RISC-V ISA & Foundation Overview

Rick O'Connor

rickoco@riscv.org

http://www.riscv.org

Outline

- RISC-V ISA Overview
- RISC-V Foundation Overview & Growth
- RISC-V Use Case Examples
  - NVIDIA and Western Digital
- Summary

RISC-V Background

- In 2010, after many years and many projects using MIPS, SPARC, and x86 as basis of research, it was time for the Computer Science team at UC Berkeley to look at what ISAs to use for their next set of projects
- Obvious choices: x86 and ARM
  - x86 impossible – too complex, IP issues
  - ARM mostly impossible – complex, IP issues
- So UC Berkeley started “3-month project” during the summer of 2010 to develop their own clean-slate ISA

RISC-V Background (cont'd)

- Four years later, in May of 2014, UC Berkeley released frozen base user spec
  - many tapeouts and several research publications along the way
- The name RISC-V (pronounced *risk-five*), was chosen to represent the fifth major RISC ISA design effort at UC Berkeley
  - RISC-I, RISC-II, SOAR, and SPUR were the first four projects with the original RISC-I publications dating back to 1981
- In August 2015, articles of incorporation were filed to create a non-profit RISC-V Foundation to govern the ISA

Why Instruction Set Architectures matter

- Why are 99%+ of laptops/desktops/servers based on AMD x86-64 ISA (over 95%+ built by Intel)?
- Why are 99%+ of mobile phones + tablets based on ARM v7/v8 ISA?
- Why can't Intel sell mobile chips?
- Why can't ARM vendors sell servers?
- How can IBM still be selling mainframes?
- ISA is most important interface in a computer system
  - Where software meets hardware

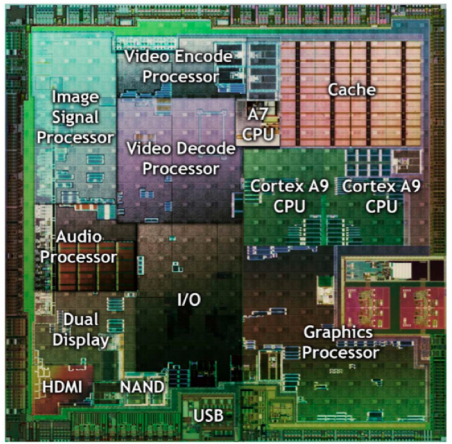
Open Software / Standards Work!

Field	*Standard*	*Free, Open Impl.*	*Proprietary Impl.*
Networking	Ethernet, TCP/IP	Many	Many
OS	Posix	Linux, FreeBSD	M/S Windows
Compilers	C	gcc, LLVM	Intel icc, ARMcc
Databases	SQL	MySQL, PostgresSQL	Oracle 12C, M/S DB2
Graphics	OpenGL	Mesa3D	M/S DirectX
ISA	??????	--	x86, ARM

- Why are there no successful free & open ISA standards and free & open implementations, like other fields?

Most CPU chips are SoCs with many ISAs

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



NVIDIA Tegra SoC

- Apps processor ISA too large for base accelerator ISA
- IP bought from different places, each proprietary ISA
- Home-grown ISA cores
- Over a dozen ISAs on some SoCs – each with unique software stack

Do we need all these different ISAs?

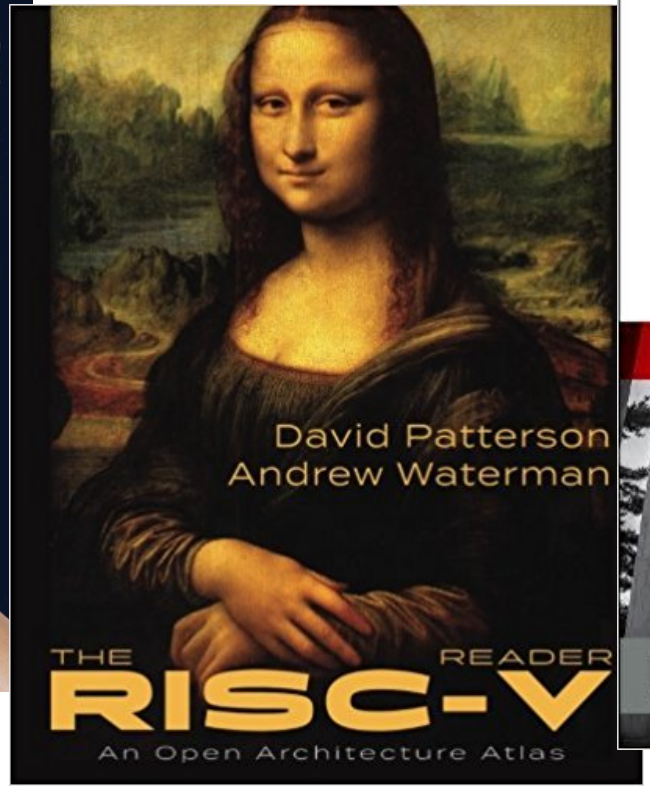
Must they be proprietary?

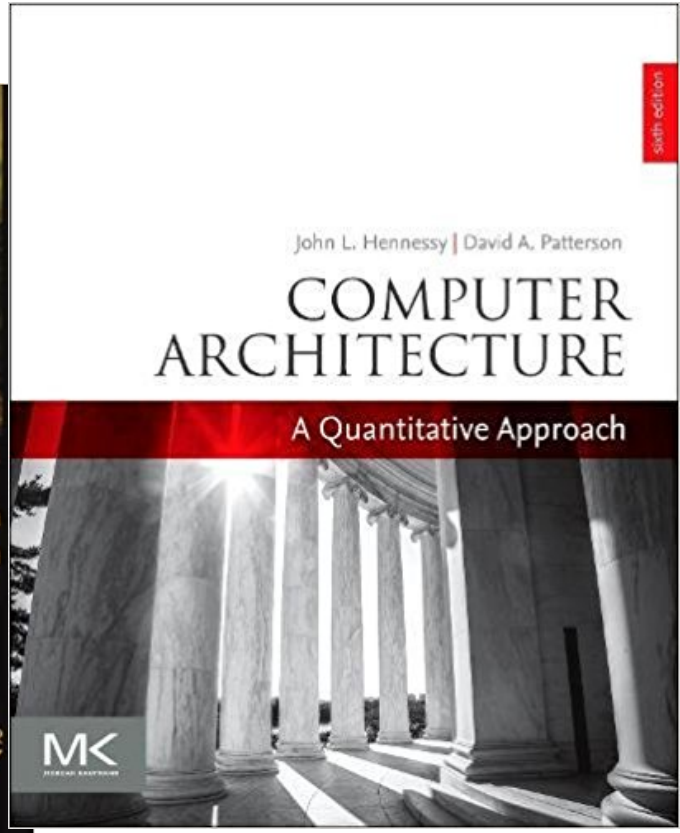
What if there was one free and open ISA everyone could use for everything?

