



RV32I BASE INTEGER INSTRUCTIONS, in alphabetical order

MNEMONIC	FMT	NAME	DESCRIPTION (in Verilog)
add	R	ADD	$R[rd] = R[rs1] + R[rs2]$
addi	I	ADD Immediate	$R[rd] = R[rs1] + \text{imm}$
and	R	AND	$R[rd] = R[rs1] \& R[rs2]$
andi	I	AND Immediate	$R[rd] = R[rs1] \& \text{imm}$
auipc	U	Add Upper Imm to PC	$R[rd] = PC + \{\text{imm}, 12'b0\}$
beq	SB	Branch Equal	if($R[rs1] = R[rs2]$) $PC = PC + \{\text{imm}, 1'b0\}$
bge	SB	Branch Greater or Equal	if($R[rs1] \geq R[rs2]$) $PC = PC + \{\text{imm}, 1'b0\}$
bgeu	SB	Branch Greater or Equal if Unsigned	if($R[rs1] \geq R[rs2]$) $PC = PC + \{\text{imm}, 1'b0\}$ 2)
blt	SB	Branch Less Than	if($R[rs1] < R[rs2]$) $PC = PC + \{\text{imm}, 1'b0\}$
bltu	SB	Branch Less Than Unsigned	if($R[rs1] < R[rs2]$) $PC = PC + \{\text{imm}, 1'b0\}$ 2)
bne	SB	Branch Not Equal	if($R[rs1] \neq R[rs2]$) $PC = PC + \{\text{imm}, 1'b0\}$
csrrc	I	Cont./Stat.RegRead& Clear	$R[rd] = CSR$; $CSR = CSR \& !R[rs1]$
csrrci	I	Cont./Stat.RegRead& Clear Imm	$R[rd] = CSR$; $CSR = CSR \& \text{imm}$
csrrs	I	Cont./Stat.RegRead& Set	$R[rd] = CSR$; $CSR = CSR R[rs1]$
csrrsi	I	Cont./Stat.RegRead& Set Imm	$R[rd] = CSR$; $CSR = CSR \text{imm}$
csrrw	I	Cont./Stat.RegRead& Write	$R[rd] = CSR$; $CSR = R[rs1]$
csrrwi	I	Cont./Stat.RegRead& Write Imm	$R[rd] = CSR$; $CSR = \text{imm}$
ebreak	I	Environment BREAK	Transfer control to debugger
ecall	I	Environment CALL	Transfer control to environment sys
fence	I	Synch thread	Synchronizes threads
fence.i	I	Sync Instr & Data	Synchronizes writes to instr stream
jal	UJ	Jump & Link	$R[rd] = PC + 4$; $PC = PC + \{\text{imm}, 1'b0\}$
jalr	I	Jump & Link Register	$R[rd] = PC + 4$; $PC = (R[rs1] + \text{imm}) \& (!1)$ 3)
lb	I	Load Byte	$R[rd] = \{24'bM[]\}(7), M[R[rs1] + \text{imm}](7:0)$ 4)
lbu	I	Load Byte Unsigned	$R[rd] = \{24'b0, M[R[rs1] + \text{imm}](7:0)\}$
lh	I	Load Halfword	$R[rd] = \{16'bM[]\}(15), M[R[rs1] + \text{imm}](15:0)$ 4)
lhu	I	Load Halfword Unsigned	$R[rd] = \{16'b0, M[R[rs1] + \text{imm}](15:0)\}$
lui	U	Load Upper Immediate	$R[rd] = \{\text{imm}, 12'b0\}$
lw	I	Load Word	$R[rd] = M[R[rs1] + \text{imm}]$ 4)
or	R	OR	$R[rd] = R[rs1] R[rs2]$
ori	I	OR Immediate	$R[rd] = R[rs1] \text{imm}$
sb	S	Store Byte	$M[R[rs1] + \text{imm}](7:0) = R[rs2](7:0)$
sh	S	Store Halfword	$M[R[rs1] + \text{imm}](15:0) = R[rs2](15:0)$
sll	R	Shift Left	$R[rd] = R[rs1] \ll R[rs2]$
slli	I	Shift Left Immediate	$R[rd] = R[rs1] \ll \text{imm}$
slt	R	Set Less Than	$R[rd] = (R[rs1] < R[rs2]) ? 1:0$
slti	I	Set Less Than Imm	$R[rd] = (R[rs1] < \text{imm}) ? 1:0$
sltu	R	Set Less Than Unsigned	$R[rd] = (R[rs1] < R[rs2]) ? 1:0$ 2)
sltiu	I	Set Less Than Imm Unsig	$R[rd] = (R[rs1] < \text{imm}) ? 1:0$ 2)
sra	R	Shift Right Arithmetic	$R[rd] = R[rs1] \ggg R[rs2]$ 5)
srai	I	Shift Right Arith Imm	$R[rd] = R[rs1] \ggg \text{imm}$ 5)
srl	R	Shift Right	$R[rd] = R[rs1] \gg R[rs2]$
srli	I	Shift Right Immediate	$R[rd] = R[rs1] \gg \text{imm}$
sub	R	SUBtract	$R[rd] = R[rs1] - R[rs2]$
sw	S	Store Word	$M[R[rs1] + \text{imm}] = R[rs2]$
xor	R	XOR	$R[rd] = R[rs1] \wedge R[rs2]$
xori	I	XOR Immediate	$R[rd] = R[rs1] \wedge \text{imm}$
uret	R	User RETURN	$PC = CSR[UEPC]$ and other settings

