WHAT WERE OUR GOALS?

• Define the RISC-V memory consistency model
  • *Specifies the values that can be returned by loads*

• Support a wide range of HW implementations

• Support Linux, C/C++, and lots of other critical SW
DECADES OF DEBATE

**Strong Models**
(e.g., x86-TSO)
- Stricter ordering rules
- Simpler for programmers and for architects

**Weak Models**
(e.g., ARM, IBM Power)
- More relaxed ordering rules
- Better perf/power/area
- More microarchitectural freedom

Linux/C/C++/Java/... work either way! That’s not a deciding factor
FINDING A COMPROMISE

• First step: narrow choice to two specific models
  • **RVTSO**: as used by SPARC, x86  (*strong*)
  • **RVWMO**: similar to ARMv8  (*weak*)
    • “**RISC-V Weak Memory Ordering**”

• Both are “multi-copy atomic” = both are simpler than the IBM Power and ARMv7 memory models
THE RISC-V MEMORY MODEL PLAN

- **Base RISC-V memory model: RVWMO**
  - HW can still be as conservative as it wants to be
  - Portable SW must nevertheless assume RVWMO

- **New ISA extension “Ztso”:** HW which implements RVTSO can (optionally) choose to expose this to SW
  - E.g., a RISC-V core can implement “IMAFD+Ztso”
ARCHITECTURAL INTUITION (FOR BOTH)

Pipelines
(In-order or OoO)

Hart-private buffering
(values can be forwarded only to loads from the same hart that issued that store)

Atomic Memory
(all values globally visible to all harts, possibly via a coherence protocol)
ARCHITECTURAL INTUITION (FOR BOTH)

- RVWMO and RVTSO differ in the degree of memory access reordering they permit at the point of global visibility

  - RVTSO: only store→load reordering can be observed
  - RVWMO: unless otherwise synchronized (e.g., via .aq, .rl, and/or fences), most memory accesses can be reordered freely
RVWMO RULES IN A NUTSHELL

• A guaranteed to happen before B **only if**:

- Same addr.
- Store
- Fence
- with pr/pw/sr/sw set appropriately
- .aq
- .rl
- .aqrl
- AMO .aqrl
- Addr/ctrl/ data dep.
- except ctrl deps. where B is a load

• Each load returns value from most recent store to same address
• No store from another hart can interrupt an AMO or LR/SC
• Available offline: Alloy/herd/operational model, lots of docs

(some last details still being finalized...)
RVTSO RULES IN A NUTSHELL

• A guaranteed to happen before B only if:

  - Each load returns value from most recent store to same address
  - No store from another hart can interrupt an AMO or LR/SC
  - To be made available: Alloy/herd/operational model, docs

Strictly stronger than RVWMO
THE RISC-V MEMORY MODEL: SOFTWARE

- Portable SW must assume RVWMO, but it will run on both RVWMO HW and Ztso HW
  - Linux, gcc, bintools, ... will target RVWMO by default

- RVTSO-only SW can be written, but it will only run on HW implementing Ztso
  - Object files will use different magic number

<table>
<thead>
<tr>
<th></th>
<th>Base HW (RVWMO)</th>
<th>HW using Ztso ISA ext.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard SW (RVWMO)</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>RVTSO-only SW</td>
<td>WILL NOT RUN</td>
<td>OK</td>
</tr>
</tbody>
</table>
EVERYONE GETS WHAT THEY WANT

• If you don’t want to think about memory models: just use the standard OSes and toolchains

• If you care about PPA or flexibility: use RVWMO

• If you have lots of legacy x86 code: use HW implementing Ztso, so that any SW will work

• If you believe TSO is the future: use HW with Ztso, and emit code with the “TSO-only” magic number
FRAGMENTATION

• **Risk:** SW will fragment into “RVWMO version” and “RVTSO version”, double the maintainer burden

• **Solution:** discourage the 2x model, and encourage RVWMO SW, since it’s portable across all HW
  
  • Current plan for Linux, gcc, binutils, etc.
  
  • Redundant fences simply become no-ops

• **However:** can’t stop people from customizing; RISC-V’s openness is a feature, not a bug
OTHER ISA CHANGES

• ld.rl and sd.aq are deprecated
• ld.aqrl and sd.aqrl means RCsc, not fully fenced
• Clarified other subtleties of atomicity and LR/SCs
• Possibly in a future extension:
  • Fences may take an address restriction parameter
  • Opcodes for “l{b|h|w|d}.aq” and “s{b|h|w|d}.rl”
I/O FENCES, FENCE.I, SFENCE.VM

• Informal descriptions given in memory model spec
• No change to I/O channel ordering behavior
• Some clarification to FENCE .pi/.po/.si/.so
• More detail available offline
• Future: V/T/J compatibility as well
DOCUMENTATION & TOOLS

- Definitions of RVWMO/RVTSO and Ztso
- A dozen pages explaining the details in plain English
- Two axiomatic models (Alloy + herd)
- One formal operational model + app
- Lots of litmus tests
  - (also to be used to test compliance)
- ...and the memory model TG as a resource to people who have questions
- Come find me or email me!

1.10 Formal Axiomatic Specification

We present two formal specifications of the RISC-V memory model. Both are written in Alloy (http://alloy.mit.edu).

The first corresponds directly to the natural language model earlier in this chapter.

```alloy
// RISC-V axiom
```

Come find me or email me!
RISC-V MEMORY MODEL ROADMAP

• We’ll discuss this further Thursday at the members-only meeting
  • Or come find me in the meantime
• We’ll aim to release the complete documentation publicly (on isa-dev) soon after that
• If all goes well, we’ll work with the Technical Committee to work towards ratification