ISA Formal Spec Technical Group Update

Rishiyur S. Nikhil
nikhil at bluespec.com

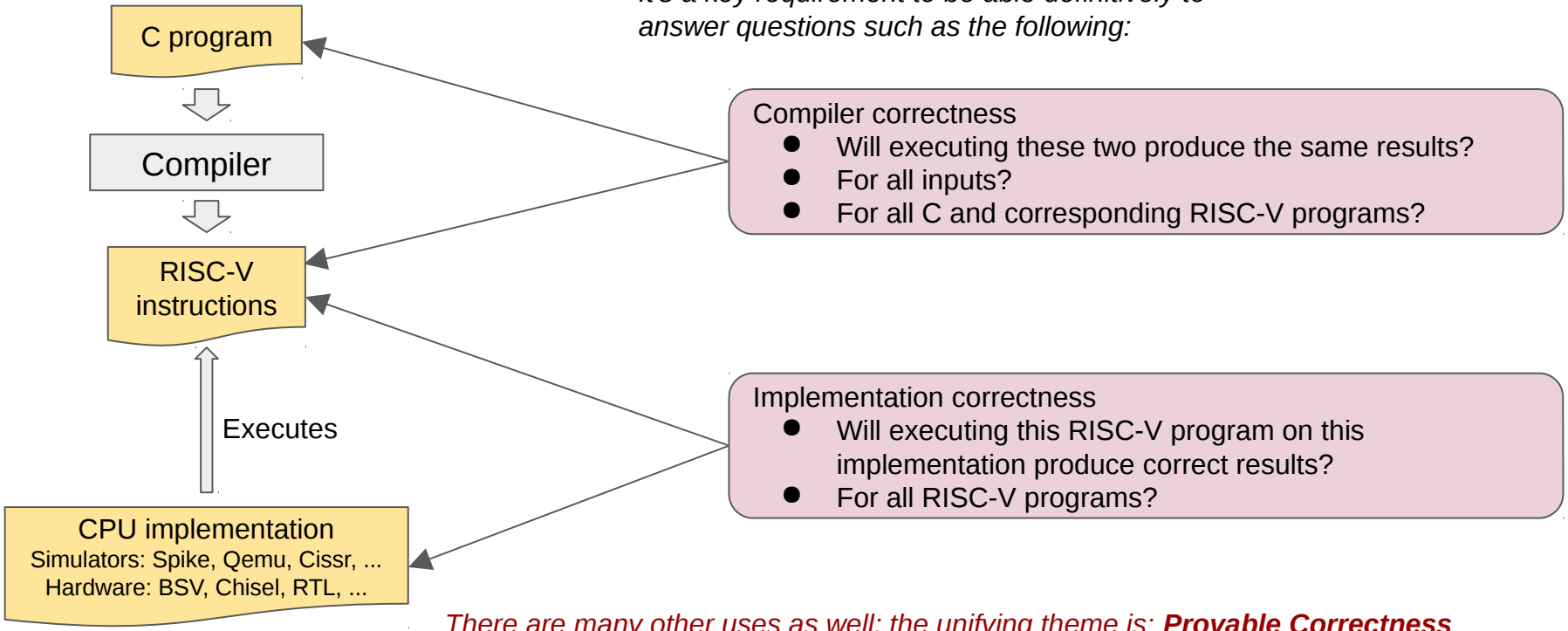
8th RISC-V Workshop
Tuesday, May 8, 2018

Technical Group Members

- 43 members, weekly meetings (about 10-15 attend regularly)
- Industry: BAE, Bluespec, Dover Microsystems, Draper, Galois, Imperas, Microsemi, NVidia, Qualcomm, SiFive, Syntacore, Tesla, VeriSilicon, Western Digital
- Non-profits: LowRISC
- Academia: IIT Madras, MIT, U.Cambridge, U.St.Andrews
- and several individual members

Of What Use is an ISA Formal Spec?

It's a key requirement to be able definitively to answer questions such as the following:



There are many other uses as well; the unifying theme is: ***Provable Correctness***

What is an ISA Formal Spec?

