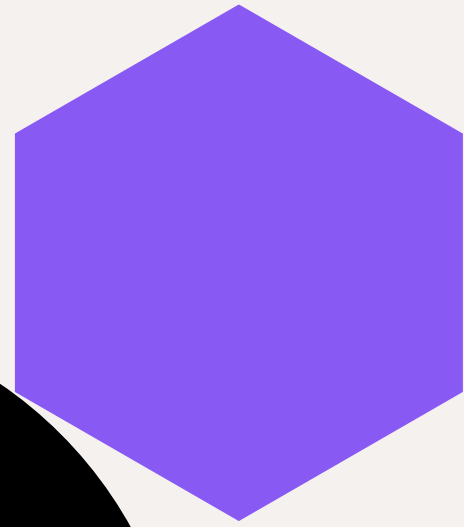
Let’s make a standard for CHERI-RISC-V



To make memory safety available for everyone, from small cores to high performance

Tariq Kurd, Chief Architect at Codasip

RISC-V Summit Munich, June 2024

→ From CHERI v8 to CHERI-RISC-V

Codasip

- 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
  - This is not a complete list…

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

Codasip

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

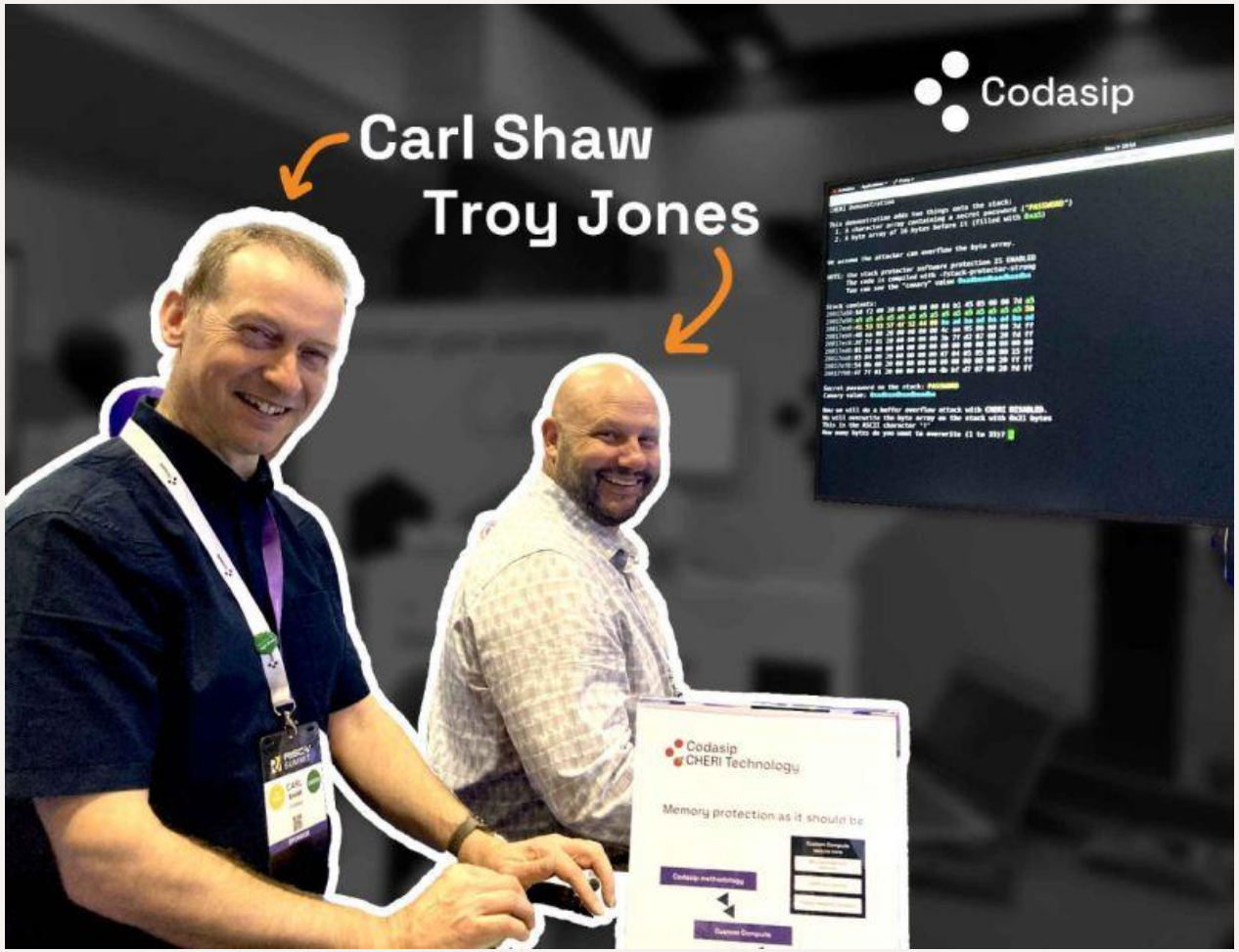
1. There was no CHERI-RISC-V debug specification (Sdtrig/Sdext)
2. Not all mnemonics were clearly specified
    1. *E.g. did c.j map to c.cj in capability mode? The semantics don't change….*
    2. *Missing encodings for 16-bit instructions….*
3. Merging the exception priorities with the standard RISC-V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