What is RISC-V?

- A high-quality, license-free, royalty-free RISC ISA specification originally from UC Berkeley
- Standard maintained by non-profit RISC-V Foundation
- Suitable for all types of computing system, microcontrollers to supercomputers
- Numerous proprietary and open-source cores
- Experiencing rapid uptake in industry and academia
- Supported by growing shared software ecosystem
- A work in progress…

RISC-V in Education, new books!





What's Different about RISC-V?

- ***Simple***
  - Far smaller than other commercial ISAs
- ***Clean-slate design***
  - Clear separation between user and privileged ISA
  - Avoids µarchitecture or technology-dependent features
- A ***modular*** ISA
  - Small standard base ISA
  - Multiple standard extensions
- Designed for ***extensibility/specialization***
  - Variable-length instruction encoding
  - Vast opcode space available for instruction-set extensions
- ***Stable***
  - Base and standard extensions are frozen
  - Additions via optional extensions, not new versions

RISC-V Base Plus Standard Extensions

- Four base integer ISAs
  - RV32E, RV32I, RV64I, RV128I
  - Only <50 hardware instructions needed for base
- 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
  - C: compressed 16b encodings for 32b instructions
- All the above are a fairly standard RISC encoding in a fixed 32-bit instruction format

RV32I

RISC-V ①

Base Integer Instructions (32|64|128)			
Category	*Name*	*Fmt*	*RV{32|64|128}I Base*
Loads	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`
Stores	Store Byte	S	`SB rs1,rs2,imm`
	Store Halfword	S	`SH rs1,rs2,imm`
	Store Word	S	`S{W|D|Q} rs1,rs2,imm`
Shifts	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`
Arithmetic	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`
Logical	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`
Compare	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`
Branches	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`
Jump & Link	J&L	UJ	`JAL rd,imm`
	Jump & Link Register	I	`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 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`

② ③ **RISC-V Reference Card** ④

+14
Privileged

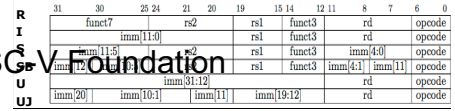
+ 8 for M

+ 34
for F, D, Q

+ 46 for C

+ 11 for A

32-bit Instruction Formats



RV32I / RV64I / RV128I + M, A, F, D, Q, C

RISC-V ① ② ③ **RISC-V Reference Card ④**

Base Integer Instructions (32|64|128)

Category	Name	Fmt	RV{32|64|128}I Base
Loads	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
Stores	Store Byte	S	SB rs1,rs2,imm
	Store Halfword	S	SH rs1,rs2,imm
	Store Word	S	S{W|D|Q} rs1,rs2,imm
Shifts	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
Arithmetic	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
Logical	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
Compare	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
Branches	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
Jump & Link	J&L	UJ	JAL rd,imm
	Jump & Link Register	I	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 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 RETired upper Half	I	RDINSTRETH rd

RV Privileged Instructions (32|64|128)

Category	Name	Fmt	RV mnemonic
CSR Access	Atomic R/W	R	CSRRW rd,csr,rs1
	Atomic Read & Set Bit	R	CSRRS rd,csr,rs1
	Atomic Read & Clear Bit	R	CSRRC rd,csr,rs1
	Atomic R/W Imm	R	CSRRWI rd,csr,imm
	Atomic Read & Set Bit Imm	R	CSRRSI rd,csr,imm
	Atomic Read & Clear Bit Imm	R	CSRRCI rd,csr,imm
Change Level	Env. Call	R	ECALL
	Environment Breakpoint	R	EBREAK
	Environment Return	R	ERET
Trap Redirect	to Superviso	R	MRTS
	Redirect Trap to Hypervisor	R	MRTH
	Hypervisor Trap to Supervisor	R	HRTS
Interrupt	Wait for Interrup	R	WFI
MMU	Supervisor FENCE	R	SFENCE.VM rs1

Optional Multiply-Divide Extension: RV32M

Category	Name	Fmt	RV32M (Mult-Div)
Multiply	MULtiply	R	MUL{|W|D} rd,rs1,rs2
	MULtiply upper Half	R	MULH rd,rs1,rs2
	MULtiply Half Sign/Uns	R	MULHSU rd,rs1,rs2
	MULtiply upper Half Uns	R	MULHU rd,rs1,rs2
Divide	DIVide	R	DIV{|W|D} rd,rs1,rs2
	DIVide Unsigned	R	DIVU rd,rs1,rs2
Remainder	REMainder	R	REM{|W|D} rd,rs1,rs2
	REMainder Unsigned	R	REMU{|W|D} rd,rs1,rs2

Optional Atomic Instruction Extension: RVA

Category	Name	Fmt	RV{32|64|128}A (Atomic)
Load	Load Reserved	R	LR.{W|D|Q} rd,rs1
Store	Store Conditiona	R	SC.{W|D|Q} rd,rs1,rs2
Swap	SWAP	R	AMOSWAP.{W|D|Q} rd,rs1,rs2
Add	ADD	R	AMOADD.{W|D|Q} rd,rs1,rs2
Logical	XOR	R	AMOXOR.{W|D|Q} rd,rs1,rs2
	AND	R	AMOAND.{W|D|Q} rd,rs1,rs2
	OR	R	AMOOR.{W|D|Q} rd,rs1,rs2
Min/Max	MINimum	R	AMOMIN.{W|D|Q} rd,rs1,rs2
	MAXimum	R	AMOMAX.{W|D|Q} rd,rs1,rs2
	MINimum Unsigned	R	AMOMINU.{W|D|Q} rd,rs1,rs2
	MAXimum Unsigned	R	AMOMAXU.{W|D|Q} rd,rs1,rs2

