CHERI SIG - Towards a standard CHERI-RISC-V

Chair: Alex Richardson (alexrichardson@google.com)
Vice-chair: Simon Moore (simon.moore@cl.cam.ac.uk)
Poster by Peter Rugg (peter.rugg@cl.cam.ac.uk)
CHERI Project URL: cheri-cpu.org

UNIVERSITY OF CAMBRIDGE
Computer Science & Technology

CHERI

Background

CHERI Architecture

CHERI is an architectural extension to add memory safety and compartmentalisation to an ISA by extending pointers with bounds and permission metadata to form an unforgeable *capability*. The hardware can then enforce the principles of *intentionality* and *least privilege*, deterministically mitigating entire vulnerability classes.

To protect itself, the intent of a program must be preserved all the way down to the machine code. This can be achieved by the compiler, e.g. for stack allocations: LLVM has been modified to perform these transformations automatically. Other software and libraries can also be modified, e.g. specifying sizes of dynamic allocations.

To support CHERI, a processor is augmented to support capabilities in registers, instructions to monotonically reduce capability privilege, checks on memory accesses, and memory tags for pointer integrity. Implementation of CHERI has been validated to incur modest hardware and software overheads, e.g. in Arm's Morello prototype.

RISC-V

RISC-V is a RISC ISA available for free use in both open and commercial implementations. Vendors can pick and choose "standard" extensions, such as "C" for compressed instructions and "V" for vector instructions, or implement their own custom extensions in reserved opcode space.

Unlike Morello, the possibility of open RTL makes RISC-V ideal for architectural research, allowing the community to experiment with CHERI designs in hardware.

RISC-V®

Objective

The RISC-V CHERI Special Interest Group aims to define a standard CHERI extension to the RISC-V ISA. This will encourage further development of capability hardware and software ecosystems, paving the way for always-on spatial- and temporal-safety by default.

Initial efforts target a "minimum viable" CHERI specification, reducing friction to adopt capabilities into the architecture, and allowing low-overhead hardware support. This minimal CHERI would support 64-bit general purpose operating systems and 32-bit vertically integrated software stacks.

There is already a strong ecosystem, without a standard to guarantee interoperability:

- RTL designs:
  - Microcontrollers: CHERIoT (Microsoft), CHERI Ibex, CHERI Flute
  - Out-of-order application class: CHERI Toooba
  - Codasip commercial implementation (just announced!)
- Compiler:
  - Full capability support in RISC-V LLVM
- OS support:
  - CheriBSD (based on FreeBSD)
  - CHERIoT RTOS (Microsoft)
  - CHERI FreeRTOS
  - CheriOS (clean-slate single address space)

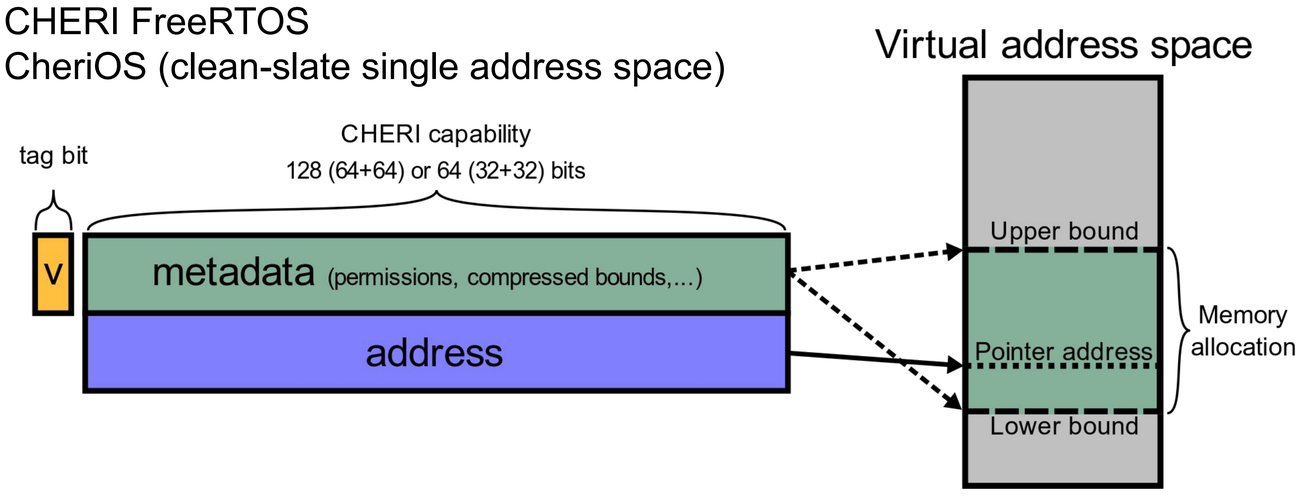

Figure 1. A (pre-standardisation) CHERI capability fitting bounds and permissions into 64 bits of metadata. Implementations need to agree on the layout.

Features to standardise

- Encoding of 128-bit capabilities (64-bit RISC-V) and 64-bit capabilities (32-bit RISC-V)
- Capability arithmetic instructions
- Capability checks on memory access
- Tagged memory
- Exception model: new and extended special registers
- Support for temporal safety
- Support for efficient sub-object bounds
- "Sentry" code-pointer sealing
- Extensibility for additional features

Deferred features

To minimise the complexity of an initial CHERI specification, some features are deferred to custom or future standard extensions pending further research:

- Code/data pair sealing using object types
- Memory access offsetting using PCC and DDC
- ECC-based memory tags (e.g. as in Morello)
- …

Specification tasks

- Quantitative architecture evaluation:
  - Instruction frequencies to inform included instructions, immediate sizes, …
  - Characterise capability register usage
- Tests and verification:
  - Existing TestRIG infrastructure
  - Integrate with RISC-V compliance test suite
- Define and optimise capability format:
  - Decide bounds precision
  - Efficiently encode permissions
- Specify interactions with other extensions
- Design for future extensibility
- Write a spec! (suitable for RISC-V committees)

RISC-V members can join the SIG: https://lists.riscv.org/g/sig-cheri