

RISC-V Instruction-Set

Erik Engheim <erik.engheim@ma.com>

Arithmetic Operation

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

Logical Operations

Mnemonic	Instruction	Type	Description
AND	rd, rs1, rs2	R	$rd \leftarrow rs1 \& rs2$
OR	rd, rs1, rs2	R	$rd \leftarrow rs1 rs2$
XOR	rd, rs1, rs2	R	$rd \leftarrow rs1 \wedge rs2$
ANDI	rd, rs1, imm12	I	$rd \leftarrow rs1 \& imm12$
ORI	rd, rs1, imm12	I	$rd \leftarrow rs1 imm12$
XORI	rd, rs1, imm12	I	$rd \leftarrow rs1 \wedge imm12$
SLL	rd, rs1, rs2	R	$rd \leftarrow rs1 \ll rs2$
SRL	rd, rs1, rs2	R	$rd \leftarrow rs1 \gg rs2$
SRA	rd, rs1, rs2	R	$rd \leftarrow rs1 \gg rs2$
SLLI	rd, rs1, shamt	I	$rd \leftarrow rs1 \ll shamt$
SRLI	rd, rs1, shamt	I	$rd \leftarrow rs1 \gg shamt$
SRAI	rd, rs1, shamt	I	$rd \leftarrow rs1 \gg shamt$

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	
R	func							rs2					rs1					func			rd					opcode							
I	immediate												rs1					func			rd					opcode							
S	immediate							rs2					rs1					func			immediate					opcode							
SB	immediate							rs2					rs1					func			immediate					opcode							
U	immediate																					rd					opcode						
UJ	immediate																					rd					opcode						

Load / Store Operations

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

Branching

Mnemonic	Instruction	Type	Description
BEQ	rs1, rs2, imm12	SB	if $rs1 == rs2$ $PC \leftarrow PC + imm12$
BNE	rs1, rs2, imm12	SB	if $rs1 != rs2$ $PC \leftarrow PC + imm12$
BGE	rs1, rs2, imm12	SB	if $rs1 \geq rs2$ $PC \leftarrow PC + imm12$
BGEU	rs1, rs2, imm12	SB	if $rs1 \geq rs2$ $PC \leftarrow PC + imm12$
BLT	rs1, rs2, imm12	SB	if $rs1 < rs2$ $PC \leftarrow PC + imm12$
BLTU	rs1, rs2, imm12	SB	if $rs1 < rs2$ $PC \leftarrow PC + imm12 \ll 1$
JAL	rd, imm20	UJ	$rd \leftarrow PC + 4$ $PC \leftarrow PC + imm20$
JALR	rd, imm12(rs1)	I	$rd \leftarrow PC + 4$ $PC \leftarrow rs1 + imm12$

Pseudo Instructions

Mnemonic	Instruction	Base instruction(s)
LI	rd, imm	Load immediate (near) ADDI rd, r0, imm
LI	rd, imm	Load immediate (far) LUI rd, D[31:12] + D[11] ADDI rd, rd, D[11:0]
LA	rd, symbol	Load absolute address (far) AUIPC rd, D[31:12] + D[11] ADDI rd, rd, D[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, r0, rs
BGT	rs, rt, offset	Branch if $rs > rt$ BLT rt, rs, offset
BLE	rs, rt, offset	Branch if $rs \geq rt$ BGE rt, rs, offset
BGTU	rs, rt, offset	Branch if $rs > rt$ (unsigned) BLTU rt, rs, offset
BLEU	rs, rt, offset	Branch if $rs \geq rt$ (unsigned) BGEU rt, rs, offset
BEQZ	rs, offset	Branch if $rs = 0$ BEQ rs, r0, offset
BNEZ	rs, offset	Branch if $rs \neq 0$ BNE rs, r0, offset
BGEZ	rs, offset	Branch if $rs \geq 0$ BGE rs, r0, offset
BLEZ	rs, offset	Branch if $rs < 0$ BNE rs, r0, offset
BGTZ	rs, offset	Branch if $rs > 0$ BGE r0, rs, offset
J	offset	Unconditional jump JAL r0, rs, offset
CALL	offset	Call subroutine JAL ra, offset
RET		Return from subroutine JALR r0, 0(ra)
NOP		No operation ADDI r0, r0, 0

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

- t0 - t6 - Temporary registers
- s0 - s11 - Saved by callee
- a0 - 17 - Function arguments
- a0 - a1 - Return value(s)

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