# DRAFT International Standard

# ISO/IEC DIS 26791-1

# Information technology — The RISC-V instruction set manual —

## Part 1: Volume I: Unprivileged ISA

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 26791-1:2026(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 Hardware Platform Terminology . . . . . 2

1.2 RISC-V Software Execution Environments and Harts . . . . . 2

1.3 RISC-V ISA Overview . . . . . 4

1.4 Memory . . . . . 6

1.5 Base Instruction-Length Encoding . . . . . 7

1.6 Exceptions, Traps, and Interrupts . . . . . 8

**2 RV32I Base Integer Instruction Set, Version 2.1** **11**

2.1 Programmers' Model for Base Integer ISA . . . . . 11

2.2 Base Instruction Formats . . . . . 14

2.3 Immediate Encoding Variants . . . . . 15

2.4 Integer Computational Instructions . . . . . 16

2.5 Control Transfer Instructions . . . . . 19

2.6 Load and Store Instructions . . . . . 22

2.7 Memory Ordering Instructions . . . . . 25

2.8 Environment Call and Breakpoints . . . . . 26

2.9 HINT Instructions . . . . . 27

**3 "Zifencei" Instruction-Fetch Fence, Version 2.0** **29**

© ISO/IEC 2026 – All rights reserved

**4 RV64I Base Integer Instruction Set, Version 2.1 31**

4.1 Register State . . . 31
4.2 Integer Computational Instructions . . . 31
4.3 Load and Store Instructions . . . 33
4.4 HINT Instructions . . . 34

**5 "M" Standard Extension for Integer Multiplication and Division, Version 2.0 37**

5.1 Multiplication Operations . . . 37
5.2 Division Operations . . . 38

**6 "A" Standard Extension for Atomic Instructions, Version 2.1 41**

6.1 Specifying Ordering of Atomic Instructions . . . 41
6.2 Load-Reserved/Store-Conditional Instructions . . . 42
6.3 Eventual Success of Store-Conditional Instructions . . . 44
6.4 Atomic Memory Operations . . . 46

**7 "Zicsr", Control and Status Register (CSR) Instructions, Version 2.0 49**

7.1 CSR Instructions . . . 49

**8 Counters 53**

8.1 "Zicntr" Base Counters and Timers, Version 2.0 . . . 53
8.2 "Zihpm" Hardware Performance Counters, Version 2.0 . . . 55

**9 "F" Standard Extension for Single-Precision Floating-Point, Version 2.2 57**

9.1 F Register State . . . 57
9.2 Floating-Point Control and Status Register . . . 59
9.3 NaN Generation and Propagation . . . 60
9.4 Subnormal Arithmetic . . . 61
9.5 Single-Precision Load and Store Instructions . . . 61
9.6 Single-Precision Floating-Point Computational Instructions . . . 61

© ISO/IEC 2026 – All rights reserved

9.7 Single-Precision Floating-Point Conversion and Move Instructions . . . . . 63

9.8 Single-Precision Floating-Point Compare Instructions . . . . . 65

9.9 Single-Precision Floating-Point Classify Instruction . . . . . 65

**10 "D" Standard Extension for Double-Precision Floating-Point, Version 2.2 67**

10.1 D Register State . . . . . 67

10.2 NaN Boxing of Narrower Values . . . . . 67

10.3 Double-Precision Load and Store Instructions . . . . . 68

10.4 Double-Precision Floating-Point Computational Instructions . . . . . 69

10.5 Double-Precision Floating-Point Conversion and Move Instructions . . . . . 69

10.6 Double-Precision Floating-Point Compare Instructions . . . . . 71

10.7 Double-Precision Floating-Point Classify Instruction . . . . . 71

**11 "Q" Standard Extension for Quad-Precision Floating-Point, Version 2.2 73**

11.1 Quad-Precision Load and Store Instructions . . . . . 73

11.2 Quad-Precision Computational Instructions . . . . . 74

11.3 Quad-Precision Convert and Move Instructions . . . . . 74

11.4 Quad-Precision Floating-Point Compare Instructions . . . . . 75

11.5 Quad-Precision Floating-Point Classify Instruction . . . . . 76

**12 RVWMO Memory Consistency Model, Version 2.0 77**

12.1 Definition of the RVWMO Memory Model . . . . . 77

12.2 CSR Dependency Tracking Granularity . . . . . 82

12.3 Source and Destination Register Listings . . . . . 82

12.4 RVWMO Operational Model . . . . . 87

12.4.1 Introduction to the RVWMO Operational Model . . . . . 88

12.4.2 Intra-instruction Pseudocode Execution . . . . . 91

12.4.3 Instruction Instance State . . . . . 92

12.4.4 Hart State . . . . . 94

© ISO/IEC 2026 – All rights reserved

12.4.5 Shared Memory State . . . . . 94
12.4.6 Transitions . . . . . 94
12.4.7 Limitations . . . . . 103

**13 "C" Standard Extension for Compressed Instructions, Version 2.0 105**

13.1 Overview . . . . . 105
13.2 Compressed Instruction Formats . . . . . 107
13.3 Load and Store Instructions . . . . . 108
13.4 Control Transfer Instructions . . . . . 112
13.5 Integer Computational Instructions . . . . . 113
13.6 Usage of C Instructions in LR/SC Sequences . . . . . 117
13.7 HINT Instructions . . . . . 118
13.8 RVC Instruction Set Listings . . . . . 119

