



# RISC-V 101

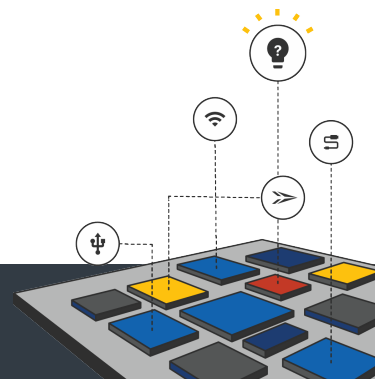
An Introduction to RISC-V Architecture  
for Embedded Developers

Drew Barbier – September 2017  
[drew@sifive.com](mailto:drew@sifive.com)



# RISC-V 101

- This presentation is targeted at embedded developers who want to learn more about RISC-V
- At the end of this presentation you should have a basic understanding of RISC-V fundamentals and know where to look for more information



# 3 Part Webinar Series

- RISC-V 101
  - The Fundamentals of RISC-V architecture
- Introduction to SiFive Coreplex IP
  - October 17<sup>th</sup>, 2017
- Getting Started with SiFive Coreplex IP
  - November 2017



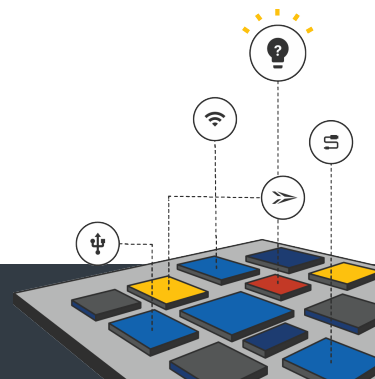
Study



Evaluate



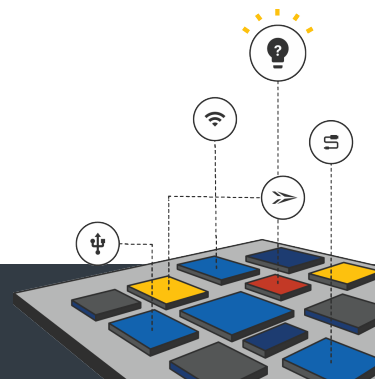
Buy



# How to ask questions



The screenshot shows a video player interface for a SiFive presentation. At the top, a green status bar reads "You are viewing Drew Barbier's screen" with a "View Options" dropdown. The main slide content includes the SiFive logo, the title "RISC-V 101" in large yellow font, the subtitle "An Introduction to RISC-V Architecture for Embedded Developers", and the presenter information "Drew Barbier – September 2017" and "drew@sifive.com". The video player controls at the bottom include buttons for Mute, Start Video, Participants (3), Q&A (highlighted with a red box), Polls, Share Screen, Chat, Pause/Stop Recording, and More. A "Recording..." indicator is visible in the top left of the player area.







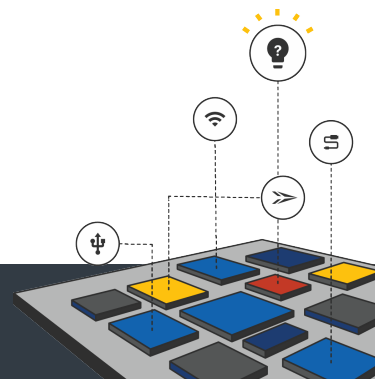
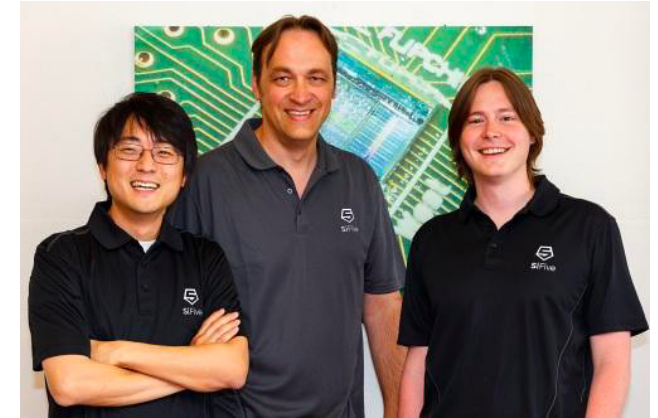
# RISC-V Introduction

Krste Asanovic




# RISC-V Origin Story (pronounced: risk five)

- Started as a research project in 2010 at UC Berkeley
  - Commercial ISAs were too complex and presented IP legal issues
- RISC-V defines an ISA specification
  - 2 Main specifications: User Level, and Privileged Level Spec
  - User Level is currently frozen at version 2.0; released in May 2014
  - Privileged Level Architecture is currently at version 1.10
- RISC-V is available freely under a permissive license
- RISC-V is not...
  - A Company
  - A CPU implementation



# RISC-V Foundation – riscv.org

- A non-profit RISC-V foundation was formed in August 2015 to publicly govern the ISA
  - Similar to the Linux Foundation
- >65 member companies representing a wide range of markets
- The Foundation creates Working Groups to guide future development of the Architecture

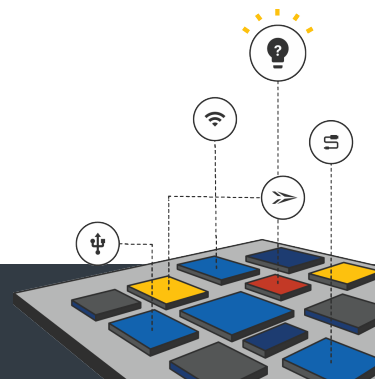


## Foundation Mission Statement

*The RISC-V Foundation is a non-profit consortium chartered to standardize, protect, and promote the free and open RISC-V instruction set architecture together with its hardware and software ecosystem for use in all computing devices.*

29 November 2016 RISC-V Foundation

5





# RISC-V Basics

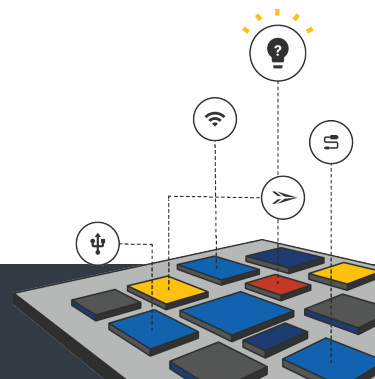


# RISC-V Instruction Set Architectures

- RISC-V uses a standard naming convention to describe the ISAs supported in a given implementation
- ISA Name format: **RV[###][abc....xyz]**
  - RV – Indicates a RISC-V architecture
  - [###] - {32, 64, 128} indicate the width of the integer register file and the size of the user address space
  - [abc...xyz] – Used to indicate the set of extensions supported by an implementation.

Hello  
my Architecture is

**RV64GC**



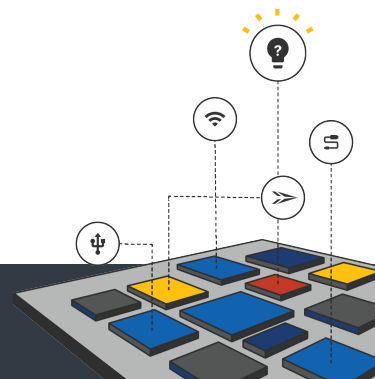
# The Standard Extensions

- Extensions add instructions
- “I” for Integer
  - The only required Extension in a RISC-V implementation
- RISC-V allows for custom, “Non-Standard”, extensions in an implementation
- Putting it all together (examples)
  - RV32I – The most basic RISC-V implementation
  - RV32IMAC – Integer + Multiply + Atomic + Compressed
  - RV64GC – 64bit IMAFDC
  - RV64GCXext – IMAFDC + a non-standard extension

Extension	Description
I	Integer
M	Integer Multiplication and Division
A	Atomics
F	Single-Precision Floating Point
D	Double-Precision Floating Point
G	General Purpose = IMAFD
C	16-bit Compressed Instructions
Non-Standard User-Level Extensions	
Xext	Non-standard extension “ext”