3 Optional FP Extensions: RV32{F|D|Q}

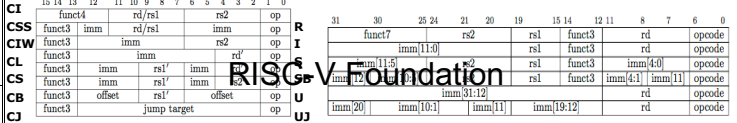
Category	Name	Fmt	RV{F|D|Q} (HP/SP,DP,QP)
Load	Load	I	FL{W,D,Q} rd,rs1,imm
Store	Store	S	FS{W,D,Q} rs1,rs2,imm
Arithmetic	ADD	R	FADD.{S|D|Q} rd,rs1,rs2
	SUBtract	R	FSUB.{S|D|Q} rd,rs1,rs2
	MULtiply	R	FMUL.{S|D|Q} rd,rs1,rs2
	DIVide	R	FDIV.{S|D|Q} rd,rs1,rs2
	SQuare RooT	R	FSQRT.{S|D|Q} rd,rs1
Mul-Add	Multiply-ADD	R	FMADD.{S|D|Q} rd,rs1,rs2,rs3
	Multiply-SUBtract	R	FMSUB.{S|D|Q} rd,rs1,rs2,rs3
	Negative Multiply-SUBtract	R	FMNSUB.{S|D|Q} rd,rs1,rs2,rs3
	Negative Multiply-ADD	R	FMNADD.{S|D|Q} rd,rs1,rs2,rs3
Sign Inject	SiGN source	R	FSGNJ.{S|D|Q} rd,rs1,rs2
	Negative SiGN source	R	FSGNJN.{S|D|Q} rd,rs1,rs2
	Xor SiGN source	R	FSGNJX.{S|D|Q} rd,rs1,rs2
Min/Max	MINimum	R	FMIN.{S|D|Q} rd,rs1,rs2
	MAXimum	R	FMAX.{S|D|Q} rd,rs1,rs2
Compare	Compare Float =	R	FEQ.{S|D|Q} rd,rs1,rs2
	Compare Float <	R	FLT.{S|D|Q} rd,rs1,rs2
	Compare Float ≤	R	FLE.{S|D|Q} rd,rs1,rs2
Categorize	Classify Type	R	FCLASS.{S|D|Q} rd,rs1
Move	Move from Integer	R	FMV.S.X rd,rs1
	Move to Integer	R	FMV.X.S rd,rs1
Convert	Convert from Int	R	FCVT.{S|D|Q}.W rd,rs1
	Convert from Int Unsigned	R	FCVT.{S|D|Q}.WU rd,rs1
	Convert to Int	R	FCVT.W.{S|D|Q} rd,rs1
	Convert to Int Unsigned	R	FCVT.WU.{S|D|Q} rd,rs1
Configuration	Read Stat	R	FRCSR rd
	Read Rounding Mode	R	FRRM rd
	Read Flags	R	FRFLAGS rd
	Swap Status Reg	R	FSCSR rd,rs1
	Swap Rounding Mode	R	FSRM rd,rs1
	Swap Flags	R	FSFLAGS rd,rs1
	Swap Rounding Mode Imm	I	FSRMI rd,imm
	Swap Flags Imm	I	FSFLAGSI rd,imm

