RISC-V Privileged Mode Review

ZZH

CSRs	指令	其他

CSRs

1. ISA 寄存器 misa

1.1 M 模式:misa

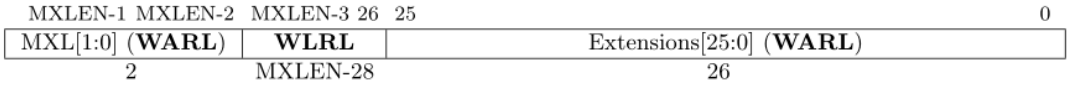


Figure 3.1: Machine ISA register (misa).

misa CSR 是一个 WARL 读写寄存器，报告 Hart 支持的 ISA。

MXL（Machine XLEN）字段对本机基本整数 ISA 宽度进行编码。

扩展字段对标准扩展的存在进行编码，字母表中每个字母有一个字段（第 7 字段是新增的 H 扩展）

2. 供应商 ID 寄存器 vendorid

2.1 M 模式：mvendorid

Figure 3.2: Vendor ID register (mvendorid).

mvendorid CSR 是一个 32 字段只读寄存器，提供内核提供者的 JEDEC 制造商标识。

3. 体系结构 ID 寄存器 archid

3.1 M 模式：marchid

Figure 3.3: Machine Architecture ID register (`marchid`).

marchid CSR 是一个 MXLEN 字段只读寄存器，编码 Hart 的基本微体系结构。

4. 实现 ID 寄存器 impid

4.1 M 模式：mimpid

Figure 3.4: Machine Implementation ID register (`mimpid`).

mimpid CSR 提供了处理器实现版本的唯一编码。

5. 硬件线程 ID 寄存器 hartid

5.1 M 模式：mhartid

Figure 3.5: Hart ID register (`mhartid`).

mhartid CSR 是一个 MXLEN 字段只读寄存器，包含运行代码的硬件线程的整数 ID。

6. 状态寄存器 status

6.1 M 模式：mstatus

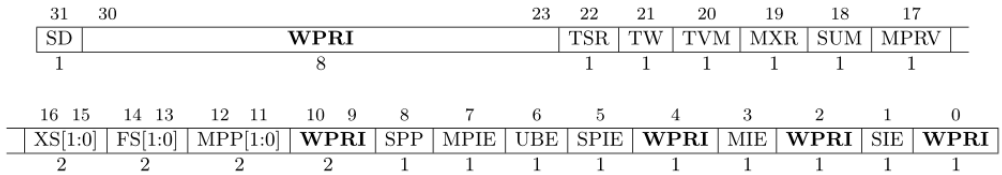


Figure 3.6: Machine-mode status register (`mstatus`) for RV32.

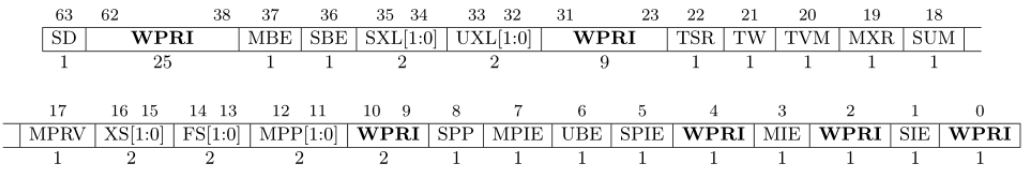


Figure 3.7: Machine-mode status register (`mstatus`) for RV64.

mstatus 状态寄存器是一个 MXLEN 字段读/写寄存器，跟踪并控制 Hart 的当前操作状态。

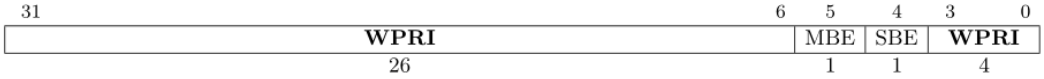


Figure 3.8: Additional machine-mode status register (`mstatush`) for RV32.

仅对于 RV32，mstatus 具有一个 32 字段读/写寄存器，对于 RV64，mstatus 字段 30：4 通常包含与 mstatus 字段 62：36 相同的字段。字段 SD、SXL 和 UXL 在 mstatush 中不存在。

MIE，SIE（中断使能）字段：全局中断使能字段

MPIE，SPIE 字段（之前的中断使能）：保存先前的全局中断使能字段

MPP，SPP 字段（之前的模式）：保持先前的特权模式

SXL，UXL 字段：分别控制 S 模式和 U 模式的 XLEN 的值。

MPRV（修改权限）字段：修改加载和存储在所有权限模式下执行的权限级别。

MXR（使可执行文件可读）字段：修改加载访问虚拟内存的权限

SUM 字段：修改 S 模式加载和存储访问虚拟内存的权限。

MBE（字符顺序）字段：控制是否从 M 模式进行非指令提取内存访问是小端还是大端。

TVM（陷阱虚拟内存）字段：支持拦截 hypervisor 虚拟内存管理操作。

TW（超时等待）字段：支持截取 WFI 指令。

TSR（陷阱 SRET）字段：支持拦截 hypervisor 异常返回指令 SRET

FS 字段：编码浮点单元的状态。

XS 字段：编码附加用户模式扩展的状态和相关状态。

SD 字段：汇总 FS 字段或 XS 字段是否表示存在需要将扩展用户上下文保存到存储器的某些 Dirty 状态。

6.2 S 模式：sstatus

sstatus 寄存器是 SXLEN 字段读/写寄存器。mstatus 的受限视图在 S 级别 ISA 中显示为 sstatus 寄存器，sstatus 寄存器是 mstatus 寄存器的子集。

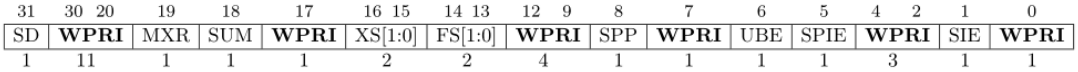


Figure 4.1: Supervisor-mode status register (`sstatus`) when SXLEN=32.

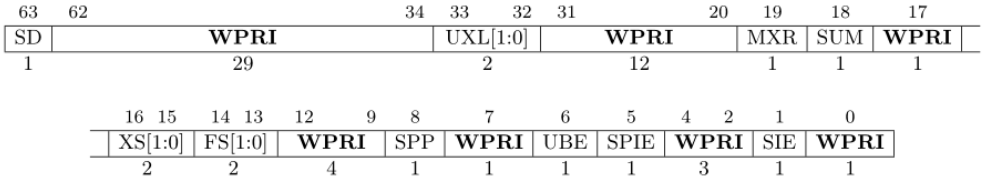


Figure 4.2: Supervisor-mode status register (`sstatus`) when SXLEN=64.

SIE，SPIE，UBE，UXL，SUM，MXR，UBE，SPP 字段同 mstatus 寄存器。

6.3 HS 模式：hstatus

hstatus 寄存器提供了类似于 mstatus 寄存器的工具，用于跟踪和控制 VS 模式 guest 的异常行为。

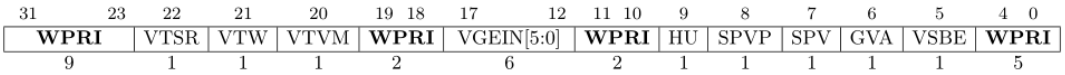


Figure 8.1: Hypervisor status register (`hstatus`) when HSXLEN=32.

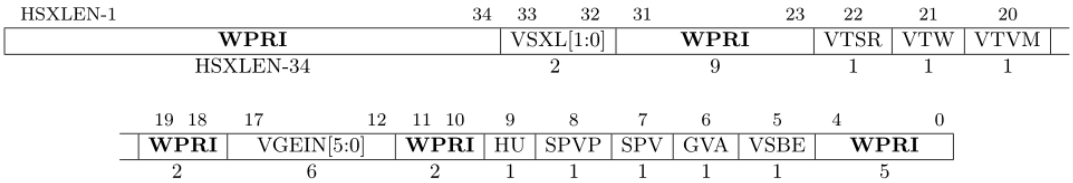


Figure 8.2: Hypervisor status register (`hstatus`) when HSXLEN=64.