Common RISC-V Standard Extensions

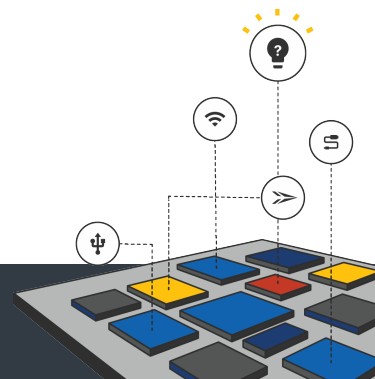
\*Not a complete list



# Register File

- 32 Integer Registers,
  - Optional 32 FP registers
- Width of Registers is determined by ISA
- RISC-V Application Binary Interface (ABI) defines standard functions for registers
  - Allows for software interoperability
- GCC assembler accepts `X##` names or ABI names

Register	ABI Name	Description	Saver
x0	zero	Hard-wired zero	-
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	-
x4	tp	Thread pointer	-
x5-7	t0-2	Temporaries	Caller
x8	s0/fp	Saved register/Frame pointer	Callee
x9	s1	Saved register	Callee
x10-11	a0-1	Function Arguments/return values	Caller
x12-17	a2-7	Function arguments	Caller
x18-27	s2-11	Saved registers	Callee
x28-31	t3-6	Temporaries	Caller

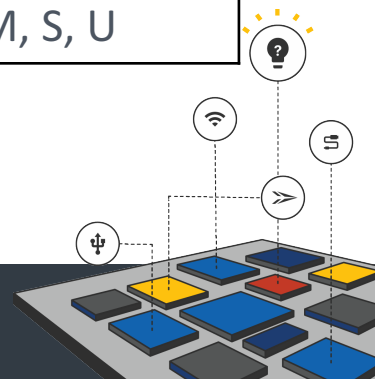


# RISC-V Modes

- RISC-V Privileged Specification defines 3 levels of privilege, called Modes
- Machine mode is the highest privileged mode and the only required mode
  - Allows for a range of targeted implementations
- Machine and Supervisor modes each have Control and Status Registers (CSRs)
  - More on these later

RISC-V Modes		
Level	Name	Abbr.
0	User/Application	U
1	Supervisor	S
Reserved		
3	Machine	M

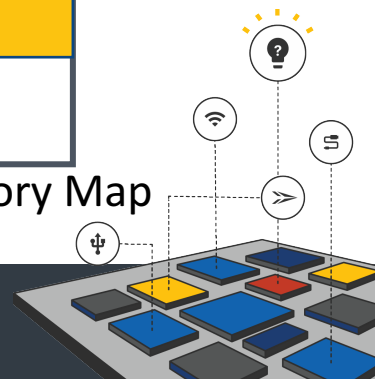
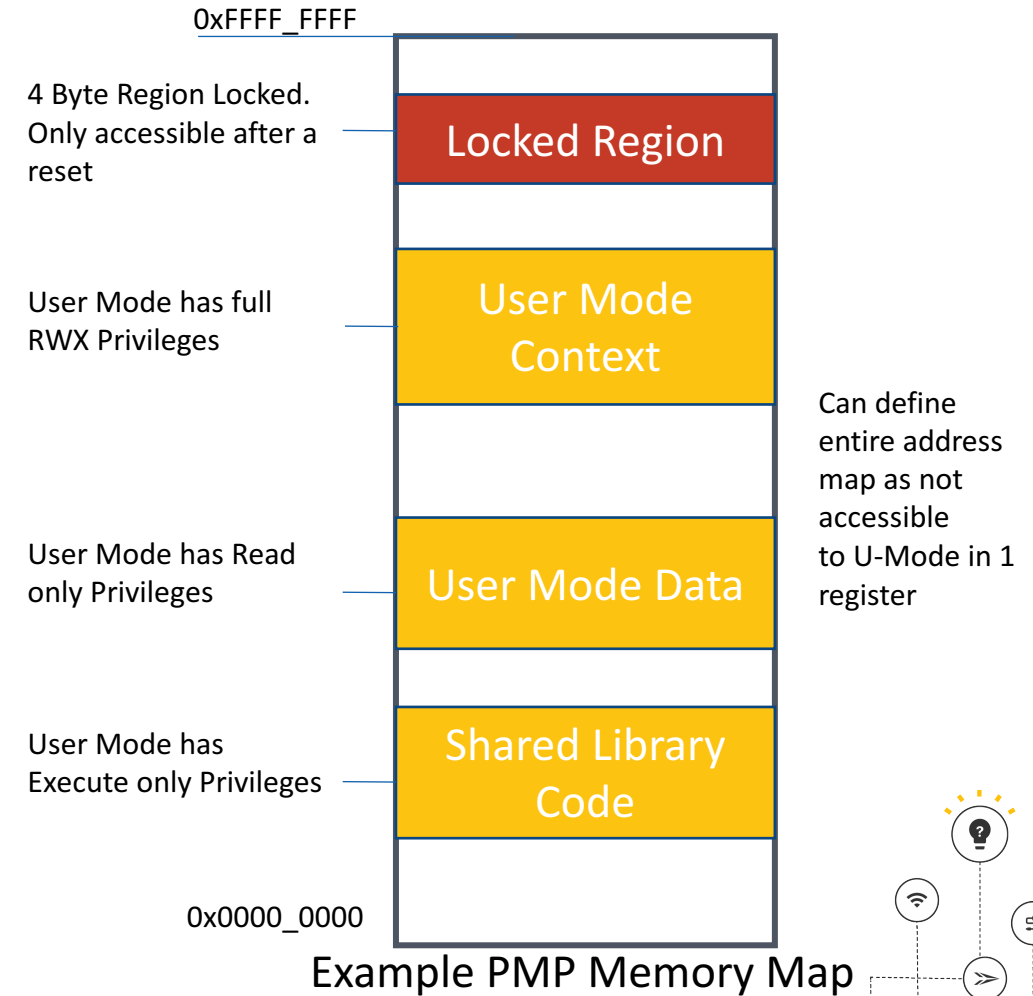
Supported Combinations of Modes	
Supported Levels	Modes
1	M
2	M, U
3	M, S, U





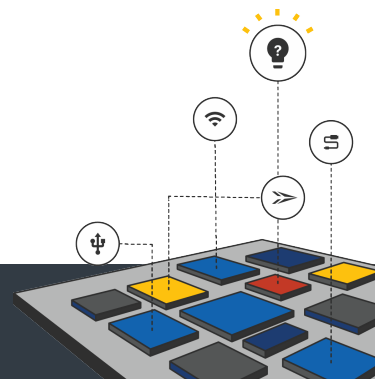
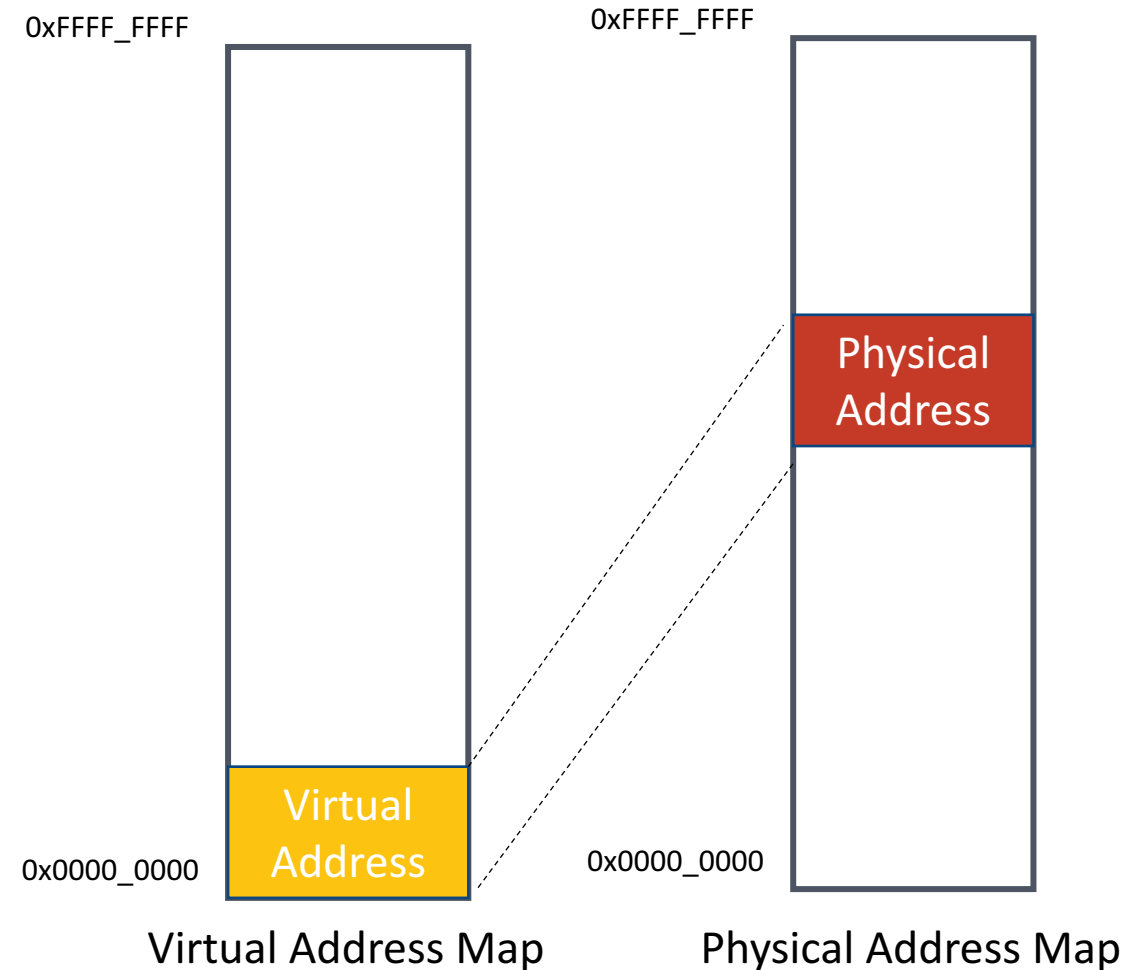
# Physical Memory Protection (PMP)

