Double Trap Extensions

Version v1.0, 2024-08-23: Ratified

Table of Contents

Preamble

This document is in the Ratified state

No changes are allowed. Any desired or needed changes can be the subject of a follow-on new extension. Ratified extensions are never revised

Copyright and license information

Contributors

This RISC-V specification has been contributed to directly or indirectly by:

Allen Baum, Andrew Waterman, Clément Léger, Deepak Gupta, Earl Killian, Greg Favor, John Hauser, Joshua Scheid, Paul Donahue, Tim Newsome, and Ved Shanbhogue

Chapter 1. Double Trap

A double trap typically arises during a sensitive phase in trap handling operations — when an exception or interrupt occurs while the trap handler (the component responsible for managing these events) is in a non-reentrant state. This non-reentrancy usually occurs in the early phase of trap handling, wherein the trap handler has not yet preserved the necessary state to handle and resume from the trap. The occurrence of a trap during this phase can lead to an overwrite of critical state information, resulting in the loss of data needed to recover from the initial trap. The trap that caused this critical error condition is henceforth called the *unexpected trap*. Trap handlers are designed to neither enable interrupts nor cause exceptions during this phase of handling. However, managing Hardware-Error exceptions [1], which may occur unpredictably, presents significant challenges in trap handler implementation due to the potential risk of a double trap.

The ISA Double Trap Extensions are devised to tackle situations where conventional fault handling mechanisms fall short. Although a double trap generally is an irrecoverable condition and signals a critical error, these extensions introduce strategies to manage such errors across privilege levels.

The following ISA extensions are defined to enhance fault handling capabilities:

- Ssdbltrp: This extension is designed to address double trap at privilege modes lower than M. It enables HS-mode to invoke a critical error handler in a virtual machine on a double trap in VS-mode. It also allows M-mode to invoke a critical error handler in the OS/Hypervisor on a double trap in S/HS-mode. The Ssdbltrp extension is specified in Section 1.1.
- Smdbltrp: This extension addresses a double trap in M-mode. When the Smrnmi extension [2] is implemented, it enables invocation of the RNMI handler on a double trap in M-mode to handle the critical error. If the Smrnmi extension is not implemented or if a double trap occurs during the RNMI handler's execution, this extension helps transition the hart to a critical error state and enables signaling the critical error to the platform. To improve error diagnosis and resolution, this extension supports debugging harts in a critical error state. The Smdbltrp extension is specified in Section 1.2.

1.1. Ssdbltrp Operational Details

The Ssdbltrp adds an S-mode-disable-trap (`SDT`) field (bit 24) to the `sstatus` CSR and a double-trap-enable (`DTE`) field (bit 59) to the `menvcfg` CSR. If the H-extension is implemented, Ssdbltrp also adds a `SDT` field (bit 24) to the `vsstatus` CSR and a `DTE` field (bit 59) to the `henvcfg` CSR. The Ssdbltrap extension requires the implementation of the `mtval2` CSR. An Exception Code (value = 16) called the double trap is introduced. The bit 16 of the `medeleg` and the `hedeleg` CSRs are read-only zero as the double trap is not delegatable.

When `menvcfg.DTE` is zero, the implementation behaves as though Ssdbltrp is not implemented. When Ssdbltrp is not implemented `sstatus.SDT`, `vsstatus.SDT`, and `henvcfg.DTE` bits are read-only zero. When `henvcfg.DTE` is zero, the implementation behaves as though Ssdbltrp is not implemented for VS-mode and the `vsstatus.SDT` bit is read-only zero.

When the `SDT` bit is set to 1 by an explicit CSR write, the `SIE` (Supervisor Interrupt Enable) bit is cleared to 0. This clearing occurs regardless of the value written, if any, to the `SIE` bit by the same write. The `SIE` bit can only be set to 1 by an explicit CSR write if the `SDT` bit is being set to 0 by the same write or is already 0.

When a trap is to be taken into VS- or S-mode, if the `SDT` bit is currently 0, it is then set to 1, and the trap is delivered as expected. However, if `SDT` is already set to 1, then this is an *unexpected trap*. In the event of an *unexpected trap*, a double-trap exception trap is delivered into M-mode. To deliver this trap, the hart writes registers, except `mcause` and `mtval2`, with the same information that the *unexpected trap* would have written if it was taken into M-mode. The `mtval2` register is then set to what would be otherwise written into the `mcause` register by the *unexpected trap*. The `mcause` register is set to 16, the double-trap exception code.

After a trap handler has saved the state, such as `scause`, `sepc`, *and* `stval`, *needed for resuming from the trap and is reentrant, it should clear the* `SDT` *bit. Resetting the* `SDT` *by an SRET enables the trap handler to detect a double trap that may occur during the tail phase, where it restores critical state to return from a trap.*

The consequence of this specification is that if a critical error condition was caused by a guest page-fault, then the GPA will not be available in `mtval2` *when the double trap is delivered to M-mode. This condition arises if the HS-mode invokes a hypervisor virtual-machine load or store instruction when* `SDT` *is 1 and the instruction raises a guest page-fault. The use of such an instruction in this phase of trap handling is not common. However, not recording the GPA is considered benign because, if required, it can still be obtained — albeit with added effort — through the process of walking the page tables.*

For a double trap that originates in VS-mode, M-mode should redirect the exception to HS-mode by copying the values of M-mode CSRs updated by the trap to HS-mode CSRs and should use an `MRET` *to resume execution at the address in* `stvec`. *Supervisor Software Events (SSE) [3], an extension to the SBI, provide a mechanism for supervisor software to register and service system events emanating from an SBI implementation, such as firmware or a hypervisor. In the event of a double trap, HS-mode and M-mode can utilize the SSE mechanism to invoke a critical-error handler in VS-mode or S/HS-mode, respectively. Additionally, the implementation of an SSE protocol can be considered as an optional measure to aid in the recovery from such critical errors.*

In S-mode, the `SRET` instruction sets `sstatus.SDT` to 0, and if the new privilege mode is VU, it also sets `vsstatus.SDT` to 0. However, in VS-mode, only `vsstatus.SDT` is set to to 0.