VSXL 字段：控制 VS 模式的有效 XLEN（称为 VSXLEN）。

（hstatus 字段 VTSR、VTW 和 VTVM 的定义类似于状态字段 TSR、TW 和 TVM，但仅在 VS 模式下影响执行，并导致伪指令异常而不是非法指令异常。）

VGEIN（虚拟 guest 外部中断号）字段：为 VS 级外部中断选择 guest 外部中断源。

HU（U 模式中的 hypervisor）字段：控制虚拟机 Load/Store 指令 HLV、HLVX 和 HSV 是否也可以在 U 模式中使用。

SPV（Supervisor Previous Virtualization 模式）字段：设置为在陷阱时虚拟化模式 V 的值。

SPVP（supervisor 先前虚拟特权）字段：在陷阱发生时设置为名义上的（nominal）特权模式，和 sstatus.SPP 相同。

GVA（guest 虚拟地址）字段：对于将 guest 虚拟地址写入 stval 的任何陷阱（断点、地址未对齐、访问错误、页面错误或 guest 页面错误），GVA 会设置为 1。

VSBE 字段：控制从 VS 模式进行的显式内存访问的字符顺序。

6.4 VS 模式：vsstatus

vsstatus 寄存器是一个 VSXLEN 字段读/写寄存器，它是 VS 模式下 supervisor sstatus 寄存器的版本。

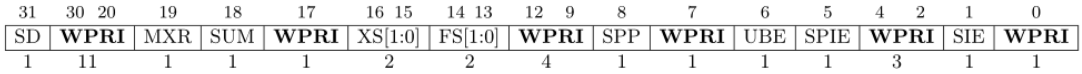


Figure 8.21: Virtual supervisor status register (`vsstatus`) when VSXLEN=32.

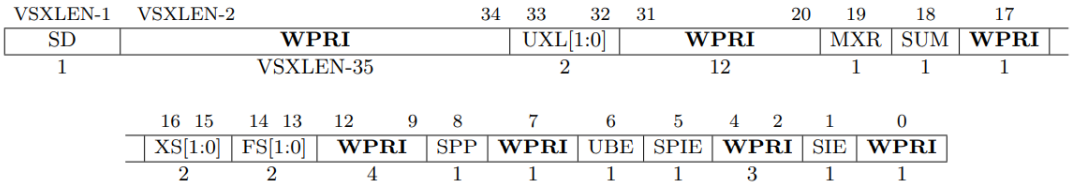


Figure 8.22: Virtual supervisor status register (`vsstatus`) when VSXLEN=64.

只读字段 SD 和 XS 概述了扩展上下文状态，因为它只对 VS 模式可见。比如说，HS 级别下的 sstatus.FS 的值并不会影响 vsstatus.SD。

当 V=0 时，除非使用虚拟机 Load/Store（HLV、HLVX 或 HSV）或 mstatus 寄存器中的 MPRV 功能来执行 load 或 store（假设 V=1），否则 vsstatus 不会直接影响机器的行为。

6.5 H 扩展 M 模式：mstatus 和 mstatush

Hypervisor 扩展向机器级别的 mstatus 或者 mstatush CSR 添加了两个字段 MPV 和 GVA，并修改了几个现有 mstatus 域的行为。

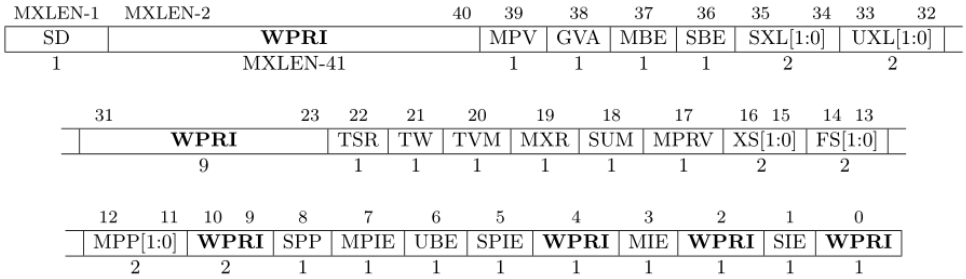


Figure 8.34: Machine status register (`mstatus`) for RV64 when the hypervisor extension is implemented.

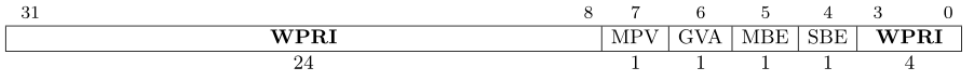


Figure 8.35: Additional machine status register (`mstatush`) for RV32 when the hypervisor extension is implemented. The format of `mstatus` is unchanged for RV32.

每当陷阱进入 M 模式时，实施都会写入 MPV 字段（机器先前虚拟化模式）。正如在陷阱时将 MPP 字段设置为（标称）特权模式一样，MPV 字段被设置为在陷阱时虚拟化模式 V 的值。

mstatus 寄存器是 HS 级别 sstatus 寄存器的扩展集，但不是 vsstatus 的扩展集。

7. 陷阱向量基地址寄存器（tvec）

7.1 M 模式：mtvec

mtvec 寄存器是一个 MXLEN-bit 读/写寄存器，用于保存陷阱矢量配置，由矢量基址（BASE）和矢量模式（MODE）组成。

Figure 3.9: Machine trap-vector base-address register (`mtvec`).

BASE 字段中的值必须始终在 4 字节边界上对齐，并且模式设置可能会对基准字段中的值施加额外的对齐约束。

Value	Name	Description
0	Direct	All exceptions set `pc` to BASE.
1	Vectored	Asynchronous interrupts set `pc` to BASE+4×cause.
≥2	—	*Reserved*

Table 3.5: Encoding of `mtvec` MODE field.

关于 MODE 字段，当 MODE=DIRECT 时，所有进入 machine 模式的陷阱都会导致将 PC 设置为基字段中的地址。当 MODE=VECTORED 时，所有进入 machine 模式的同步异常导致 PC 被设置为基字段中的地址，而中断导致 PC 被设置为基字段中的地址加上四倍的中断原因号。

7.2 S 模式：stvec

stvec 寄存器是一个 SXLEN 位读/写寄存器，保存陷阱矢量配置，由矢量基址（base）和矢量模式（mode）组成。

Figure 4.3: Supervisor trap vector base address register (`stvec`).

不同：除“direct”之外的 MODE 设置可能会对 BASE 中的值施加其他对齐约束领域。

7.3 VS 模式：vstvec

vstvec 寄存器是 VSXLEN 位读/写寄存器，是 VS 模式版本的管理寄存器，当 V=1 时，vstvec 替换通常的 stvec，因此通常读取或修改 stvec 的指令实际上访问的是 vstvec。当 V=0 时，vstvec 不会直接影响机器的行为。

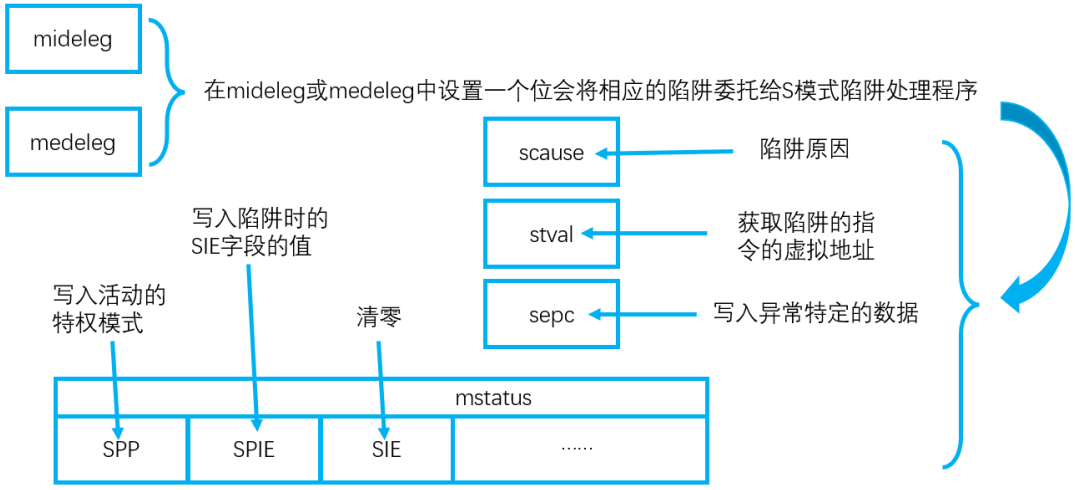
VSXLEN-1	2 1	0
BASE[VSXLEN-1:2] (**WARL**)	MODE (**WARL**)	
VSXLEN-2	2	

Figure 8.27: Virtual supervisor trap vector base address register (`vstvec`).

