RVB23 Profiles

Version 1.0, 2024-10-17: This document is in Ratified state.

Table of Contents

Introduction

This document specifies the RVB23 profile family. RVB23 is the first major release of the RVB series of RISC-V Application Processor Profile.

RVB profiles are intended to be used for customized 64-bit application processors that will run rich OS stacks, but usually as a custom build of standard OS source-code distributions. The approach is to provide a large guaranteed set of relatively inexpensive and/or widely beneficial features but allow optionality for more expensive and/or more targeted extensions.

Unlike the RVA profiles, it is explicitly a non-goal of RVB profiles to provide a single standard ISA interface supporting a wide variety of binary kernel and binary application software distributions. However, individual software ecosystems may build upon RVB profiles to produce a more targeted standard interface for a certain market.

Profile-Defined Extensions

RVB23 has been ratified alongside RVA23, and the same set of new profile-defined extensions defined in RVA23 are present in RVB23. These profile-defined extensions will soon move to the combined ISA manual. Future releases of RVA and RVB profiles might not proceed through ratification at the same time, and future profile-defined extensions will be presented as an edit to the combined ISA manual.

RVB23 Profiles

Only user-mode (RVB23U64) and supervisor-mode (RVB23S64) profiles are specified in this family.

RVB23U64 Profile

The RVB23U64 profile specifies the ISA features available to user-mode execution environments in 64-bit RVB applications processors.

RVB23U64 Mandatory Base

RV64I is the mandatory base ISA for RVB23U64 and is little-endian. As per the unprivileged architecture specification, the `ECALL` instruction causes a requested trap to the execution environment.

RVB23U64 Mandatory Extensions

The following mandatory extensions in RVB23U64 were also mandatory in RVA22U64.

- **M** Integer multiplication and division.
- **A** Atomic instructions.
- **F** Single-precision floating-point instructions.
- **D** Double-precision floating-point instructions.
- **C** Compressed instructions.
- **B** Bit-manipulation instructions.
- **Zicsr** CSR instructions. These are implied by presence of F.
- **Zicntr** Base counters and timers.
- **Zihpm** Hardware performance counters.
- **Ziccif** Main memory regions with both the cacheability and coherence PMAs must support instruction fetch, and any instruction fetches of naturally aligned power-of-2 sizes up to min(ILEN,XLEN) (i.e., 32 bits for RVB23) are atomic.
- **Ziccrse** Main memory regions with both the cacheability and coherence PMAs must support RsrvEventual.
- **Ziccamoa** Main memory regions with both the cacheability and coherence PMAs must support all atomics in A.
- **Zicclsm** Misaligned loads and stores to main memory regions with both the cacheability and coherence PMAs must be supported.
- **Za64rs** Reservation sets are contiguous, naturally aligned, and a maximum of 64 bytes.
- **Zihintpause** Pause hint.
- **Zic64b** Cache blocks must be 64 bytes in size, naturally aligned in the address space.
- **Zicbom** Cache-block management instructions.
- **Zicbop** Cache-block prefetch instructions.
- **Zicboz** Cache-block zero instructions.

- **Zkt** Data-independent execution latency.

The following mandatory extensions are also present in RVA23U64:

- **Zihintntl** Non-temporal locality hints.
- **Zicond** Integer conditional operations.
- **Zimop** May-be-operations.
- **Zcmop** Compressed may-be-operations.
- **Zcb** Additional compressed instructions.
- **Zfa** Additional floating-point instructions.
- **Zawrs** Wait-on-reservation-set instructions.

RVB23U64 Optional Extensions

RVB23U64 has 18 profile options listed below.

Localized Options

The following extensions are localized options in both RVA23U64 and RVB23U64:

- **Zvkng** Vector crypto NIST Algorithms with GCM.
- **Zvksg** Vector crypto ShangMi Algorithms with GCM.

The following extensions options are localized options in RVB23U64 but are not present in RVA23U64:

- **Zvkg** Vector GCM/GMAC instructions.
- **Zvknc** Vector crypto NIST algorithms with carryless multiply.
- **Zvksc** Vector crypto ShangMi algorithms with carryless multiply.

RVA profiles mandate the higher-performing but more expensive GHASH options when adding vector crypto. To reduce implementation cost, RVB profiles also allow these carryless multiply options (Zvknc and Zvksc) to implement GCM efficiently, with GHASH available as a separate option.

- **Zkn** Scalar crypto NIST algorithms.
- **Zks** Scalar crypto ShangMi algorithms.

