Risc V Reference Card

Instruction Formats

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[11:6]	imm[5:0]	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[31:12]				rd	opcode	U-type
imm[20|10:1|11|19:12]				rd	opcode	J-type

* This is a special case of the RV64I I-type format used by slli, srli and srai instructions where the lower 6 bits in the immediate are used to determine the shift amount (shamt). If slliw, srliw and sraiw are used it should generate an error if imm[6] $\neq 0$

RV64I Base Instructions

Name	Fmt	Opcode	Funct3	Funct7/ imm[11:5]	Assembly	Description (in C)
Add	R	0110011	000	0000000	add rd, rs1, rs2	rd = rs1 + rs2
Subtract	R	0110011	000	0100000	sub rd, rs1, rs2	rd = rs1 − rs2
AND	R	0110011	111	0000000	and rd, rs1, rs2	rd = rs1 & rs2
OR	R	0110011	110	0000000	or rd, rs1, rs2	rd = rs1 | rs2
XOR	R	0110011	100	0000000	xor rd, rs1, rs2	rd = rs1 ^ rs2
Shift Left Logical	R	0110011	001	0000000	sll rd, rs1, rs2	rd = rs1 $\ll$ rs2
Set Less Than	R	0110011	010	0000000	slt rd, rs1, rs2	rd = (rs1 < rs2)?1:0
Set Less Than (U)*	R	0110011	011	0000000	sltu rd, rs1, rs2	rd = (rs1 < rs2)?1:0
Shift Right Logical	R	0110011	101	0000000	srl rd, rs1, rs2	rd = rs1 $\gg$ rs2
Shift Right Arithmetic†	R	0110011	101	0100000	sra rd, rs1, rs2	rd = rs1 $\gg$ rs2
Add Word	R	0111011	000	0000000	addw rd, rs1, rs2	rd = rs1 + rs2
Subtract Word	R	0111011	000	0100000	subw rd, rs1, rs2	rd = rs1 − rs2
Shift Left Logical Word	R	0111011	001	0000000	sllw rd, rs1, rs2	rd = rs1 $\ll$ rs2
Shift Right Logical Word	R	0111011	101	0000000	srlw rd, rs1, rs2	rd = rs1 $\gg$ rs2
Shift Right Arithmetic Word†	R	0111011	101	0100000	sraw rd, rs1, rs2	rd = rs1 $\gg$ rs2
Add Immediate	I	0010011	000		addi rd, rs1, imm	rd = rs1 + imm
AND Immediate	I	0010011	111		and rd, rs1, imm	rd = rs1 & imm
OR Immediate	I	0010011	110		or rd, rs1, imm	rd = rs1 | imm
XOR Immediate	I	0010011	100		xor rd, rs1, imm	rd = rs1 ^ imm
Shift Left Logical Immediate	I	0010011	001	0000000	slli rd, rs1, shamt	rd = rs1 $\ll$ shamt
Shift Right Logical Immediate	I	0010011	101	0000000	srli rd, rs1, shamt	rd = rs1 $\gg$ shamt
Shift Right Arithmetic Immediate†	I	0010011	101	0100000	srai rd, rs1, shamt	rd = rs1 $\gg$ shamt
Set Less Than Immediate	I	0010011	010		slti rd, rs1, imm	rd = (rs1 < imm)?1:0
Set Less Than Immediate (U)*	I	0010011	011		sltiu rd, rs1, imm	rd = (rs1 < imm)?1:0
Add Immediate Word	I	0011011	000		addiw rd, rs1, imm	rd = rs1 + imm
Shift Left Logical Immediate Word	I	0011011	001	0000000	slliw rd, rs1, shamt	rd = rs1 $\ll$ shamt
Shift Right Logical Immediate Word	I	0011011	101	0000000	srliw rd, rs1, shamt	rd = rs1 $\gg$ shamt
Shift Right Arithmetic Imm Word†	I	0011011	101	0100000	sraiw rd, rs1, shamt	rd = rs1 $\gg$ shamt
Load Byte	I	0000011	000		lb rd, rs1, imm	rd = M[rs1+imm][0:7]
Load Half	I	0000011	001		lh rd, rs1, imm	rd = M[rs1+imm][0:15]
Load Word	I	0000011	010		lw rd, rs1, imm	rd = M[rs1+imm][0:31]
Load Doubleword	I	0000011	011		ld rd, rs1, imm	rd = M[rs1+imm][0:63]
Load Byte (U)*	I	0000011	100		lbu rd, rs1, imm	rd = M[rs1+imm][0:7]
Load Half (U)*	I	0000011	101		lhu rd, rs1, imm	rd = M[rs1+imm][0:15]
Load Word (U)*	I	0000011	110		lwu rd, rs1, imm	rd = M[rs1+imm][0:31]
Store Byte	S	0100011	000		sb rs1, rs2, imm	M[rs1+imm][0:7] = rs2[0:7]
Store Half	S	0100011	001		sh rs1, rs2, imm	M[rs1+imm][0:15] = rs2[0:15]
Store Word	S	0100011	010		sw rs1, rs2, imm	M[rs1+imm][0:31] = rs2[0:31]
Store Doubleword	S	0100011	011		sd rs1, rs2, imm	M[rs1+imm][0:63] = rs2[0:63]
Branch If Equal	B	1100011	000		beq rs1, rs2, imm	if(rs1 == rs2) PC += imm
Branch Not Equal	B	1100011	001		bne rs1, rs2, imm	if(rs1 != rs2) PC += imm
Branch Less Than	B	1100011	100		blt rs1, rs2, imm	if(rs1 < rs2) PC += imm
Branch Greater Than Or Equal	B	1100011	101		bge rs1, rs2, imm	if(rs1 ≥ rs2) PC += imm
Branch Less Than (U)*	B	1100011	110		bltu rs1, rs2, imm	if(rs1 < rs2) PC += imm
Branch Greater Than Or Equal (U)*	B	1100011	111		bgeu rs1, rs2, imm	if(rs1 ≥ rs2) PC += imm
Load Upper Immediate	U	0110111			lui rd, imm	rd = imm $\ll$ 12
Add Upper Immediate To PC	U	0010111			auipc rd, imm	rd = PC + (imm $\ll$ 12)
Jump And Link	J	1101111			jal rd, imm	rd = PC + 4; PC += imm
Jump And Link Register	I	1100111	000		jalr rd, rs1, imm	rd = PC + 4; PC = rs1 + imm

