RV64IM_Zicsr Instruction Quick Reference

Base Integer Instructions: RV64I

Name	Instruction	RTL
Shift Left Logical	SLL rd,rs1,rs2	rd ← rs1 << rs2
Shift Left Log. Imm	SLLI rd,rs1,imm	rd ← rs1 << imm[4:0]
Shift Right Logical	SRL rd,rs1,rs2	rd ← rs1 >> rs2
Shift Right Log. Imm	SRLI rd,rs1,imm	rd ← rs1 >> imm[4:0]
Shift Right Arithmetic	SRA rd,rs1,rs2	rd ← rs1 >> rs2
Shift Right Arith. Imm	SRAI rd,rs1,imm	rd ← rs1 >> imm[4:0]
ADD	ADD rd,rs1,rs2	rd ← rs1 + rs2
ADD Immediate	ADDI rd,rs1,imm	rd ← rs1 + imm[11:0]
SUBtract	SUB rd,rs1,rs2	rd ← rs1 - rs2
Load Upper Imm	LUI rd,imm	rd ← imm[19:0] << 12
Add Upper Imm to PC	AUIPC rd,imm	rd ← PC + (imm << 12)
XOR	XOR rd,rs1,rs2	rd ← rs1 ^ rs2
XOR Immediate	XORI rd,rs1,imm	rd ← rs1 ^ imm[11:0]
OR	OR rd,rs1,rs2	rd ← rs1 | rs2
OR Immediate	ORI rd,rs1,imm	rd ← rs1 | imm[11:0]
AND	AND rd,rs1,rs2	rd ← rs1 & rs2
AND Immediate	ANDI rd,rs1,imm	rd ← rs1 & imm[11:0]
Set if <	SLT rd,rs1,rs2	rd ← (rs1 < rs2) ? 1 : 0
Set if < Immediate	SLTI rd,rs1,imm	rd ← (rs1 < imm) ? 1 : 0
Set if < Unsigned	SLTU rd,rs1,rs2	rd ← (rs1 < rs2) ? 1 : 0
Set if < Imm Unsigned	SLTIU rd,rs1,imm	rd ← (rs1 < imm) ? 1 : 0
Branch =	BEQ rs1,rs2,imm	if(rs1 == rs2) PC += imm
Branch ≠	BNE rs1,rs2,imm	if(rs1 != rs2) PC += imm
Branch <	BLT rs1,rs2,imm	if(rs1 < rs2) PC += imm
Branch ≥	BGE rs1,rs2,imm	if(rs1 >= rs2) PC += imm
Branch < Unsigned	BLTU rs1,rs2,imm	if(rs1 < rs2) PC += imm
Branch ≥ Unsigned	BGEU rs1,rs2,imm	if(rs1 >= rs2) PC += imm
Jump & Link	JAL rd,imm	rd ← PC+4; PC += imm
Jump & Link Register	JALR rd,rs1,imm	rd ← PC+4; PC ← rs1 + imm
Synch thread	FENCE	Fence on all memory and I/O
Environment CALL	ECALL	Transfer control to OS
Load Byte	LB rd,imm(rs1)	rd ← M[rs1+imm][7:0]
Load Half word	LH rd,imm(rs1)	rd ← M[rs1+imm][15:0]
Load Word	LW rd,imm(rs1)	rd ← M[rs1+imm][31:0]
Load Double Word	LD rd,imm(rs1)	rd ← M[rs1+imm][63:0]
Load Byte Unsigned	LBU rd,imm(rs1)	rd ← M[rs1+imm][7:0]
Load Half word Unsigned	LHU rd,imm(rs1)	rd ← M[rs1+imm][15:0]
Load Word Unsigned	LWU rd,imm(rs1)	rd ← M[rs1+imm][31:0]
Store Byte	SB rs2,imm(rs1)	M[rs1+imm][7:0] ← rs2[7:0]
Store Half word	SH rs2,imm(rs1)	M[rs1+imm][15:0] ← rs2[15:0]
Store Word	SW rs2,imm(rs1)	M[rs1+imm][31:0] ← rs2[31:0]
Store Double Word	SD rs2,imm(rs1)	M[rs1+imm][63:0] ← rs2[63:0]
Read/Write	CSRRW rd,csr,rs1	rd ← csr ← rs1
Read & Set bit	CSRRS rd,csr,rs1	rd ← csr ← csr | rs1
Read & Clear bit	CSRRC rd,csr,rs1	rd ← csr ← csr & ~rs1
Read/Write Imm	CSRRWI rd,csr,imm	rd ← csr ← imm
Read & Set bit Imm	CSRRSI rd,csr,imm	rd ← csr ← csr | imm
Read & Clear bit Imm	CSRRCI rd,csr,imm	rd ← csr ← csr & ~imm

