RISC-V Instruction Set Summary

32-bit Base Integer ISA + Multiplication and Division (RV32IM)

Made by BME-AUT (10/2024)

Arithmetic Operations

Mnemonic	Instruction	Type	Description
ADD rd, rs1, rs2	Add	R	rd ← rs1 + rs2
SUB rd, rs1, rs2	Subtract	R	rd ← rs1 - rs2
ADDI rd, rs1, imm12	Add immediate	I	rd ← rs1 + imm12
SLT rd, rs1, rs2	Set less than	R	rd ← rs1 < rs2 ? 1 : 0
SLTI rd, rs1, imm12	Set less than immediate	I	rd ← rs1 < imm12 ? 1 : 0
SLTU rd, rs1, rs2	Set less than unsigned	R	rd ← rs1 < rs2 ? 1 : 0
SLTIU rd, rs1, imm12	Set less than immediate unsigned	I	rd ← rs1 < imm12 ? 1 : 0
LUI rd, imm20	Load upper immediate	U	rd ← imm20 << 12
AUIPC rd, imm20	Add upper immediate to PC	U	rd ← pc + imm20 << 12

Logical Operations

Mnemonic	Instruction	Type	Description
AND rd, rs1, rs2	AND	R	rd ← rs1 & rs2
OR rd, rs1, rs2	OR	R	rd ← rs1 | rs2
XOR rd, rs1, rs2	XOR	R	rd ← rs1 ^ rs2
ANDI rd, rs1, imm12	AND immediate	I	rd ← rs1 & imm12
ORI rd, rs1, imm12	OR immediate	I	rd ← rs1 | imm12
XORI rd, rs1, imm12	XOR immediate	I	rd ← rs1 ^ imm12
SLL rd, rs1, rs2	Shift left logical	R	rd ← rs1 << rs2
SRL rd, rs1, rs2	Shift right logical	R	rd ← rs1 >> rs2
SRA rd, rs1, rs2	Shift right arithmetic	R	rd ← rs1 >> rs2
SLLI rd, rs1, shamt	Shift left logical immediate	I	rd ← rs1 << shamt
SRLI rd, rs1, shamt	Shift right logical immediate	I	rd ← rs1 >> shamt
SRAI rd, rs1, shamt	Shift right arithmetic immediate	I	rd ← rs1 >> shamt

Load/Store Operations

Mnemonic	Instruction	Type	Description
LW rd, imm12(rs1)	Load word	I	rd ← mem[rs1 + imm12]
LH rd, imm12(rs1)	Load halfword	I	rd ← mem[rs1 + imm12]
LB rd, imm12(rs1)	Load byte	I	rd ← mem[rs1 + imm12]
LHU rd, imm12(rs1)	Load halfword unsigned	I	rd ← mem[rs1 + imm12]
LBU rd, imm12(rs1)	Load byte unsigned	I	rd ← mem[rs1 + imm12]
SW rs2, imm12(rs1)	Store word	S	rs2(31:0) → mem[rs1 + imm12]
SH rs2, imm12(rs1)	Store halfword	S	rs2(15:0) → mem[rs1 + imm12]
SB rs2, imm12(rs1)	Store byte	S	rs2(7:0) → mem[rs1 + imm12]

Branching

Mnemonic	Instruction	Type	Description
BEQ rs1, rs2, imm12	Branch equal	B	if rs1 = rs2 pc ← pc + imm12
BNE rs1, rs2, imm12	Branch not equal	B	if rs1 ≠ rs2 pc ← pc + imm12
BGE rs1, rs2, imm12	Branch greater than or equal	B	if rs1 ≥ rs2 pc ← pc + imm12
BGEU rs1, rs2, imm12	Branch greater than or equal unsigned	B	if rs1 ≥ rs2 pc ← pc + imm12
BLT rs1, rs2, imm12	Branch less than	B	if rs1 < rs2 pc ← pc + imm12
BLTU rs1, rs2, imm12	Branch less than unsigned	B	if rs1 < rs2 pc ← pc + imm12
JAL rd, imm20	Jump and link	J	rd ← pc + 4; pc ← pc + imm20
JALR rd, imm12(rs1)	Jump and link register	I	rd ← pc + 4; pc ← rs1 + imm12

Integer Multiplication and Division Operations

Mnemonic	Instruction	Type	Description
MUL rd, rs1, rs2	Multiply (LSW)	R	rd ← rs1 * rs2 (signed)
MULH rd, rs1, rs2	Multiply (MSW)	R	rd ← (rs1 * rs2) >> 32 (signed)
MULHSU rd, rs1, rs2	Multiply signed-unsigned (MSW)	R	rd ← (rs1 * rs2) >> 32 (signed)
MULHU rd, rs1, rs2	Multiply unsigned-unsigned (MSW)	R	rd ← (rs1 * rs2) >> 32 (unsigned)
DIV rd, rs1, rs2	Divide signed by signed (RTZ)	R	rd ← rs1 / rs2 (signed)
DIVU rd, rs1, rs2	Divide unsigned by unsigned (RTZ)	R	rd ← rs1 / rs2 (unsigned)
REM rd, rs1, rs2	Remainder signed by signed	R	rd ← rs1 % rs2 (signed)
REMU rd, rs1, rs2	Remainder unsigned by unsigned	R	rd ← rs1 % rs2 (unsigned)

