*(a) the guest OS within the VM assigns device interrupt handling to one hart while the device is also accessed by a different hart outside of an interrupt handler, or (b) control of the device (or partial control) is being migrated from one hart to another, such as for interrupt load balancing within the VM. For such cases, guest software within the VM is expected to properly coordinate access to the (emulated) device across multiple harts using mutex locks and/or interprocessor interrupts as usual, which in part entails executing I/O fences. But those I/O fences may not be sufficient if some of the device "I/O" is actually main memory, unknown to the guest. Setting FIOM=1 modifies those fences (and all other I/O fences executed in U-mode) to include main memory, too.*

*Software can always avoid the need to set FIOM by never using main memory to emulate a device memory buffer that should be I/O space. However, this choice usually requires trapping all U-mode accesses to the emulated buffer, which might have a noticeable impact on performance. The alternative offered by FIOM is sufficiently inexpensive to implement that we consider it worth supporting even if only rarely enabled.*

The definition of the CBZE field will be furnished by the forthcoming Zicboz extension. Its allocation within `senvcfg` may change prior to the ratification of that extension.

The definitions of the CBCFE and CBIE fields will be furnished by the forthcoming Zicbom extension. Their allocations within `senvcfg` may change prior to the ratification of that extension.

### 4.1.11 Supervisor Address Translation and Protection (`satp`) Register

The `satp` register is an SXLEN-bit read/write register, formatted as shown in Figure 4.14 for SXLEN=32 and Figure 4.15 for SXLEN=64, which controls supervisor-mode address translation and protection. This register holds the physical page number (PPN) of the root page table, i.e., its supervisor physical address divided by 4 KiB; an address space identifier (ASID), which facilitates address-translation fences on a per-address-space basis; and the MODE field, which selects the current address-translation scheme. Further details on the access to this register are described in Section 3.1.6.5.

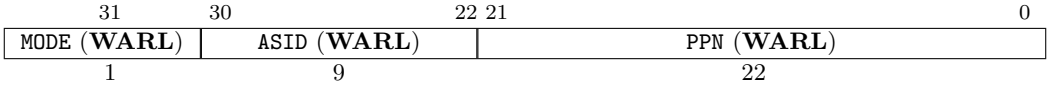


Figure 4.14: Supervisor address translation and protection register `satp` when SXLEN=32.

*Storing a PPN in* `satp`*, rather than a physical address, supports a physical address space larger than 4 GiB for RV32.*

*The* `satp`*.PPN field might not be capable of holding all physical page numbers. Some platform standards might place constraints on the values* `satp`*.PPN may assume, e.g., by requiring that all physical page numbers corresponding to main memory be representable.*

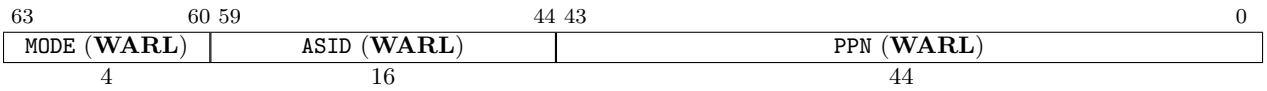


Figure 4.15: Supervisor address translation and protection register `satp` when SXLEN=64, for MODE values Bare, Sv39, Sv48, and Sv57.

*We store the ASID and the page table base address in the same CSR to allow the pair to be changed atomically on a context switch. Swapping them non-atomically could pollute the old virtual address space with new translations, or vice-versa. This approach also slightly reduces the cost of a context switch.*

Table 4.4 shows the encodings of the MODE field when SXLEN=32 and SXLEN=64. When MODE=Bare, supervisor virtual addresses are equal to supervisor physical addresses, and there is no additional memory protection beyond the physical memory protection scheme described in Section 3.7. To select MODE=Bare, software must write zero to the remaining fields of `satp` (bits 30–0 when SXLEN=32, or bits 59–0 when SXLEN=64). Attempting to select MODE=Bare with a nonzero pattern in the remaining fields has an UNSPECIFIED effect on the value that the remaining fields assume and an UNSPECIFIED effect on address translation and protection behavior.

