RISC-V REFERENCE

James Zhu <jameszhu@berkeley.edu>

RISC-V Instruction Set

Core Instruction Formats

31 27 26 25	24 20	19	15	14	12	11	7	6	0	
funct7 rs2		rs1 fu		fun	funct3		rd	opcode		R-type
imm[11:0]		rs1	rs1 ft			rd		opcode		I-type
imm[11:5]	rs2	rs1		fun	ct3	imm	[4:0]	opcode		S-type
imm[12 10:5]	rs2	rs1		fun	ct3	imm[4:1 11]		opcode		B-type
	imm[31	1	rd	opcode		U-type				
im			1	rd	opcode		J-type			

RV32I Base Integer Instructions

Inst	Name	FMT	Opcode	funct3	funct7	Description (C)	Note
add	ADD	R	0110011	0x0	0x00	rd = rs1 + rs2	
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	0x2	0x00	rd = rs1 >> rs2	
sra	Shift Right Arith*	R	0110011	0x3	0x20	rd = rs1 >> rs2	msb-extends
slt	Set Less Than	R	0110011	0x2		rd = (rs1 < rs2)?1:0	
sltu	Set Less Than (U)	R	0110011	0x3		rd = (rs1 < rs2)?1:0	zero-extends
addi	ADD Immediate	I	0010011	0x0	0x00	rd = rs1 + imm	
xori	XOR Immediate	I	0010011	0x4	0x00	rd = rs1 ^ imm	
ori	OR Immediate	I	0010011	0x6	0x00	rd = rs1 imm	
andi	AND Immediate	I	0010011	0x7	0x00	rd = rs1 & imm	
slli	Shift Left Logical Imm	I	0010011	0x1	0x00	rd = rs1 << imm	
srli	Shift Right Logical Imm	I	0010011	0x1	0x00	rd = rs1 >> imm	
srai	Shift Right Arith Imm	I	0010011	0x3	0x20	rd = rs1 >> imm	msb-extends
slti	Set Less Than Imm	I	0010011	0x2		rd = (rs1 < imm)?1:0	
sltiu	Set Less Than Imm (U)	I	0010011	0x3		rd = (rs1 < imm)?1:0	zero-extends
lb	Load Byte	I	0000011	0x0		rd = M[rs1+imm][0:7]	
lh	Load Half	I	0000011	0x1		rd = M[rs1+imm][0:15]	
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]	zero-extends
lhu	Load Half (U)	I	0000011	0x5		rd = M[rs1+imm][0:15]	zero-extends
sb	Store Byte	S	0100011	0x0		M[rs1+imm][0:7] = rs2[0:7]	
sh	Store Half	S	0100011	0x1		M[rs1+imm][0:15] = rs2[0:15]	
SW	Store Word	S	0100011	0x2		M[rs1+imm][0:31] = rs2[0:31]	
beq	Branch ==	В	1100011	0x0		if(rs1 == rs2) PC += imm	
bne	Branch !=	В	1100011	0x1		if(rs1 != rs2) PC += imm	
blt	Branch <	В	1100011	0x4		if(rs1 < rs2) PC += imm	
bge	Branch ≥	В	1100011	0x5		if(rs1 >= rs2) PC += imm	
bltu	Branch < (U)	В	1100011	0x6		if(rs1 < rs2) PC += imm	zero-extends
bgeu	Branch \geq (U)	В	1100011	0x7		if(rs1 >= rs2) PC += imm	zero-extends
jal	Jump And Link	J	1101111			rd = PC+4; PC += imm	
jalr	Jump And Link Reg	I	1100111	0x0		rd = PC+4; PC = rs1 + imm	
lui	Load Upper Imm	U	0110111			rd = imm << 12	
auipc	Add Upper Imm to PC	U	0010111			rd = PC + (imm << 12)	
ecall	Environment Call	I	1110011	0x0	0x00	Transfer control to OS	imm: 0x000
ebreak	Environment Break	I	1110011	0x0	0x00	Transfer control to debugger	imm: 0x001

Standard Extensions

RV32M Multiply Extension

Inst	Name	FMT	Opcode	funct3	funct7	Description (C)
mul	MUL	R	0110011	0x0	0x01	rd = (rs1 * rs2)[31:0]
mulh	MUL High	R	0110011	0x1	0x01	rd = (rs1 * rs2)[63:32]
mulsu	MUL High (S) (U)	R	0110011	0x2	0x01	rd = (rs1 * rs2)[63:32]
mulu	MUL High (U)	R	0110011	0x3	0x01	rd = (rs1 * rs2)[63:32]
div	DIV	R	0110011	0x4	0x01	rd = rs1 / rs2
divu	DIV (U)	R	0110011	0x5	0x01	rd = rs1 / rs2
rem	Remainder	R	0110011	0x6	0x01	rd = rs1 % rs2
remu	Remainder (U)	R	0110011	0x7	0x01	rd = rs1 % rs2

