Let's make a standard for CHERI-RISC-V

Tariq Kurd

Codasip

→ History of CHERI

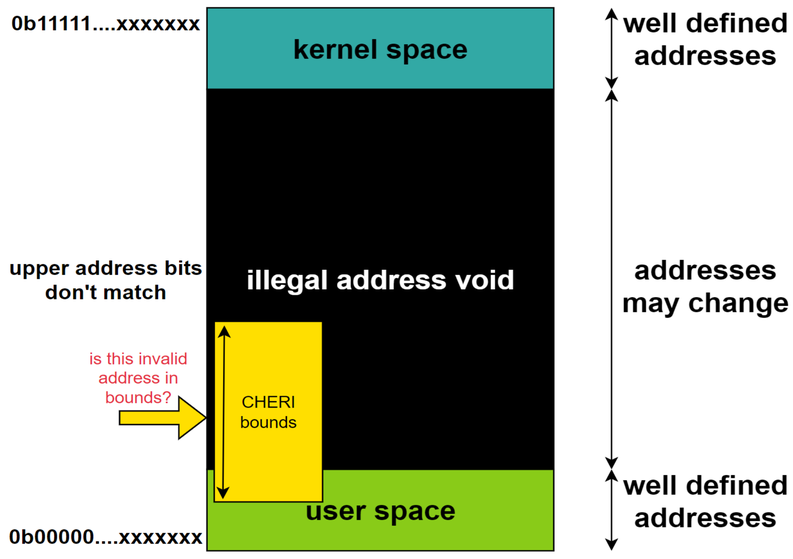
Year	Version	Description
2010-2012	**ISAv1**	**RISC capability-system model w/64-bit MIPS** Capability registers, tagged memory Guarded manipulation of registers
2012	**ISAv2**	Extended tagging to capability registers Capability-aware exception handling **Boots an MMU-based OS with CHERI support**
2014	**ISAv3**	**Fat pointers + capabilities, compiler support** Instructions to optimize hybrid code Sealed capabilities, CCall/CReturn
2015	**ISAv4**	MMU-CHERI integration (TLB permissions) **ISA support for compressed 128-bit capabilities** HW-accelerated domain switching Multicore instructions: full suite of LL/SC variants
2016	**ISAv5**	CHERI-128 compressed capability model Improved generated code efficiency **Initial in-kernel privilege limitations**
2017	**ISAv6**	Mature kernel privilege limitations Further generated code efficiency Architectural portability: CHERI-x86 and **CHERI-RISC-V** sketches **Exception-free domain transition**
2019	**ISAv7**	64-bit capabilities for 32-bit architectures **Elaborated draft CHERI-RISC-V ISA** Architectural performance optimization for C++ applications **Temporal memory safety** Microarchitectural side-channel resistance features

→ The future

All new Codasip cores will have CHERI

CHERI world domination – all cores memory safe..?

→ Invalid address handling: the problem



Invalid addresses may change when written to registers such as MEPC for Sv39 or Sv48

Is the new address in bounds or not? Do we need to check? It's not cheap to do so

→ From CHERI v8 to CHERI-RISC-V

- CHERI v8 imported into CHERI v9 repo in April 2022
- CHERI v9 changes
  - CHERI-MIPS removed
  - CHERI-RISC-V now the base architecture
  - CHERI-x86 sketch added
- And of course ARMv8 – Morello – also exists

- CHERI-RISC-V changes in CHERI v9
  - Merged register-file only, split option removed
    - This was legacy from MIPs
  - Removal of DDC, PCC offsetting
  - Add CGetHigh, CSetHigh for capability creation and querying
  - Add per-privilege enables into menvfg, senvcfg CSRs
  - Moving to tag clearing to reduce exception sources

→ From CHERI-RISC-V v9 to the Codasip Demo

At Codasip we worked independently to fill in gaps in the specification to allow a product to be built

1. There was no CHERI-RISC-V debug specification (Sdtrig/Sdext)
2. Not all mnemonics were clearly specified
   A. E.g. did c.j map to c.cj in capability mode? The semantics don't change....
   B. Missing encodings for 16-bit instructions....
3. Merging the exception priorities with the standard RISC-V ones
4. And various other changes

→ Getting to the RISC-V Github repo

- Started working with Cambridge University on the CHERI-RISC-V specification after a discussion at the RISC-V Summit in Barcelona in June 2023
- We were already working in the background on a different version of the CHERI specification document
  - Extracting well defined features from CHERI v9
    - Postponing experimental and less well-defined features
    - Defining a stable base architecture
  - Written as an implementation spec
  - Covers all the necessary questions asked by the implementation and verification teams to allow the product to be built
- We tested this spec on our A730-CHERI core development

