

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	CSRRW	rd,csr,rs1
	Load Halfword	I	LH	rd,rs1,imm			Atomic Read & Set Bit	CSRRS	rd,csr,rs1
	Load Word	I	LW	rd,rs1,imm	L{D Q}		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 D}U		Atomic Read & Set Bit Imm	CSRRSI	rd,csr,imm
Stores	Store Byte	S	SB	rs1,rs2,imm		Change Level	Env. Call	ECALL	
	Store Halfword	S	SH	rs1,rs2,imm			Environment Breakpoint	EBREAK	
	Store Word	S	SW	rs1,rs2,imm	S{D Q}		Environment Return	ERET	
Shifts	Shift Left	R	SLL	rd,rs1,rs2	SLL{W D}	Trap Redirect	to Supervisor	MRTS	
	Shift Left Immediate	I	SLLI	rd,rs1,shamt	SLLI{W D}		Redirect Trap to Hypervisor	MRTH	
	Shift Right	R	SRL	rd,rs1,rs2	SRL{W D}		Hypervisor Trap to Supervisor	HRTS	
	Shift Right Immediate	I	SRLI	rd,rs1,shamt	SRLI{W D}	Interrupt	Wait for Interrupt	WFI	
	Shift Right Arithmetic	R	SRA	rd,rs1,rs2	SRA{W D}		Supervisor FENCE	SFENCE.VM	rs1
Arithmetic	ADD	R	ADD	rd,rs1,rs2	ADD{W D}	Optional Compressed (16-bit) Instruction Extension: RVC			
	ADD Immediate	I	ADDI	rd,rs1,imm	ADDI{W D}				
	SUBtract	R	SUB	rd,rs1,rs2	SUB{W D}	Category Name Fmt RVC RVI equivalent			
	Load Upper Imm	U	LUI	rd,imm					
	Add Upper Imm to PC	U	AUIPC	rd,imm		Loads	Load Word	CL	C.LW rd',rs1',imm LW rd',rs1',imm*4
Logical	XOR	R	XOR	rd,rs1,rs2			Load Word SP	CI	C.LWSP rd,imm LW rd,sp,imm*4
	XOR Immediate	I	XORI	rd,rs1,imm			Load Double	CL	C.LD rd',rs1',imm LD rd',rs1',imm*8
	OR	R	OR	rd,rs1,rs2			Load Double SP	CI	C.LDSP rd,imm LD rd,sp,imm*8
	OR Immediate	I	ORI	rd,rs1,imm			Load Quad	CL	C.LQ rd',rs1',imm LQ rd',rs1',imm*16
Compare	Set <	R	SLT	rd,rs1,rs2			Load Quad SP	CI	C.LQSP rd,imm LQ rd,sp,imm*16
	Set < Immediate	I	SLTI	rd,rs1,imm		Stores	Store Word	CS	C.SW rs1',rs2',imm SW rs1',rs2',imm*4
	Set < Unsigned	R	SLTU	rd,rs1,rs2			Store Word SP	CSS	C.SWSP rs2,imm SW rs2,sp,imm*4
	Set < Imm Unsigned	I	SLTIU	rd,rs1,imm			Store Double	CS	C.SD rs1',rs2',imm SD rs1',rs2',imm*8
							Store Double SP	CSS	C.SDSP rs2,imm SD rs2,sp,imm*8
Branches	Branch =	SB	BEQ	rs1,rs2,imm			Store Quad	CS	C.SQ rs1',rs2',imm SQ rs1',rs2',imm*16
	Branch ≠	SB	BNE	rs1,rs2,imm			Store Quad SP	CSS	C.SQSP rs2,imm SQ rs2,sp,imm*16
	Branch <	SB	BLT	rs1,rs2,imm		Arithmetic	ADD	CR	C.ADD rd,rs1 ADD rd,rd,rs1
	Branch ≥	SB	BGE	rs1,rs2,imm			ADD Word	CR	C.ADDW rd,rs1 ADDW rd,rd,imm
	Branch < Unsigned	SB	BLTU	rs1,rs2,imm			ADD Immediate	CI	C.ADDI rd,imm ADDI rd,rd,imm
Jump & Link	Jump & Link	J&L	JAL	rd,imm			ADD Word Imm	CI	C.ADDIW rd,imm ADDIW rd,rd,imm
	Jump & Link Register	UJ	JALR	rd,rs1,imm			ADD SP Imm * 16	CI	C.ADDI16SP x0,imm ADDI sp,sp,imm*16
Synch	Synch thread	I	FENCE				ADD SP Imm * 4	CIW	C.ADDI4SPN rd',imm ADDI rd',sp,imm*4
	Synch Instr & Data	I	FENCE.I				Load Immediate	CI	C.LI rd,imm ADDI rd,x0,imm
System	System CALL	I	SCALL			Shifts	Load Upper Imm	CI	C.LUI rd,imm LUI rd,imm
	System BREAK	I	SBREAK				MoVe	CR	C.MV rd,rs1 ADD rd,rs1,x0
Counters	ReaD CYCLE	I	RDCYCLE	rd			SUB	CR	C.SUB rd,rs1 SUB rd,rd,rs1
	ReaD CYCLE upper Half	I	RDCYCLEH	rd			Shift Left Imm	CI	C.SLLI rd,imm SLLI rd,rd,imm
	ReaD TIME	I	RDTIME	rd		Branches	Branch=0	CB	C.BEQZ rs1',imm BEQ rs1',x0,imm
	ReaD TIME upper Half	I	RDTIMEH	rd			Branch≠0	CB	C.BNEZ rs1',imm BNE rs1',x0,imm
	ReaD INSTR RETired	I	RDINSTRET	rd		Jump	Jump	CJ	C.J imm JAL x0,imm
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
							Jump & Link Register	CR	C.JALR rs1 JALR ra,rs1,0
System	Env. BREAK	CI	C.EBREAK			System Env. BREAK			
						CI C.EBREAK EBREAK			