8. 陷阱委托寄存器（edeleg 和 ideleg）

8.1 M 模式：medeleg 和 mideleg

machine 异常委派寄存器（medeleg）和 machine 中断委派寄存器（mideleg）是 MXLEN 位读/写寄存器。为了提高性能，实现可以在 medeleg 和 mideleg 内提供单独的读/写位，以指示某些异常和中断应该由较低的特权级别直接处理。



（画的比较丑，请见谅）

陷阱永远不会从权限较高的模式转换到权限较低的模式。

8.2 HS 模式：hedeleg 和 hideleg

hedeleg 和 hideleg 寄存器是 HSXLEN 位读/写寄存器。默认情况下，任何特权级别的所有陷阱都在 M 模式下处理，尽管 M 模式通常使用 hedeleg 和 hideleg CSR 将一些陷阱委派给 HS 模式。hedeleg 和 hideleg CSR 允许将这些陷阱进一步委托给 VS 模式的 guest；它们的布局与 medeleg 和 mideleg 相同。

8.3 H 扩展 M 模式：mideleg

当实现 hypervisor 扩展时，mideleg 的位 10、6 和 2（对应于标准 VS 级中断）都是只读的。此外，如果实现了任何 guest 外部中断（GEILEN 非零），则 mideleg 的位 12（对应于 supervisor 级 guest 外部中断）也是只读的。VS 级中断和 guest 外部中断总是通过 M 模式委托给 HS 模式。

对于位为零的 mideleg，hideleg、hip 和 leave 中的相应位是只读零。

9. 中断寄存器（ip 和 ie）

9.1 M 模式：mip 和 mie

mip 寄存器是包含挂起中断信息的 MXLEN 位读/写寄存器，而 mie 寄存器是对应的包含中断使能位的 MXLEN 位读/写寄存器。中断原因编号对应于 mip 和 mie 的第 i 位。位 15：0 仅分配给标准中断原因，而位 16 及以上指定用于平台或自定义。

15 12	11	10	9	8	7	6	5	4	3	2	1	0
0	MEIP	0	SEIP	0	MTIP	0	STIP	0	MSIP	0	SSIP	0
4	1	1	1	1	1	1	1	1	1	1	1	1

Figure 3.14: Standard portion (bits 15:0) of `mip`.

15 12	11	10	9	8	7	6	5	4	3	2	1	0
0	MEIE	0	SEIE	0	MTIE	0	STIE	0	MSIE	0	SSIE	0
4	1	1	1	1	1	1	1	1	1	1	1	1

Figure 3.15: Standard portion (bits 15:0) of `mie`.

mip.MEIP 和 mie.MEIE 位：机器级外部中断的中断挂起和中断使能位。
mip.MTIP 和 mie.MTIE 位：机器定时器中断的中断挂起位和中断使能位。
mip.MSIP 和 mie.MSIE 位：机器级软件中断的中断挂起和中断使能位。
如果实施 hypervisor 模式：
mip.STIP 和 mie.STIE 位：hypervisor 级别定时器中断的中断挂起和中断使能位。
mip.SSIP 和 mie.SSIE 位：hypervisor 级软件中断的中断挂起和中断使能位。

9.2 S 模式：sip 和 sie

mip 和 mie 的受限视图显示为 supervisor 等级的 sip 和 sie。sip 寄存器是包含挂起中断信息的 SXLEN 位读/写寄存器，而 sie 寄存器对应于包含中断使能位的 SXLEN 位读/写寄存器。

15 10	9	8 6	5	4 2	1	0
0	SEIP	0	STIP	0	SSIP	0
6	1	3	1	3	1	1

Figure 4.6: Standard portion (bits 15:0) of `sip`.

15 10	9	8 6	5	4 2	1	0
0	SEIE	0	STIE	0	SSIE	0
6	1	3	1	3	1	1

Figure 4.7: Standard portion (bits 15:0) of `sie`.

sip.SEIP 和 sie.SEIE 位：监控级外部中断的中断挂起和中断使能位。
sip.STIP 和 sie.STIE 位：监控级定时器中断的中断挂起和中断使能位。
sip.SSIP 和 sie.SSIE 位：监控级软件中断的中断挂起和中断使能位。

9.3 HS 模式：hvip，hip 和 hie

hvip 寄存器是一个 HSXLEN 位读/写寄存器，hypervisor 可以写入该寄存器，以指示用于 VS 模式的虚拟中断。

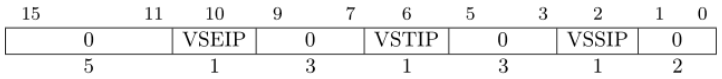


Figure 8.6: Standard portion (bits 15:0) of `hvip`.

VSEIP 位：VS 级外部中断有效控制。

VSTIP 位：VS 级定时器中断有效控制。

VSSIP 位：VS 级软件中断有效控制。

寄存器 hip 和 hie 是 HSXLEN 位读/写寄存器，分别补充 HS 级的 sip 和 sie。hip 寄存器指示挂起的 VS 级中断和特定于 hypervisor 的中断，同时包含用于相同中断的使能位。

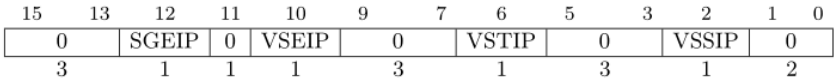


Figure 8.9: Standard portion (bits 15:0) of `hip`.

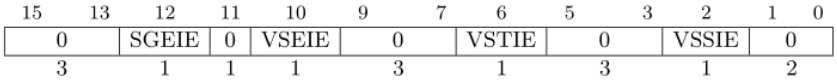


Figure 8.10: Standard portion (bits 15:0) of `hie`.

hip.SGEIP 和 hie.SGEIE 位：supervisor 级（HS 级）guest 外部中断的中断挂起和中断使能位。

hip.VSEIP 和 hie.VSEIE 位：VS 级外部中断的中断挂起和中断使能位。

hip.VSTIP 和 hie.VSTIE 位：VS 级定时器中断的中断挂起和中断使能位。

hip.VSSIP 和 hie.VSSIE 位：VS 级软件中断的中断挂起和中断使能位。

9.4 VS 模式：vsip 和 vsie

vsip 和 vsie 寄存器是 VSXLEN 位读/写寄存器，它们是 VS 模式下的 supervisor sip 和 sie CSR 版本。当 V=1 时，vsip 和 vsie 将替代通常的 sip 和 sie，因此通常读取或修改 sip 和 sie 的指令实际上访问的是 VSIP/VSI。

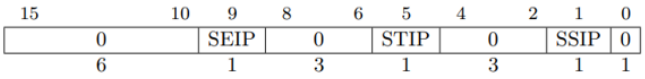


Figure 8.25: Standard portion (bits 15:0) of `vsip`.

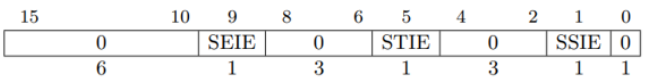


Figure 8.26: Standard portion (bits 15:0) of `vsie`.

当 hideleg 的第 10 位为零时，vsip.SEIP 和 vsie.SEIE 为只读零。否则，vsip.SEIP 和 vsie.SEIE 是 hip.vseip 和 hie.vseie 的别名。

当 hideleg 的第 6 位为零时，vsip.STIP 和 vsie.STIE 为只读零。否则，vsip.STIP 和 vsie.STIE 是 hip.VSTIP 和 hie.VSTIE 的别名

当 hideleg 的第 2 位为零时，vsip.SSIP 和 vsie.SSIE 为只读零。否则，vsip.SSIP 和 vsie.SSIE 是 hip.VSSIP 和 hie.VSSIE 的别名。

9.5 H 扩展 M 模式：mip 和 mie

Hypervisor 扩展为添加了 hypervisor 的中断提供 mip 和 mie 寄存器附加的活动位。

15 — 13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	SGEIP	MEIP	VSEIP	SEIP	0	MTIP	VSTIP	STIP	0	MSIP	VSSIP	SSIP	0
3	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 8.36: Standard portion (bits 15:0) of `mip`.

15 — 13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	SGEIE	MEIE	VSEIE	SEIE	0	MTIE	VSTIE	STIE	0	MSIE	VSSIE	SSIE	0
3	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 8.37: Standard portion (bits 15:0) of `mie`.

mip 中的 SGEIP、VSEIP、VSTIP 和 VSSIP 位是 hypervisor CSR hip 中相同位的别名。

mie 中的 SGEIE、VSEIE、VSTIE 和 VSSIE 位是 hypervisor CSR hie 中相同位的别名。

10. 硬件性能监视器

10.1 M 模式:硬件性能监视器

mhpmcounters 计数器是 WARL 寄存器，在 RV32 和 RV64 上支持高达 64 位的精度。

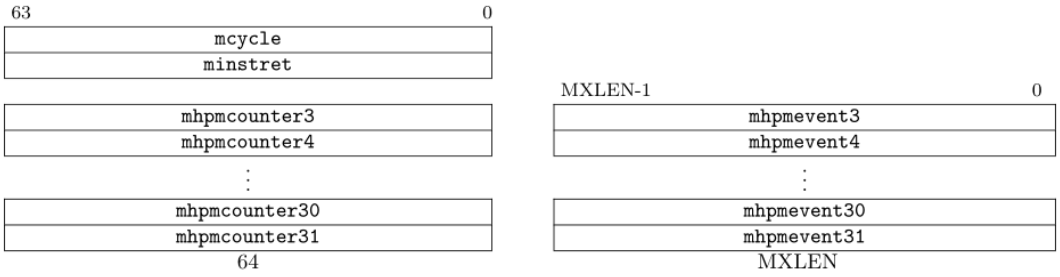


Figure 3.16: Hardware performance monitor counters.