RVA23 profiles drop support for scalar crypto as an option, as the vector extension is now mandatory in RVA23. RVB23 profiles support scalar crypto, as the vector extension is optional in RVB23.

Development Options

The following are new development options intended to become mandatory in a later RVB profile:

- **Zabha** Byte and halfword atomic memory operations.
- **Zacas** Compare-and-Swap instructions.

- **Ziccamoc** Main memory regions with both the cacheability and coherence PMAs must provide `AMOCASQ` level PMA support.
- **Zama16b** Misaligned loads, stores, and AMOs to main memory regions that do not cross a naturally aligned 16-byte boundary are atomic.

Expansion Options

The following are expansion options in RVB23U64, but are mandatory in RVA23U64.

- **Zfhmin** Half-precision floating-point.
- **V** Vector extension.

Unclear if other Zve extensions should also be supported in RVB.*

- **Zvfhmin** Vector minimal half-precision floating-point.
- **Zvbb** Vector basic bit-manipulation instructions.
- **Zvkt** Vector data-independent execution latency.
- **Supm** Pointer masking, with the execution environment providing a means to select PMLEN=0 and PMLEN=7 at minimum.

The following extensions are expansion options in both RVA23U64 and RVB23U64:

- **Zfh** Scalar half-precision floating-point.
- **Zbc** Scalar carryless multiplication.
- **Zicfilp** Landing Pads.
- **Zicfiss** Shadow Stack.
- **Zvfh** Vector half-precision floating-point.
- **Zfbfmin** Scalar BF16 converts.
- **Zvfbfmin** Vector BF16 converts.
- **Zvfbfwma** Vector BF16 widening mul-add.

The following are expansion options for RVB23U64 as they are not intended to be made mandatory in future RVB profiles, but are listed as RVA23U64 development options as they are intended to become mandatory in future RVA profiles.

- **Zvbc** Vector carryless multiplication.

Transitory Options

There are no transitory options in RVB23U64.

RVB23U64 Recommendations

Implementations are strongly recommended to raise illegal-instruction exceptions on attempts to execute unimplemented opcodes.

RVB23S64 Profile

The RVB23S64 profile specifies the ISA features available to a supervisor-mode execution environment in 64-bit applications processors. RVB23S64 is based on privileged architecture version 1.13.

Priv 1.13 is still being defined.

RVB23S64 Mandatory Base

RV64I is the mandatory base ISA for RVB23S64 and is little-endian. The `ECALL` instruction operates as per the unprivileged architecture specification. An `ECALL` in user mode causes a contained trap to supervisor mode. An `ECALL` in supervisor mode causes a requested trap to the execution environment.

RVB23S64 Mandatory Extensions

The following unprivileged extensions are mandatory:

- The RVB23S64 mandatory unprivileged extensions include all the mandatory unprivileged extensions in RVB23U64.
- **Zifencei** Instruction-Fetch Fence.

Zifencei is mandated as it is the only standard way to support instruction-cache coherence in RVB23 application processors. A new instruction-cache coherence mechanism is under development (tentatively named Zjid) which might be added as an option in the future.

The following privileged extensions are mandatory, and are also mandatory in RVA23S64.

- **Ss1p13** Supervisor architecture version 1.13.

Ss1p13 supersedes Ss1p12 but is not yet ratified.

- **Svnapot** NAPOT translation contiguity.

Svnapot is very low cost to provide, so is made mandatory even in RVB.

- **Svbare** The `satp` mode Bare must be supported.
- **Sv39** Page-Based 39-bit Virtual-Memory System.
- **Svade** Page-fault exceptions are raised when a page is accessed when A bit is clear, or written when D bit is clear.
- **Ssccptr** Main memory regions with both the cacheability and coherence PMAs must support hardware page-table reads.
- **Sstvecd** `stvec.MODE` must be capable of holding the value 0 (Direct). When `stvec.MODE=Direct`, `stvec.BASE` must be capable of holding any valid four-byte-aligned address.
- **Sstvala** `stval` must be written with the faulting virtual address for load, store, and instruction page-fault, access-fault, and misaligned exceptions, and for breakpoint exceptions other than those caused by execution of the `EBREAK` or `C.EBREAK` instructions. For virtual-instruction and illegal-instruction exceptions, `stval` must be written with the faulting instruction.
- **Sscounterenw** For any `hpmcounter` that is not read-only zero, the corresponding bit in `scounteren` must be writable.

