RISC-V RV32I Reference

www.maven-silicon.com

Maven Silicon Confidential

All the presentations, books, documents [hard copies and soft copies] labs and projects [source code] that you are using and developing as part of the training course are the proprietary work of Maven Silicon and it is fully protected under copyright and trade secret laws. You may not view, use, disclose, copy, or distribute the materials or any information except pursuant to a valid written license from Maven Silicon

RISC-V RV32I Instructions Format

31	[30 --------- 25]	[24 -- 21]	20	[19 -- 15]	[14 --- 12]	[11 ------ 8]	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]	imm [10:5]	rs2		rs1	funct3	imm [4:1]	imm [11]	opcode	**B - type**
imm [31:12]						rd		opcode	**U - type**
imm [20]	imm [10:1]		imm [11]	imm [19:12]		rd		opcode	**J - type**

RV32I Base Integer Instructions

Inst	Name	FMT	Opcode	funct3	funct7	Description	Note
add	ADD	R	011_0011	000	000_0000	rd = rs1 + rs2	
sub	SUB	R	011_0011	000	010_0000	rd = rs1 - rs2	
xor	XOR	R	011_0011	100	000_0000	rd = rs1 ^ rs2	
or	OR	R	011_0011	110	000_0000	rd = rs1 | rs2	
and	AND	R	011_0011	111	000_0000	rd = rs1 & rs2	
sll	Shift Left Logical	R	011_0011	001	000_0000	rd = rs1 << rs2	
srl	Shift Right Logical	R	011_0011	101	000_0000	rd = rs1 >>u rs2	
sra	Shift Right Arith*	R	011_0011	101	010_0000	rd = rs1 >>s rs2	
slt	Set Less Than	R	011_0011	010	000_0000	rd = (rs1 s < rs2) ? 1 : 0	
sltu	Set Less Than (U)	R	011_0011	011	000_0000	rd = (rs1 u < rs2) ? 1 : 0	
addi	ADD Immediate	I	001_0011	000		rd = rs1 + IMMI	
xori	XOR Immediate	I	001_0011	100		rd = rs1 ^ IMMI	
ori	OR Immediate	I	001_0011	110		rd = rs1 | IMMI	
andi	AND Immediate	I	001_0011	111		rd = rs1 & IMMI	
slli	Shift Left Logical Imm	I	001_0011	001	imm[11:5] = 000_0000	rd = rs1 << imm[4:0]	IMMI = SXT (imm[11: 0])
srli	Shift Right Logical Imm	I	001_0011	101	imm[11:5] = 000_0000	rd = rs1 >>u imm[4:0]	
srai	Shift Right Arith Imm	I	001_0011	101	imm[11:5] = 010_0000	rd = rs1 >>s imm[4:0]	
slti	Set Less Than Imm	I	001_0011	010		rd = (rs1 s < IMMI) ? 1 : 0	
sltiu	Set Less Than Imm (U)	I	001_0011	011		rd = (rs1 u < IMMI) ? 1 : 0	

RV32I Base Integer Instructions

Inst	Name	FMT	Opcode	funct3	Description	Note
lb	Load Byte	I	000_0011	000	rd = M[rs1 + IMMI]	IMMI = SXT (imm[11:0])
lh	Load Half	I	000_0011	001	rd = M[rs1 + IMMI]	
lw	Load Word	I	000_0011	010	rd = M[rs1 + IMMI]	
lbu	Load Byte (U)	I	000_0011	100	rd = M[rs1 + IMMI]	
lhu	Load Half (U)	I	000_0011	101	rd = M[rs1 + IMMI]	
sb	Store Byte	S	010_0011	000	M[rs1 + IMMS] = rs2	IMMS = SXT (imm[11:0])
sh	Store Half Word	S	010_0011	001	M[rs1 + IMMS] = rs2	
sw	Word	S	010_0011	010	M[rs1 + IMMS] = rs2	
beq	Branch ==	B	110_0011	000	if (rs1 == rs2) PC += IMMB	IMMB = SXT ({imm[12:1],1'b0})
bne	Branch !=	B	110_0011	001	if(rs1 != rs2) PC += IMMB	
blt	Branch <	B	110_0011	100	if(rs1 s< rs2) PC += IMMB	
bge	Branch ≥	B	110_0011	101	if(rs1 >=s rs2) PC += IMMB	
bltu	Branch < (U)	B	110_0011	110	if(rs1 u< rs2) PC +=IMMB	
bgeu	Branch ≥ (U)	B	110_0011	111	if(rs1 >=u rs2) PC += IMMB	
jal	Jump And Link	J	110_1111		rd= PC+4; PC += IMMJ	IMMJ = SXT ({imm[20:1],1'b0})
jalr	Jump And Link Reg	I	110_0111	000	rd= PC+4; PC={(rs1 + IMMI),1'b0}	IMMI = SXT (imm[11:0])
lui	Load Upper Imm	U	011_0111		rd = IMMU	IMMU = { imm[31:12],12'b0 }
auipc	Add Upper Imm to PC	U	001_0111		rd = PC + IMMU	
ecall	Environment Call	I	111_0011	000	Transfer Control to OS	imm[11:0] = 0000_0000_0000
ebreak	Environment Break	I	111_0011	000	Transfer Control to debugger	imm[11:0] = 0000_0000_0001