Workshop Introduction

Krste Asanović
krste@eecs.berkeley.edu

http://www.riscv.org

First RISC-V Workshop, Monterey, CA
January 14, 2015

ISAs don't matter

Most of the performance and energy running software on a computer is due to:

- Algorithms
- Application code
- Compiler
- OS/Runtimes
- ISA (Instruction Set Architecture)
- Microarchitecture (core + memory hierarchy)
- Circuit design
- Physical design
- Fabrication process

In a *system*, there's also displays, radios, DC/DC convertors, sensors, actuators, …

ISAs do matter

- Most important interface in computer system
- Large cost to port and tune all ISA-dependent parts of a modern software stack
- Large cost to recompile/port/QA all supposedly ISA-independent parts of stack
- Proprietary closed-source, don't have code
- Bit rot, lose ability to compile own source code
- Lost your own source code

- Most of the cost of developing a new chip is developing software for it
- Most big new chips have several ISAs…

So…

If choice of ISA doesn't have much impact on system energy/performance,
and it costs a lot to use different ones,

why isn't there just one industry-standard ISA?

ISAs Should Be Free and Open

While ISAs may be proprietary for historical or business reasons, there is no good technical reason for the lack of free, open ISAs:

- It's not an error of omission.
- Nor is it because the companies do most of the software development.
- Neither do companies exclusively have the experience needed to design a competent ISA.
- Nor are the most popular ISAs wonderful ISAs.
- Neither can only companies verify ISA compatibility.
- Finally, proprietary ISAs are not guaranteed to last.

Benefits from Viable Freely Open ISA

- ***Greater innovation via free-market competition*** from many core designers, closed-source and open-source.
- ***Shared open core designs***, shorter time to market, lower cost from reuse, fewer errors given more eyeballs, transparency makes it difficult for government agencies to add secret trap doors.
- ***Processors becoming affordable for more devices***, which would help expand the Internet of Things (IoTs), which could cost as little as $1.
- ***Software stacks survive for long time*** upgrade software on systems embedded in concrete 50 years ago
- ***Make architecture research and education more real*** with fully open hardware and software stacks

What Style of ISA?

- If ISA doesn't matter, then pick an easy-to-implement, efficient ISA

- Seymour Cray's 50-year-old load-store design, refined by RISC research 30+ years ago, still a good choice

ISA innovation over last 30 years?

- Load-store ISAs predate Moore's Law
- Very little (nothing?) stuck for general-purpose code, failures:
  - CISC (microcode worse than instruction cache in VLSI)
  - Stack machines (compiler can target registers)
  - VLIW (dynamic predictors beat static scheduling)
- Industry has reconverged on simpler RISC ideas:
  - ARM v8, MIPS r6, both similar ideas to RISC-V

- Specialized Accelerators? Only way to improve performance until post-CMOS (20-30 years away)
  - G. Estrin, "Organization of computer systems: the fixed plus variable structure computer," Proc. Western Joint Computer Conf., pp. 33-40; 1960
  - GPUs – bad version of Cray vectors

Other Open ISAs

- ***SPARC V8*** - To its credit, Sun Microsystems made SPARC V8 an IEEE standard in 1994. Sun, Gaisler offered open-source cores. ISA now owned by Oracle.
- ***OpenRISC*** - GNU open-source effort started in 1999, based on DLX. 64-bit ISA was in progress.
- ***Open Processor Foundation*** - Patents expiring on Hitachi SH architecture, so open-sourcing tools and implementations
- ***Lattice Micro 32 (LM32)*** – Simple FPGA soft core.

