Sail, RISC-V, and CHERI-RISC-V

Prashanth Mundkur and Peter G. Neumann, SRI International

(most of this work done by University of Cambridge)

Robert Norton-Wright, Jon French, Brian Campbell*, Alasdair Armstrong, Thomas Bauereiss, Shaked Flur, Peter Sewell

University of Cambridge (*University of Edinburgh)

Ninth Summer School on Formal Techniques, May 23, 2019

Menlo College, Atherton, CA

This work was partially supported by EPSRC grant EP/K008528/1 (REMS), an ARM iCASE award, and EPSRC IAA KTF funding. Approved for public release; distribution is unlimited. This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contracts FA8750-10-C-0237 ("CTSRD") and FA8650-18-C-7809 ("CIFV"). The views, opinions, and/or findings contained in this article/presentation are those of the author(s)/presenter(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

ISA Specification

The problem:

- ISA specifications use a mixture of prose and pseudocode
- Often *many* thousands of pages
- Sometimes loosely worded and containing errors

Without machine-readable specifications

- Cannot do machine-checked proofs
- Hard to test or formally verify implementations against specification

Existing Formal ISA Models

- CakeML - HOL models for x86-64, ARMv6, ARMv8, RISCV-64, MIPS-64
- CompCert - Coq models for PowerPC, ARM, x86, RISC-V (32- and 64-bit)
- seL4 - Isabelle/HOL ARMv7 model
- ACL2 (x86) - Goel et al
- RockSalt SFI - Coq model of x86 (Morrisett et al)
- ... and others

Existing Formal ISA Models

- CakeML - HOL models for x86-64, ARMv6, ARMv8, RISCV-64, MIPS-64
- CompCert - Coq models for PowerPC, ARM, x86, RISC-V (32- and 64-bit)
- seL4 - Isabelle/HOL ARMv7 model
- ACL2 (x86) - Goel et al
- RockSalt SFI - Coq model of x86 (Morrisett et al)
- ... and others
- Public release of ARMv8-A specification by ARM

Existing Formal ISA Models

- CakeML - HOL models for x86-64, ARMv6, ARMv8, RISCV-64, MIPS-64
- CompCert - Coq models for PowerPC, ARM, x86, RISC-V (32- and 64-bit)
- seL4 - Isabelle/HOL ARMv7 model
- ACL2 (x86) - Goel et al
- RockSalt SFI - Coq model of x86 (Morrisett et al)
- ... and others
- Public release of ARMv8-A specification by ARM

  but no *public* tool support

Existing Formal ISA Models

- CakeML - HOL models for x86-64, ARMv6, ARMv8, RISCV-64, MIPS-64
- CompCert - Coq models for PowerPC, ARM, x86, RISC-V (32- and 64-bit)
- seL4 - Isabelle/HOL ARMv7 model
- ACL2 (x86) - Goel et al
- RockSalt SFI - Coq model of x86 (Morrisett et al)
- ... and others
- Public release of ARMv8-A specification by ARM

