
# RISC-V Server Platform Specification

Server Platform Task Group

Version v1.0, 2026-05-06: Ratified

# Table of Contents

# Preamble

*This document is in the Ratified state*

No changes are allowed. Any necessary or desired modifications must be addressed through a follow-on extension. Ratified extensions are never revised.

# Copyright and license information

# Contributors

This RISC-V specification has been contributed to directly or indirectly by (in alphabetical order):

Andrea Bolognani, Andrei Warkentin, Andrew Jones, Greg Favor, Heinrich Schuchardt, Martin Maas, Paul Walmsley, Radim Krčmář, Ravi Sahita, Ved Shanbhogue

# Chapter 1. Introduction

The RISC-V Server Platform specification defines a standardized set of hardware and sofware capabilities, that portable system software, such as operating systems and hypervisors, can rely on being present in a RISC-V server platform.

A server is a computing system designed to manage and distribute resources, services, and data to other computers or devices on a network. It is often referred to as a 'server' because it serves or provides information and resources upon request. Such computing systems are designed to operate continually and have higher requirements for capabilities such as RAS, security, performance, and quality of service. Examples of servers include web servers, file servers, database servers, mail servers, game servers, and more. This specification focuses on defining requirements for general-purpose server computing systems that may be used for one or more of these purposes.

