RISC-V Privileged Architecture

Andrew Waterman
SiFive Inc.
andrew@sifive.com

6th RISC-V Workshop
Shanghai Jiao Tong University
May 9, 2017
New Draft Specifications are Released

- **User ISA v2.2**
  - Improvements to documentation
  - Defines how narrow-precision FP types are represented in wider FP registers
  - Defines behavior using wider FP operations on narrow operands
- **Privileged ISA v1.10**
  - Subject of this talk
- **Specs available on Github (source & PDF)**
- Moved to editorship model
Privileged Architecture is Stable

- 1.10 keeps compatibility with 1.9.1 for machine-mode-only implementations
- Future releases should be compatible with 1.10 for supervisor ISA, too
- Caveat: these are proposals; not yet ratified by Foundation
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 Privilege Modes

- Three privilege modes
  - User (U-mode)
  - Supervisor (S-mode)
  - Machine (M-mode)

- Supported combinations of modes:
  - M (simple embedded systems)
  - M, U (embedded systems with protection)
  - M, S, U (systems running Unix-like operating systems)

- Planned support for hypervisors
Simple Embedded Systems

- Simplest implementation needs only M-mode
- No address translation
- Minimal memory protection
  - 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
Embedded Systems with Protection

- Application code is not trusted
- Add U-mode; run app code in U-mode and trusted code in M-mode
- Possibly add N extension for user-level interrupts
- Still no address translation
- Need mechanism to protect physical memory
Physical Memory Protection Unit

- Optional new feature in v1.10
- When PMP is implemented, modes below M-mode have no memory permissions by default
- Can grant R/W/X permissions on ≥4-byte granularity
- Up to 16 PMP regions
- Each PMP region is any naturally aligned power-of-2 number of bytes
- Can configure adjacent PMP registers to form an arbitrary base-and-bounds region instead

- PMPs can be locked (can’t be rewritten until reset), in which case they affect M-mode, too
Support for Unix-like Operating Systems

- Add S-mode to provide virtual memory
- Memory divided into 4 KiB base pages
- Radix-tree page tables
  - 2 levels for RV32 (Sv32)
  - 3 or 4 levels for RV64 (Sv39, Sv48)
  - Encoding space reserved for Sv57/Sv64
- Superpages possible at all levels of page table
  - e.g. 2 MiB and 1 GiB for Sv39
- Hardware PT walks specified in supervisor ISA
  - Can trap to M-mode for software TLB refill
Interaction between PMP and VM

- Physical Memory Protection and page-based virtual memory are composable
- Address translation happens first, possibly generating page-fault exceptions
- PMP unit checks translated address, possibly generating access exceptions
- Useful when S-mode code is untrusted
RISC-V Page Table Entries

- Separately controlled R, W, X permissions
  - Supports X-only pages
  - W & ~R combination reserved
- Supervisor can’t access user pages by default
- Global bit indicates the mapping belongs to all address spaces (e.g. kernel pages in a Unix system)
- Accessed/Dirty bits optionally managed by HW
  - Updates must be atomic w.r.t. permissions check
  - Complex to implement, so permit trapping when A/D not set instead
By default, S-mode can’t access user pages
- Helps detect OS/driver bugs
- Still need ability to read user memory, e.g. on system call
- Set “Supervisor Access to User Memory” bit in sstatus to read user memory, then turn it off again

Similarly, S-mode can’t read execute-only pages
- Set “Make ExecutableReadable” bit in sstatus to override
- Useful for illegal-instruction trap handlers

S-mode can enable/disable VM and choose pagetable depth in satp register
RISC-V Interrupt Design Goals

- Simplicity
- Support all kinds of platforms from microcontrollers to virtualized servers
- Enable tradeoffs between performance and implementation cost
- Flexibility to support specialized needs
Interrupt Uses in Different Applications

- High-performance Unix-like systems
  - Interrupt handling small fraction of processing time
    - Fast cores, smart devices
  - Minimal interrupt handler
  - Scheduling in software

- Low/mid embedded systems
  - Interrupt handling significant fraction of processor time
    - Slow cores, dumb devices
  - Significant fraction of code in handlers
  - Interrupt controller acts as task scheduler

- High-performance real-time systems
  - Can’t waste time on interrupt overhead
    - Handlers poll I/O devices with regular heartbeat