RV32A Atomic Extension

31	27	' 26	25	24	20	19	15 14	12	2 11 7	7 6 0		
func	t5	aq	rl	r	rs2		fu	nct3	rd	opcode		
5		1	1	•	5	5		3 5 7				
Inst	Name			FMT	Opcode	funct3	funct5	et5 Description (C)				
lr.w	Load	Reserv	ed	R	0101111	0x2	0x02	rd	= M[rs1], rese	rve M[rs1]		
SC.W	Store	Condit	ional	R	0101111	0x2	0x03	if (reserved) { M[rs1] = rs2; rd = 0				
								els	else { rd = 1 }			
amoswap.w	Atomi	ic Swa _l)	R	0101111	0x2	0x01	rd	rd = M[rs1]; swap(rd, rs2); M[rs1] = r			
amoadd.w	Atomi	ic ADD		R	0101111	0x2	0x00	rd	= M[rs1] + rs2	; M[rs1] = rd		
amoand.w	Atomi	ic AND		R	0101111	0x2	0x0C	rd	= M[rs1] & rs2	; M[rs1] = rd		
amoor.w	Atomi	ic OR		R	0101111	0x2	0x0A	rd	rd = M[rs1] rs2; M[rs1] = rd			
amoxor.w	Atomi	ix XOR		R	0101111	0x2	0x04	rd = M[rs1] ^ rs2; M[rs1] = rd				
amomax.w	Atomi	ic MAX		R	0101111	0x2	0x14	rd = max(M[rs1], rs2); M[rs1] = rd				
amomin.w	Atomi	ic MIN		R	0101111	0x2	0x10	rd = min(M[rs1], rs2); M[rs1] = rd				

RV32F Single-Precision Floating-Point Extension

R4-type instructions

						sti uctio					
			24 20		15 14		12 11		7 6		0
	rs3	fmt	rs2	rs1		func	t3	rd		opcode	:
	5	2	5	5		3		5		7	
Inst	Name		FMT	Opcode	func	t3/rm	funct7	rsi	2 [2024]	fmt	Description (C)
flw	Flt Load Word		I	0000111	010						rd = M[rs1 + imm]
fsw	Flt Store Word		I	0100111	010						M[rs1 + imm] = rs2
fadd.s	Flt Add		R	1010011			0000000				rd = rs1 + rs2
fsub.s	Flt Sub		R	1010011			0000010				rd = rs1 - rs2
fmul.s	Flt Mul		R	1010011			0000100				rd = rs1 * rs2
fdiv.s	Flt Div		R	1010011			0000110				rd = rs1 / rs2
fsgnj.s	Flt Sign Injection		R	1010011	000		0001000				rd = abs(rs1) * sgn(rs2)
fsgnjn.s	Flt Sign Neg Inject	tion	R	1010011	001		0001000				rd = abs(rs1) * -sgn(rs2)
fsgnjx.s	Flt Sign Xor Inject	ion	R	1010011	010		0001000				rd = rs1 * sgn(rs2)
fmin.s	Flt Minimum		R	1010011			0001100				rd = min(rs1, rs2)
fmax.s	Flt Maximum		R	1010011			0001100				rd = max(rs1, rs2)
fsqrt.s	Flt Square Root		R	1010011			0101100	00	000		rd = sqrt(rs1)
fle.s	Float Less / Equal		R	1010011	000		0101000				rd = (rs1 <= rs2) ? 1 : 0
flt.s	Float Less Than		R	1010011	001		0101000				rd = (rs1 < rs2) ? 1 : 0
feq.s	Float Equality		R	1010011	010		0101000				rd = (rs1 == rs2) ? 1 : 0
fcvt.w.s	Flt Convert to Int		R	1010011			0110000	00	000		rd = (int32_t) rs1
fcvt.wu.s	Flt Convert to Int		R	1010011			0110000	00	001		rd = (uint32_t) rs1
fmv.x.w	Move Float to Int		R	1010011	000		0111000	00	000		rd = *((int*) &rs1)
fclass.s	Float Classify		R	1010011	001		0111000	00	000		rd = 09
fcvt.s.w	Flt Conv from Sign	n Int	R	1010011			0110100	00	000		rd = (float) rs1
fcvt.s.wu	Flt Conv from Uns	Int	R	1010011			0110100	00	001		rd = (float) rs1
fmv.w.x	Move Int to Float		R	1010011	000		0111100	00	000		rd = *((float*) &rs1)
fmadd.s	Flt Fused Mul-Ado	1	R4	1000011						00	rd = rs1 * rs2 + rs3
fmsub.s	Flt Fused Mul-Sub)	R4	1000111						00	rd = rs1 * rs2 - rs3
fnmadd.s	Flt Neg Fused Mul	l-Add	R4	1001011						00	rd = -rs1 * rs2 + rs3
fnmsub.s	Flt Neg Fused Mul	l-Sub	R4	1001111						00	rd = -rs1 * rs2 - rs3
							_	_		_	

