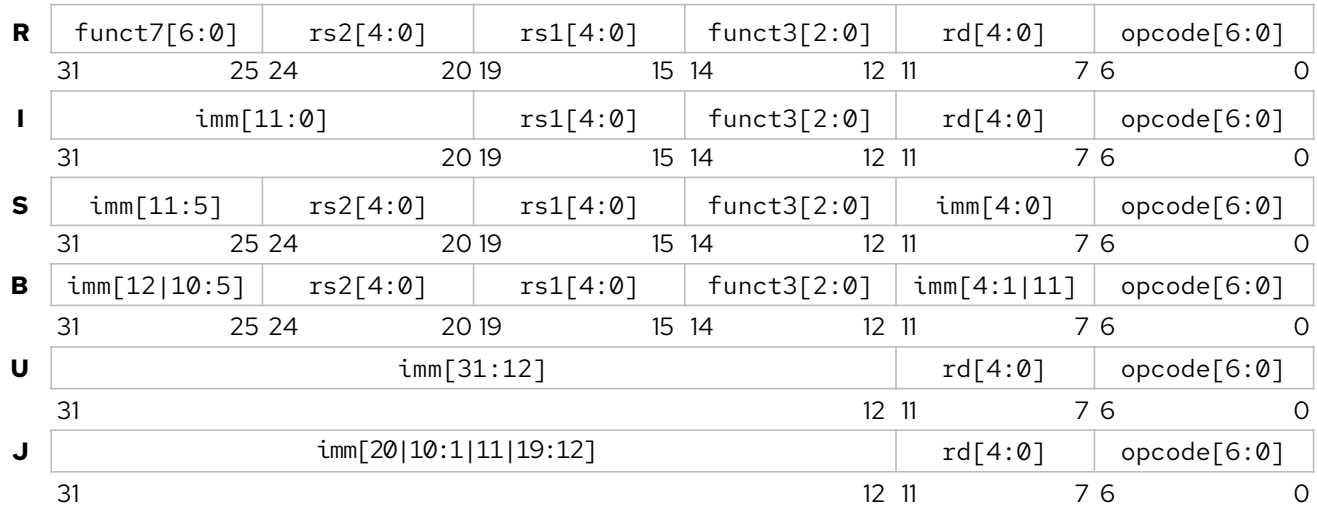
RISC-V RV32I Reference Card
(V2.1)

