec.

Delegation: Unlike PMP, which uses access faults for violations, SPMP uses SPMP/page faults for violations. The benefit of using SPMP/page faults is that we can delegate the violations caused by SPMP to S-mode, while the access violations caused by PMP can still be handled by machine mode.

2.8. Context Switching Optimization

With SPMP, each context switch requires the OS to store 64 address registers and 8 configuration registers (RV64), which is costly and unnecessary. So the SPMP proposes an optimization to minimize the overhead caused by context switching.

We add two CSRs called ***spmpswitch0*** and ***spmpswitch1***, which are XLEN-bit read/write registers, as shown in Figure 7. For RV64, only ***spmpswitch0*** is used. Each bit of this register holds the on/off status of the corresponding SPMP entry. During the context switch, the OS can store and restore spmpswitch as part of the context. An SPMP entry is activated only when both corresponding bits in spmpswitch and A field of spmpicfg are set. (i.e., spmpswitch[i] & spmp[i]cfg.A!=0)

Figure 7. SPMP domain switch register format (RV64)

Chapter 3. Summary of Hardware Changes

Item	Changes
CSRs for SPMP address	64 new CSRs
CSRs for SPMP configuration	16 new CSRs for RV32 and 8 for RV64
CSR for Domain switch	2 new CSRs for RV32 and 1 for RV64
Renamed exception code	*Instruction page fault* renamed to *Instruction SPMP/page fault* *Load page fault* renamed to *Load SPMP/page fault* *Store/AMO page fault* renamed to *Store/AMO SPMP/page fault*

Chapter 4. Interaction with hypervisor extension

There are some further changes to support both SPMP and hypervisor extension.

4.1. vSPMP extension

This extension describes how SPMP is used in a guest VM.

1. A set of vSPMP CSRs for the VS-mode are required, including 64 vSPMP address registers and 16 configuration registers. When V=1, vSPMP CSR substitutes for the usual SPMP CSR, so instructions that normally read or modify SPMP CSR access vSPMP CSR instead. This is consistent with the paging in VS-mode (i.e., vsatp).
2. For HLV, HLVX, and HSV instructions, the hardware should check vSPMP before G-stage address translation (or hgPMP protection when hgatp.BARE is set to zero).
3. The vSPMP checking is performed in the guest physical addresses before G-stage address translation (or hgPMP protection when hgatp.BARE is set to zero).

4.2. hgPMP extension

This extension describes how SPMP protects a hypervisor from guests (only enabled when hgatp.BARE is set to zero).

1. When hgPMP is enabled, all guest memory accesses will be checked by hgPMP; while hypervisor (in HS mode) and HU mode applications will not be affected.
2. A set of hgPMP CSRs for the HS-mode are required, including 64 hgPMPaddr address registers and 16 hgPMPcfg configuration registers. When V=1, and hgatp.MODE=Bare, hgPMP provides isolation between the hypervisor and guest VMs.
3. XLEN-bit read/write hgpmpswitch0 and hgpmpswitch1 CSRs are also provided in hgPMP, which are identical to spmpswitch0 and spmpswitch1 shown in Figure 7. Only hgpmpswitch0 is used for RV64. During the context switch, the hypervisor can simply store and restore hgpmpswitch (we use hgpmpswitch to represent either hgpmpswitch0 or hgpmpswitch1) as part of the context. An hgPMP entry is activated only when both corresponding bits in hgpmpswitch and A field of hgpmpicfg are set. (i.e., hgpmpswitch[i] & hgpmpicfg.A)
4. The hgPMP checking is performed after the guest address translation (or vSPMP checking), before PMP checking.

As hgPMP does not apply to the hypervisor, the encodings of configuration registers are simplified in the following table.

The encodings of hgpmpcfg are shown in the table:

Bits on *hgpmpcfg* register				Result
S	R	W	X	V Mode (VS + VU)
0	0	0	0	Inaccessible region (Access Exception)

Bits on *hgpmpcfg* register				**Result**
0	0	0	1	Execute-only region
0	1	0	0	Read-only region
0	1	0	1	Read/Execute region
0	1	1	0	Read/Write region
0	1	1	1	Read/Write/Execute region
Others				Reserved

Chapter 5. Interaction with other proposals

This section discusses how SPMP interacts with other proposals.

RISC-V PMP enhancements: SPMP is compatible with the ePMP proposal and uses almost the same encoding as ePMP.

J-extension pointer masking proposal: When both PM and SPMP are used, SPMP checking should be performed using the actual addresses generated by PM (pointer masking).

Index

Bibliography