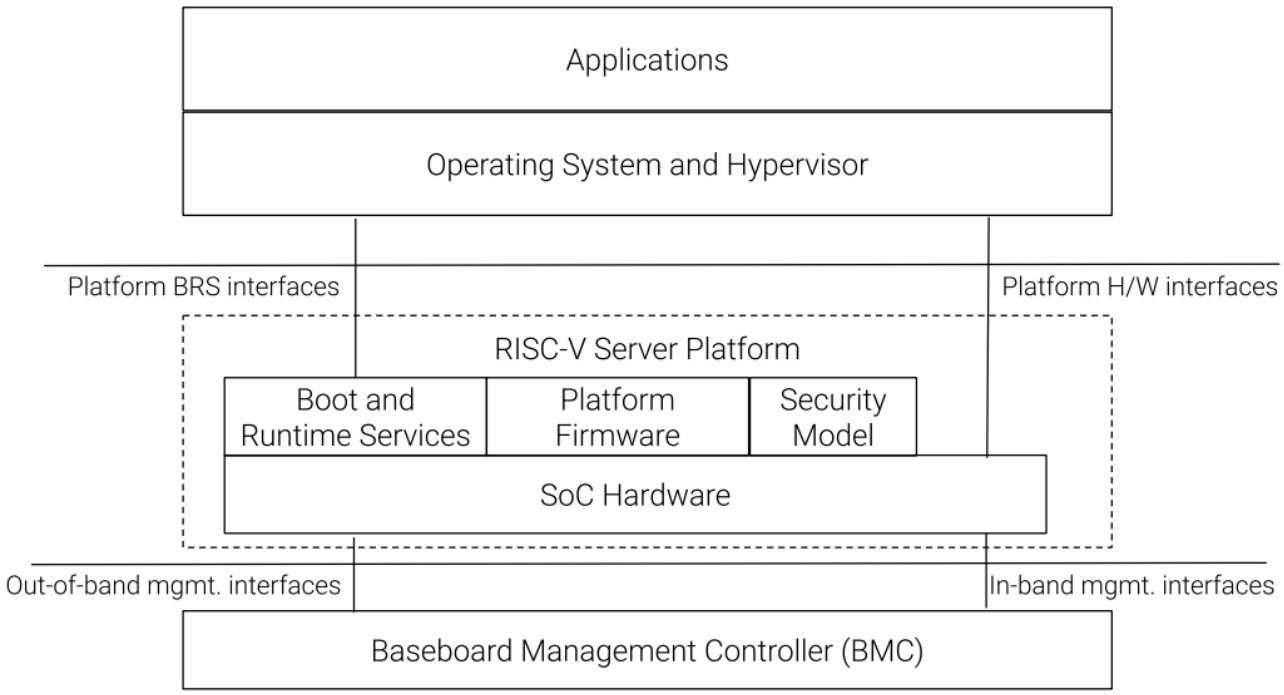


*Figure 1. Components of a RISC-V Server Platform*

The RISC-V server platform is defined as the collection of RVA profile-compliant application processor harts, SoC hardware, peripherals, platform firmware, boot/runtime services, and platform security services. The platform provides hardware interfaces (e.g., harts, timers, interrupt controllers, PCIe root ports, etc.) to portable system software. It also offers a set of standardized boot and runtime services based on the UEFI and ACPI standards. To support provisioning and platform management, it interfaces with a baseboard management controller (BMC) through both in-band and out-of-band (OOB) management interfaces. The in-band management interfaces support the use of standard manageability specifications like MCTP, PLDM, IPMI, and Redfish for provisioning and management of the operating system executing on the platform. The OOB interface supports the use of standard manageability specifications like MCTP, PLDM, Redfish, and IPMI for functions such as power management, telemetry, debug, and provisioning. The platform security model includes guidelines and requirements for aspects such as debug authorization, secure/measured boot, firmware updates, firmware resilience, and confidential computing, among others. A primary goal of this specification is to enable interoperability between conforming

servers and the large ecosystem of portable system software, available from a variety of sources, such that conforming servers are not limited to any single system software distribution.

The platform firmware, typically operating at privilege level M, is considered part of the platform and is usually expected to be customized and tailored to meet the requirements of the SoC hardware (e.g., initialization of address decoders, memory controllers, RAS, etc.), boot/runtime services and platform security.

This specification standardizes the rules for hardware and software interfaces and capabilities by building on top of relevant RISC-V standards, such as the RISC-V Architecture Profiles, Server SoC, Boot and Runtime Services and Platform Security specifications for server software executing on the application processor harts at privilege levels below M. It enables operating system and hypervisor vendors to support such platforms with a single binary OS image distribution model. The rules provided by this specification represent a standard set of infrastructural capabilities, encompassing areas where divergence is typically unnecessary and where novelty is absent across implementations.

To be compliant with this specification, the server platform MUST support all mandatory rules and MUST support the listed versions of the specifications. This standard set of capabilities MAY be extended by a specific implementation with additional standard or custom capabilities, including compatible later versions of listed standard specifications. Portable system software MUST support the specified mandatory capabilities to be compliant with this specification.

The rules in this specification use the following format:

| ID# | Rules |
|---|---|
| CAT_NNN | The `CAT` is a category prefix that logically groups the rules and is followed by 3 digits - `NNN` - assigning a numeric ID to the rule.<br><br>The rules use the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" that are to be interpreted as described in RFC 2119 [1] when, and only when, they appear in all capitals, as shown here. When these words are not capitalized, they have their normal English meanings. |
| *A rule or a group of rules may be followed by non-normative text providing context or justification for the rule. The non-normative text may also be used to reference sources that are the origin of the rule.* | |

This specification groups the rules in the following broad categories:

- Hardware
- Firmware
- Security

## 1.1. Glossary

Most terminology has the standard RISC-V meaning. This table captures other terms used in the document. Terms in the document prefixed by 'PCIe' have the meaning defined in the PCI Express (PCIe) Base Specification [2] (even if they are not in this table).

*Table 1. Terms and definitions*

| Term | Definition |
|---|---|
| ACPI | Advanced Configuration and Power Interface [3]. |
| BMC | Baseboard Management Controller. A motherboard resident management controller that provides functions for platform management. |
| Guest | Software in a virtual machine. |
| ID | Identifier. |
| OS | Operating System or Hypervisor. |
| SBI | RISC-V Supervisor Binary Interface Specification [4]. |
| SEE | Supervisor Execution Environment. The environment in which supervisor-level software (such as an OS) executes. |
| SoC | System on a chip, also referred as system-on-a-chip and system-on-chip. |
| UEFI | Unified Extensible Firmware Interface [5]. |
| VM | Virtual Machine. |