- Can be used to enforce access restrictions on less privileged modes
  - Prevent User Mode software from accessing unwanted memory
- Ability to Lock a region
  - A locked region enforces permissions on all accesses, including M-Mode
  - Only way to unlock a region is a Reset
- Up to 16 regions with a minimum region size of 4 bytes



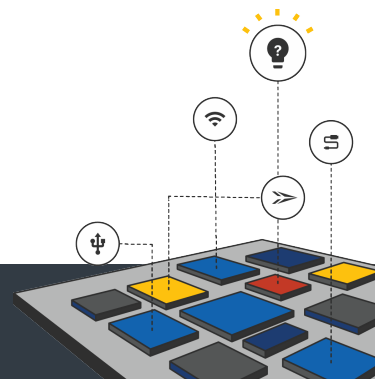
# Virtual Memory

- RISC-V has support for Virtual Memory allowing for sophisticated memory management and OS support (Linux).
- Requires an S-Mode implementation
- Sv32
  - 32bit Virtual Address
  - 4KiB, 4MiB page tables (2 Levels)
- Sv39 (requires an RV64 implementation)
  - 39bit Virtual Address
  - 4KiB, 2MiB, 1GiB page tables (3 Levels)
- Page Tables also contain access permission attributes



# Terms

- Hart – HARdware Thread
  - For the context of this webinar, hart = core



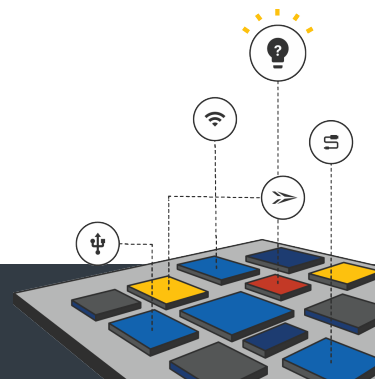
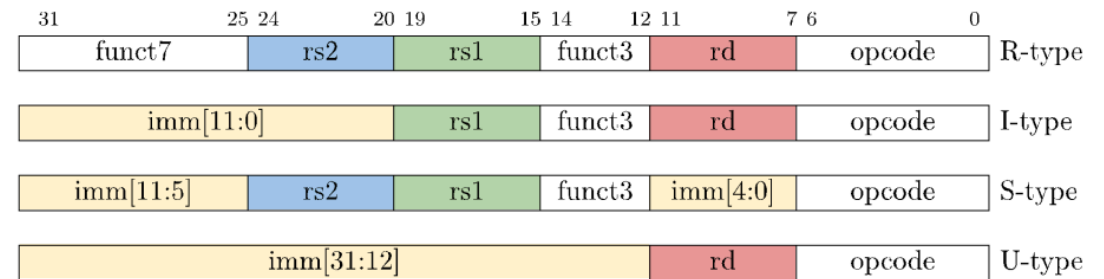


# RISC-V Instructions



# Base Integer ISA Encoding

- 32-bit fixed-width, naturally aligned instructions
- **rd/rs1/rs2** in fixed location, no implicit registers
- Immediate field (instr[31]) always sign-extended
- Instruction Encoding Types
  - R-type – Register
  - I-type – Immediate
  - S-type – Stores
  - U-Type – Loads with immediate



Base Integer Instructions (32 64 128)				
Category	Name	Fmt	RV{32 64 128}I Base	
<b>Loads</b>	Load Byte	I	LB	rd,rs1,imm
	Load Halfword	I	LH	rd,rs1,imm
	Load Word	I	L{W D Q}	rd,rs1,imm
	Load Byte Unsigned	I	LBU	rd,rs1,imm
	Load Half Unsigned	I	L{H W D}U	rd,rs1,imm
<b>Stores</b>	Store Byte	S	SB	rs1,rs2,imm
	Store Halfword	S	SH	rs1,rs2,imm
	Store Word	S	S{W D Q}	rs1,rs2,imm
<b>Shifts</b>	Shift Left	R	SLL{W D}	rd,rs1,rs2
	Shift Left Immediate	I	SLLI{W D}	rd,rs1,shamt
	Shift Right	R	SRL{W D}	rd,rs1,rs2
	Shift Right Immediate	I	SRLI{W D}	rd,rs1,shamt
	Shift Right Arithmetic	R	SRA{W D}	rd,rs1,rs2
	Shift Right Arith Imm	I	SRAI{W D}	rd,rs1,shamt
<b>Arithmetic</b>	ADD	R	ADD{W D}	rd,rs1,rs2
	ADD Immediate	I	ADDI{W D}	rd,rs1,imm
	SUBtract	R	SUB{W D}	rd,rs1,rs2
	Load Upper Imm	U	LUI	rd,imm
	Add Upper Imm to PC	U	AUIPC	rd,imm
<b>Logical</b>	XOR	R	XOR	rd,rs1,rs2
	XOR Immediate	I	XORI	rd,rs1,imm
	OR	R	OR	rd,rs1,rs2
	OR Immediate	I	ORI	rd,rs1,imm
	AND	R	AND	rd,rs1,rs2
	AND Immediate	I	ANDI	rd,rs1,imm
<b>Compare</b>	Set <	R	SLT	rd,rs1,rs2
	Set < Immediate	I	SLTI	rd,rs1,imm
	Set < Unsigned	R	SLTU	rd,rs1,rs2
	Set < Imm Unsigned	I	SLTIU	rd,rs1,imm
<b>Branches</b>	Branch =	SB	BEQ	rs1,rs2,imm
	Branch ≠	SB	BNE	rs1,rs2,imm
	Branch <	SB	BLT	rs1,rs2,imm
	Branch ≥	SB	BGE	rs1,rs2,imm
	Branch < Unsigned	SB	BLTU	rs1,rs2,imm
	Branch ≥ Unsigned	SB	BGEU	rs1,rs2,imm
<b>Jump &amp; Link</b>	J&L	UJ	JAL	rd,imm
	Jump & Link Register	I	JALR	rd,rs1,imm
<b>Synch</b>	Synch thread	I	FENCE	
	Synch Instr & Data	I	FENCE.I	
<b>System</b>	System CALL	I	SCALL	
	System BREAK	I	SBREAK	
<b>Counters</b>	Read CYCLE	I	RDCYCLE	rd
	Read CYCLE upper Half	I	RDCYCLEH	rd
	Read TIME	I	RDTIME	rd
	Read TIME upper Half	I	RDTIMEH	rd
	Read INSTR RETired	I	RDINSTRET	rd
	Read INSTR upper Half	I	RDINSTRETH	rd