January 2024 — After review with Cambridge Uni, the Codasip CHERI spec document became v0.7.0 on Github

Then the real specification work started refining the architecture

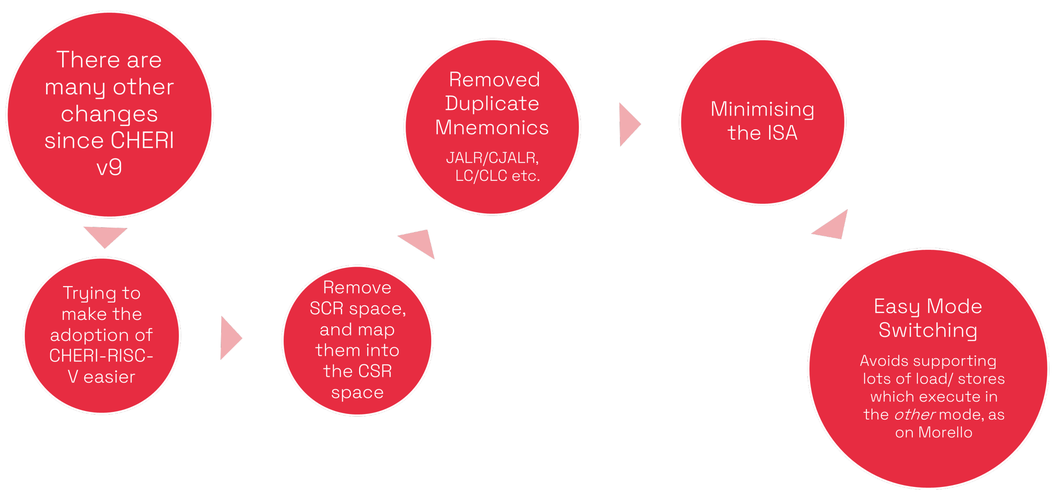
→ Illegal address handling: the solution

A new CHERI exception type

- For running software in a CHERI compartment only
  - Take an invalid address CHERI exception, so we don't care if the address is representable or in bounds or not
- Existing RISC-V code on a CHERI core still takes an access fault

Reduces the size of the bounds comparators

- Previously CHERI required full 64-bit address comparators
  - Now we need 39-bit for Sv39, 48-bit for Sv48
  - This gives a nice power and area saving, and is simpler
- This also compatible with pointer masking as we don't need to compare the masked range of the address



Extension	Status	Description
ZcheriPurecap	Stable – bug fixes only	Base architecture for a CHERI purecap machine
ZcheriHybrid	Stable – bug fixes only	Implies ZcheriPureCap. Adds legacy RISC-V support
ZabhlrscStable – bug fixes only		Byte/half LR/SC support (independent of CHERI)
Zstid	Software prototyping	Secure thread ID for **Compartmentalisation**
ZcheriHypervisor	Prototype, need PR	CHERI and **Hypervisor** support
ZcheriVector	Prototype, need PR	CHERI and **Vector** optimised support to allow Vector capability **memcpy**
ZcheriPTE	Prototype, PR needs update	Optimised **Revocation** support by supporting capability **accessed** and **dirty** in page tables
ZcheriTransitive	Prototype, PR needs update	Support for **reducing capability permissions** on loading
ZcheriMultiLevel	Research, need PR	Support for locally/globally accessible capabilities with multiple levels
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)

→ RV32: Format

The RV32 format poses challenges due to limited encoding space

- CHERI v9 has
  - 12-bit permissions
  - 8-bit mantissa (encoded as 8 for Base, 6 for Top) – 14-bits
  - 1-bit flag (Mode)
  - 4-bit Otype
  - 1-bit Internal Exponent flag

The final encoding has:

- 2 software defined permissions
- 5 architectural permissions (6 including the Mode bit)
  - With space for more to be added
- 4 reserved bits (for local/global?)
- 1 sealed bit
- T8 gives an extra mantissa bit when the exp is zero, or a 5-bit exp field

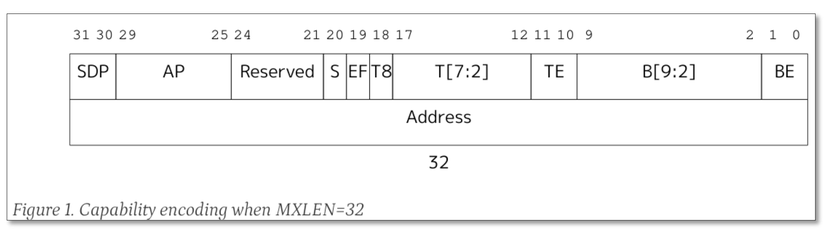


Figure 1. Capability encoding when MXLEN=32

Find out more