The `MRET` instructions sets `sstatus.SDT` to 0, if the new privilege mode is U, VS, or VU. Additionally, if it is VU, then `vsstatus.SDT` is also set to 0.

When a hart resumes from Debug Mode, if the new privilege mode is U, VS, or VU, then `sstatus.SDT` is set to 0. Additionally, if it is VU, then `vsstatus.SDT` is also set to 0.

1.2. Smdbltrp Operational Details

The Smdbltrp extension adds an M-mode-disable-trap (`MDT`) field (bit 42) to the `mstatus` CSR for RV64. For RV32, this field is located in the `mstatush` CSR at bit position 10. Upon reset, the `MDT` field is set to 1. An Exception Code (value = 16) called the double trap is introduced.

When the `MDT` bit is set to 1 by an explicit CSR write, the `MIE` (Machine Interrupt Enable) bit is cleared to 0. For RV64, this clearing occurs regardless of the value written, if any, to the `MIE` bit by the same write. The `MIE` bit can only be set to 1 by an explicit CSR write if the `MDT` bit is already 0 or, for RV64, is being set to 0 by the same write (For RV32, the `MDT` bit is in `mstatush` and the `MIE` bit in `mstatus` register).

When a trap is to be taken into M-mode, if the `MDT` bit is currently 0, it is then set to 1, and the trap is delivered as expected. However, if `MDT` is already set to 1, then this is an *unexpected trap*. When the Smrnmi extension is implemented, a trap caused by an RNMI is not considered an *unexpected trap* irrespective of the state of the `MDT` bit. A trap caused by an RNMI does not set the `MDT` bit. However, a trap that occurs when executing in M-mode with `mnstatus.NMIE` set to 0 is an *unexpected trap*. The *unexpected trap* is handled as follows:

- When the Smrnmi extension is implemented and `mnstatus.NMIE` is 1, the hart traps to the RNMI handler. To deliver this trap, the `mnepc` and `mncause` registers are written with the values that the *unexpected trap* would have written to the `mepc` and `mcause` registers respectively. The privilege mode information fields in the `mnstatus` register are written to indicate M-mode and its `NMIE` field is set to 0.

The consequence of this specification is that on occurrence of double trap, the RNMI handler is not provided with information that a trap reports in the `mtval` *and the* `mtval2` *registers. This information, if needed, can be obtained by the RNMI handler by decoding the instruction at the address in* `mnepc` *and examining its source register contents.*

- When the Smrnmi extension is not implemented, or if the Smrnmi extension is implemented and `mnstatus.NMIE` is 0, the hart enters a critical-error state without updating any architectural state, including the `pc`. This state involves ceasing execution, disabling all interrupts (including NMIs), and asserting a `critical-error` signal to the platform.

The actions performed by the platform when a hart asserts a `critical-error` *signal are platform-specific. The range of possible actions include restarting the affected hart or restarting the entire platform, among others.*

The `MRET` and `SRET` instructions, when executed in M-mode, set the `MDT` bit to 0. If the new privilege mode is U, VS, or VU, then `sstatus.SDT` is also set to 0. Additionally, if it is VU, then `vsstatus.SDT` is also set to 0.

The Smdbltrp extension introduces a read-write critical-error-trigger (`cetrig`) field (bit 19) to the `dcsr` CSR [4]. When `cetrig` is set to 1, a hart in a critical error state enters Debug Mode instead of asserting the critical-error signal to the platform. Upon such entry into Debug Mode, the `cause` field of `dcsr` is set to 7, and its `extcause` field is set to 0, indicating a critical error triggered the Debug Mode entry. This cause has the highest priority among all reasons for entering Debug Mode. Resuming from Debug Mode following an entry from the critical error state returns the hart to the critical error state.

When a hart resumes from Debug Mode, if the new privilege mode is not M, then the `MDT` bit is set to 0. If it is U, VS, or VU, then `sstatus.SDT` is also set to 0. Additionally, if it is VU, then `vsstatus.SDT` is also set to 0.

The `MNRET` instruction sets the `MDT` bit to 0 if the new privilege mode is not M. If it is U, VS, or VU, then `sstatus.SDT` is also set to 0. Additionally, if it is VU, then `vsstatus.SDT` is also set to 0.

The implication of this specification is that resuming from Debug Mode, following an entry due to a critical error will either result in an immediate re-entry into Debug Mode due to the critical error if `cetrig` *is set to 1, or it will cause the critical-error signal to be asserted to the platform. The debugger can resume with* `cetrig` *set to 0 to allow the platform-defined actions on critical-error signal to occur. Other possible actions include initiating a hart or platform reset using the Debug Module reset control.*

Bibliography

[1] “RISC-V Instruction Set Manual, Volume II: Privileged Architecture.” [Online]. Available: github.com/riscv/riscv-isa-manual.

[2] “‘Smrnmi’ Standard Extension for Resumable Non-Maskable Interrupts.” [Online]. Available: github.com/riscv/riscv-isa-manual/blob/main/src/rnmi.adoc.

[3] “SBI: Supervisor Software Events Extension.” [Online]. Available: github.com/riscv-non-isa/riscv-sbi-doc.

[4] “The RISC-V Debug Specification.” [Online]. Available: github.com/riscv/riscv-debug-spec.