+14  
Privileged

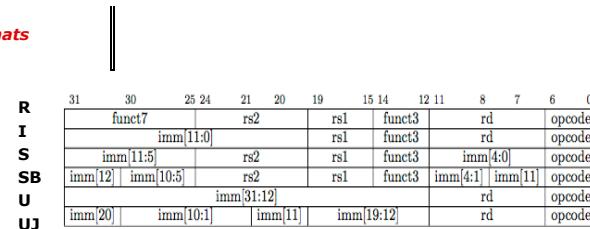
+ 8 for M

+ 34  
for F, D, Q

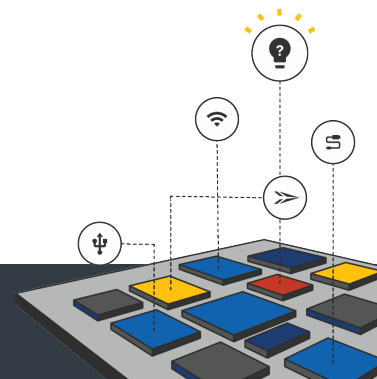
+ 46 for C

+ 11 for A

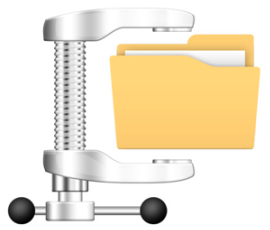
32-bit Instruction Formats



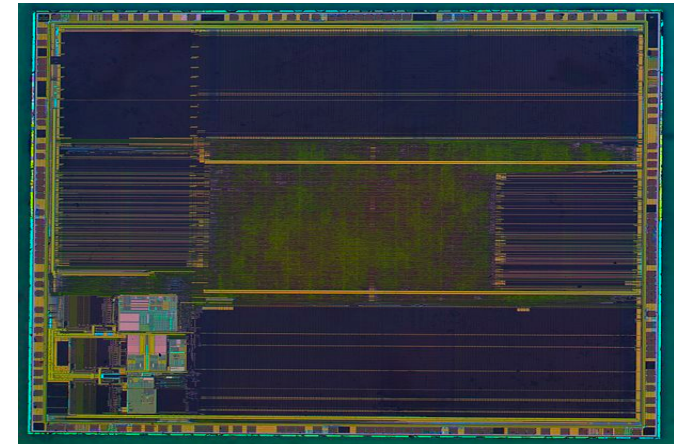
RV32I



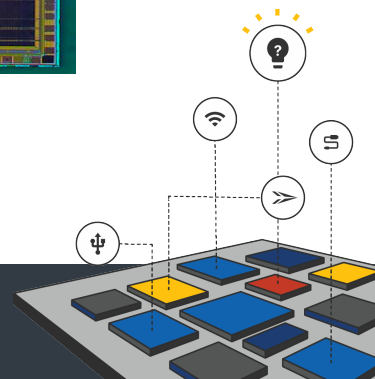
# Compressed Instructions (C Extension)



- Most of the base integer instructions “Compress” to 16-bit equivalents
  - 1:1 mapping of compressed instructions to standard instructions
- Smaller code size can reduce cost in embedded systems
- Smaller code size can increase performance in Cache based cores
- RV64 can also use the C Extension

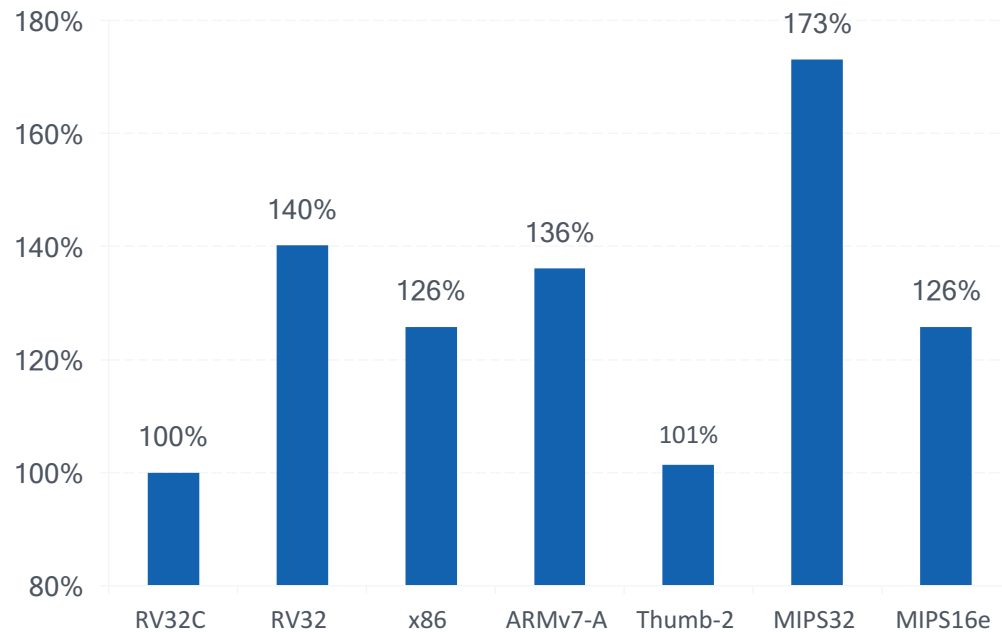


A Microcontroller

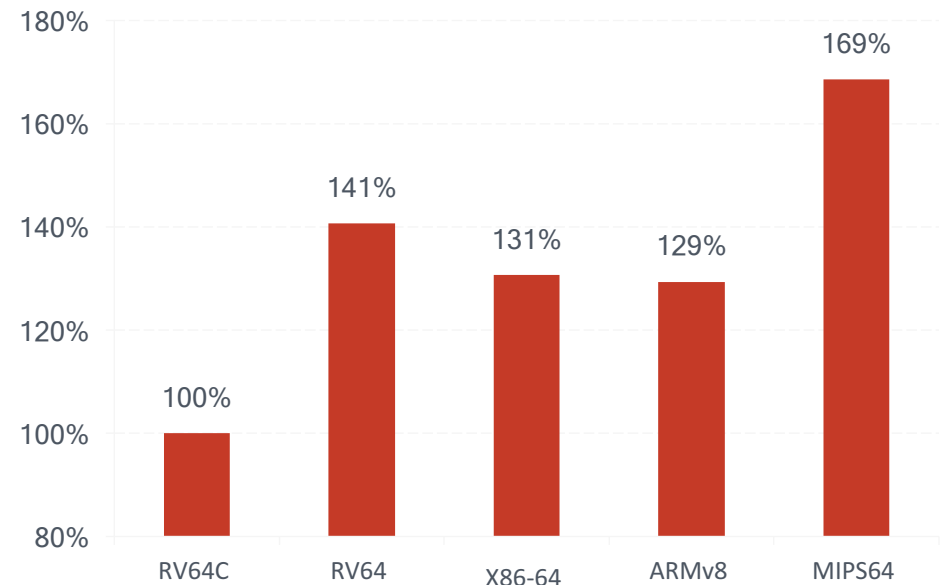


# SPECint2006 compressed code size with save/restore optimization (relative to “standard” RVC)

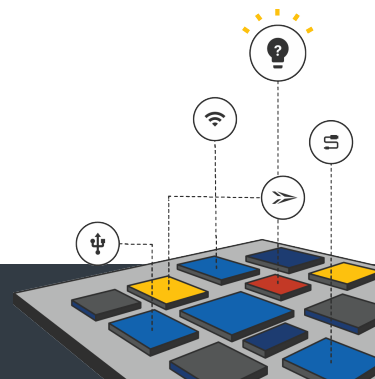
## 32-bit Architectures



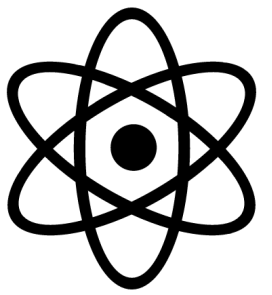
## 64-bit Architectures



- RISC-V now smallest ISA for 32- and 64-bit addresses
- All results with same GCC compiler and options





