RISC-V RV32IM ISA Reference Sheet v1.2

31 25	24 20	19 15	14 12	11 7	6 0	
funct7	rs2	rs1	funct3	rd	opcode	R-type
imm[11	:0]	rs1	funct3	rd	opcode	I-type
imm[11:5]	rs2	rs1	funct3	imm[4:0]	opcode	S-type
imm[12,10:5]	rs2	rs1	funct3	imm[4:1,11]	opcode	B-type
	imm[31	1:12]	rd	opcode	U-type	
in	nm[20,10:1	,11,19:12]	rd	opcode	J-type	

reg	alias	reg	alias
x0	zero	x5-x7	t0-t2
x1	ra	x8, x9	s0/fp, s1
x2	sp	x10-x17	a0-a7
x3	gp	x18-x27	s2-s11
x4	tp	x28-x31	t3-t6

imm[20,10:1,11,19:12]		rc	l	opcode J-type x4 tp x28-x31 t3-t6		
instruction	fmt	opcode	fun3	fun7	semantics	
lui rd,imm20	U	7'd 55			rd = imm20 << 12	
auipc rd,imm20	U	23			rd = pc + (imm20 << 12)	
addi rd,rs1,imm12	I	19	000		rd = rs1 + se(imm12)	
slti rd,rs1,imm12	I	19	010		rd = rs1 <signed 0<="" 1="" :="" ?="" se(imm12)="" td=""></signed>	
sltiu rd,rs1,imm12	I	19	011		rd = rs1 <unsign 0<="" 1="" :="" ?="" se(imm12)="" td=""></unsign>	
xori rd,rs1,imm12	I	19	100		rd = rs1 ^ se(imm12)	
ori rd,rs1,imm12	I	19	110		rd = rs1 se(imm12)	
andi rd,rs1,imm12	I	19	111		rd = rs1 & se(imm12)	
slli rd,rs1,imm12	Ι	19	001	0x0	rd = rs1 << imm12[4:0]	
srli rd,rs1,imm12	I	19	101	0x0	rd = rs1 >> imm12[4:0]	
srai rd,rs1,imm12	Ι	19	101	0x20	rd = rs1 >>> imm12[4:0]	
add rd,rs1,rs2	R	51	000	0x0	rd = rs1 + rs2	
sub rd,rs1,rs2	R	51	000	0x20	rd = rs1 - rs2	
sll rd,rs1,rs2	R	51	001	0x20	rd = rs1 << rs2[4:0]	
slt rd,rs1,rs2	R	51	010	0x0	rd = rs1 << rs2[4.0] rd = rs1 <signed 0<="" 1="" :="" ?="" rs2="" td=""></signed>	
sltu rd,rs1,rs2	R	51	010	0x0	rd = rs1 \signed rs2 ? 1 : 0	
xor rd,rs1,rs2	R	51	100	0x0	rd = rs1 \ rs2 : 1 . 0	
	R	51	100	0x0	$rd = rs1 \rightarrow rs2$ rd = rs1 >> rs2[4:0]	
	R	51	101	0x0	rd = rs1 >> rs2[4:0] rd = rs1 >>> rs2[4:0]	
sra rd,rs1,rs2					- -	
or rd,rs1,rs2	R	51	110	0x0	$rd = rs1 \mid rs2$	
and rd,rs1,rs2	R	51	111	0x0	rd = rs1 & rs2	
lb rd,imm12(rs1)	I	3	000		rd = se(mem[rs1+se(imm12)][7:0])	
lh rd,imm12(rs1)	I	3	001		rd = se(mem[rs1+se(imm12)][15:0])	
lw rd,imm12(rs1)	I	3	010		rd = mem[rs1+se(imm12)][31:0]	
lbu rd,imm12(rs1)	I	3	100		rd = ze(mem[rs1+se(imm12)][7:0])	
lhu rd,imm12(rs1)	I	1	3 101 rd = ze(mem[rs1+se(imm12)][15:0])			
sb rs2,imm12(rs1)	S	35	000		mem[rs1+se(imm12)][7:0] = rs2[7:0]	
sh rs2,imm12(rs1)	S	35	001		mem[rs1+se(imm12)][15:0] = rs2[15:0]	
sw rs2,imm12(rs1)	S	35	010		mem[rs1+se(imm12)][31:0] = rs2	
jal rd,targ20	J	111			rd = pc+4; pc += se(targ20<<1)	
_jalr rd,imm12(rs1)	I	103	000		rd = pc+4; pc = (rs1+se(imm12)) & ~0x1	
beq rs1,rs2,targ12	В	99	000		if (rs1 == rs2) pc += se(targ12<<1)	
bne rs1,rs2,targ12	В	99	001		if (rs1 != rs2) pc += se(targ12<<1)	
blt rs1,rs2,targ12	В	99	100		<pre>if (rs1 <signed +="se(targ12<<1)</pre" pc="" rs2)=""></signed></pre>	
bge rs1,rs2,targ12	В	99	101		if (rs1 ≥signed rs2) pc += se(targ12<<1)	
bltu rs1,rs2,targ12	В	99	110		<pre>if (rs1 <unsign +="se(targ12<<1)</pre" pc="" rs2)=""></unsign></pre>	
bgeu rs1,rs2,targ12	В	99	111		if (rs1 ≥unsign rs2) pc += se(targ12<<1)	
ecall			transfer control to OS			
fence.i		15	001	0x0	order data memory writes and instruction memory fetch	
