RISC-V 32i QUICK REFERENCE CARD; FM extensions, RARS implementation [CC-BY-SA downtothewires (via youtube)]

Rev 0.2 Sep 2025

Instruction Formats

any unspecified values should be zero

	31 25	24 20	19 15	14 12	11 7	6 0
R	funct7	rs2	rs1	fnct3	rd	opcode
I	imm[11:0]		rs1	fnct3	rd	opcode
S	imm[11:5]	rs2	rs1	fnct3	imm[4:0]	opcode
B	imm[12|10:5]	rs2	rs1	fnct3	imm[4:1|11]	opcode
U	imm[31:12]				rd	opcode
J	imm[20|10:1|11]		imm[19:12]		rd	opcode

Resulting Immediate Value

	31 12	11 5	4 0
R	*n/a*		
I	SE←inst[31]	inst[31:20]	
S	SE←inst[31]	inst[31:25]	inst[11:7]
B	SE←inst[31]	inst[7|30:25]	[11:8] **0**
U	inst[31:12]	**0 0 0 0 0 0 0 0 0 0 0 0**	
J	SE←inst[31]|inst[19:12]|inst[20|30:25]|[24:21]		**0**

Register File

↑x00↑

sp→ **loc**
s[]
fp↓
fp→ **ra**
CFA **arg↓**

↓xFF↓

Register	Name	Convention	Saver
x0	zero	constant **0**	na
x1	ra	return address	caller
x2	sp	stack pointer	callee
x3	gp	global pointer	na
x4	tp	thread pointer	na
x5-x7	t0-t2	temporary	caller
x8	s0/fp	saved/frame ptr	callee
x9	s1	saved	callee
x10-x11	a0-a1	arg/return val	caller
x12-x17	a2-a7	arguments	caller
x18-x27	s2-s11	saved	callee
x28-x31	t3-t6	temporary	caller

CFA=canonical frame address → sp at call → new fp

ALU

eg: *ADDI t0, t1, 10* *ˢ=unsigned*

funct3	0110011(R)	0010011(I)	operation	description
000	ADD/SUB[1]	ADDI	add / subtract	rd←rs1±{rs2|imm}
001	SLL	SLLI	shift← logical	rd←rs1«{rs2|imm[4:0]}
010	SLT	SLTI	set less than	rd←rs1<{rs2|imm}
011	SLTU	SLTUI	set^{u} (unsigned)	rd←rs1^{u}<{rs2|imm}
100	XOR	XORI	exclusive OR	rd←rs1⊕{rs2|imm}
101	SRL/SRA[1]	SRLI/SRAI[1]	shift→ lgcl/arth	rd←rs1»{rs2|imm[4:0]}
110	OR	ORI	bitwise OR	rd←rs1|{rs2|imm}
111	AND	ANDI	bitwise AND	rd←rs1&{rs2|imm}

[1]*inst[30]:Sub/Add; arith/logical*

L/S

eg: *LW t0, 8(t1)*

funct3	0000011(I)	0100011(S)	data	description
000	LB	SB	[0:7]	load: rd←M[rs1+imm]
001	LH	SH	[0:15]	store: M[rs1+imm]←rs2
010	LW	SW	[0:31]	*B=Byte / H=Half / W=Word*
100	LBU		ZE[0:7]	*far addr use LA pseudo for rs1 or: AUIPC rd, %pcrel_hi(symb)*
101	LHU		ZE[0:15]	*LW rd, %pcrel_lo(symb)(rd)*

Branch 1100011(B)

funct3	PC+=imm if:	
000	BEQ	rs1=rs2
001	BNE	rs1≠rs2
100	BLT	rs1<rs2
101	BGE	rs1≥rs2
110	BLTU	rs1^{u}<rs2
111	BGEU	rs1^{u}≥rs2

System 1110011(I)

func3	name	imm[]	description
000	ECALL	x000	syscall trap
	EBREAK	x001	breakpoint trap
	xRET	*p mode*	ret from trap to P.mode (x=M/H/S/U)
	WFI	x105	wait for interrupt
001[2]	CSRRW	*csr id*	Write rd←CSR←rs1

[2]*(010..111): CSR ops write, set, clear etc*

Op (Fmt) Address Construction

eg: *AUIPC t0, 4096*

0110111(U)	LUI	load upper immediate	rd←imm«12
0010111(U)	AUIPC	Add upper immediate to PC	rd←PC+imm«12