  but no *public* tool support

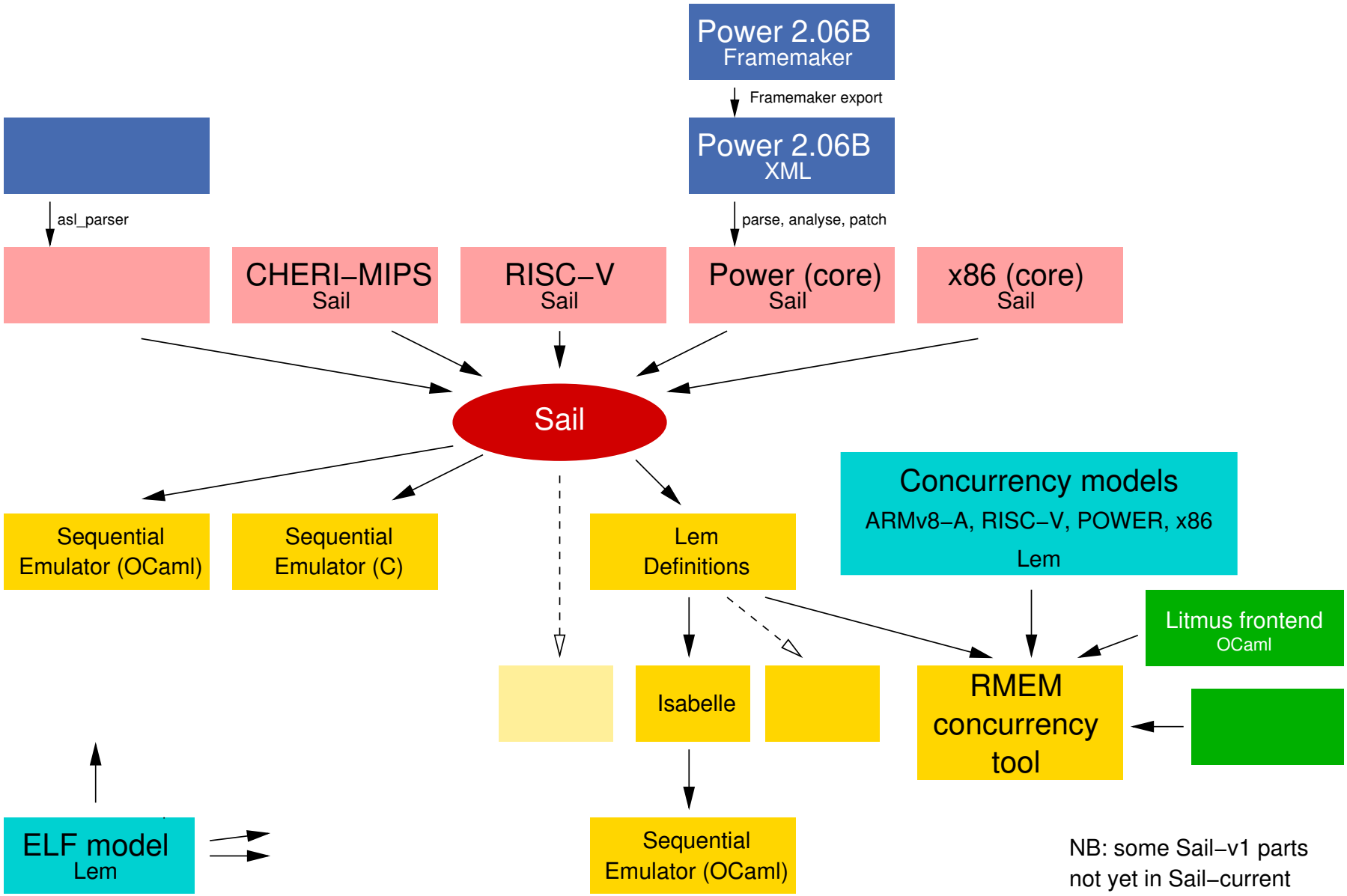
- Few include full system-level specifications
- Tied to specific use-cases or theorem provers

Sail design goals

ISA models which are:

- similar to existing pseudocode
- cover the full scope of the architecture
- translatable into executable sequential emulator code
- translatable into idiomatic theorem prover definitions
  - For multiple provers!
- offer fine-grained execution information for relaxed-memory model integration
- be well-validated

Sail Overview



Sail Models

Architecture	LOS	Boots	Generates	
ARMv8.3-A	23 000		C, OCaml	Isabelle, HOL4
ARMv8.5-A	100 000	Linux	C, OCaml	
RISC-V	5 000	seL4, Linux, FreeBSD	C, OCaml	Isabelle, HOL4, Coq
MIPS	2 000	FreeBSD	C, OCaml	Isabelle, HOL4, Coq
CHERI-MIPS	4 000	FreeBSD	C, OCaml	Isabelle, HOL4

