## RISC-V: Open Hardware for Your Open Source Software

Arun Thomas
@arunthomas
arun.thomas@acm.org
FOSDEM 2017



### Talk Overview

- Goal: Give you a tour of the RISC-V ISA and ecosystem
  - RISC-V 101
  - Hardware Landscape
  - Software Landscape



# RISC-V is an open instruction set specification.

# You can build open source or proprietary implementations. Your **choice**.

## No licensing fees, No contract negotiations, No **lawyers**.







### RISC-V

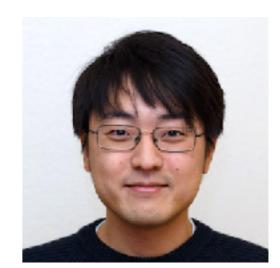
- Modest Goal: "Become the standard ISA for all computing devices"
  - Microcontrollers to supercomputers
- Designed for
  - Research
  - Education
  - Commercial use

## RISC-V: The Responsible Parties









Krste Asanović, David Patterson, Andrew Waterman, and Yunsup Lee





## Origin of RISC-V

- Krste et al. began searching for a common research ISA
  - x86 and ARM: too complex, IP issues
  - Embarked on a "3-month project" to develop their own clean-slate ISA (Summer 2010)
- Released frozen User specification in May 2014
- RISC-V Foundation formed in August 2015

### RISC-V 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.

### RISC-V Foundation

- RISC-V Foundation is a non-profit organization
- 50+ members have joined the Foundation
- Broad commercial and academic interest
  - Sold out the 5th Workshop (350+ attendees representing 107 companies & 29 universities)

## RISC-V Foundation: Platinum Members



### RISC-V Foundation: Gold, Silver, & Auditor Members



### RISC-V ISA

- Fifth RISC ISA from Berkeley, so RISC-V
- Modular ISA: Simple base instruction set plus extensions
  - 32-bit, 64-bit, and 128-bit ISAs
  - <50 hardware instructions in the base ISA</li>
- Designed for extension/customization

### RISC-V ISA Overview

- Base integer ISAs
  - RV32I, RV64I, RV128I
- Standard extensions
  - M: Integer multiply/divide
  - A: Atomic memory operations
  - **F**: Single-precision floating point
  - **D**: Double-precision floating point
  - **G**: IMAFD, "General purpose" ISA