+ 2 for 64{F|D|Q}/ 128{F|D|Q}

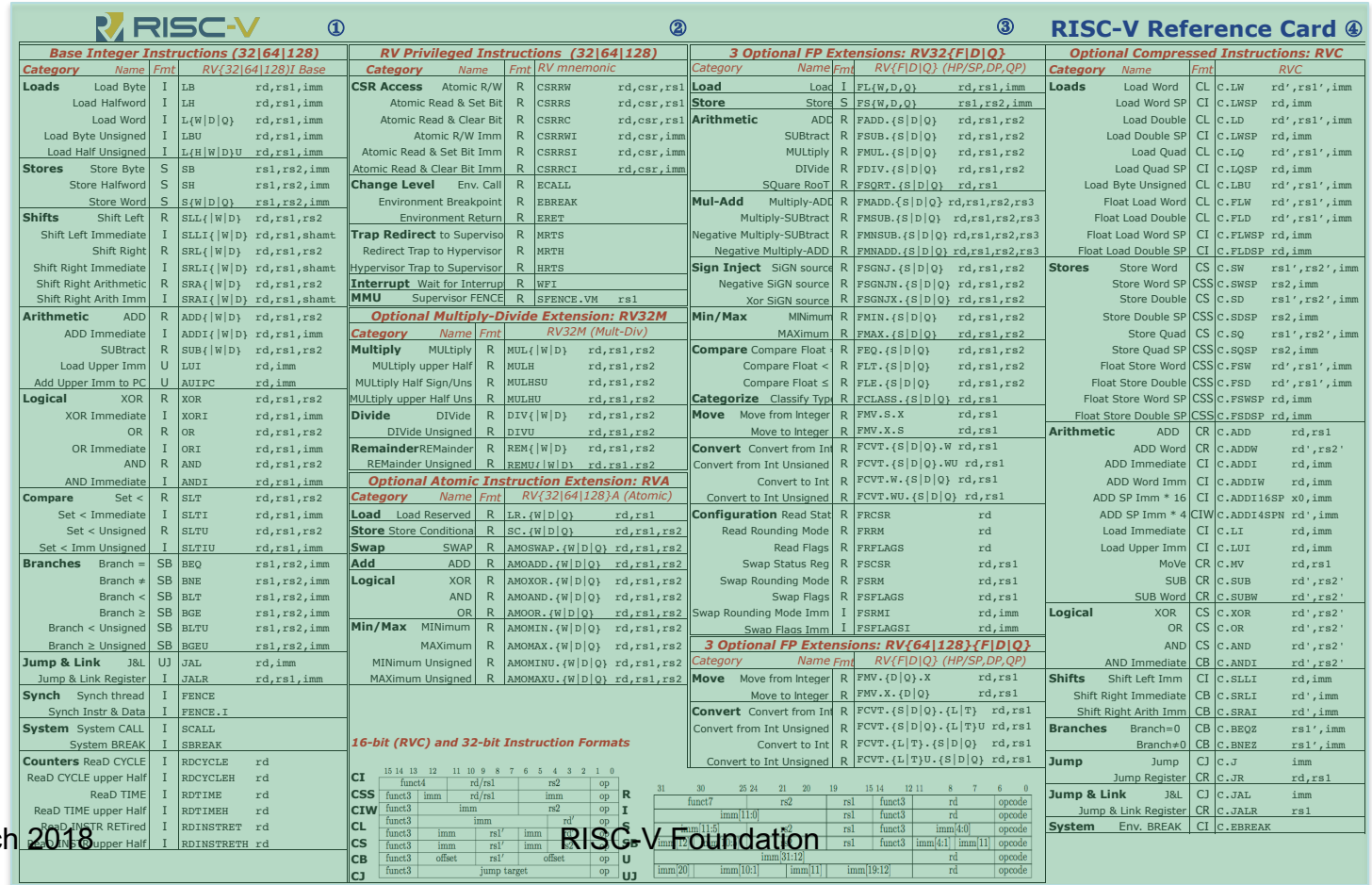
Optional Compressed Instructions: RVC

Category	Name	Fmt	RVC
Loads	Load Word	CL	C.LW rd',rs1',imm
	Load Word SP	CI	C.LWSP rd,imm
	Load Double	CL	C.LD rd',rs1',imm
	Load Double SP	CI	C.LWSP rd,imm
	Load Quad	CL	C.LQ rd',rs1',imm
	Load Quad SP	CI	C.LQSP rd,imm
	Load Byte Unsigned	CL	C.LBU rd',rs1',imm
	Float Load Word	CL	C.FLW rd',rs1',imm
	Float Load Double	CL	C.FLD rd',rs1',imm
	Float Load Word SP	CI	C.FLWSP rd,imm
	Float Load Double SP	CI	C.FLDSP rd,imm
Stores	Store Word	CS	C.SW rs1',rs2',imm
	Store Word SP	CSS	C.SWSP rs2,imm
	Store Double	CS	C.SD rs1',rs2',imm
	Store Double SP	CSS	C.SDSP rs2,imm
	Store Quad	CS	C.SQ rs1',rs2',imm
	Store Quad SP	CSS	C.SQSP rs2,imm
	Float Store Word	CSS	C.FSW rd',rs1',imm
	Float Store Double	CSS	C.FSD rd',rs1',imm
	Float Store Word SP	CSS	C.FSWSP rd,imm
	Float Store Double SP	CSS	C.FSDSP rd,imm
Arithmetic	ADD	CR	C.ADD rd,rs1
	ADD Word	CR	C.ADDW rd',rs2'
	ADD Immediate	CI	C.ADDI rd,imm
	ADD Word Imm	CI	C.ADDIW rd,imm
	ADD SP Imm * 16	CI	C.ADDI16SP x0,imm
	ADD SP Imm * 4	CIW	C.ADDI4SPN rd',imm
	Load Immediate	CI	C.LI rd,imm
	Load Upper Imm	CI	C.LUI rd,imm
	MoVe	CR	C.MV rd,rs1
	SUB	CR	C.SUB rd',rs2'
	SUB Word	CR	C.SUBW rd',rs2'
Logical	XOR	CS	C.XOR rd',rs2'
	OR	CS	C.OR rd',rs2'
	AND	CS	C.AND rd',rs2'
	AND Immediate	CB	C.ANDI rd',rs2'
Shifts	Shift Left Imm	CI	C.SLLI rd,imm
	Shift Right Immediate	CB	C.SRLI rd',imm
	Shift Right Arith Imm	CB	C.SRAI rd',imm
Branches	Branch=0	CB	C.BEQZ rs1',imm
	Branch≠0	CB	C.BNEZ rs1',imm
Jump	Jump	CJ	C.J imm
	Jump Register	CR	C.JR rd,rs1
Jump & Link	J&L	CJ	C.JAL imm
	Jump & Link Register	CR	C.JALR rs1
System	Env. BREAK	CI	C.EBREAK

16-bit (RVC) and 32-bit Instruction Formats



RV32I / RV64I / RV128I + M, A, F, D, Q, C
RISC-V "Green Card"



RISC-V ① ② ③ RISC-V Reference Card ④

Base Integer Instructions (32|64|128)

Category	Name	Fmt	RV{32|64|128}I Base
Loads	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
Stores	Store Byte	S	SB rs1,rs2,imm
	Store Halfword	S	SH rs1,rs2,imm
	Store Word	S	S{W|D|Q} rs1,rs2,imm
Shifts	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
Arithmetic	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
Logical	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
Compare	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
Branches	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
Jump & Link	J&L	UJ	JAL rd,imm
	Jump & Link Register	I	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 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 RETired upper Half	I	RDINSTRETH rd

RV Privileged Instructions (32|64|128)

Category	Name	Fmt	RV mnemonic
CSR Access	Atomic R/W	R	CSRRW rd,csr,rs1
	Atomic Read & Set Bit	R	CSRRS rd,csr,rs1
	Atomic Read & Clear Bit	R	CSRRC rd,csr,rs1
	Atomic R/W Imm	R	CSRRWI rd,csr,imm
	Atomic Read & Set Bit Imm	R	CSRRSI rd,csr,imm
	Atomic Read & Clear Bit Imm	R	CSRRCI rd,csr,imm
Change Level	Env. Call	R	ECALL
	Environment Breakpoint	R	EBREAK
	Environment Return	R	ERET
Trap Redirect	to Superviso	R	MRTS
	Redirect Trap to Hypervisor	R	MRTH
	Hypervisor Trap to Supervisor	R	HRTS
Interrupt	Wait for Interrup	R	WFI
MMU	Supervisor FENCE	R	SFENCE.VM rs1