- Notes:
- 2) Operation assumes unsigned integers (instead 2's complement)
 - 3) The least significant bit of the branch address in jalr is set to 0
 - 4) (signed) Load instructions extend the sign bit of data
 - 5) Replicates the sign bit to fill in the leftmost bits of the result during right shift
 - 6) Multiply with one operand signed and one unsigned
 - 8) Classify writes a 10-bit mask to show which properties are true (e.g. -inf, -0, +0, +inf, denorm...)
- The immediate field is sign-extended in RISC-V

ARITHMETIC CORE INSTRUCTION SET

RV32M Multiply Extension

MNEMONIC	FMT	NAME	DESCRIPTION (in Verilog)
mul	R	MULTiply	$R[rd] = R[rs1] * R[rs2](31:0)$
mulh	R	MULTiply upper Half	$R[rd] = R[rs1] * R[rs2](63:32)$
mulhsu	R	MULTiply upper Half Sign/Unsig	$R[rd] = R[rs1] * R[rs2](63:32)$ 6)
mulhu	R	MULTiply upper Half Unsigned	$R[rd] = R[rs1] * R[rs2](63:32)$ 2)
div	R	DIVide	$R[rd] = (R[rs1] / R[rs2])$
divu	R	DIVide Unsigned	$R[rd] = (R[rs1] / R[rs2])$ 2)
rem	R	REMAinder	$R[rd] = (R[rs1] \% R[rs2])$
remu	R	REMAinder Unsigned	$R[rd] = (R[rs1] \% R[rs2])$ 2)