Pseudoinstructions

Mnemonic	Instruction	Base instruction(s)
LA rd, sym	Load address (far)	AUIPC rd, (sym-pc)[31:12] ADDI rd, rd, (sym-pc)[11:0]
LI rd, imm12	Load immediate (near)	ADDI rd, zero, imm12
LI rd, imm	Load immediate (far)	LUI rd, imm[31:12] ADDI rd, rd, imm[11:0]
L{b|h|w} rd, sym	Load global	AUIPC rd, (sym-pc)[31:12] L{b|h|w} rd, (sym-pc)[11:0]
S{b|h|w} rd, sym, rt	Store global	AUIPC rt, (sym-pc)[31:12] S{b|h|w} rd, (sym-pc)[11:0](rt)
MV rd, rs	Copy register	ADDI rd, rs, 0
NOP	No operation	ADDI zero, zero, 0
NOT rd, rs	One's complement	XORI rd, rs, -1
NEG rd, rs	Two's complement	SUB rd, zero, rs
SEXT.{b|h} rd, rs	Sign extend {byte|halfword}	ADDI rd, rs, 0
SEQZ rd, rs	Set if rs = zero	SLTIU rd, rs, 1
SNEZ rd, rs	Set if rs ≠ zero	SLTU rd, zero, rs
SLTZ rd, rs	Set if rs < zero	SLT rd, rs, zero
SGTZ rd, rs	Set if rs > zero	SLT rd, zero, rs
BEQZ rs, offset	Branch if rs = 0	BEQ rs, zero, offset
BNEZ rs, offset	Branch if rs ≠ 0	BNE rs, zero, offset
BLEZ rs, offset	Branch if rs ≤ 0	BGE zero, rs, offset
BGEZ rs, offset	Branch if rs ≥ 0	BGE rs, zero, offset
BLTZ rs, offset	Branch if rs < 0	BLT rs, zero, offset
BGTZ rs, offset	Branch if rs > 0	BLT zero, rs, offset
BGT rs1, rs2, offset	Branch if rs1 > rs2	BLT rs2, rs1, offset
BLE rs1, rs2, offset	Branch if rs1 ≤ rs2	BGE rs2, rs1, offset
BGTU rs1, rs2, offset	Branch if rs1 > rs2 (unsigned)	BLTU rs2, rs1, offset
BLEU rs1, rs2, offset	Branch if rs1 ≤ rs2 (unsigned)	BGEU rs2, rs1, offset
J offset	Jump	JAL zero, offset
TAIL sym	Jump far (tail call far-away subroutine)	AUIPC t1, (sym-pc)[31:12] JALR zero, (sym-pc)[11:0](t1)

Pseudoinstructions

Mnemonic	Instruction	Base instruction(s)
JAL offset	Jump and link (call near subroutine)	JAL ra, offset
JR rs	Jump register	JALR zero, 0(rs)
JALR rs	Jump and link register (call near subroutine)	JAL ra, 0(rs)
CALL sym	Call subroutine (far)	AUIPC t1/ra[1], (sym-pc)[31:12] JALR ra, (sym-pc)[11:0](t1/ra[1]) [1] *t1* or *ra* depending on the RISC-V spec. version
RET	Return from subroutine	JALR zero, 0(ra)

RISC-V Registers & Aliases

Register	ABI alias	Register function	Saver
x0	zero	Zero register	-
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	-
x4	tp	Thread pointer	-
x5 – x7	t0 – t2	Temporary registers	Caller
x8	s0/fp	Saved register/Frame pointer	Callee
x9	s1	Callee saved register	Callee
x10 – x11	a0 – a1	Function argument/return value	Caller
x12 – x17	a2 – a7	Argument registers	Caller
x18 – x27	s2 – s11	Saved registers	Callee
x28 – x31	t3 – t6	Temporary registers	Caller

32-bit Instruction Formats

	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
R	funct	rs2	rs1	funct	rd	opcode
I	immediate[11:0]		rs1	funct	rd	opcode
S	imm[11:5]	rs2	rs1	funct	imm[4:0]	opcode
B	imm[12][10:5]	rs2	rs1	funct	imm[4:1][11]	opcode
U	immediate[31:12]				rd	opcode
J	imm[20][10:1][11]		imm[19:12]		rd	opcode