- All RISC ISAs

RISC-V Background

- In 2010, after many years and many projects using MIPS, SPARC, and x86 as basis of research, time to look at ISA for next set of projects
- Obvious choices: x86 and ARM
- x86 impossible – too complex, IP issues

Intel x86 “AAA” Instruction

- ASCII Adjust After Addition
- AL register is default source and destination

- If the low nibble is > 9 decimal, or the auxiliary carry flag AF = 1, then
  - Add 6 to low nibble of AL and discard overflow
  - Increment high byte of AL
  - Set CF and AF
- Else
  - CF = AF = 0

- Single byte instruction

RISC-V Background

- In 2010, after many years and many projects using MIPS, SPARC, and x86 as basis of research, time to look at ISA for next set of projects
- Obvious choices: x86 and ARM
- x86 impossible – too complex, IP issues
- ARM mostly impossible - complex, IP issues
- So we started "3-month project" in summer 2010 to develop our own clean-slate ISA
- Four years later, we released frozen base user spec
  - But also many tapeouts and several research publications along the way

2015 RISC-V Project Goal

Become the industry-standard ISA for all computing devices

RISC-V is NOT an Open-Source Processor

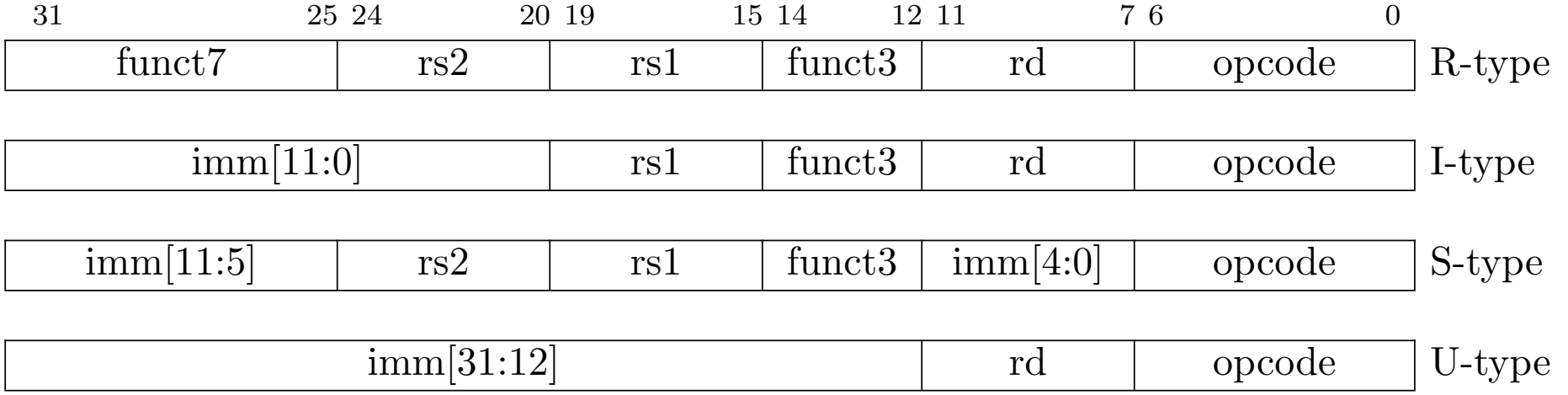
- RISC-V is an ISA specification
- Will be expanding to be specifications for SoCs including I/O and accelerators
- Most of cost of chip design is in software, so want to make sure software can be reused across many chip designs

- Want to encourage both open-source and proprietary implementations of the RISC-V ISA specification

RISC-V Base Plus Standard Extensions

- Three base integer ISAs, one per address width
  - RV32I, RV64I, RV128I
  - Only 40 hardware instructions needed
- Standard extensions
  - M: Integer multiply/divide
  - A: Atomic memory operations (AMOs + LR/SC)
  - F: Single-precision floating-point
  - D: Double-precision floating-point
  - G = IMAFD, “General-purpose” ISA
  - Q: Quad-precision floating-point
- All the above are a fairly standard RISC encoding in a fixed 32-bit instruction format
- Now frozen

RISC-V Standard Base ISA Details



- 32-bit fixed-width, naturally aligned instructions
- 31 integer registers x1-x31, plus x0 zero register
- No implicit registers, rs1/rs2/rd in fixed location
- Floating-point adds f0-f31 registers plus FP CSR, also fused mul-add four-register format
- Designed to support PIC and dynamic linking

"A": Atomic Operations Extension

Two classes:

- Atomic Memory Operations (AMO)
  - Fetch-and-op, op=ADD,OR,XOR,MAX,MIN,MAXU,MINU
- Load–Reserved/Store Conditional
  - With forward progress guarantee for short sequences

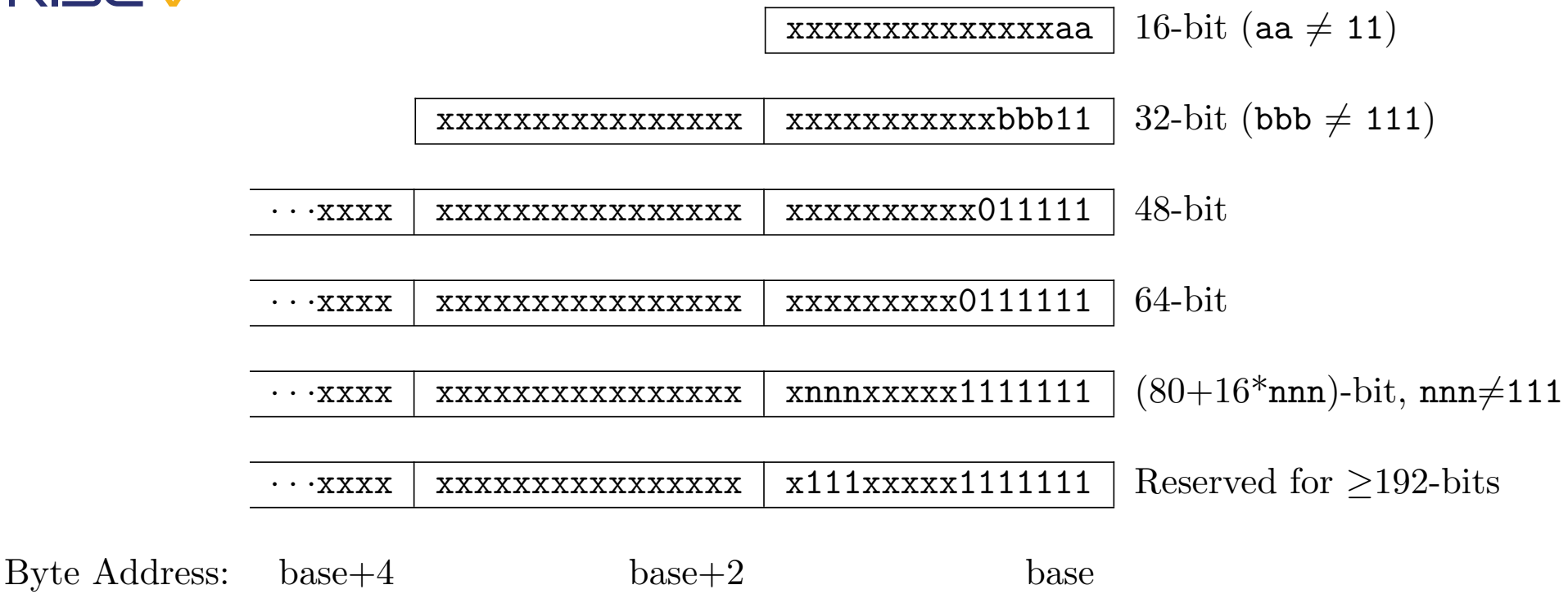
- All atomic operations can be annotated with two bits (Acquire/Release) to implement release consistency or sequential consistency

Why do SoCs have so many ISAs?

- Applications processor (usually ARM)
- Graphics processors
- Image processors
- Radio DSPs
- Audio DSPs
- Security processors
- Power management processor

- IP bought from different places, each proprietary ISA
- Home-grown ISA cores
- Apps processor ISA too huge for base accelerator ISA
- Rebuild whole software stack for each ISA to speed up <1% of the total code!!!