mul rd,rs1,rs2	R	51	000	0x01	rd = (rs1 * rs2)[31:0]	
mulh rd,rs1,rs2	R	51	001	0x01	rd = (signed(rs1) * signed(rs2))[63:32]	
mulhsu rd,rs1,rs2	R	51	010	0x01	rd = (signed(rs1) * unsign(rs2))[63:32]	
mulhu rd,rs1,rs2	R	51	011	0x01	, , , , , , , , , , , , , , , , , , , ,	
, ,		rd = rs1 /signed rs2				
divu rd,rs1,rs2	R	51	101	0x01	rd = rs1 /signed rs2	
rem rd,rs1,rs2	R	51	110	0x01	rd = rs1 %signed rs2	
remu rd,rs1,rs2	R	51	111	0x01	rd = rs1 %signed rs2	
Tellia Taji 31ji 32	10	31	111	UAUI	14 - 131 Mulistell 132	

Register Map

				0			
reg	alias	description	saver	reg	alias	description	saver
x0	zero	hard-wired zero		t7	t2	temporary	caller
x1	ra	return address	caller	x8	s0/fp	saved reg/frame pointer	callee
x2	sp	stack pointer	callee	x9	s1	saved reg	callee
x3	gp	global pointer		x10-x11	a0-a1	function args/return values	caller
x4	tр	thread pointer		x12-x17	a2-a7	function args	caller
x5	t0	temporary/alt link reg	caller	x18-x27	s2-s11	saved regs	callee
х6	t1	temporary	caller	x28-x31	t3-t6	temporaries	caller

la rd, symbol auipc rd, symbol[31:12] addi rd, rd, symbol[11:0] l{b h w d} rd, symbol auipc rd, symbol[11:0] (rd) s{b h w d} rd, symbol, rt symbol[31:12] s{b h w d} rd, symbol[31:12] s{b h w d} rd, symbol[31:12] s{b h w d} rd, symbol[31:12] store global nop addi x0, x0,0 no operation lui rd, immediate[31:12] addi rd, x0, immediate[11:0] NB: assembler may omit numeessary hui addi 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 set if = zero snez rd,rs subw rd, x0,rs set if = zero snez rd,rs sltiu rd,rs,1 set if < zero snez rd,rs sltiu rd,x0,rs set if < zero sgtz rd,rs sltiu rd,x0,rs set if > zero beqz rs,target beq rs, x0, target branch if ≤ zero beqz rs,target bge x0,rs, target branch if ≤ zero bezz rs,target bge x0,rs, target branch if ≤ zero blez rs, target blt x0,rs, target branch if > zero bltz rs, target blt x0,rs, target branch if > zero bgtz rs, target blt x0,rs, target branch if > zero bgtz rs, target blt x0,rs, target branch if > zero bgtz rs, target blt x0,rs, target branch if > zero bgtz rs, target blt x1,rs, target branch if > zero bgt rs,rt, target blt x1,rs, target branch if > zero bgt rs,rt, target blt x1,rs, target branch if > zero bgt rs,rt, target blt x1,rs, target branch if > zero bgt rs,rt, target blt x1,rs, target branch if > zero bgt rs,rt, target blt x1,rs, target branch if > unsigned bleu rs,rt, target blu rt,rs, target branch if > unsigned bleu rs,rt, target jal x0, target jump jal target jal x1, target jump and link jr rs jalr x0,rs,0 jump register return from subroutine call target jalr x0,x1,0 call target jalr x1,x6, target[31:12] jalr x1,x6,target[11:0] fence fence iorw, jorw fence on all memory and I/O	pseudoinstruction	base instruction(s)	meaning
addir rq,rq,symbol[11:0]	la rd symbol		load address
Tell	14 14,55,11001		load address