	inst	operands	name	type	opcode	funct3	funct7	description	asm example
register arithmetic	**add**	**rd, rs1, rs2**	add	R	**011 0011**	**000**	**000 0000**	r[rd] = r[rs1] + r[rs2]	add t3, t1, t2
	sub	**rd, rs1, rs2**	subtract	R	**011 0011**	**000**	**010 0000**	r[rd] = r[rs1] - r[rs2]	sub t3, t1, t2
	sll	**rd, rs1, rs2**	shift left logical	R	**011 0011**	**001**	**000 0000**	r[rd] = r[rs1] << r[rs2][4:0]	sll t3, t1, t2
	slt	**rd, rs1, rs2**	set less than	R	**011 0011**	**010**	**000 0000**	r[rd] = (r[rs1] < r[rs2]) ? 1 : 0	slt t3, t1, t2
	sltu	**rd, rs1, rs2**	set less than unsigned	R	**011 0011**	**011**	**000 0000**	r[rd] = (r[rs1] $<_u$ r[rs2]) ? 1 : 0	sltu t3, t1, t2
	xor	**rd, rs1, rs2**	exclusive or	R	**011 0011**	**100**	**000 0000**	r[rd] = r[rs1] ^ r[rs2]	xor t3, t1, t2
	srl	**rd, rs1, rs2**	shift right logical	R	**011 0011**	**101**	**000 0000**	r[rd] = r[rs1] >> r[rs2][4:0]	srl t3, t1, t2
	sra	**rd, rs1, rs2**	shift right arithmetic	R	**011 0011**	**101**	**010 0000**	r[rd] = r[rs1] >>> r[rs2][4:0]	sra t3, t1, t2
	or	**rd, rs1, rs2**	or	R	**011 0011**	**110**	**000 0000**	r[rd] = r[rs1] | r[rs2]	or t3, t1, t2
	and	**rd, rs1, rs2**	and	R	**011 0011**	**111**	**000 0000**	r[rd] = r[rs1] & r[rs2]	and t3, t1, t2
immediate arithmetic	**addi**	**rd, rs1, imm**	add immediate	I	**001 0011**	**000**		r[rd] = r[rs1] + signext(imm)	addi t2, t1, 5
	slli	**rd, rs1, shamt**	shift left logical imm.	I	**001 0011**	**001**	**000 0000**	r[rd] = r[rs1] << shamt	sll t2, t1, 2
	slti	**rd, rs1, imm**	set less than imm.	I	**001 0011**	**010**		r[rd] = r[rs1] < signext(imm)	slti t2, t1, 5
	sltiu	**rd, rs1, imm**	set less than imm. uns.	I	**001 0011**	**011**		r[rd] = r[rs1] $<_u$ signext(imm)	sltiu t2, t1, 5
	xori	**rd, rs1, imm**	exclusive or imm.	I	**001 0011**	**100**		r[rd] = r[rs1] ^ signext(imm)	xori t2, t1, 5
	srli	**rd, rs1, shamt**	shift right logical imm.	I	**001 0011**	**101**	**000 0000**	r[rd] = r[rs1] >> shamt	srl t2, t1, 2
	srai	**rd, rs1, shamt**	shift right arith. imm.	I	**001 0011**	**101**	**010 0000**	r[rd] = r[rs1] >>> shamt	sra t2, t1, 2
	ori	**rd, rs1, imm**	or immediate	I	**001 0011**	**110**		r[rd] = r[rs1] | signext(imm)	ori t2, t1, 5
	andi	**rd, rs1, imm**	and immediate	I	**001 0011**	**111**		r[rd] = r[rs1] & signext(imm)	andi t2, t1, 5
	lui	**rd, imm**	load upper imm.	U	**011 0111**			r[rd] = {imm, 12'b0}	lui t1, 0xacafe
	auipc	**rd, imm**	add upper imm. to pc	U	**001 0111**			r[rd] = pc + {imm, 12'b0}	auipc t1, 0xacafe
load	**lb**	**rd, offset(rs1)**	load byte	I	**000 0011**	**000**		data = mem[r[rs1] + signext(imm)] r[rd] = signext(data[7:0])	lb t2, 5(t1)
	lh	**rd, offset(rs1)**	load half-word	I	**000 0011**	**001**		data = mem[r[rs1] + signext(imm)] r[rd] = signext(data[15:0])	lh t2, 2(t1)
	lw	**rd, offset(rs1)**	load word	I	**000 0011**	**010**		r[rd] = mem[r[rs1] + signext(imm)]	lw t2, 4(t1)
	lbu	**rd, offset(rs1)**	load byte unsigned	I	**000 0011**	**100**		data = mem[r[rs1] + signext(imm)] r[rd] = zeroext(data[7:0])	lbu t2, 5(t1)
	lhu	**rd, offset(rs1)**	load half-word uns.	I	**000 0011**	**101**		data = mem[r[rs1] + signext(imm)] r[rd] = zeroext(data[15:0])	lhu t2, 2(t1)

***slli, srli, and srai use 5-bit shamt (shift amount) instead of imm field. Format: [31-25]: funct7, [24:20]: shamt, [19-15]: rs1; [14:12]: funct3; [11:7]: rd; [6:0]: opcode.**

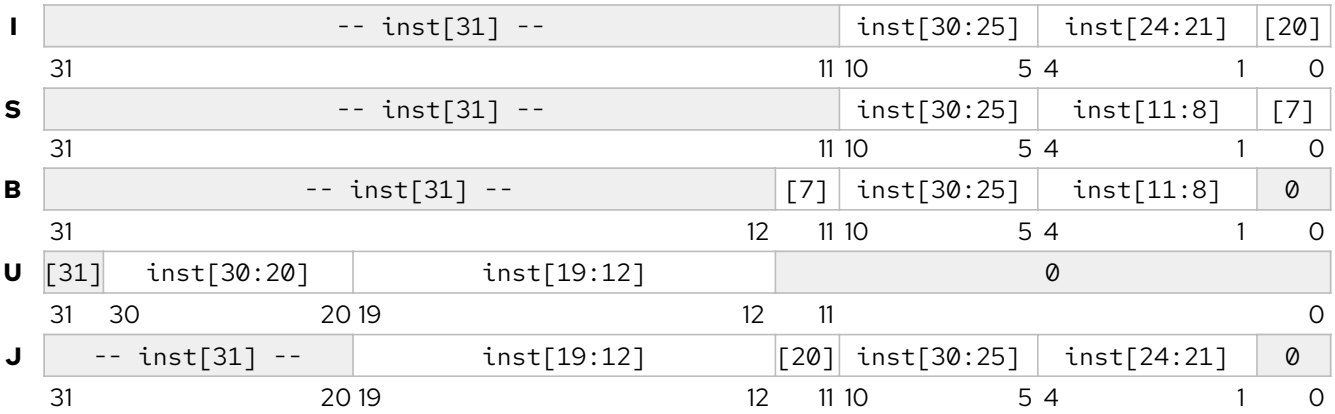
Format of instruction types



RISC-V RV32I Reference Card
(V2.1)

