# **RISC-V Instruction-Set**

Erik Engheim <erik.engheim@ma.com>

# Arithmetic Operation

Mnemonic	Instruction	Туре	Description
ADD rd, rs1, rs	2 Add	R	rd ← rs1 + rs2
SUB rd, rs1, rs	2 Subtract	R	rd ← rs1 - rs2
ADDI rd, rs1, im	n12 Add immediate	I	rd ← rs1 + imm12
SLT rd, rs1, rs	Set less than	R	rd ← rs1 < rs2 ? 1 : 0
SLTI rd, rs1, im	Set less than immediate	I	rd ← rs1 < imm12 ? 1 : 0
SLTU rd, rs1, rs	Set less than unsigned	R	rd ← rs1 < rs2 ? 1 : 0
SLTIU rd, rs1, im	Set less than immediate unsigned	I	rd ← rs1 < imm12 ? 1 : 0
LUI rd, imm20	Load upper immediate	U	rd ← imm20 << 12
AUIP rd, imm20	Add upper immediate to PC	U	rd ← PC + imm20 << 12

# **Logical Operations**

SB

UJ

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	ı	rd ← rs1  << shamt
SRLI rd, rs1, shamt	Shift right logical imm.	ı	rd ← rs1 >> shamt
SRAI rd, rs1, shamt	Shift right arithmetic immediate	I	rd ← rs1 >> shamt

Mnemonic	Instruction	Туре	Description
LD rd, imm12(rs1)	Load doubleword	I	rd ← mem[rs1 + imm12]
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]
LWU rd, imm12(rs1)	Load word unsigned	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]
SD rs2, imm12(rs1)	Store doubleword	S	rs2 → 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) → em[rs1 + imm12]

Load / Store Operations

# Branching

Mnemonic	Instruction	Туре	Description
BEQ rs1, rs2, imm12	Branch equal	SB	if rs1 == rs2 pc ← pc + imm12
BNE rs1, rs2, imm12	Branch not equal	SB	if rs1 != rs2 pc ← pc + imm12
BGE rs1, rs2, imm12	Branch greater than or equal	SB	if rs1 >= rs2 pc ← pc + imm12
BGEU rs1, rs2, imm12	Branch greater than or equal unsigned	SB	if rs1 >= rs2 pc ← pc + imm12
BLT rs1, rs2, imm12	Branch less than	SB	if rs1 < rs2 pc ← pc + imm12
BLTU rs1, rs2, imm12	Branch less than unsigned	SB	if rs1 < rs2 pc ← pc + imm12 << 1
JAL rd, imm20	Jump and link	UJ	rd ← pc + 4 pc ← pc + imm20
JALR rd, imm12(rs1)	Jump and link register	I	rd ← pc + 4 pc ← rs1 + imm12

## 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
func	rs2	rs1	func	rd	opcode
immediate		rs1	func	rd	opcode
immediate	rs2	rs1	func	immediate	opcode
immediate				rd	opcode

#### Pseudo Instructions

Mnemonic	Instruction	Base instruction(s)
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]
LA rd, sym	Load address (far)	AUIPC rd, sym[31:12] ADDI rd, rd, sym[11:0]
MV rd, rs	Copy register	ADDI rd, rs, 0
NOT rd, rs	One's complement	XORI rd, rs, -1
NEG rd, rs	Two's complement	SUB rd, zero, rs
BGT rs, rt, offset	Branch if rs > rt	BLT rt, rs, offset
BLE rs, rt, offset	Branch if rs ≥ rt	BGE rt, rs, offset
BGTU rs, rt, offset	Branch if rs > rt (unsigned)	BLTU rt, rs, offset
BLEU rs, rt, offset	Branch if rs ≥ rt (unsigned)	BGEU rt, rs, offset
BEQZ rs, offset	Branch if rs = 0	BEQ rs, zero, offset
BNEZ rs, offset	Branch if rs ≠ 0	BNE rs, zero, offset
BGEZ rs, offset	Branch if rs ≥ 0	BGE rs, zero, offset
BLEZ rs, offset	Branch if rs < 0	BNE rs, zero, offset
BGTZ rs, offset	Branch if rs > 0	BGE zero, rs, offset
J offset	Unconditional jump	JAL zero, offset
CALL offset12	Call subroutine (near)	JALR ra, ra, offset12
CALL offset	Call subroutine (far)	AUIPC ra, offset[31:12]  JALR ra, ra, offset[11:0]
RET	Return from subroutine	JALR zero, 0(ra)
NOP	No operation	ADDI zero, zero, 0

### Register File

٢0	r1	r2	r3
Ր4	r5	r6	r7
r8	г9	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	га	sp	9P
tр	t0	t1	t2
s0/fp	s1	a0	a1
a2	a3	a4	a5
a6	a7	s2	s3
s <b>4</b>	s5	<b>s</b> 6	s7
<b>s</b> 8	<b>s</b> 9	s10	s11
t3	t4	t5	t6

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

t0 - t6 - Temporary registerss0 - s11 - Saved by calleea0 - 17 - Function argumentsa0 - a1 - Return value(s)