s{0 n w d} rd,symbol,rt s{b h w d} rd,symbol[11:0](rt) nop addi x0,x0,0 lui rd,immediate[31:12] addi rd,x0,immediate[31:12] NB: assembler may omit numcessary hit addi mv rd,rs addi rd,rs,0 ropy register nor rd,rs not rd,rs sub rd,x0,rs seq rd,rs sub rd,x0,rs set if = zero snez rd,rs sub rd,x0,rs set if ≠ zero snez rd,rs sltu rd,rs,1 set if > zero sltz rd,rs sltu rd,x0,rs set if > zero sltz rd,rs sltu rd,x0,rs set if > zero beq rs,x0,target beq rs,x0,target branch if ≤ zero blez rs,target bge x0,rs,target bge xs,x0,target branch if ≤ zero bgz rs,target blt rs,x0,target blt rs,x0,target branch if > zero bgtz rs,target blt x0,rs,target branch if > zero bgtz rs,target blt x0,rs,target branch if > zero bgtz rs,target blt rt,rs,target blt rt,rs,target branch if > zero bgt rs,rt,target bgt rs,rt,target bge rt,rs,target branch if > zero bgt rs,rt,target bgt rs,rt,target bgt rs,rt,target bge rt,rs,target branch if > branch if	l{b h w d} rd,symbol	$l\{b h w d\}$ rd, symbol[11:0](rd)	load global
li rd,immediate lui rd,immediate[31:12] addi rd,x0,immediate[11:0] mv rd,rs addi rd,rs,0 copy register not rd,rs not rd,rs sub rd,x0,rs sub rd,x0,rs set if = zero seqz rd,rs subw rd,x0,rs set if = zero sltz rd,rs sltu rd,rs,1 set if < zero set if > zero bet > zero set if > zero bet > zero bet > zero set if > zero bet	s{b h w d} rd,symbol,rt	$s\{b h w d\}$ rd, $symbol[11:0](rt)$	store global
li rd, immediate addi rd, x0, immediate[31:12] addi rd, x0, immediate[11:0] mv rd,rs not rd,rs xori rd,rs,-1 neg rd,rs sub rd, x0,rs subw rd, x0,rs set if = zero snez rd,rs sltiu rd, rs,0 set if ≠ zero sltz rd,rs sltiu rd, rs,1 set if < zero set if > zero beqz rs,target beq rs, x0, target branch if = zero blez rs, target bge x0,rs, target bltz rs, target bltz rs, target bltz rs, target blt x0,rs, target blt x0,rs, target blt x0,rs, target blt rt,rs, target blt rt,rs, target blt rs, rt, target blt rt,rs, target blt rs, rt, target blt rt,rs, target blt rt,rs, target blt rs, rt, target blt rt,rs, target blt rt,rs, target bla rs, rt, target blt rt,rs, target bla rs, rt, target bla rx, rt, target branch if > branch	nop	addi x0,x0,0	
not rd,rs	li rd,immediate		NB: assembler may omit
neg rd,rs sub rd,x0,rs set if = zero snez rd,rs snez rd,rs sltiu rd,rs,0 set if ≠ zero sltz rd,rs sltiu rd,rs,1 set if < zero sgtz rd,rs sltiu rd,x0,rs set if < zero sgtz rd,rs sltiu rd,x0,rs set if < zero sgtz rd,rs sltiu rd,x0,rs set if < zero set if > zero beqz rs,target beq rs,x0,target branch if = zero bnez rs,target bne rs,x0,target branch if ≤ zero blez rs,target bge x0,rs,target branch if ≤ zero blez rs,target bge rs,x0,target branch if < zero blez rs,target blt rs,x0,target branch if < zero bltz rs,target blt x0,rs,target branch if > zero bgtz rs,rt,target blt rt,rs,target branch if > zero bgt rs,rt,target blt rt,rs,target branch if > branch if > ble rs,rt,target blt rt,rs,target branch if ≤ bgtu rs,rt,target bge rt,rs,target branch if ≤ bgtu rs,rt,target bge rt,rs,target branch if ≤ branc	mv rd,rs	addi rd,rs,0	copy register