# Chapter 2. Server Platform Hardware Rules

## 2.1. RISC-V Harts

A RISC-V server platform includes one or more RISC-V application processors and may include one or more service processors. These service processors may provide services such as security and power management to software executing on the application processors, and they may themselves implement the RISC-V ISA. The rules in this section apply solely to harts in the application processors of the SoC.

| ID# | Rule |
|---|---|
| RVA_010 | The RISC-V application processor harts in the SoC MUST support the RVA23S64 ISA profile [6]. |
| RVA_020 | The RISC-V application processor harts in the SoC MUST support the following extensions:<br>• Sv48<br>• Sdtrig [7]<br>• Sdext [7]<br>• Zkr<br>• Ssccfg<br>• Ssstrict [6]<br>• Ssaia [8] |
| *These mandates may be moved into a new ISA profile specification. The motivation for requiring Zkr is that servers typically need access to an entropy source at boot time.* | |
| RVA_030 | The RISC-V application processor harts in the SoC SHOULD support the Svadu extension. |
| *Svadu is expected to become mandatory in a future version of this specification.* | |
| RVA_040 | The RISC-V application processor harts in the SoC SHOULD support the Ssctr extension ([9]). If the Ssctr extension is implemented, a CTR depth of 32 MUST be supported. Additional CTR depths MAY also be supported. |
| *The motivation for suggesting Ssctr is that similar capabilities from other architectures are used by profile-guided optimization (PGO) tools to improve builds for workloads typical of servers. Implementing a CTR depth of 32 provides a common CTR depth across implementations for purposes of VM migration. Ssctr is expected to become mandatory in a future version of this specification.* | |
| RVA_050 | The RISC-V application processor harts within a supervisor execution environment (SEE) must all support the same ISA and non-ISA extensions and the same implementation-dependent choices with those extensions, except when such differences do not interfere with architectural state migration and subsequent execution of threads between harts. |

| ID# | Rule |
|---|---|
| *Architectural state comprises all ISA-visible state relevant to the SEE, including but not limited to: XLEN, VLEN, cache block sizes, endianness, privilege levels, supported extensions and their versions, and architecturally defined CSRs and their fields.*<br><br>*Permitted divergences (non-exhaustive):*<br><br>• *Behaviors the ISA marks as RESERVED or UNSPECIFIED (e.g., outcomes explicitly left undefined, reserved encodings, or implementation-defined choices) when such differences do not affect state save/restore or execution correctness upon migration. If an execution thread is migrated, then the state saved from a first hart when restored on a second hart will result in the same execution results when execution is constrained to behavior that is neither RESERVED nor UNSPECIFIED.*<br>• *Microarchitectural properties not architecturally visible (e.g., cache sizes/associativity/latency, pipeline depth, frequency, power states).*<br><br>*Prohibited divergences (non-exhaustive):*<br><br>• *Differences in XLEN, VLEN, cache block sizes, endianness, mandatory privilege features, or any extension that is reported as present within the SEE on any application processor hart.*<br>• *Differences in the advertised memory consistency model or coherency behavior required by the ISA and platform specification.* | |
| RVA_060 | The RISC-V application processor MUST support at least 6 hardware performance counters defined by the Zihpm extension in addition to the three counters defined by the Zicntr extension. |
| *The motivation for including at least six performance counters, in addition to the three counters defined by the Zicntr extension, originates from prior experience with x86 servers.* | |

## 2.2. RISC-V Hart SEE

| ID# | Rule |
|---|---|
| SEE_010 | The RISC-V application processor hart MUST support:<br><br>• Single stepping using the step bit in `dcsr` |

