RISC-V Updates

Krste Asanović
krste@berkeley.edu

http://www.riscv.org

3rd RISC-V Workshop
Oracle, Redwood Shores, CA
January 5, 2016
Agenda

- UC Berkeley updates
- RISC-V transition out of Berkeley
- Outstanding issues, seeds for discussion
**Brief RISC-V History/Status**

- Starting life in 2010 as “3-month” project to develop clean-slate ISA for research and education at Berkeley
- Initial spec, May 2011
- Frozen user-level ISA spec (IMAFDQ), May 2014

1.5 years later:
- 3rd Workshop, sold out 160+ registrants from ~80 companies and universities
- Foundation launching with major industrial support
- Many Universities adopting for education
- Indian Govt adopted as ISA standard
- Many RISC-V implementations underway: academic, open-source, proprietary
- Ecosystem filling out
Compressed Extension “C”

- Original version developed 2011
- Extensive v1.7, May 2015
  - Daddy Bear
- Slashed v1.8, Sep 2015
  - Baby Bear
- Reworked expanded v1.9, Nov 2015
  - Mummy Bear?

- No feedback since, so we intend to bless v1.9 to be considered frozen (RV32C/RV64C), and become v2.0. Should discuss any issues here at workshop.
V1.7 May 2015

Much feedback, we’re still digesting and reworking design

Behind schedule, but hope to release updated draft addressing feedback in next month or two
  – QEMU now being updated to match current spec

We doubt this will really settle down until next summer
  – Need more OS and driver development against it
  – Need to define platform interaction below
  – A good topic for breakout discussion here
Vector Extension “V”

- Sketch of design at last workshop
- Not ready to release draft in time for this workshop
- But we open-sourced RTL for our Hwacha vector coprocessor, including 3 techreports with documentation in December.

- Hwacha ISA is NOT the V extension
- Hwacha is designed to push limits of in-order decoupled data-parallel accelerators (e.g., GPUs)
- Working OpenCL compiler in LLVM
- Hwacha microarchitecture demonstrates several ideas that will appear in V
Ongoing ISA Research at UCB

- Virtual Local Store (VLS, Cook MS 2009)
  - Use portion of cache as scratchpad, plays nice with OS
- User-level DMA
  - Copy data between DRAM and VLS directly

- Still working through details, might form ISA proposal at later stage
New Berkeley Open-Source Cores

- BOOM: Berkeley Out-of-Order Machine, open-sourced Dec 2015, talk tomorrow by Chris Celio

- V-Scale: Verilog implementation of Z-Scale, open-sourced Sep 2015
Transitioning RISC-V Out of Berkeley

- Many dimensions to the transition

- RISC-V Foundation to take over standards process

- Tools being upstreamed
  - Binutils first, glibc, gcc, Linux, LLVM to follow
  - Lots of paperwork, lawyers involved

- Students graduating
  - Lead RISC-V developers no longer at UCB
  - But many newer students working on RISC-V @UCB-BAR
Freeing Silicon

... because Moore’s Law only ends once.

“By bringing the power of open-source and agile hardware design to the semiconductor industry, SiFive reduces the cost to harness the performance and energy-efficiency of customized silicon, enabling whole new ranges of applications.”

sifive.com

• Founders: Yunsup Lee, Andrew Waterman, Krste Asanovic
• CEO: Stefan Dyckerhoff
• Investors: Sutter Hill Ventures
RISC-V Landscape

RISC-V Foundation
(US 501(c)(6) non-profit)
15+ members in 2016

UC Berkeley
(Originators, developers)

IIT Madras
(Indian Govt.)

MIT

Non-Profit
Private
Ventures

lowRISC
(UK non-profit)

Big Co. A

Big Co. B

Big Co. C

Big Co. D

Big Co. ...

Commercial
Internal
Users

Bluespec

Rumble

SiFive

...
Most Urgent Outstanding Issues

- Holes and ambiguities in specification
- Platform specification
Working on holes in specification

- Floating-point NaNs
  - Resolved, updated spec
- CSR read/write semantics
  - Resolved, updated spec
- Memory model
  - Far from resolved
- Hypervisor support
  - No proposal
- Other holes to be discovered

- Formal spec of whole ISA+memory model desired, would be industry-first for real systems
Although RISC-V was designed in reusable layers, some concrete standards for hardware platforms are desirable:

- Memory Maps (RAM/ROM, I/O, CSRs, Debug CSRs)
- Interrupt Controls
- Power management
Platform Landscape

- **Microcontroller**
  - Memory-mapped I/O, no caches or DRAM
  - One or more cores

- **Apps processor**
  - Single-chip OS-capable uni/multiprocessor

- **Rack-scale machine**
  - Hundreds of nodes

- To what extent can platform specs be shared among these machines?
Questions?