seqz rd,rs snez rd,rs snez rd,rs sltu rd,rs,0 set if ≠ zero sltz rd,rs sltu rd,rs,1 set if < zero sgtz rd,rs sltu rd,x0,rs set if > zero sgtz rd,rs sltu rd,x0,rs set if > zero beqz rs,target beq rs,x0,target bnez rs,target bge x0,rs,target bge x0,rs,target branch if ≤ zero blez rs,target bge rs,x0,target branch if ≤ zero blez rs,target bge rs,x0,target branch if ≤ zero blez rs,target blt rs,x0,target branch if ≤ zero bltz rs,target blt x0,rs,target branch if < zero bgtz rs,target blt x0,rs,target branch if > zero bgtz rs,rt,target blt rt,rs,target branch if > branch if > ble rs,rt,target blt rt,rs,target branch if > branch if > branch if > ble rs,rt,target blt rt,rs,target branch if > branch	not rd,rs	xori rd,rs,-1	one's complement
snez rd,rs sltz rd,rs sltiu rd,rs,1 set if < zero sgtz rd,rs sltiu rd,x0,rs set if > zero beqz rs,target beq rs,x0,target branch if = zero bnez rs,target bge x0,rs,target bge x0,rs,target bge x0,rs,target bge rs,x0,target branch if ≤ zero bgez rs,target bge rs,x0,target branch if ≤ zero bgez rs,target blt rs,x0,target branch if ≤ zero bgtz rs,target blt x0,rs,target branch if < zero bgtz rs,target blt x0,rs,target branch if > zero bgtz rs,target blt x1,rs,target branch if > branch if > branch if > ble rs,rt,target branch if > branch if > ble rs,rt,target branch if > bran	neg rd,rs	sub rd,x0,rs	two's complement
sltz rd,rs sgtz rd,rs sltu rd,x0,rs set if < zero beqz rs,target beq rs,x0,target branch if = zero bnez rs,target bne rs,x0,target branch if ≠ zero blez rs,target bge x0,rs,target bge x0,rs,target branch if ≤ zero bgez rs,target bge rs,x0,target branch if ≤ zero bgez rs,target blt rs,x0,target branch if < zero bgtz rs,target blt x0,rs,target branch if > zero bgt rs,rt,target blt rt,rs,target branch if > zero bgt rs,rt,target bge rt,rs,target branch if > if	seqz rd,rs	subw rd,x0,rs	
sgtz rd,rs sltu rd,x0,rs beqz rs,target beq rs,x0,target branch if = zero bnez rs,target bne rs,x0,target branch if ≠ zero blez rs,target bge x0,rs,target branch if ≤ zero bgez rs,target bge rs,x0,target branch if ≤ zero bgez rs,target ble rs,x0,target branch if ≤ zero bltz rs,target branch if > zero bgtz rs,target blt x0,rs,target branch if > zero bgt rs,rt,target blt rt,rs,target branch if > branch if > ble rs,rt,target branch if > branch if > ble rs,rt,target branch if > branch if > ble rs,rt,target branch if > nusigned branch if >	snez rd,rs	addiw rd,rs,0	set if \neq zero
beqz rs,target beq rs,x0,target branch if = zero bnez rs,target bne rs,x0,target branch if ≠ zero blez rs,target bge x0,rs,target branch if ≤ zero bgez rs,target bge rs,x0,target branch if ≤ zero bltz rs,target blt rs,x0,target branch if ≤ zero bgtz rs,target blt x0,rs,target branch if > zero bgtz rs,rt,target blt rt,rs,target branch if > bernch if > condition if ≤ begun rs,rt,target branch if ≤ nussigned bleu rs,rt,target branch if ≤ nussigned bleu rs,rt,target bgeun rt,rs,target branch if ≤, unsigned bleu rs,rt,target jal x0,target jump in and link if if it is important in it is	sltz rd,rs	sltiu rd,rs,1	set if < zero