Variable-Length Encoding

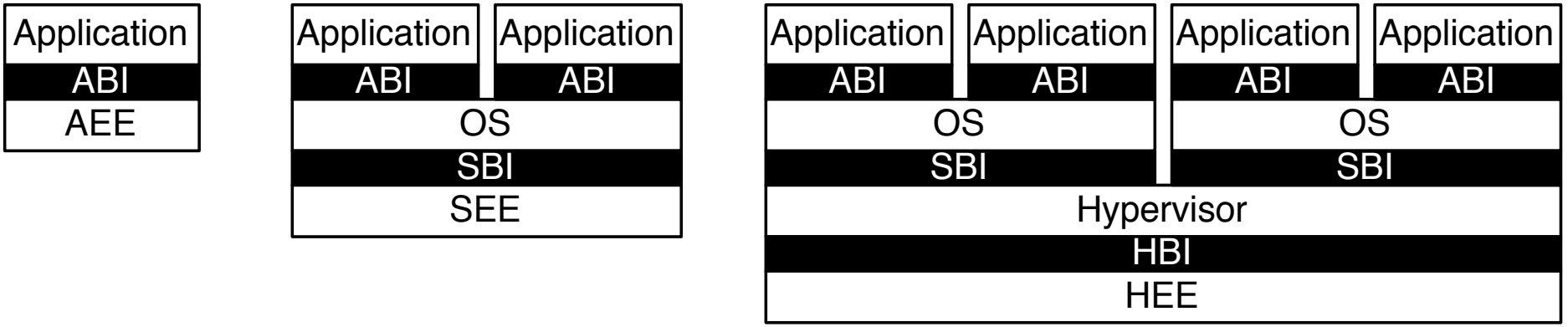


- Extensions can use any multiple of 16 bits as instruction length
- Branches/Jumps target 16-bit boundaries even in fixed 32-bit base

"C": Compressed Instruction Extension

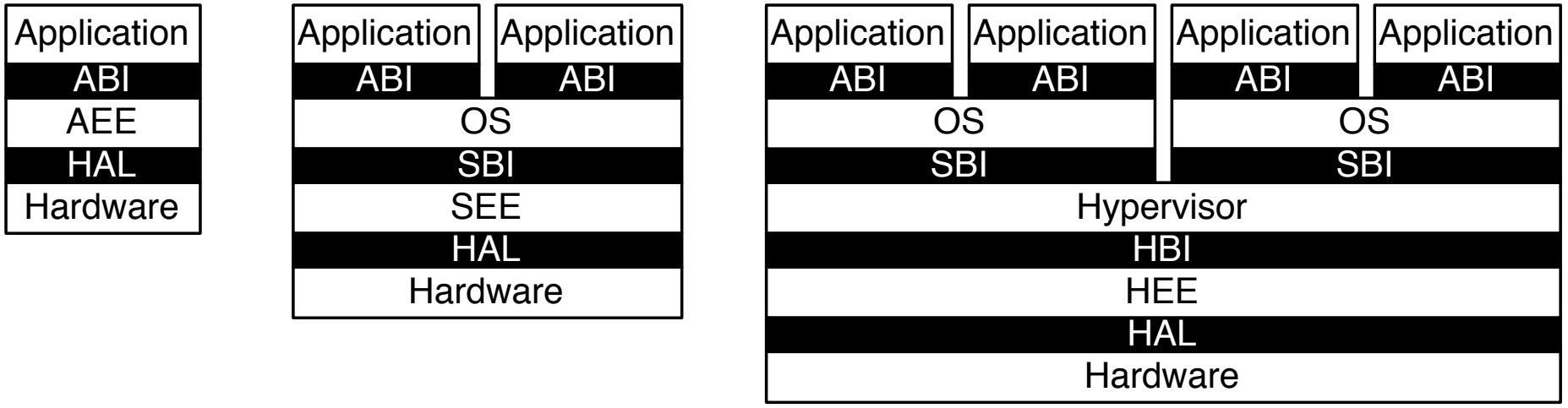
- Compressed code important for:
  - low-end embedded to save static code space
  - high-end commercial workloads to reduce cache footprint
- Standard extension (still in draft, not frozen) adds 16-bit compressed instructions
  - 2-address forms with all 32 registers
  - 3-address forms with most frequent 8 registers
- Each C instruction expands to a single base ISA instruction
- All original 32-bit instructions retain encoding but now can be 16-bit aligned
- Approximately 25% reduction in code size, with performance improvement from reduced I$ misses