| ID# | Rule |
|---|---|
| SEE_020 | The RISC-V application processor hart MUST support:<br>• At least 4 instruction address match triggers.<br>• At least 4 load/store address match triggers.<br>• At least one icount trigger to support single stepping.<br>• At least one interrupt trigger.<br>• At least one exception trigger.<br>• All triggers MUST support<br>◦ Privilege mode filtering (VS, VU, S, U).<br>◦ Meeting the requirements for configuring action=0.<br>◦ Matching all legal addresses using instruction and load/store address triggers with action=0.<br>◦ Filtering using mhselect values of 0, 2, and 6.<br>◦ Filtering using sselect values of 0 and 2.<br>• Triggers MAY support<br>◦ Filtering using mhselect values of 1 and 5.<br>◦ Filtering using sselect values of 1.<br>• Supported mhselect and sselect values MUST be identical for all triggers.<br>• When trigger filtering using hcontext is supported, hcontext MUST be at least 14 bits wide, and the filter must support matching all values that can be held in hcontext.<br>• When trigger filtering using scontext is supported, scontext MUST be at least 32 bits wide, and the filter must support matching all values that can be held in scontext. |
| *The motivation for including at least four instruction address and four load/store address match triggers originates from prior experience with x86 servers.* | |

## 2.3. RISC-V SoC

| ID# | Rule |
|---|---|
| HSOC_010 | RISC-V SoCs MUST comply with the Server SoC v1.0 specification [10]. |
| HSOC_020 | All SoC peripherals that are intended by the platform design to be assignable to a virtual machine or exposed to a user-space device driver MUST be PCIe devices or comply with the rules for SoC-integrated PCIe devices ([10], Section 2.4). |

| ID# | Rule |
|---|---|
| *This rule does not apply to components not intended by the platform for virtual machine or user-space assignment such as:*<br>• *Interrupt controllers (e.g., APLIC).*<br>• *IOMMUs.*<br>• *Debug modules, trace control blocks, RAS banks, and performance monitoring units.*<br>• *Controllers, including the root of trust (RoT) controllers, power management controllers, and other SoC management controllers.* | |

## 2.4. Peripherals

| ID# | Rule |
|---|---|
| HPER_010 | For remote-access and system engineering purposes in early boot software, either a fully 16550-compatible [11] or a fully pl011-compatible [12] UART MUST be implemented. |
| *This is a stronger requirement than the Server SoC MNG_030 rule [10]. The intention here is to simplify early boot software support requirements. This UART is not intended to be directly assignable to virtual machines, and thus there is no requirement for this UART to appear as a PCI device. This specification does not provide guidance around how the UART is physically exposed, i.e. via RS232 signalling, USB, a BMC or other mechanism.* | |
| HPER_020 | The implemented UART MUST support:<br>• Interrupt-driven operation using a wired interrupt.<br>• Flow control.<br>• 115200 baud operation. |
| HPER_030 | If a USB controller is implemented, it MUST comply with XHCI 1.2 or later [13]. |
| HPER_040 | Implemented XHCI controllers MUST support:<br>• 64-bit addressing (AC64 = '1').<br>• A 4K PAGESIZE. |
| HPER_050 | If a SATA controller is implemented, it MUST comply with AHCI 1.3.1 or later [14]. |
| HPER_060 | Implemented AHCI controllers MUST support:<br>• 64-bit addressing (S64A = '1'). |
| HPER_070 | A battery-backed Real Time Clock (the "Server Platform RTC") MUST be implemented for use by platform firmware for UEFI certificate validity checking. This RTC MAY optionally be used by other system functions. |

| ID# | Rule |
|---|---|
| HPER_080 | If the operating system does not have access to its own OS-managed Real Time Clock, the Server Platform RTC SHOULD be exposed to the operating system for clock read access via EFI_GET_TIME, and, if the system security profile allows the operating system to change the Server Platform RTC clock, for clock setting access via EFI_SET_TIME. |
| *Allowing operating systems to change the time and date used for UEFI certificate validity checks may have unexpected consequences, including, for example, disrupting certificate verification in platform firmware, or affecting system functions other than the OS that rely on the Server Platform RTC.* | |
| HPER_090 | A Trusted Platform Module (TPM) MUST be implemented and adhere to the TPM 2.0 Library specification [15]. |
| *It is common for secure systems to support multiple trust chains with their own root of trust. For example, a TPM can be secondary root of trust for UEFI boot flows while a hardware RoT is the root of trust for platform firmware, platform attestation, security lifecycle management of the secondary roots of trust, among others.* | |