Optional Multiply-Divide Extension: RV32M

Category	Name	Fmt	RV32M (Mult-Div)
Multiply	MULtiply	R	MUL{|W|D} rd,rs1,rs2
	MULtiply upper Half	R	MULH rd,rs1,rs2
	MULtiply Half Sign/Uns	R	MULHSU rd,rs1,rs2
	MULtiply upper Half Uns	R	MULHU rd,rs1,rs2
Divide	DIVide	R	DIV{|W|D} rd,rs1,rs2
	DIVide Unsigned	R	DIVU rd,rs1,rs2
Remainder	REMainder	R	REM{|W|D} rd,rs1,rs2
	REMainder Unsigned	R	REMU{|W|D} rd,rs1,rs2

Optional Atomic Instruction Extension: RVA

Category	Name	Fmt	RV{32|64|128}A (Atomic)
Load	Load Reserved	R	LR.{W|D|Q} rd,rs1
Store	Store Conditiona	R	SC.{W|D|Q} rd,rs1,rs2
Swap	SWAP	R	AMOSWAP.{W|D|Q} rd,rs1,rs2
Add	ADD	R	AMOADD.{W|D|Q} rd,rs1,rs2
Logical	XOR	R	AMOXOR.{W|D|Q} rd,rs1,rs2
	AND	R	AMOAND.{W|D|Q} rd,rs1,rs2
	OR	R	AMOOR.{W|D|Q} rd,rs1,rs2
Min/Max	MINimum	R	AMOMIN.{W|D|Q} rd,rs1,rs2
	MAXimum	R	AMOMAX.{W|D|Q} rd,rs1,rs2
	MINimum Unsigned	R	AMOMINU.{W|D|Q} rd,rs1,rs2
	MAXimum Unsigned	R	AMOMAXU.{W|D|Q} rd,rs1,rs2

3 Optional FP Extensions: RV32{F|D|Q}

Category	Name	Fmt	RV{F|D|Q} (HP/SP,DP,QP)
Load	Load	I	FL{W,D,Q} rd,rs1,imm
Store	Store	S	FS{W,D,Q} rs1,rs2,imm
Arithmetic	ADD	R	FADD.{S|D|Q} rd,rs1,rs2
	SUBtract	R	FSUB.{S|D|Q} rd,rs1,rs2
	MULtiply	R	FMUL.{S|D|Q} rd,rs1,rs2
	DIVide	R	FDIV.{S|D|Q} rd,rs1,rs2
	SQuare RooT	R	FSQRT.{S|D|Q} rd,rs1
Mul-Add	Multiply-ADD	R	FMADD.{S|D|Q} rd,rs1,rs2,rs3
	Multiply-SUBtract	R	FMSUB.{S|D|Q} rd,rs1,rs2,rs3
	Negative Multiply-SUBtract	R	FMNSUB.{S|D|Q} rd,rs1,rs2,rs3
	Negative Multiply-ADD	R	FMNADD.{S|D|Q} rd,rs1,rs2,rs3
Sign Inject	SiGN source	R	FSGNJ.{S|D|Q} rd,rs1,rs2
	Negative SiGN source	R	FSGNJN.{S|D|Q} rd,rs1,rs2
	Xor SiGN source	R	FSGNJX.{S|D|Q} rd,rs1,rs2
Min/Max	MINimum	R	FMIN.{S|D|Q} rd,rs1,rs2
	MAXimum	R	FMAX.{S|D|Q} rd,rs1,rs2
Compare	Compare Float =	R	FEQ.{S|D|Q} rd,rs1,rs2
	Compare Float <	R	FLT.{S|D|Q} rd,rs1,rs2
	Compare Float ≤	R	FLE.{S|D|Q} rd,rs1,rs2
Categorize	Classify Type	R	FCLASS.{S|D|Q} rd,rs1
Move	Move from Integer	R	FMV.S.X rd,rs1
	Move to Integer	R	FMV.X.S rd,rs1
Convert	Convert from Int	R	FCVT.{S|D|Q}.W rd,rs1
	Convert from Int Unsigned	R	FCVT.{S|D|Q}.WU rd,rs1
	Convert to Int	R	FCVT.W.{S|D|Q} rd,rs1
	Convert to Int Unsigned	R	FCVT.WU.{S|D|Q} rd,rs1
Configuration	Read Stat	R	FRCSR rd
	Read Rounding Mode	R	FRRM rd
	Read Flags	R	FRFLAGS rd
	Swap Status Reg	R	FSCSR rd,rs1
	Swap Rounding Mode	R	FSRM rd,rs1
	Swap Flags	R	FSFLAGS rd,rs1
	Swap Rounding Mode Imm	I	FSRMI rd,imm
	Swap Flags Imm	I	FSFLAGSI rd,imm

3 Optional FP Extensions: RV{64|128}{F|D|Q}

Category	Name	Fmt	RV{F|D|Q} (HP/SP,DP,QP)
Move	Move from Integer	R	FMV.{D|Q}.X rd,rs1
	Move to Integer	R	FMV.X.{D|Q} rd,rs1
Convert	Convert from Int	R	FCVT.{S|D|Q}.{L|T} rd,rs1
	Convert from Int Unsigned	R	FCVT.{S|D|Q}.{L|T}U rd,rs1
	Convert to Int	R	FCVT.{L|T}.{S|D|Q} rd,rs1
	Convert to Int Unsigned	R	FCVT.{L|T}U.{S|D|Q} rd,rs1