ARM model generated from ARM ASL, other models hand-written

RISC-V

Open ISA, developed by broad industrial and academic community

- Test system features by booting seL4, FreeBSD and Linux
- Validated against RISC-V test suite, and via trace comparison with Spike simulator
- Led to contributions to original ISA specification, e.g.
  - description of page-faults in page-table walks
  - ambiguities in the specification of interrupt delegation
  - bug fixes in Spike simulator
- Integration with RMEM concurrency tool
  - Used with the 6874 litmus tests for the RISC-V memory model

MIPS and CHERI-MIPS

CHERI: Research architecture that extends 64-bit MIPS with hardware capabilities for fine-grained memory protection and secure compartmentalisation

The Sail model:

- Sufficient privileged architecture features to boot FreeBSD, but excluding floating-point and other optional extensions
- Under continuous development with CHERI project
- Owned and developed by hardware researchers
- Used in upcoming CHERI ISA specification document

Successful example of hardware/software/semantics codesign

The Sail Language

- Imperative first-order language for describing ISA specifications
- Lightweight dependent types
  - Purely syntax directed bi-directional approach
  - Prove important properties for MiniSail fragment:
    - Type safety
    - Decidability of type checking
  - SMT solver to make dependent typechecking mostly automatic

As simple as possible, but no simpler

Emulator Generation

Need reasonably efficient emulator generation for ISA validation
Simple OCaml translation, optimised C translation for speed
Key optimisations:

- Use dependent types and SMT to pack integers into 64-bit machine words
- Similarly, identify bitvectors that can be packed into single 64-bit words
- Statically allocate all storage where possible

1M IPS for MIPS, 80 000 IPS for ARM

Generating Theorem Prover Definitions

- Currently targeting Isabelle/HOL, HOL4, and Coq
- State monad for sequential reasoning
- Free monad over memory effects for concurrent reasoning
- Use dependent type information to integrate with machine word libraries
- Validation of translation via testing
  - Code extraction from Isabelle model of CHERI-MIPS to OCaml
  - Successfully (albeit slowly) execute CHERI-MIPS test suite

Example Proof for ARMv8-A

Key question: Is such a large specification actually useable for proof?

Address translation: Most complex part of ARMv8 model!

- 9000 lines of specification required
- Page table walk: Over 500 LOS excluding helper functions
  - . . . and there are *lots* of page table helper functions
- Involves iteration, variable-length bitvectors, memory effects, nondeterminism, . . .

Example Proof for ARMv8-A

We define a simple characterisation of address translation suitable for reasoning about non-system code
About 500 lines of Isabelle total