仅在 RV32 上，对 mcycle、minstret 和 mhpmcountern CSRs 的读取返回低 32 位，而对 mcycleh、minstreth 和 mhpmcounternh CSRs 的读取返回相应计数器的 63-32 位。

10.2 S 模式：计时器和性能计数器

监控软件使用与用户模式软件相同的硬件性能监控工具，包括 time、cycle 和 instret CSRs。实现应该提供一种修改计数器值的机制。

11. 计数器使能寄存器（counteren）

11.1 M 模式：mcounteren

计数器使能寄存器 mcounteren 是 32 位寄存器，它们将硬件性能监视计数器的可用性控制到次低特权模式。

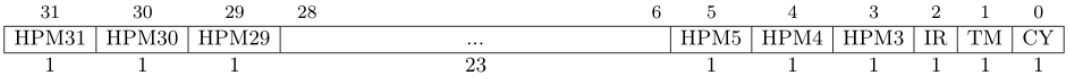


Figure 3.18: Counter-enable registers (`mcounteren` and `scounteren`).

当 mcounteren 寄存器中的 CY、TM、IR 或 HPMn 位被清除时，在 S 模式或 U 模式下执行时尝试读取 cycle、time、instret 或 hpmcountern 寄存器将导致非法指令异常。

11.2 S 模式：scounteren

scounteren 计数器使能寄存器是一个 32 位寄存器，用于控制硬件性能监控计数器在 U 模式下的可用性。

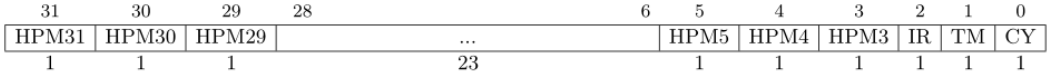


Figure 4.8: Counter-enable register (`scounteren`).

当计数器寄存器中的 CY、TM、IR 或 HPMn 位清零时，在 U 模式下执行时试图读取 cycle、time、instret 或 hpmcountern 寄存器将导致非法指令异常。当其中一个位被置位时，允许访问相应的寄存器。

11.3 HS 模式：hcounteren

hcounteren 计数器使能寄存器是一个 32 位寄存器，用于控制硬件性能监视计数器对 guest 虚拟机的可用性。

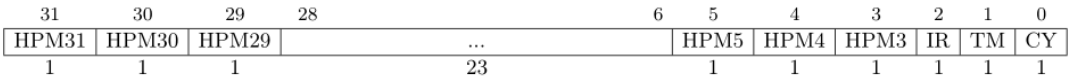


Figure 8.14: Hypervisor counter-enable register (`hcounteren`).

当 hcounteren 寄存器中的 CY、TM、IR 或 HPMn 位被清除时，如果相同的位在 mcounteren 中为 1，在 V=1 的情况下尝试读取 cycle、time、instret 或 hpmcountern 寄存器将导致伪指令异常。当这些位中的一个被设置时，V=1 时允许访问相应的寄存器。在 VU 模式下，除非在 hcounteren 和 scounteren 中都设置了适用的位，否则计数器是不可读的。

12. 计数器抑制 CSR（countinhibit）

12.1 M 模式：mcountinhibit

计数器抑制寄存器 mcountinhibit 是一个 32 位的 WARL 寄存器，它控制哪些硬件性能监控计数器递增。该寄存器中的设置仅控制计数器是否递增；其可访问性不受该寄存器设置的影响。

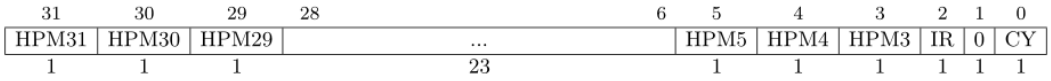


Figure 3.19: Counter-inhibit register `mcountinhibit`.

当 mcountinhibit 寄存器中的 CY、IR 或 HPMn 的 bit 位被清除时，cycle、instret 或 hpmcountern 寄存器将照常递增。设置 CY、IR 或 HPMn 的 bit 位时，相应的计数器不会递增。

13. 暂存寄存器（scratch）

13.1 M 模式：mscratch

mscratch 寄存器是 machine 模式专用的 MXLEN 位读/写寄存器。通常，它用于保存指向 machine 模式 Hart 本地上下文空间的指针，并在进入 M 模式陷阱处理程序时与用户寄存器交换。

Figure 3.20: Machine-mode scratch register.

13.2 S 模式：sscratch

sscratch 寄存器是一个 SXLEN 位读/写寄存器，专用于监控程序。通常，当 Hart 执行用户代码时，sscratch 是用于保存一个指向 Hart 本地管理器上下文的指针。在陷阱处理程序开始时，sscratch 与用户寄存器交换，以提供初始工作寄存器。

13.3 VS 模式：vsscratch

vsscratch 寄存器是一个 VSXLEN 位读/写寄存器，它是 VS 模式下的监控寄存器 scratch 版本。当 V=1 时，vsscratch 将替代通常的 scratch，因此通常读取或修改 scratch 的指令实际上访问的是 vsscratch 而不是 scratch。scratch 的内容永远不会直接影响机器的行为

4. 异常程序计数器的（epc）

14.1 M 模式：mepc

mepc 是一个 MXLEN 位读/写寄存器。mepc 的低位（mepc[0]）始终为零。在仅支持 IALIGN=32 的实现中，两个低位（mepc[1：0]）始终为零。

如果实现允许 IALIGN 为 16 或 32（例如，通过更改 CSR misa），则只要 IALIGN=32，读取时 mepc[1]位就会被掩码，使其显示为 0。此掩码也适用于 MRET 指令的隐式读取。尽管被掩码，但当 IALIGN=32 时，MEPC[1]保持可写。

mepc 是一个 WARL 寄存器，必须能够保存所有有效的物理和虚拟地址。它不需要能够保存所有可能的无效地址。在将一些无效地址模式写入 pc 之前，实现可能会将其转换为其他无效地址。

当陷阱进入 M 模式时，mepc 是写入被中断或遇到异常的指令的虚拟地址。否则，mepc 永远不是通过实现写入的。

14.2 S 模式：sepc

sepc 是一个 SXLEN 位读/写寄存器。其余同 mepc。

14.3 VS 模式：vsepc

vsepc 寄存器是一个 VSXLEN 位读/写寄存器，是 VS 模式版本下的 supervisor sepc 寄存器。当 V=1 时，vsepc 将替代通常的 sepc，因此通常读取或修改的指令实际上访问的是 vsepc。当 V=0 时，vsepc 不会直接影响机器的行为。

vsepc 是一个 WARL 寄存器，必须能够保存与 sepc 可以保存的值集合相同的值。

15. 原因寄存器（cause）

15.1 M 模式：mcause

mcause 寄存器是一个 MXLEN 位读写寄存器。当陷阱进入 M 模式时，mcause 会写入一个代码，指示导致陷阱的事件。

Figure 3.22: Machine Cause register `mcause`.

如果陷阱是由中断引起的，则会设置原因寄存器中的中断位。异常代码字段包含标识最后一个异常的代码。表 3.6 列出了可能的 machine 级别异常代码。异常代码是一个 WLRL 字

段，因此仅保证包含支持的异常代码。

Interrupt	Exception Code	Description
1	0	User software interrupt
1	1	Supervisor software interrupt
1	2	*Reserved for future standard use*
1	3	Machine software interrupt
1	4	User timer interrupt
1	5	Supervisor timer interrupt
1	6	*Reserved for future standard use*
1	7	Machine timer interrupt
1	8	User external interrupt
1	9	Supervisor external interrupt
1	10	*Reserved for future standard use*
1	11	Machine external interrupt
1	12–15	*Reserved for future standard use*
1	≥16	*Reserved for platform use*
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode
0	9	Environment call from S-mode
0	10	*Reserved*
0	11	Environment call from M-mode
0	12	Instruction page fault
0	13	Load page fault
0	14	*Reserved for future standard use*
0	15	Store/AMO page fault
0	16–23	*Reserved for future standard use*
0	24–31	*Reserved for custom use*
0	32–47	*Reserved for future standard use*
0	48–63	*Reserved for custom use*
0	≥64	*Reserved for future standard use*

Table 3.6: Machine cause register (`mcause`) values after trap.

15.2 S 模式：scause

scause 寄存器是一个 SXLEN 位读写寄存器。当陷阱进入 S 模式时，scause 会写入一个代码，指示导致陷阱的事件。否则，scause 永远不会通过实现编写。

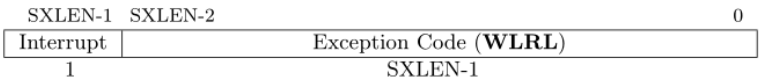


Figure 4.11: Supervisor Cause register `scause`.

