RISC-V Privilege System

1. Introduction

1.1. Priveledge Architecture

RISC-V supports 3 executions modes (privilege levels). At any time core is one of three modes:

M machine mode:
- used at startup, typically not accessed later
- handles traps

S Supervisor mode:
- kernel runs in S-mode
- handles virtual memory
- handles traps delegated from M-mode

U user mode:
- user space

Current mode is not stored in any register, it's implicit.

1.2. Control and Status registers.

RISC-V has separate 12-bit address space for up to 4096 CSRs. Access to CSR is limited to specific modes (S-mode register can be accessed in M and S modes). Access from wrong mode causes **Illegal Instruction Exception**.

Some bits of the address specify register mode and if is read-only.

2. Trap Processing

Exceptions Problem caused by instruction

Interrupts Other sources

Trap handler runs in S-mode or M-mode. Handler can only service interrupts from lower modes.

2.1. Phases of trap processing

Hardware Parse
- trap happens
- previous state is saved (**sstatus**: SPP, SPIE)
- interrupts are disabled (**sstatus**: SIE)
- **scasuse** set to code number
- **sepc** set to intrusion address
- **PC** set to **stvec**

Software Phase
- resisters saved (using **sscratch**)
- trap is processed based on **scause**
- registers are restored
- **sret** instruction exectuted
  - SIE ⟵ SPIE
  - mode ⟵ SPP
  - **PC ⟵ sepc**

M-mode trap handler uses equivalent M-mode fields and registers.

2.2. Delegation

By default all traps from S and U modes are handled by M-mode trap handler.
Some traps may be delegated to S-mode trap handler.
Delegation is determined by content of **medeleg** (exceptions) and **mideleg** (interrupts) registers. If **medeleg** contains 1 on position of exception code the trap is delegated to S-mode.

Al M-mode traps are serviced by M-mode handler.

2.3. Interrupts

mip interrupts pending
mie interrupts enabled

Bit for each type of interrupts (same as in **mideleg**):

- software interrupts
- timer interrupts
- external interrupts
- local counter overflow

Supervisor interrupt bits are mirrored into **sip** and **sie**.

Global interrupt masking with **mstatus:MIE** and **sstatus:SIE**:

- U-Mode
  - interrupts always enabled
- S-mode
  - S-mode interrupts may be disabled with SIE
  - M-mode interrupts always served (regardless of MIE)
- M-mode
  - S-mode interrupts disabled; remain pending
  - M-mode interrupts may be disabled with MIE

3. Control and Status Registers

3.1. Status registers

Defines trap handling and configuration. Status is stored in **mstatus** M-mode register. Register **sstatus** mirrors **mstatus** and provides access to some fields of mstatus for S-mode operations.

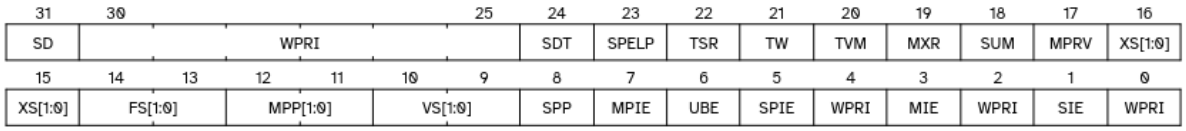


Figure 1: **mstatus** register

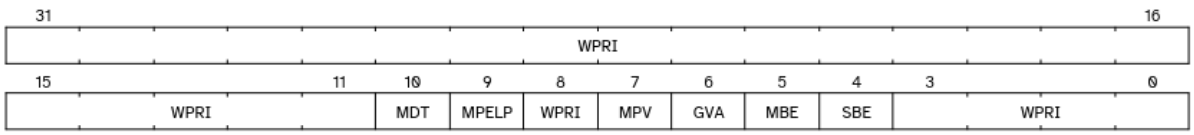


Figure 2: **mstatush** register

3.1.1. MPP, SPP, MPIE, SPIE, MIE and SIE

Machine mode variants are only available in **mstatus**.
MIE and SIE global interrupt masking
MPIE and SPIE previous interrupt enable

MPP and SPP previous privilege level (MPP is 2 bit wide).

3.1.2. MBE, SBE and UBE

Machine-, Supervisor-, User- mode big endian. Affects loads and stores endianness is specific modes, when set the order is big endian.

By default riscv is little endian and instruction fetches are always little endian (detection of compressed instructions).