_	
74	ι \
(	1)
<b>\</b> 4	.,

Load Sudd Byte   List   Lis											
Load Byte	Base Integer Instructions: RV32I, RV64I, and RV										
Load Halfword   Load Word   Load Unisigned		_			+RV{6	4,128}					
Load Word   I Law   rd, rs1, j.mm   Load Red Norman   Load Red N	•	1		imm						l	rd,csr,rs1
Load Byte Unsigned   I LBU rd,rsl,imm   Atomic Read & Clear Bit Imm   CRRRI rd,csr,imm   Atomic Read & Clear Bit Imm   CRR   CRL   Imm   CRRRI rd,csr,imm   Atomic Read & Clear Bit Imm   CRRRI rd,csr,imm   Atomic Read & Clea	Load Halfword	1	LH rd,rs1,	imm						1	rd,csr,rs1
Load Helf Unsigned   Store Store Store Store Store Store Word   Store Store Helfword   S	Load Word	1	LW rd,rs1,	imm L{D	Q} r	d,rs1,	imm	Aton	nic Read & Clear Bit	CSRRC	rd,csr,rs1
Store   Store   Store   Store   Store   Hardword   Store   Store   Hardword   Store   Store   Hardword   Store   St	Load Byte Unsigned	I	LBU rd,rs1,							1	
Store Halfword   S   SR	Load Half Unsigned	_	LHU rd, rs1,	imm L{W	D}U r	d,rs1,	i.mm	Atomic	Read & Set Bit Imm	CSRRSI	rd,csr,imm
Shifts   Shift   R   SLL   rd,rs1,rs2,imm   SLL(W D)   rd,rs1,rs2   Environment Breakpoint   EBREAK	<b>Stores</b> Store Byte	S	SB rs1,rs2	,imm				Atomic Re	ead & Clear Bit Imm	CSRRCI	rd,csr,imm
Shift	Store Halfword	S	SH rs1,rs2	,imm				Change I	<b>Level</b> Env. Call	ECALL	
Shift Left Immediate   Shift Right   R   SRL   rd,rsl,rsl,mamt   SRL(W D) rd,rsl,shamt   SRL(W D) rd,rsl,shamt   SRL(W D) rd,rsl,shamt   SRL(W D) rd,rsl,rs2   SRL(W D) rd,rsl	Store Word	S	SW rs1,rs2	,imm S{D	Q} r	s1,rs2	,imm	Envir	onment Breakpoint	EBREAK	
Shift Left Immediate   I   SLLI   rd,rs1,rs2   SRL(W D) rd,rs1,shamt   Shift Right Immediate   I   SRLI   rd,rs1,rs2   SRL(W D) rd	Shifts Shift Left	R	SLL rd,rs1,	rs2 SLL	{W D} r	d,rs1,	rs2		Environment Return	ERET	
Shift Right Immediate   SRL rd,rs1,rs2   SRL(W D) rd,rs1,shamt   SRA rd,rs1,rs2   SRL(W D) rd,rs1,shamt   SRA rd,rs1,rs2   SRA(W D) rd,rs1,shamt   S	Shift Left Immediate	I			!{W D} r	d,rs1,s	shamt				
Shift Right Arithmetic   Shift Right Replay   Shift Replay   Shift Right Replay   Shift Replay   Shift Right Replay   Shift Replay   Shift Right Replay   Shift Right Replay   Shift Replay	Shift Right	R		I					· · · · · · · · · · · · · · · · · · ·	1	
Shift Right Arith Imm	•	1	1 ' '		• ' . '					1	
Shift Right Arith Imm   I   SRAI   rd,rsl,shamt   ADD   rd,rsl,rs2	•	1	1 ' '	<b>I</b>							
ADD rd,rsl,rs2 ADD word,rsl,rs2 ADD word word,rsl,rs2 AND mediate AND word,rsl,rs2 AND mediate AND word,rsl,rs2 AND mediate AND word,rsl,rs2 AND word,rsl,rs2 AND word,rsl,rs2 AND word,rsl,rs2 AND word,rsl,rs2 AND word word,rsl,rs2 AND word word word word,rsl,rsl,rs2 AND word word word word word word word word	-	_									.VM rs1
ADD Immediate SUBtract Load Upper Imm to PC Logical XOR R XOR rd,rs1,rs2 Load Upper Imm to PC Logical XOR R XOR rd,rs1,rs2 OR Immediate AND R AND R AND rd,rs1,imm AND Immediate AND Immediate AND Immediate I AND rd,rs1,rs2 AND Immediate I AND rd,rs1,imm Load Word SP Load Quad SP Load Quad SP CL C.LD rd',rs1',imm LD rd',rs1',imm*8 LD rd,sp,imm*8 Load Quad SP CL C.LD rd',rs1',imm LD rd',sp,imm*8 Load Quad SP CL C.LDSP rd,imm LD rd',rs1',imm*16 LD rd',sp,imm*8 Load Quad SP CL C.LOSP rd,imm LD rd',rs1',imm*16 LD rd',sp,imm*8 LOad Quad SP CL C.LOSP rd,imm LD rd',rs1',imm*16 LD rd',sp,imm*8 LOad Quad SP CC C.SC SS C.SWS rs1',rs2',imm Set < Immediate Set < Immediate Set < Immunisigned I StTTU rd,rs1,imm Set < Immunisigned Set Signe Word SP Set < Immunisigned Set Signe Word SP Set Si									- Cupe. 1.501 1.2.102	DI DI(OD	
SUBtract   Load Upper Imm to PC   U AUID   rd, imm   Add Upper Imm to PC   U AUID   rd, imm   Add Upper Imm to PC   U AUID   rd, imm   Add Upper Imm to PC   U AUID   rd, imm   Add Upper Imm to PC   U AUID   rd, imm   Add Upper Imm to PC   U AUID   rd, imm   Add Upper Imm to PC   U AUID   rd, imm   Add Upper Imm to PC   U AUID   rd, imm   Add Upper Imm to PC   U AUID   rd, imm   AUD   R   R   R   R   R   R   R   R   R		1	1 ' '								
Load Upper Imm				I							
Add Upper Imm to PC		''		I DOD				cod (16	hit) Instructio	n Evto	ncioni DVC
Logical   XOR   R   XOR   rd,rs1,rs2   XOR   Immediate   I XORI   rd,rs1,imm   CI   CI   C.LWS   rd,imm   LW rd',rs1',imm*4   CI   C.LWS   rd,imm   LW rd',rs1',imm*4   LW rd',sp,imm*4   CI   C.LWS   rd,imm   LW rd',sp,imm*4   CI   C.LWS   rd,imm   LW rd',sp,imm*8   LOad Double   C.LD   rd',rs1',imm   LD rd',rs1',imm*8   LD rd',rs1',imm*10   LD rd',rs1',imm*10   LD rd',rs1',imm*10   LD rd',rs1',imm*10		1	· '	Cat				Seu (10-			
XOR Immediate			·					0.711			•
OR R OR rd,rs1,rs2 OR Immediate I ORI rd,rs1,imm AND R AND Immediate I ANDI rd,rs1,imm AND Immediate I ANDI rd,rs1,imm Compare Set R SLT rd,rs1,rs2 Set < Immediate I SLTI rd,rs1,imm Set < Immediate I SLTI rd,rs2,imm Set < Immediate I SLTI rd,rs1,imm Set < Immediate I SLTI rd,rs2,imm Set < Immediate I SLTI rd,rim,rs2,imm Set < Immediate I Rd	_	1						l		1	
OR Immediate   AND R   AND R   AND R   AND I    AND I				ımm					ra,ımm		= :
AND Immediate I AND rd,rs1,rs2 AND Immediate I ANDI rd,rs1,imm  Compare Set < R SLT rd,rs1,rs2 Set < Immediate I SLTII rd,rs1,imm Set < Unsigned Set < R SLT rd,rs1,imm Set < Immediate I SLTIU rd,rs2,imm Set < Immediate I Sltiu rd,rs1,imm Set < Immediate I ADD rd,rs1,rs2,imm Set < Immediate I ADD rd,rs1,rs2,imm ADD Word Imme I I RDD Word Imme I I R		1	OR rd,rs1,	rs2	Load	Double		l	rd',rs1',imm	LD rd'	rs1′,imm*8,
AND Immediate   I   AND   rd,rs1,imm   Load Quad SP   CI   C.LQSP   rd,imm   LQ rd,sp,imm*16	OR Immediate	1	ORI rd,rs1,	imm	Load Do	uble SP		l	•		
Set < Immediate Set < R SLT rd,rs1,rs2 Set < Immediate Set <	AND	1	AND rd,rs1,	rs2		٠ ١		C.LQ	rd',rs1',imm	LQ rd'	rs1′,imm*16,
Set < Immediate Set < Unsigned R SLTU rd,rs1,imm Store Word SP Set < Imm Unsigned I SLTU rd,rs1,rs2 Store Double SP Store Double SP CSS C.SDSP rs2,imm SD rs2,sp,imm*4  Branches Branch = SB BEQ rs1,rs2,imm Branch < SB BLT rs1,rs2,imm Branch < SB BLT rs1,rs2,imm Branch < SB BLT rs1,rs2,imm Branch > SB BGE rs1,rs2,imm Branch > Unsigned SB BGE rs1,rs2,imm Branch ≥ Unsigned SB BGE rs1,rs2,imm ADD Word Imm CI C.ADDI rd,rs1 ADD rd,rd,imm ADD rd,rd,imm ADD rd,rd,imm ADD rd,rd,imm ADD RDP Imm*4 CI C.ADDI rd,imm ADD rd,rd,imm ADD RDP Imm*4 CI C.ADDI RDP	AND Immediate		ANDI rd,rs1,	imm	Load C	uad SP	CI	C.LQSP	rd,imm	LQ rd,	sp,imm*16