→ We demonstrated the result of this development at the RISC-V Summit in Santa Clara, November 2023

→ The Demo at the RISC-V Summit Santa Clara Nov ‘23

Codasip

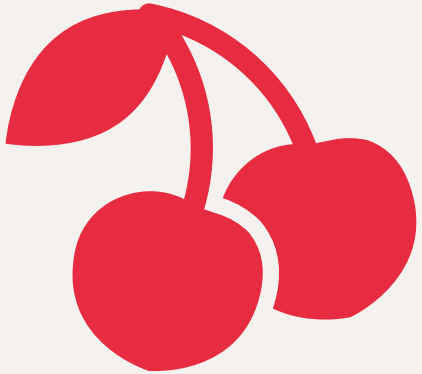
Carl Shaw and Troy Jones showing the A730-CHERI prototype detecting a buffer overrun on the stack before the stack canary spotted it



→ Getting to the RISC-V Github repo

Codasip

- 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



→ CHERI-RISC-V v0.7.0

Codasip

January 2024

After review with Cambridge Uni, the Codasip CHERI spec document became v0.7.0 on Github:
https://github.com/riscv/riscv-cheri/releases/

The Task Group was formed

Then the real specification work started refining the architecture

→ RV32: Old Capability Format

Codasip

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

A few observations

- That’s a lot of permission bits
- Otype has been deferred
- The mantissa is too short for accurate bounds
- Mode doesn’t need a whole bit, it’s only relevant for executable capabities
- We want space for future expansion
- No software defined permissions