Goals:

- *Clear and understandable to the human reader*
  - *Not cluttered with implementation considerations; no micro-architectural detail*
  - *Accessible to the practicing engineer: HW implementers, simulator implementers, compiler writers, OS implementers, concurrent shared data structure authors, ...*
- *Precise and complete*
  - *(including allowed non-determinism of some instructions)*
- *Machine readable*
- *Executable (run RISC-V programs, boot an OS)*
- *Usable with various formal tools (theorem provers, model checkers, verifiers, …)*

English text specs, and instruction-set simulators (Spike, Qemu, Cissr) can be regarded as specs, but typically do not meet many of these goals.

Key Issues (1)

Modularity: RISC-V is perhaps the “most modular” of ISAs, with plug-and-play options for

- RV32 / RV64 / RV128
- Standard options: C, A, M, F, D, V, …
- Optional Privilege Spec with M, U+M, U+S+M optional combinations
- Within S privilege, for RV64, multiple Virtual Memory options (Sv39, Sv48, …)

Some of these options are *dynamically* selectable in an implementation.
Some of these options can vary between privilege levels (e.g., fundamental XLEN)

Almost all actual implementations only implement a small subset of these choices,
and usually not dynamically selectable,
but the spec needs to be parameterized to capture all combinations, and remain readable.

(unclear whether this has ever been attempted before)

Key Issues (2)

Extensibility:

- RISC-V is open and deliberately aimed at facilitating architecture research
- Creating the ISA formal spec is not a one-time project with a definite “completion”.

riscv.org would like to extend the formal spec to cover all current and future official extensions.

Others would like to extend the formal spec to incorporate their extensions and variants.

(unclear whether this has ever been attempted before)

Key Issues (3)

Allowed non-determinism, driven by concurrency (pipelining, speculation, out-of-order, ...)

- With RISC-V's Weak Memory Model
  - A “LOAD” can return different values on different runs
  - A “LOAD” can return different values depending on presence/absence of FENCE instructions, even though FENCEs don't access architectural state

How should the formal spec express this, while remaining readable for the practicing engineer?

Status

- *RISC-V Foundation decided early to have dedicated groups for ISA formalization and for Memory Model Formalization, as each is a significant task, and they are relatively orthogonal*

- *Because of key issues described earlier (specifically: modularity, concurrency), and the fact that solutions have no clear precedents in the literature, teams within the group are exploring various approaches:*
  - *3 projects in Haskell*
  - *1 project in SAIL (new language at U.Cambridge specifically designed for ISA specs)*
  - *1 project in L3 (will merge into SAIL project)*
  - *1 project in 'functional subset' of Verilog*

Status

- *One of the Haskell projects (led by team at MIT) is furthest along*
  - *Currently modeling RV32I, RV64I, M, Priv U+S+M, Sv39 VM*
  - *There was a tutorial on this yesterday (Monday May 7).*

- *Already in use:*
  - *As a simulator for executing RISC-V binaries*
    - *It can be used as a "Tandem Verifier" for RISC-V implementations*
  - *Transformed into Coq, where it is being used to prove correctness of a small compiler targeting RISC-V*
  - *Transformed into Coq, where it is being used to prove correctness of certain hardware implementations*
  - *Transformed into Verilog (via the tool 'Clash') where it has been model-checked against Spike and Clifford Wolf's Verilog-based riscv-spec*

https://github.com/mit-plv/riscv-semantics

- *Galois, Inc. is exploring use of the Formal Spec to prove Security properties of standard RISC-V, and RISC-V with security extensions*

- *We have not done any work on floating point (F and D) yet.*
  - *Is this simple? (i.e., can we ride on any existing spec for IEEE floating point?)*

Next steps

Within a couple of months, publish a Formal Spec (from one of the versions cited earlier) that is "complete" for the following:

- *RV32IMAC, RV64IMAC*
- *Priv Spec: User, Supervisor, Machine*
  - *Virtual Memory Sv32 and Sv39*
- *Simple sequential memory model*

Liaise with "Compliance" group, to use Formal Spec as "golden reference" for compliance suites.

Immediate follow-up: Integration with Weak Memory Model (this may be a new group, likely sharing members with current ISA and Mem Model groups)

- *Generalize the ISA Formal Spec to express concurrency and allowed non-determinism, and incorporate Weak Memory Model*

Questions? Discussion? Thank you!

Other interesting follow-up projects (by community, not necessarily by the Formal Spec Group)

- *Formally show equivalences between different ISA formalizations*
- *Demonstrate extensibility to other standard options (e.g., Vector, Crypto, …)*
- *Use it! (to prove correctness of HW implementations, compilers, ...)*