Set < Unsigned R SLTU rd,rs1,rs2 Store Double CS C.SD rs1',rs2',imm SD rs1',rs2',imm*8  Branches Branch = SB BEQ rs1,rs2,imm Branch ≥ SB BET rs1,rs2,imm Branch ≥ SB BLT rs1,rs2,imm Branch ≥ Unsigned SB BLTU rs1,rs2,imm Branch ≥ Unsigned Branch ≥ Unsigned SB BLTU rs1,rs2,imm Branch ≥ Unsigned SB BLTU rs1,rs2,imm Branch ≥ Unsigned Branch ≥ Unsigned Branch ≥ Unsigned SB BCEU rs1,rs2,imm ADD Word Imm CI C.ADDI rd,imm ADDI rd,rd,imm ADDI rd,rd,imm ADD SP Imm*16  Jump & Link Register UJ JALL rd,imm ADD SP Imm*4 CI C.ADDI16SP x0,imm ADDI rd,rd,imm ADDI sp,sp,imm*16  Synch Synch thread I FENCE LOad Unper Imm CI C.LII rd,imm ADDI rd,sp,imm*16  System System CALL I SCALL System BREAK I SBREAK SUB CR C.SUB rd,rs1 SUB rd,rd,rs1  ReaD CYCLE upper Half I RDTIME rd ReaD TIME upper Half I RDTIME rd ReaD INSTR upper Half I RDTIME rd ReaD INSTR upper Half I RDINSTRETT rd Load Imme & Link Register CR C.JALL imm JALL ra,imm JALR ra,rs1,00	Compare Set <	R	SLT rd,rs1,	rs2 <b>Sto</b> l	res Stor	e Word		C.SW	rs1',rs2',imm	SW rs1	',rs2',imm*4
Set < Imm Unsigned I SLTIU rd,rs1,imm Store Double SP CSS C.SDSP rs2,imm SD rs2,sp,imm*8  Branches Branch = SB BeQ rs1,rs2,imm Branch ≠ SB Bex rs1,rs2,imm Branch ≤ SB Bex rs1,rs2,imm Branch ≥ SB Bex rs1,rs2,imm Branch ≥ SB Bex rs1,rs2,imm Branch ≥ Unsigned SB Bex rs1,rs2,imm Branch ≥ Unsigned Branch ≥ Unsigned SB Bex rs1,rs2,imm Branch ≥ Unsigned SB Bex rs1,rs2,imm Branch ≥ Unsigned Branch ≥ Unsigned SB Bex rs1,rs2,imm ADD Word CR C.ADDW rd,rs1 ADDW rd,rd,imm ADDI rd,rd,imm ADDI rd,rd,imm ADDI rd,rd,imm ADDI rd,rd,imm ADDI rd,rd,imm ADDI sp,sp,imm*16  Jump & Link Register UJ JALR rd,rs1,imm ADD SP Imm*4 CI C.ADDI rd,imm ADDI rd,rd,imm ADDI rd,rd,imm ADDI sp,sp,imm*16  Synch Synch Instr & Data I FENCE LOad Upper Imm CI C.LUI rd,imm ADDI rd,x0,imm LUI rd,imm ADDI rd,x0,imm CI C.LUI rd,imm LUI rd,imm LUI rd,imm LUI rd,imm CI C.LUI rd,imm LUI rd,imm LUI rd,imm Box rd,rs1,x0  System BrEAK I SBREAK SUB CR C.SUB rd,rs1 SUB rd,rd,rs1  ReaD CYCLE upper Half I RDTIME rd ReaD INSTR RETired ReaD INSTR RETired ReaD INSTR RETired ReaD INSTR RETIRED READ INSTRETH rd  ReaD INSTR upper Half I RDINSTRETH rd  Store Quad SP CSS C.SQSP rs2,imm SQ rs1',rs2',imm*16  Store Quad SP CSC C.SQSP rs2,imm SQ rs1',rs2',imm*16  Sq. rs1,rs2',imm SQ rs1',rs2',imm SQ rs1',rs2',imm*16  Sq. rs1,rs2',imm SQ rs1',rs2',imm SQ rs1',rs2',imm*16  Sq. rs1,rs2',imm SQ rs1',rs2',imm*16  Sq. rs1,rs2',imm SQ rs1',rs2',imm*16  Sq. rs1,rs2',imm SQ rs1',rs2',imm*16  Sq. rs1',rs2',imm*16  Sq. rs1',rs2',imm SQ rs1',rs2',imm*16  ADD Word Imm CI C.ADDI rd,rim ADDI rd,rd,imm ADDI rd,rd,imm ADDI rd,rd,imm ADDI rd,rd,imm CI C.LUI rd,imm SULI rd,imm SULI rd,rimm SULI rd,rimm SULI rd,rimm SULI rd,rimm SULI rd,rimm SULI rd,rd,imm Branch 2 CB C.BEQZ rs1',imm BNE rs1',x0,imm Branch 2 CB C.BEQZ rs1',imm BNE rs1',x0,imm BNE rs1',x0,imm Branch 2 CB C.BEQZ rs1',imm BNE rs1',x0,imm BNE rs1',x0,imm Branch 2 CB C.BEQZ rs1',imm BNE rs1',x0,imm Branch	Set < Immediate	1	SLTI rd,rs1,	imm	Store V	Vord SP		C.SWSP	rs2,imm	SW rs2	sp,imm*4
Branches Branch = SB Branch = SL Ir rd, rmm Sr rd, rsl yrsl rd, rsl yrs	Set < Unsigned	R		ll ll	Store	Double		C.SD	rs1',rs2',imm	SD rs1	',rs2',imm*8
Branch ≠ SB Branch = SB Branch = SB Branch < Unsigned SB Branch < SB Branch < Unsigned SB Segrours < Unsigned SP Segrours < Unsigned Segrours < Unsigned SP Segrours < Unsigned SP Segrours < Unsign	Set < Imm Unsigned	I	SLTIU rd, rs1,	imm	Store Do	uble SP	CSS	C.SDSP	rs2,imm	SD rs2	sp,imm*8
Branch < SB BLT rs1,rs2,imm Branch ≥ SB BGE rs1,rs2,imm Branch ≥ SB BGE rs1,rs2,imm Branch > SB BGE rs1,rs2,imm Branch > Unsigned SB BLTU rs1,rs2,imm Branch ≥ Unsigned SB BGEU rs1,rs2,imm ADD Immediate CI C.ADDI rd,imm ADDI rd,rd,imm ADDI wrd,rd,imm ADDI sp,sp,imm*16 CI C.ADDI4SPN rd',imm ADDI rd',sp,imm*4 CI C.ADDI4SPN rd',imm ADDI rd',sp,imm*4 Load Immediate CI C.LI rd,imm ADDI rd,x0,imm ADDI rd,x0,imm CI C.LUI rd,imm ADDI rd,x0,imm C	Branches Branch =	SB	BEQ rs1,rs2	,imm	Stor	e Quad	CS	c.so	rs1',rs2',imm	SQ rs1	',rs2',imm*16
Branch ≥ SB BGE rs1,rs2,imm Branch < Unsigned SB BLTU rs1,rs2,imm Branch > Unsigned SB BLTU rs1,rs2,imm Branch ≥ Unsigned SB BGEU rs1,rs2,imm  Jump & Link J&L UJ JAL rd,imm Jump & Link Register UJ JALR rd,rs1,imm  Synch Synch thread I FENCE Synch Instr & Data I FENCE.I  System System CALL System BREAK I SBREAK  Counters ReaD CYCLE I ROCYCLE rd ReaD TIME upper Half I RDTIME rd ReaD TIME upper Half I Reptimen rd ReaD INSTR upper Half I RDINSTRETH rd  Branch ≥ Unsigned SB BLTU rs1,rs2,imm  ADD Word Imm ADD I rd,rd,imm ADDI rd,rd,imm ADDI sp,sp,imm*16 CI C.ADDI16SP x0,imm ADDI sp,sp,imm*16 CI C.ADDI4SPN rd',imm ADDI rd,rd,imm ADDI rd,imm ADDI rd,rd,imm ADDI rd,	Branch ≠	SB	BNE rs1,rs2	,imm	Store C	uad SP	CSS	C.SQSP	rs2,imm	SQ rs2	,sp,imm*16
Branch < Unsigned SB BLTU rs1,rs2,imm Branch ≥ Unsigned SB BGEU rs1,rs2,imm SB BGEU rs1,rs2,imm SB BGEU rs1,rs2,imm SB BGEU rs1,rs2,imm ADD Immediate CI C.ADDI rd,imm ADDI rd,rd,imm ADDI rd,rd,imm ADDI sp,sp,imm*16 CI C.ADDI rd,imm ADDI sp,sp,imm*16 ADD SP Imm * 16 CI C.ADDI rd,imm ADDI sp,sp,imm*16 ADD SP Imm * 4 CIW C.ADDI4SPN rd',imm ADDI rd',sp,imm*4 CIW C.AUDI4SPN rd',imm ADDI rd',sp,imm*4 CIW C.A	Branch <	SB	BLT rs1,rs2	,imm Arit	hmetic	ADD	CR	C.ADD	rd,rs1	ADD	rd,rd,rs1
