

RISC-V ISA

Additional information

RISC-V ISA is divided into extensions

I	Integer instructions (frozen)
E	Reduced number of registers
M	Multiplication and Division (frozen)
A	Atomic instructions (frozen)
F	Single-Precision Floating-Point (frozen)
D	Double-Precision Floating-Point (frozen)
C	Compressed Instructions (frozen)
X	Non Standard Extensions

- Kept very simple and extendable
 - Wide range of applications from IoT to HPC
- RV + word-width + extensions
 - RV32**IMC**: 32bit, integer, multiplication, compressed
- User specification:
 - Separated into extensions, only **I** is mandatory
- Privileged Specification (WIP):
 - Governs OS functionality: Exceptions, Interrupts
 - Virtual Addressing
 - Privilege Levels

Work continues on new RISC-V extensions

- Foundation members work in **task-groups**
- Dedicated task-groups
 - Formal specification
 - Memory Model
 - Marketing
 - External Debug Specification

Q Quad-precision Floating-Point

L Decimal Floating Point

B Bit Manipulation

T Transactional Memory

P Packed SIMD

J Dynamically Translated Languages

V Vector Operations

N User-Level Interrupts

Reduced Instruction Set: all in one page

Free & Open RISC-V Reference Card

Base Integer Instructions: RV32I, RV64I, and RV128I					RV Privileged Instructions				
Category	Name	Fmt	RV32I Base	+RV(64,128)	Category	Name	RV mnemonic		
Loads	Load Byte	I	LB rd,rs1,imm		CSR Access	Atomic R/W	CSRWR rd,csr,rs1		
	Load Halfword	I	LH rd,rs1,imm			Atomic Read & Set Bit	CSRRS rd,csr,rs1		
	Load Word	I	LD rd,rs1,imm	L(D)Q rd,rs1,imm		Atomic Read & Clear Bit	CSRRC rd,csr,rs1		
	Load Byte Unsigned	I	LBU rd,rs1,imm			Atomic R/W Imm	CSRRWI rd,csr,imm		
	Load Half Unsigned	I	LHU rd,rs1,imm	L(W)DU rd,rs1,imm		Atomic Read & Clear Imm	CSRRCI rd,csr,imm		
Stores	Store Byte	S	SB rs1,rs2,imm		Change Level	Env. Call	ECALL		
	Store Halfword	S	SH rs1,rs2,imm			Environment Base	EBREAK		
	Store Word	S	SW rs1,rs2,imm	S(D)Q rs1,rs2,imm		Environment Pointer	EPURK		
Shifts	Shift Left	R	SLL rd,rs1,rs2	SLL(W)D rd,rs1,rs2	Trap Redirect	Redirect to Supervisor	MRTS		
	Shift Left Immediate	I	SLLI rd,rs1,shamt	SLLI(W)D rd,rs1,shamt		Redirect Trap to Hypervisor	MRTH		
	Shift Right	R	SRL rd,rs1,rs2	SRL(W)D rd,rs1,rs2		Hypervisor Trap to Supervisor	BRTS		
	Shift Right Immediate	I	SRLI rd,rs1,shamt	SRLI(W)D rd,rs1,shamt		Interrupt Wait for Interrupt	WFI		
	Shift Right Arithmetic	R	SRA rd,rs1,rs2	SRA(W)D rd,rs1,rs2	MMU	Supervisor FENCE	BFENCE.VM rs1		
Arithmetic	Shift Right Arith Imm	I	SRAI rd,rs1,shamt	SRAI(W)D rd,rs1,shamt					
	ADD	I	ADD rd,rs1,rs2	ADD(W)D rd,rs1,rs2					
	ADD Immediate	I	ADDI rd,rs1,imm	ADDI(W)D rd,rs1,imm					
	SUBtract	I	SUB rd,rs1,rs2						
	Load Upper Imm	I	LUI rd,imm						
Logical	ADD Upper Imm to PC	I	AUIPC pc,imm						
	XOR	I	XOR rd,rs1,rs2						
	XOR Immediate	I	XORI rd,rs1,imm						
	OR	I	OR rd,rs1,rs2						
	OR Immediate	I	ORI rd,rs1,imm						
Compare	AND	I	AND rd,rs1,rs2						
	AND Immediate	I	ANDI rd,rs1,imm						
	Set <	I	SLT rd,rs1,rs2						
	Set < Immediate	I	SLTI rd,rs1,imm						
	Set < Unsigned	R	SLTU rd,rs1,rs2						
Branches	Set < Imm Unsigned	R	SLTIU rd,rs1,imm						
	Branch =	I	BEQ rs1,rs2,imm						
	Branch !=	I	BNE rs1,rs2,imm						
	Branch <	I	BLT rs1,rs2,imm						
	Branch <=	I	BGE rs1,rs2,imm						
Jump & Link	Branch <= Unsigned	I	BLTU rs1,rs2,imm						
	Branch <= Imm Unsigned	I	BGTU rs1,rs2,imm						
	Jump & Link Register	I	JAL rd,rs1,imm						
	Synch Thread	I	FENCE						
	Synch Instr & Data	I	FENCE.I						
System	System CALL	I	SCALL						
	System BREAK	I	SBREAK						
	Counters Read CYCLE	I	RDYCYCLE rd						
	Read CYCLE upper Half	I	RDYCYCLEH rd						
	Read TIME	I	RDTIME rd						
Counters	Read TIME upper Half	I	RDTIMEH rd						
	Read INSTR RETired	I	RDINSTRET rd						
	Read INSTR RETired upper Half	I	RDINSTRETH rd						