# Chapter 3. Server Platform Firmware Rules

| ID# | Rule |
|---|---|
| FIRM_010 | RISC-V SoCs MUST comply with the BRS-I recipe described in the Boot and Runtime Service v1.0 specification [16]. |
| FIRM_020 | The firmware MUST implement the SBI v3.0 Debug Triggers (DBTR) extension [4]. |
| *Supervisor software needs DBTR in order to utilize Sdtrig, which is mandated by rule RVA_020.* | |
| FIRM_030 | If the software running on the application processor supports RAS functionality for RISC-V components, the firmware MUST implement the SBI v3.0 Supervisor Software Events (SSE) extension [4]. |
| FIRM_040 | The firmware MUST include configuration infrastructure, supporting relevant HII protocols ([17] number 2). |
| FIRM_050 | The firmware SHOULD include the ability to boot from disk (block) device, supporting relevant protocols ([17] number 5). |
| FIRM_060 | The firmware SHOULD include the ability to perform a TFTP-based boot from a network device ([17] number 6). |
| FIRM_070 | The firmware SHOULD include the ability to validate boot images. |
| FIRM_080 | The firmware SHOULD support UEFI general purpose network applications, including IPv4, IPv6, DNS, TLS, IPSec and VLAN features, supporting relevant protocols ([17] number 7). |
| FIRM_090 | The firmware SHOULD support RISC-V option ROMs, compiled for the RVA20 profile or a later profile ([16] BRS-I Recipe), from devices not permanently attached to the platform ([17] number 19). |
| FIRM_100 | The firmware SHOULD support 64-bit Intel architecture (aka x64, aka AMD64) UEFI option ROM drivers for additional compatibility with the third-party IHV ecosystem. |
| *Since expansion cards for GPUs, High Speed NICs, etc. move faster than most platform vendors can integrate drivers into their platform firmware package (as well as those drivers making said firmware images extremely large), supporting UEFI Option ROM Drivers in x86_64 via emulation enables more hardware without having to wait for the platform vendor to port a drvier and ship it natively into their firmware. This is how Aarch64 systems solve the problem of no native drivers for the similar devices. The use of EFI Byte Code (EBC) is typically not used by hardware vendors because the compilers have not been available for some time and no open source compilers exist. Most add-in boards only ship x86_64 COFF EFI Drivers which are supported by github.com/tianocore/edk2-non-osi/tree/master/Emulator/X86EmulatorDxe if it's included in the EDK2 build.* | |
| FIRM_105 | If the firmware supports option ROMs, then it MUST support the ability to authenticate them ([17] number 19). |
| FIRM_110 | The firmware SHOULD support the ability to perform a HTTP-based boot from a network device, including support for HTTPS and DNS, supporting relevant HII protocols ([17] number 22). |

| ID# | Rule |
|---|---|
| FIRM_120 | The firmware MUST support software that runs from EFI firmware to install Load Option Variables (Boot####, or Driver####, or SysPrep####) consistent with [17] number 27. |
| FIRM_130 | The firmware MUST support software that runs from EFI firmware to register for notifications when a call to ResetSystem is called, consistent with [17] number 32. |
| FIRM_140 | IOMMU MUST be described using the RIMT ACPI table [18]. |
| FIRM_150 | If the firmware allows forward-edge control-flow integrity (FCFI) to be enabled for the supervisor execution environment, the runtime services MUST be compiled to support FCFI. |
| *The supervisor execution environment SHOULD enable FCFI through the SBI FWFT LANDING_PAD interface.* | |
| FIRM_160 | The support for forward-edge control-flow integrity in runtime services MUST be signaled by the EFI_MEMORY_ATTRIBUTES_FLAGS_RT_FORWARD_CONTROL_FLOW_GUARD flag ([5] Section 4.6.3 EFI_MEMORY_ATTRIBUTES_TABLE). |
| FIRM_170 | If the runtime services support forward-edge control-flow integrity, the instruction at the entry address of any runtime service MUST be a 4-byte aligned, unlabeled landing pad (`lpad 0`). |

