

## Core Instruction Formats

31	27	26	25	24	20	19	15	14	12	11	7	6	0		
funct7				rs2		rs1		funct3		rd		opcode		R-type	e.g: add rd, rs1, rs2
imm[11:0]						rs1		funct3		rd		opcode		I-type	e.g: addi rd, rs1, imm / lw rd, imm(rs1)
imm[11:5]				rs2		rs1		funct3		imm[4:0]		opcode		S-type	e.g: sw rs2, imm(rs1)
imm[12 10:5]				rs2		rs1		funct3		imm[4:1 11]		opcode		B-type	e.g: beq rs1, rs2, imm
imm[31:12]										rd		opcode		U-type	e.g: lui rd, imm
imm[20 10:1 11 19:12]										rd		opcode		J-type	e.g: jal rd, imm

## RV32I Base Integer Instructions

Inst	Name	FMT	Opcode	funct3	funct7	Description (C)	Note
add	ADD	R	0110011	0x0	0x00	rd = rs1 + rs2	msb-extends
sub	SUB	R	0110011	0x0	0x20	rd = rs1 - rs2	
xor	XOR	R	0110011	0x4	0x00	rd = rs1 ^ rs2	
or	OR	R	0110011	0x6	0x00	rd = rs1   rs2	
and	AND	R	0110011	0x7	0x00	rd = rs1 & rs2	
sll	Shift Left Logical	R	0110011	0x1	0x00	rd = rs1 << rs2	
srl	Shift Right Logical	R	0110011	0x5	0x00	rd = rs1 >> rs2	
sra	Shift Right Arith*	R	0110011	0x5	0x20	rd = rs1 >> rs2	
slt	Set Less Than	R	0110011	0x2	0x00	rd = (rs1 < rs2)?1:0	
addi	ADD Immediate	I	0010011	0x0	imm[5:11]=0x00 imm[5:11]=0x00 imm[5:11]=0x20	rd = rs1 + imm	msb-extends
xori	XOR Immediate	I	0010011	0x4		rd = rs1 ^ imm	
ori	OR Immediate	I	0010011	0x6		rd = rs1   imm	
andi	AND Immediate	I	0010011	0x7		rd = rs1 & imm	
slli	Shift Left Logical Imm	I	0010011	0x1		rd = rs1 << imm[0:4]	
srl	Shift Right Logical Imm	I	0010011	0x5		rd = rs1 >> imm[0:4]	
srai	Shift Right Arith Imm	I	0010011	0x5		rd = rs1 >> imm[0:4]	
slti	Set Less Than Imm	I	0010011	0x2		rd = (rs1 < imm)?1:0	
lb	Load Byte	I	0000011	0x0		rd = M[rs1+imm][0:7]	zero-extends
lw	Load Word	I	0000011	0x2		rd = M[rs1+imm][0:31]	
lbu	Load Byte (U)	I	0000011	0x4		rd = M[rs1+imm][0:7]	
sb	Store Byte	S	0100011	0x0		M[rs1+imm][0:7] = rs2[0:7]	
sw	Store Word	S	0100011	0x2		M[rs1+imm][0:31] = rs2[0:31]	
beq	Branch ==	B	1100011	0x0		if(rs1 == rs2) PC += imm	zero-extends zero-extends
bne	Branch !=	B	1100011	0x1		if(rs1 != rs2) PC += imm	
blt	Branch <	B	1100011	0x4		if(rs1 < rs2) PC += imm	
bge	Branch ≥	B	1100011	0x5		if(rs1 ≥ rs2) PC += imm	
bltu	Branch < (U)	B	1100011	0x6		if(rs1 < rs2) PC += imm	
bgeu	Branch ≥ (U)	B	1100011	0x7		if(rs1 ≥ rs2) PC += imm	
jal	Jump And Link	J	1101111	0x0		rd = PC+4; PC += imm	
jalr	Jump And Link Reg	I	1100111			rd = PC+4; PC = rs1 + imm	

## Pseudo Instructions

Pseudoinstruction	Base Instruction(s)	Meaning
nop	addi x0, x0, 0	No operation
li rd, immediate	<i>Myriad sequences</i>	Load 32-bit immediate
not rd, rs	xori rd, rs, -1	One's complement
j offset	jal x0, offset	Unconditional Jump
jal offset	jal x1, offset	Jump and link
jr rs	jalr x0, rs, 0	Jump register
jalr rs	jalr x1, rs, 0	Jump and link register

## Registers

Register	ABI Name	Description	Saver
x0	zero	Zero constant	—
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	—
x4	tp	Thread pointer	—
x5-x7	t0-t2	Temporaries	Caller
x8	s0 / fp	Saved / frame pointer	Callee
x9	s1	Saved register	Callee
x10-x11	a0-a1	Fn args/return values	Caller
x12-x17	a2-a7	Fn args	Caller
x18-x27	s2-s11	Saved registers	Callee
x28-x31	t3-t6	Temporaries	Caller