RISC-V Instruction-Set

Erik Engheim <erik.engheim@ma.com>

Arithmetic Operation

	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	Jrd, rs1, imm12	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

SB rs2, imm12(rs1)

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	1	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 imm.	I	rd ← rs1	>> shamt
SRAI rd, rs1, shamt	Shift right arithmetic immediate	I	rd ← rs1	>> shamt

Mnemonic Description Type Instruction Load doubleword rd ← mem[rs1 + imm12] LD rd, imm12(rs1) Load word LW rd, imm12(rs1) $rd \leftarrow mem[rs1 + imm12]$ Load halfword LH rd, imm12(rs1) $rd \leftarrow mem[rs1 + imm12]$ Load byte LB rd, imm12(rs1) $rd \leftarrow mem[rs1 + imm12]$ Load word unsigned LWU rd, imm12(rs1) $rd \leftarrow mem[rs1 + imm12]$ Load halfword $rd \leftarrow mem[rs1 + imm12]$ LHU rd, imm12(rs1) unsigned Load byte unsigned $rd \leftarrow mem[rs1 + imm12]$ LBU rd, imm12(rs1) Store doubleword SD rs2, imm12(rs1) rs2 \rightarrow mem[rs1 + imm12] SW rs2, imm12(rs1) Store word S $rs2(31:0) \rightarrow mem[rs1 + imm12]$ rs2(15:0) → mem[rs1 + imm12] Store halfword SH rs2, imm12(rs1) Store byte

Load / Store Operations

Branching

 $rs2(7:0) \rightarrow mem[rs1 + imm12]$

Mnemonic	Mnemonic Instruction		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		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 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
BEQZ rs1, offset	Branch if rs1 = 0	BEQ rs1, zero, offset
BNEZ rs1, offset	Branch if rs1 ≠ 0	BNE rs1, zero, offset
BGEZ rs1, offset	Branch if rs1 ≥ 0	BGE rs1, zero, offset
BLEZ rs1, offset	Branch if rs1 ≤ 0	BGE zero, rs1, offset
BGTZ rs1, offset	Branch if rs1 > 0	BLT zero, rs1, 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	
r4	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	s 3
s4	s5	s6	s7
s 8	s 9	s10	s11
t3	t4	t5	t6

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

s0 - s11 - Saved by callee **a0 - a7** - Function arguments

t0 - t6 - Temporary registers

a0 - a1 - Return value(s)