如果陷阱是由中断引起的，则设置原因寄存器中的中断位。异常代码字段包含标识最后一个异常的代码。表 4.2 列出了当前 supervisor ISAs 的可能异常代码。异常代码是一个 WLRL 字段，它需要保存值 0–31（即必须实现位 4–0），但否则只能保证保存支持的异常代码。

Interrupt	Exception Code	Description
1	0	*Reserved*
1	1	Supervisor software interrupt
1	2–4	*Reserved*
1	5	Supervisor timer interrupt
1	6–8	*Reserved*
1	9	Supervisor external interrupt
1	10–15	*Reserved*
1	≥16	*Designated for platform use*
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode
0	9	Environment call from S-mode
0	10–11	*Reserved*
0	12	Instruction page fault
0	13	Load page fault
0	14	*Reserved*
0	15	Store/AMO page fault
0	16–23	*Reserved*
0	24–31	*Designated for custom use*
0	32–47	*Reserved*
0	48–63	*Designated for custom use*
0	≥64	*Reserved*

Table 4.2: Supervisor cause register (`scause`) values after trap. Synchronous exception priorities are given by Table 3.7.

15.3 VS 模式：vscause

vscause 寄存器是一个 VSXLEN 位读/写寄存器，是 VS 模式下监控寄存器的版本，V=1 时，用 vscause 代替通常的原因，因此通常读取或修改 scause 的指令实际上访问 vscause。当 V=0 时，vscause 不会直接影响机器的行为。

vscause 是一个 WLRL 寄存器，它必须能够保存与 scause 可以保存的值集相同的值。

16. 陷阱值寄存器（tval）

16.1 M 模式：mtval

mtval 寄存器是一个 MXLEN 位读写寄存器。当陷阱进入 M 模式时，mtval 值要么设置为零，要么写入特定于异常的信息，以帮助软件处理陷阱。硬件平台将指定哪些异常必须信息性地设置，以及哪些异常可以无条件地将其设置为零。如果硬件平台指定没有异常将 mtval 值设置为非零值，则 mtval 值为只读零。

如果在指令提取、加载或存储上发生断点、地址未对齐、访问错误或页面错误异常时，使用非零值写入 mtval 值，则 mtval 值将包含出错的虚拟地址。

当未对齐的加载或存储导致访问错误或页面错误异常时，如果使用非零值写入 mtval 值，则 mtval 值将包含导致错误的访问部分的虚拟地址。

如果在具有可变长度指令的系统上发生指令访问错误或页面错误异常时，使用非零值写入 mtval 值，则 mtval 值将包含导致错误的指令部分的虚拟地址，而 mepc 将指向指令的开头。

对于其他陷阱，mtval 值设置为零，但未来的标准可能会重新定义其他陷阱的 mtval 值设置。

16.2 S 模式：stval

stval 寄存器是一个 SXLEN 位读写寄存器。其余同 mtval。

16.3 HS 模式：htval

htval 寄存器是一个 HSXLEN 位读/写寄存器。当陷阱进入 HS 模式时，htval 会写入附加的特定于异常的信息，以及 stval，以帮助软件处理陷阱。

当 guest 页面故障陷阱进入 HS 模式时，htval 会写入零或出现故障的 guest 物理地址，向右移位 2 位。对于其他陷阱，htval 值设置为零，但未来的标准或扩展可能会重新定义 htval 对其他陷阱的设置。

guest 页面错误可能是由于第一阶段（VS 阶段）地址转换期间的隐式存储器访问引起的，在这种情况下，写入 htval 的 guest 物理地址是出错的隐式存储器访问的 guest 物理地址。

否则，对于导致 guest 页错误的未对齐加载和存储，非零 guest 物理地址 htval 对应于由 stval 中的虚拟地址指示的访问的出错部分。对于具有可变长度指令的系统上的指令 guest 页错误，非零 htval 对应于由 stval 中的虚拟地址指示的指令的出错部分。

16.4 VS 模式：vstval

vstval 寄存器是 VSXLEN 位读/写寄存器，是 VS 模式版本下的 supervisor stval 寄存器。当 V=1 时，vstval 值将替换通常的 stval，因此通常读取或修改 stval 值的指令实际上访问的是 vstval 值。当 V=0 时，vstval 值不会直接影响机器的行为。

vstval 是一个 WARL 寄存器，它必须能够保存与 scause 可以保存的值集相同的值。

16.5 H 扩展 M 模式：mtval2

mtval2 寄存器是一个 MXLEN 位读/写寄存器。当陷阱进入 M 模式时，mtval2 会写入额外的特定于异常的信息，以帮助软件处理陷阱。

当 guest 页面故障陷阱进入 M 模式时，mtval2 同 htval。

17. 配置指针寄存器（configptr）

17.1 M 模式：mconfigptr

mconfigptr 是一个 MXLEN 位只读 CSR，保存配置数据结构的物理地址。软件可以遍历这个数据结构来发现关于 Harts、平台及其配置的信息。

mconfigptr 必须实现，但它可能为零，表示配置数据结构不存在，或者必须使用替代机制来定位它。

18. 环境配置寄存器（envcfg 和 envcfgh）

18.1 M 模式：menvcfg 和 menvcfgh

menvcfg CSR 是一个 MXLEN 位读/写寄存器，格式为 MXLEN=64，如图 3.25 所示，它控制特权低于 M 的模式的执行环境的某些特性。

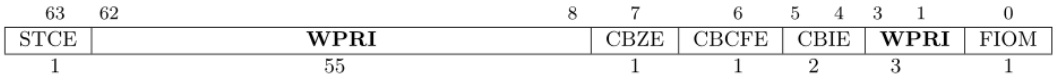


Figure 3.25: Machine environment configuration register (`menvcfg`) for MXLEN=64.

如果位 FIOM（Fence off I/O implies Memory）在环境中设置为 1，则在特权低于 M 的模式下执行的 Fence 指令被修改，因此对设备输入/输出的访问进行排序的要求也意味着对主内存访问进行排序的要求。表 3.8 详细说明了当 FIOM=1 时，特权低于 M 的模式的 Fence 指令位 PI、PO、SI 和 SO 的修改解释。

Instruction bit	Meaning when set
PI	Predecessor device input and memory reads (PR implied)
PO	Predecessor device output and memory writes (PW implied)
SI	Successor device input and memory reads (SR implied)
SO	Successor device output and memory writes (SW implied)

Table 3.8: Modified interpretation of FENCE predecessor and successor sets for modes less privileged than M when FIOM=1.

类似地，当 FIOM=1 时，对于特权低于 M 的模式，如果一个原子指令访问一个按设备输入/输出排序的区域，并设置了它的 aq 和/或者 rl 位，那么该指令就像访问设备输入/输出和存储器一样被排序。

STEC，CBZE，CBCFE，CBIE 字段将由新的扩展来提供，文档中还未说明。

当 MXLEN=32 时，menvcfg 包含的字段与 MXLEN=64 时 menvcfg 的位 31：0 相同。此外，当 MXLEN=32 时，menvcfgh 这是一个 32 位读/写寄存器，包含与 MXLEN=64 时的 63：32 位相同的字段。当 MXLEN=64 时，寄存器 menvcfgh 不存在。

18.2 S 模式：senvcfg

senvcfg CSR 是一个 SXLEN 位读/写寄存器，格式如图 4.13 所示，控制 U 模式执行环境的某些特性。

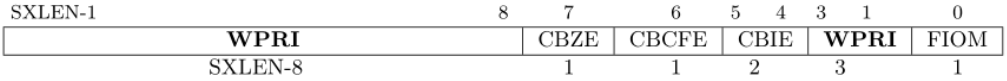


Figure 4.13: Supervisor environment configuration register (**senvcfg**).

FIOM 位同 menvcfg 的 FIOM 位的功能。

CBZE，CBCFE，CBIE 将由新的扩展提供，文档里并没有进行说明。

19. 安全配置寄存器（seccfg）

19.1 M 模式：mseccfg

mseccfg 是一个可选的 MXLEN 位读/写寄存器，格式如图 3.26 所示，控制安全特性。仅当 MXLEN=32 时，mseccfgh 是一个 32 位读/写寄存器，包含与 MXLEN=64 时 mseccfg 的[63：32]位相同的字段。

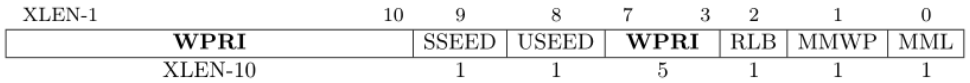


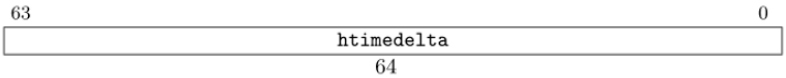
Figure 3.26: Machine security configuration register (**mseccfg**).

