RISC-V Quick Reference

Registers - Integer

Name	ABI Name	Description
x0	zero	zero constant
x1	ra	return address
x2	sp	stack pointer
x3	gp	global pointer
x4	tp	thread pointer
x5-x7,x28-x31	t0-t2, t3-t6	temperary regs.
x8-x9,x18-x27	s0-s1,s2-s11	saved regs.
x10-17	a0-a7	function args

Registers - Float

ABI Name	Description
ft0-11	temporary
fs0-11	saved
fa0-7	function args

Formats

31 25	24 20	19 15	14 12	11 7	6 0	
func7	r2	r1	func3	rd	opcode	R-Type
imm[11:0]		r1	func3	rd	opcode	I-Type
imm[11:5]	r2	r1	func3	imm[4:0]	opcode	S-Type
imm[12|10 : 5]	r2	r1	func3	imm[4 : 1|11]	opcode	B-Type
imm[31:12]				rd	opcode	U-Type
imm[20|10 : 1|11|19 : 12]				rd	opcode	J-Type

RV32I - Base

Inst.	Description	Type	func7	func3	opcode	format
add	rd = rs1 + rs2	R	000 0000	000	0110011	add rd,rs1,rs2
sub	rd = rs1 - rs2	R	010 0000	000	0110011	sub rd,rs1,rs2
sll	rd = rs1 << rs2	R	000 0000	001	0110011	sll rd,rs1,rs2
slt	rd = (rs1<rs2)?1:0	R	000 0000	010	0110011	slt rd,rs1,rs2
xor	rd = rs1∧rs2	R	000 0000	100	0110011	xor rd,rs1,rs2
srl	rd = rs1 >> rs2	R	000 0000	101	0110011	srl rd,rs1,rs2
sra	rd = rs1 >> rs2	R	010 0000	101	0110011	sra rd,rs1,rs2
or	rd = rs1 | rs2	R	000 0000	110	0110011	or rd,rs1,rs2
and	rd = rs1 & rs2	R	000 0000	111	0110011	and rd,rs1,rs2
addi	rd = rs1 + imm	I		000	0010011	addi rd,rs1,imm
slli	rd = rs1 << imm	I	0x00	001	0010011	slli rd,rs1,imm*
xori	rd = rs1 ∧ imm	I		100	0010011	xori rd,rs1,imm
srli	rd = rs1 >> imm	I	0x00	101	0010011	srli rd,rs1,imm*
srai	rd = rs1 >> imm	I	0x20	101	0010011	srai rd,rs1,imm*
ori	rd = rs1 | imm	I		110	0010011	ori rd,rs1,imm
andi	rd = rs1 & imm	I		111	0010011	andi rd,rs1,imm
lw	rd = M(rs1+imm)	I		010	0000011	lw rd, imm(rs1)
sw	M(rs1+imm) = rs2	S		010	0100011	sw rs2, imm(rs1)
sb	M(rs1+imm) = rs2	S		000	0100011	sw rs2, imm(rs1)
beq	if(rs1 == rs2) PC += SE(imm)	B		000	1100011	beq rs1,rs2,imm
bne	if(rs1 != rs2) PC += SE(imm)	B		001	1100011	bne rs1,rs2,imm
blt	if(rs1 < rs2) PC += SE(imm)	B		100	1100011	blt rs1,rs2,imm
bge	if(rs1 >= rs2) PC += SE(imm)	B		101	1100011	bge rs1,rs2,imm
lui	rd = imm << 12	U		000	0110111	lui rd, imm
auipc	rd = PC + (imm << 12)	U		000	0010111	auipc rd, imm
jal	rd = PC+4; PC += imm	J		000	1101111	jal rd,imm
jalr	rd = PC+4; PC = rs1 + imm	J		000	1100111	jal rd,rs1,imm
ecall		I	0x0	000	1110011	ecall
ebreak		I	0x1	000	1110011	ebreak

Multiplication/Division (M) - Extension

Inst.	Description	Format
mul	rd=rs1*rs2	mul rd, rs1, rs2
mulh	rd=rs1*rs2	mulh rd, rs1, rs2
div	rd=rs1/rs2	div rd, rs1, rs2
divu	rd=rs1/rs2	divu rd, rs1, rs2
rem	rd=rs1%rs2	rem rd, rs1, rs2
remu	rd=rs1%rs2	remu rd, rs1, rs2

Float (F) - Extension

Inst.	Description	Format
flw	rd = M[rs1+imm]	flw rd, imm(rs1)
fsw	M[rs1+imm] = rs2	fsw rs2, imm(rs1)
fadd.s	rd = rs1 + rs2	fadd.s rd, rs1, rs2
fsub.s	rd = rs1 - rs2	fsub.s rd, rs1, rs2
fmul.s	rd = rs1 * rs2	fmul.s rd, rs1, rs2
fdiv.s	rd = rs1 / rs2	fdiv.s rd, rs1, rs2
fsqrt.s	rd = sqrt(rs1)	fsqrt.s rd, rs1
fmin.s	rd = min(rs1,rs2)	fmin.s rd, rs1, rs2
fmax.s	rd = max(rs1,rs2)	fmax.s rd, rs1, rs2
feq.s	rd = (rs1==rs2)? 1 : 0	feq.s rd, rs1, rs2
flt.s	rd = (rs1<rs2)? 1 : 0	flt.s rd, rs1, rs2
fle.s	rd = (rs1<=rs2)? 1 : 0	fle.s rd, rs1, rs2
fcvt.w.s	t0 = (int) ft0	fcvt.w.s t0 ft0
fcvt.w.d	t0 = (int) ft0	fcvt.w.d t0 ft0
fcvt.d.w	ft0 = (float) t0	fcvt.s.w ft0 t0
fcvt.s.w	ft0 = (double) t0	fcvt.d.w ft0 t0

Pseudo Instructions

Inst.	Description	Format
li	load immediate	la rd, imm
la	load address	la rd, symb
nop	no operation	nop
not	1's complement	not rd, rs1
mv	copy register	mv rd,rs1
neg	2's complement	neg rd, rs1
seqz	set if == 0	seqz rd, rs1
snez	set if != 0	snez rd, rs1
sltz	set if < 0	sltz rd, rs1
sgtz	set if > 0	sgtz rd, rs1
j	jump to label	j label
ret	return to ra	ret

Ecall Inputs

Command (a7)	Data/Address (a)	description
1	a0	print int
2	fa0	print float
3	fa0	print double
4	a0 (a)	print string
11	a0	print char

ASCII - Hex Code

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	”	#	$	%	&	’	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	∧	_
6	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	∼	del

Base Conversion

Dec	0	1	2	3	4	5	6	7	8	9	10	11
Hex	00	01	02	03	04	05	06	07	08	09	0A	0B
Bin	00000	00001	00010	00011	00100	00101	00110	00111	01000	01001	01010	01011
Dec	12	13	14	15	16	17	18	19	20	21	22	23
Hex	0C	0D	0E	0F	10	11	12	13	14	15	16	17
Bin	01100	01101	01110	01111	10000	10001	10010	10011	10100	10101	10110	10111
Dec	24	25	26	27	28	29	30	31				
Hex	18	19	1A	1B	1C	1D	1E	1F				
Bin	11000	11001	11010	11011	11100	11101	11110	11111				