Branch ≥ Unsigned SB BGEU rs1,rs2,imm  Jump & Link J&L UJ JAL rd,imm    Jump & Link Register UJ JALR rd,rs1,imm  Synch Synch thread I FENCE    Synch Instr & Data I FENCE.I  System System CALL J System BREAK I SBREAK  Counters ReaD CYCLE I Red CYCLE upper Half ReaD TIME upper Half ReaD INSTR RETired ReaD INSTR RETired ReaD INSTR upper Half I RDINSTRETH rd  Branch ≥ Unsigned SB BGEU rs1,rs2,imm  ADD Word Imm CI C.ADDIW rd,imm ADDI sp,sp,imm*16  ADD SP Imm * 16  CI C.ADDIW rd,imm ADDI sp,sp,imm*16  ADD SP Imm * 16  CI C.ADDIW rd,imm ADDI sp,sp,imm*16  ADD SP Imm * 16  CI C.ADDIW rd,imm ADDI sp,sp,imm*16  ADD Word Imm ADDI sp,sp,imm*16  ADD Word Imm ADD I sp,sp,imm*16  ADD Word Imm ADD I sp,sp,imm*16  ADD Word Imm ADDI sp,sp,imm*16  ADD Word Imm ADD I sp,sp,imm*16  ADD I sp,sp,imm*16  ADD Word Imm ADD I sp,sp,imm*16  ADD I sp,sp,imm*16  ADD Word Imm ADD I sp,sp,imm*16  ADD Word Imm ADD I sp,sp,imm*16  ADD I	Branch ≥	SB	BGE rs1,rs2	,imm	AD	D Word	CR	C.ADDW	rd,rs1	ADDW	rd,rd,imm
Jump & LinkJ&LUJJALrd,immADD SP Imm * 16CIC.ADDI16SP x0,immADDI sp,sp,imm*16Jump & Link RegisterUJJALR rd,rs1,immADD SP Imm * 16CIWC.ADDI14SPN rd',immADDI rd',sp,imm*4Synch Synch thread Synch Instr & Data Synch Instr & Data System System CALL System BREAKIFENCE ILoad Immediate Load Upper ImmCIC.LII C.LUIrd,immADDI rd',sp,imm*4System System CALL System BREAKISCALL SEREAKMove CR C.SUBCR C.SUBC.MV RC.SUBrd,rs1ADD rd,rs1,x0Counters ReaD CYCLE ReaD CYCLE upper Half ReaD TIME upper Half ReaD INSTR RETired ReaD INSTR upper HalfIRDTIME RDINSTRETH RDINSTRETHADD SP Imm * 16 Load Immediate ADD SP Imm * 4 Load Immediate Move SUB SUB CR C.SUB CR C.SUB CR C.SUBADD Imm RDI Immediate CIC.LII C.LUIrd,imm RDI Immediate ADD rd,rs1 SUB SUB SUB RDI rd,rs1SUB Shifts Shift Left Imm Branches Branch=0 Branches Branch=0 CB<	Branch < Unsigned	SB	BLTU rs1,rs2	,imm	ADD Imr	nediate	CI	C.ADDI	rd,imm	ADDI	rd,rd,imm
Jump & Link Register UJ Jalr rd,rs1,imm  Synch Synch thread I FENCE Synch Instr & Data I FENCE.I  System System CALL I SCALL System BREAK I SBREAK  Counters ReaD CYCLE I RDCYCLE rd ReaD TIME ReaD TIME I RDTIME rd ReaD INSTR RETired ReaD INSTR upper Half I RDINSTRETH rd  ReaD INSTR upper Half I RDINSTRETH rd  ADDI rd',sp,imm*4  CI C.LI rd,imm ADDI rd',sp,imm*4  CI C.LII rd,imm ADDI rd',sp,imm*4  CI C.LII rd,imm ADDI rd',sp,imm*4  CI C.LII rd,imm LUI rd,imm SUB rd,rs1,x0  SUB rd,rs1  SUB rd,rd,rs1  Branches Branch=0 CB C.BEQZ rs1',imm BEQ rs1',x0,imm Branch#0 CB C.BNEZ rs1',imm BNE rs1',x0,imm JAL x0,imm JAL x0,imm JAL x0,imm JAL x0,imm JAL ra,imm	Branch ≥ Unsigned	SB	BGEU rs1,rs2	,imm	ADD Wo	rd Imm	CI	C.ADDIW	rd,imm	ADDIW	rd,rd,imm
Synch Synch thread I FENCE       Load Immediate CI C.LI rd,imm       ADDI rd,x0,imm         System System CALL I System BREAK I System BREAK I System BREAK I System BREAK I System CYCLE I RECYCLE READ CYCLE I RECYCLE READ CYCLE Upper Half I READ TIME READ TIME Upper Half I READ TIME READ TIME Upper Half I READ TIME READ INSTRETIRED I READ INSTRUMENTANTE I READ INSTRUMENT	Jump & Link J&L	UJ	JAL rd,imm	,	ADD SP Im	m * 16	CI	C.ADDI16	SP x0,imm	ADDI	sp,sp,imm*16
Synch Synch thread I FENCE Synch Instr & Data I FENCE.I  System System CALL I SCALL System BREAK I SBREAK  Counters ReaD CYCLE I RDCYCLE rd ReaD TIME ReaD TIME I ReaD TIME ReaD INSTR RETired ReaD INSTR upper Half I RDINSTRETH rd  ReaD INSTR upper Half I RDINSTRETH rd  Synch Instr & Data I FENCE Load Immediate CI C.LII rd, imm ADDI rd,x0,imm LUI rd,imm SUB rd,rs1,x0 SUB rd,rs1 SUB rd	Jump & Link Register	UJ	JALR rd,rs1,	imm	ADD SP I	mm * 4	CIW	C.ADDI4S	PN rd',imm	1	
System System CALL I SCALL System BREAK I SBREAK System CYCLE I RDCYCLE rd ReaD CYCLE upper Half ReaD INSTR RETired ReaD INSTR upper Half I RDINSTRETH rd I	Synch Synch thread	I	FENCE		Load Imr	nediate	CI	C.LI	rd,imm	1	·
System System CALL I SCALL SUB FEAK I SBREAK I SBREAK SUB CR C.SUB rd,rs1 SUB rd,rd,rs1  Counters ReaD CYCLE I RDCYCLE rd ReaD TIME I RDTIME rd ReaD TIME Upper Half I RDTIME rd ReaD INSTR RETired ReaD INSTR upper Half I RDINSTRETH rd System CALL RD C.S.MV rd,rs1 ADD rd,rs1,x0 SUB rd,rd,rs1  SUB CR C.SUB rd,rs1 SUB rd,rd,imm SLLI rd,imm SLLI rd,imm BEQ rs1',x0,imm BEQ rs1',x0,imm BNE rs1',x0,imm BNE rs1',x0,imm BNE rs1',x0,imm SLDINGREGISTER CR C.JR rd,rs1 JALR x0,rs1,0  Jump Register CR C.JR rd,rs1 JALR x0,rs1,0  Jump & Link Register CR C.JALR rs1 JALR ra,rs1,0	Synch Instr & Data	I	FENCE.I		Load Upp	er Imm	CI	C.LUI		1	
System BREAK I SBREAK SUB CR C.SUB rd,rs1 SUB rd,rd,rs1  Counters 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 ReaD INSTR upper Half I RDINSTRETH rd Jump & Link Register CR C.JALR rs1 JALR ra,rs1,0	System System CALL					MoVe	CR	C.MV	•	l	•
Counters ReaD CYCLE   I RDCYCLE   I RDCYCLE   I RDCYCLE   I RDCYCLE   I RDCYCLE   rd ReaD CYCLE   upper Half   I RDCYCLE   rd ReaD TIME   I RDTIME   rd ReaD TIME   I RDTIME   rd ReaD INSTR RETired ReaD INSTR upper Half   I RDINSTRET   rd ReaD INSTR upper Half   I RDINSTRETH   rd ReaD INSTR upper Half   I RDINSTRETH   rd RDINSTRETH   rd ReaD INSTR upper Half   I RDINSTRETH   rd RD	System BREAK	I	1					l	•	l	
ReaD CYCLE upper Half ReaD TIME ReaD TIME ReaD TIME upper Half ReaD INSTR RETired ReaD INSTR upper Half ReaD I	Counters ReaD CYCLE	I	RDCYCLE rd	Shif	fts Shift L	eft Imm	CI	C.SLLI			
ReaD TIME I RDTIME rd ReaD TIME upper Half I RDTIME rd ReaD INSTR RETired ReaD INSTR upper Half I RDINSTRETH rd Jump Register CR C.JR rd,rs1 JALR x0,rs1,0  Jump & Link J&L CJ C.JAL imm JAL ra,imm	ReaD CYCLE upper Half	1	1	Bra	nches Br	anch=0					
ReaD TIME upper Half I RDTIMEH rd ReaD INSTR RETired I RDINSTRET rd ReaD INSTR upper Half I RDINSTRETH rd ReaD INSTR upper Half I RDINSTRETH rd Jump Register CR C.JR rd,rs1 JALR x0,rs1,0  Jump & Link J&L CJ C.JAL imm JAL ra,imm JAL ra,imm JAL ra,imm JAL ra,imm JAL ra,rs1,0	• • •	_	1			I				1	
ReaD INSTR RETired I RDINSTRET rd ReaD INSTR upper Half I RDINSTRETH rd Jump Register CR C.JR rd,rs1 JALR x0,rs1,0  Jump & Link J&L CJ C.JAL imm JAL ra,imm JAL ra,imm JAL ra,rs1,0		_	1	Jun							
ReaD INSTR upper Half I RDINSTRETH rd Jump & Link J&L CJ C.JAL imm JAL ra,imm  Jump & Link Register CR C.JALR rs1 JALR ra,rs1,0		1	1		•			l		1	•
Jump & Link Register CR C.JALR rs1 JALR ra,rs1,0			1	Jum						1	
			1		•			l		l	•
System Env. BREAK CI C.EBREAK EBREAK				ll ll	-	- 1					
System Env. BREAK CI   C. EBREAK   EBREAK   EBREAK   BEREAK   32-bit Instruction Formats   16-bit (RVC) Instruction Formats	_		CIII LIIV.	DIVLAR	CI						