bnez rs,target bne rs,x0,target branch if ≠ zero blez rs,target bge x0,rs,target branch if ≤ zero bgez rs,target bge rs,x0,target branch if ≤ zero bltz rs,target blt rs,x0,target branch if ≤ zero bltz rs,target blt x0,rs,target branch if > zero bgtz rs,target blt x0,rs,target branch if > zero bgt rs,rt,target blt rt,rs,target branch if > ble rs,rt,target bge rt,rs,target branch if > ble rs,rt,target blu rt,rs,target branch if >, unsigned bleu rs,rt,target bgeu rt,rs,target branch if >, unsigned bleu rs,rt,target bgeu rt,rs,target branch if ≤, unsigned j target jal x0,target jump jal target jal x1,target jump and link jr rs jalr x0,rs,0 jump and link register ret jalr x0,x1,0 return from subroutine call target auipc x6,target[31:12] jalr x1,x6,target[11:0] tail call far-away subroutine tail target jalr x0,x6,target[11:0]	sgtz rd,rs	sltu rd,x0,rs	set if > zero
blez rs,target bge x0,rs,target branch if ≤ zero bgez rs,target bge rs,x0,target branch if ≤ zero bltz rs,target blt rs,x0,target branch if < zero bgtz rs,target blt x0,rs,target branch if > zero bgtz rs,target blt x0,rs,target branch if > zero bgt rs,rt,target blt rt,rs,target branch if > beranch if >	beqz rs,target	beq rs,x0,target	branch if $=$ zero
bgez rs,target bge rs,x0,target branch if ≥ zero bltz rs,target blt x0,rs,target branch if > zero bgtz rs,target blt x0,rs,target branch if > zero bgt rs,rt,target blt rt,rs,target branch if > bench if > b	bnez rs,target	bne rs,x0,target	branch if \neq zero
bltz rs,target blt rs,x0,target branch if < zero bgtz rs,target blt x0,rs,target branch if > zero bgt rs,rt,target blt rt,rs,target branch if > branch if > ble rs,rt,target bge rt,rs,target branch if ≤ bgtu rs,rt,target bltu rt,rs,target branch if >, unsigned bleu rs,rt,target bgeu rt,rs,target branch if ≤, unsigned bleu rs,rt,target bgeu rt,rs,target branch if ≤, unsigned j target jal x0,target jump jal target jal x1,target 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 target auipc x6,target[31:12] jalr x1,x6,target[11:0] call far-away subroutine tail target jalr x0,x6,target[11:0]	blez rs,target	bge x0,rs,target	branch if \leq zero
bgtz rs,target blt x0,rs,target branch if > zero bgt rs,rt,target blt rt,rs,target branch if > branch if > ble rs,rt,target bge rt,rs,target branch if ≤ bgtu rs,rt,target bltu rt,rs,target branch if >, unsigned bleu rs,rt,target bgeu rt,rs,target branch if ≤, unsigned bleu rs,rt,target bgeu rt,rs,target branch if ≤, unsigned j target jal x0,target jump jal target jal x1,target 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 target auipc x6,target[31:12] jalr x1,x6,target[11:0] tail target tail call far-away subroutine tail target jalr x0,x6,target[11:0]	bgez rs,target	bge rs,x0,target	branch if \geq zero
bgt rs,rt,target blt rt,rs,target branch if > ble rs,rt,target bge rt,rs,target branch if ≤ bgtu rs,rt,target bltu rt,rs,target branch if >, unsigned bleu rs,rt,target bgeu rt,rs,target branch if ≤, unsigned bleu rs,rt,target bgeu rt,rs,target branch if ≤, unsigned j target jal x0,target jump jal target jal x1,target 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 target auipc x6,target[31:12] jalr x1,x6,target[11:0] tail target tail call far-away subroutine	bltz rs,target	blt rs,x0,target	branch if < zero
