RV32I Reference Cards

Usage Template		Type	Description	Detailed Description
add	rd, rs1, rs2	R	Add	rd ← rs1 + rs2, pc ← pc+4
addi	rd, rs1, imm	I	Add Immediate	rd ← rs1 + imm_i, pc ← pc+4
and	rd, rs1, rs2	R	And	rd ← rs1 ∧ rs2, pc ← pc+4
andi	rd, rs1, imm	I	And Immediate	rd ← rs1 ∧ imm_i, pc ← pc+4
auipc	rd, imm	U	Add Upper Immediate to PC	rd ← pc + imm_u, pc ← pc+4
beq	rs1, rs2, pcrel_13	B	Branch Equal	pc ← pc + ((rs1==rs2) ? imm_b : 4)
bge	rs1, rs2, pcrel_13	B	Branch Greater or Equal	pc ← pc + ((rs1>=rs2) ? imm_b : 4)
bgeu	rs1, rs2, pcrel_13	B	Branch Greater or Equal Unsigned	pc ← pc + ((rs1>=rs2) ? imm_b : 4)
blt	rs1, rs2, pcrel_13	B	Branch Less Than	pc ← pc + ((rs1<rs2) ? imm_b : 4)
bltu	rs1, rs2, pcrel_13	B	Branch Less Than Unsigned	pc ← pc + ((rs1<rs2) ? imm_b : 4)
bne	rs1, rs2, pcrel_13	B	Branch Not Equal	pc ← pc + ((rs1!=rs2) ? imm_b : 4)
csrrw	rd, csr, rs1	I	Atomic Read/Write	rd ← csr, csr ← rs1, pc ← pc+4
csrrs	rd, csr, rs1	I	Atomic Read and Set	rd ← csr, csr ← csr ∨ rs1, pc ← pc+4
csrrc	rd, csr, rs1	I	Atomic Read and Clear	rd ← csr, csr ← csr ∧ ∼rs1, pc ← pc+4
csrrwi	rd, csr, zimm	I	Atomic Read/Write Immediate	rd ← csr, csr ← zimm, pc ← pc+4
csrrsi	rd, csr, zimm	I	Atomic Read and Set Immediate	rd ← csr, csr ← csr ∨ zimm, pc ← pc+4
csrrci	rd, csr, zimm	I	Atomic Read and Clear Immediate	rd ← csr, csr ← csr ∧ ∼zimm, pc ← pc+4
ecall		I	Environment Call	Transfer Control to Debugger
ebreak		I	Environment Break	Transfer Control to Operating System
jal	rd, pcrel_21	J	Jump And Link	rd ← pc+4, pc ← pc+imm_j
jalr	rd, imm(rs1)	I	Jump And Link Register	rd ← pc+4, pc ← (rs1+imm_i) & ∼1
lb	rd, imm(rs1)	I	Load Byte	rd ← sx(m8(rs1+imm_i)), pc ← pc+4
lbu	rd, imm(rs1)	I	Load Byte Unsigned	rd ← zx(m8(rs1+imm_i)), pc ← pc+4
lh	rd, imm(rs1)	I	Load Halfword	rd ← sx(m16(rs1+imm_i)), pc ← pc+4
lhu	rd, imm(rs1)	I	Load Halfword Unsigned	rd ← zx(m16(rs1+imm_i)), pc ← pc+4
lui	rd, imm	U	Load Upper Immediate	rd ← imm_u, pc ← pc+4