RV32D Double-Precision Floating-Point Extension

Inst	Name	FMT	Opcode	funct3/rm	funct7	rs2 [2024]	fmt	Description (C)
fld	Flt Load Word	I	0000111	011				rd = M[rs1 + imm]
fsd	Flt Store Word	I	0100111	011				M[rs1 + imm] = rs2
fadd.d	Flt Add	R	1010011		0000001			rd = rs1 + rs2
fsub.d	Flt Sub	R	1010011		0000011			rd = rs1 - rs2
fmul.d	Flt Mul	R	1010011		0000101			rd = rs1 * rs2
fdiv.d	Flt Div	R	1010011		0000111			rd = rs1 / rs2
fsgnj.d	Flt Sign Injection	R	1010011	000	0001001			rd = abs(rs1) * sgn(rs2)
fsgnjn.d	Flt Sign Neg Injection	R	1010011	001	0001001			rd = abs(rs1) * -sgn(rs2)
fsgnjx.d	Flt Sign Xor Injection	R	1010011	010	0001001			rd = rs1 * sgn(rs2)
fmin.d	Flt Minimum	R	1010011		0001101			rd = min(rs1, rs2)
fmax.d	Flt Maximum	R	1010011		0001101			rd = max(rs1, rs2)
fsqrt.d	Flt Square Root	R	1010011		0101101	00000		rd = sqrt(rs1)
fle.d	Float Less / Equal	R	1010011	000	0101001			rd = (rs1 <= rs2) ? 1 : 0
flt.d	Float Less Than	R	1010011	001	0101001			rd = (rs1 < rs2) ? 1 : 0
feq.d	Float Equality	R	1010011	010	0101001			rd = (rs1 == rs2) ? 1 : 0
fcvt.w.d	Flt Convert to Int	R	1010011		0110001	00000		rd = (int32_t) rs1
fcvt.wu.d	Flt Convert to Int	R	1010011		0110001	00001		rd = (uint32_t) rs1
fcvt.s.d	Double Flt to Single	R	1010011	000	0010000	00001		rd = *((float*) &rs1)
fmv.x.d	Move Int to Float	R	1010011	000	0111101	00000		rd = *((float*) &rs1)
fclass.d	Float Classify	R	1010011	001	0111001	00000		rd = 09
fcvt.d.w	Flt Conv from Sign Int	R	1010011		0110101	00000		rd = (float) rs1
fcvt.d.wu	Flt Conv from Uns Int	R	1010011		0110101	00001		rd = (float) rs1
fcvt.d.s	Single Flt to Double	R	1010011	000	0010001	00001		rd = *((float*) &rs1)
fmadd.d	Flt Fused Mul-Add	R4	1000011				01	rd = rs1 * rs2 + rs3
fmsub.d	Flt Fused Mul-Sub	R4	1000111				01	rd = rs1 * rs2 - rs3
fnmadd.d	Flt Neg Fused Mul-Add	R4	1001011				01	rd = -rs1 * rs2 + rs3
fnmsub.d	Flt Neg Fused Mul-Sub	R4	1001111				01	rd = -rs1 * rs2 - rs3

RV32C Compressed Extension

15 14 13	12	11	10	9	8	7	6	5	4	3	2	1	0		
funct	4		ro	l/rs	1				rs2			О	p	CR-type	
funct3	imm	imm rd/rs1					imm				О	p	CI-type		
funct3	imm				rs2				op CSS-type						
funct3		imm						rd'			0	p	CIW-type		
funct3	in	ım			rs1'		im	m		rd'		О	p	CL-type	
funct3	in	imm rd'/rs1'		im	m	rs2'			О	p	CS-type				
funct3	in	imm rs1'					imm				0	p	CB-type		
funct3	offset						0	p	CJ-type						