So, let’s rework the format

→ RV32: New Format

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

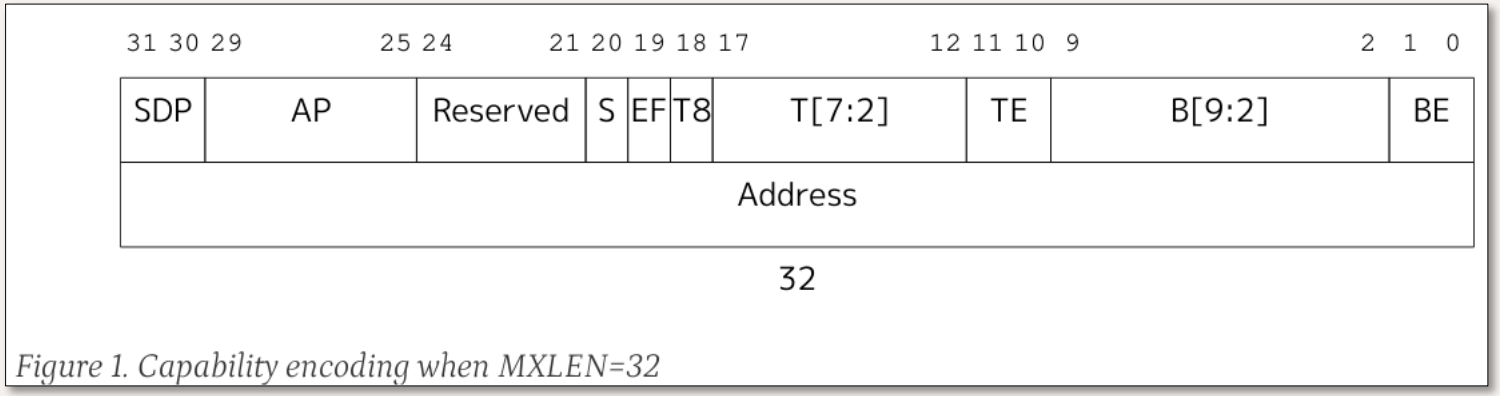
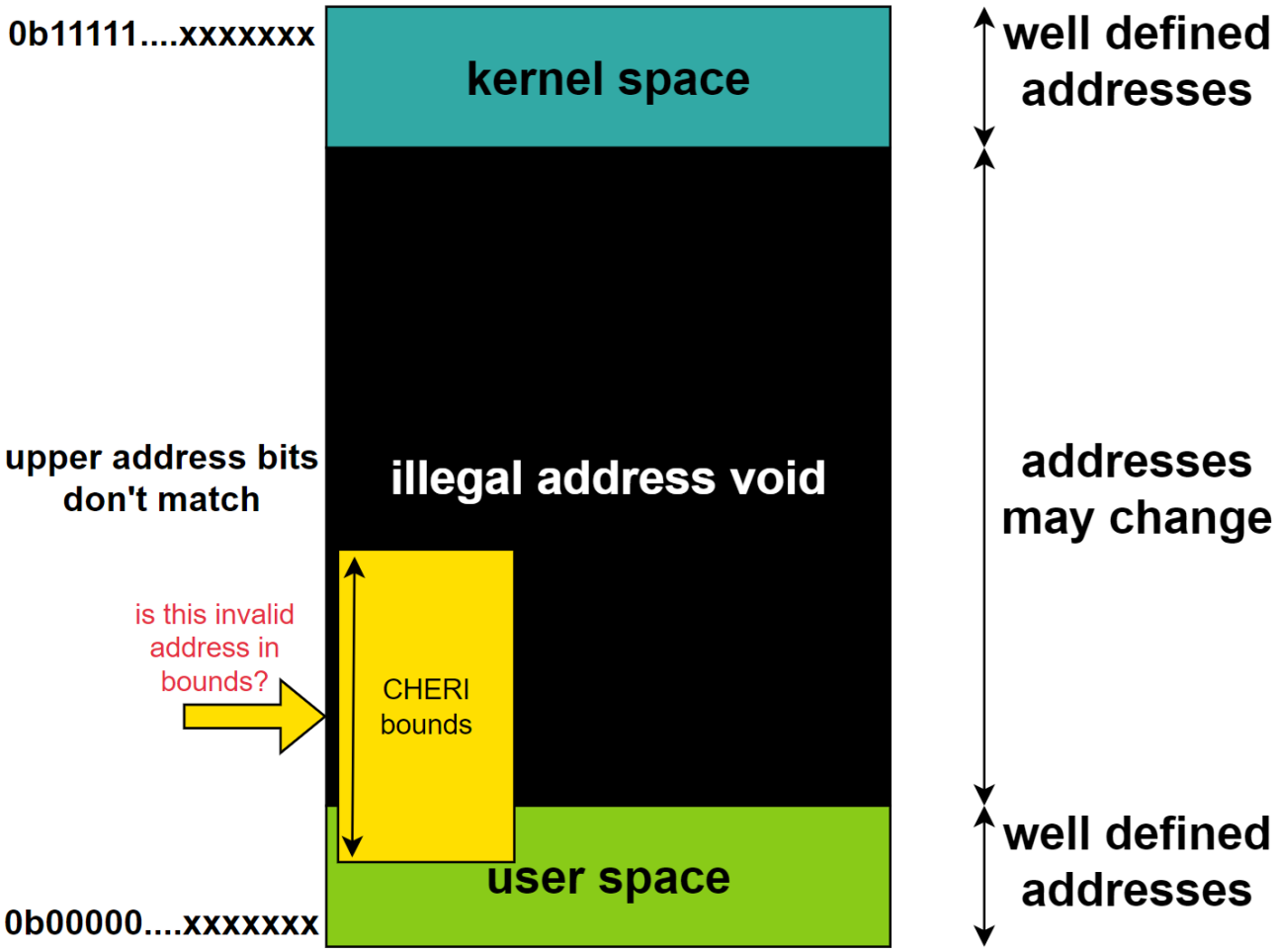


Figure 1. Capability encoding when MXLEN=32

→ 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.