Optional Compressed (16-bit) Instruction Extension: RVC				
Category	Name	Fmt	RVC	RVI equivalent
Loads	Load Word	CL	C.LW rd',rs1',imm	LW rd',rs1',imm*4
	Load Word SP	CL	C.LWSP rd,imm	LW rd,sp,imm*4
	Load Double	CL	C.LD rd',rs1',imm	LD rd',rs1',imm*8
	Load Double SP	CL	C.LDSP rd,imm	LD rd,sp,imm*8
	Load Quad	CL	C.LQ rd',rs1',imm	LQ rd',rs1',imm*16
Stores	Load Quad SP	CL	C.LQSP rd,imm	LQ rd,sp,imm*16
	Store Word	CS	C.SW rs1',rs2',imm	SW rs1',rs2',imm*4
	Store Word SP	CS	C.SWSP rs2,imm	SW rs2,sp,imm*4
	Store Double	CS	C.SD rs1',rs2',imm	SD rs1',rs2',imm*8
	Store Double SP	CS	C.SDSP rs2,imm	SD rs2,sp,imm*8
Arithmetic	Store Quad	CS	C.SQ rs1',rs2',imm	SQ rs1',rs2',imm*16
	Store Quad SP	CS	C.SQSP rs2,imm	SQ rs2,sp,imm*16
	ADD	CR	C.ADD rd,rs1	ADD rd,rs1,imm
	ADD Word	CR	C.ADDW rd,rs1	ADDW rd,rs1,imm
	ADD Immediate	CR	C.ADDI rd,imm	ADDI rd,rs1,imm
Shifts	ADD Word Imm	CR	C.ADDWI rd,imm	ADDWI rd,rs1,imm
	ADD SP Imm * 13	CR	C.ADDI4SPN rd,imm	ADDI rd,sp,imm*4
	ADD SP Imm * 4	CR	C.ADDI4SPN rd,imm	ADDI rd,sp,imm*4
	Load Immediate	CR	C.LI rd,imm	LD rd,x0,imm
	Load Upper Imm	CR	C.LUI rd,imm	LUI rd,imm
Branches	MoVe	CR	C.MV rd,rs1	ADD rd,rs1,x0
	Shift Left Imm	CR	C.SLLI rd,imm	SUB rd,rs1,rs1
	Branch=0	CB	C.BEQZ rs1',x0,imm	SLLI rd,rs1,imm
	Branch!=0	CB	C.BNEZ rs1',x0,imm	SLLI rd,rs1,imm
	Jump	CR	C.CJ imm	JAL x0,imm
Jump & Link	Jump Register	CR	C.JR rd,rs1	JALR x0,rs1,0
	Jump & Link	CR	C.JAL imm	JAL ra,imm
	Jump & Link Register	CR	C.JALR rs1	JALR ra,rs1,0
	System Env. BREAK	CI	C.EBREAK	EBREAK

32-bit Instruction Format				
31	30	25	24	19
R	func4	rs2	rs1	func3
I	imm[7]	imm[10]	rd	opcode
S	imm[11:5]	rs2	rs1	func3
SB	imm[12]	imm[10:5]	rs2	imm[4:1]
U	imm[20]	imm[10:1]	imm[11]	imm[19:12]

RV32M (Multiply-Divide) Instructions				
Category	Name	Fmt	RV32M (Multiply-Divide)	+RV(64,128)
Multiply	MUL	R	MUL rd,rs1,rs2	MUL(W)D rd,rs1,rs2
	Multiply upper Half	R	MULH rd,rs1,rs2	
	Multiply Half Sign	R	MULHSU rd,rs1,rs2	
	Multiply upper Half	R	MULHU rd,rs1,rs2	
	Multiply upper Half	R	MULHSU rd,rs1,rs2	
Divide	DIVide	R	DIV rd,rs1,rs2	DIV(W)D rd,rs1,rs2
	DIVide Unsigned	R	DIVU rd,rs1,rs2	
	REMAinder	R	REM rd,rs1,rs2	REM(W)D rd,rs1,rs2
Remainder	REMAinder Unsigned	R	REMU rd,rs1,rs2	REMU(W)D rd,rs1,rs2