Multiply-Divide Instructions: RV64M

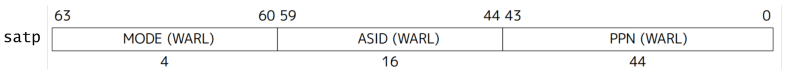
Name	Instruction	RTL
MULtiply	MUL rd,rs1,rs2	rd ← (rs1 * rs2)[63: 0]
MULtiply High	MULH rd,rs1,rs2	rd ← (rs1 * rs2)[127:64]
MULtiply High S/U	MULHSU rd,rs1,rs2	rd ← (rs1 * rs2)[127:64]
MULtiply High Uns	MULHU rd,rs1,rs2	rd ← (rs1 * rs2)[127:64]
DIVide	DIV rd,rs1,rs2	rd ← rs1 / rs2
DIVide Unsigned	DIVU rd,rs1,rs2	rd ← rs1 / rs2
REMainder	REM rd,rs1,rs2	rd ← rs1 % rs2
REMainder Unsigned	REMU rd,rs1,rs2	rd ← rs1 % rs2

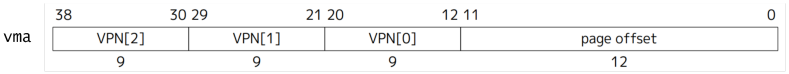
RV Privileged Instructions

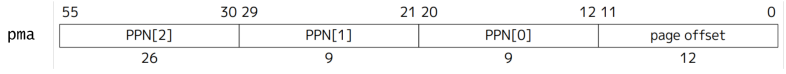
M-mode trap RETurn	MRET	
S-mode trap RETurn	SRET	
Wait For Interrupt	WFI	

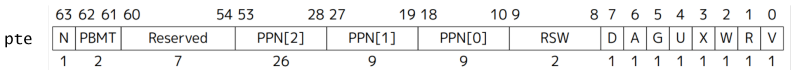
Pseudo-Instructions

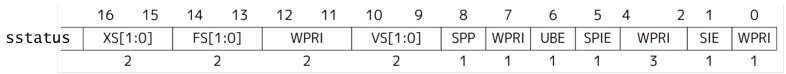
Name	Instruction	RTL
Load address	LA rd, symbol	rd ← symbol
No operation	NOP	
Load immediate	LI rd, imm	rd ← imm
Copy register	MV rd, rs1	rd ← rs1
One's complement	NOT rd, rs1	rd ← ~rs1
Two's complement	NEG rd, rs1	rd ← ~rs1 + 1
Two's complement word	NEGW rd, rs1	rd ← ~rs1 + 1
Sign extend word	SEXT.W rd, rs1	rd ← rs1[31:0]
Set if = zero	SEQZ rd, rs1	rd ← (rs1 == 0) ? 1 : 0
Set if ≠ zero	SNEZ rd, rs1	rd ← (rs1 != 0) ? 1 : 0
Set if < zero	SLTZ rd, rs1	rd ← (rs1 < 0) ? 1 : 0
Set if > zero	SGTZ rd, rs1	rd ← (rs1 > 0) ? 1 : 0
Branch if = zero	BEQZ rs1, imm	if(rs1 == 0) PC += imm
Branch if ≠ zero	BNEZ rs1, imm	if(rs1 != 0) PC += imm
Branch if ≤ zero	BLEZ rs1, imm	if(rs1 <= 0) PC += imm
Branch if ≥ zero	BGEZ rs1, imm	if(rs1 >= 0) PC += imm
Branch if < zero	BLTZ rs1, imm	if(rs1 < 0) PC += imm
Branch if > zero	BGTZ rs1, imm	if(rs1 > 0) PC += imm
Branch >	BGT rs1, rs2 imm	if(rs1 > rs2) PC += imm
Branch ≤	BLE rs1, rs2 imm	if(rs1 <= rs2) PC += imm
Branch > unsigned	BGTU rs1, rs2 imm	if(rs1 > rs2) PC += imm
Branch ≤ unsigned	BLEU rs1, rs2 imm	if(rs1 <= rs2) PC += imm
Jump	J symbol	PC ← symbol
Jump And Link	JAL symbol	PC ← symbol; ra ← PC+4
Jump Register	JR rs1	PC ← rs1
Jump And Link Register	JALR rs1	PC ← rs1; ra ← PC+4
RETurn from subroutine	RET	PC ← ra
CALL subroutine	CALL symbol	PC ← symbol; ra ← PC+4

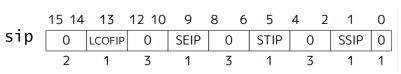


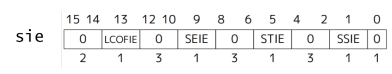












Register	ABI Name	Description	Saver
x0	zero	Hard-wired zero	—
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	—
x4	tp	Thread pointer	—
x5-7	t0-2	Temporaries	Caller
x8	s0/fp	Saved register/frame pointer	Callee
x9	s1	Saved register	Callee
x10-11	a0-1	Function arguments/return values	Caller
x12-17	a2-7	Function arguments	Caller
x18-27	s2-11	Saved registers	Callee
x28-31	t3-6	Temporaries	Caller

Please try to enjoy each instruction equally and not show preference for any over the others