# Chapter 4. Server Platform Security Rules

Security rules straddle hardware and firmware.

| ID# | Rule |
|---|---|
| SEC_010 | The server platform MUST implement a hardware Root of Trust (RoT) ([19]) as a dedicated and trusted subsystem, isolated from the application processor, to provide security-specific functions. |
| *A Root of Trust (RoT) is a component that performs one or more security-specific functions, such as measurement, storage, reporting, verification, update, security lifecycle management, and key derivation.*<br><br>*An RoT is typically a combination of a minimal amount of hardware and firmware that must be implicitly trusted by all system components to always behave as expected, since its misbehavior cannot be detected under normal operation.*<br><br>*A hardware RoT moves critical functions and assets off the application processor hart to a dedicated and isolated trusted subsystem, which provides stronger protection against both physical and logical attacks.*<br><br>*Examples of open-source RoT IPs include OpenTitan ([20]) and Caliptra ([21]).* | |
| SEC_020 | The hardware RoT MUST manage a security lifecycle. |
| *A security lifecycle reflects the trustworthiness of a system throughout its lifetime and indicates the lifecycle state of hardware-provisioned assets.*<br><br>*The minimum security lifecycle should include the following states:*<br><br>• *Manufacture – The system may not yet be locked down and contains no hardware-provisioned assets.*<br>• *Security Provisioning – The process of provisioning hardware-provisioned assets.*<br>• *Secured – Hardware-provisioned assets are locked (immutable); only authorized software may be executed, and revealing debug capabilities are disabled.*<br>• *Recoverable Debug – Part of the system is in a revealing debug state. The RoT remains uncompromised, and hardware-provisioned secrets remain protected.*<br>• *Terminated – Hardware-provisioned assets are permanently inaccessible and revoked prior to entering this state. This includes derived assets such as attestation keys.* | |
| SEC_030 | The hardware RoT SHOULD implement a secure identity and SHOULD support platform attestation. |

| ID# | Rule |
|---|---|
| *A **secure identity** is an element capable of generating a cryptographic signature that can be verified by a relying party. It represents the immutable part of the secure platform—such as immutable hardware, configurations, and firmware. Immutable components cannot be modified after the completion of security provisioning. See ([22]) for examples of secure identity derivation and use.*<br><br>***Attestation** is the process of vouching for the accuracy of information ([19]). Platform attestation enables a relying party to determine the trustworthiness of the platform before submitting sensitive assets to it. See ([23]) for an example of the protocols used for attestation.*<br><br>*The attestation must be signed by the hardware RoT using a hardware-provisioned secure identity or a cryptographic key derived in a verifiable manner from that identity.* | |
| SEC_040 | The firmware MUST implement UEFI Secure Boot and Driver Signing ([5] Section 32, "Secure Boot and Driver Signing") |
| SEC_050 | For systems that are not intended to be locked down, or that are intended to be locked down but have not been locked down yet, it MUST be possible for a physically present and/or strongly authenticated out-of-band management user to disable Secure Boot enforcement, thus allowing unsigned code to be executed. |
| SEC_060 | For systems that are not intended to be locked down, or that are intended to be locked down but have not been locked down yet, it MUST be possible for a physically present and/or strongly authenticated out-of-band management user to fully manage the contents of the PK, KEK, db and dbx Secure Boot key stores. This includes the ability to delete all factory-provided keys, enroll their own custom keys, and reset the key stores to their factory state. |
| *The term "locked down" refers to the (optional) ability to prevent the Secure Boot configuration from being modified further once the desired security lifecycle state has been reached. This could be implemented, for example, via an eFuse.*<br><br>*Note that the "locked down" state is distinct from the "Deployed Mode" Secure Boot state defined in the UEFI spec.*<br><br>*Being able to prevent even a physically present user from altering the Secure Boot configuration can be useful in the context of highly regulated industries or government bodies.* | |
| SEC_070 | The platform and firmware MUST back the UEFI Authenticated Variables implementation with a mechanism that cannot be accessed or tampered by an unauthorized software or hardware agent. |
| SEC_080 | The firmware MUST implement in-band firmware updates as per [16]. |
| SEC_090 | Firmware update payloads MUST be digitally signed. |
| SEC_100 | Firmware update signatures MUST be validated before being applied. |