32-bit Instruction Formats

	31	30	25	24	21	20	19	15	14	12	11	8	7	6	0			
R	funct7				rs2		rs1	funct3				rd		opcode				
I	imm[11:0]						rs1	funct3				rd		opcode				
S	imm[11:5]				rs2		rs1	funct3		imm[4:0]		opcode						
SB	imm[12]	imm[10:5]				rs2		rs1	funct3		imm[4:1]	imm[11]	opcode					
U	imm[31:12]												rd				opcode	
UJ	imm[20]	imm[10:1]				imm[11]	imm[19:12]						rd		opcode			

16-bit (RVC) Instruction Formats

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR	funct4				rd/rs1				rs2				op			
CI	funct3		imm		rd/rs1				imm				op			
CSS	funct3		imm				rs2				op					
CIW	funct3		imm						rd'				op			
CL	funct3		imm		rs1'		imm		rd'		op					
CS	funct3		imm		rs1'		imm		rs2'		op					
CB	funct3		offset		rs1'		offset				op					
CJ	funct3		jump target										op			

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

Optional Multiply-Divide Instruction Extension: RVM								
Category	Name	Fmt	RV32M (Multiply-Divide)		+RV{64,128}			
Multiply	MULTiply	R	MUL	rd,rs1,rs2	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	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		
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	Store Conditional	R	SC.W	rd,rs1,rs2	SC.{D Q}	rd,rs1,rs2		
Swap	SWAP	R	AMOSWAP.W	rd,rs1,rs2	AMOSWAP.{D Q}	rd,rs1,rs2		
Add	ADD	R	AMOADD.W	rd,rs1,rs2	AMOADD.{D Q}	rd,rs1,rs2		
Logical	XOR	R	AMOXOR.W	rd,rs1,rs2	AMOXOR.{D Q}	rd,rs1,rs2		
	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	AMOMIN.W	rd,rs1,rs2	AMOMIN.{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		
Three Optional Floating-Point Instruction Extensions: RVF, RVD, & RVQ								
Category	Name	Fmt	RV32{F D Q} (HP/SP,DP,QP FI Pt)		+RV{64,128}			
Move	Move from Integer	R	FMV.{H S}.X	rd,rs1	FMV.{D Q}.X	rd,rs1		
	Move to Integer	R	FMV.X.{H S}	rd,rs1	FMV.X.{D Q}	rd,rs1		
Convert	Convert from Int	R	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		
Load	Load	I	FL{W,D,Q}	rd,rs1,imm	RISC-V Calling Convention			
Store	Store	S	FS{W,D,Q}	rs1,rs2,imm	Register	ABI Name	Saver	Description
Arithmetic	ADD	R	FADD.{S D Q}	rd,rs1,rs2	x0	zero	---	Hard-wired zero
	SUBtract	R	FSUB.{S D Q}	rd,rs1,rs2	x1	ra	Caller	Return address
	MULTiply	R	FMUL.{S D Q}	rd,rs1,rs2	x2	sp	Callee	Stack pointer
	DIVide	R	FDIV.{S D Q}	rd,rs1,rs2	x3	gp	---	Global pointer
	SQuare RooT	R	FSQRT.{S D Q}	rd,rs1	x4	tp	---	Thread pointer
Mul-Add	MULTiply-ADD	R	FMADD.{S D Q}	rd,rs1,rs2,rs3	x5-7	t0-2	Caller	Temporaries
	MULTiply-SUBtract	R	FMSUB.{S D Q}	rd,rs1,rs2,rs3	x8	s0/fp	Callee	Saved register/frame pointer
	Negative MULTiply-SUBtract	R	FNMSUB.{S D Q}	rd,rs1,rs2,rs3	x9	s1	Callee	Saved register
	Negative MULTiply-ADD	R	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	Caller	Function arguments
	Negative SIGN source	R	FSGNJN.{S D Q}	rd,rs1,rs2	x18-27	s2-11	Callee	Saved registers
	Xor SIGN source	R	FSGNJX.{S D Q}	rd,rs1,rs2	x28-31	t3-t6	Caller	Temporaries
Min/Max	MINimum	R	FMIN.{S D Q}	rd,rs1,rs2	f0-7	ft0-7	Caller	FP temporaries
	MAXimum	R	FMAX.{S D Q}	rd,rs1,rs2	f8-9	fs0-1	Callee	FP saved registers
Compare	Compare Float =	R	FEQ.{S D Q}	rd,rs1,rs2	f10-11	fa0-1	Caller	FP arguments/return values
	Compare Float <	R	FLT.{S D Q}	rd,rs1,rs2	f12-17	fa2-7	Caller	FP arguments
	Compare Float ≤	R	FLE.{S D Q}	rd,rs1,rs2	f18-27	fs2-11	Callee	FP saved registers
Categorization	Classify Type	R	FCLASS.{S D Q}	rd,rs1	f28-31	ft8-11	Caller	FP temporaries
Configuration	Read Status	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				

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-, double-, 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 riscv.org. (8/21/15 revision)