lw	rd, imm(rs1)	I	Load Word	rd ← sx(m32(rs1+imm_i)), pc ← pc+4
or	rd, rs1, rs2	R	Or	rd ← rs1 ∨ rs2, pc ← pc+4
ori	rd, rs1, imm	I	Or Immediate	rd ← rs1 ∨ imm_i, pc ← pc+4
sb	rs2, imm(rs1)	S	Store Byte	m8(rs1+imm_s) ← rs2[7:0], pc ← pc+4
sh	rs2, imm(rs1)	S	Store Halfword	m16(rs1+imm_s) ← rs2[15:0], pc ← pc+4
sll	rd, rs1, rs2	R	Shift Left Logical	rd ← rs1 << (rs2%XLEN), pc ← pc+4
slli	rd, rs1, shamt	I	Shift Left Logical Immediate	rd ← rs1 << shamt_i, pc ← pc+4
slt	rd, rs1, rs2	R	Set Less Than	rd ← (rs1 < rs2) ? 1 : 0, pc ← pc+4
slti	rd, rs1, imm	I	Set Less Than Immediate	rd ← (rs1 < imm_i) ? 1 : 0, pc ← pc+4
sltiu	rd, rs1, imm	I	Set Less Than Immediate Unsigned	rd ← (rs1 < imm_i) ? 1 : 0, pc ← pc+4
sltu	rd, rs1, rs2	R	Set Less Than Unsigned	rd ← (rs1 < rs2) ? 1 : 0, pc ← pc+4
sra	rd, rs1, rs2	R	Shift Right Arithmetic	rd ← rs1 >> (rs2%XLEN), pc ← pc+4
srai	rd, rs1, shamt	I	Shift Right Arithmetic Immediate	rd ← rs1 >> shamt_i, pc ← pc+4
srl	rd, rs1, rs2	R	Shift Right Logical	rd ← rs1 >> (rs2%XLEN), pc ← pc+4
srli	rd, rs1, shamt	I	Shift Right Logical Immediate	rd ← rs1 >> shamt_i, pc ← pc+4
sub	rd, rs1, rs2	R	Subtract	rd ← rs1 - rs2, pc ← pc+4
sw	rs2, imm(rs1)	S	Store Word	m32(rs1+imm_s) ← rs2[31:0], pc ← pc+4
xor	rd, rs1, rs2	R	Exclusive Or	rd ← rs1 ⊕ rs2, pc ← pc+4
xori	rd, rs1, imm	I	Exclusive Or Immediate	rd ← rs1 ⊕ imm_i, pc ← pc+4

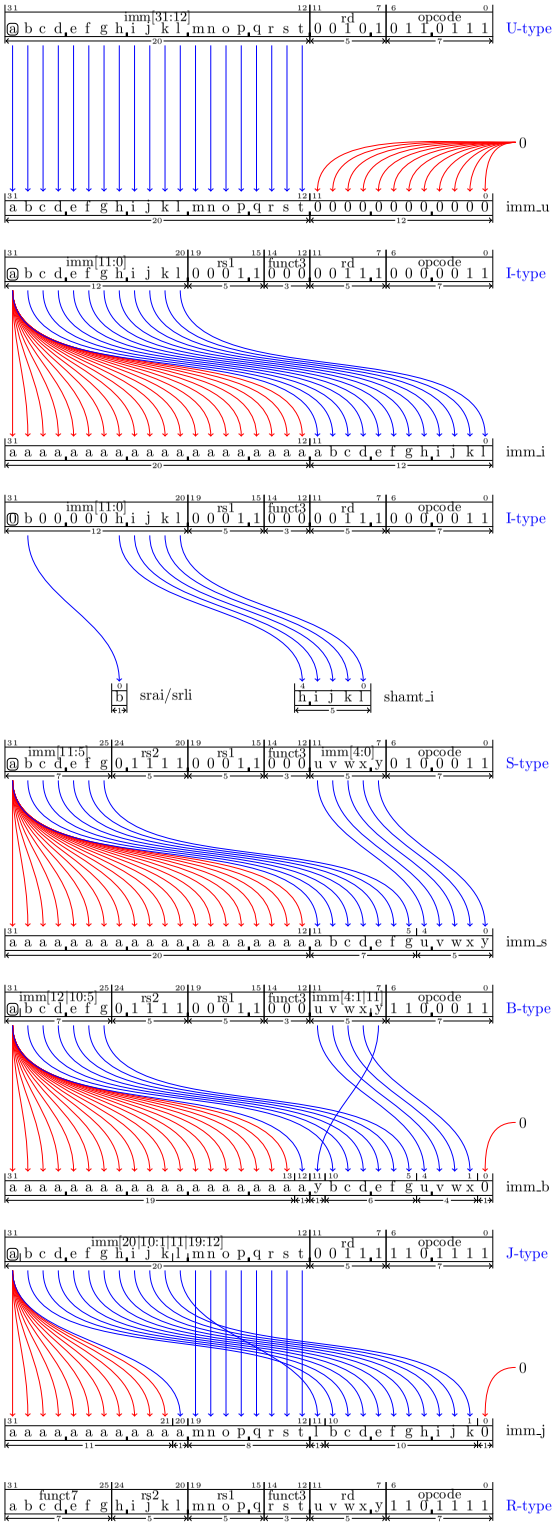
RV32I Base Instruction Set Encoding [1, p. 104]

31 25	24 20	19 15	14 12	11 7	6 0			
imm[31:12]				rd	0 1 1 0 1 1 1	U-type	lui	rd,imm
imm[31:12]				rd	0 0 1 0 1 1 1	U-type	auipc	rd,imm
imm[20|10:1|11|19:12]				rd	1 1 0 1 1 1 1	J-type	jal	rd,pcrel_21
imm[11:0]		rs1	0 0 0	rd	1 1 0 0 1 1 1	I-type	jalr	rd,imm(rs1)
imm[12|10:5]	rs2	rs1	0 0 0	imm[4:1|11]	1 1 0 0 0 1 1	B-type	beq	rs1,rs2,pcrel_13
imm[12|10:5]	rs2	rs1	0 0 1	imm[4:1|11]	1 1 0 0 0 1 1	B-type	bne	rs1,rs2,pcrel_13
imm[12|10:5]	rs2	rs1	1 0 0	imm[4:1|11]	1 1 0 0 0 1 1	B-type	blt	rs1,rs2,pcrel_13
imm[12|10:5]	rs2	rs1	1 0 1	imm[4:1|11]	1 1 0 0 0 1 1	B-type	bge	rs1,rs2,pcrel_13