Inst	Name	FMT	OP	Funct	Description
c.lwsp	Load Word from SP	CI	10	010	lw rd, (4*imm)(sp)
c.swsp	Store Word to SP	CSS	10	110	sw rs2, (4*imm)(sp)
c.lw	Load Word	CL	00	010	lw rd', (4*imm)(rs1')
C.SW	Store Word	CS	00	110	sw rs1', (4*imm)(rs2')
c.j	Jump	CJ	01	101	jal x0, 2*offset
c.jal	Jump And Link	CJ	01	001	jal ra, 2*offset
c.jr	Jump Reg	CR	10	1000	jalr x0, rs1, 0
c.jalr	Jump And Link Reg	CR	10	1001	jalr ra, rs1, 0
c.beqz	Branch == 0	CB	01	110	beq rs', x0, 2*imm
c.bnez	Branch != 0	CB	01	111	bne rs', x0, 2*imm
c.li	Load Immediate	CI	01	010	addi rd, x0, imm
c.lui	Load Upper Imm	CI	01	011	lui rd, imm
c.addi	ADD Immediate	CI	01	000	addi rd, rd, imm
c.addi16sp	ADD Imm * 16 to SP	CI	01	011	addi sp, sp, 16*imm
c.addi4spn	ADD Imm * 4 + SP	CIW	00	000	addi rd', sp, 4*imm
c.slli	Shift Left Logical Imm	CI	10	000	slli rd, rd, imm
c.srli	Shift Right Logical Imm	CB	01	100x00	srli rd', rd', imm
c.srai	Shift Right Arith Imm	CB	01	100x01	srai rd', rd', imm
c.andi	AND Imm	CB	01	100x10	andi rd', rd', imm
c.mv	MoVe	CR	10	1000	add rd, x0, rs2
c.add	ADD	CR	10	1001	add rd, rd, rs2
c.and	AND	CS	01	10001111	and rd', rd', rs2'
c.or	OR	CS	01	10001110	or rd', rd', rs2'
c.xor	XOR	CS	01	10001101	xor rd', rd', rs2'
c.sub	SUB	CS	01	10001100	sub rd', rd', rs2'
c.nop	No OPeration	CI	01	000	addi x0, x0, 0
c.ebreak	Environment BREAK	CR	10	1001	ebreak