Optional Compressed Instructions: RVC

Category	Name	Fmt	RVC
Loads	Load Word	CL	C.LW rd',rs1',imm
	Load Word SP	CI	C.LWSP rd,imm
	Load Double	CL	C.LD rd',rs1',imm
	Load Double SP	CI	C.LWSP rd,imm
	Load Quad	CL	C.LQ rd',rs1',imm
	Load Quad SP	CI	C.LQSP rd,imm
	Load Byte Unsigned	CL	C.LBU rd',rs1',imm
	Float Load Word	CL	C.FLW rd',rs1',imm
	Float Load Double	CL	C.FLD rd',rs1',imm
	Float Load Word SP	CI	C.FLWSP rd,imm
	Float Load Double SP	CI	C.FLDSP rd,imm
Stores	Store Word	CS	C.SW rs1',rs2',imm
	Store Word SP	CSS	C.SWSP rs2,imm
	Store Double	CS	C.SD rs1',rs2',imm
	Store Double SP	CSS	C.SDSP rs2,imm
	Store Quad	CS	C.SQ rs1',rs2',imm
	Store Quad SP	CSS	C.SQSP rs2,imm
	Float Store Word	CSS	C.FSW rd',rs1',imm
	Float Store Double	CSS	C.FSD rd',rs1',imm
	Float Store Word SP	CSS	C.FSWSP rd,imm
	Float Store Double SP	CSS	C.FSDSP rd,imm
Arithmetic	ADD	CR	C.ADD rd,rs1
	ADD Word	CR	C.ADDW rd',rs2'
	ADD Immediate	CI	C.ADDI rd,imm
	ADD Word Imm	CI	C.ADDIW rd,imm
	ADD SP Imm * 16	CI	C.ADDI16SP x0,imm
	ADD SP Imm * 4	CIW	C.ADDI4SPN rd',imm
	Load Immediate	CI	C.LI rd,imm
	Load Upper Imm	CI	C.LUI rd,imm
	MoVe	CR	C.MV rd,rs1
	SUB	CR	C.SUB rd',rs2'
	SUB Word	CR	C.SUBW rd',rs2'
Logical	XOR	CS	C.XOR rd',rs2'
	OR	CS	C.OR rd',rs2'
	AND	CS	C.AND rd',rs2'
	AND Immediate	CB	C.ANDI rd',rs2'
Shifts	Shift Left Imm	CI	C.SLLI rd,imm
	Shift Right Immediate	CB	C.SRLI rd',imm
	Shift Right Arith Imm	CB	C.SRAI rd',imm
Branches	Branch=0	CB	C.BEQZ rs1',imm
	Branch≠0	CB	C.BNEZ rs1',imm
Jump	Jump	CJ	C.J imm
	Jump Register	CR	C.JR rd,rs1
Jump & Link	J&L	CJ	C.JAL imm
	Jump & Link Register	CR	C.JALR rs1
System	Env. BREAK	CI	C.EBREAK

16-bit (RVC) and 32-bit Instruction Formats

Outline

- RISC-V ISA Overview
- RISC-V Foundation Overview & Growth
- RISC-V Use Case Examples
  - NVIDIA and Western Digital
- Summary

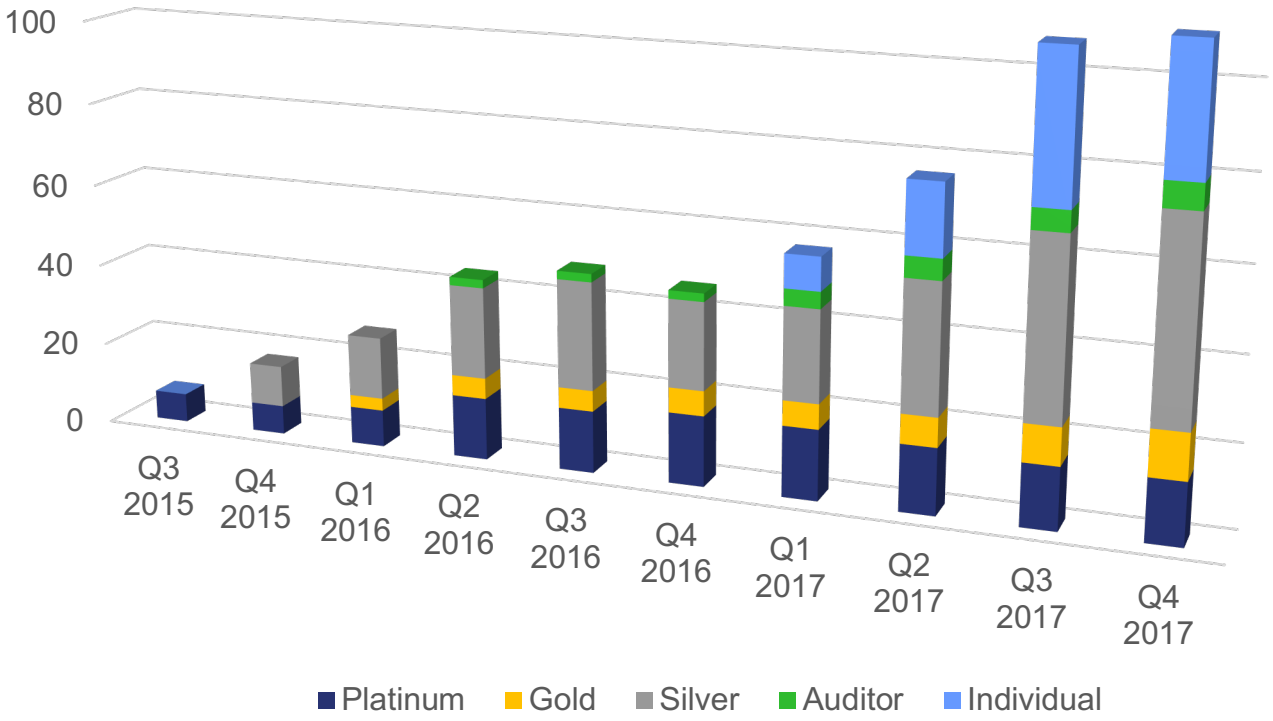
RISC-V Foundation Overview

- Incorporated August, 2015 as a 501c6 non-profit Foundation
- Membership Agreement & Bylaws ratified December 2016
- The RISC-V ISA and related standards shall remain open and license-free to all parties
  - RISC-V ISA specifications shall always be publicly available as an online download
- The compatibility test suites shall always be publicly available as a source code download
- To protect the standard, only members (with commercial RISC-V products) of the Foundation in good standing can use “RISC-V” and associated trademarks, and only for devices that pass the tests in the open-source compatibility suites maintained by the Foundation


