

# RISC-V Instruction-Set

Erik Engheim <erik.engheim@ma.com>

## Arithmetic Operation

Mnemonic	Instruction	Type	Description
ADD	rd, rs1, rs2	R	rd ← rs1 + rs2
SUB	rd, rs1, rs2	R	rd ← rs1 - rs2
ADDI	rd, rs1, imm12	I	rd ← rs1 + imm12
SLT	rd, rs1, rs2	R	rd ← rs1 < rs2 ? 1 : 0
SLTI	rd, rs1, imm12	I	rd ← rs1 < imm12 ? 1 : 0
SLTU	rd, rs1, rs2	R	rd ← rs1 < rs2 ? 1 : 0
SLTIU	rd, rs1, imm12	I	rd ← rs1 < imm12 ? 1 : 0
LUI	rd, imm20	U	rd ← imm20 << 12
AUIP	rd, imm20	U	rd ← PC + imm20 << 12

## Logical Operations

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

## Load / Store Operations

Mnemonic	Instruction	Type	Description
LD	rd, imm12(rs1)	I	rd ← mem[rs1 + imm12]
LW	rd, imm12(rs1)	I	rd ← mem[rs1 + imm12]
LH	rd, imm12(rs1)	I	rd ← mem[rs1 + imm12]
LB	rd, imm12(rs1)	I	rd ← mem[rs1 + imm12]
LWU	rd, imm12(rs1)	I	rd ← mem[rs1 + imm12]
LHU	rd, imm12(rs1)	I	rd ← mem[rs1 + imm12]
LBU	rd, imm12(rs1)	I	rd ← mem[rs1 + imm12]
SD	rs2, imm12(rs1)	S	rs2 → mem[rs1 + imm12]
SW	rs2, imm12(rs1)	S	rs2(31:0) → mem[rs1 + imm12]
SH	rs2, imm12(rs1)	S	rs2(15:0) → mem[rs1 + imm12]
SB	rs2, imm12(rs1)	S	rs2(7:0) → mem[rs1 + imm12]

## Branching

Mnemonic	Instruction	Type	Description
BEQ	rs1, rs2, imm12	SB	if rs1 = rs2 pc ← pc + imm12
BNE	rs1, rs2, imm12	SB	if rs1 ≠ rs2 pc ← pc + imm12
BGE	rs1, rs2, imm12	SB	if rs1 ≥ rs2 pc ← pc + imm12
BGEU	rs1, rs2, imm12	SB	if rs1 ≥ rs2 pc ← pc + imm12
BLT	rs1, rs2, imm12	SB	if rs1 < rs2 pc ← pc + imm12
BLTU	rs1, rs2, imm12	SB	if rs1 < rs2 pc ← pc + imm12 << 1
JAL	rd, imm20	UJ	rd ← pc + 4 pc ← pc + imm20
JALR	rd, imm12(rs1)	I	rd ← pc + 4 pc ← rs1 + imm12

## Pseudo Instructions

Mnemonic	Instruction	Base instruction(s)
LI	rd, imm12	ADDI rd, zero, imm12
LI	rd, imm	LUI rd, imm[31:12] ADDI rd, rd, imm[11:0]
LA	rd, sym	AUIPC rd, sym[31:12] ADDI rd, rd, sym[11:0]
MV	rd, rs	ADDI rd, rs, 0
NOT	rd, rs	XORI rd, rs, -1
NEG	rd, rs	SUB rd, zero, rs
BGT	rs1, rs2, offset	BLT rs2, rs1, offset
BLE	rs1, rs2, offset	BGE rs2, rs1, offset
BGTU	rs1, rs2, offset	BLTU rs2, rs1, offset
BLEU	rs1, rs2, offset	BGEU rs2, rs1, offset
BEQZ	rs1, offset	BEQ rs1, zero, offset
BNEZ	rs1, offset	BNE rs1, zero, offset
BGEZ	rs1, offset	BGE rs1, zero, offset
BLEZ	rs1, offset	BGE zero, rs1, offset
BGTZ	rs1, offset	BLT zero, rs1, offset
J	offset	JAL zero, offset
CALL	offset12	JALR ra, ra, offset12
CALL	offset	AUIPC ra, offset[31:12] JALR ra, ra, offset[11:0]
RET		JALR zero, 0(ra)
NOP		ADDI zero, zero, 0

## 32-bit instruction format

	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
<b>R</b>	func							rs2				rs1				func		rd				opcode										
<b>I</b>	immediate											rs1				func		rd				opcode										
<b>SB</b>	immediate							rs2				rs1				func		immediate				opcode										
<b>UJ</b>	immediate																	rd				opcode										

## Register File

r0	r1	r2	r3
r4	r5	r6	r7
r8	r9	r10	r11
r12	r13	r14	r15
r16	r17	r18	r19
r20	r21	r22	r23
r24	r25	r26	r27
r28	r29	r30	r31

## Register Aliases

zero	ra	sp	gp
tp	t0	t1	t2
s0/fp	s1	a0	a1
a2	a3	a4	a5
a6	a7	s2	s3
s4	s5	s6	s7
s8	s9	s10	s11
t3	t4	t5	t6

**ra** - return address  
**sp** - stack pointer  
**gp** - global pointer  
**tp** - thread pointer

**t0 - t6** - Temporary registers  
**s0 - s11** - Saved by callee  
**a0 - a7** - Function arguments  
**a0 - a1** - Return value(s)