Standardizing CHERI-RISC-V, CHERI TG specification and status update

Tariq Kurd, Chief Architect

Codasip

→ RISC-V is about to have new base architectures

New unratified CHERI base architectures are:

- RV64Y – 32 registers, 64-bit, 128-bit capabilities
- RV32Y – 32 registers, 32-bit, 64-bit capabilities
- CHERIoT will define a new base architecture which will inherit from RV32Y
- All mnemonics will be renamed to start with Y and the data type will be Y e.g. LY/SY

Current ratified base architectures are:

- RV64I – 32 registers, 64-bit
- RV64E – 16 registers, 64-bit
- RV32I – 32 registers, 32-bit
- RV32E – 16 registers, 32-bit

The other unratified base architecture is:

- RV128I – 32 registers, 128-bit

→ Documentation status

- The current CHERI spec 0.9.5 on GitHub is 278 pages
  ~ 170 are instruction pages
- All the features have been in one document and are currently being split into different pieces
  - Unprivileged architecture – RV64CH, RV32CH
  - Privileged architecture – Smcheri, Sscheri
  - Debug architecture – Sdcheri

→ Being a base architecture is useful

It means that we can choose what we import from RV64I/ RV32I. In particular we can ditch the fairly useless double precision floating point load/store encodings and have useful ones instead.

RV32I/ RV64I	Description	RV32CH/ RV64CH	Description
C.FSD	Store double via rs1	C.SC	Store capability via cs1
C.FLD	Load double via rs1	C.LC	Load capability via cs1
C.FSDSP	Store double via stack pointer	C.SCSP	Store capability via stack pointer
C.FLDSP	Load double via stack pointer	C.LCSP	Load capability via stack pointer

→ Unprivileged Architecture RV32Y/RV64Y

- This is the definition of the pure-capability machine
- 32 CLEN-wide registers (extended from XLEN-wide)
- All CHERI instructions are listed in the base architecture
  - E.g. load/store capability LY/SY
  - Capability manipulation
- Application Code programmer's view
- No hybrid features
  - DDC, mode switch

→ Privileged Archiecture Smcheri/Sscheri

- This includes all pure capability machine CSRs visible from M or S-mode
  - E.g. extra bits in Xenvcrg, mseccfg
- Virtual memory changes
  - PTE.CW – to restrict capability writes
  - PTE.UCRG – separate extension Svcheri (changed from Svucrg in 0.9.5)
- CHERI exception types and handling
  - Usage of Xtval2 for example
- Hybrid mode as an option on the base RISC-V architecture
- Updated debug specification

→ What extensions do we have?

	Extension	Status	Description
0.9.5	ZcheriPurecap	Stable – bug fixes only	Base architecture for a CHERI purecap machine
	ZcheriHybrid	Stable – bug fixes only	Implies ZcheriPureCap. Adds legacy RISC-V support
	Zish4add	Stable – bug fixes only	SH4ADD as the datatype is 128-bit (RV64 only)
	Zabhlrsc	Stable – bug fixes only	Byte/half LR/SC support (independent of CHERI)
	Svucrg	Stable – in software prototyping	Optimised **Revocation** support by supporting capability **accessed** and **dirty** in page tables
	Zstid	Stable – in software prototyping	Secure thread ID for **Compartmentalisation**
	ZcheriLevels	Stable – in software prototyping	Support for locally/globally accessible capabilities with multiple levels
Future work	ZcheriVector	Research, Need PR	CHERI and **Vector** optimised support to allow Vector capability **memcpy**
	ZcheriTraceTag	Research, Need PR	Support for data capability trace with tags
	ZcheriSanitary	Research, Need PR	Support for cleaning capabilities on compartment switching
	ZcheriSystem	Research, Need PR	Support for exposing compartment IDs to the system (a better WorldGuard)

→ Thanks to the contributors!

Writers of the specification

Codasip Google University of Cambridge

Reviewers of the specification

RISC-V® SCI Semiconductor lowRISC

→ Ratification process key milestones

• **Freeze**	Jun 15, 2025
• **Public Review Start**	Jun 16, 2025
• **TSC Ratification Approval**	Aug 27, 2025
• **BoD Ratification Approval**	Aug 28, 2025