RISC-V Instruction Set Reference

RISC-V Instruction Set Reference

This document is a derivative of “The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 20190608-Base-Ratified”, Editors Andrew Waterman and Krste Asanovi#, RISC-V Foundation, March 2019.

Table of Contents

List of Tables

Chapter 1. Base Instruction Formats

Base Instruction Formats

Table 1.1. R-type

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
funct7							rs2					rs1					funct3			rd					opcode						

Table 1.2. I-type

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:0]												rs1					funct3			rd					opcode						

Table 1.3. S-type

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:5]							rs2					rs1					funct3			imm[4:0]					opcode						

Table 1.4. U-type

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[31:12]																				rd					opcode						

Chapter 2. Integer Register-Immediate Instructions

Integer Register-Immediate Instructions

Table 2.1. Integer Register-Immediate Instructions

Instruction	Description
ADDI	Add Signed Immediate
ANDI	And Signed Immediate
SLTI	Set Less Than Signed Immediate
SLTIU	SLTIU
ORI	Or Signed Immediate
XORI	Exclusive Or Signed Immediate

Chapter 3. Integer Computational Instructions

Integer Computational Instructions

Table 3.1. Integer Computational Instructions

Instruction	Description
LUI	Load Upper Immediate
AUIPC	Add Upper Immediate

Chapter 4. Alphabetical List of RV32I Instructions

alphabetize later

LUI

Table 4.1. LUI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[31:12]																				rd					0	1	1	0	1	1	1

LUI (load upper immediate) is used to build 32-bit constants and uses the U-type format. LUI places the U-immediate value in the top 20 bits of the destination register rd, filling in the lowest 12 bits with zeros.

AUIPC

Table 4.2. AUIPC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[31:12]																				rd					0	0	1	0	1	1	1

AUIPC description AUIPC (add upper immediate to pc) is used to build pc-relative addresses and uses the U-type format. AUIPC forms a 32-bit offset from the 20-bit U-immediate, filling in the lowest 12 bits with zeros, adds this offset to the address of the AUIPC instruction, then places the result in register rd.

The AUIPC instruction supports two-instruction sequences to access arbitrary offsets from the PC for both control-flow transfers and data accesses. The combination of an AUIPC and the 12-bit immediate in a JALR can transfer control to any 32-bit PC-relative address, while an AUIPC plus the 12-bit immediate offset in regular load or store instructions can access any 32-bit PC-relative data address.

The current PC can be obtained by setting the U-immediate to 0. Although a JAL +4 instruction could also be used to obtain the local PC (of the instruction following the JAL), it might cause pipeline breaks in simpler microarchitectures or pollute BTB structures in more complex microarchitectures.

JAL

Table 4.3. JAL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[20|10:1|11|19:12]																				rd					1	1	0	1	1	1	1

JAL rd,label

J label (pseudo)

The jump and link (JAL) instruction uses the J-type format, where the J-immediate encodes a signed offset in multiples of 2 bytes. The offset is sign-extended and added to the address of the jump instruction to

form the jump target address. Jumps can therefore target a ±1 MiB range. JAL stores the address of the instruction following the jump (pc+4) into register rd. The standard software calling convention uses x1 as the return address register and x5 as an alternate link register.

Plain unconditional jumps (assembler pseudoinstruction J) are encoded as a JAL with rd=x0.

JALR

Table 4.4. JALR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:0]												rs1					0	0	0	rd				1	1	0	0	1	1	1

The indirect jump instruction JALR (jump and link register) uses the I-type encoding. The target address is obtained by adding the sign-extended 12-bit I-immediate to the register rs1, then setting the least-significant bit of the result to zero. The address of the instruction following the jump (pc+4) is written to register rd. Register x0 can be used as the destination if the result is not required.

BEQ

Table 4.5. BEQ

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[12|10:5]							rs2					rs1					0	0	0	imm[4:1|11]				1	1	0	0	0	1	1

BEQ rs1,rs2,label

All branch instructions use the B-type instruction format. The 12-bit B-immediate encodes signed offsets in multiples of 2 bytes. The offset is sign-extended and added to the address of the branch instruction to give the target address. The conditional branch range is ±4 KiB.

Branch instructions compare two registers.

BEQ takes the branch if registers rs1 and rs2 are equal.

BNE

Table 4.6. BNE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[12|10:5]							rs2					rs1					0	0	1	imm[4:1|11]				1	1	0	0	0	1	1

BNE rs1,rs2,label

All branch instructions use the B-type instruction format. The 12-bit B-immediate encodes signed offsets in multiples of 2 bytes. The offset is sign-extended and added to the address of the branch instruction to give the target address. The conditional branch range is ±4 KiB.

Branch instructions compare two registers.

BNE takes the branch if registers rs1 and rs2 are not equal.

