The future of operating systems on RISC-V

lowRISC

Alex Bradbury asb@lowrisc.org @asbradbury

Structure of this talk

- Introduction to RISC-V
- RISC-V status
- Selected RISC-V topics
- RISC-V and open hardware: the future
- Conclusion

Introduction to RISC-V

- RISC-V: an open standard instruction set architecture (ISA)
  - But wait, what's an ISA?
  - Ecosystem of both open and proprietary implementations
- Allows / encourages custom extension
- Open standards, open(ish) development process, and (often) open implementations: a new model of development for the hardware industry?
- Managed by the RISC-V Foundation
- A "boring" design is a good thing in an ISA

Introduction to RISC-V: Details

- Key aim: flexibility. Scale up to HPC and down to deeply embedded MMU-less devices.
  - If standard solutions don’t work, add your own extensions
  - Flexibility can be a disadvantages. Opportunities, but also challenges
- “Base” ISAs: RV32I, RV32E, RV64I, RV128I
- Standard extensions: MAFDC
- Instruction encoding: 16-bit, 32-bit, 48-bit, ...
- Privileged vs unprivileged ISA
- Beyond the ISA

Background: FPGAs, ASICs, semiconductor economics

- FPGA: Field Programmable Gate Array
  - Pictured: Nexys A7, ~$270, Xilinx Artix-7 FPGA
  - Can run Rocket at 50MHz, boot Linux etc
- ASIC vs FPGA vs simulation
- ASIC volumes. 10s (multi-project wafer) or millions (volume run) are “easy”. Middle ground isn’t really viable
- Semiconductor licensing models

RISC-V status

- Specifications
  - User-level and privileged ISAs going through “ratification”
  - New extensions in development. Also debug, interrupt controller etc
- Compilers, libc, languages
  - gcc, LLVM/Clang, glibc, musl, Go, Rust
- Simulation platforms
  - Qemu, gem5, spike, tinyemu
- Hardware
  - SiFive “Freedom” boards, Kendryte, open-isa.org, FPGA-ready distributions (e.g. lowrisc.org)
  - Open source implementations: Rocket, PULP, ...

RISC-V status

- OS
  - FreeRTOS, Zephyr, seL4, Tock
  - HarveyOS, HelenOS
  - Linux, FreeBSD
- Bootloader
  - Coreboot, u-boot, bbl, OpenSBI
- Linux distributions
  - Debian, Fedora, Alpine, ...

Aside: Why are RISC-V pages 4KB? Check the spec

For many applications, the choice of page size has a substantial performance impact. A large page size increases TLB reach and loosens the associativity constraints on virtually-indexed, physically-tagged caches. At the same time, large pages exacerbate internal fragmentation, wasting physical memory and possibly cache capacity.

After much deliberation, we have settled on a conventional page size of 4 KiB for both RV32 and RV64. We expect this decision to ease the porting of low-level runtime software and device drivers. The TLB reach problem is ameliorated by transparent superpage support in modern operating systems [2]. Additionally, multi-level TLB hierarchies are quite inexpensive relative to the multi-level cache hierarchies whose address space they map.

RISC-V Privileged specification 1.10

RISC-V: Selected topics

SBI: Background

- Supervisor Binary Interface (SBI)
- Privilege levels
  - M: Machine
  - S: Supervisor
  - U: User
- SBI provides an interface between the OS and Supervisor Execution Environment (SEE)
- M-mode has full system access, can be used to emulate missing functionality

SBI

- Aim: Allow a single OS binary to run on all SEE implementations
- Current interface minimal (timer, inter-processor interrupts, remote fences, console, shutdown). Proposals to extend: power management, even context switch
- Controversy: puts large amount of trust in potentially opaque binary blobs. See arguments from e.g. Ron Minnich (Coreboot)

Virtualisation

- See: “Proposal for virtualization without H mode” by Paolo Bonzini (KVM maintainer).
  - Also “RISC-V Hypervisor Extension” slides (Dec 2017, Bonzini+Hauser+Waterman)
- Rather than having H, M, S, U mode, add “virtualized supervisor” and “virtualized user” modes. Introduces “background” CSRs.
- Great example of collaborative development, benefitting from expert input

RISC-V and open hardware: the future

Ingredients for rapid hardware/software innovation

- Ideas
- Open standards
- High quality, well tested + verified open implementations
- Active development community
- Mechanism for “capturing” contributions.
  - Process for reviewing and agreeing proposals / code contributions.
  - Then shipping in future spec or hardware

Malleable hardware

- Is this a “clean slate” opportunity?
- Same old challenges (security, energy efficiency, performance). Potential for new solutions if changes are possible across ISA, microarchitecture, OS, compiler, languages, …
- More viable for some market segments than others: normal market forces still in play

Idea -> prototype

- Plan changes
- Prototype in simulator, make necessary software changes
- Modify a hardware implementation and test with FPGA / Verilator
- Publish changes and write up
- Pathway to inclusion in shipping hardware is more difficult, though multiple groups working on this
    - lowRISC is aiming for regular tapeouts so community members can see their contributions realised
    - SiFive aiming to lower barrier for new silicon
    - Whole array of other startups and organisations

Example: direct segments (University of Wisconsin)

- Direct segments: optimisation for page-based virtual memory. Avoid TLB miss overhead by mapping part of a process' virtual address space to contiguous physical memory
- Proposed originally in 2013, but evaluated using a simple analysis based on counting TLB misses
- Thanks to availability of easily modifiable hardware implementation, can perform a better analysis
- Added 50 lines of Chisel code to Rocket and 400 lines to Linux kernel
- https://carrv.github.io/2018/papers/CARRV_2018_paper_4.pdf
- Novel HDLs: does it make it easier?

Novel security solutions

- Tagged memory (see lowRISC tagged memory releases and HWASAN for Arm)
  - See Katie Lim's write-up on adding Linux kernel support https://www.lowrisc.org/docs/tagged-memory-os-enablement-internship-2017/
- Spectre mitigations: same story as any other ISA, but access to open source superscalar processors like BOOM for research helps a lot
- Capabilities (see CHERI)
- ...

Minion cores

- Small micro-controller class cores scattered across the SoC
- Using same RISC-V ISA
- Open, not hidden (a la management engine)
- Potential use cases: soft / virtualized peripherals, security policies, near data computation, debug trace processing, …
- Prototyped on lowRISC platform (using PULP core), previous GSoC student ran TCP/IP stack using Rump kernels
- See also: custom accelerators

End goal: productive + public feedback loop between application engineers, compiler authors, micro-architects, ISA designers, ...

Conclusion

- Key challenges
  - Lowering the barrier to entry
  - Increasing the incentive for participation
  - Diversity and novel solutions are great. But how to maximise code reuse and infrastructure sharing?
- Questions?
- Contact: asb@lowrisc.org
- Sound interesting? We are hiring! 7 open positions: **www.lowrisc.org/jobs**

Overflow

New extensions: vector, bitmanip

Collaborative development: observations

Compliance and testing

Open FPGA toolchains

Memory model

LLVM status, development approach