**14 RV32/64G Instruction Set Listings 121**

© ISO/IEC 2026 – All rights reserved

# Chapter 1

# Introduction

RISC-V (pronounced "risk-five") is a new instruction-set architecture (ISA) that was originally designed to support computer architecture research and education, but which we now hope will also become a standard free and open architecture for industry implementations. Our goals in defining RISC-V include:

- A completely *open* ISA that is freely available to academia and industry.
- A *real* ISA suitable for direct native hardware implementation, not just simulation or binary translation.
- An ISA that avoids "over-architecting" for a particular microarchitecture style (e.g., microcoded, in-order, decoupled, out-of-order) or implementation technology (e.g., full-custom, ASIC, FPGA), but which allows efficient implementation in any of these.
- An ISA separated into a *small* base integer ISA, usable by itself as a base for customized accelerators or for educational purposes, and optional standard extensions, to support general-purpose software development.
- Support for the revised 2008 IEEE-754 floating-point standard [5].
- An ISA supporting extensive ISA extensions and specialized variants.
- Both 32-bit and 64-bit address space variants for applications, operating system kernels, and hardware implementations.
- An ISA with support for highly-parallel multicore or manycore implementations, including heterogeneous multiprocessors.
- Optional *variable-length instructions* to both expand available instruction encoding space and to support an optional *dense instruction encoding* for improved performance, static code size, and energy efficiency.
- An ISA that simplifies experiments with new privileged architecture designs.

---

*Commentary on our design decisions is formatted as in this paragraph. This non-normative text can be skipped if the reader is only interested in the specification itself.*

---

*The name RISC-V was chosen to represent the fifth major RISC ISA design from UC Berkeley (RISC-I [12], RISC-II [6], SOAR [18], and SPUR [9] were the first four). We also pun on the use of the Roman numeral "V" to signify "variations" and "vectors", as support for a range of architecture research, including various data-parallel accelerators, is an explicit goal of the ISA design.*



© ISO/IEC 2026 – All rights reserved

The RISC-V ISA is defined avoiding implementation details as much as possible (although commentary is included on implementation-driven decisions) and should be read as the software-visible interface to a wide variety of implementations rather than as the design of a particular hardware artifact. The RISC-V manual is structured in two volumes. This volume covers the design of the base *unprivileged* instructions, including optional unprivileged ISA extensions. Unprivileged instructions are those that are generally usable in all privilege modes in all privileged architectures, though behavior might vary depending on privilege mode and privilege architecture. The second volume provides the design of the first ("classic") privileged architecture. The manuals use IEC 80000-13:2008 conventions, with a byte of 8 bits.

---

*In the unprivileged ISA design, we tried to remove any dependence on particular microarchitectural features, such as cache line size, or on privileged architecture details, such as page translation. This is both for simplicity and to allow maximum flexibility for alternative microarchitectures or alternative privileged architectures.*

## 1.1 RISC-V Hardware Platform Terminology

A RISC-V hardware platform can contain one or more RISC-V-compatible processing cores together with other non-RISC-V-compatible cores, fixed-function accelerators, various physical memory structures, I/O devices, and an interconnect structure to allow the components to communicate.

A component is termed a *core* if it contains an independent instruction fetch unit. A RISC-V-compatible core might support multiple RISC-V-compatible hardware threads, or *harts*, through multithreading.

A RISC-V core might have additional specialized instruction-set extensions or an added *coprocessor*. We use the term *coprocessor* to refer to a unit that is attached to a RISC-V core and is mostly sequenced by a RISC-V instruction stream, but which contains additional architectural state and instruction-set extensions, and possibly some limited autonomy relative to the primary RISC-V instruction stream.

We use the term *accelerator* to refer to either a non-programmable fixed-function unit or a core that can operate autonomously but is specialized for certain tasks. In RISC-V systems, we expect many programmable accelerators will be RISC-V-based cores with specialized instruction-set extensions and/or customized coprocessors. An important class of RISC-V accelerators are I/O accelerators, which offload I/O processing tasks from the main application cores.

The system-level organization of a RISC-V hardware platform can range from a single-core microcontroller to a many-thousand-node cluster of shared-memory manycore server nodes. Even small systems-on-a-chip might be structured as a hierarchy of multicomputers and/or multiprocessors to modularize development effort or to provide secure isolation between subsystems.

## 1.2 RISC-V Software Execution Environments and Harts

The behavior of a RISC-V program depends on the execution environment in which it runs. A RISC-V execution environment interface (EEI) defines the initial state of the program, the number

© ISO/IEC 2026 – All rights reserved