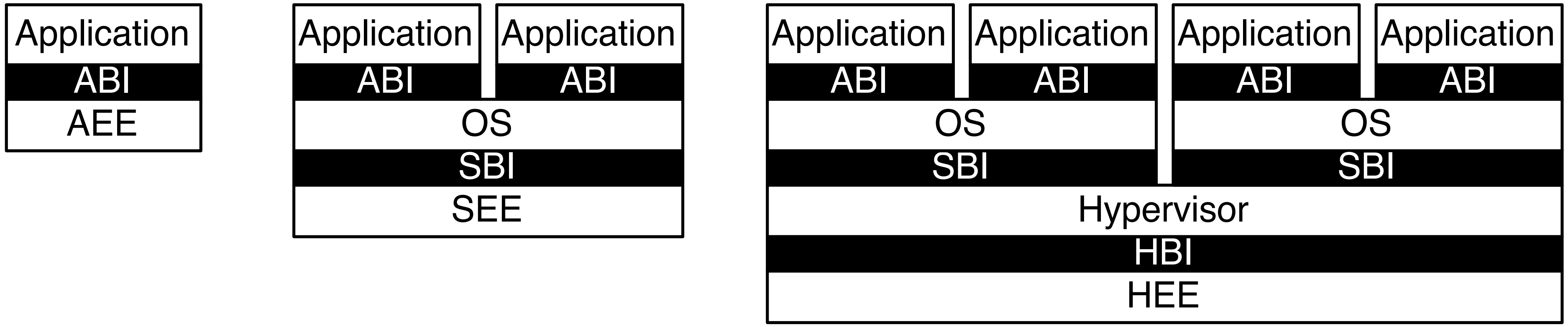
RISC-V Privileged Architecture Proposal

Krste Asanović, Rimas Avizienis, Yunsup Lee, David Patterson, Andrew Waterman
waterman@eecs.berkeley.edu
http://www.riscv.org

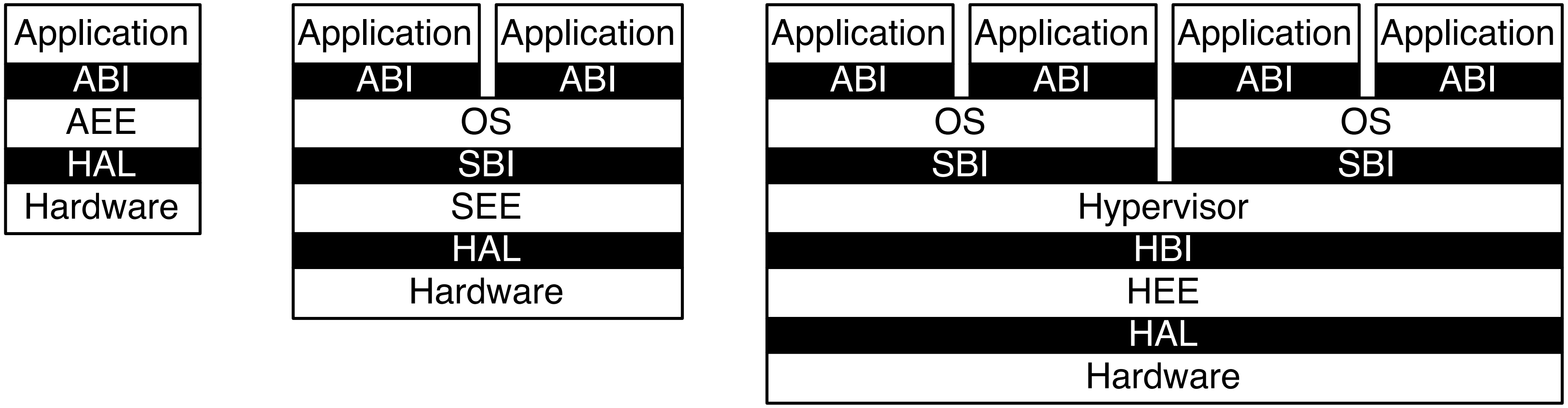
2nd RISC-V Workshop, Berkeley, CA
June 29, 2015

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

Privilege Modes

- Four privilege modes
  - User (U-mode)
  - Supervisor (S-mode)
  - Hypervisor (H-mode)
  - Machine (M-mode)
- Supported combinations of modes:

– M	(simple embedded systems)
– M, U	(embedded systems with protection)
– M, S, U	(systems running Unix-style operating systems)
– M, H, S, U	(systems running Hypervisors)