RV32F Floating-Point Extensions

MNEMONIC	FMT	NAME	DESCRIPTION (in Verilog)
flw	I	Load	$F[rd] = M[R[rs1] + \text{imm}]$
fsw	S	Store	$M[R[rs1] + \text{imm}] = F[rs2]$
fadd.s	R	ADD	$F[rd] = F[rs1] + F[rs2]$
fsub.s	R	SUBtract	$F[rd] = F[rs1] - F[rs2]$
fmul.s	R	MULTiply	$F[rd] = F[rs1] * F[rs2]$
fdiv.s	R	DIVide	$F[rd] = F[rs1] / F[rs2]$
fsqrt.s	R	SQuare RooT	$F[rd] = \text{sqrt}(F[rs1])$
fmadd.s	R4	Multiply-ADD	$F[rd] = F[rs1] * F[rs2] + F[rs3]$
fmsub.s	R4	Multiply-SUB	$F[rd] = F[rs1] * F[rs2] - F[rs3]$
fmnadd.s	R4	Negative Multiply-ADD	$F[rd] = -(F[rs1] * F[rs2] + F[rs3])$
fmnsb.s	R4	Negative Multiply-SUB	$F[rd] = -(F[rs1] * F[rs2] - F[rs3])$
fsgnj.s	R	SiGN source	$F[rd] = \{F[rs2](31), F[rs1](30:0)\}$
fsgnjn.s	R	Negative SiGN source	$F[rd] = \{!F[rs2](31), F[rs1](30:0)\}$
fsgnjx.s	R	Xor SiGN source	$F[rd] = \{F[rs2](31) \wedge F[rs1](31), F[rs1](30:0)\}$
fmin.s	R	MINimum	$F[rd] = (F[rs1] < F[rs2]) ? F[rs1] : F[rs2]$
fmax.s	R	MAXimum	$F[rd] = (F[rs1] > F[rs2]) ? F[rs1] : F[rs2]$
feq.s	R	Compare Float Equal	$R[rd] = (F[rs1] == F[rs2]) ? 1:0$
flt.s	R	Compare Float Less Than	$R[rd] = (F[rs1] < F[rs2]) ? 1:0$
fle.s	R	Compare Float Less or Equal	$R[rd] = (F[rs1] \leq F[rs2]) ? 1:0$
fclass.s	R	Classify type	$R[rd] = \text{class}(F[rs1])$ 8)
fmv.s.x	R	Move from Integer	$F[rd] = R[rs1]$
fmv.x.s	R	Move to Integer	$R[rd] = F[rs1]$
fcvt.s.w	R	Convert from Integer	$F[rd] = \text{float}(R[rs1])$
fcvt.s.wu	R	Convert from Unsig Integer	$F[rd] = \text{float}(R[rs1])$ 2)
fcvt.w.s	R	Convert to Integer	$R[rd] = \text{integer}(F[rs1])$
fcvt.wu.s	R	Convert to Unsig Integer	$R[rd] = \text{integer}(F[rs1])$ 2)

CORE INSTRUCTION FORMATS

	31	25 24	20 19	15 14	12 11	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 10:5]	rs2	rs1	funct3	imm[4:1 11]	opcode	
U	imm[31:12]				rd	opcode	
UJ	imm[20 10:1 11 19:12]				rd	opcode	

PSEUDO INSTRUCTIONS

MNEMONIC	NAME	DESCRIPTION	USES
beqz	Branch == Zero	If($R[rs1] == 0$) $PC = PC + \{\text{imm}, 1'b0\}$	beq
bnez	Branch != Zero	If($R[rs1] != 0$) $PC = PC + \{\text{imm}, 1'b0\}$	bne
fabs.s	Absolut Value	$F[rd] = (F[rs1] < 0) ? -F[rs1] : F[rs1]$	fsgnx
fmv.s	FP move	$F[rd] = F[rs1]$	fsgnj
fneg.s	FP negate	$F[rd] = -F[rs1]$	fsgnjn
j	Jump	$PC = \{\text{imm}, 1'b0\}$	jal
jr	Jump Register	$PC = R[rs1]$	jalr
la	Load Address	$R[rd] = \text{address}$	auipc
li	Load Immediate	$R[rd] = \text{immediate}$	addi
mv	Move	$R[rd] = R[rs1]$	addi
neg	Negate	$R[rd] = -R[rs1]$	sub
nop	No Operation	$R[\text{zero}] = R[\text{zero}] + \text{zero}$	addi
not	Not	$R[rd] = !R[rs1]$	xori
ret	Return	$PC = R[ra]$	jalr
seqz	Set if == Zero	$R[rd] = (R[rs1] == 0) ? 1:0$	sltiu
snez	Set if != Zero	$R[rd] = (R[rs1] != 0) ? 1:0$	sltu