*Assumes values are unsigned integers and zero extends † Fills in with sign bit during right shift and msb (most significant bit) extends

RV64M Standard Extension Instructions

Name	Fmt	Opcode	Funct3	Funct7	Assembly	Description (in C)
Multiply	R	0110011	000	0000001	mul rd, rs1, rs2	rd = (rs1 · rs2)[63:0]
Multiply Upper Half	R	0110011	001	0000001	mulh rd, rs1, rs2	rd = (rs1 · rs2)[127:64]
Multiply Upper Half Sign/Unsigned†	R	0110011	010	0000001	mulhsu rd, rs1, rs2	rd = (rs1 · rs2)[127:64]
Multiply Upper Half (U)*	R	0110011	011	0000001	mulhu rd, rs1, rs2	rd = (rs1 · rs2)[127:64]
Divide	R	0110011	100	0000001	div rd, rs1, rs2	rd = rs1 / rs2
Divide (U)*	R	0110011	101	0000001	divu rd, rs1, rs2	rd = rs1 / rs2
Remainder	R	0110011	110	0000001	rem rd, rs1, rs2	rd = rs1 % rs2
Remainder (U)*	R	0110011	111	0000001	remu rd, rs1, rs2	rd = rs1 % rs2
Multiply Word	R	0111011	000	0000001	mulw rd, rs1, rs2	rd = (rs1 · rs2)[63:0]
Divide Word	R	0111011	100	0000001	divw rd, rs1, rs2	rd = rs1 / rs2
Divide Word (U)*	R	0111011	101	0000001	divuw rd, rs1, rs2	rd = rs1 / rs2
Remainder Word	R	0111011	110	0000001	remw rd, rs1, rs2	rd = rs1 % rs2
Remainder Word (U)*	R	0111011	111	0000001	remuw rd, rs1, rs2	rd = rs1 % rs2

*Assumes values are unsigned integers and zero extends † Multiply with one operand signed and the other unsigned

Bibliography

[1] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978. ISSN 1557-7317.

[2] B. Vinter and K. Skovhede. Synchronous message exchange for hardware designs. *Communicating Process Architectures*, pages 201–212, 2014.