SSEED，USEEDRLB，MMWP 和 MML 字段将由新的扩展提供，文档里并没有进行说明。

20. 时间增量寄存器（htimedelta 和 htimedeltah）

20.1 HS 模式：htimedelta 和 htimedeltah

htimedelta CSR 是一个读/写寄存器，它包含 time CSR 值与 VS 模式或 VU 模式返回值之间的增量。也就是说，在 VS 或 VU 模式下读取 time CSR 将返回 htimedelta 的内容与时间的实际值之和。



仅对于 HSXLEN=32，htimedelta 保存增量的低 32 位，htimedeltah 保存增量的高 32 位。

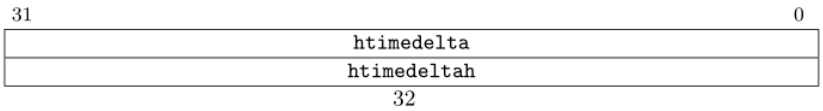


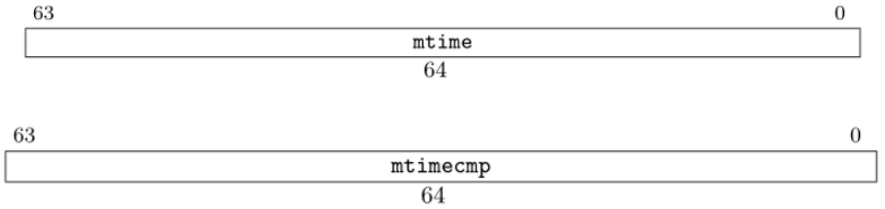
Figure 8.16: Hypervisor time delta registers, HSXLEN=32.

21. 计时寄存器（timer 和 timercmp）

21.1 M 模式：mtimer 和 mtimercmp

mtime 必须以恒定频率运行，并且平台必须提供用于确定 mtime 的时基的机制时间寄存器在所有 RV32 和 RV64 系统上都具有 64 位精度。

平台提供 64 位内存映射的 machine 模式定时器比较寄存器（mtimecmp），当定时器寄存器包含大于或等于 mtimecmp 寄存器中的值时，会导致定时器中断发布。在通过写入 mtimecmp 寄存器将其清除之前，中断将一直保持在 POST 状态。只有在启用中断并且在寄存器中设置了 MTIE 位时，才会执行中断。



22. 地址转换和保护寄存器（atp）

22.1 S 模式：satp

satp 寄存器是一个 SXLEN 位读/写寄存器，格式如图 4.11 所示（SXLEN=32）和图 4.12 所示（SXLEN=64），用于控制 supervisor 下的地址转换和保护。

该寄存器保存根页表的 PPN（物理页号）：它的 supervisor 物理地址除以 4 KiB;

ASID（地址空间标识符）：在每个地址空间的基础上促进地址转换 Fences。

MODE 字段：选择当前地址转换方案。

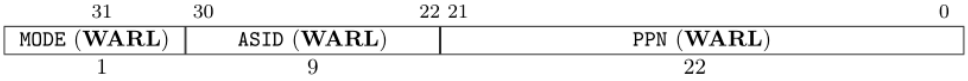


Figure 4.14: Supervisor address translation and protection register **satp** when SXLEN=32.

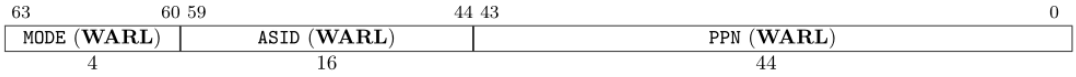


Figure 4.15: Supervisor address translation and protection register **satp** when SXLEN=64, for MODE values Bare, Sv39, Sv48, and Sv57.

表 4.4 显示了当 SXLEN=32 和 SXLEN=64 时 MODE 字段的编码。当 MODE=Bare 时，supervisor 虚拟地址等于 supervisor 物理地址。为了选择 MODE=Bare，软件必须将 0 写入 satp 的剩余字段。试图在剩余字段中选择具有非零模式的“MODE=Bare”对剩余字段所采用的值没有特定影响，对地址转换和保护行为也没有特定影响。

当 SXLEN=64 时，定义了三种分页虚拟内存方案：Sv39、Sv48、 Sv57 和 Sv64。其余的模式设置保留供将来使用，并可能定义对其他字段的不同解释。

实现不需要支持所有的 MODE 设置，如果用不支持的 MODE 写的 satp，整个写没有效果；satp 中没有字段被修改。

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.

ASID 位数是不指定的，可以为零。可以通过向 ASID 字段中的每个比特位置写入 1 来确定实现的 ASID 比特的数量（termed ASIDLEN），然后重新读 satp 中的值，查看 ASID 字段中的哪些位位置是 1。

22.1 HS 模式：hgatp

hgatp 寄存器是一个 HSXLEN 位读/写寄存器，格式如图 8.19 所示（HSXLEN=32）和图 8.20（HSXLEN=64），控制 G 级地址转换和保护，这是 guest 虚拟地址两阶段转换的第二阶段。

与 satp CSR 类似，该寄存器保存 guest 物理根页表的物理页码（PPN）；

VMID（虚拟机标识符），它便于在每个虚拟机的基础上进行地址转换；

MODE 模式字段，它选择 guest 物理地址的地址转换方案。当 mstatus.TVM=1 时，在 HS 模式下执行时尝试读取或写入 hgatp 将引发非法指令异常。

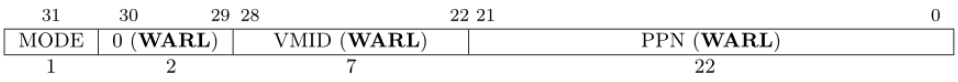


Figure 8.19: Hypervisor guest address translation and protection register `hgatp` when HSXLEN=32.

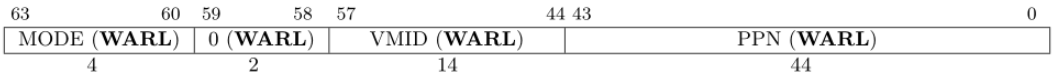


Figure 8.20: Hypervisor guest address translation and protection register `hgatp` when HSXLEN=64, for MODE values Bare, Sv39x4, Sv48x4, and Sv57x4.

表 8.4 显示了当 HSXLEN=32 和 HSXLEN=64 时模式字段的编码。当 MODE=Bare 时，guest 物理地址等于 hypervisor 物理地址。在这种情况下，hgatp 中的其余字段必须设置为零。

当 HSXLEN=64 时的剩余模式设置被保留以供将来使用，并且可以定义对 hgatp 中的其他字段的不同解释。

HSXLEN=32		
Value	Name	Description
0	Bare	No translation or protection.
1	Sv32x4	Page-based 34-bit virtual addressing (2-bit extension of Sv32).

HSXLEN=64		
Value	Name	Description
0	Bare	No translation or protection.
1–7	—	*Reserved*
8	Sv39x4	Page-based 41-bit virtual addressing (2-bit extension of Sv39).
9	Sv48x4	Page-based 50-bit virtual addressing (2-bit extension of Sv48).
10	Sv57x4	Page-based 59-bit virtual addressing (2-bit extension of Sv57).
11–15	—	*Reserved*

Table 8.4: Encoding of `hgatp` MODE field.

22.3 VS 模式：vsatp

vsatp 寄存器是一个 VSXLEN 位读/写寄存器，是 VS 模式下 supervisor satp 寄存器的版本，格式如图 8.32 所示（VSXLEN=32）和图 8.33（VSXLEN=64）。当 V=1 时，vsatp 将替代通常的 satp，因此通常读取或修改 satp 的指令实际上访问 vsatp。vsatp 控制 VS 阶段地址转换，即 guest 虚拟地址的两阶段转换的第一阶段（参见第 8.5 节）

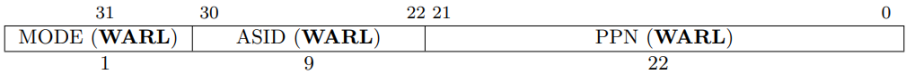


Figure 8.32: Virtual supervisor address translation and protection register `vsatp` when VSXLEN=32.

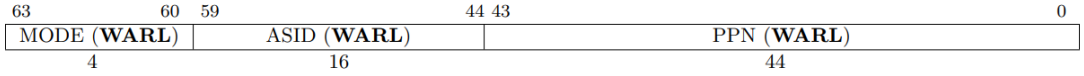


Figure 8.33: Virtual supervisor address translation and protection register `vsatp` when VSXLEN=64, for MODE values Bare, Sv39, Sv48, and Sv57.

