	Group	Format	Name Pseudocode		F E D C B A 9 8 7 6 5 4 3 2 1 0 F E D C		BA9876	5 5 4 3 2 1 0
U	RV 32	LUI rd,imm	Load Upper Immediate (Top 20 bits of rd)rd ← imm		Imm [31:12]		rd	0010111
U UJ		AUIPC rd,offset JAL rd,offset	Add Upper Immediate to PC Jump and Link	$rd \leftarrow pc + offset$ $rd \leftarrow pc + len(inst) \cdots pc \leftarrow pc + off$	Imm [31:12 [20][10:1]	2] [11][19:12]	rd rd	0010111 1101111
Ī		JALR rd,rs1,offset	Jump and Link  Jump and Link Register (return to rs1 when off=0)	$rd \leftarrow pc + len(inst) \cdots pc \leftarrow pc + on$ $rd \leftarrow pc + len(inst) \cdots pc \leftarrow (rs1+off) \land -2$	Imm [11:0]	Rs1 000	rd	1100111
SB		BEQ rs1,rs2,offset	Branch Equal	if rs1 = rs2 then pc ← pc + offset	Imm [12][10:0] Rs2	Rs1 000	lm [11][4:1]	1100011
SB		BNE rs1,rs2,offset	Branch Not Equal	if rs1 $\neq$ rs2 then pc $\leftarrow$ pc + offset	Imm [12][10:1] Rs2	Rs1 001	Im [11] [4:1]	1100011
SB SB		BLT rs1,rs2,offset BGE rs1,rs2,offset	Branch Less Than	if rs1 < rs2 then pc ← pc + offset	Imm [12][10:2] Rs2 Imm [12][10:3] Rs2	Rs1 100 Rs1 101	im [11][4:1] im [11][4:1]	1100011 1100011
SB		BLTU rs1,rs2,offset	Branch Greater than Equal Branch Less Than Unsigned	if rs1 $\geq$ rs2 then pc $\leftarrow$ pc + offset if rs1 $\leq$ rs2 then pc $\leftarrow$ pc + offset	Imm [12][10:4] Rs2	Rs1 101 Rs1 110	Im [11] [4:1]	1100011
SB		BGEU rs1,rs2,offset	Branch Greater than Equal Unsigned	if rs1 ≥ rs2 then pc ← pc + offset	Imm [12][10:5] Rs2	Rs1 111	lm [11][4:1]	1100011
I		LB rd,offset(rs1)	Load Byte	rd ← s8[rs1 + offset]	Imm [11:0]	Rs1 000	rd	0000011
I		LH rd,offset(rs1) LW rd,offset(rs1)	Load Half Load Word	rd ← s16[rs1 + offset] rd ← s32[rs1 + offset]	Imm [11:0] Imm [11:0]	Rs1 001 Rs1 010	rd rd	0000011 0000011
I		LBU rd.offset(rs1)	Load Byte Unsigned	rd ← u8[rs1 + offset]	Imm [11:0]	Rs1 100	rd	0000011
I	RV 32	<b>LHU</b> rd.offset(rs1)	Load Half Unsigned	rd ← u16[rs1 + offset]	Imm [11:0]	Rs1 101	rd	0000011
S S		SB rs2,offset(rs1)	Store Byte	u8[rs1 + offset] ← rs2	Imm [11:5] Rs2		Imm [4:0]	0100011
S S		SH rs2,offset(rs1) SW rs2,offset(rs1)	Store Half Store Word	u16[rs1 + offset] ← rs2 u32[rs1 + offset] ← rs2	Imm [11:5] Rs2 Imm [11:5] Rs2		Imm [4:0] Imm [4:0]	0100011 0100011
Ī		ADDI rd,rs1,imm	Add Immediate	rd ← rs1 + sx(imm)	Imm [11:0]	Rs1 000	rd	0010011
I		SLTI rd,rs1,imm	Set Less Than Immediate	$rd \leftarrow sx(rs1) < sx(imm)$	Imm [11:0]	Rs1 010	rd	0010011
I		SLTIU rd,rs1,imm XORI rd.rs1.imm	Set Less Than Immediate Unsigned Xor Immediate	rd ← ux(rs1) < ux(imm) rd ← ux(rs1) ⊕ ux(imm)	Imm [11:0]	Rs1 011 Rs1 100	rd rd	0010011 0010011