imm[12|10:5]	rs2	rs1	1 1 0	imm[4:1|11]	1 1 0 0 0 1 1	B-type	bltu	rs1,rs2,pcrel_13
imm[12|10:5]	rs2	rs1	1 1 1	imm[4:1|11]	1 1 0 0 0 1 1	B-type	bgeu	rs1,rs2,pcrel_13
imm[11:0]		rs1	0 0 0	rd	0 0 0 0 0 1 1	I-type	lb	rd,imm(rs1)
imm[11:0]		rs1	0 0 1	rd	0 0 0 0 0 1 1	I-type	lh	rd,imm(rs1)
imm[11:0]		rs1	0 1 0	rd	0 0 0 0 0 1 1	I-type	lw	rd,imm(rs1)
imm[11:0]		rs1	1 0 0	rd	0 0 0 0 0 1 1	I-type	lbu	rd,imm(rs1)
imm[11:0]		rs1	1 0 1	rd	0 0 0 0 0 1 1	I-type	lhu	rd,imm(rs1)
imm[11:5]	rs2	rs1	0 0 0	imm[4:0]	0 1 0 0 0 1 1	S-type	sb	rs2,imm(rs1)
imm[11:5]	rs2	rs1	0 0 1	imm[4:0]	0 1 0 0 0 1 1	S-type	sh	rs2,imm(rs1)
imm[11:5]	rs2	rs1	0 1 0	imm[4:0]	0 1 0 0 0 1 1	S-type	sw	rs2,imm(rs1)
imm[11:0]		rs1	0 0 0	rd	0 0 1 0 0 1 1	I-type	addi	rd,rs1,imm
imm[11:0]		rs1	0 1 0	rd	0 0 1 0 0 1 1	I-type	slti	rd,rs1,imm
imm[11:0]		rs1	0 1 1	rd	0 0 1 0 0 1 1	I-type	sltiu	rd,rs1,imm
imm[11:0]		rs1	1 0 0	rd	0 0 1 0 0 1 1	I-type	xori	rd,rs1,imm
imm[11:0]		rs1	1 1 0	rd	0 0 1 0 0 1 1	I-type	ori	rd,rs1,imm
imm[11:0]		rs1	1 1 1	rd	0 0 1 0 0 1 1	I-type	andi	rd,rs1,imm
0 0 0 0 0 0 0	shamt	rs1	0 0 1	rd	0 0 1 0 0 1 1	I-type	slli	rd,rs1,shamt
0 0 0 0 0 0 0	shamt	rs1	1 0 1	rd	0 0 1 0 0 1 1	I-type	srli	rd,rs1,shamt
0 1 0 0 0 0 0	shamt	rs1	1 0 1	rd	0 0 1 0 0 1 1	I-type	srai	rd,rs1,shamt
0 0 0 0 0 0 0	rs2	rs1	0 0 0	rd	0 1 1 0 0 1 1	R-type	add	rd,rs1,rs2
0 1 0 0 0 0 0	rs2	rs1	0 0 0	rd	0 1 1 0 0 1 1	R-type	sub	rd,rs1,rs2
0 0 0 0 0 0 0	rs2	rs1	0 0 1	rd	0 1 1 0 0 1 1	R-type	sll	rd,rs1,rs2
0 0 0 0 0 0 0	rs2	rs1	0 1 0	rd	0 1 1 0 0 1 1	R-type	slt	rd,rs1,rs2
0 0 0 0 0 0 0	rs2	rs1	0 1 1	rd	0 1 1 0 0 1 1	R-type	sltu	rd,rs1,rs2
0 0 0 0 0 0 0	rs2	rs1	1 0 0	rd	0 1 1 0 0 1 1	R-type	xor	rd,rs1,rs2
0 0 0 0 0 0 0	rs2	rs1	1 0 1	rd	0 1 1 0 0 1 1	R-type	srl	rd,rs1,rs2
0 1 0 0 0 0 0	rs2	rs1	1 0 1	rd	0 1 1 0 0 1 1	R-type	sra	rd,rs1,rs2
0 0 0 0 0 0 0	rs2	rs1	1 1 0	rd	0 1 1 0 0 1 1	R-type	or	rd,rs1,rs2
0 0 0 0 0 0 0	rs2	rs1	1 1 1	rd	0 1 1 0 0 1 1	R-type	and	rd,rs1,rs2