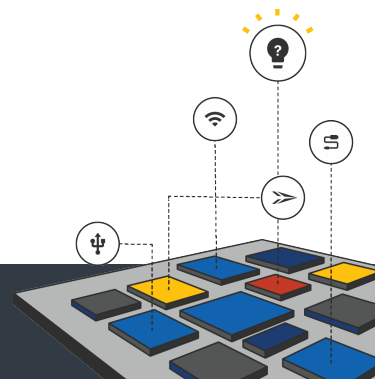


# Atomics (A Extension)

- Atomic memory operations (AMO) perform a Read-Modify-Write in a single Atomic instruction
  - Logical, Arithmetic, Swap
  - Acquire (aq) and Release (rl) bits for release consistency
- Load-Reserved/Store-Conditional pairs
  - Guaranteed forward progress for short sequences

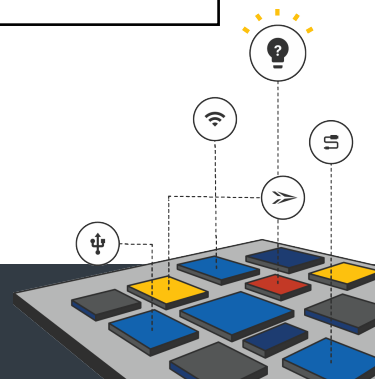
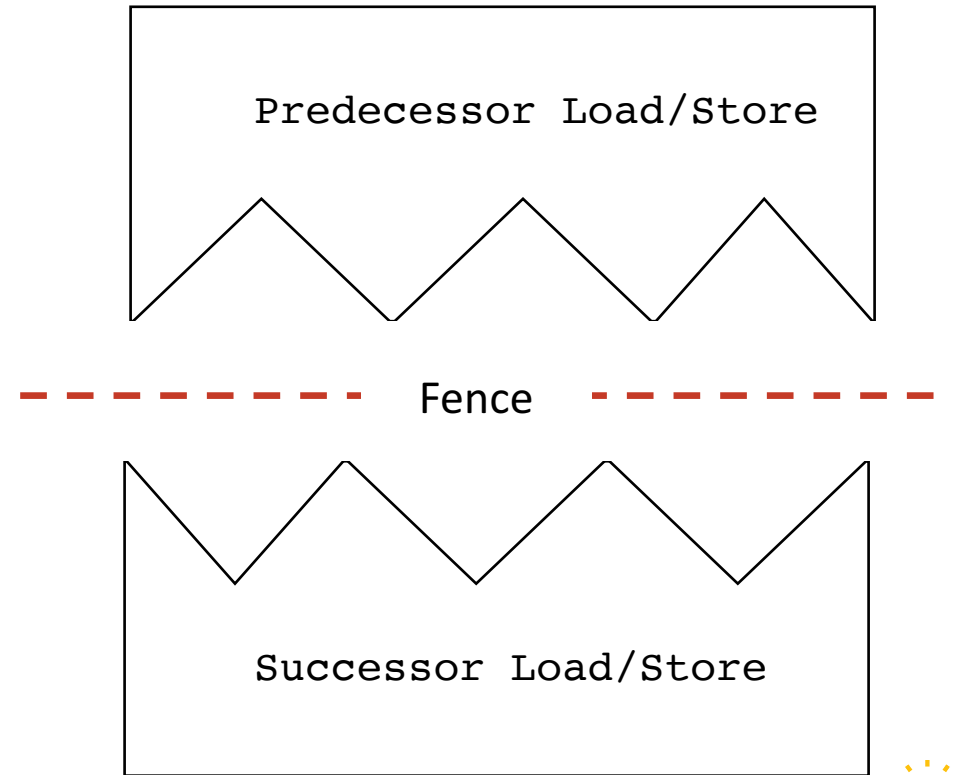
```
li t0, 1 # Initialize swap value.
again:
  amoswap.w.aq t0, t0, (a0) # Attempt to acquire lock.
  bnez t0, again # Retry if held.
  # ...
  # Critical section.
  # ...
  amoswap.w.rl x0, x0, (a0) # Release lock by storing 0.
```

Example RISC-V Spinlock



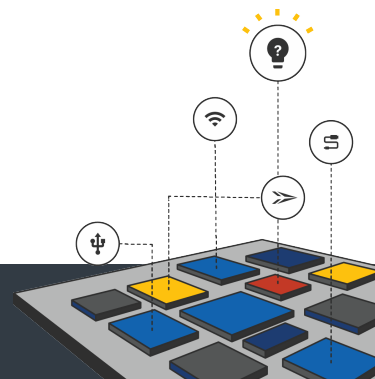
# Fence Instructions

- Fences are used to enforce order on device I/O and memory accesses
- FENCE instruction format
  - FENCE predecessor, successor
  - Predecessor/successor can be
    - R,W,I,O
  - FENCE RWIO, RWIO – full barrier



# CSR and ECALL Instructions

- Control and Status Registers (CSRs) have their own dedicated instructions :
  - Read/Write
  - Read and Set bit
  - Read and Clear bit
- Environment Call instruction used to transfer control to the execution environment and a higher privileged mode
  - Triggers a synchronous Interrupt (discussed later)
  - Example: User mode program can use an ECALL to transfer control to a Machine mode OS kernel



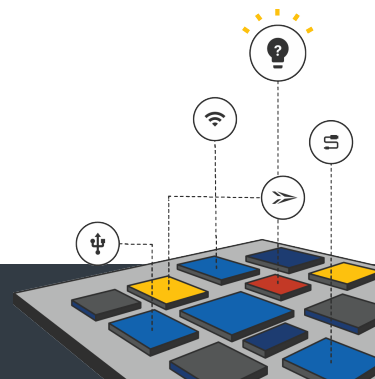


# RISC-V Control and Status Registers (CSR)



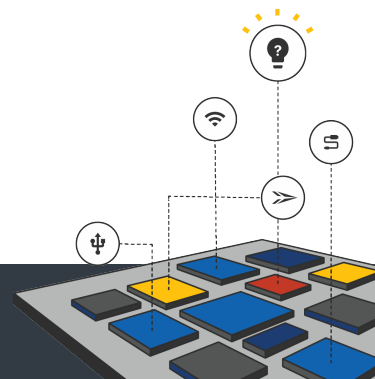
# What are Control and Status Registers (CSRs)

- CSRs are Registers which contain the working state of a RISC-V machine
- CSRs are specific to a Mode
  - Machine Mode has ~17 CSRs (not including performance monitor CSRs)
  - Supervisor Mode has a similar number, though most are subsets of their equivalent Machine Mode CSRs
    - Machine Mode can also access Supervisor CSRs
- CSRs are defined in the RISC-V privileged specification
  - We will cover a few key CSRs here



# Identification CSRs

- *misa* – Machine ISA Register
  - Reports the ISA supported by the hart (i.e. RV32IMAC)
- *mhartid* – Machine hart ID
  - Integer ID of the Hardware Thread
- *mvendorid* – Machine Vendor ID
  - JEDEC Vendor ID
- *marchid* – Machine Architecture ID
  - Used along with *mvendorid* to identify a implementation. No format specified
- *mimpid* - Machine Implementation ID
  - Implementation defined format



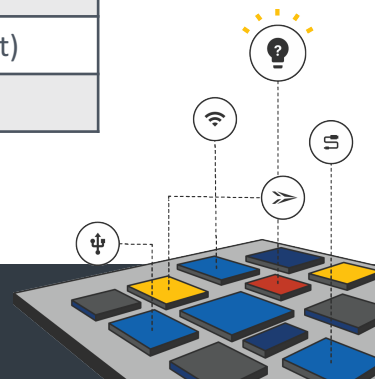
# Machine Status (*mstatus*) - The Most Important CSR