I		ORI rd,rs1,imm	Or Immediate	$rd \leftarrow ux(rs1) \otimes ux(imm)$ $rd \leftarrow ux(rs1) \lor ux(imm)$	Imm [11:0] Imm [11:0]	Rs1 100	ra rd	0010011
Ī		ANDI rd rs1 imm	And Immediate	$rd \leftarrow ux(rs1) \land ux(imm)$	Imm [11:0]	Rs1 111	rd	0010011
R		SLLI rd,rs1,imm	Shift Left Logical Immediate	$rd \leftarrow ux(rs1) \ll ux(imm)$	0000000 shamt	Rs1 001	rd	0010011
R R		SRLI rd,rs1,imm SRAI rd,rs1,imm	Shift Right Logical Immediate Shift Right Arithmetic Immediate	$rd \leftarrow ux(rs1) \gg ux(imm)$ $rd \leftarrow sx(rs1) \gg ux(imm)$	0000000 shamt 0100000 shamt	Rs1 101 Rs1 101	rd rd	0010011 0010011
S		ADD rd,rs1,rs2	Add	$rd \leftarrow sx(rs1) \gg ux(lmm)$ $rd \leftarrow sx(rs1) + sx(rs2)$	0000000 shamt	Rs1 000	ra rd	0110011
S	RV32	SUB rd,rs1,rs2	Subtract	$rd \leftarrow sx(rs1) - sx(rs2)$	0100000 shamt	Rs1 000	rd	0110011
s s		SLL rd,rs1,rs2	Shift Left Logical	$rd \leftarrow ux(rs1) \ll rs2$	0000000 shamt	Rs1 001	rd	0110011
S		SLT rd,rs1,rs2 SLTU rd,rs1,rs2	Set Less Than Set Less Than Unsigned	$rd \leftarrow sx(rs1) < sx(rs2)$ $rd \leftarrow ux(rs1) < ux(rs2)$	0000000 shamt 0000000 shamt	Rs1 010 Rs1 011	rd rd	0110011 0110011
S		<b>XOR</b> rd,rs1,rs2	Xor	$rd \leftarrow ux(rs1) < ux(rs2)$ $rd \leftarrow ux(rs1) \oplus ux(rs2)$	0000000 shamt	Rs1 100	ra rd	0110011
S S S S S S	RV32	SRL rd,rs1,rs2	Shift Right Logical	$rd \leftarrow ux(rs1) \gg rs2$	0000000 shamt	Rs1 101	rd	0110011
S		SRA rd,rs1,rs2	Shift Right Arithmetic	$rd \leftarrow sx(rs1) \gg rs2$	0100000 shamt	Rs1 101	rd	0110011
s S		OR rd,rs1,rs2 AND rd,rs1,rs2	Or And	rd $\leftarrow$ ux(rs1) $\lor$ ux(rs2) rd $\leftarrow$ ux(rs1) $\land$ ux(rs2)	0000000 shamt 0000000 shamt	Rs1 110 Rs1 111	rd rd	0110011 0110011
I		FENCE pred succ	Fence		Fm Pred Suc	Rs1 000	rd	1110011
I		FENCE.I	Fence Instruction		Imm	Rs1 001	rd	0001111
I		LWU rd.offset(rs1) LD rd.offset(rs1)	Load Word Unsigned Load Double	rd $\leftarrow$ u32[rs1 + offset] rd $\leftarrow$ u64[rs1 + offset]	Imm [11:0] Imm [11:0]	Rs1 110 Rs1 011	rd rd	0000011 0000011
		SD rs2,offset(rs1)	Store Double	u64[rs1 + offset] ← rs2	Imm [11:5] Rs2	Rs1 011	Imm [4:0]	0100011
		SLLI rd,rs1,imm	Shift Left Logical Immediate	rd ← ux(rs1) ≪ sx(imm)	000000 shamt	Rs1 001	rd	0010011
		SRLI rd,rs1,imm	Shift Right Logical Immediate	$rd \leftarrow ux(rs1) \gg sx(imm)$	000000 shamt	Rs1 101	rd	0010011
		SRAI rd,rs1,imm ADDIW rd,rs1,imm	Shift Right Arithmetic Immediate	$rd \leftarrow sx(rs1) \gg sx(imm)$ $rd \leftarrow s32(rs1) + imm$	010000 shamt	Rs1 101 Rs1 000	rd rd	0010011 0011011
		SLLIW rd,rs1,imm	Add Immediate Word Shift Left Logical Immediate Word	$rd \leftarrow s32(rs1) + imm$ $rd \leftarrow s32(u32(rs1) \ll imm)$	Imm [11:0] 000000 shamt	Rs1 000 Rs1 001	ra rd	0011011