BLT

Table 4.7. BLT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[12|10:5]							rs2					rs1					1	0	0	imm[4:1|11]				1	1	0	0	0	1	1

BGE

Table 4.8. BGE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[12|10:5]							rs2					rs1					1	0	1	imm[4:1|11]				1	1	0	0	0	1	1

BLTU

Table 4.9. BLTU

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[12|10:5]							rs2					rs1					1	1	0	imm[4:1|11]				1	1	0	0	0	1	1

BGEU

Table 4.10. BGEU

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[12|10:5]							rs2					rs1					1	1	1	imm[4:1|11]				1	1	0	0	0	1	1

LB

Table 4.11. LB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:0]												rs1					0	0	0	rd				0	0	0	0	0	1	1

LH

Table 4.12. LH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:0]												rs1					0	0	1	rd				0	0	0	0	0	1	1

LW

Table 4.13. LW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:0]												rs1					0	1	0	rd					0	0	0	0	0	1	1

LBU

Table 4.14. LBU

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:0]												rs1					1	0	0	rd					0	0	0	0	0	1	1

LHU

Table 4.15. LHU

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:0]												rs1					1	0	1	rd					0	0	0	0	0	1	1

SB

Table 4.16. SB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:5]							rs2					rs1					0	0	0	imm[4:0]					0	1	0	0	0	1	1

SH

Table 4.17. SH

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:5]							rs2					rs1					0	0	1	imm[4:0]					0	1	0	0	0	1	1

SW

Table 4.18. SW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:5]							rs2					rs1					0	1	0	imm[4:0]					0	1	0	0	0	1	1

ADDI

Table 4.19. ADDI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:0]												rs1					0	0	0	rd				0	0	1	0	0	1	1	

SLTI

Table 4.20. SLTI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:0]												rs1					0	1	0	rd				0	0	1	0	0	1	1	

SLTIU

Table 4.21. SLTIU

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:0]												rs1					0	1	1	rd				0	0	1	0	0	1	1	

XORI

Table 4.22. XORI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:0]												rs1					1	0	0	rd				0	0	1	0	0	1	1	

ORI

Table 4.23. ORI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:0]												rs1					1	1	0	rd				0	0	1	0	0	1	1	

ANDI

Table 4.24. ANDI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
imm[11:0]												rs1					1	1	1	rd				0	0	1	0	0	1	1	

SLLI

Table 4.25. SLLI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	shamt					rs1					0	0	1	rd					0	0	1	0	0	1	1

SRLI

Table 4.26. SRLI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	shamt					rs1					1	0	1	rd					0	0	1	0	0	1	1

SRAI

Table 4.27. SRAI

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	shamt					rs1					1	0	1	rd					0	0	1	0	0	1	1

ADD

Table 4.28. ADD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	rs2					rs1					0	0	0	rd					0	1	1	0	0	1	1

SUB

Table 4.29. SUB

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	rs2					rs1					0	0	0	rd					0	1	1	0	0	1	1

SLL

Table 4.30. SLL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	rs2					rs1					0	0	1	rd					0	1	1	0	0	1	1

SLT

Table 4.31. SLT

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	rs2					rs1					0	1	0	rd					0	1	1	0	0	1	1

SLTU

Table 4.32. SLTU

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	rs2					rs1					0	1	1	rd					0	1	1	0	0	1	1

XOR

Table 4.33. XOR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	rs2					rs1					1	0	0	rd					0	1	1	0	0	1	1

SRL

Table 4.34. SRL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	rs2					rs1					1	0	1	rd					0	1	1	0	0	1	1

SRA

Table 4.35. SRA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	rs2					rs1					1	0	1	rd					0	1	1	0	0	1	1

OR

Table 4.36. OR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	rs2					rs1					1	1	0	rd					0	1	1	0	0	1	1

AND

Table 4.37. AND

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	rs2					rs1					1	1	1	rd					0	1	1	0	0	1	1

FENCE

Table 4.38. FENCE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
fm					pred			succ				rs1					0	0	0	rd					0	0	0	1	1	1	1

ECALL

Table 4.39. ECALL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	1

EBREAK

Table 4.40. EBREAK

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	1

RV32I Opcode map

Table 4.41. RV32I Base Instruction Set

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	inst
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[31:12]																				rd					opcode							U-Type
imm[20|10:1|11|19:12]																				rd					opcode							J-Type
imm[31:12]																				rd					0	1	1	0	1	1	1	LUI
imm[31:12]																				rd					0	0	1	0	1	1	1	AUIPC
imm[20|10:1|11|19:12]																				rd					1	1	0	1	1	1	1	JAL
imm[11:0]												rs1					0	0	0	rd					1	1	0	0	1	1	1	JALR

Chapter 5. Alphabetical List of RV32C Instructions

alphabetize later

Illegal instruction

Table 5.1. Illegal instruction

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reserved

Table 5.2. Reserved

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	-											0	0

C.ADDI4SPN

Table 5.3. C.ADDI4SPN (nzuimm !=0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	nzuimm[5:4|9:6|2|3]								rd'			0	0

C.LW

Table 5.4. C.LW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	uimm[5:3]			rs1'			uimm[2|6]		rd'			0	0

C.SW

Table 5.5. C.SW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	uimm[5:3]			rs1'			uimm[2|6]		rs2'			0	0

C.NOP

Table 5.6. C.NOP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

C.ADDI

Table 5.7. C.ADDI (nzimm != 0)(rs1/rd != 0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	nz-imm[5]	rs1/rd					nzimm[4:0]					0	1

C.JAL

Table 5.8. C.JAL

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	imm[11|4|9:8|10|6|7|3:1|5]											0	1

C.JAL is an RV32C-only instruction that performs the same operation as C.J, but additionally writes the address of the instruction following the jump (pc+2) to the link register, x1. C.JAL expands to jal x1, offset[11:1].

C.LI

Table 5.9. C.LI (rd != 0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	imm[5]	rd					imm[4:0]					0	1