CHERI-RISC-V SIG - an extension for memory safety

Alex Richardson, Simon W. Moore, Robert N. M. Watson, Jessica Clarke, Brooks Davis, Lawrence Esswood, Ben Laurie, Peter Rugg, Alexandre Joannou, and Peter Sewell
Department of Computer Science and Technology, University of Cambridge, Google and SRI International

Contact: aj443@cam.ac.uk

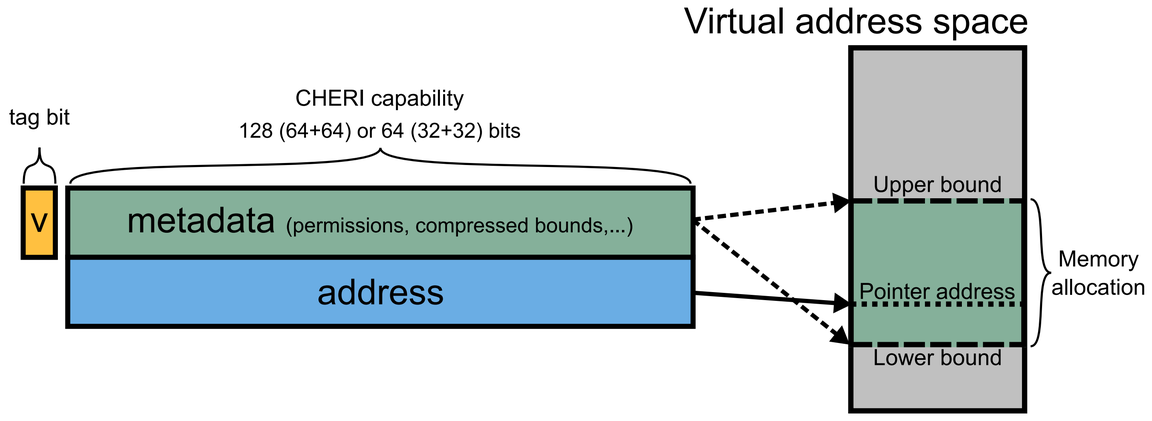
Project URL: cheri-cpu.org

UNIVERSITY OF CAMBRIDGE
Computer Science & Technology

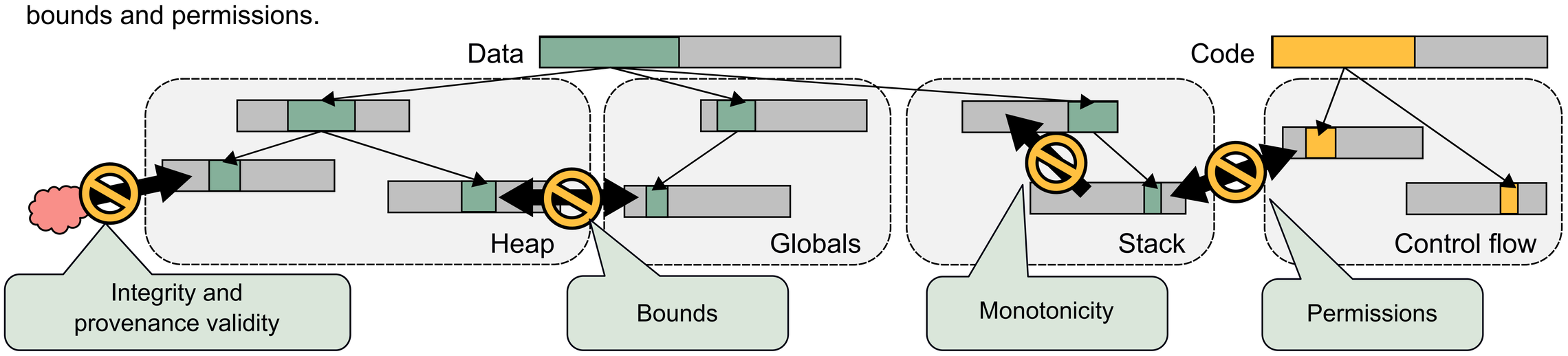
The CHERI architecture

With CHERI, pointers to data and code are replaced by CHERI capabilities that include bounds and permissions to guarantee spatial memory safety, and contain a hidden validity tag that guarantees integrity and provenance.

- CHERI capabilities (with their metadata) are stored in an extended integer register file and in memory with regular data. Capability integrity is guaranteed with a tag bit per capability sized word (stored out of band) which is atomically cleared on capability overwrite (no forging of capabilities or invalid capability manipulation).
- Compiler and linker support allows the majority of C/C++ code to be recompiled without code change to gain strong spatial safety. Temporal memory safety can be enforced as well with software support.



- The CHERI extended ISA allows user space to access data using capabilities, with bounds and permission violations raising an exception on memory access. New instructions allow the compiler/programmer to derive new capabilities but only with reduced bounds and permissions.



The CHERI-RISC-V extension

CHERI aspects under consideration for ratification

- CHERI features to support safe, capability-aware exception handling (supervisor/machine mode support)
- CHERI features to support efficient temporal memory safety
- Tagging behavior for memory
- CHERI features to support sealed entry (sentry) protection of control-flow pointers
- CHERI features to support C/C++ memory protection including subobject bounds and efficient stack alignment
- 128-bit capabilities over a 64-bit baseline ISA, and 64-bit capabilities over a 32-bit baseline ISA, with a specific architectural format and a clear path to future expansion
- Opcode assignments for instructions implied by the above

Idx	Register	Modes	Access	Reset	Extends
0	Program counter capability (**PCC**)	U, S, M	RO	∞	**PC**
1	Default data capability (**DDC**)	U, S, M	-	∞	-
4	User trap code capability (**UTCC**)	U, S, M	ASR	∞	utvec
5	User trap data capability (**UTDC**)	U, S, M	ASR	∅	-
6	User scratch capability (**UScratchC**)	U, S, M	ASR	∅	-
7	User exception PC capability (**UEPCC**)	U, S, M	ASR	∞	uepc
12	Supervisor trap code capability (**STCC**)	S, M	ASR	∞	stvec
13	Supervisor trap data capability (**STDC**)	S, M	ASR	∅	-
14	Supervisor scratch capability (**SScratchC**)	S, M	ASR	∅	-
15	Supervisor exception PC capability (**SEPCC**)	S, M	ASR	∞	sepc
28	Machine trap code capability (**MTCC**)	M	ASR	∞	mtvec
29	Machine trap data capability (**MTDC**)	M	ASR	∅	-
30	Machine scratch capability (**MScratchC**)	M	ASR	∅	-
31	Machine exception PC capability (**MEPCC**)	M	ASR	∞	mepc

Table 1. New CHERI Special Capability Registers

Initially deferred CHERI aspects

- Sealed capability pairs with object types
- The use of DRAM metadata / ECC bit-stealing or CHERI tag cache / controllers
- Relocation / offsetting PCC (Program Counter Capability) and DDC (Default Data Capability) - initially, PCC / DDC will only perform bounds & permission checks
- One vs. multiple capability roots and formats (the proposed standardization effort would introduce only memory capabilities)
- As-user CHERI memory instructions
- Specialist capability conversion instructions (CFromPtr and CToPtr) used to optimize certain types of pointer manipulation in Hybrid C; these are likely to be deprecated in CHERI
- MMU support for page capability-dirty tracking
- MMU support for load-barrier temporal memory safety
- Morello-specific domain-transition mechanisms such as executive/ restricted modes and indirect sealed jumps
- Local-global mechanisms