Pseudo Instructions

1a rd, symbol	Pseudoinstruction	Base Instruction(s)	Meaning
	la rd, symbol		Load address
S(b h w d) rd, symbol, rt S(b h w d) rd, symbol[31:12] S(b h w d) rd, symbol[31:12] fl(w d) rd, symbol[31	l{b h w d} rd, symbol		Load global
	s{b h w d} rd, symbol, rt	auipc rt, symbol[31:12]	Store global
Fs(w d) rd, symbol, rt	fl{w d} rd, symbol, rt	auipc rt, symbol[31:12]	Floating-point load global
No operation No operation	fs{w d} rd, symbol, rt	auipc rt, symbol[31:12]	Floating-point store global
li rd, immediate mv rd, rs addi rd, rs, 0 Copy register One's complement one rd, rs sub rd, x0, rs Two's complement Two's confile Two's complement Two's complement Two's complement Two's confi	nop		No operation
mv rd, rs not rd, rs not rd, rs neg rd, rs sub rd, x0, rs negw rd, rs subw rd, x0, rs sext.w rd, rs set if > zero sext.x res sext.x rd, rs sext.x rd sext.x rd sext.x rd sext.x rd sext.x rd sext.x r			-
not rd, rs neg rd, rs neg rd, rs sub rd, x0, rs negw rd, rs subw rd, x0, rs sext.w rd, rs sext.w rd, rs seqz rd, rs sets sltu rd, rs, 1 seqz rd, rs sltu rd, x0, rs sets if = zero snez rd, rs sltu rd, x0, rs set if ≠ zero sltz rd, rs slt rd, x0, rs set if ≠ zero slt rd, rs slt rd, x0, rs set if ≠ zero set if ≥ zero set if	<i>'</i>		Copy register
neg rd, rs negw rd, rs subw rd, x0, rs sext.w rd, rs set addiw rd, rs, 0 seqz rd, rs sltu rd, x0, rs sltu rd, x0, rs set if ≠ zero snez rd, rs sltu rd, x0, rs slt rd, x0, rs set if ≠ zero sltz rd, rs slt rd, x0, rs slt rd, x0, rs set if ≠ zero set if ≥	•		10 0
negw rd, rs subw rd, x0, rs addiw rd, rs, 0 sign extend word sext.w rd, rs stitu rd, rs, 1 set if $=$ zero snez rd, rs sttu rd, x0, rs set if \neq zero sltz rd, rs stt rd, x0, rs set if \neq zero sgtz rd, rs stt rd, x0, rs set if \neq zero sgtz rd, rs stt rd, x0, rs set if \neq zero sgtz rd, rs stt rd, x0, rs set if \neq zero sgtz rd, rs stt rd, x0, rs set if \neq zero sgtz rd, rs stt rd, x0, rs set if \neq zero sgtz rd, rs stgnj.s rd, rs, rs single-precision register fabs.s rd, rs fsgnjs.s rd, rs, rs single-precision absolute value fneg.s rd, rs fsgnj.s rd, rs, rs single-precision negate fwv.d rd, rs fsgnj.d rd, rs, rs copy double-precision negate fwd.d rd, rs fsgnj.d rd, rs, rs pouble-precision negate for a standard rd, rs fsgnj.d rd, rs, rs pouble-precision negate beqz rs, offset beq rs, x0, offset branch if \neq zero bnez rs, offset beq rs, x0, offset branch if \neq zero blez rs, offset bge x0, rs, offset branch if \neq zero bgz rs, offset blt rs, x0, offset branch if \neq zero bgt rs, rt, offset blt x0, rs, offset branch if \neq zero bgt rs, rt, offset blt rt, rs, offset branch if \neq zero bgt rs, rt, offset blt rt, rs, offset branch if \neq zero bgt rs, rt, offset blt rt, rs, offset branch if \neq zero bgt rs, rt, offset blt rt, rs, offset branch if \neq zero bgt rs, rt, offset blt rt, rs, offset branch if \neq branch if \neq branch if \neq branch if \neq point if set if the register branch if \neq branch	•		-
sext.w rd, rs seqz rd, rs stiu rd, rs, 0 seqz rd, rs stiu rd, rs, 1 set if = zero seqz rd, rs stiu rd, x0, rs set if \neq zero seqz rd, rs stiu rd, x0, rs set if \neq zero sequ rd, rs stir rd, x0, rs set if \neq zero sequ rd, rs set rd, rs stir rd, x0, rs set if \neq zero sequence results rd, rs stir rd, x0, rs set if \neq zero sequence results rd, rs sequence results rd, rs sequence results rd, rs sequence results rd, rs, rs sequence results rd, rs fearly x0, rs, rs sequence results r			•
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 \neq zero sgtz rd, rs slt rd, x0, rs Set if \neq zero sgtz rd, rs slt rd, x0, rs Set if \neq zero fmv.s rd, rs fsgnjs. rd, rs, rs Copy single-precision register fabs.s rd, rs fsgnjx.s rd, rs, rs Single-precision absolute value fneg.s rd, rs fsgnjx.s rd, rs, rs Single-precision negate fmv.d rd, rs fsgnjx.d rd, rs, rs Copy double-precision register fabs.d rd, rs fsgnjx.d rd, rs, rs Copy double-precision negate fmv.d rd, rs fsgnjx.d rd, rs, rs Copy double-precision negate for gas, of rs fsgnjx.d rd, rs, rs Copy double-precision negate for gas, of rs fsgnjx.d rd, rs, rs Double-precision negate for gas, of rs fsgnjx.d rd, rs, rs Double-precision negate for gas, of rs fsgnjx.d rd, rs, rs Double-precision negate for gas, of rs fsgnjx.d rd, rs, rs fsgnjx.d rd, rs, rs Double-precision negate for gas, of rs fsgnjx.d rd, rs, rs fsgnjx.d rd, rs, rs Double-precision negate from the recision pount of gas from the recision negate			