- And everything in between
Categorizing Sources of RISC-V Interrupts

- **Local Interrupts**
  - Directly connected to one hart
  - No arbitration between harts to service
  - Determine source directly through xcause CSR
  - Only two standard local interrupts (software, timer)

- **Global (External) Interrupts**
  - Routed via Platform-Level Interrupt Controller (PLIC)
  - PLIC arbitrates between multiple harts claiming interrupt
  - Read of memory-mapped register returns source
Machine Interrupt Pending CSR (mip)

(Add Non-Standard
Local Interrupts Here)

- **mip** reflects pending status of interrupts for hart
- Separate interrupts for each supported privilege level (M/S/U)
- User-level interrupt handling ("N") optional feature when U-mode present (discussed later)
Software Interrupts

- **MSIP**
  - Only writeable in machine-mode via memory-mapped control register (mapping is platform-specific)
  - One hart can write to different hart’s MSIP register
  - Mechanism for inter-hart interrupts

- **SSIP and USIP**
  - Hart can only write bit xSIP in own mip register when running at privilege mode x or greater

- **App/OS can only perform inter-hart interrupts via ABI/SBI calls**
  - Destination virtual hart might be descheduled
  - Interrupts virtualized by M-mode software using MSIP
Timer Interrupts

- **MTIP**
  - Single 64-bit real-time hardware timer and comparator in M-mode
  - MTIP set when \texttt{mtime} \geq \texttt{mtimecmp}
  - MTIP cleared by writing new \texttt{mtimecmp} value

- **STIP and UTIP**
  - M-mode multiplexes single hardware timer and comparator for lower-privilege modes on same hart
  - ABI/SBI calls to set up timer
  - M-mode software writes/clears STIP/UTIP

- Most systems will also have other hardware timers attached via PLIC etc.
External Interrupts

- MEIP, SEIP, UEIP
  - Inputs from a Platform-Level Interrupt Controller (PLIC)
  - Each privilege mode has its own input from PLIC
  - Interrupts cleared with loads/stores to PLIC
  - Software can inject SEIP and UEIP interrupts to support virtualizing the PLIC
Platform-Level Interrupt Controller (PLIC)
Platform-Level Interrupt Controller

Interrupt 1 Signals

Gateway

Priority

Interrupt Request

IP

Interrupt 2 Signals

Gateway

Priority

Interrupt Request

IP

PLIC Gateways

PLIC Core

Interrupt Notification

Interrupt ID

Interrupt Notification

Interrupt ID

To Target 0

To Target 1

Max Pri.

Threshold

Max ID

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE

EIP

IE
Machine Interrupt Enable CSR (mie)

- mie mirrors layout of mip
- provides per-interrupt enables
- Also, global interrupt enables in mstatus for each privilege mode
- Interrupts always disabled for lower privilege modes; always enabled for higher privilege modes
All interrupts trap to M-mode by default

- `mcause` register indicates which interrupt occurred
- M-mode can redirect to appropriate privilege level using MRET instruction

<table>
<thead>
<tr>
<th>Interrupt</th>
<th>Exception Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>User software interrupt</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Supervisor software interrupt</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>Machine software interrupt</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>User timer interrupt</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>Supervisor timer interrupt</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>Machine timer interrupt</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>User external interrupt</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>Supervisor external interrupt</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>Machine external interrupt</td>
</tr>
<tr>
<td>1</td>
<td>≥12</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
Optional Interrupt Handler Delegation

- Can delegate interrupt (and exception) handling to lower privilege level to reduce overhead
- `mideleg` has same layout as `mip`
- If a bit is set in `mideleg` then corresponding interrupt delegated to next lowest privilege level (S or U)
- Can be delegated again using `sideleg`
- Once delegated, the interrupt will not affect current privilege level (`mie` setting ignored)
Hypervisor Status

- Previous spec sketched a fourth privilege mode, H, above S (M/H/S/U)
- Designed for Type-1 hypervisor support
- Feedback from community led us to pursue HW support for Type-2 hypervisors (like KVM) instead
  - Still works well for Type-1 hypervisors
- Plan is to make a full proposal by September
Implementation Status

- Spike and UCB Rocket-Chip conform to v1.10
- Linux port to v1.10 works with Spike/Rocket
  - Working on upstreaming the Linux kernel
- Upstream GCC and binutils ports are compatible
Questions?

Specs available at
https://github.com/riscv/riscv-isa-manual