DRAFT International Standard

ISO/IEC DIS 26791-2

Information technology — The RISC-V instruction set manual —

Part 2: Volume II: Privileged architecture

ICS: 35.160

ISO/IEC JTC **1**

Secretariat: **ANSI**

Voting begins on:
2026-07-31

Voting terminates on:
2026-10-23

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENTS AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

This document has not been edited by the ISO Central Secretariat.

Reference number
ISO/IEC DIS 26791-2:2026(en)

© ISO/IEC 2026



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2026

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

1 Introduction **1**

1.1 RISC-V Privileged Software Stack Terminology 1

1.2 Privilege Levels 2

1.3 Debug Mode 4

2 Control and Status Registers (CSRs) **5**

2.1 CSR Address Mapping Conventions 5

2.2 CSR Listing 7

2.3 CSR Field Specifications 11

2.4 CSR Width Modulation 12

3 Machine-Level ISA, Version 1.11 **13**

3.1 Machine-Level CSRs 13

3.1.1 Machine ISA Register `misa` 13

3.1.2 Machine Vendor ID Register `mvendorid` 16

3.1.3 Machine Architecture ID Register `marchid` 16

3.1.4 Machine Implementation ID Register `mimpid` 17

3.1.5 Hart ID Register `mhartid` 17

3.1.6 Machine Status Register (`mstatus`) 18

3.1.6.1 Privilege and Global Interrupt-Enable Stack in `mstatus` register . . 18

© ISO/IEC 2026 – All rights reserved

3.1.6.2 Base ISA Control in `mstatus` Register . . . 19
3.1.6.3 Memory Privilege in `mstatus` Register . . . 20
3.1.6.4 Virtualization Support in `mstatus` Register . . . 21
3.1.6.5 Extension Context Status in `mstatus` Register . . . 21
3.1.7 Machine Trap-Vector Base-Address Register (`mtvec`) . . . 24
3.1.8 Machine Trap Delegation Registers (`medeleg` and `mideleg`) . . . 25
3.1.9 Machine Interrupt Registers (`mip` and `mie`) . . . 27
3.1.10 Machine Timer Registers (`mtime` and `mtimecmp`) . . . 29
3.1.11 Hardware Performance Monitor . . . 30
3.1.12 Counter-Enable Registers (`[m|s]counteren`) . . . 31
3.1.13 Machine Counter-Inhibit CSR (`mcountinhibit`) . . . 32
3.1.14 Machine Scratch Register (`mscratch`) . . . 33
3.1.15 Machine Exception Program Counter (`mepc`) . . . 33
3.1.16 Machine Cause Register (`mcause`) . . . 34
3.1.17 Machine Trap Value (`mtval`) Register . . . 36
3.2 Machine-Mode Privileged Instructions . . . 37
3.2.1 Environment Call and Breakpoint . . . 37
3.2.2 Trap-Return Instructions . . . 38
3.2.3 Wait for Interrupt . . . 39
3.3 Reset . . . 40
3.4 Non-Maskable Interrupts . . . 40
3.5 Physical Memory Attributes . . . 41
3.5.1 Main Memory versus I/O versus Empty Regions . . . 42
3.5.2 Supported Access Type PMAs . . . 42
3.5.3 Atomicity PMAs . . . 42
3.5.4 Memory-Ordering PMAs . . . 44
3.5.5 Coherence and Cacheability PMAs . . . 44
3.5.6 Idempotency PMAs . . . 45

© ISO/IEC 2026 – All rights reserved

3.6 Physical Memory Protection 46
3.6.1 Physical Memory Protection CSRs 46
3.6.2 Physical Memory Protection and Paging 50

4 Supervisor-Level ISA, Version 1.11 51

4.1 Supervisor CSRs 51
4.1.1 Supervisor Status Register (`sstatus`) 51
4.1.2 Base ISA Control in `sstatus` Register 52
4.1.3 Memory Privilege in `sstatus` Register 53
4.1.4 Supervisor Trap Vector Base Address Register (`stvec`) 53
4.1.5 Supervisor Interrupt Registers (`sip` and `sie`) 54
4.1.6 Supervisor Timers and Performance Counters 55
4.1.7 Counter-Enable Register (`scounteren`) 56
4.1.8 Supervisor Scratch Register (`sscratch`) 56
4.1.9 Supervisor Exception Program Counter (`sepc`) 56
4.1.10 Supervisor Cause Register (`scause`) 57
4.1.11 Supervisor Trap Value (`stval`) Register 57
4.1.12 Supervisor Address Translation and Protection (`satp`) Register 59
4.2 Supervisor Instructions 61
4.2.1 Supervisor Memory-Management Fence Instruction 61
4.3 Sv32: Page-Based 32-bit Virtual-Memory Systems 63
4.3.1 Addressing and Memory Protection 63
4.3.2 Virtual Address Translation Process 66
4.4 Sv39: Page-Based 39-bit Virtual-Memory System 67
4.4.1 Addressing and Memory Protection 67
4.5 Sv48: Page-Based 48-bit Virtual-Memory System 68
4.5.1 Addressing and Memory Protection 68

5 RISC-V Privileged Instruction Set Listings 71

© ISO/IEC 2026 – All rights reserved

Chapter 1

Introduction

This document describes the RISC-V privileged architecture, which covers all aspects of RISC-V systems beyond the unprivileged ISA, including privileged instructions as well as additional functionality required for running operating systems and attaching external devices.

