RISC-V Cheat Sheet

Name	Syntax	Result	IF
Arithmetic			
add	add rd, rs1, rs2	rd = rs1 + rs2	R
addi	addi rd, rs1, rs2	rd = rs1 + imm	I
sub	sub rd, rs1, rs2	rd = rs1 - rs2	R
lui	lui rd, imm	rd = imm << 12	U
neg	neg rd, rs1	rd = -rs1	(p)
li	li rd, imm	rd = imm	(p)
mv	mv rd, rs1	rd = rs1	(p)
nop	nop	---	(p)
Logical			
and	and rd, rs1, rs2	rd = rs1 & rs2	R
andi	andi rd, rs1, imm	rd = rs1 & imm	I
or	or rd, rs1, rs2	rd = rs1 | rs2	R
ori	ori rd, rs1, imm	rd = rs1 | imm	I
xor	xor rd, rs1, rs2	rd = rs1 ^ rs2	R
xori	xori rd, rs1, imm	rd = rs1 ^ imm	I
not	not rd, rs1	rd = ~rs1	(p)
Shifts			
sll	sll rd rs1, rs2	rd = rs1 << rs2	R
slli	slli rd rs1, imm	rd = rs1 << imm	I
srl	srl rd rs1, rs2	rd = rs1 >> rs2	R
srli	srli rd rs1, imm	rd = rs1 >> imm	I
sra	sra rd rs1, rs2	rd = rs1 >>> rs2	R
srai	srai rd rs1, imm	rd = rs1 >>> imm	I
Compares			
slt	slt rd, rs1, rs2	rd = (rs1 < rs2)	R
slti	slti rd, rs1, imm	rd = (rs1 < imm)	I
sltu	sltu rd, rs1, rs2	rd = (rs1 < rs2)	R
sltiu	sltiu rd, rs1, imm	rd = (rs1 < imm)	I
seqz	seqz rd, rs1	rd = (rs1 == 0)	(p)
snez	snez rd, rs1	rd = (rs1 != 0)	(p)
sltz	sltz rd, rs1	rd = (rs1 < 0)	(p)
sgtz	sgtz rd, rs1	rd = (rs1 > 0)	(p)
Functions			
call	call label	ra = pc + 4; pc = &label	(p)
ret	ret	pc = ra	(p)
Multiplication		(M-Extension)	
mul	mul rd, rs1, rs2	rd = (rs1*rs2)[31:0]	R
mulh	mulh rd, rs1, rs2	rd = (rs1*rs2)[63:32]	R
mulhu	mulhu rd, rs1, rs2	rd = (rs1*rs2)[63:32]	R
mulhsu	mulhsu rd, rs1, rs2	rd = (rs1*rs2)[63:32]	R

(p): Pseudo-Instruction
rd: destination register
rs1: source register 1
rs2: source register 2
imm: immediate value
pc: program counter

Name	Syntax	Result	IF
Branches			
beq	beq rs1, rs2, imm	if(rs1 == rs2): pc += imm	SB
bne	bne rs1, rs2, imm	if(rs1 != rs2): pc += imm	SB
blt	blt rs1, rs2, imm	if(rs1 < rs2): pc += imm	SB
bltu	bltu rs1, rs2, imm	if(rs1 < rs2): pc += imm	SB
bge	bge rs1, rs2, imm	if(rs1 >= rs2): pc += imm	SB
bgeu	bgeu rs1, rs2, imm	if(rs1 >= rs2): pc += imm	SB
auipc	auipc rd, imm	rd = pc + (imm << 12)	U
beqz	beqz rs1, imm	if(rs1 == 0): pc += imm	(p)
bnez	bnez rs1, imm	if(rs1 != 0): pc += imm	(p)
bltz	bltz rs1, imm	if(rs1 < 0): pc += imm	(p)
bgt	bgt rs1, rs2, imm	if(rs1 > rs2): pc += imm	(p)
bgtu	bgtu rs1, rs2, imm	if(rs1 > rs2): pc += imm	(p)
bgtz	bgtz rs1, imm	if(rs1 > 0): pc += imm	(p)
ble	ble rs1, rs2, imm	if(rs1 <= rs2): pc += imm	(p)
bleu	bleu rs1, rs2, imm	if(rs1 <= rs2): pc += imm	(p)
blez	blez rs1, imm	if(rs1 <= 0): pc += imm	(p)
bgez	bgez rs1, imm	if(rs1 >= 0): pc += imm	(p)
Jumps			
jal	jal rd, imm	rd = pc + 4; pc += imm	UJ
jalr	jalr rd, imm	rd = pc + 4; pc += imm	UJ
j	j imm	pc += imm	(p)
Loads			
lw	lw rd, imm(rs1)	rd = mem[rs1 + imm]	I
lh	lh rd, imm(rs1)	rd = mem[rs1 + imm][0:15]	I
lhu	lhu rd, imm(rs1)	rd = mem[rs1 + imm][0:15]	I
lb	lb rd, imm(rs1)	rd = mem[rs1 + imm][0:7]	I
lbu	lbu rd, imm(rs1)	rd = mem[rs1 + imm][0:7]	I
Stores			
sw	sw rs2, imm(rs1)	mem[rs1 + imm] = rs2	S
sh	sh rs2, imm(rs1)	mem[rs1 + imm][0:15] = rs2	S
sb	sb rs2, imm(rs1)	mem[rs1 + imm][0:7] = rs2	S
Division		(M-Extension)	
div	div rd, rs1, rs2	rd = rs1 / rs2	R
rem	rem rd, rs1, rs2	rd = rs1 % rs2	R
divu	divu rd, rs1, rs2	rd = rs1 / rs2	(p)
remu	remu rd, rs1, rs2	rd = rs1 % rs2	(p)

Register	Alias	Description
x0	zero	hard-wired zero
x1	ra	return address
x2	sp	stack pointer
x3	gp	global pointer
x4	tp	thread pointer
x5-7	t0-2	temporaries
x8	s0/fp	saved register/frame pointer
x9	s1	saved register
x10-11	a0-1	function arguments/return values
x12-17	a2-7	function arguments
x18-27	s2-11	saved registers
x28-31	t3-6	temporaries

	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	funct7							rs2					rs1					funct3			rd					opcode						
I	imm[11:0]												rs1					funct3			rd					opcode						
S	imm[11:5]							rs2					rs1					funct3			imm[4:0]					opcode						
SB	imm[12]	imm[10:5]						rs2					rs1					funct3			imm[4:1]				imm[11]	opcode						
U	imm[31:12]																				rd					opcode						
UJ	imm[20]	imm[10:1]										imm[11]	imm[19:12]								rd					opcode						