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 fsgnjx.s rd, rs, rs Single-precision negate fmv.d rd, rs fsgnjx.d rd, rs, rs Copy double-precision register fabs.d rd, rs fsgnjx.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 fsgnjx.d rd, rs, rs Double-precision absolute value fneg.d rd, rs fsgnjx.d rd, rs, rs Double-precision negate beqz rs, offset beq rs, x0, offset Branch if = zero brez rs, offset but rs, x0, offset Branch if \neq zero blez rs, offset bgx x0, rs, offset Branch if \neq zero blez rs, offset bgx x0, rs, offset Branch if \neq zero bltz rs, offset blt rs, x0, offset Branch if \neq zero bgtz rs, offset blt x0, rs, offset Branch if \neq zero bgtz rs, offset blt x0, rs, offset Branch if \neq zero bgtz rs, rt, offset blt rt, rs, offset Branch if \neq zero bgt rs, rt, offset blt rt, rs, offset Branch if \neq zero bgt rs, rt, offset bge rt, rs, offset Branch if \neq zero bgt rs, rt, offset bgr rt, rs, offset Branch if \neq zero bgt rs, rt, offset bgr rt, rs, offset Branch if \neq zero bgt rs, rt, offset bgr rt, rs, offset Branch if \neq zero bgt rs, rt, offset bgr rt, rs, offset Branch if \neq zero bgt rs, rt, offset bgr rt, rs, offset Branch if \neq zero bgt rs, rt, offset bgr rt, rs, offset Branch if \neq zero bgt rs, rt, offset bgr rt, rs, offset Branch if \neq zero bgt rs, rt, offset bgr rt, rs, offset Branch if \neq zero bgt rs, rt, offset bgr rt, rs, offset Branch if \neq zero bgt rs, rt, offset bgr rt, rs, offset Branch if \neq zero bgt rs, rt, offset bgr rt, rs, offset Branch if \neq zero bgt rs, rt, offset bgr rt, rs, offset Branch if \neq zero bgt rt, rs, offset Branch if \neq zero bgt rt, rs, offset bgr rt, rs, offset Branch if \neq zero bgt rt, rs, offset bgr rt, rs, offset bgr rt, rs, offset bgr rt, rs, offset bgr rt			
sltz rd, rs sgtz rd, rs sgt rd, rs sgtz rd, rs single-precision register Single-precision register Single-precision register Single-precision register Single-precision register Stingle-precision register Single-precision register Supele-precision register Single-precision register Supele-recision register Supele-recision register Supele-recision register Supele-recison single-precision Single-precision Single-precision Single-precision Single-pr			
sgtz rd, rs fmv.s rd, rs fsgnj.s rd, rs, rs fsgnjx.s rd, rs, rs fsgnje-precision negate fmv.d rd, rs fsgnjx.d rd, rs, rs fsgnjx.d rd, rs, rs fopy double-precision register Single-precision negate Copy double-precision register Copy double-precision register Copy double-precision register Single-precision negate Copy double-precision negate Copy double-precision register Single-precision negate Copy double-precision negate Branch if ≤ zero Branch i			•
$\begin{array}{cccccccccccccccccccccccccccccccccccc$			
fabs.s rd, rs fsgnjx.s rd, rs, rs fneg.s rd, rs fneg.s rd, rs fsgnjn.s rd, rs, rs fsgnjn.s rd, rs, rs fsgnje-precision absolute value fneg.s rd, rs fsgnjn.s rd, rs, rs fsgnje-precision negate 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 absolute value beqz rs, offset beq rs, x0, offset bnez rs, offset bnez rs, offset blez rs, offset blt rs, x0, offset blt rs, x0, offset blt rs, x0, offset blt rs, x0, offset blt rs, rt, offset blt rs, rt, offset blt rt, rs, offset ble rs, rt, offset blt rt, rs, offset bleu rs, rt, offset bleu rs, rt, offset bleu rs, rt, offset jal x0, offset jal x1, offset jal x1, offset jal x1, offset jal x2, x1, x1, x3 Jump and link jr rs jalr rs jalr x1, x1, x3 Jump and link register ret jalr x0, x1, 0 Return from subroutine Call far-away subroutine tail offset jal x0, x6, offset[11:0] Tail call far-away subroutine			
fneg.s rd, rs fsgnjn.s rd, rs, rs fsgnje-precision negate fmv.d rd, rs fsgnj.d rd, rs, rs fsgnjx.d rd, rs, rs pouble-precision absolute value fneg.d rd, rs beq rs, soffset beq rs, x0, offset blez rs, offset beg x0, rs, offset beg rs, x0, offset blt rs, rt, offset blt rr, rs, offset blt rr, rs, offset ble rs, rt, offset blu rt, rs, offset blu rs, rt, offset blu rt, rs, offset blu rt, rs, offset blu rs, rt, offset blu rt, rs, offset blu rs, rt, offset blu rs, rt, offset blu rt, rs, offset blu rs, rt, offset blu rs, rt, offset blu rt, rs, offset blu rs, rt, offset blu rs, rt, offset blu rs, rt, offset blu rt, rs, offset blu rs, rt, offset blu rs, rt, offset blu rt, rs, offset blu rs, rt, offset blu rs, rt, offset blu rs, rt, offset blu rt, rs, offset blu rs, rt, offset blu rs, rt, offset blu rs, rt, offset blu rt, rs, offset blu rs, rt, offset blu rt, rs, offset blu rs, rt, offset blu rs, rt, offset blu rt, rs, offset blu rs, rt, offset blu rs, rt, offset blu rt, rs, offset blu rs, rt, offset blu rs, rt, offset blu rt, rs, offset blu rs, rt, offset blu rt, rs, offset branch if ≤ zero branch if ≤	,		