OPCODES IN NUMERICAL ORDER BY OPCODE

MNEMONIC	FMT	OPCODE	FUNCT3	FUNCT7	RS2	HEX
lb	I	0000011	000			03/0
lh	I	0000011	001			03/1
lw	I	0000011	010			03/2
lbu	I	0000011	100			03/4
lhu	I	0000011	101			03/5
addi	I	0010011	000			13/0
slli	I	0010011	001	0000000		13/1/00
slti	I	0010011	010			13/2
sltiu	I	0010011	011			13/3
xori	I	0010011	100			13/4
srlr	I	0010011	101	0000000		13/5/00
srair	I	0010011	101	0100000		13/5/20
ori	I	0010011	110			13/6
andi	I	0010011	111			13/7
auipc	U	0010111				17
sb	S	0100011	000			23/0
sh	S	0100011	001			23/1
sw	S	0100011	010			23/2
add	R	0110011	000	0000000		33/0/00
sub	R	0110011	000	0100000		33/0/20
sll	R	0110011	001	0000000		33/1/00
slt	R	0110011	010	0000000		33/2/00
sltu	R	0110011	011	0000000		33/3/00
xor	R	0110011	100	0000000		33/4/00
srl	R	0110011	101	0000000		33/5/00
sra	R	0110011	101	0100000		33/5/20
or	R	0110011	110	0000000		33/6/00
and	R	0110011	111	0000000		33/7/00
lui	U	0110111				37
beq	SB	1100011	000			63/0
bne	SB	1100011	001			63/1
blt	SB	1100011	100			63/4
bge	SB	1100011	101			63/5
bltu	SB	1100011	110			63/6
bgeu	SB	1100011	111			63/7
jalr	I	1100111	000			67/0
jal	UJ	1101111				6F
ecall	I	1110011	000	0000000 00000		73/0/000
csrrw	I	1110011	001			73/1
csrrs	I	1110011	010			73/2
csrrc	I	1110011	011			73/3
csrrwi	I	1110011	101			73/5
csrrsi	I	1110011	110			73/6
csrrci	I	1110011	111			73/7
mul	R	0110011	000	0000001		33/0/01
mulh	R	0110011	001	0000001		33/1/01
mulhsu	R	0110011	010	0000001		33/2/01
mulhu	R	0110011	011	0000001		33/3/01
div	R	0110011	100	0000001		33/4/01
divu	R	0110011	101	0000001		33/5/01
rem	R	0110011	110	0000001		33/6/01
remu	R	0110011	111	0000001		33/7/01
fadd.s	R	1010011	rm	0000000		53/rm/00
fclass.s	R	1010011	001	1110000		53/1/E0
fcvt.s.w	R	1010011	rm	1101000 00000		53/rm/D00
fcvt.s.wu	R	1010011	rm	1101000 00001		53/rm/D01
fcvt.w.s	R	1010011	rm	1100000 00000		53/rm/C00
fcvt.wu.s	R	1010011	rm	1100000 00001		53/rm/C01
fdiv.s	R	1010011	rm	0001100		53/rm/0C
feq.s	R	1010011	010	1010000		53/2/50
fle.s	R	1010011	000	1010000		53/0/50
flt.s	R	1010011	001	1010000		53/1/50
flw	I	0000111	010			07/2
fmax.s	R	1010011	001	0010100		53/1/14
fmin.s	R	1010011	000	0010100		53/0/14
fmul.s	R	1010011	rm	0001000		53/rm/08
fmv.w.x	R	1010011	000	1111000 00000		53/0/F00
fmv.x.w	R	1010011	000	1110000 00000		53/0/E00
fsgnj.s	R	1010011	000	0010000		53/0/10
fsgnjn.s	R	1010011	001	0010000		53/1/10
fsgnjx.s	R	1010011	010	0010000		53/2/10
fsqrt.s	R	1010011	rm	0101100 00000		53/rm/580
fsub.s	R	1010011	rm	0000100		53/rm/04
fsw	S	0100111	010			27/2
uret	R	1110011	000	0010000 00000		73/0/200