当有效特权模式为 VS 模式或 VU 模式时，出于地址转换算法的目的，认为该寄存器是活动的。当执行 virtual-machine Load/Store 指令（HLV、HLVX 或 HSV）时，还短暂地认为该寄存器是活动的，因此当执行这样的指令时，可以推测性地为任何 guest 虚拟地址执行地址转换算法。

当 V=0 时，不会忽略具有不支持的模式值的对 vsatp 的写入，因为它是为了 satp。取而代之的是，vsatp 的字段以正常的方式排列。

当 V=0 时，除非使用 virtual-machine Load/Store（HLV、HLVX 或 HSV）或状态寄存器中的 MPRV 功能来执行加载或存储（假设 V=1），否则 vsatp 不会直接影响机器的行为。

23. 陷阱指令寄存器（tinst）

23.1 HS 模式：htinst

htinst 寄存器是一个 HSXLEN 位读/写寄存器，格式如图 8.18 所示。当陷阱进入 HS 模式

时，htinst 会写入一个值，如果不是零，则该值提供有关捕获的指令的信息，以帮助软件处理陷阱。在 htinst 上可以写入的值在第 8.6.3 节中有记录。

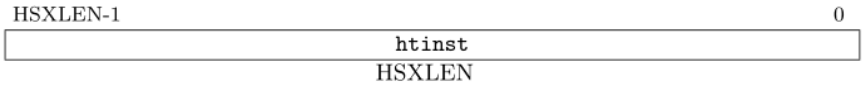


Figure 8.18: Hypervisor trap instruction register (**htinst**).

htinst 是一个 WARL 寄存器，该寄存器只需要能够保存实现可以在陷阱上自动写入的值。

23.2 H 扩展 M 模式：mtinst

mtinst 寄存器是一个 MXLEN 位读/写寄存器，格式如图 8.39 所示。当陷阱进入 M 模式时，mtinst 被写入一个值，如果非零，该值提供关于被陷阱捕获的指令的信息，以帮助软件处理陷阱。第 8.6.3 节记录了可能写入陷阱的值。

Figure 8.39: Machine trap instruction register (**mtinst**).

mtinst 是一个 WARL 寄存器，只需要能够保存实现可能在陷阱上自动写入的值。

指令

1． M 模式指令

1.1 环境调用和断点--ECALL 指令和 BREAK 指令

31 20	19 15	14 12	11 7	6 0
funct12	rs1	funct3	rd	opcode
12	5	3	5	7
ECALL	0	PRIV	0	SYSTEM
EBREAK	0	PRIV	0	SYSTEM

ECALL 指令用于向支持执行环境发出请求。当在 U 模式、S 模式或 M 模式下执行时，它分别生成 U 模式的环境调用异常、S 模式的环境调用异常或 M 模式的环境调用异常，并且不执行其他操作。

调试器使用 EBREAK 指令将控制传回调试环境。它生成断点异常，不执行任何其他操作。

1.2 陷阱返回指令-- MRET、SRET 和 URET 指令

从陷阱返回的指令在 PRIV 次要操作码下编码。

31 20	19 15	14 12	11 7	6 0
funct12	rs1	funct3	rd	opcode
12	5	3	5	7
MRET/SRET/URET	0	PRIV	0	SYSTEM

要在处理陷阱后返回，每个特权级别都有单独的陷阱返回指令：MRET、SRET 和 URET。MRET 是始终提供的。如果支持 Supervisor 模式，则必须提供 SRET，否则应引发非法指令异常。

在 mstatus 中，当 TSR=1 时，SRET 还应引发非法指令异常。仅当支持用户模式陷阱时才提供 URET，否则应引发非法指令。xRET 指令可以在特权模式或更高特权模式下执行，其中执行低特权 xRET 指令将弹出相关的低特权中断使能和特权模式堆栈。除了按照 3.1.6.1 节所述操作特权堆栈之外，xRET 还将 pc 设置为存储在 xepc 寄存器中的值。

如果支持 A 扩展，则允许 xRET 指令清除任何未完成的 LR 地址保留，但不需要这样做。如果需要，陷阱处理程序应在执行 xRET 之前显式清除保留（例如，使用虚拟 SC）。

MRET 或 SRET 指令分别用于在 M 模式或 S 模式下从陷阱返回。在执行一个 xRET 指令时，假设 xPP 为值 y，xIE 设置为 xPIE；特权模式改为 y；xPIE 设置为 1；xPP 设置为最低特权支持模式（如果执行了 U 模式，则设置为 U，否则设置为 M）。如果 xPP 不等于 M，则 xRET 还设置 MPRV=0。

1.3 等待中断--WFI 指令

等待中断指令（WFI）为实现提供了一个提示，即当前 Hart 可以暂停，直到中断可能需

要服务。WFI 指令的执行还可以用于通知硬件平台适当的中断应该优先路由到该 Hart。WFI 可在所有特权模式下使用，也可选择在 U 模式下使用。在 mstatus 中，当 TW=1 时，此指令可能会引发非法指令异常。

31 20	19 15	14 12	11 7	6 0
funct12	rs1	funct3	rd	opcode
12	5	3	5	7
WFI	0	PRIV	0	SYSTEM

如果在 Hart 停止时出现启用的中断或稍后出现中断，则中断异常将发生在以下指令上，即在陷阱处理程序中恢复执行，并且 mepc = pc + 4。

WFI 指令的目的是为实现提供提示，因此合法实现只是将 WFI 实现为 NOP。

当中断被禁用时，也可以执行 WFI 指令。WFI 的操作必须不受全局中断位 mstatus（MIE/SIE/UIE）和中断委托寄存器（mideleg / sideleg）的影响（即，如果本地启用的中断变得挂起，则 Hart 必须恢复，即使它已被委派到较低特权模式），但应遵守单独的中断启用（例如，MTIE）（即，如果中断挂起但不是单独启用，则实现应避免恢复 Hart）。WFI 还需要恢复在任何特权级别挂起的本地启用中断的执行，而不考虑在每个特权级别启用的全局中断。

如果导致 Hart 恢复执行的事件没有引起中断，则将在 pc+4 执行恢复，并且软件必须确定要采取什么操作，包括在没有可操作事件的情况下循环回来以重复执行 WFI。

1.4 自定义系统说明

图 3.30 所示的系统主操作码的子空间被指定为自定义使用。建议这些指令使用位[29:28]来指定所需的最低权限模式，其他系统指令也是如此。

31 26	25 15	14 12	11 7	6 0	
funct6	*custom*	funct3	*custom*	opcode	Recommended Purpose
6	11	3	5	7	
100011	*custom*	0	*custom*	SYSTEM	Unprivileged or User-Level
110011	*custom*	0	*custom*	SYSTEM	Unprivileged or User-Level
100111	*custom*	0	*custom*	SYSTEM	Supervisor-Level
110111	*custom*	0	*custom*	SYSTEM	Supervisor-Level
101011	*custom*	0	*custom*	SYSTEM	Hypervisor-Level
111011	*custom*	0	*custom*	SYSTEM	Hypervisor-Level
101111	*custom*	0	*custom*	SYSTEM	Machine-Level
111111	*custom*	0	*custom*	SYSTEM	Machine-Level

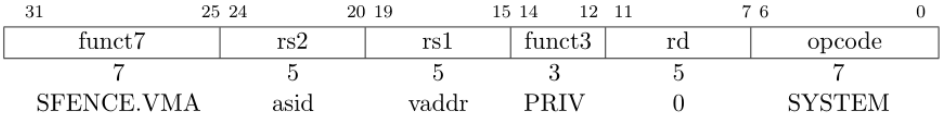
Figure 3.30: SYSTEM instruction encodings designated for custom use.

2． S 模式指令

2.1 Supervisor 内存管理 Fence 指令--SFENCE.VMA

Supervisor 内存管理 Fence 指令 SFENCE.VMA 用于将内存中内存管理数据结构的更新与当前执行同步。执行指令会导致对这些数据结构的隐式读取和写入；然而，相对于显式加

载和存储，这些隐式引用通常是没有排序的。执行 SFENCE.VMA 指令可保证任何对当前 RISC-V Hart 可见的先前存储在该 Hart 中的后续指令对内存管理数据结构的某些隐式引用之前排序。SFENCE.VMA 排序的特定操作集由 rs1 和 rs2 确定，如下所述。SFENCE.VMA 还用于使地址转换缓存中与 Hart 关联的条目无效（参见第 4.3.2 节）。第 3.1.6.5 节和第 3.7.2 节介绍了本指令行为的更多细节。



SFENCE.VMA 只命令本地 Hart 对内存管理数据结构的隐式引用。

对于仅针对单个地址映射（即，onepage 或 superpage）修改了转换数据结构的常见情况，rs1 可以在该映射中指定虚拟地址，以仅针对该映射实现转换 Fence。此外，对于仅针对单个地址空间标识符修改翻译数据结构的常见情况，rs2 可以指定地址空间。SFENCE.VMA 的行为依赖于如下的 rs1 和 rs2：

（1）如果 rs1=x0 并且 rs2=x0，对于所有地址空间，Fence 对任何级别的页表的所有读和写进行排序。Fence 还使所有地址空间的所有地址转换高速缓存条目无效。

（2）如果 rs1=x0 并且 rs2 不等于 x0，Fence 命令对任何级别的页表进行的所有读和写，但只对由整数寄存器 rs2 标识的地址空间。对全局映射（参见第 4.3.1 节）的访问是没有排序的。Fence 还使所有与整数寄存器 rs2 标识的地址空间匹配的地址转换高速缓存条目无效，除了包含全局映射的条目。

