

Category	Instruction	Example	Meaning	Comments
Arithmetic	Add	add x5, x6, x7	$x5 = x6 + x7$	Three register operands
	Subtract	sub x5, x6, x7	$x5 = x6 - x7$	Three register operands
	Add immediate	addi x5, x6, 20	$x5 = x6 + 20$	Used to add constants
	Set if less than	slt x5, x6, x7	$x5 = 1$ if $x5 < x6$ , else 0	Three register operands
	Set if less than, unsigned	sltu x5, x6, x7	$x5 = 1$ if $x5 < x6$ , else 0	Three register operands
	Set if less than, immediate	slti x5, x6, x7	$x5 = 1$ if $x5 < x6$ , else 0	Comparison with immediate
	Set if less than immediate, uns.	sltiu x5, x6, x7	$x5 = 1$ if $x5 < x6$ , else 0	Comparison with immediate
	Multiply	mul x5, x6, x7	$x5 = x6 \times x7$	Lower 64 bits of 128-bit product
	Multiply high	mulh x5, x6, x7	$x5 = (x6 \times x7) \gg 64$	Upper 64 bits of 128-bit signed product
	Multiply high, unsigned	mulhu x5, x6, x7	$x5 = (x6 \times x7) \gg 64$	Upper 64 bits of 128-bit unsigned product
	Multiply high, signed-unsigned	mulhsu x5, x6, x7	$x5 = (x6 \times x7) \gg 64$	Upper 64 bits of 128-bit signed-unsigned product
	Divide	div x5, x6, x7	$x5 = x6 / x7$	Divide signed 64-bit numbers
	Divide unsigned	divu x5, x6, x7	$x5 = x6 / x7$	Divide unsigned 64-bit numbers
	Remainder	rem x5, x6, x7	$x5 = x6 \% x7$	Remainder of signed 64-bit division
	Remainder unsigned	remu x5, x6, x7	$x5 = x6 \% x7$	Remainder of unsigned 64-bit division
Data transfer	Load doubleword	ld x5, 40(x6)	$x5 = \text{Memory}[x6 + 40]$	Doubleword from memory to register
	Store doubleword	sd x5, 40(x6)	$\text{Memory}[x6 + 40] = x5$	Doubleword from register to memory
	Load word	lw x5, 40(x6)	$x5 = \text{Memory}[x6 + 40]$	Word from memory to register
	Load word, unsigned	lwu x5, 40(x6)	$x5 = \text{Memory}[x6 + 40]$	Unsigned word from memory to register
	Store word	sw x5, 40(x6)	$\text{Memory}[x6 + 40] = x5$	Word from register to memory
	Load halfword	lh x5, 40(x6)	$x5 = \text{Memory}[x6 + 40]$	Halfword from memory to register
	Load halfword, unsigned	lhu x5, 40(x6)	$x5 = \text{Memory}[x6 + 40]$	Unsigned halfword from memory to register
	Store halfword	sh x5, 40(x6)	$\text{Memory}[x6 + 40] = x5$	Halfword from register to memory
	Load byte	lb x5, 40(x6)	$x5 = \text{Memory}[x6 + 40]$	Byte from memory to register
	Load byte, unsigned	lbu x5, 40(x6)	$x5 = \text{Memory}[x6 + 40]$	Byte halfword from memory to register
	Store byte	sb x5, 40(x6)	$\text{Memory}[x6 + 40] = x5$	Byte from register to memory
	Load reserved	lr.d x5, (x6)	$x5 = \text{Memory}[x6]$	Load; 1st half of atomic swap
	Store conditional	sc.d x7, x5, (x6)	$\text{Memory}[x6] = x5; x7 = 0/1$	Store; 2nd half of atomic swap
	Load upper immediate	lui x5, 0x12345	$x5 = 0x12345000$	Loads 20-bit constant shifted left 12 bits
	Add upper immediate to PC	auipc x5, 0x12345	$x5 = \text{PC} + 0x12345000$	Used for PC-relative data addressing
Logical	And	and x5, x6, x7	$x5 = x6 \& x7$	Three reg. operands; bit-by-bit AND
	Inclusive or	or x5, x6, x8	$x5 = x6   x8$	Three reg. operands; bit-by-bit OR
	Exclusive or	xor x5, x6, x9	$x5 = x6 \wedge x9$	Three reg. operands; bit-by-bit XOR
	And immediate	andi x5, x6, 20	$x5 = x6 \& 20$	Bit-by-bit AND reg. with constant
	Inclusive or immediate	ori x5, x6, 20	$x5 = x6   20$	Bit-by-bit OR reg. with constant
	Exclusive or immediate	xori x5, x6, 20	$x5 = x6 \wedge 20$	Bit-by-bit XOR reg. with constant
Shift	Shift left logical	sll x5, x6, x7	$x5 = x6 \ll x7$	Shift left by register
	Shift right logical	srl x5, x6, x7	$x5 = x6 \gg x7$	Shift right by register
	Shift right arithmetic	sra x5, x6, x7	$x5 = x6 \gg x7$	Arithmetic shift right by register
	Shift left logical immediate	slli x5, x6, 3	$x5 = x6 \ll 3$	Shift left by immediate
	Shift right logical immediate	srlr x5, x6, 3	$x5 = x6 \gg 3$	Shift right by immediate
	Shift right arithmetic immediate	srair x5, x6, 3	$x5 = x6 \gg 3$	Arithmetic shift right by immediate
Conditional branch	Branch if equal	beq x5, x6, 100	if $(x5 == x6)$ go to PC+100	PC-relative branch if registers equal
	Branch if not equal	bne x5, x6, 100	if $(x5 \neq x6)$ go to PC+100	PC-relative branch if registers not equal
	Branch if less than	blt x5, x6, 100	if $(x5 < x6)$ go to PC+100	PC-relative branch if registers less
	Branch if greater or equal	bge x5, x6, 100	if $(x5 \geq x6)$ go to PC+100	PC-relative branch if registers greater or equal
	Branch if less, unsigned	bltu x5, x6, 100	if $(x5 < x6)$ go to PC+100	PC-relative branch if registers less
	Branch if greater/eq, unsigned	bgeu x5, x6, 100	if $(x5 \geq x6)$ go to PC+100	PC-relative branch if registers greater or equal
Unconditional branch	Jump and link	jal x1, 100	$x1 = \text{PC}+4$ ; go to PC+100	PC-relative procedure call
	Jump and link register	jalr x1, 100(x5)	$x1 = \text{PC}+4$ ; go to $x5+100$	Procedure return; indirect call

Figure 1: RISC-V instructions (taken from "Computer Organization and Design" by David A. Patterson, John L. Hennessy)