IEEE 754 FLOATING-POINT STANDARD

$(-1)^S \times (1 + \text{Fraction}) \times 2^{(\text{Exponent}-\text{Bias})}$
where Half-precision Bias=15, Single-Precision Bias=127,
Double-Precision Bias=1023, Quad-Precision Bias=16383

IEEE Half, Single, Double, and Quad-Precision Formats:

S	Exponent	Fraction
15	14:10	9:0
S	Exponent	Fraction
31	30:23	22:0
S	Exponent	Fraction
63	62:52	51:0
S	Exponent	Fraction
127	126:112	111:0

REGISTER NAME, USE, CALLING CONVENTION

REGISTER	NAME	USE	SAVED?
x0	zero	The constant value 0	N.A.
x1	ra	Return Address	No
x2	sp	Stack Pointer	Yes
x3	gp	Global Pointer	--
x4	tp	Thread Pointer	--
x5-x7	t0-t2	Temporaries	No
x8	s0/fp	Saved Register/Frame Pointer	Yes
x9	s1	Saved Register	Yes
x10-x11	a0-a1	Function Arguments/Return Values	No
x12-x17	a2-a7	Function Arguments	No
x18-x27	s2-s11	Saved Registers	Yes
x28-x31	t3-t6	Temporaries	No
f0-f7	ft0-ft7	FP Temporaries	No
f8-f9	fs0-fs1	FP Saved Registers	Yes
f10-f11	fa0-fa1	FP Function Arguments/Return Values	No
f12-f17	fa2-fa7	FP Function Arguments	No
f18-f27	fs2-fs11	Saved Registers	Yes
f28-f31	ft8-ft11	Temporaries	No

FCSR (Float-point Control and Status Register)

31	...	8	7	6	5	4	3	2	1	0
Reserved			Round Mode			NV	DZ	OF	UF	NX

Round Mode(rm)

000	to even
001	to zero
010	to -∞
011	to +∞
100	to max mag
111	N.A. (Rars)

Flags

NV	Invalid Operation
DZ	Divide by Zero
OF	OverFlow
UF	UnderFlow
NX	Inexact

Service	a7	Input	Output
Print Integer	1	a0=integer	Print an Integer on console
Print Float	2	fa0=float	Print a Float on console
Print String	4	a0=address of the string	Print a null-terminated string
Read Integer	5		Return in a0 the integer read from console
Read Float	6		Return in fa0 the float read from console
Read String	8	a0=buffer address, a1=max num characters	Return in a0 address the string read from console
Print Char	11	a0=char (ASCII)	Print a char a0 (ASCII)
Exit	10		Return to operational system
Read Char	12		Return in a0 the ASCII code of a pressed key
Time	30		Return in {a1,a0} the system time
Sleep	32	a0=time(ms)	Sleep for a0 miliseconds
Print Int Hex	34	a0=integer	Print an integer a0 in hexadecimal
Rand	41		Return a random number in a0

Decimal Prefix				Binary Prefix	
mili(m)	10 ⁻³	kilo(k)	10 ³	kibi(ki)	2 ¹⁰
micro(μ)	10 ⁻⁶	Mega(M)	10 ⁶	Mebi(Mi)	2 ²⁰
nano(n)	10 ⁻⁹	Giga(G)	10 ⁹	Gibi(Gi)	2 ³⁰
pico(p)	10 ⁻¹²	Tera(T)	10 ¹²	Tebi(Ti)	2 ⁴⁰
femto(f)	10 ⁻¹⁵	Peta(P)	10 ¹⁵	Pebi(Pi)	2 ⁵⁰
atto(a)	10 ⁻¹⁸	Exa(E)	10 ¹⁸	Exbi(Ei)	2 ⁶⁰
zepto(z)	10 ⁻²¹	Zetta(Z)	10 ²¹	Zebi(Zi)	2 ⁷⁰
yocto(y)	10 ⁻²⁴	Yotta(Y)	10 ²⁴	Yobi(Yi)	2 ⁸⁰