32	-hit	Inct	ructio	n Forr	nate

31	30	25 24	21	20	19	15 14	12	2 11 8	7	6	0	
fı	ınct7		rs2		rs1	rs1 funct3 rd				opcode		
imm[11:0]					rs1	fu	nct3	1	ď	d opco		
imr	n[11:5]		rs2		rs1	fu	nct3	imn	n[4:0]	opco	de	
imm[12]	imm[10:	5]	rs2		rs1	fu	nct3	imm[4:1]	imm[11]	opco	de	
	•			1	d	opco	de					
imm[20] imm[10:1] imm[11]		in	imm[19:12]		1	opco	de					
	imr imm[12]	funct7 imm[11:5] imm[12]   imm[10:	$\begin{array}{c c} \text{funct7} & & \\ & \text{imm}[11:0] \\ \hline & \text{imm}[11:5] & \\ \hline & \text{imm}[12] \mid & \text{imm}[10:5] \\ \end{array}$	$\begin{array}{c c} \text{funct7} & \text{rs2} \\ \hline \text{imm}[11:0] \\ \hline \text{imm}[11:5] & \text{rs2} \\ \hline \text{imm}[12] & \text{imm}[10:5] & \text{rs2} \\ \hline \text{imm}[31:1] \\ \hline \end{array}$	$\begin{array}{c c} \text{funct7} & \text{rs2} \\ \hline \text{imm}[11:0] \\ \hline \text{imm}[11:5] & \text{rs2} \\ \hline \text{imm}[12] \mid \text{imm}[10:5] & \text{rs2} \\ \hline \text{imm}[31:12] \\ \end{array}$	$\begin{array}{c cccc} funct7 & rs2 & rs1 \\ \hline imm[11:0] & rs1 \\ \hline imm[11:5] & rs2 & rs1 \\ \hline imm[12] & imm[10:5] & rs2 & rs1 \\ \hline imm[31:12] & \\ \hline \end{array}$	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	$\begin{array}{c ccccc} funct7 & rs2 & rs1 & funct3 \\ \hline imm[11:0] & rs1 & funct3 \\ \hline imm[11:5] & rs2 & rs1 & funct3 \\ \hline imm[12] & imm[10:5] & rs2 & rs1 & funct3 \\ \hline imm[31:12] & \hline \end{array}$	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	