Op (Fmt) Jump

eg: *JALR ra, 0(t1)*

1101111(J)	JAL	jump immediate, and link	rd←PC+4; PC← PC+imm
1100111(I)	JALR	jump register, and link	rd←PC+4; PC←rs1+imm

0001111(I) Fence

pipeline control: enforce pred before succ

f3=000	FENCE 1,1	imm=[0|pred|succ]	[iorw] mem access order
f3=001	FENCE.i	enforce instruction completion order	

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

common RARS syscalls (service# in a7)

a7	description	arguments
1	print integer	a0→integer
4	print string	a0→str address
5	read integer	a0←integer
8	read string	a0←buffer; a1←max
9	allocate heap	a0→bytes to alloc
10	exit, code 0	N/A
11	print character	a0[7:0]→char
12	read character	a0[7:0]←char
50	Y/N Dialog	a0→prompt address

RARS directives

.text	code
.data	data
.byte **.half** **.word**	store b/h/w
.asciz	0-term string
.space	reserve

RARS pipeline

IF	ID	EX	MEM	WB

Control/Status Registers

CSR	name	description
0x300	mstatus	machine status
0x301	misa	ISA/extensions
0x304	mie	interrupts enable
0x305	mtvec	trap vector addr
0x340	mscratch	trap handler temp
0x341	mepc	trap resume addr
0x342	mcause	trap cause
0x343	mtval	trap type
0x344	mip	interrupt pending

** Note RV32i holds 64 bit counters*

Count/Time CSRs*

0xC00	cycle{h} *# cycles*
0xC80	
0xC01	time{h} *# ticks*
0xC81	
0xC02	instret{h} *#retired inst*
0xC82	

Privilege Modes

Machine (M)	top	11
Hypervsr(H)	high	
Supervsr(S)	mid	01
User(U)	low	00

RV32F Float Extension (excerpt)

registers	32 bit f0-f31
ST/LD: OP(F)=0$^{S}/_{I}$00111($^{S}/_{I}$);f3=010	
FSW f1 8(a0); FLW f1 8(a0)	
Calc:OP(F)=1010011(R);f3=round mode	
FADD.S FSUB.S FMUL.S FDIV.S FSQRT.S	
FMIN.A FMAX.S FEQ.S FLT.S FLE.S	
F{N}MADD.S	rd←{-}(rs1×rs2)+rs3
F{N}MSUB.S	rd←{-}(rs1×rs2)-rs3
FSGNJ.S	sign injection (+/-/⊕)
FMV.X.W	bitcopy rf←fpu (fpu←rf)
FCVT.S.W	convert f←i (i←f;s/u)
FCLASS.S	classify fp number

Common Pseudoinstructions/implementations

MV	move	ADDI rd, rs1, 0
SNEZ	set ≠ 0[3]	SLTU rd, x0, rs2
SLTZ	set < 0[3]	SLT rd, rs1, x0
NOT	boolean negate	XORI rd, rs1, -1
NEG	integer negate	SUB rd, x0, rs
Z/S EXT	eg ZEXT.B zero/sign ext	SLLI rd,rs,24 SRAI rd,rs,24
NOP	no-operation	ADDI x0, x0, 0
LI	load immediate	ADDI rd, x0, imm[4]
LA	load address	AUIPC rd, addr[31:12] ADDI rd, addr[11:0]
BGT	branch greater[3]	BLT rs2, rs1, label
BEQZ	branch zero[3]	BEQ rs, x0, label
B	branch always	J (***not*** BNE x0, x0)
J	jump	JAL x0
JAL	jump and link	JAL x1
JR	jump register	JALR x0
RET	return	JALR x0, 0(ra)
CALL	call distant subroutine	AUIPC x1, ofst[31:12] JALR x1,x1,ofst[11:0]
TAIL	tail call	as CALL with ra=***x0***

[3]*similar syntactic sugar for other set/branch*
[4]*for imm<±2KiB*

RV32M Multiply/Divide Extension

OP(F)=0110011(R); funct7=0x01

fn3	op	rs1	rs2	description
000	MUL	S	S	rd←(rs1*rs2) *low* [31:0]
001	MULH	S	S	rd←(rs1*rs2) *high* [63:32]
010	MULHSU	S	U	
011	MULHU	U	U	
10{01}	DIV{U}	S/U	S/U	rd←rs1/rs2
11{01}	REM{U}	S/U	S/U	rd←rs1%rs2