Berkeley Architecture Research
NXP
bluespec
SANECHIPS 中兴微电子
SiFive
Western Digital
Micron
cortus
Google
Rambus Cryptography Research
NVIDIA
Microsemi
SAMSUNG
C-SKY 中天微
QUALCOMM
RISC-V
Foundation 100+ Members
ESPRESSIF
MEDIATEK
SECURE RF
ANDES TECHNOLOGY
antmicro EMBEDDED SYSTEMS
runtime.io
NUS National University of Singapore
BAE SYSTEMS
Esperanto Technologies
inside secure DRIVING TRUST
galois
TRINAMIC MOTION CONTROL
SEGGER
ASHLING
GOWIN
VeriSilicon
REM
GRAY RESEARCH
ETH zürich
SIEMENS
INTRINSIX
MIT CSAIL
imperas
UBILITE Hello IoT World
MINRES TECHNOLOGIES
EMBECOSM
LAUTERBACH DEVELOPMENT TOOLS
DOVER MICROSYSTEMS
IDT
VectorBlox embedded supercomputing
Mentor A Siemens Business
SEAGATE
GREENWAVES TECHNOLOGIES
Rumble Development
BERKELEY LAB
Minima
XtremeEDA
codasip
EMDALO TECHNOLOGIES
Roa Logic
Syntacore Custom cores and tools
aselsan
优矽科技 UC TECH IP
expresslogic
ultraSoC
SH CONSULTING
Mellanox TECHNOLOGIES
lowRISC
ICT
GLOBALFOUNDRIES
Technolution
CSEM
LATTICE SEMICONDUCTOR
PRINCETON UNIVERSITY
TechanaLye
IBM
HUAWEI
CEVA

RISC-V Foundation Growth History

August 2015 to November 2017



Foundation Organization

- The Board of Directors consists of seven+ members, whose replacements are elected by the membership
- The Board can amend the By-Laws of the RISC-V foundation via a two-thirds affirmative vote
- The Board appoints chairs of ad-hoc committees to address issues concerning RISC-V, and has the final vote of approval of the recommendation of the ad-hoc committees.
  - Technical Committee Chair – Yunsup Lee, SiFive
  - Marketing Committee Chair – Ted Marena, Microsemi
- All members of committees must be members of the RISC-V Foundation

RISC-V Foundation Board of Directors

- Krste Asanović, Chairman
  - Professor in the EECS Department at UC Berkeley
- David Patterson, Vice-Chairman
  - Google Architect, Retired Professor Computer Science UC Berkeley
- Zvonimir Bandić
  - Senior Director of Next Generation Platform Technologies at Western Digital Corporation
- Charlie Hauck
  - CEO of Bluespec Inc.
- Rob Oshana
  - Director Global SW Development at NXP
- Frans Sijstermans
  - Vice President Engineering at NVIDIA
- Ted Speers
  - Technical Fellow, Head of Product Architecture for Microsemi's SoC Group

Some RISC-V Workshop Stats…

- 1st RISC-V workshop Jan 14-15, 2015 in Monterey, CA
  - Sold out: 144 (33 companies & 14 universities) Slides & videos can be found here
- 2nd RISC-V workshop Jun 29-30, 2015 at UC Berkeley, CA
  - Sold out: 120 (30 companies & 20 universities) Slides & videos can be found here
- 3rd RISC-V workshop Jan 5-6, 2016 at Oracle Redwood City, CA
  - Sold out: 157 (42 companies & 26 universities) Slides & videos can be found here
- 4th RISC-V workshop Jul 12-13, 2016 at MIT Cambridge, MA
  - Sold out: 252 (63 companies & 42 universities) Slides & videos can be found here
- 5th RISC-V workshop Nov 29-30, 2016 at Google Mountain View, CA
  - Sold out: 350 (107 companies & 29 universities) Slides & videos can be found here
- 6th RISC-V workshop May 8-11, 2017 at NVIDIA / SJTU Shanghai, China
  - Sold out: 287 (52 companies & 27 universities) Slides & videos can be found here
- 7th RISC-V workshop Nov 28-30, 2017 at Western Digital Milpitas, CA
  - Sold out: 498 (138 companies & 35 universities) Slides & videos can be found here

Outline

- RISC-V ISA Overview
- RISC-V Foundation Overview & Growth
- RISC-V Use Case Examples
  - NVIDIA and Western Digital
- Summary

6th RISC-V Workshop - May 8th – 11th , 2017 Shanghai China

Following slides excerpt from Frans Sijstermans, VP Engineering, NVIDIA Keynote Address - Full presentation is here

Falcon's history

- Embedded in 15+ designs
- Taped out in ~50 chips
- Shipped ~3 billion times
- No stop-ship bugs

Falcons shipped estimate	
dGPU Volume /year	50M*
Years Falcon shipping	10
Avg. #Falcons / GPU	10
Avg. NVIDIA market share	60%
Total shipped	3 billion

http://www.anandtech.com/show/10864/discrete-desktop-gpu-market-trends-q3-2016

Selecting the next architecture

Technical criteria

- >2x performance of Falcon
- <2x area cost of Falcon
- Support for caches as well tightly coupled memories
- 64-bit addresses
- Suitable for modern OS

Considered architectures

- ARM
- Imagination Technologies MIPS
- Synopsys ARC
- Cadence Tensilica
- RISC-V

Why RISC-V for Falcon Next

RISC-V is the only architecture that meets all our criteria
https://riscv.org/wp-content/uploads/2016/07/Tue1100_Nvidia_RISCV_Story_V2.pdf