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 absolute value beqz rs, offset beq rs, x0, offset Branch if = zero blez rs, offset bge x0, rs, offset Branch if \neq zero blez rs, offset bge x0, rs, offset Branch if \leq zero bgez rs, offset blt rs, x0, offset Branch if \leq zero bgtz rs, offset blt rs, x0, offset Branch if \leq zero bgtz rs, offset blt x0, rs, offset Branch if $>$ zero bgtz rs, offset blt x0, rs, offset Branch if $>$ ble rs, rt, offset blt rt, rs, offset Branch if $>$ ble rs, rt, offset bge rt, rs, offset Branch if $>$ bgtu rs, rt, offset bge rt, rs, offset Branch if $>$ bgtu rs, rt, offset bge rt, rs, offset Branch if $>$ bgtu rs, rt, offset bgeu rt, rs, offset Branch if $>$ unsigned bleu rs, rt, offset bgeu rt, rs, offset Branch if $>$ unsigned j offset jal x0, offset Jump jal offset jal x1, offset Jump and link jr rs jalr x0, rs, 0 Jump and link register ret jalr x0, x1, 0 Return from subroutine call offset jal x1, offset[31:12] jalr x1, x1, offset[31:12] jalr x1, x1, offset[31:12] Tail call far-away subroutine			
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 bge x0, rs, offset Branch if ≤ zero blez rs, offset bge x0, rs, offset Branch if ≤ zero bgez rs, offset bge rs, x0, offset Branch if ≤ zero bgtz 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 > blt rt, rs, offset Branch if > blt rt, rs, offset Branch if ≤ bgu rs, rt, offset bge rt, rs, offset Branch if > blt rt, rs, offset Branch if > unsigned bleu rs, rt, offset bltu rt, rs, offset Branch if >, unsigned Branch if ≤, unsigned j offset jal x0, offset Jump jal offset jal x1, offset Jump jal offset jal x1, offset Jump and link jr rs jalr x0, rs, 0 Jump and link jr rs jalr x0, rs, 0 Jump register ret jalr x0, x1, 0 Return from subroutine call offset auipc x1, offset[31:12] jalr x1, x1, offset[11:0] tail offset Tail call far-away subroutine Tail call far-away subroutine			
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 ≠ zero blez rs, offset bge x0, rs, offset Branch if ≤ zero bgez rs, offset bge rs, x0, offset Branch if ≤ zero bgtz rs, offset blt rs, x0, offset Branch if < zero bgtz rs, offset blt rs, x0, offset Branch if > zero bgt rs, rt, offset blt rt, rs, offset Branch if > beg rt, rs, offset Branch if > ble rs, rt, offset bge rt, rs, offset Branch if ≤ bgtu rs, rt, offset bge rt, rs, offset Branch if ≤ bgtu rs, rt, offset bltu rt, rs, offset Branch if ≤ bgtu rs, rt, offset bgeu rt, rs, offset Branch if ≤, unsigned bleu rs, rt, offset bgeu rt, rs, offset Branch if ≤, unsigned j offset jal x0, offset Jump jal offset jal x0, rs, 0 Jump and link jr rs jalr x0, 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] tail offset jalr x0, x6, offset[11:0] Tail call far-away subroutine			
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 \leq zero bltz rs, offset blt rs, x0, offset Branch if \leq zero bgtz rs, offset blt x0, rs, offset Branch if \leq zero bgt rs, rt, 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 bge rt, rs, offset Branch if \leq bgtu rs, rt, offset bltu rt, rs, offset Branch if \leq nunsigned bleu rs, rt, offset bgeu rt, rs, offset Branch if \leq , unsigned bleu rs, rt, 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] Tail call far-away subroutine			
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 \leq zero bltz rs, offset blt rs, x0, offset Branch if \leq 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 $>$ blurs, rt, offset bge rt, rs, offset Branch if $>$ blurs, rt, offset bltu rt, rs, offset Branch if $>$ unsigned bleu rs, rt, offset bgeu rt, rs, offset Branch if \leq unsigned bleu rs, rt, offset bgeu rt, rs, offset Branch if \leq unsigned jal offset jal x0, offset Jump jal offset jal x1, offset Jump and link jr rs jalr x0, rs, 0 Jump register jalr rs jalr x0, 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] Tail call far-away subroutine			
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 $>$ blt rt, rs, offset Branch if $>$ blt rt, rs, offset Branch if $>$ blt rt, rs, offset Branch if \leq bgtu rs, rt, offset bltu 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 bleu rs, rt, offset bgeu rt, rs, offset Branch if \leq , unsigned bleu rs, rt, offset bgeu rt, rs, offset Branch if \leq , unsigned bleu rs, rt, offset bgeu rt, rs, offset Branch if \leq , unsigned bleu rs, rt, offset bgeu rt, rs, offset Branch if \leq unsigned bleu rs, rt, offset bgeu rt, rs, offset Branch if \leq unsigned bleu rs, rt, offset bgeu rt, rs, offset Branch if \leq unsigned bleu rs, rt, offset Branch if \leq unsigned bleu rs, rt, offset Branch if \leq unsigned bleu rs, rt, offset Branch if \leq unsigned Branch if \leq unsigned bleu rs, rt, offset Branch if \leq unsigned			