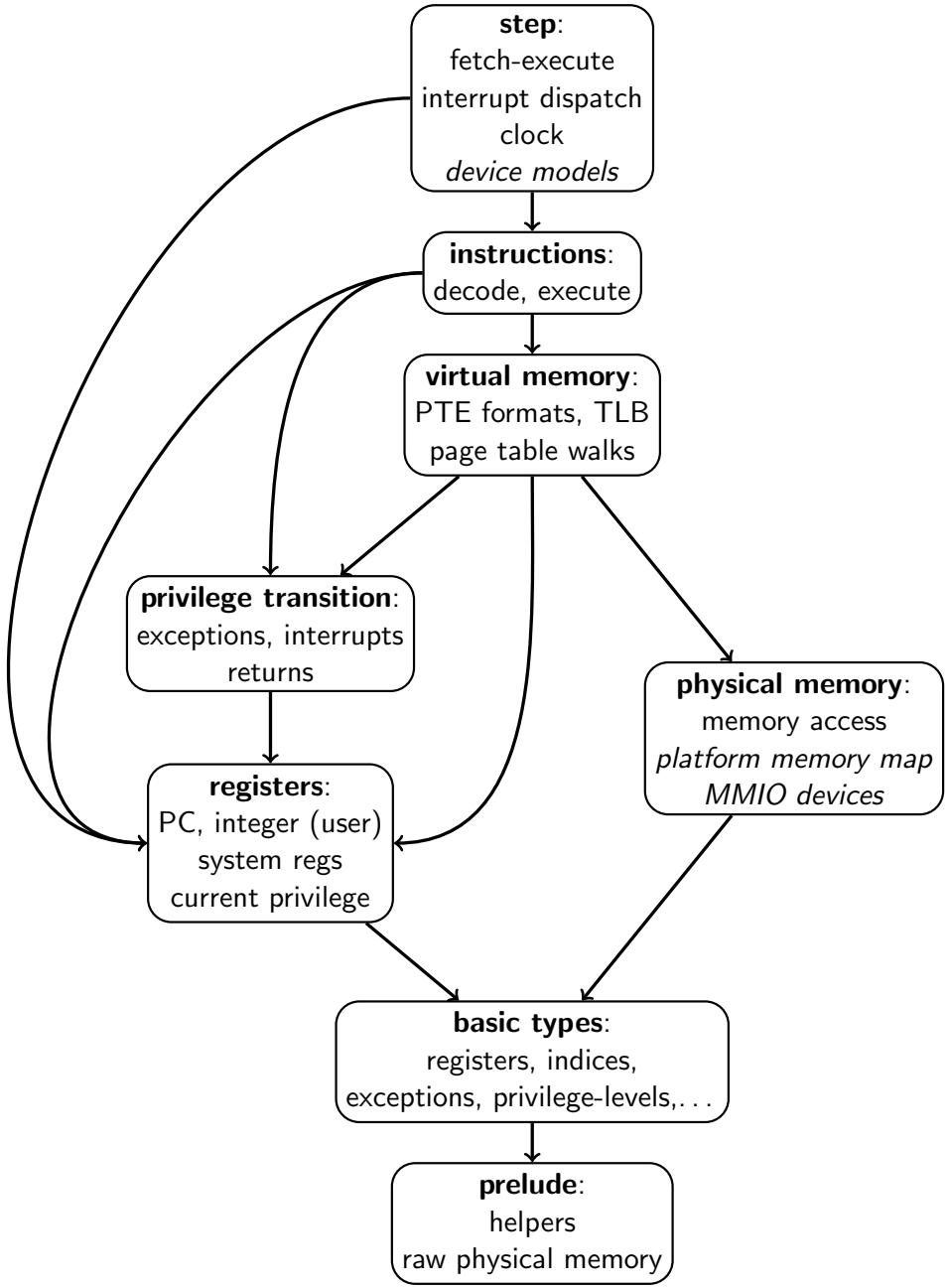
Theorem
Simplified address translation is equivalent to full ARMv8 address translation under certain useful assumptions
user mode, no virtualisation, valid translation tables, hardware updating of translation table flags

Uncovered a few small bugs in the ASL specification

RISC-V in Sail

```
sail-riscv
+---- model                        // Sail specification modules
+---- generated_definitions        // Files generated by Sail
|     +---- c, ocaml, lem, isabelle, coq, hol4, latex
|---- handwritten_support          // Prover support files
+---- c_emulator                   // supporting platform files
+---- ocaml_emulator               // supporting platform files
+---- doc                          // documentation
+---- test                         // test files
```