Simple Embedded Systems

- Simplest implementation needs only M-mode
- No address translation/protection
  - "Mbare" bare-metal mode
  - Trap bad physical addresses precisely
- Application code is trusted

- Low implementation cost
  - 2^7 bits of architectural state (in addition to user ISA)
  - $+2^7$ more bits for timers
  - $+2^7$ more for basic performance counters

Small System Memory-Management Architectures

- User mode (M+U) adds basic translation/protection
- Mbb
  - Base-and-bounds translation/protection
  - PA = VA + `mbase`; VA $\in$ [0, `mbound`-1]
  - Sufficient for basic multiprogramming, provided segments fit in memory
  - +2^6 bits of arch. state vs. Mbare
- Mbbid
  - Separate base-and-bounds for instructions & data
  - Can share instruction segment between multiple processes
  - +2^6 bits of arch. state vs. Mbb

Virtual Memory Architectures

- Designed to support current Unix-style operating systems
- Sv32 (RV32)
  - Demand-paged 32-bit virtual address spaces
  - 2-level page table
  - 4 KiB pages, 4 MiB megapages
- Sv39 (RV64)
  - Demand-paged 39-bit virtual address spaces
  - 3-level page table
  - 4 KiB pages, 2 MiB megapages, 1 GiB gigapages
- Sv48, Sv57, Sv64
  - Sv39 + 1/2/3 more page table levels

Why 4 KiB Pages?

- Initially planned to scale base page size w/XLEN
  - 8 KiB for RV64, 16 KiB for RV128
  - Greater TLB reach & smaller miss penalty
- Concerns about porting low-level software

```
size += 4095;
size &= ~4095;
addr = mmap(0, size, …);
```

- Internal fragmentation exacerbated by larger pages
- Transparent superpage support kind of works

- Upshot: bite the bullet; deal with 4K pages in µ-arch.

When 2^{64} Bytes Doesn't Cut It

- RV128I natively supports much larger address space
- Proposal: natural extension of RV32/RV64 schemes
- Sv68, 76, 84, …
  - 7+ level page tables
  - Mitigate TLB miss cost by skipping levels in sparse regions
  - Considered, but rejected, inverted page tables
- Also Sv44, 52, 60 for apps that want XLEN=128 but don't need large VA spaces

Physical Memory Attributes

- Most VM systems encode properties of physical memory region in the page tables
  - Cacheability, write-through-ness, consistency model
- Wrong place for this info
  - Granularity not necessarily tied to page size
  - Virtualization hole
- Less applicable in SoC era
  - Phyiscal memory attributes may be known at design time
  - Cheap coherent DMA might render cacheability control irrelevant

Interrupts & Devices

- Supervisor software sees only three kinds of interrupt
  - Timer interrupts
  - Software interrupts
  - Device interrupts
- Device interactions via virtio-style interface
  - Supports clean virtualization
  - OS isolated from driver code
- Can still support classic, virtualization-unfriendly OS by running part of OS in M-mode

Supervisor Binary Interface

- Platform-specific functionality abstracted behind SBI
  - Query physical memory map
  - Enumerate devices
  - Get hardware thread ID and # of hardware threads
  - Save/restore coprocessor state
  - Query timer properties, set up timer interrupts
  - Send interprocessor interrupts
  - Send TLB shootdowns
  - Reboot/shutdown
- Simplifies hardware acceleration
- Simplifies virtualization

- Draft SBI to be released with next priv. ISA draft

New Instructions

- Only 4 new instructions to support M+S modes
  - SFENCE.VM (synchronize page table updates)
  - ERET (exception return)
  - MRTS (redirect trap from machine to supervisor)
  - WFI (wait for interrupt)

Hypervisor Support

- Straightforward to support classic virtualization
  - OS runs in user mode
  - Privileged operations trapped & emulated
- HW-accelerated virtualization (H-mode) planned but not yet specified
- SBI abstractions should simplify H-mode design

Implementation Status

- Draft v1.7 in Spike, Rocket, BOOM, Z-scale
  - Sv32, Sv39 in Spike
  - Sv39 in Rocket, BOOM
  - Mbare => Mbb in Z-scale

- Expect to tape out Rocket implementation in Sept.

- SMP Linux port up & running on Spike

- Draft spec v1.8 this summer
- Frozen spec v2.0 this fall

Macore Moss

"We'd now like to open the floor to shorter speeches disguised as questions."