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 $>$ blt rt, rs, offset Branch if \leq bgt rs, rt, offset bgt rt, rs, offset Branch if \leq bgt rs, rt, offset bltu rt, rs, offset Branch if \leq bgt rs, rt, offset bltu rt, rs, offset Branch if \leq unsigned bleu rs, rt, offset bgeu rt, rs, offset Branch if \leq , unsigned \neq j offset jal x0, offset Jump jal offset jal x1, offset Jump and link jr rs jalr x0, rs, 0 Jump and link register ret jalr x0, x1, 0 Return from subroutine call offset jalr x1, x1, offset[31:12] jalr x1, x1, offset[31:12] Call far-away subroutine tail offset jalr x0, x6, offset[31:12] Tail call far-away subroutine			•
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 ≤ bgtu rs, rt, offset bltu rt, rs, offset Branch if ≤ bgtu rs, rt, offset bltu rt, rs, offset Branch if >, unsigned bleu rs, rt, offset bgeu rt, rs, offset Branch if ≤, 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 jalr x1, x1, offset[31:12] jalr x1, x1, offset[11:0] tail offset jalr x0, x6, offset[11:0] Tail call far-away subroutine	blez rs, offset		
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 ≤ bgtu rs, rt, offset bltu rt, rs, offset Branch if >, unsigned bleu rs, rt, offset bgeu rt, rs, offset Branch if ≤, 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 jalr x1, x1, offset[31:12] jalr x1, x1, offset[31:12] jalr x0, x6, offset[31:12] Tail call far-away subroutine	bgez rs, offset	bge rs, x0, offset	Branch if \geq zero
bgt rs, rt, offset ble rs, rt, offset bge rt, rs, offset bgtu rs, rt, offset bgu rs, rt, offset blu rt, rs, offset blu rs, rt, offset bgu rt, rs,	bltz rs, offset	blt rs, x0, offset	
ble rs, rt, offset bge rt, rs, offset Branch if \leq bgtu rs, rt, offset blu rt, rs, offset Branch if $>$, unsigned bleu rs, rt, offset bgeu rt, rs, offset Branch if \leq , unsigned \neq 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[31:12] jalr x0, x6, offset[31:12] Tail call far-away subroutine	bgtz rs, offset	blt x0, rs, offset	Branch if > zero
bgtu rs, rt, offset bltu rt, rs, offset Branch if >, unsigned bleu rs, rt, offset bgeu rt, rs, offset Branch if ≤, 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] tail offset jalr x0, x6, offset[11:0] Tail call far-away subroutine	bgt rs, rt, offset	blt rt, rs, offset	Branch if >
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] Tail call far-away subroutine jalr x0, x6, offset[11:0]	ble rs, rt, offset	bge rt, rs, offset	Branch if \leq
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] Tail call far-away subroutine jalr x0, x6, offset[11:0]			Branch if >, unsigned
j offsetjal x0, offsetJumpjal offsetjal x1, offsetJump and linkjr rsjalr x0, rs, 0Jump registerjalr rsjalr x1, rs, 0Jump and link registerretjalr x0, x1, 0Return from subroutinecall offsetauipc x1, offset[31:12] jalr x1, x1, offset[11:0]Call far-away subroutinetail offsetauipc x6, offset[31:12] jalr x0, x6, offset[11:0]Tail call far-away subroutine	bleu rs, rt, offset		
jal offsetjal x1, offsetJump and linkjr rsjalr x0, rs, 0Jump registerjalr rsjalr x1, rs, 0Jump and link registerretjalr x0, x1, 0Return from subroutinecall offsetauipc x1, offset[31:12] jalr x1, x1, offset[11:0]Call far-away subroutinetail offsetauipc x6, offset[31:12] jalr x0, x6, offset[11:0]Tail call far-away subroutine			
jr rsjalr x0, rs, 0Jump registerjalr rsjalr x1, rs, 0Jump and link registerretjalr x0, x1, 0Return from subroutinecall offsetauipc x1, offset[31:12] jalr x1, x1, offset[11:0]Call far-away subroutinetail offsetauipc x6, offset[31:12] jalr x0, x6, offset[11:0]Tail call far-away subroutine	-	=	
jalr rsjalr x1, rs, 0Jump and link registerretjalr x0, x1, 0Return from subroutinecall offsetauipc x1, offset[31:12] jalr x1, x1, offset[11:0]Call far-away subroutinetail offsetauipc x6, offset[31:12] jalr x0, x6, offset[11:0]Tail call far-away subroutine	-		-
ret jalr x0, x1, 0 Return from subroutine call offset auipc x1, offset[31:12] Call far-away subroutine tail offset jalr x0, x6, offset[31:12] Tail call far-away subroutine			1 0
call offset auipc x1, offset[31:12] call far-away subroutine tail offset auipc x6, offset[31:12] auipc x6, offset[31:12] jalr x0, x6, offset[11:0] Tail call far-away subroutine	_		
tail offset jalr x1, x1, offset[11:0] tail offset jalr x0, x6, offset[31:12] jalr x0, x6, offset[11:0] Tail call far-away subroutine			
tail offset auipc x6, offset[31:12] auipc x6, offset[11:0] Tail call far-away subroutine	call offset		Call far-away subroutine
jalr x0, x6, offset[11:0]			
fence fence iorw, iorw Fence on all memory and I/O	tail offset	<pre>jalr x0, x6, offset[11:0]</pre>	<u> </u>
	fence	fence iorw, iorw	Fence on all memory and I/O

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	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
f0-7	ft0-7	FP temporaries	Caller
f8-9	fs0-1	FP saved registers	Callee
f10-11	fa0-1	FP args/return values	Caller
f12-17	fa2-7	FP args	Caller
f18-27	fs2-11	FP saved registers	Callee
f28-31	ft8-11	FP temporaries	Caller