	10 14 10	12	11	10	9	0	4	О	ð	4	3	2	1	U	
	func	t4	rd/rs1					rs2					op		
_	funct3	imm		rd	l/rs	1			j	imm	1		0	p	
5	funct3		ir	imm						rs2			op		
V	funct3		imm						rd'					p	
Ī	funct3	im	ım		1	rs1'		im	m		rd'		0	p	
	funct3	im		rs1'			im	ım rs2′				op			
	funct3	off		rs1'			offset				0	p			
	funct3	jump target									0	p			

RISC-V Integer Base (RV32I/64I/128I), privileged, and optional compressed extension (RVC). Registers x1-x31 and the pc are 32 bits wide in RV32I, 64 in RV64I, and 128 in RV128I (x0=0). RV64I/128I add 10 instructions for the wider formats. The RVI base of <50classic integer RISC instructions is required. Every 16-bit RVC instruction matches an existing 32-bit RVI instruction. See risc.org.

### Free & Open RISC-V Reference Card (riscv.org) 2

Cotto mana	l =		Instruction Extension: RVM	
Category Name Multiply MULtiply	Fmt R	MUL rd,rs1,rs2	+RV{64,128}	
Multiply MULtiply  MULtiply upper Half		MUL rd,rs1,rs2 MULH rd,rs1,rs2	MUL{W D} rd,rs1,rs2	
MULtiply Half Sign/Uns	1	MULHSU rd,rs1,rs2		
MULtiply upper Half Uns	1	MULHU rd,rs1,rs2		
<b>Divide</b> DIVide	1	DIV rd,rs1,rs2	DIV{W D} rd,rs1,rs2	
DIVide Unsigned	R	DIVU rd,rs1,rs2		
Remainder REMainder	R	REM rd,rs1,rs2	REM{W D} rd,rs1,rs2	
REMainder Unsigned	R	REMU rd,rs1,rs2	REMU{W D} rd,rs1,rs2	
Ор	tiona	al Atomic Instruction Extension	on: RVA	
<b>Category</b> Name	Fmt	RV32A (Atomic)	+RV{64,128}	
<b>Load</b> Load Reserved		LR.W rd,rs1	LR.{D Q} rd,rs1	
Store Store Conditiona		SC.W rd,rs1,rs2	SC.{D Q} rd,rs1,rs2	
Swap SWAF	_	AMOSWAP.W rd,rs1,rs2	AMOSWAP.{D Q} rd,rs1,rs2	
Add ADD		AMOADD.W rd,rs1,rs2	AMOADD. {D Q} rd,rs1,rs2	
Logical XOF	1	AMOXOR.W rd,rs1,rs2	AMOXOR. {D Q} rd,rs1,rs2	
AND OR	1	AMOAND.W rd,rs1,rs2 AMOOR.W rd,rs1,rs2	AMOAND.{D Q} rd,rs1,rs2 AMOOR.{D Q} rd,rs1,rs2	
	<del>                                     </del>			
Min/Max MINimun MAXimum	1	AMOMIN.W rd,rs1,rs2	AMOMIN.{D Q} rd,rs1,rs2 AMOMAX.{D Q} rd,rs1,rs2	
MINimum Unsigned	1	AMOMAX.W rd,rs1,rs2 AMOMINU.W rd,rs1,rs2	AMOMAX.{D Q} rd,rs1,rs2 AMOMINU.{D Q} rd,rs1,rs2	
MAXimum Unsigned	1	AMOMAXU.W rd,rs1,rs2	AMOMAXU.{D Q} rd,rs1,rs2	
		ng-Point Instruction Extensio		
Category Name	Fmt			
Move Move from Integer		FMV.{H S}.X rd,rs1	FMV.{D Q}.X rd,rs1	
Move to Integer		FMV.X.{H S} rd,rs1	FMV.X.{D Q} rd,rs1	
Convert Convert from Int		FCVT.{H S D Q}.W rd,rs1	$FCVT.\{H S D Q\}.\{L T\}$ rd,rs1	
Convert from Int Unsigned	R	FCVT.{H S D Q}.WU rd,rs1	FCVT.{H S D Q}.{L T}U rd,rs1	
Convert to Int	R	FCVT.W.{H S D Q} rd,rs1	FCVT. $\{L T\}$ . $\{H S D Q\}$ rd,rs1	
Convert to Int Unsigned	R	FCVT.WU.{H S D Q} rd,rs1	FCVT.{L T}U.{H S D Q} rd,rs1	
<b>Load</b> Load	I	FL{W,D,Q} rd,rs1,imm		g Convention
Store Store		FS{W,D,Q} rs1,rs2,imm	Register ABI Name Saver	Description
Arithmetic ADD	1	FADD.{S D Q} rd,rs1,rs2	x0 zero	Hard-wired zero
SUBtract	1	FSUB.{S D Q} rd,rs1,rs2	x1 ra Caller	Return address
MULtiply DIVide	1	FMUL.{S D Q} rd,rs1,rs2	x2 sp Callee	Stack pointer Global pointer
SQuare RooT	1	FDIV.{S D Q} rd,rs1,rs2  FSQRT.{S D Q} rd,rs1	""   gp	Thread pointer
Mul-Add Multiply-ADD		FMADD.{S D Q} rd,rs1,rs2,rs3	x4 tp x5-7 t0-2 <b>Caller</b>	Temporaries
Multiply-SUBtract		FMSUB. {S D Q} rd,rs1,rs2,rs3	x8 s0/fp Callee	Saved register/frame pointer
Negative Multiply-SUBtract	1	FNMSUB. {S D Q} rd,rs1,rs2,rs3	x9 s1 Callee	Saved register
Negative Multiply-ADD	1	FNMADD. {S D Q} rd,rs1,rs2,rs3	x10-11 a0-1 Caller	Function arguments/return values
Sign Inject SiGN source	R	FSGNJ.{S D Q} rd,rs1,rs2	x12-17 a2-7 <b>Caller</b>	Function arguments
Negative SiGN source	R	FSGNJN.{S D Q} rd,rs1,rs2	x18-27 s2-11 Callee	Saved registers
Xor SiGN source	1	FSGNJX. $\{S D Q\}$ rd,rs1,rs2	x28-31 t3-t6 <b>Caller</b>	Temporaries
Min/Max MINimum		FMIN.{S D Q} rd,rs1,rs2	f0-7 ft0-7 Caller	FP temporaries
MAXimum	1	FMAX.{S D Q} rd,rs1,rs2	f8-9 fs0-1 Callee	FP saved registers
Compare Compare Float =	1	FEQ. {S D Q} rd,rs1,rs2	f10-11 fa0-1 Caller	FP arguments/return values
Compare Float <		FLT.{S D Q} rd,rs1,rs2	f12-17 fa2-7 Caller	FP arguments
Compare Float ≤	1	FLE. {S D Q} rd,rs1,rs2	f18-27 fs2-11 Callee	FP saved registers
Categorization Classify Type		FCLASS.{S D Q} rd,rs1	f28-31 ft8-11 <b>Caller</b>	FP temporaries
Configuration Read Status		FRCSR rd		
Read Rounding Mode	1	FRRM rd		
Read Flags	1	FRFLAGS rd		
Swap Status Reg Swap Rounding Mode	1	FSCSR rd,rs1		
•		FSRM rd,rs1		
Swap Flags Swap Rounding Mode Imm	1	FSFLAGS rd,rs1 FSRMI rd,imm		
Swap Rounding Mode Imm Swap Flags Imm	1	FSFLAGSI rd,imm		
Swap riays IIIIII	1 1	LOLUMOT TOOUTE	Ш	