RISC-V Privileged Architecture



- Provide clean split between layers of the software stack
- Application communicates with Application Execution Environment (AEE) via Application Binary Interface (ABI)
- OS communicates via Supervisor Execution Environment (SEE) via System Binary Interface (SBI)
- Hypervisor communicates via Hypervisor Binary Interface to Hypervisor Execution Environment
- All levels of ISA designed to support virtualization

RISC-V Hardware Abstraction Layer



- Execution environments communicate with hardware platforms via Hardware Abstraction Layer (HAL)
- Details of execution environment and hardware platforms isolated from OS/Hypervisor ports

Four Supervisor Architectures

- Mbare
  - Bare metal, no translation or protection
- Mbb
  - Base and bounds protection
- Sv32
  - Demand-paged 32-bit virtual address spaces
- Sv43
  - Demand-paged 43-bit virtual address spaces

- Designed to support current popular operating systems
- Where's the draft spec????
  - This is taking some time, very soon now.
  - It won't be finalized for a while, since we need feedback

RISC-V Ecosystem

www.riscv.org

- ***Documentation***
  - User-Level ISA Spec v2
  - Reviewing Privileged ISA
- ***Software Tools***
  - GCC/glibc/GDB
  - LLVM/Clang
  - Linux
  - Yocto
  - Verification Suite
- ***Hardware Tools***
  - Zynq FPGA Infrastructure
  - Chisel
- ***Software Implementations***
  - ANGEL, JavaScript ISA Sim.
  - Spike, In-house ISA Sim.
  - QEMU
- ***Hardware Implementations***
  - Rocket Chip Generator
    - RV64G single-issue in-order pipe
  - Sodor Processor Collection

1st Workshop Goals

- Start building the wider RISC-V community
- Identify needs, and find volunteers to help
- Figure out structure to manage future RISC-V development
- Determine goals/structure for a RISC-V foundation
- Where/when/format of next RISC-V event?

- Bootcamp: Hands-on tutorial with current toolchain and Rocket-Chip generator
  - 2nd Bootcamp at HPCA/CGO/PPoPP in SFO, February

Many things we know are needed…

- Formal ISA spec
- Privileged ISA draft spec with matching Linux port
- Compressed ISA draft spec with compiler support
- Vector unit ISA spec with compiler support
- JVM
- Debug/tracing support
- I/O node specification
- Better test suite
- …

Workshop Format

- Adapted from the Berkeley Retreat model
- Some short talks, plus 3-minute poster previews
- Dinner tables with breakout topics
- Poster sessions with lubricated discussions
- Breakout summary next day
- Group photo
- Feedback/discussion from attendees

- Bootcamp follows, including time to get assistance with tools or deeper discussions.

Dinner Table Organization

- No two people from same organization on same table (no more than 2-3 from Berkeley)
- Berkeley student acts as scribe
- Standard questions for all attendees:
  - What's your interest in RISC-V?
  - Primary thing preventing you using RISC-V?
  - Want a RISC-V foundation, if so, what should it do?
  - What can you contribute?
  - What format and location for next meeting?

  - Free form discussion after this

Using the Bootcamp VM Images

- Runs on Oracle Virtualbox (installers are on drive)
- File -> Import Appliance (`riscv.ova/.ovf`)
  User: riscv-bootcamp
  Password: riscv
- To use GCC 4.9+ tools, follow instructions in `~/.bashrc` comments
- Find Albert Magyar for help!

RISC-V Research Project Sponsors

- DoE Isis Project
- DARPA PERFECT program
- DARPA POEM program (Si photonics)
- STARnet Center for Future Architectures (C-FAR)
- Lawrence Berkeley National Laboratory
- Industrial sponsors (ParLab + ASPIRE)
  - Intel, Google, Huawei, LG, NEC, Microsoft, Nokia, NVIDIA, Oracle, Samsung