- Keeps track of and controls the hart's current operating state

Bits	Field Name	Description
0	UIE	User Interrupt Enable
1	SIE	Supervisor Interrupt Enable
2	Reserved	
3	MIE	Machine Interrupt Enable
4	UPIE	User Previous Interrupt Enable
5	SPIE	Supervisor Previous Interrupt Enable
6	Reserved	
7	MPIE	Machine Previous Interrupt Enabler
8	SPP	Supervisor Previous Privilege
[10:9]	Reserved	
[12:11]	MPP	Machine Previous Privilege

Bits	Field Name	Description
[14:13]	FS	Floating Point State
[16:15]	XS	User Mode Extension State
17	MPRIV	Modify Privilege (access memory as MPP)
18	SUM	Permit Supervisor User Memory Access
19	MXR	Make Executable Readable
20	TVM	Trap Virtual memory
21	TW	Timeout Wait (traps S-Mode wfi)
22	TSR	Trap SRET
[23:30]	Reserved	
[31]	SD	State Dirty (FS and XS summary bit)

RV32 *mstatus* CSR





# Timer CSRs

- *mtime*

- RISC-V defines a requirement for a real-time counter exposed as a memory mapped register
- There is no frequency requirement on the timer, but
  - It must run at a constant frequency
  - The platform must expose frequency

Bits	Field Name	Description
[63:0]	mtime	Machine Time Register

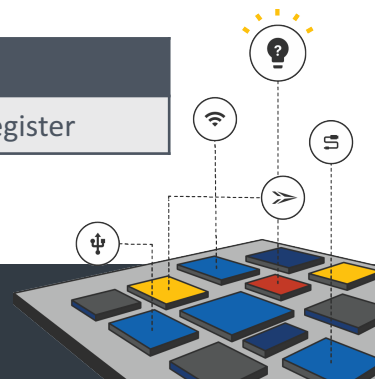
*mtime* CSR

- *mtimecmp*

- RISC-V defines a memory mapped timer compare register
- Triggers an interrupt when *mtime* is greater than or equal to *mtimecmp*

Bits	Field Name	Description
[63:0]	mtimecmp	Machine Time Compare Register

*mtimecmp* CSR





# Supervisor CSRs

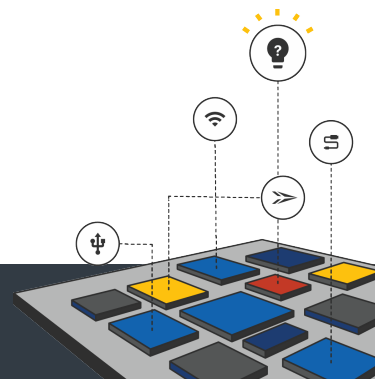
- Most of the Machine mode CSRs have Supervisor mode equivalents
  - Supervisor mode CSRs can be used to control the state of Supervisor and User Modes.
  - Most equivalent Supervisor CSRs have the same mapping as Machine mode without Machine mode control bits
  - *sstatus*, *stvec*, *sip*, *sie*, *sepc*, *scause*, *satp*, and more
- *satp* - Supervisor Address Translation and Protection Register
  - Used to control Supervisor mode address translation and protection

Bits	Field Name	Description
[21:0]	PPN	Physical Page Number of the root page table
[30:22]	ASID	Address Space Identifier
31	MODE	MODE=1 uses Sv32 Address Translation

*RV32 satp CSR*

Bits	Field Name	Description
[43:0]	PPN	Physical Page Number of the root page table
[59:44]	ASID	Address Space Identifier
[63:60]	MODE	Encodings for Sv32, Sv39, Sv48

*RV64 satp CSR*



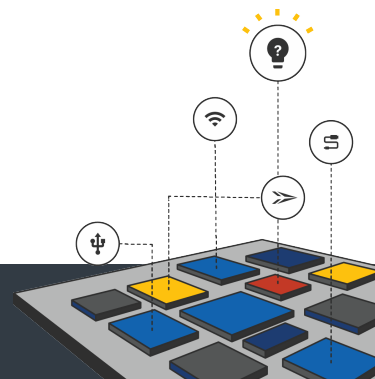


# RISC-V Interrupts



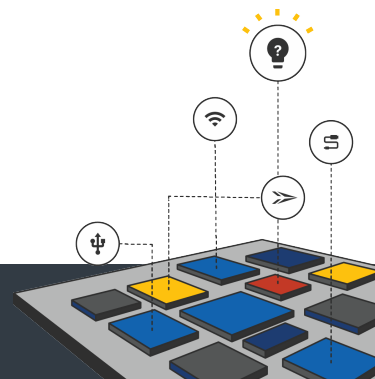
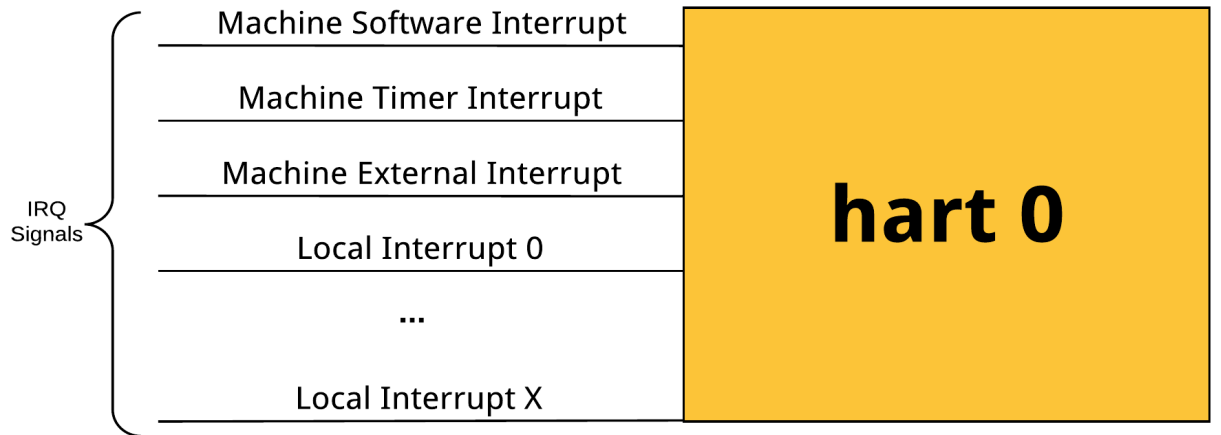
# What is an Interrupt in RISC-V

- An event which triggers a change in the program flow
- 2 Types of Interrupts
  - Synchronous – A CPU instruction which generates an Exception (ECALL, Faults, etc...)
  - Asynchronous – An Exception which is triggered by an external event (peripheral and other IO devices)



# RISC-V Interrupts

- RISC-V defines the following interrupts
  - Software
  - Timer
  - External
  - Local
- Local interrupts are optional and implementation specific
  - Can be used for peripheral interrupts
  - Great for latency sensitive embedded systems



# Interrupt Cause CSR

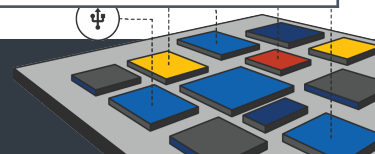
- Interrupts are identified by reading the *mcause* CSR
- The *interrupt* field determines if an Interrupt was Synchronous or Asynchronous

Bits	Field Name	Description
XLEN-1	Interrupt	Identifies if an interrupt was synchronous or asynchronous
[XLEN-2:0]	Exception Code	Identifies the exception

*mcause CSR*

Interrupt = 1 (Asynchronous)	
Exception Code	Description
0	User Software Interrupt
1	Supervisor Software Interrupt
2	Reserved
3	Machine Software Interrupt
4	User Timer Interrupt
5	Supervisor Timer Interrupt
6	Reserved
7	Machine Timer Interrupt
8	User External Interrupt
9	Supervisor External Interrupt
10	Reserved
11	Machine External Interrupt
12 - 15	Reserved
≥16	Local Interrupt X