0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0	0 0 0	0 0 0 0 0		1 1 1 0 0 1 1	I-type	ecall	
0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0	0 0 0	0 0 0 0 0		1 1 1 0 0 1 1	I-type	ebreak	
csr[11:0]		rs1	0 0 1	rd	1 1 1 0 0 1 1	I-type	csrrw	rd,csr,rs1
csr[11:0]		rs1	0 1 0	rd	1 1 1 0 0 1 1	I-type	csrrs	rd,csr,rs1
csr[11:0]		rs1	0 1 1	rd	1 1 1 0 0 1 1	I-type	csrrc	rd,csr,rs1
csr[11:0]		zimm[4:0]	1 0 1	rd	1 1 1 0 0 1 1	I-type	csrrwi	rd,csr,zimm
csr[11:0]		zimm[4:0]	1 1 0	rd	1 1 1 0 0 1 1	I-type	csrrsi	rd,csr,zimm
csr[11:0]		zimm[4:0]	1 1 1	rd	1 1 1 0 0 1 1	I-type	csrrci	rd,csr,zimm

https://github.com/johnwinans/rvalp

Instruction		Description	Operation	Type	funct7 31–25	funct3 14–12	opcode 6–0
lui	rd,imm	Load Upper Immediate	rd ← imm_u, pc ← pc+4	U	imm[31:12]		rd 0 1 1 0 1 1 1
auipc	rd,imm	Add Upper Immediate to PC	rd ← pc + imm_u, pc ← pc+4	U	imm[31:12]		rd 0 0 1 0 1 1 1
jal	rd,pcrel_21	Jump And Link	rd ← pc+4, pc ← pc+imm_j	J	imm[20|10:1|11|19:12]		rd 1 1 0 1 1 1 1
jalr	rd,imm(rs1)	Jump And Link Register	rd ← pc+4, pc ← (rs1+imm_i) ∧ ∼1	I	imm[11:0]	rs1 0 0 0	rd 1 1 0 0 1 1 1
beq	rs1,rs2,pcrel_13	Branch Equal	pc ← pc + ((rs1==rs2) ? imm_b : 4)	B	imm[12|10:5] rs2	rs1 0 0 0	imm[4:1|11] 1 1 0 0 0 1 1
bne	rs1,rs2,pcrel_13	Branch Not Equal	pc ← pc + ((rs1!=rs2) ? imm_b : 4)	B	imm[12|10:5] rs2	rs1 0 0 1	imm[4:1|11] 1 1 0 0 0 1 1
blt	rs1,rs2,pcrel_13	Branch Less Than	pc ← pc + ((rs1<rs2) ? imm_b : 4)	B	imm[12|10:5] rs2	rs1 1 0 0	imm[4:1|11] 1 1 0 0 0 1 1
bge	rs1,rs2,pcrel_13	Branch Greater or Equal	pc ← pc + ((rs1>=rs2) ? imm_b : 4)	B	imm[12|10:5] rs2	rs1 1 0 1	imm[4:1|11] 1 1 0 0 0 1 1
bltu	rs1,rs2,pcrel_13	Branch Less Than Unsigned	pc ← pc + ((rs1<rs2) ? imm_b : 4)	B	imm[12|10:5] rs2	rs1 1 1 0	imm[4:1|11] 1 1 0 0 0 1 1
bgeu	rs1,rs2,pcrel_13	Branch Greater or Equal Unsigned	pc ← pc + ((rs1>=rs2) ? imm_b : 4)	B	imm[12|10:5] rs2	rs1 1 1 1	imm[4:1|11] 1 1 0 0 0 1 1
lb	rd,imm(rs1)	Load Byte	rd ← sx(m8(rs1+imm_i)), pc ← pc+4	I	imm[11:0]	rs1 0 0 0	rd 0 0 0 0 0 1 1