- **Svpbmt** Page-based memory types.
- **Svinval** Fine-grained address-translation cache invalidation.
- **Sstc** supervisor-mode timer interrupts.
- **Sscofpmf** Count overflow and mode-based filtering.
- **Ssu64xl** `sstatus.UXL` must be capable of holding the value 2 (i.e., UXLEN=64 must be supported).

RVB23S64 Optional Extensions

RVB23S64 has the same unprivileged options as RVB23U64,

The privileged options in RVB23S64 are listed in the following sections.

Localized Options

There are no privileged localized options in RVB23S64.

Development Options

There are no privileged development options in RVB23S64.

Expansion Options

The following are privileged expansion options in RVB23S64, but are mandatory in RVA23S64:

- **Ssnpm** Pointer masking, with `senvcfg.PME` supporting at minimum, settings PMLEN=0 and PMLEN=7.
- **Sha** The augmented hypervisor extension.

When the hypervisor extension is implemented, the following are also mandatory:

- If the hypervisor extension is implemented and pointer masking (Ssnpm) is supported then `henvcfg.PME` must support at minimum, settings PMLEN=0 and PMLEN=7.

The following are privileged expansion options in RVB23S64 that are also privileged expansion options in RVA23S64:

- **Sv48** Page-based 48-bit virtual-memory system.
- **Sv57** Page-based 57-bit virtual-memory system.
- **Svadu** Hardware A/D bit updates.
- **Zkr** Entropy CSR.
- **Sdtrig** Debug triggers.
- **Ssstrict** No non-conforming extensions are present. Attempts to execute unimplemented opcodes or access unimplemented CSRs in the standard or reserved encoding spaces raises an illegal instruction exception that results in a contained trap to the supervisor-mode trap handler.

Ssstrict does not prescribe behavior for the custom encoding spaces or CSRs.

Ssstrict definition applies to the execution environment claiming to be RVA23-compatible, which must have the hypervisor extension. That execution environment will take a

> *contained trap to supervisor-mode (however that trap is implemented, including, but not limited to, emulation/delegation in the outer execution environment). Ssstrict (and all the other RVA23 mandates and options) do not apply to any guest VMs run by a hypervisor. An RVA23 hypervisor can provide guest VMs that are also RVA23-compatible but with an expanded set of emulated standard instructions. An RVA23 hypervisor can also choose to implement guest VMs that are not RVA23 compatible (e.g., lacking H, or only RVA20).*

- **Svvptc** Transitions from invalid to valid PTEs will be visible in bounded time without an explicit memory-management fence.
- **Sspm** Supervisor-mode pointer masking, with the supervisor execution environment providing a means to select PMLEN=0 and PMLEN=7 at minimum.

RVB23S64 Recommendations

- Implementations are strongly recommended to raise illegal-instruction exceptions when attempting to execute unimplemented opcodes.

Glossary of ISA Extensions

The following unprivileged ISA extensions are defined in Volume I of the RISC-V Instruction Set Manual.

- M Extension for Integer Multiplication and Division
- A Extension for Atomic Memory Instructions
- F Extension for Single-Precision Floating-Point
- D Extension for Double-Precision Floating-Point
- H Hypervisor Extension
- Q Extension for Quad-Precision Floating-Point
- C Extension for Compressed Instructions
- B Extension for Bit Manipulation
- V Extension for Vector Computation
- Zifencei Instruction-Fetch Fence Extension
- Zicsr Extension for Control and Status Register Access
- Zicntr Extension for Basic Performance Counters
- Zihpm Extension for Hardware Performance Counters
- Zihintpause Pause Hint Extension
- Zfh Extension for Half-Precision Floating-Point
- Zfhmin Minimal Extension for Half-Precision Floating-Point
- Zfinx Extension for Single-Precision Floating-Point in x-registers
- Zdinx Extension for Double-Precision Floating-Point in x-registers
- Zhinx Extension for Half-Precision Floating-Point in x-registers
- Zhinxmin Minimal Extension for Half-Precision Floating-Point in x-registers
- Zba Address Computation Extension
- Zbb Bit Manipulation Extension
- Zbc Carryless Multiplication Extension
- Zbs Single-Bit Manipulation Extension
- Zk Standard Scalar Cryptography Extension
- Zkn NIST Cryptography Extension
- Zknd AES Decryption Extension
- Zkne AES Encryption Extension
- Zknh SHA2 Hashing Extension
- Zkr Entropy Source Extension
- Zks ShangMi Cryptography Extension
- Zksed SM4 Block Cypher Extension
- Zksh SM3 Hashing Extension