RISC-V calling convention and five optional extensions: 10 multiply-divide instructions (RV32M); 11 optional atomic instructions (RV32A); and 25 floating-point instructions each for single-, and quadruple-precision (RV32F, RV32D, RV32Q). The latter add registers f0-f31, whose width matches the widest precision, and a floating-point control and status register fcsr. Each larger address adds some instructions: 4 for RVM, 11 for RVA, and 6 each for RVF/D/Q. Using regex notation, {} means set, so L{D|Q} is both LD and LQ. See risc.org. (8/21/15 revision)

## See RISC-V specs for more details

### RISC-V Specs

- User-Level ISA Specification v2.1 (May 2016)
- Privileged ISA Specification v1.9.1 (Nov 2016)
  - v1.10 will be released soon
- Spec sources: <a href="https://github.com/riscv/riscv-isa-manual">https://github.com/riscv/riscv-isa-manual</a>

# RISC-V Hardware Landscape

### RISC-V Hardware

- RISC-V cores/SoCs for many different use cases
  - Education, research, commercial products
  - Small, low-cost microcontrollers to highperformance multicore chips
- Written in a variety of HDLs
  - VHDL, Verilog, Chisel, Bluespec SystemVerilog

## UC Berkeley and SiFive

- Berkeley Architecture Research (UCB BAR) group created RISC-V
- UCB BAR designed several open source cores/ SoCs for research and teaching
- RISC-V creators formed a startup (SiFive) to design custom RISC-V chips for customers
  - Building on their open source cores

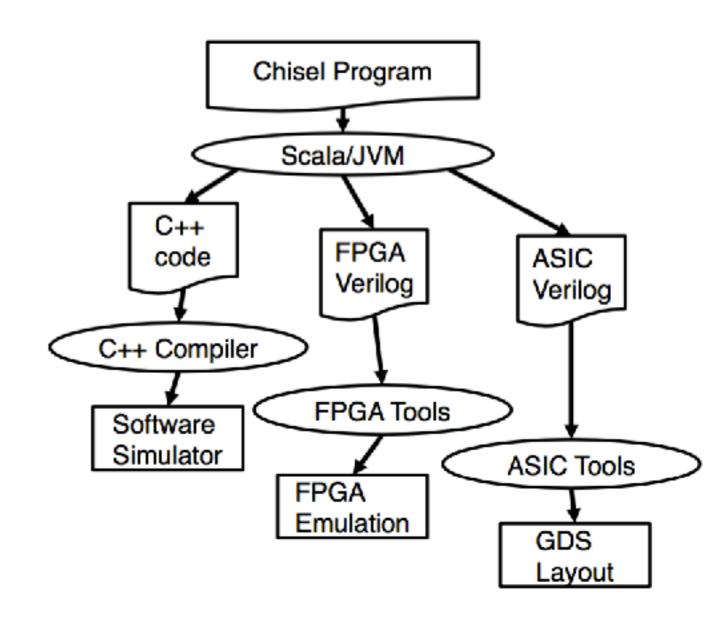
### Berkeley Hardware

- System on Chip (SoC)
  - Rocket Chip Parameterized RISC-V SoC generator
- Cores
  - Rocket Core 5 stage pipeline, single-issue
  - BOOM Out-of-order core
  - Sodor Educational cores (1-5 stage)
- https://github.com/ucb-bar



### Rocket Chip SoC Generator

- Parameterized RISC-V SoC Generator written in Chisel HDL
- Can target C++
   software simulator,
   FPGA emulation, or
   ASIC tools
- Can use this as the basis for your own SoC

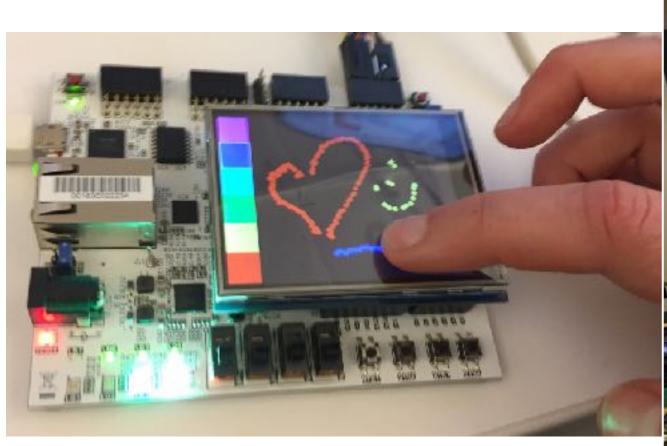


### SiFive Hardware

- Building on Rocket Core and Rocket Chip SoC Gen
- Freedom Everywhere: Low cost, 32-bit microcontrollers
- Freedom Unleashed: High performance, 64-bit multi-core SoCs
- https://github.com/sifive