3.1.3. MXR, SUM and MPVR

MXR make executable readable:

- when set, loads can be made on pages marked "X" ("R" doesn't need to be set)

SUM supervisor user memory:

- U-mode pages can only be accessed in U-mode
- S-mode pages can only be accessed in S-mode
- when set, supervisor can access user pages to process system calls

MPVR modify privilege level **(only in mstatus)**:

- loads and stores are executed with privileges of level on which exception occurred (saved in MPP)
- mret and sret resets MPRV to 0

3.1.4. SD, XS, FS and VS

On context switch kernel needs to:

- save previous process statea
- load state on next state

State of a process is:

- general purpose registers and PC
- floating point registers **..FS**
- vector registers **..VS**
- other implementation-dependent state **..XS**

Bits XS, FS and VS mark what needs to be saved and restored on context switches.

00 not used

- kernel doesn't save/restore them
- user code makes syscall to "turn FP on"

01 initialized and ready to be used

- will be used but have not yet been used
- **start:** kernel initializes to zero
- **end:** kenel does nothing

10 previously modified, but not this timeline

- **start:** kernel restores previous values
- **end:** kenel does nothing

11 modified ("dirty")

- **start:** kernel restores previous values
- **end:** kenel saves them

Hardware watches for changes to FP registers and on any modification changes status to 11 (modified). On 00 (not used) exception is raised.

SD bit marks that any of FS, VS or XS is dirty. It can be check as a sign of **mstatus** register.

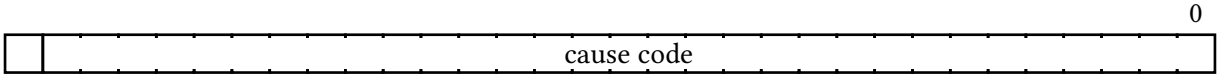
$$\text{SD} = (\text{FS is "dirty"}) \text{ or } (\text{VS is "dirty"}) \text{ or } (\text{XS is "dirty"})$$

3.1.5. TSR, TW, TVM

Concern hypervisor support, available only in **mstatus**.

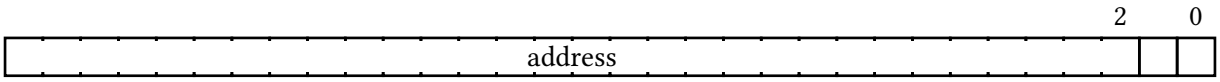
3.2. Trap CSRs

3.3. mcasue and scause



Hardware stores trap cause code to mcause or scause when handling a trap. Traps with 1 on msb are interrupts (negative), other are exceptions.

3.4. mtvec and stvec



Contain address of Trap Handler. Address must be word aligned, end with 00. Two least significant bits are ignored on jump, they indicate type of trap handler.

00 Type 1:
All traps go to one trap handler at *address*.

01 Type 2:
All exceptions go to trap handler ant *address*.
Each interrupt goes to unique trap handler at *address* + (4 · *CauseCode*)

3.5. Counters

All counters are 64 bit. On 32 bit machine "h" register contains upper half.

3.5.1. Machine mode Counters

mcycle – count of clock cycles since reset
~~**mtime**~~ – not a CSR
minstret – count of completed instructions
mhpmcounter3 – hardware performance monitor counters
...
mhpmcounter31

Registers **mhpmevent3** through **mhpmevent31** determine what hpm counters are counting.

3.5.2. minhibiten

When n-th bit is set, the n-th counter is paused.

3.5.3. Supervisor mode Counters

Those registers are mirrors of there M-mode equivalents. Are strictly **read-only**.
cycle – count of clock cycles since reset
time – is a CSR, mirrors RTC time
instret – count of completed instructions
hpmcounter3 – hardware performance monitor counters
...
hpmcounter31

3.5.4. mcounteren

When n-th bit is set, the n-th S-mode counter is visible in S-mode.

3.5.5. scounteren

When n-th bit is set, the n-th S-mode counter is visible in U-mode.

3.5.6. Timer Interrupts

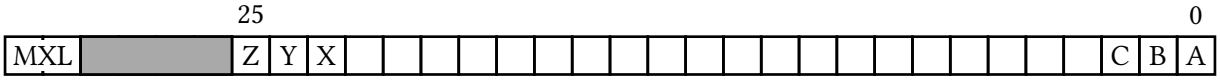
mtimecmp is not a CSR, determines when machine timer interrupt should be fired.
When **mtimecmp** ≤ **mtime** MTIP of mip is set.

Machine mode simulates timer interrupts for S-mode by setting STIP of **mip**. STIP is mirrors into read-only **sip**.

Sstc extension provides **stimecmp** CSR. When it's implemented th STIP bit of **sip** is set by **stimecmp** ≤ **mtime**. Both **sip**:STIP and **mip**:STIP are read-only.

3.6. Miscellanies CSRs

3.6.1. misa



Bits 0 to 25 correspond to main RISCV extensions.
Field MXL indicates word length:

- 00: no info
- 01: 32 bits
- 10: 64 bits
- 11: 128 bits

3.6.2. mhartid

Contains hardware thread (HART) id. One HART must have id 0.

3.6.3. mvendorid, marchid and mimplid