		SRLIW rd,rs1,imm	Shift Right Logical Immediate Word	rd ← s32(u32(rs1) ≫ imm)	000000 shamt	Rs1 101	rd	0011011
		SRAW rd,rs1,imm	Shift Right Arithmetic Immediate Word	rd ← s32(rs1) ≫ imm	010000 shamt	Rs1 101	rd	0011011
		ADDW rd,rs1,rs2 SUBW rd,rs1,rs2	Add Word Subtract Word	rd ← s32(rs1) + s32(rs2) rd ← s32(rs1) - s32(rs2)	000000 Rs2 010000 Rs2	Rs1 000 Rs1 000	rd rd	0111011 0111011
		SLLW rd,rs1,rs2	Shift Left Logical Word	rd ← s32(u32(rs1) ≪ rs2)	000000 Rs2	Rs1 000	rd	0111011
	RV 641	SRLW rd,rs1,rs2	Shift Right Logical Word	rd ← s32(u32(rs1) ≫ rs2)	000000 Rs2	Rs1 101	rd	0111011
_		SRAW rd,rs1,rs2	Shift Right Arithmetic Word	rd ← s32(rs1) ≫ rs2	010000 Rs2	Rs1 101	rd	0111011
S S		MUL rd,rs1,rs2 MULH rd,rs1,rs2	Multiply Multiply High Signed Signed	$rd \leftarrow ux(rs1) \times ux(rs2)$ $rd \leftarrow (sx(rs1) \times sx(rs2)) \gg xlen$	0000001 Rs2 0000001 Rs2	Rs1 000 Rs1 001	Rd Rd	0110011 0110011
S		MULHSU rd,rs1,rs2	Multiply High Signed Unsigned	$rd \leftarrow (sx(rs1) \times ux(rs2)) \gg xlen$	0000001 Rs2	Rs1 010	Rd	0110011
S		MULHU rd,rs1,rs2	Multiply High Unsigned Unsigned	$rd \leftarrow (ux(rs1) \times ux(rs2)) \gg x len$	0000001 Rs2	Rs1 011	Rd	0110011
S S		DIV rd,rs1,rs2 DIVU rd,rs1,rs2	Divide Signed Divide Unsigned	$rd \leftarrow sx(rs1) \div sx(rs2)$ $rd \leftarrow ux(rs1) \div ux(rs2)$	0000001 Rs2 0000001 Rs2	Rs1 100 Rs1 101	Rd Rd	0110011 0110011
S		REM rd,rs1,rs2	Remainder Signed	$rd \leftarrow ux(rs1) \div ux(rs2)$ $rd \leftarrow sx(rs1) \mod sx(rs2)$	0000001 Rs2	Rs1 101	Rd	0110011
S	RV32M	REMU rd,rs1,rs2	Remainder Unsigned	rd ← ux(rs1) mod ux(rs2)	0000001 Rs2	Rs1 111	Rd	0111011
S S S		MULW rd,rs1,rs2	Multiple Word	$rd \leftarrow u32(rs1) \times u32(rs2)$	0000001 Rs2	Rs1 000	Rd	0111011
S		DIVW rd,rs1,rs2 DIVUW rd,rs1,rs2	Divide Signed Word Divide Unsigned Word	$rd \leftarrow s32(rs1) \div s32(rs2)$ $rd \leftarrow u32(rs1) \div u32(rs2)$	0000001 Rs2 0000001 Rs2	Rs1 100 Rs1 101	Rd Rd	0111011 0111011
S		REMW rd,rs1,rs2	Remainder Signed Word	rd ← s32(rs1) mod s32(rs2)	0000001 Rs2	Rs1 110	Rd	0111011
S		REMUW rd,rs1,rs2	Remainder Unsigned Word	rd ← u32(rs1) mod u32(rs2)	0000001 Rs2	Rs1 111	Rd	0111011
		.2byte .4byte	16-bit comma separated words (unaligned) 32-bit comma separated words (unaligned) http://www.chibiakumas.com/asm					
		.4byte .8byte	64-bit comma separated words (unaligne	neep.// www.criibiaku	us.com/asn	•		
		.half	16-bit comma separated words (naturall					
		.word	32-bit comma separated words (naturall					
		.dword .byte	64-bit comma separated words (naturall					