RISC-V Specification Structure

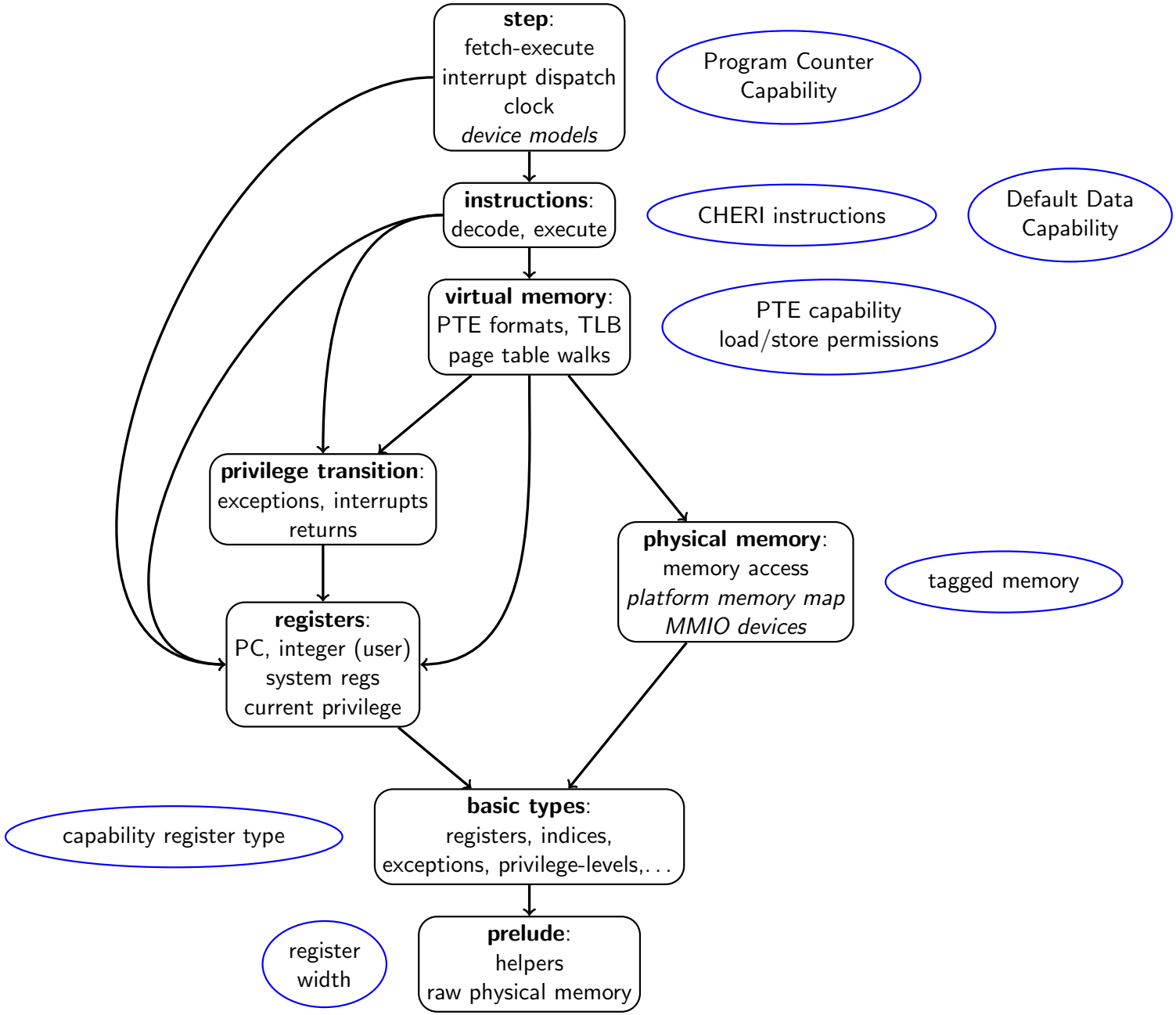


Extendable ISA Specifications

Possible extension points

- register width (e.g. 32/64, 32+64)
- new registers (floating point, vector)
- privilege levels (e.g. M-only, M/U, M/S/U, virtualization)
- physical memory (tagged memory)
- address translation (virtualization, security extensions)
- adding new instructions
- adding co-processors (debug, crypto, vector)

Extendable ISA Specifications



Extensions in Sail RISC-V

- draft 'N' standard extension
- draft 'Xcheri' non-standard extension

'N' Extension in Sail RISC-V

- additional control/status registers
- changes to exception/interrupt handling

'Xcheri' Extension in Sail RISC-V

- register formats (capability format)
- new processor exceptions
- physical memory access (tag metadata)
- virtual memory (permissions, PTE formats)
- new instructions
- semantics of existing instructions
- changing memory access due to instruction fetch

Conclusion

The RISC-V and CHERI-RISCV models are available:

`https://github.com/rems-project/sail-riscv`

`https://github.com/CTSRD-CHERI/sail-cheri-riscv`

Feedback welcome!