	inst	operands	name	type	opcode	funct3	description	asm example
store	**sb**	**rs2, offset(rs1)**	store byte	S	**010 0011**	**000**	`target = r[rs1] + signext(imm)` `mem[target] = r[rs2][7:0]`	`sb t2, 5(t1)`
	sh	**rs2, offset(rs1)**	store half-word	S	**010 0011**	**001**	`target = r[rs1] + signext(imm)` `mem[target] = r[rs2][15:0]`	`sh t2, 2(t1)`
	sw	**rs2, offset(rs1)**	store word	S	**010 0011**	**010**	`target = r[rs1] + signext(imm)` `mem[target] = r[rs2]`	`sw t2, 4(t1)`
conditional branch	**beq**	**rs1, rs2, offset**	branch if equal	B	**110 0011**	**000**	`if (r[rs1] == r[rs2])` `pc += signext({imm, 1'b0})`	`beq t2, t1, label` *if r1 == r2 go to label*
	bne	**rs1, rs2, offset**	branch if not equal	B	**110 0011**	**001**	`if (r[rs1] != r[rs2])` `pc += signext({imm, 1'b0})`	`bne t2, t1, label` *if r1 != r2 go to label*
	blt	**rs1, rs2, offset**	branch if less than	B	**110 0011**	**100**	`if (r[rs1] < r[rs2])` `pc += signext({imm, 1'b0})`	`blt t2, t1, label` *if r1 < r2 go to label*
	bge	**rs1, rs2, offset**	branch if greater than or equal to	B	**110 0011**	**101**	`if (r[rs1] >= r[rs2])` `pc += signext({imm, 1'b0})`	`bge t2, t1, label` *if r1 >= r2 go to label*
	bltu	**rs1, rs2, offset**	branch if less than unsigned	B	**110 0011**	**110**	`if (r[rs1] <u r[rs2])` `pc += signext({imm, 1'b0})`	`bltu t2, t1, label` *if r1 <u r2 go to label*
	bgeu	**rs1, rs2, offset**	branch if greater than or equal to uns.	B	**110 0011**	**111**	`if (r[rs1] >=u r[rs2])` `pc += signext({imm, 1'b0})`	`bgeu t2, t1, label` *if r1 >=u r2 go to label*
jump	**jal**	**rd, offset**	jump (to address) and link	J	**110 1111**		`r[rd] = pc + 4` `pc += signext({offset, 1'b0})`	`jal x1, label` *save pc in x1,* *go to label*
	jalr	**rd, offset(rs1)**	jump (to) register and link	I	**110 0111**		`r[rd] = pc + 4` `target = r[rs1] + signext(offset)` `pc = {target[31:1], 1'b0}`	`jalr x1, 16(s1)` *save pc in x1,* *go to addr s1 + 16*

Formats of immediate produced by instruction types



Register types

#	name	description	#	name	description	#	name	description	#	name	description
x0	**zero**	constant value 0	x8	**s0/fp**	frame pointer	x16	**a6**	arguments	x24	**s8**	(callee) saved
x1	**ra**	return address	x9	**s1**	(callee) saved	x17	**a7**		x25	**s9**	
x2	**sp**	stack pointer	x10	**a0**	arguments/ return values	x18	**s2**	(callee) saved	x26	**s10**	
x3	**gp**	global pointer	x11	**a1**		x19	**s3**		x27	**s11**	
x4	**tp**	thread pointer	x12	**a2**	arguments	x20	**s4**		x28	**t3**	(caller saved) temporaries
x5	**t0**	(caller saved) temporaries	x13	**a3**		x21	**s5**		x29	**t4**	
x6	**t1**		x14	**a4**		x22	**s6**		x30	**t5**	
x7	**t2**		x15	**a5**		x23	**s7**		x31	**t6**	
caller saved						callee saved					