（3）如果 rs1 等于 0 并且 rs2=x0，对于所有地址空间，Fence 命令只读取和写入对应于虚拟地址 rs1 的 leaf 页表条目。对于所有地址空间，Fence 还使包含对应于 rs1 中虚拟地址的 leaf 页表条目的所有地址转换高速缓存条目无效。

（4）如果 rs1 不等于 x0 并且 rs2 不等于 x0，对于由整数寄存器 rs2 标识的地址空间，Fence 命令只读取和写入对应于虚拟地址 rs1 的 leaf 页表条目。对全局映射的访问没有排序。Fence 还使除包含全局映射的条目之外的所有地址转换高速缓存条目无效，所述条目包含对应于 rs1 中虚拟地址的 leaf 页表条目，并且与由整数寄存器 rs2 标识的地址空间相匹配。

如果 rs1 中保存的值不是有效的虚拟地址，则 SFENCE.VMA 指令无效。在这种情况下不会引发任何异常。

当 rs2 不等于 x0，rs2 中 SXLEN-1：ASIDMAX 位保存的值保留供将来使用，应由软件置零，并被当前实现忽略。此外，如果 ASIDLEN<ASIDMAX，实现应忽略保持在 rs2 中的值的 ASIDMAX-1：ASIDLEN 位。

对内存管理数据结构的隐式读取可能返回自包含该地址的最新 SFENCE.VMA 以来任何时间有效的地址的任何转换。SFENCE.VMA 暗示的排序不会以与标准 RVWMO 排序规则完全交互的方式将对内存管理数据结构的隐式读取和写入放入全局内存顺序。特别地，即使 SFENCE.VMA 在随后的隐式访问之前对先前的显式访问进行排序，并且这些隐式访问在其相关联的显式访问之前排序，SFENCE.VMA 也不一定在全局存储器顺序中将先前的显式访问置于随后的显式访问之前。这些隐式加载也不需要以其他方式遵守关于先前加载或存储到相同地址的正常程序顺序语义。

3． H 扩展指令

3.1 Hypervisor Virtual-Machine 的 Load/Store 指令

31 25	24 20	19 15	14 12	11 7	6 0
funct7	rs2	rs1	funct3	rd	opcode
7	5	5	3	5	7
HLV.*width*	[U]	addr	PRIVM	dest	SYSTEM
HLVX.HU/WU	HLVX	addr	PRIVM	dest	SYSTEM
HSV.*width*	src	addr	PRIVM	0	SYSTEM

Hypervisor 虚拟机加载和存储指令仅在 M 模式或 HS 模式下有效，或在 hstatus.HU=1 的 U 模式下有效。每条指令执行显式内存访问，就好像 V=1 一样；即，地址转换和保护以及字符顺序，适用于 VS 模式或 VU 模式下的内存访问。Hstatus 的 SPVP 字段控制访问的权限级别。当 SPVP=0 时，显式存储器访问在 VU 模式下进行，当 SPVP=1 时，在 VS 模式下进行显式存储器访问。当 V=1 时，通常应用两阶段地址转换，并且 HS 级别 sstatus.SUM 被忽略。HS 级别 sstatus.MXR 使只执行页面对于地址转换的两个阶段（VS-阶段和 G-阶段）都是可读的，而 HS 级别 vsstatus。MXR 只影响第一个转换阶段（VS-Stage）。

对于每个 RV32I 或 RV64I 加载指令 LB、LBU、LH、LHU、LW、LWU 和 LD，存在对应的虚拟机加载指令：HLV.B、HLV.BU、HLV.H、HLV.HU、HLV.W、HLV.WU 和 HLV.D。对于每个 RV32I 或 RV64I 存储指令 SB、SH、SW 和 SD，存在对应的虚拟机存储指令：HSV.B、HSV.H、HSV.H、HSV.D。当然，HSV.D 对 RV32 无效。

指令 HLVX.HU 和 HLVX.WU 与 HLV.HU 和 HLV.WU 相同，只是在地址转换过程中执行权限取代了读取权限。也就是说，被读取的内存在地址转换的两个阶段都必须是可执行的，但不需要读取权限。对于地址转换产生的主控引擎物理地址，主控引擎物理内存属性必须同时授予 EXECUTE 和 READ 权限。（Supervisor 物理内存属性是针对 Supervisor 级别的物理内存保护第 3.7 节修改的机器物理内存属性。）

在执行虚拟机 Load/Store 指令（HLV、HLVX 或 HSV）时，对于地址转换算法而言，hgatp 和 vsatp 寄存器被认为是活动的。

HLVX.WU 对 RV32 有效，即使 LWU 和 HLV.WU 无效。（对于 RV32，HLVX.WU 可以被视为 HLV.W 的变体，因为符号扩展与 32 位值无关。）

当 V=1 时，尝试执行 virtual-machine Load/Store 指令（HLV、HLVX 或 HSV）会导致伪指令陷阱。尝试从 U 模式执行其中一条相同的指令（当 hstatus.HU=0）会导致非法指令陷阱。

3.2 Hypervisor 内存管理 Fence 指令

31 25	24 20	19 15	14 12	11 7	6 0
funct7	rs2	rs1	funct3	rd	opcode
7	5	5	3	5	7
HFENCE.VVMA	asid	vaddr	PRIV	0	SYSTEM
HFENCE.GVMA	vmid	gaddr	PRIV	0	SYSTEM

Hypervisor 内存管理 Fence 指令 HFENCE.VVMA 和 HFENCE.GVMA 执行类似于 SFENCE.VMA（第 4.2.1 节）的功能，除了应用于由 vsatp CSR 控制的 VS 级内存管理数据结

构(HFENCE.VVMA)或由 hgatp CSR 控制的 guest 物理内存管理数据结构(HFENCE.GVMA)。指令 SFENCE.VMA 仅适用于由当前 satp 控制的内存管理数据结构（当 V=0 时为 HS 级别 satp，当 V=1 时为 vsatp)。

HFENCE.VVMA 仅在 M 模式或 HS 模式下有效。其效果与临时进入 VS 模式并执行 SFENCE.VMA 基本相同。执行 HFENCE.VVMA 保证了在 Hart 对 VS 级存储器管理数据结构所有后续隐式读取之前对当前 Hart 可见的任何先前存储进行排序，当这些隐式读取是针对:

（1）在 HFENCE.VVMA 之后，以及

（2）执行当 hgatp.VMID 的设置与执行 HFENCE.VVMA 时的设置相同。

当 hgatp.VMID 与执行 HFENCE.VVMA 时不同时，不需要对隐式读取进行排序。如果操作数 rs1 不等于 x0，则指定单个 guest 虚拟地址；如果操作数 rs2 不等于 x0，则指定单个 guest 地址空间标识符（ASID)。

当 rs2 不等于 X0 时，保留在 rs2 中的值的位[XLEN-1：ASIDMAX]保留供将来标准使用。在它们的使用由标准扩展定义之前，它们应该由软件归零，并被当前的实现忽略。此外，如果 ASIDLEN<ASIDMAX，则实现将忽略保存在 rs2 中的值的位[ASIDMAX-1：ASIDLEN]。

mstatus.TVM 和 hstatus.VTVM 都不能导致 HFENCE.VVMA 被陷阱捕获。

HFENCE.GVMA 仅在 mstatus.TVM=0 时在 HS 模式下有效，或在 M 模式下有效（与 mstatus.TVM 无关)。执行 HFENCE.GVMA 指令可保证当前 Hart 已经可见的任何先前存储在该 HART 针对 HFENCE.GVMA 之后的指令执行的 guest 物理内存管理数据结构的所有后续隐式读取之前排序。如果操作数 rs1 不等于 x0，则它指定单个 guest 物理地址，右移 2 位；如果操作数 rs2 不等于 x0，则它指定单个虚拟机标识符（VMID)。

当 rs2 不等于 x0 时，保留在 rs2 中的值的位[XLEN-1：VMIDMAX]被保留以供将来标准使用。在它们的使用由标准扩展定义之前，它们应该由软件归零，并被当前的实现忽略。此外，如果 VMIDLEN<VMIDMAX，则该实现将忽略保存在 rs2 中的值的位[VMIDMAX-1：VMIDLEN]。

如果针对给定的 VMID 更改了 hgatp.MODE，则必须执行带有 rs1=x0（并且 rs2 设置为 x0 或 VMID）的 HFENCE.GVMA，以便订购具有模式更改的后续 guest 翻译--即使旧模式或新模式是空的。

当 V=1 时尝试执行 HFENCE.VVMA 或 HFENCE.GVMA 会导致伪指令陷阱，而在 U 模式下执行相同操作会导致非法指令陷阱。当 mstatus.TVM=1 时，试图在 HS 模式下执行 HFENCE.GVMA 也会导致非法指令陷阱。

其他