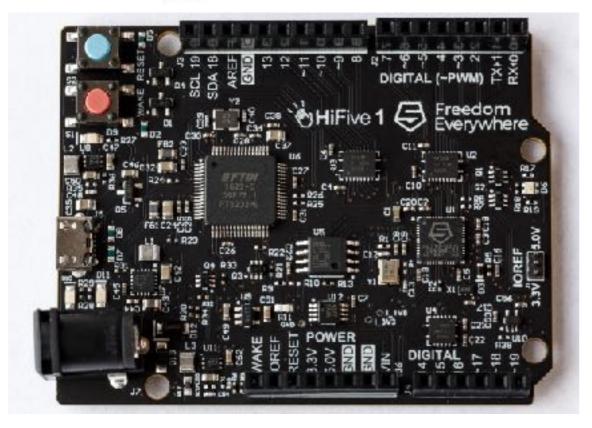
### FPGA Dev Kits





### HiFive1

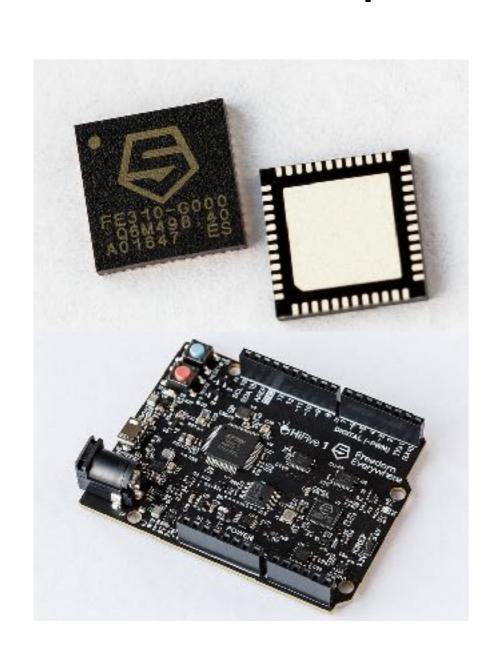
## `WHiFive 1



- First generally available opensource RISC-V silicon
- Freedom Everywhere 310
- Arduino Compatible
- Supports Arduino IDE
- \$59 US from Crowd Supply

https://www.crowdsupply.com/sifive/hifive1

## Freedom Everywhere 310 is Open Source



#### RTL & FPGA scripts

- https://github.com/ucb-bar/rocket-chip
- https://github.com/sifive/sifive-blocks
- https://github.com/sifive/freedom

#### Board Support Packages (BSP)

- https://github.com/sifive/freedom-e-sdk
- <a href="https://github.com/sifive/freedom-u-sdk">https://github.com/sifive/freedom-u-sdk</a>

#### Documentation & Board Schematic

https://dev.sifive.com

### LowRISC

- Based at the University of Cambridge
- Aiming to build a low cost development board.
   "Raspberry Pi for Grownups"
- Builds on Rocket Chip SoC Generator
- Tagged architecture and minion cores
- See Alex Bradbury's FOSDEM 2015 talk
- https://github.com/lowrisc



### Shakti (IIT Madras)

- RISC-V is the "standard ISA" for India
- IIT Madras building 6 open-source cores, from microcontrollers to supercomputers
- Using Bluespec SystemVerilog as HDL
- See <a href="https://bitbucket.org/casl/shakti\_public">https://bitbucket.org/casl/shakti\_public</a>



### Other RISC-V SoCs/Cores

- PULPino (ETH Zurich) <a href="https://github.com/pulp-platform/pulpino">https://github.com/pulp-platform/pulpino</a>
- PicoRV32 <a href="https://github.com/cliffordwolf/picorv32">https://github.com/cliffordwolf/picorv32</a>
- Many, many more commercial and open source RISC-V cores

## Customizing RISC-V

- Modify the tunable parameters of an existing core
- Implement a RISC-V based accelerator
- Implement your own RISC-V instruction set extension (Ch. 9, User Spec)
- Implement your own RISC-V core

# RISC-V Software Landscape

### RISC-V Emulation/Simulation

- Spike RISC-V ISA simulator (riscv-isa-sim)
- **QEMU** Full system and user emulation
- RISCVEMU Boot RISC-V/Linux in your browser
- gem5 (in progress) Full-system simulator

### Spike and QEMU

- Spike is great for prototyping hardware features
  - Simple code base
  - Easy to add instructions
- QEMU is a better tool for software development
  - Faster emulation
  - Handy debugging features (e.g., GDB stub, monitor console)

### RISC-V Toolchain

#### Binutils

RISC-V support will be in 2.28

#### • GCC

 Steering Committee accepted port for inclusion. Port will likely make it into 7.1

### LLVM (in progress)

Some patches upstream. Codegen patches in progress.

## RISC-V OS and Firmware (Ports in Progress)

- Firmware: coreboot and UEFI
- Linux: Fedora, Debian, Gentoo, Yocto/Poky
- **BSD**: FreeBSD, NetBSD
- Other OS: seL4, Genode, HelenOS, Zephyr, FreeRTOS, RTEMS, MyNewt
- FreeBSD and Zephyr have upstream RISC-V support

### RISC-V Language Efforts

- OpenJDK
- JikesRVM
- Go
- Ocaml
- Haskell
- CompCert

### What's Next?

- Preparing Linux kernel, glibc, QEMU patches for upstreaming
- Standards work (ABI, Privileged spec)
- Improving/upstreaming Linux distribution support
- Porting more software packages to RISC-V

# We need your help to push RISC-V forward.

### RISC-V Resources

- GitHub: <a href="https://github.com/riscv">https://github.com/riscv</a>
- Mailing Lists: <a href="http://riscv.org/mailing-lists">http://riscv.org/mailing-lists</a>
- RISC-V Workshop Proceedings: <a href="http://riscv.org/category/workshops/proceedings">http://riscv.org/category/workshops/proceedings</a>
- Stack Overflow: <a href="http://stackoverflow.com/questions/tagged/riscv">http://stackoverflow.com/questions/tagged/riscv</a>
- "The Case for Open Instruction Sets", Microprocessor Report
- "RISC-V Offers Simple, Modular ISA", Microprocessor Report
- "An Agile Approach to Building RISC-V Microprocessors", IEEE Micro

### Upcoming RISC-V Events

- May 9-10: 6th RISC-V Workshop (Shanghai, China)
- Mar. 14-16: Embedded World
- Apr. 2-3: IIT Madras RISC-V Conference (Chennai, India)
- Sept. 8-10: ORConf (Hebden Bridge, UK)

### Summary

- RISC-V is an open instruction set specification
- Several options for open-source RISC-V SoCs
- RISC-V software stack is steadily coming together
- We are looking for contributors

### Questions?

- Takeaway: You should hack on RISC-V!
- Contact info:
  - arun.thomas@acm.org @arunthomas
- See you at one of the many 2017 RISC-V events!