→ Illegal address handling: the solution

Codasip

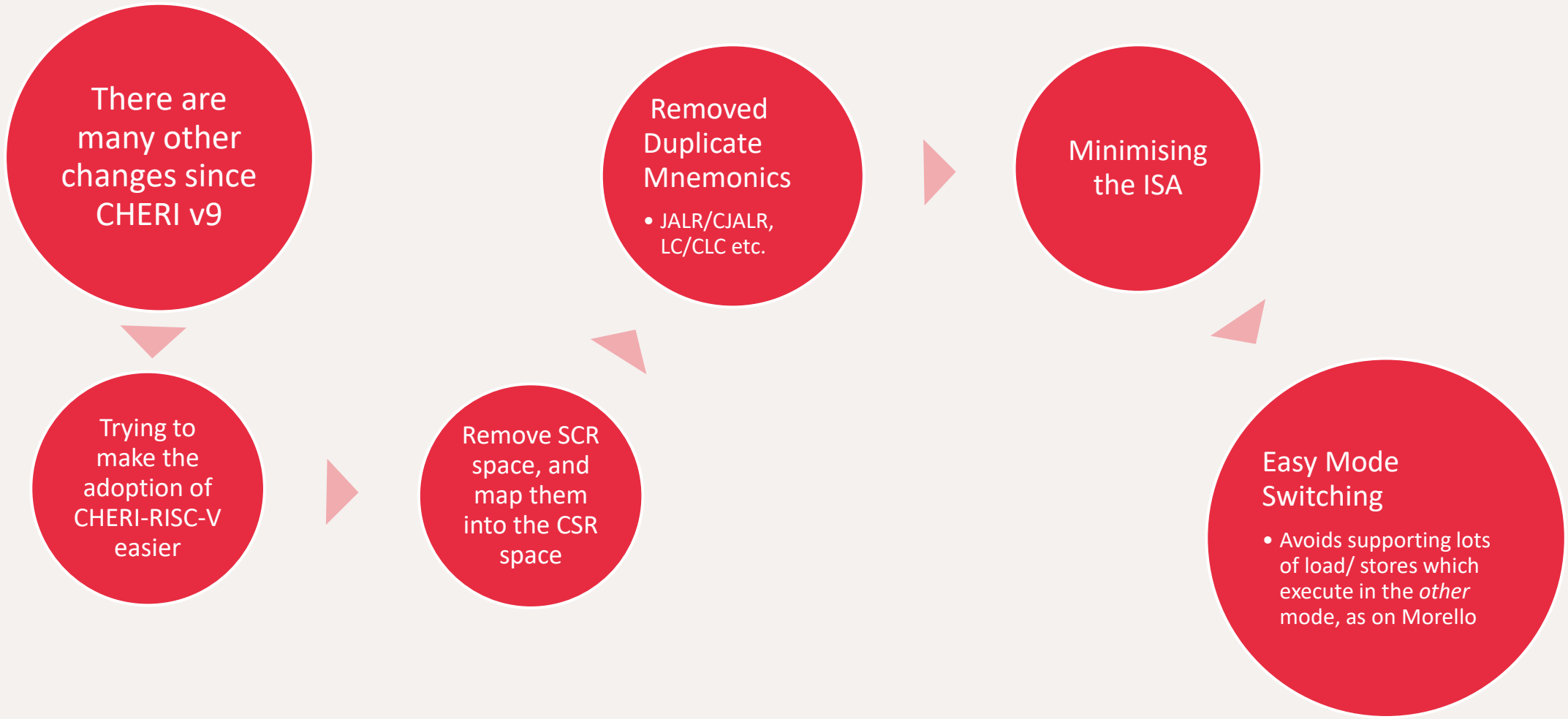
A new CHERI exception type

- For running CHERI software only
  - Take an invalid address CHERI exception, so we don’t care if the address is representable or in bounds or not
- Legacy 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

→ Other changes?



→ What extensions do we have?

Codasip

Extension	Status	Description
ZcheriPurecap	Stable – bug fixes only	Base architecture for a CHERI purecap machine
ZcheriLegacy	Stable – bug fixes only	Implies ZcheriPureCap. Adds legacy RISC-V support
Zabhlrsc	Stable – 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)

→ Getting to RVA23 Compatibility

Codasip

We're currently have RVA22+CHERI fully working for *mandatory* extensions. We want to get to RVA23+CHERI, which has some gaps to fill:

- **Vector+CHERI: Fairly simple – check every unmasked byte of every load/store against the bounds**
  - But the devil is in the details for complex sequenced masked load/stores
  - Indexed loads where the base is zero and the entire address is in the element are a problem: the capability will need to have the address field set to zero, and so the bounds must start at zero
  - Consider adding VLC/VSC to load/store full vector registers including caps, and an associated VCMV to move a whole vector register to support Vector capability memcpy
- **Vector+Hypervisor: more on the next talk**
- **Pointer masking: seems easy – needs confirmation**

→ Code Size Reduction

Codasip

→Already in the CHERI-RISC-V spec

Zcmt – table jump

- JVT becomes a capability

Zcmp – push/pop

- The data-width doubles, and only RV32 is of interest, so effectively use the RV64 stack layout for RV32-CHERI-RISC-V

→ Conclusions

Codasip

CHERI is 100% compatible with RVA22 mandatory extensions

Will soon be with RVA23 too

CHERI runs existing RISC-V code with 100% compatibility

This makes adoption of CHERI much easier

We’re working on CHERIoT compatibility

We want one ratified specification to cover all variants

→ The future….

Codasip

All new Codasip cores will have CHERI

CHERI world domination – all cores memory safe..?

Codasip

↘

That’s all folks

- Collaborate with us: https://github.com/riscv/riscv-cheri
- Join the TG and the bi-weekly meeting
- Tell us what’s missing and help fill in the gaps
- Help drive CHERI to world domination