Optional Atomic Instruction Extension: RVA				
Category	Name	Fmt	RV32A (Atomic)	+RV(64,128)
Load	Load Reserved	R	LR.W rd,rs1	LR.(D)Q rd,rs1
	Store Conditional	R	SC.W rd,rs1,rs2	SC.(D)Q rd,rs1,rs2
Store	Swap	R	AMOSWAP.W rd,rs1,rs2	AMOSWAP.(D)Q rd,rs1,rs2
	ADD	R	AMOADD.W rd,rs1,rs2	AMOADD.(D)Q rd,rs1,rs2
Logical	AND	R	AMOAND.W rd,rs1,rs2	AMOAND.(D)Q rd,rs1,rs2
	OR	R	AMOOR.W rd,rs1,rs2	AMOOR.(D)Q rd,rs1,rs2
Min/Max	MINimum	R	AMOIN.W rd,rs1,rs2	AMOIN.(D)Q rd,rs1,rs2
	MAXimum	R	AMOMAX.W rd,rs1,rs2	AMOMAX.(D)Q rd,rs1,rs2
	MINimum Unsigned	R	AMOMINU.W rd,rs1,rs2	AMOMINU.(D)Q rd,rs1,rs2
	MAXimum Unsigned	R	AMOMAXU.W rd,rs1,rs2	AMOMAXU.(D)Q rd,rs1,rs2

Optional Floating-Point Instruction Extension: RVF, RVFD, & RVFQ				
Category	Name	Fmt	RV32(FD)Q (FP/SP,DP,FP,FP)	+RV(64,128)
Move	Move from Integer	R	FMV.W.(H)S.X rd,rs1	FMV.(D)Q.X rd,rs
	Move to Integer	R	FMV.X.(H)S rd,rs1	FMV.X.(D)Q rd,rs
Convert	Convert from Int	R	FCVT.W.(H)S.D.Q.W rd,rs1	FCVT.(H)S(D)Q.(L)T rd,rs
	Convert from Int Unsigned	R	FCVT.W.(H)S.D.Q.WU rd,rs1	FCVT.(H)S(D)Q.(L)TU rd,rs
	Convert to Int	R	FCVT.W.(H)S.D.Q rd,rs1	FCVT.(L)T.(H)S(D)Q rd,rs
	Convert to Int Unsigned	R	FCVT.WU.(H)S.D.Q rd,rs1	FCVT.(L)TU.(H)S(D)Q rd,rs
Load	Load	I	FLW rd,rs1,imm	
	Store	S	FSW rs1,rs2,imm	
Arithmetic	ADD	I	FADD.(S)D.Q rd,rs1,rs2	
	SUBtract	I	FSUB.(S)D.Q rd,rs1,rs2	
	Multiply	R	FNMUL.(S)D.Q rd,rs1,rs2	
	DIVide	R	FNDIV.(S)D.Q rd,rs1,rs2	
	Square Root	R	FSQRT.(S)D.Q rd,rs1	
Mul-Add	Multiply-ADD	R	FPMADD.(S)D.Q rd,rs1,rs2,rs	
	Multiply-SUB	R	FPMMSUB.(S)D.Q rd,rs1,rs2,rs	
	Negative Multiply-SUB	R	FPMNSUB.(S)D.Q rd,rs1,rs2,rs	
	Negative Multiply-ADD	R	FPMNADD.(S)D.Q rd,rs1,rs2,rs	
Sign Inject	SIGN source	R	FSIGNJ.(S)D.Q rd,rs1,rs2	
	Negative SIGN source	R	FSIGNNJ.(S)D.Q rd,rs1,rs2	
	Xor SIGN source	R	FSIGNX.(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	FBEQ.(S)D.Q rd,rs1,rs2	
	Compare Float <	R	FBLT.(S)D.Q rd,rs1,rs2	
	Compare Float >	R	FBLE.(S)D.Q rd,rs1,rs2	
Categorization	Classify Type	R	FCLASS.(S)D.Q rd,rs1	
Configuration	Read Status	R	FRCSR rd	
	Read Rounding Mode	R	FRSRM rd	
	Read Flags	R	FRFLAGS rd	
	Swap Status Reg	R	FRSCSR rd,rs1	
	Swap Rounding Mode	R	FRSRM rd,rs1	
Swap	Swap Flags	R	FRSFLAGS rd,rs1	
	Swap Rounding Mode Imm	I	FRSRMI rd,imm	

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 RV1 base of <50 classic integer RISC instructions is required. Every 16-bit RVC instruction matches an existing 32-bit RV1 instruction. See risc.org.

<

Encoding of the instructions, main groups

- **Reserved** opcodes for standard extensions
- Rest of opcodes free for **custom** implementations
- Standard extensions will be frozen/not change in the future

inst[4:2]	000	001	010	011	100	101	110	111
inst[6:5]								(> 32b)
00	LOAD	LOAD-FP	<i>custom-0</i>	MISC-MEM	OP-IMM	AUIPC	OP-IMM-32	48b
01	STORE	STORE-FP	<i>custom-1</i>	AMO	OP	LUI	OP-32	64b
10	MADD	MSUB	NMSUB	NMADD	OP-FP	<i>reserved</i>	<i>custom-2/rv128</i>	48b
11	BRANCH	JALR	<i>reserved</i>	JAL	SYSTEM	<i>reserved</i>	<i>custom-3/rv128</i>	≥ 80b