Interrupt = 0 (Synchronous)	
Exception Code	Description
0	Instruction Address Misaligned
1	Instruction Access Fault
2	Illegal Instruction
3	Breakpoint
4	Load Address Misaligned
5	Load Access Fault
6	Store/AMO Address Misaligned
7	Store/AMO Access Fault
8	Environment Call from U-mode
9	Environment Call from S-mode
10	Reserved
11	Environment Call from M-mode
12	Instruction Page Fault
13	Load Page Fault
14	Reserved
15	Store/AMO Page Fault
≥16	Reserved

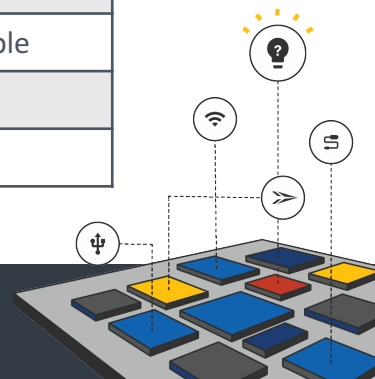


# Machine Interrupt-Enable and Pending CSRs

- *mie* used to enable/disable a given interrupt
- *mip* indicates which interrupts are currently pending
  - Can be used for polling
- Lesser privilege bits in *mip* are writeable
  - i.e. Machine mode software can be used to generate a supervisor interrupt by setting the STIP bit
- *mip* has the same mapping as *mie*

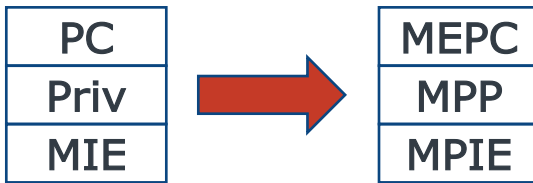
Bits	Field Name	Description
0	USIE	User Software Interrupt Enable
1	SSIE	Supervisor Software Interrupt Enable
2	Reserved	
3	MSIE	Machine Software Interrupt Enable
4	UTIE	User Timer Interrupt Enable
5	STIE	Supervisor Timer Interrupt Enable
6	Reserved	
7	MTIE	Machine Timer Interrupt Enable
8	UEIE	User External Interrupt Enable
9	SEIE	Supervisor External Interrupt Enable
10	Reserved	
11	MEIE	Machine External Interrupt Enable
12-15	Reserved	
≥16	LIE	Local Interrupt Enable

*mie CSR*

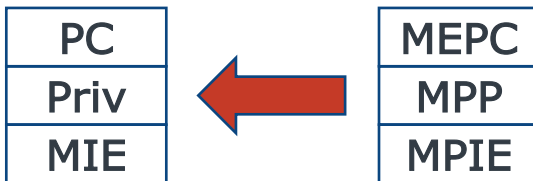


# Interrupt Handler – Entry and Exit

- On entry, the RISC-V hart will
  - Save the current state



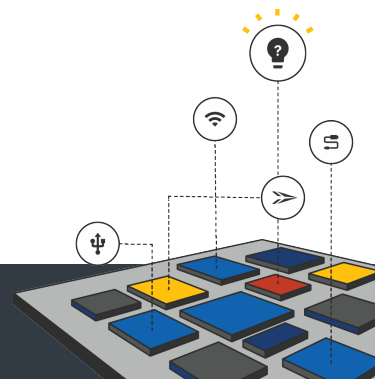
- Then set PC = *mtvec*
- MRET instruction restores state



- Typical interrupt handler software will

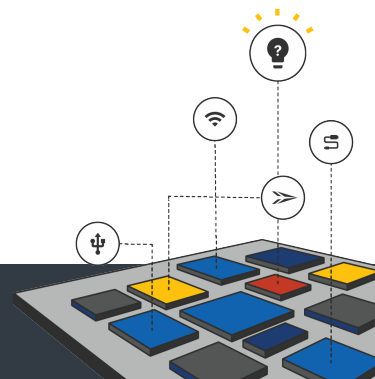
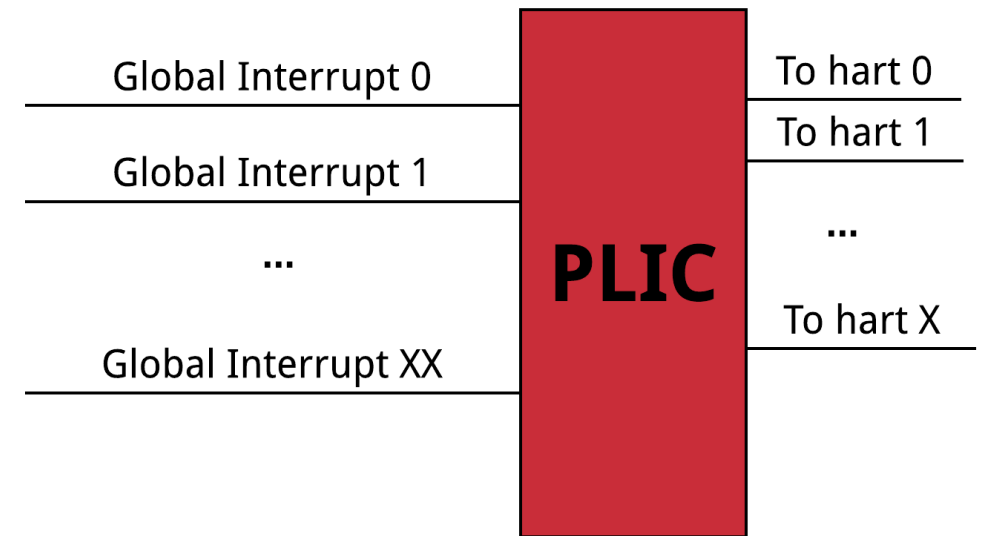
```
Push Registers
...
async_irq = mcause.msb
if async_irq
    branch async_handler[mcause.code]
else
    branch synch_handler[mcause.code]
...
Pop Registers
MRET
```

*Interrupt handler pseudo code*



# RISC-V Global Interrupts

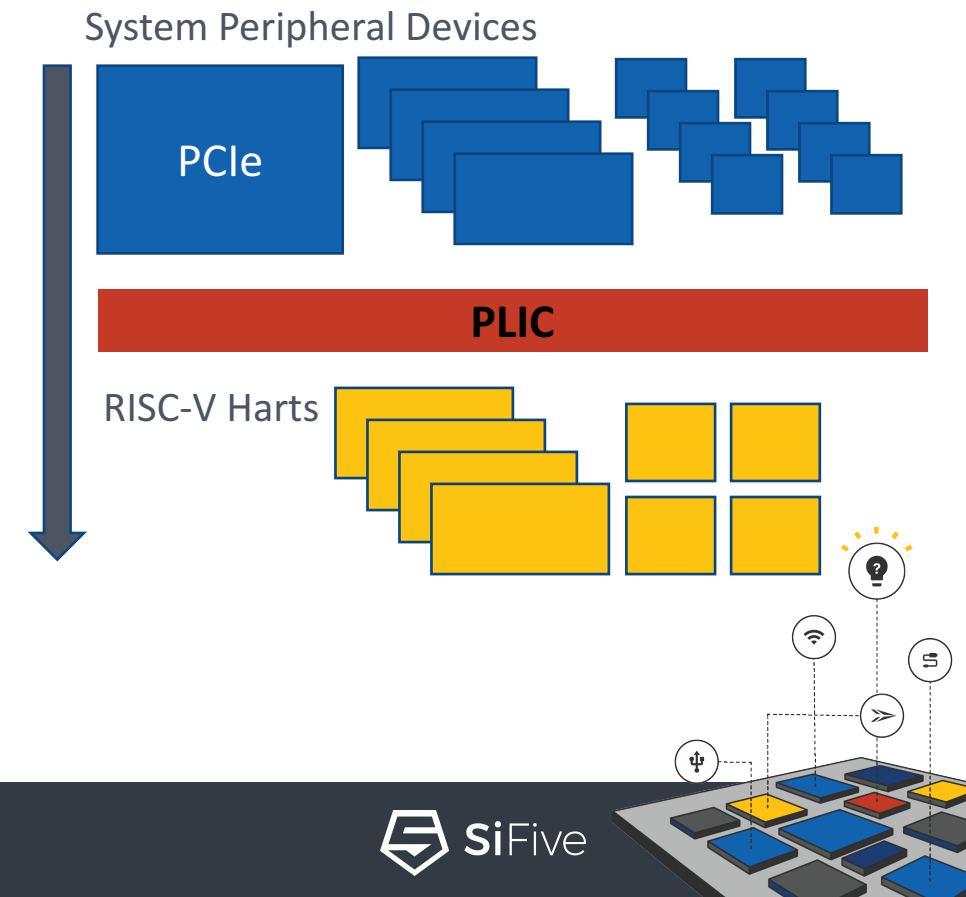
- RISC-V defines **Global Interrupts** as a Interrupt which can be routed to any *hart* in a system
- Global Interrupts are prioritized and distributed by the Platform Level Interrupt Controller (PLIC)
- The PLIC is connected to the External Interrupt signal for 1 or more *harts* in an implementation