| ID# | Rule |
| --- | --- |
| *Cryptographic algorithms and key sizes used for firmware update signing and verification should comply with the security and regulatory requirements of the geographies or markets in which the platform operates. Examples of relevant standards include the U.S. National Institute of Standards and Technology (NIST) cryptographic guidelines and the Commercial National Security Algorithm (CNSA) Suite.* | |
| SEC_110 | It MUST NOT be possible to bypass secure boot, authentication or digital signature failures, except as specified in SEC_050 and SEC_060. |

# Bibliography


[1] “Key words for use in RFCs to Indicate Requirement Levels.” [Online]. Available: datatracker.ietf.org/doc/html/rfc2119.

[2] “PCI Express® Base Specification Revision 6.0.” [Online]. Available: pcisig.com/pci-express-6.0-specification.

[3] “Advanced Configuration and Power Interface (ACPI) Specification.” [Online]. Available: uefi.org/specifications.

[4] “RISC-V Supervisor Binary Interface Specification v3.0.” [Online]. Available: github.com/riscv-non-isa/riscv-sbi-doc.

[5] “Unified Extensible Firmware Interface.” [Online]. Available: uefi.org/specifications.

[6] “RVA23 Profiles.” [Online]. Available: github.com/riscv/riscv-profiles.

[7] “RISC-V Debug Specification v1.0.” [Online]. Available: github.com/riscv/riscv-debug-spec.

[8] “RISC-V Advanced Interrupt Architecture (AIA) v1.0.” [Online]. Available: github.com/riscv/riscv-aia.

[9] “RISC-V Control Transfer Records (CTR) v1.0.” [Online]. Available: github.com/riscv/riscv-control-transfer-records.

[10] “RISC-V Server SoC Specification v1.0.” [Online]. Available: github.com/riscv-non-isa/server-soc.

[11] “National Semiconductor PC16550D UART Datasheet.” [Online]. Available: www.scs.stanford.edu/10wi-cs140/pintos/specs/pc16550d.pdf.

[12] “PrimeCell UART (PL011) Technical Reference Manual.” [Online]. Available: developer.arm.com/documentation/ddi0183/latest/.

[13] “eXtensible Host Controller Interface for Universal Serial Bus 1.2.” [Online]. Available: www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/extensible-host-controler-interface-usb-xhci.pdf.

[14] “Advanced Host Controller Interface (AHCI).” [Online]. Available: www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/serial-ata-ahci-spec-rev1-3-1.pdf.

[15] “TPM 2.0 Library.” [Online]. Available: trustedcomputinggroup.org/resource/tpm-library-specification/.

[16] “RISC-V Boot and Runtime Services Specification v1.0.” [Online]. Available: github.com/riscv-non-isa/riscv-brs.

[17] “Unified Extensible Firmware Interface, 2.6.2 ‘Platform-Specific Elements.’” [Online]. Available: uefi.org/specifications.

[18] “RISC-V IO Mapping Table v1.0.” [Online]. Available: github.com/riscv-non-isa/riscv-acpi-rimt.

[19] “TCG Glossary.” [Online]. Available: trustedcomputinggroup.org/resource/tcg-glossary/.

[20] “OpenTitan.” [Online]. Available: opentitan.org/.

[21] “Caliptra.” [Online]. Available: github.com/chipsalliance/Caliptra.

[22] “DICE Attestation Architecture.” [Online]. Available: trustedcomputinggroup.org/work-groups/dice-architectures/.

[23] “DSP0274: Security Protocol and Data Model (SPDM) Specification.” [Online]. Available: www.dmtf.org/standards/spdm.