Base format: Inst rd, rs1, rs2 (rd: destination register, rs1, rs2 source registers).
Other formats: Inst rs1, rs2
               Inst rs1, immediate

RV32I Base Integer Instructions

Inst	Name	Description (C)
add	ADD	rd = rs1 + rs2
sub	SUB	rd = rs1 - rs2
xor	XOR	rd = rs1 ^ rs2
or	OR	rd = rs1 | rs2
and	AND	rd = rs1 & rs2
sll	Shift Left Logical	rd = rs1 << rs2
srl	Shift Right Logical	rd = rs1 >> rs2
sra	Shift Right Arith*	rd = rs1 >> rs2
slt	Set Less Than	rd = (rs1 < rs2)?1:0
sltu	Set Less Than (U)	rd = (rs1 < rs2)?1:0
addi	ADD Immediate	rd = rs1 + imm
xori	XOR Immediate	rd = rs1 ^ imm
ori	OR Immediate	rd = rs1 | imm
andi	AND Immediate	rd = rs1 & imm
slli	Shift Left Logical Imm	rd = rs1 << imm[0:4]
srli	Shift Right Logical Imm	rd = rs1 >> imm[0:4]
srai	Shift Right Arith Imm	rd = rs1 >> imm[0:4]
slti	Set Less Than Imm	rd = (rs1 < imm)?1:0
sltiu	Set Less Than Imm (U)	rd = (rs1 < imm)?1:0
lb	Load Byte	rd = M[rs1+imm][0:7]
lh	Load Half	rd = M[rs1+imm][0:15]
lw	Load Word	rd = M[rs1+imm][0:31]
lbu	Load Byte (U)	rd = M[rs1+imm][0:7]
lhu	Load Half (U)	rd = M[rs1+imm][0:15]
sb	Store Byte	M[rs1+imm][0:7] = rs2[0:7]
sh	Store Half	M[rs1+imm][0:15] = rs2[0:15]
sw	Store Word	M[rs1+imm][0:31] = rs2[0:31]
beq	Branch ==	if(rs1 == rs2) PC += imm
bne	Branch !=	if(rs1 != rs2) PC += imm
blt	Branch <	if(rs1 < rs2) PC += imm
bge	Branch ≤	if(rs1 >= rs2) PC += imm
bltu	Branch < (U)	if(rs1 < rs2) PC += imm
bgeu	Branch ≥ (U)	if(rs1 >= rs2) PC += imm
jal	Jump And Link	rd = PC+4; PC += imm
jalr	Jump And Link Reg	rd = PC+4; PC = rs1 + imm
lui	Load Upper Imm	rd = imm << 12
auipc	Add Upper Imm to PC	rd = PC + (imm << 12)
ecall	Environment Call	Transfer control to OS
ebreak	Environment Break	Transfer control to debugger

Registers

Register	ABI Name	Description	Saver
x0	zero	Zero constant	—
x1	ra	Return address	Caller
x2	sp	Stack pointer	—
x3	gp	Global pointer	—
x4	tp	Thread pointer	Callee
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

RV32M Multiply Extension

Inst	Name	Description (C)
mul	MUL	rd = (rs1 * rs2)[31:0]
mulh	MUL High	rd = (rs1 * rs2)[63:32]
mulsu	MUL High (S) (U)	rd = (rs1 * rs2)[63:32]
mulu	MUL High (U)	rd = (rs1 * rs2)[63:32]
div	DIV	rd = rs1 / rs2
divu	DIV (U)	rd = rs1 / rs2
rem	Remainder	rd = rs1 % rs2
remu	Remainder (U)	rd = rs1 % rs2

Pseudo Instructions

Pseudoinstruction	Base Instruction(s)	Meaning
la rd, symbol	auipc rd, symbol[31:12] addi rd, rd, symbol[11:0]	Load address
l{b|h|w|d} rd, symbol	auipc rd, symbol[31:12] l{b|h|w|d} rd, symbol[11:0](rd)	Load global
s{b|h|w|d} rd, symbol, rt	auipc rt, symbol[31:12] s{b|h|w|d} rd, symbol[11:0](rt)	Store global
fl{w|d} rd, symbol, rt	auipc rt, symbol[31:12] fl{w|d} rd, symbol[11:0](rt)	Floating-point load global
fs{w|d} rd, symbol, rt	auipc rt, symbol[31:12] fs{w|d} rd, symbol[11:0](rt)	Floating-point store global
nop	addi x0, x0, 0	No operation
li rd, immediate	*Myriad sequences*	Load immediate
mv rd, rs	addi rd, rs, 0	Copy register
not rd, rs	xori rd, rs, -1	One's complement
neg rd, rs	sub rd, x0, rs	Two's complement
negw rd, rs	subw rd, x0, rs	Two's complement word
sext.w rd, rs	addiw rd, rs, 0	Sign extend word
seqz rd, rs	sltiu rd, rs, 1	Set if $=$ zero
snez rd, rs	sltu rd, x0, rs	Set if $\neq$ zero
sltz rd, rs	slt rd, rs, x0	Set if $<$ zero
sgtz rd, rs	slt rd, x0, rs	Set if $>$ zero
fmv.s rd, rs	fsgnj.s rd, rs, rs	Copy single-precision register
fabs.s rd, rs	fsgnjx.s rd, rs, rs	Single-precision absolute value
fneg.s rd, rs	fsgnjn.s rd, rs, rs	Single-precision negate
fmv.d rd, rs	fsgnj.d rd, rs, rs	Copy double-precision register
fabs.d rd, rs	fsgnjx.d rd, rs, rs	Double-precision absolute value
fneg.d rd, rs	fsgnjn.d rd, rs, rs	Double-precision negate
beqz rs, offset	beq rs, x0, offset	Branch if $=$ zero
bnez rs, offset	bne rs, x0, offset	Branch if $\neq$ zero
blez rs, offset	bge x0, rs, offset	Branch if $\leq$ zero
bgez rs, offset	bge rs, x0, offset	Branch if $\geq$ zero
bltz rs, offset	blt rs, x0, offset	Branch if $<$ zero
bgtz rs, offset	blt x0, rs, offset	Branch if $>$ zero
bgt rs, rt, offset	blt rt, rs, offset	Branch if $>$
ble rs, rt, offset	bge rt, rs, offset	Branch if $\leq$
bgtu rs, rt, offset	bltu rt, rs, offset	Branch if $>$, unsigned
bleu rs, rt, offset	bgeu rt, rs, offset	Branch if $\leq$, unsigned
j offset	jal x0, offset	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
ret	jalr x0, x1, 0	Return from subroutine
call offset	auipc x1, offset[31:12] jalr x1, x1, offset[11:0]	Call far-away subroutine
tail offset	auipc x6, offset[31:12] jalr x0, x6, offset[11:0]	Tail call far-away subroutine
fence	fence iorw, iorw	Fence on all memory and I/O