RISC-V RV32I Instruction Set Reference

Instructions:

Inst.	Name	Usage	Meaning
add	add	add rd, rs1, rs2	rd = rs1 + rs2
sub	subtract	sub rd, rs1, rs2	rd = rs1 - rs2
and	and	and rd, rs1, rs2	rd = rs1 & rs2
or	or	or rd, rs1, rs2	rd = rs1 | rs2
xor	xor	xor rd, rs1, rs2	rd = rs1 ^ rs2
slt	set less than	slt rd, rs1, rs2	rd = rs1 < rs2
sltu	set less than (U)	sltu rd, rs1, rs2	rd = rs1 < rs2
sll	shift left logic.	sll rd, rs1, rs2	rd = rs1 << rs2[4:0]
srl	shift right logic.	srl rd, rs1, rs2	rd = rs1 >> rs2[4:0]
sra	shift right arith.	sra rd, rs1, rs2	rd = rs1 >> rs2[4:0]
addi	add immediate	addi rd, rs, const	rd = rs + const
andi	and immediate	andi rd, rs, const	rd = rs & const
ori	or immediate	ori rd, rs, const	rd = rs | const
xori	xor immediate	xori rd, rs, const	rd = rs ^ const
slti	set less than imm.	slti rd, rs, const	rd = rs < const
sltiu	set less than imm. (U)	sltiu rd, rs, const	rd = rs < const
slli	shift left logic. imm.	slli rd, rs, const	rd = rs << const[4:0]
srli	shift right logic. imm.	srli rd, rs, const	rd = rs >> const[4:0]
srai	shift right arith. imm.	srai rd, rs, const	rd = rs >> const[4:0]
beq	branch if ==	beq rs1, rs2, offset	if (rs1 == rs2) pc += offset
bne	branch if !=	bne rs1, rs2, offset	if (rs1 != rs2) pc += offset
bltu	branch if < (U)	bltu rs1, rs2, offset	if (rs1 < rs2) pc += offset
blt	branch if <	blt rs1, rs2, offset	if (rs1 < rs2) pc += offset
bgeu	branch if >= (U)	bgeu rs1, rs2, offset	if (rs1 >= rs2) pc += offset
bge	branch if >=	bge rs1, rs2, offset	if (rs1 >= rs2) pc += offset
lui	load upper imm.	lui rd, const	rd = const << 12
auipc	add upper imm. to pc	auipc rd, const	rd = pc + const << 12
jal	jump and link	jal rd, offset	rd = pc + 4; pc += offset
jalr	jump and link reg.	jalr rd, offset(rs)	rd = pc + 4; pc += rs + offset
lw	load word	lw rd, offset(rs)	rd = mem[rs + offset][31:0]
lh	load half	lh rd, offset(rs)	rd = mem[rs + offset][15:0]
lhu	load half (U)	lhu rd, offset(rs)	rd = mem[rs + offset][15:0]
lb	load byte	lb rd, offset(rs)	rd = mem[rs + offset][7:0]
lbu	load byte (U)	lbu rd, offset(rs)	rd = mem[rs + offset][7:0]
sw	store word	sw rs2, offset(rs1)	mem[rs1 + offset][31:0] = rs2
sh	store half	sh rs2, offset(rs1)	mem[rs1 + offset][15:0] = rs2[15:0]
sb	store byte	sb rs2, offset(rs1)	mem[rs1 + offset][7:0] = rs2[7:0]

Pseudo-Instructions:

Inst.	Name	Usage	Meaning or Implementation
nop	no operation	nop	addi x0, x0, 0
li	load immediate	li rd, constant	rd = constant
mv	move	mv rd, rs	addi rd, rs, 0
not	not	not rd, rs	xori rd, rs, -1
neg	negate	neg rd, rs	sub rd, x0, rs
seqz	set equal to zero	seqz rd, rs	sltui rd, rs, 1
snez	set not equal to zero	snez rd, rs	sltu rd, x0, rs
sltz	set less than zero	sltz rd, rs	slt rd, rs, x0
sgtz	set greater than zero	sgtz rd, rs	slt rd, x0, rs
beqz	branch if == 0	beqz rs, offset	beq rs, x0, offset
bnez	branch if != 0	bnez rs, offset	bne rs, x0, offset
blez	branch if <= 0	blez rs, offset	bge x0, rs, offset
bgez	branch if >= 0	bgez rs, offset	bge rs, 0, offset
bltz	branch if < 0	bltz rs, offset	blt rs, x0, offset
bgtz	branch if > 0	bgtz rs, offset	blt x0, rs, offset
bgt	branch if >	bgt rs1, rs2, offset	blt rs2, rs1, offset
bgtu	branch if > (U)	bgtu rs1, rs2, offset	bltu rs2, rs1, offset
ble	branch if <=	ble rs1, rs2, offset	bge rs2, rs1, offset
bleu	branch if <= (U)	bleu rs1, rs2, offset	bgeu rs2, rs1, offset
j	jump	j offset	jal x0, offset
jr	jump register	jr rs	jalr x0, 0(rs)
call	call subroutine	call offset	jump & save return add. in x1
ret	return from subroutine	ret	jalr x0, 0(x1)

Registers:

Reg.	Name	Purpose	Reg.	Name	Purpose
x0	zero	zero	x8	s0/fp	saved/frame pointer
x1	ra	return address	x9	s1	saved
x2	sp	stack pointer	x10-x11	a0-a1	args/return values
x3	gp	global pointer	x12-x17	a2-a7	args
x4	tp	thread pointer	x17-x27	s2-s11	saved
x5-x7	t0-t2	temporary	x28-x31	t3-t6	temporary