- Zkt Extension for Data-Independent Execution Latency
- Zicbom Extension for Cache-Block Management
- Zicbop Extension for Cache-Block Prefetching
- Zicboz Extension for Cache-Block Zeroing
- Zawrs Wait-on-reservation-set instructions
- Zacas Extension for Atomic Compare-and-Swap (CAS) instructions
- Zabha Extension for Byte and Halfword Atomic Memory Operations
- Zbkb Extension for Bit Manipulation for Cryptography
- Zbkc Extension for Carryless Multiplication for Cryptography
- Zbkx Crossbar Permutation Extension
- Zvbb - Vector Basic Bit-manipulation
- Zvbc - Vector Carryless Multiplication
- Zvkng - NIST Algorithm Suite with GCM
- Zvksg - ShangMi Algorithm Suite with GCM
- Zvkt - Vector Data-Independent Execution Latency

The following privileged ISA extensions are defined in Volume II of the RISC-V Instruction Set Manual.

- Sv32 Page-based Virtual Memory Extension, 32-bit
- Sv39 Page-based Virtual Memory Extension, 39-bit
- Sv48 Page-based Virtual Memory Extension, 48-bit
- Sv57 Page-based Virtual Memory Extension, 57-bit
- Svpbmt, Page-Based Memory Types
- Svnapot, NAPOT Translation Contiguity
- Svinval, Fine-Grained Address-Translation Cache Invalidation
- Hypervisor Extension
- Sm1p11, Machine Architecture v1.11
- Sm1p12, Machine Architecture v1.12
- Ss1p11, Supervisor Architecture v1.11
- Ss1p12, Supervisor Architecture v1.12
- Ss1p13, Supervisor Architecture v1.13
- Sstc Extension for Supervisor-mode Timer Interrupts
- Sscofpmf Extension for Count Overflow and Mode-Based Filtering
- Smstateen/Ssstateen Extension for State-enable
- Svvptc Obviating Memory-management Instructions after Marking PTEs valid
- Svadu Hardware Updating of A/D Bits

The following extensions have not yet been incorporated into the RISC-V Instruction Set Manual; the

hyperlinks lead to their separate specifications.

- Zve32x Extension for Embedded Vector Computation (32-bit integer)
- Zve32f Extension for Embedded Vector Computation (32-bit integer, 32-bit FP)
- Zve32d Extension for Embedded Vector Computation (32-bit integer, 64-bit FP)
- Zve64x Extension for Embedded Vector Computation (64-bit integer)
- Zve64f Extension for Embedded Vector Computation (64-bit integer, 32-bit FP)
- Zve64d Extension for Embedded Vector Computation (64-bit integer, 64-bit FP)
- **Ziccif**: Main memory supports instruction fetch with atomicity requirement
- **Ziccrse**: Main memory supports forward progress on LR/SC sequences
- **Ziccamoa**: Main memory supports all atomics in A
- **Ziccamoc** Main memory supports atomics in Zacas
- **Zicclsm**: Main memory supports misaligned loads/stores
- **Zama16b**: Misaligned loads, stores, and AMOs to main memory regions that do not cross a naturally aligned 16-byte boundary are atomic.
- **Za64rs**: Reservation set size of at most 64 bytes
- **Za128rs**: Reservation set size of at most 128 bytes
- **Zic64b**: Cache block size is 64 bytes
- **Svbare**: Bare mode virtual-memory translation supported
- **Svade**: Raise exceptions on improper A/D bits
- **Ssccptr**: Main memory supports page table reads
- **Sscounterenw**: Support writeable enables for any supported counter
- **Sstvecd**: `stvec` supports Direct mode
- **Sstvala**: `stval` provides all needed values
- **Ssu64xl**: UXLEN=64 must be supported
- **Sha**: Augmented hypervisor extension
- **Shcounterenw**: Support writeable enables for any supported counter
- **Shvstvala**: `vstval` provides all needed values
- **Shtvala**: `htval` provides all needed values
- **Shvstvecd**: `vstvec` supports Direct mode
- **Shvsatpa**: `vsatp` supports all modes supported by `satp`
- **Shgatpa**: SvNNx4 mode supported for all modes supported by `satp`, as well as Bare
- **Ssstrict**: Unimplemented reserved encodings raise illegal instruction exceptions and no non-conforming extension are present