lh	rd,imm(rs1)	Load Halfword	rd ← sx(m16(rs1+imm_i)), pc ← pc+4	I	imm[11:0]	rs1 0 0 1	rd 0 0 0 0 0 1 1
lw	rd,imm(rs1)	Load Word	rd ← sx(m32(rs1+imm_i)), pc ← pc+4	I	imm[11:0]	rs1 0 1 0	rd 0 0 0 0 0 1 1
lbu	rd,imm(rs1)	Load Byte Unsigned	rd ← zx(m8(rs1+imm_i)), pc ← pc+4	I	imm[11:0]	rs1 1 0 0	rd 0 0 0 0 0 1 1
lhu	rd,imm(rs1)	Load Halfword Unsigned	rd ← zx(m16(rs1+imm_i)), pc ← pc+4	I	imm[11:0]	rs1 1 0 1	rd 0 0 0 0 0 1 1
sb	rs2,imm(rs1)	Store Byte	m8(rs1+imm_s) ← rs2[7:0], pc ← pc+4	S	imm[11:5] rs2	rs1 0 0 0	imm[4:0] 0 1 0 0 0 1 1
sh	rs2,imm(rs1)	Store Halfword	m16(rs1+imm_s) ← rs2[15:0], pc ← pc+4	S	imm[11:5] rs2	rs1 0 0 1	imm[4:0] 0 1 0 0 0 1 1
sw	rs2,imm(rs1)	Store Word	m32(rs1+imm_s) ← rs2[31:0], pc ← pc+4	S	imm[11:5] rs2	rs1 0 1 0	imm[4:0] 0 1 0 0 0 1 1
addi	rd,rs1,imm	Add Immediate	rd ← rs1 + imm_i, pc ← pc+4	I	imm[11:0]	rs1 0 0 0	rd 0 0 1 0 0 1 1
slti	rd,rs1,imm	Set Less Than Immediate	rd ← (rs1 < imm_i) ? 1 : 0, pc ← pc+4	I	imm[11:0]	rs1 0 1 0	rd 0 0 1 0 0 1 1
sltiu	rd,rs1,imm	Set Less Than Immediate Unsigned	rd ← (rs1 < imm_i) ? 1 : 0, pc ← pc+4	I	imm[11:0]	rs1 0 1 1	rd 0 0 1 0 0 1 1
xori	rd,rs1,imm	Exclusive Or Immediate	rd ← rs1 ⊕ imm_i, pc ← pc+4	I	imm[11:0]	rs1 1 0 0	rd 0 0 1 0 0 1 1
ori	rd,rs1,imm	Or Immediate	rd ← rs1 ∨ imm_i, pc ← pc+4	I	imm[11:0]	rs1 1 1 0	rd 0 0 1 0 0 1 1
andi	rd,rs1,imm	And Immediate	rd ← rs1 ∧ imm_i, pc ← pc+4	I	imm[11:0]	rs1 1 1 1	rd 0 0 1 0 0 1 1
slli	rd,rs1,shamt	Shift Left Logical Immediate	rd ← rs1 << shamt_i, pc ← pc+4	I	0 0 0 0 0 0 0 shamt	rs1 0 0 1	rd 0 0 1 0 0 1 1
srli	rd,rs1,shamt	Shift Right Logical Immediate	rd ← rs1 >> shamt_i, pc ← pc+4	I	0 0 0 0 0 0 0 shamt	rs1 1 0 1	rd 0 0 1 0 0 1 1
srai	rd,rs1,shamt	Shift Right Arithmetic Immediate	rd ← rs1 >> shamt_i, pc ← pc+4	I	0 1 0 0 0 0 0 shamt	rs1 1 0 1	rd 0 0 1 0 0 1 1
add	rd,rs1,rs2	Add	rd ← rs1 + rs2, pc ← pc+4	R	0 0 0 0 0 0 0 rs2	rs1 0 0 0	rd 0 1 1 0 0 1 1
sub	rd,rs1,rs2	Subtract	rd ← rs1 - rs2, pc ← pc+4	R	0 1 0 0 0 0 0 rs2	rs1 0 0 0	rd 0 1 1 0 0 1 1