Item	Requirement	ARM A53	ARM A9	ARM R5	RISC-V Rocket	NV RISC-V
Core perf	>2x falcon	Yes	Yes	Yes	Yes	Yes
Area (16ff)	<0.1mm^2	No	No	Yes	Yes	Yes
Security	Yes	TZ	TZ	No	Yes	Yes
TCM	Yes	Yes	No	Yes	No	Yes
L1 I/D $	Yes	Yes	Yes	Yes	Yes	Yes
Addressing	64bit	Yes	No	No	Yes	Yes
Extensible ISA	Yes	No	No	No	Yes	Yes
Safety (ECC/Parity)	Yes	Yes	Yes	Yes	Yes	Yes
Functional Simulation model	Yes	Yes	No	No	No	Yes

7th RISC-V Workshop - November 28th – 30th , 2017 Milpitas, California

Western Digital®

Following slides excerpt from Martin Fink, CTO, Western Digital Keynote Address

Full presentation is here

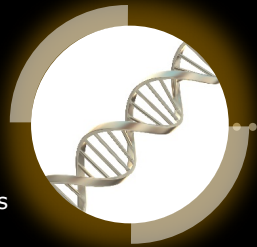
RISC-V Meets the Needs of Big Data and Fast Data

Big Data

Fast Data


Predictive Analytics


Autonomous Machines


Genomics

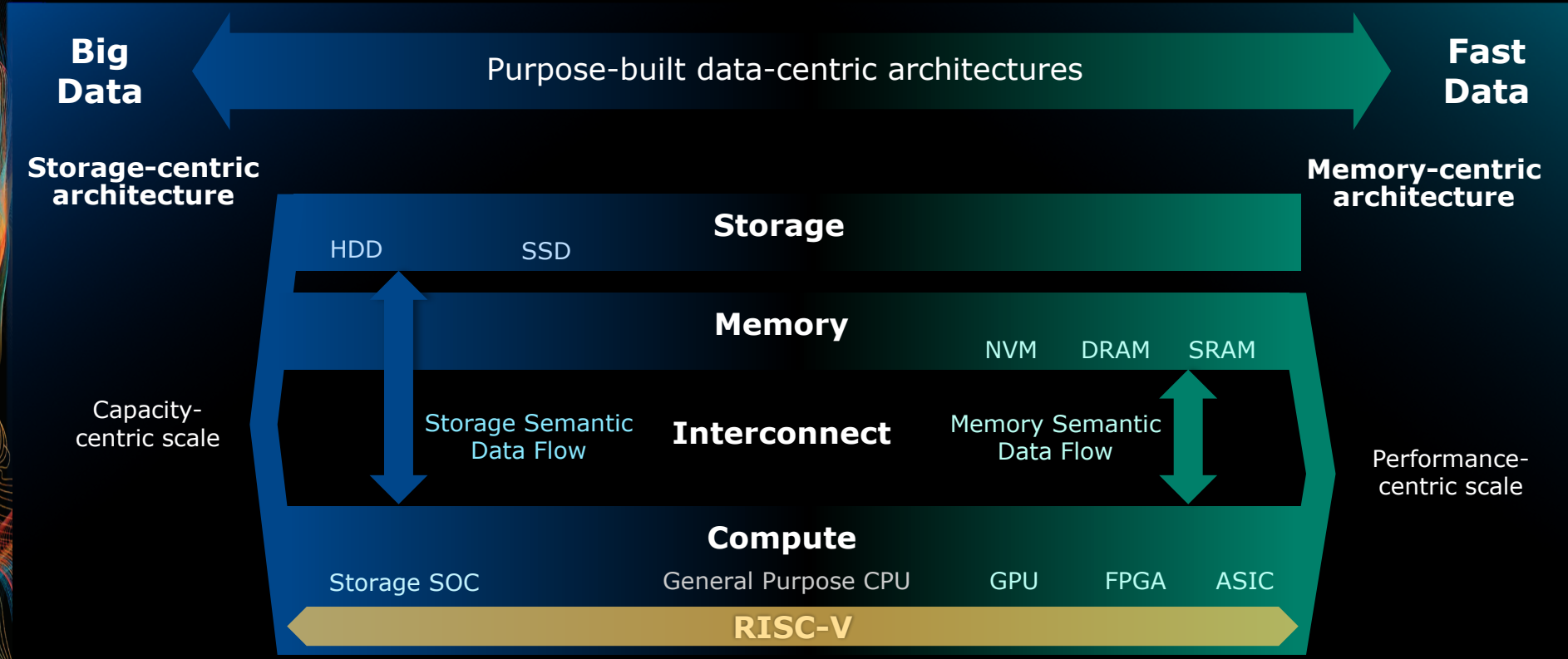
RISC-V


Safety & Security


Machine Learning


Private Exchange

RISC-V Enables Purpose-Built Environments for Big Data and Fast Data Applications



Driving Momentum

Western Digital ships in excess of
1 Billion cores per year
…and we expect to **double that**.

Western Digital.

Accelerating the RISC-V Ecosystem

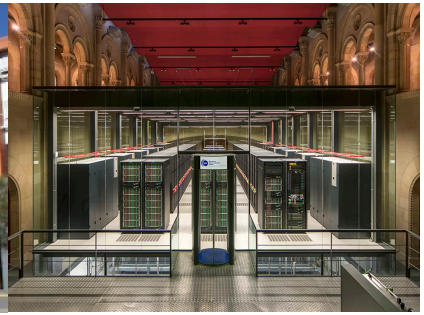
Western Digital to contribute one billion cores annually to fuel RISC-V

1. Support development of open source IP building blocks for the community
2. Actively partner and invest in the ecosystem
3. Accelerate development of purpose-built processors for a broad range of Big Data and Fast Data environments
4. Multi-year transition of Western Digital devices, platforms and systems to RISC-V purpose-built architectures

RISC-V Barcelona Workshop

May 7th – 10th, 2018

- Co-Hosted at the Barcelona Supercomputing Center (BSC) and Universitat Politècnica de Catalunya (UPC)
- Co-Sponsored by NXP and Western Digital

RISC-V ISA & Foundation Summary

- The free and open RISC-V ISA is enabling a new innovation frontier across all computing devices

- Strong Industry Support
  - 100+ members; Broad commercial and academic interest

- RISC-V Twitter http://twitter.com/risc_v @risc_v

- RISC-V LinkedIn Page
  - http://www.linkedin.com/company/risc-v-foundation