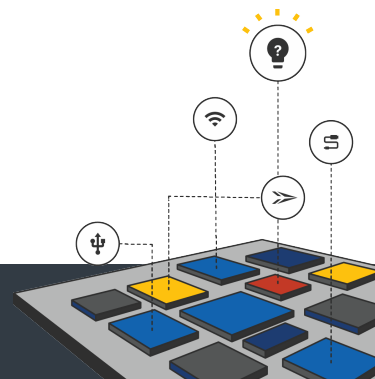
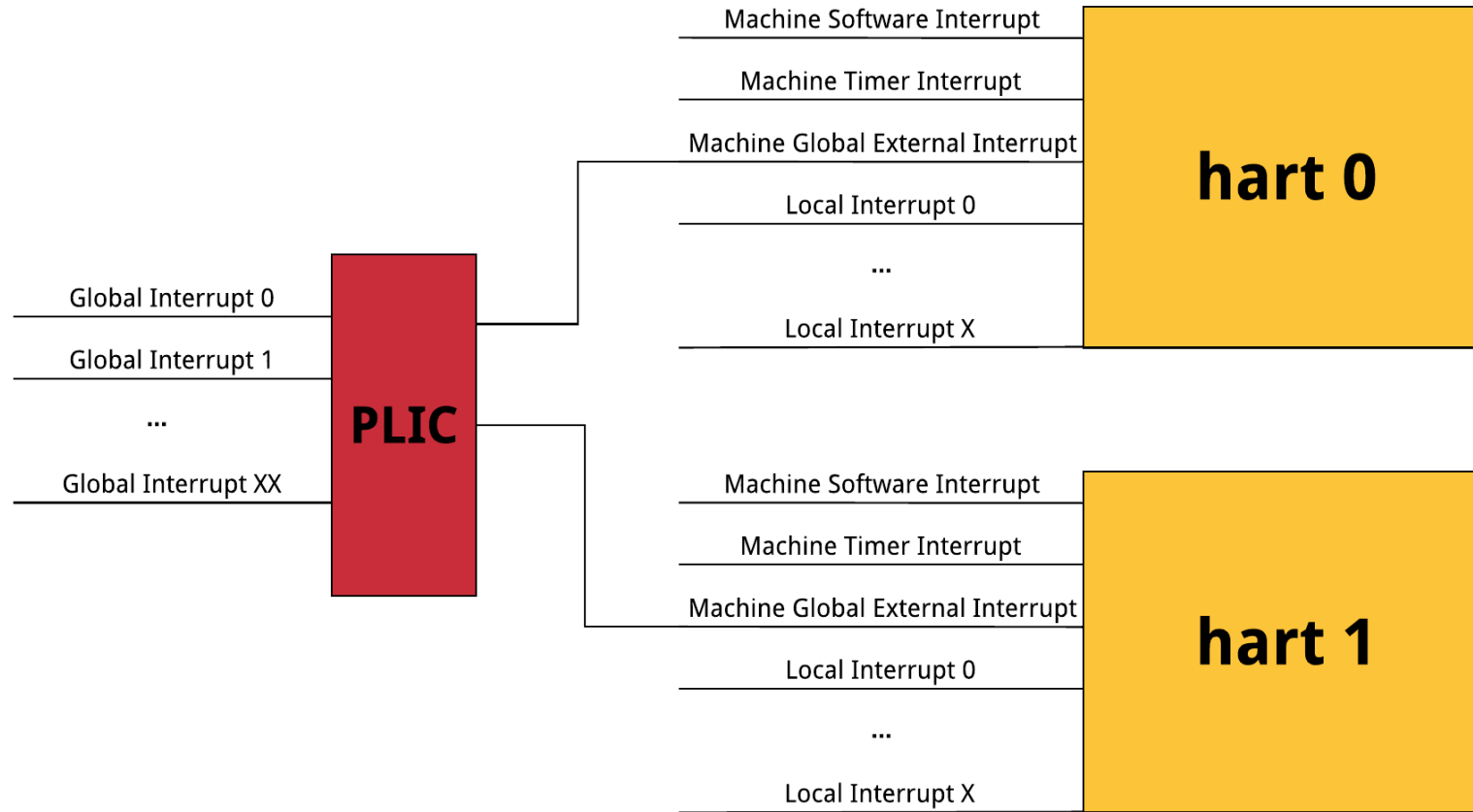


# Platform Level Interrupt Controller

- The PLIC is loosely defined in the RISC-V Privileged Spec allowing for a variety of implementations
  - Can handle as many interrupts as required in a given implementation
  - Scalable to multi-core implementations
- Decouples the complexity of some interrupt types from the hart (such as PCIe MSI)



# RISC-V Interrupt System Architecture (M-mode only example)





# Questions



# 3 Part Webinar Series

- RISC-V 101
  - The Fundamentals of RISC-V architecture
- Introduction to SiFive Coreplex IP
  - October 17<sup>th</sup>, 2017
- Getting Started with SiFive Coreplex IP
  - November 2017



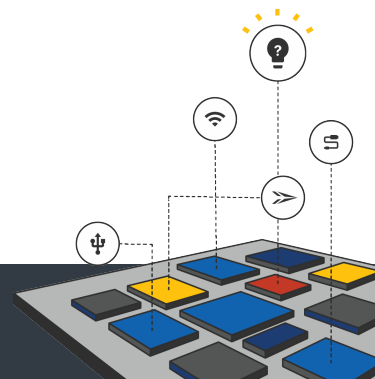
Study



Evaluate

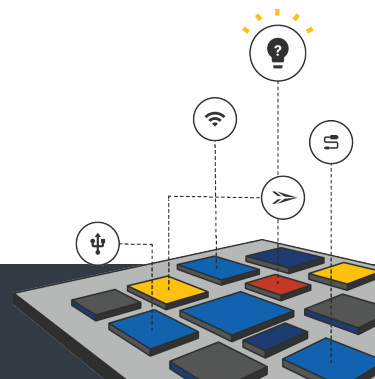


Buy

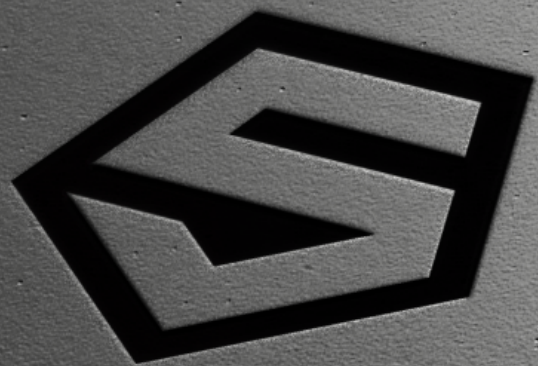


# Resources

- <https://riscv.org/>
  - RISC-V Specifications
  - Links to the RISC-V mailing lists
  - Workshop proceedings
- GitHub
  - <https://github.com/sifive/>
  - <https://github.com/riscv/>
- <https://www.sifive.com/>
  - RISC-V IP and Boards
  - RISC-V Tools
  - Forums







*SiFive*