sll	rd,rs1,rs2	Shift Left Logical	rd ← rs1 << (rs2%XLEN), pc ← pc+4	R	0 0 0 0 0 0 0 rs2	rs1 0 0 1	rd 0 1 1 0 0 1 1
slt	rd,rs1,rs2	Set Less Than	rd ← (rs1 < rs2) ? 1 : 0, pc ← pc+4	R	0 0 0 0 0 0 0 rs2	rs1 0 1 0	rd 0 1 1 0 0 1 1
sltu	rd,rs1,rs2	Set Less Than Unsigned	rd ← (rs1 < rs2) ? 1 : 0, pc ← pc+4	R	0 0 0 0 0 0 0 rs2	rs1 0 1 1	rd 0 1 1 0 0 1 1
xor	rd,rs1,rs2	Exclusive Or	rd ← rs1 ⊕ rs2, pc ← pc+4	R	0 0 0 0 0 0 0 rs2	rs1 1 0 0	rd 0 1 1 0 0 1 1
srl	rd,rs1,rs2	Shift Right Logical	rd ← rs1 >> (rs2%XLEN), pc ← pc+4	R	0 0 0 0 0 0 0 rs2	rs1 1 0 1	rd 0 1 1 0 0 1 1
sra	rd,rs1,rs2	Shift Right Arithmetic	rd ← rs1 >> (rs2%XLEN), pc ← pc+4	R	0 1 0 0 0 0 0 rs2	rs1 1 0 1	rd 0 1 1 0 0 1 1
or	rd,rs1,rs2	Or	rd ← rs1 ∨ rs2, pc ← pc+4	R	0 0 0 0 0 0 0 rs2	rs1 1 1 0	rd 0 1 1 0 0 1 1
and	rd,rs1,rs2	And	rd ← rs1 ∧ rs2, pc ← pc+4	R	0 0 0 0 0 0 0 rs2	rs1 1 1 1	rd 0 1 1 0 0 1 1
ecall		Trap to Debugger		I	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0	0 0 0	0 0 0 0 0 1 1 1 0 0 1 1
ebreak		Trap to Operating System		I	0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0	0 0 0	0 0 0 0 0 1 1 1 0 0 1 1
csrrw	rd,csr,rs1	Atomic Read/Write	rd ← csr, csr ← rs1, pc ← pc+4	I	csr[11:0]	rs1 0 0 1	rd 1 1 1 0 0 1 1
csrrs	rd,csr,rs1	Atomic Read and Set	rd ← csr, csr ← csr ∨ rs1, pc ← pc+4	I	csr[11:0]	rs1 0 1 0	rd 1 1 1 0 0 1 1
csrrc	rd,csr,rs1	Atomic Read and Clear	rd ← csr, csr ← csr ∧ ∼rs1, pc ← pc+4	I	csr[11:0]	rs1 0 1 1	rd 1 1 1 0 0 1 1
csrrwi	rd,csr,zimm	Atomic Read/Write Immediate	rd ← csr, csr ← zimm, pc ← pc+4	I	csr[11:0]	zimm[4:0] 1 0 1	rd 1 1 1 0 0 1 1
csrrsi	rd,csr,zimm	Atomic Read and Set Immediate	rd ← csr, csr ← csr ∨ zimm, pc ← pc+4	I	csr[11:0]	zimm[4:0] 1 1 0	rd 1 1 1 0 0 1 1
csrrci	rd,csr,zimm	Atomic Read and Clear Immediate	rd ← csr, csr ← csr ∧ ∼zimm, pc ← pc+4	I	csr[11:0]	zimm[4:0] 1 1 1	rd 1 1 1 0 0 1 1

RV32I Reference Card



RVALP
RV32I Reference Card

https://github.com/johnwinans/rvalp
v0.18.3-0-g8a08bae