Commentary on our design decisions is formatted as in this paragraph, and can be skipped if the reader is only interested in the specification itself.

We briefly note that the entire privileged-level design described in this document could be replaced with an entirely different privileged-level design without changing the unprivileged ISA, and possibly without even changing the ABI. In particular, this privileged specification was designed to run existing popular operating systems, and so embodies the conventional level-based protection model. Alternate privileged specifications could embody other more flexible protection-domain models. For simplicity of expression, the text is written as if this was the only possible privileged architecture.

1.1 RISC-V Privileged Software Stack Terminology

This section describes the terminology we use to describe components of the wide range of possible privileged software stacks for RISC-V.

Figure 1.1 shows some of the possible software stacks that can be supported by the RISC-V architecture. The left-hand side shows a simple system that supports only a single application running on an application execution environment (AEE). The application is coded to run with a particular application binary interface (ABI). The ABI includes the supported user-level ISA plus a set of ABI calls to interact with the AEE. The ABI hides details of the AEE from the application to allow greater flexibility in implementing the AEE. The same ABI could be implemented natively on multiple different host OSs, or could be supported by a user-mode emulation environment running on a machine with a different native ISA.

Our graphical convention represents abstract interfaces using black boxes with white text, to separate them from concrete instances of components implementing the interfaces.

© ISO/IEC 2026 – All rights reserved

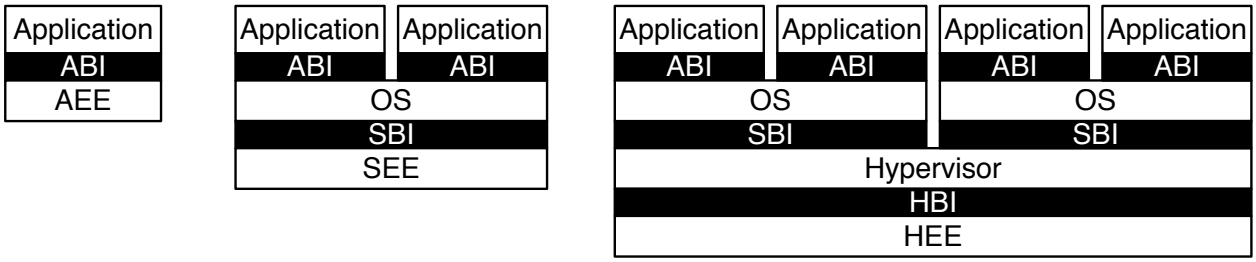


Figure 1.1: Different implementation stacks supporting various forms of privileged execution.

The middle configuration shows a conventional operating system (OS) that can support multiprogrammed execution of multiple applications. Each application communicates over an ABI with the OS, which provides the AEE. Just as applications interface with an AEE via an ABI, RISC-V operating systems interface with a supervisor execution environment (SEE) via a supervisor binary interface (SBI). An SBI comprises the user-level and supervisor-level ISA together with a set of SBI function calls. Using a single SBI across all SEE implementations allows a single OS binary image to run on any SEE. The SEE can be a simple boot loader and BIOS-style IO system in a low-end hardware platform, or a hypervisor-provided virtual machine in a high-end server, or a thin translation layer over a host operating system in an architecture simulation environment.

Most supervisor-level ISA definitions do not separate the SBI from the execution environment and/or the hardware platform, complicating virtualization and bring-up of new hardware platforms.

The rightmost configuration shows a virtual machine monitor configuration where multiple multiprogrammed OSs are supported by a single hypervisor. Each OS communicates via an SBI with the hypervisor, which provides the SEE. The hypervisor communicates with the hypervisor execution environment (HEE) using a hypervisor binary interface (HBI), to isolate the hypervisor from details of the hardware platform.

The ABI, SBI, and HBI are still a work-in-progress, but we are now prioritizing support for Type-2 hypervisors where the SBI is provided recursively by an S-mode OS.

Hardware implementations of the RISC-V ISA will generally require additional features beyond the privileged ISA to support the various execution environments (AEE, SEE, or HEE).

1.2 Privilege Levels

At any time, a RISC-V hardware thread (*hart*) is running at some privilege level encoded as a mode in one or more CSRs (control and status registers). Three RISC-V privilege levels are currently defined as shown in Table 1.1.

Privilege levels are used to provide protection between different components of the software stack, and attempts to perform operations not permitted by the current privilege mode will cause an exception to be raised. These exceptions will normally cause traps into an underlying execution environment.

© ISO/IEC 2026 – All rights reserved