When SXLEN=32, the `satp` encodings corresponding to MODE=Bare and ASID[8:7]=3 are designated for custom use, whereas the encodings corresponding to MODE=Bare and ASID[8:7]≠3 are reserved for future standard use. When SXLEN=64, all `satp` encodings corresponding to MODE=Bare are reserved for future standard use.

*Version 1.11 of this standard stated that the remaining fields in* `satp` *had no effect when MODE=Bare. Making these fields reserved facilitates future definition of additional translation and protection modes, particularly in RV32, for which all patterns of the existing MODE field have already been allocated.*

When SXLEN=32, the only other valid setting for MODE is Sv32, a paged virtual-memory scheme described in Section 4.3.

When SXLEN=64, three paged virtual-memory schemes are defined: Sv39, Sv48, and Sv57, described in Sections 4.4, 4.5, and 4.6, respectively. One additional scheme, Sv64, will be defined in a later version of this specification. The remaining MODE settings are reserved for future use and may define different interpretations of the other fields in `satp`.

Implementations are not required to support all MODE settings, and if `satp` is written with an unsupported MODE, the entire write has no effect; no fields in `satp` are modified.

The number of ASID bits is UNSPECIFIED and may be zero. The number of implemented ASID bits, termed *ASIDLEN*, may be determined by writing one to every bit position in the ASID field, then reading back the value in `satp` to see which bit positions in the ASID field hold a one. The least-significant bits of ASID are implemented first: that is, if ASIDLEN > 0, ASID[ASIDLEN-1:0] is writable. The maximal value of ASIDLEN, termed ASIDMAX, is 9 for Sv32 or 16 for Sv39, Sv48, and Sv57.

*For many applications, the choice of page size has a substantial performance impact. A large page size increases TLB reach and loosens the associativity constraints on virtually indexed, physically tagged caches. At the same time, large pages exacerbate internal fragmentation, wasting physical memory and possibly cache capacity.*

*After much deliberation, we have settled on a conventional page size of 4 KiB for both RV32 and RV64. We expect this decision to ease the porting of low-level runtime software and device drivers. The TLB reach problem is ameliorated by transparent superpage support in modern*

| SXLEN=32 | | |
|---|---|---|
| Value | Name | Description |
| 0 | Bare | No translation or protection. |
| 1 | Sv32 | Page-based 32-bit virtual addressing (see Section 4.3). |

| SXLEN=64 | | |
|---|---|---|
| Value | Name | Description |
| 0 | Bare | No translation or protection. |
| 1–7 | — | *Reserved for standard use* |
| 8 | Sv39 | Page-based 39-bit virtual addressing (see Section 4.4). |
| 9 | Sv48 | Page-based 48-bit virtual addressing (see Section 4.5). |
| 10 | Sv57 | Page-based 57-bit virtual addressing (see Section 4.6). |
| 11 | *Sv64* | *Reserved for page-based 64-bit virtual addressing.* |
| 12–13 | — | *Reserved for standard use* |
| 14–15 | — | *Designated for custom use* |

Table 4.4: Encoding of `satp` MODE field.

*operating systems [2]. Additionally, multi-level TLB hierarchies are quite inexpensive relative to the multi-level cache hierarchies whose address space they map.*

The `satp` register is considered *active* when the effective privilege mode is S-mode or U-mode. Executions of the address-translation algorithm may only begin using a given value of `satp` when `satp` is active.

---

*Translations that began while* `satp` *was active are not required to complete or terminate when* `satp` *is no longer active, unless an SFENCE.VMA instruction matching the address and ASID is executed. The SFENCE.VMA instruction must be used to ensure that updates to the address-translation data structures are observed by subsequent implicit reads to those structures by a hart.*

Note that writing `satp` does not imply any ordering constraints between page-table updates and subsequent address translations, nor does it imply any invalidation of address-translation caches. If the new address space's page tables have been modified, or if an ASID is reused, it may be necessary to execute an SFENCE.VMA instruction (see Section 4.2.1) after, or in some cases before, writing `satp`.

---

*Not imposing upon implementations to flush address-translation caches upon* `satp` *writes reduces the cost of context switches, provided a sufficiently large ASID space.*

## 4.2 Supervisor Instructions

In addition to the SRET instruction defined in Section 3.3.2, one new supervisor-level instruction is provided.