ble rs,rt,target bge rt,rs,target branch if ≤ bgtu rs,rt,target bltu rt,rs,target branch if >, unsigned bleu rs,rt,target bgeu rt,rs,target branch if ≤, unsigned j target jal x0,target jump jal target jal x1,target 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 target auipc x6,target[31:12] jalr x1,x6,target[11:0] tail target jalr x0,x6,target[11:0]		blt x0,rs,target	branch if > zero
bgtu rs,rt,target bltu rt,rs,target branch if >, unsigned bleu rs,rt,target bgeu rt,rs,target branch if ≤, unsigned j target jal x0,target jump jal target jal x1,target 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 auipc x6,target[31:12] jalr x1,x6,target[11:0] tail target jalr x0,x6,target[11:0]	bgt rs,rt,target	blt rt,rs,target	branch if >
bleu rs,rt,target bgeu rt,rs,target branch if ≤, unsigned j target jal x0,target jump jal target jal x1,target 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 target auipc x6,target[31:12] jalr x1,x6,target[11:0] tail target jalr x0,x6,target[11:0]	ble rs,rt,target	bge rt,rs,target	branch if \leq
j target jal x0,target jump jal target jal x1,target 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 target auipc x6,target[31:12] jalr x1,x6,target[11:0] tail target jalr x0,x6,target[11:0]	bgtu rs,rt,target	bltu rt,rs,target	
jal targetjal x1,targetjump and linkjr rsjalr x0,rs,0jump registerjalr rsjalr x1,rs,0jump and link registerretjalr x0,x1,0return from subroutinecall targetauipc x6,target[31:12] jalr x1,x6,target[11:0]call far-away subroutinetail targetjalr x0,x6,target[11:0]	bleu rs,rt,target	bgeu rt,rs,target	branch if ≤, unsigned
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 target jalr x1,x6,target[31:12] tail target auipc x6,target[11:0] tail target jalr x0,x6,target[11:0]	j target	jal x0,target	jump
jalr rsjalr x1,rs,0jump and link registerretjalr x0,x1,0return from subroutinecall targetauipc x6,target[31:12] jalr x1,x6,target[11:0]call far-away subroutinetail targetauipc x6,target[31:12] jalr x0,x6,target[11:0]tail call far-away subroutine	jal target	jal x1,target	jump and link
ret jalr x0,x1,0 return from subroutine call target auipc x6,target[31:12] jalr x1,x6,target[11:0] tail target jalr x0,x6,target[11:0] tail target jalr x0,x6,target[11:0]	jr rs	jalr x0,rs,0	jump register
call target auipc x6,target[31:12] jalr x1,x6,target[11:0] tail target auipc x6,target[11:0] tail target auipc x6,target[11:0] tail call far-away subroutine jalr x0,x6,target[11:0]	jalr rs		
tail target jalr x1,x6,target[11:0] auipc x6,target[31:12] jalr x0,x6,target[11:0] call far-away subroutine tail call far-away subroutine	ret	jalr x0,x1,0	return from subroutine
jalr x0,x6,target[11:0]	call target	· · · · · · · · · · · · · · · · · · ·	call far-away subroutine
fence fence iorw, iorw fence on all memory and I/O	tail target	<pre>jalr x0,x6,target[11:0]</pre>	•
	fence	fence iorw,iorw	fence on all memory and I/O