		.byte .dtpreidword	8-bit comma separated words 64-bit thread local word					
		.dtprelword	32-bit thread local word					
		sleb 128 expression	signed little endian base 128, DWARF					
		.uleb128 expression .asciz "string"	unsigned little endian base 128, DWARF					
		.asciz string	emit string (alias for .string) emit string					
		.incbin "filename"		emit the included file as a binary sequence of octets				
		zero integer	zero bytes					
		_align integer _balign b,[pad_val=0]	align to power of 2 (alias for .p2align)	align to power of 2 (alias for .p2align) byte align				
		.palign b, [pad_val=0] .p2align p2,[pad_val=0],max	align to power of 2					
		.globi symbol_name	emit symbol_name to symbol table (scope GLOBAL)					
		.local symbol_name	emit symbol_name to symbol table (scop					
		.equ name, value .text	constant definition	ako aurrant				
		.text .data	emit text section (if not present) and me emit data section (if not present) and m					
		.rodata	emit .rodata section (if not present) and in					
			Sing it stated 555 son At the processing and make out one					

Directive .bss emit .bss section (if not present) and make current Directive .comm sym\_nam,sz,aln emit common object to .bss section Directive .common sym\_name,sz,aln emit common object to bss section Directive .section sect emit section (if not present, default .text [[.text,.data,.rodata,.bss]] Directive .option opt RISC-V options {rvc,norvc,pic,nopic,push,pop} Directive .macro name arg1 [, argn] begin macro definition ¥argname to substitute Directive .endm end macro definition Directive .file "filename" emit filename FILE LOCAL symbol table Directive .ident "string" accepted for source compatibility Directive \_size symbol, symbol accepted for source compatibility Directive .type symbol, @function accepted for source compatibility Psuedo NOP No operation addi zero,zero,0 Psuedo LI rd, expression Load immediate (several expansions) Psue do LA rd, symbol Load address (several expansions) Psuedo MV rd, rs1 addird, rs, 0 Copy register Psue do NOT rd. rs1 One's complement xori rd, rs, -1 Psue do NEG rd, rs1 Two's complement sub rd, x0, rs Psuedo NEGW rd, rs1 Two's complement Word subw rd, x0, rs Psuedo SEXT.W rd, rs1 Sign extend Word addiw rd, rs, 0 Psue do SEQZ rd, rs1 Set if = zero sltiu rd rs 1 Psue do SNEZ rd, rs1 Set if ≠ zero sltu rd. x0. rs Psue do **SLTZ** rd, rs1 Set if < zero slt rd, rs, x0 Psue do SGTZ rd, rs1 Set if > zero slt rd, x0, rs Psuedo FMV.S frd, frs1 Single-precision move fsgnj.s frd, frs, frs Psue do FAB.S frd, frs1 Single-precision absolute value fsgnix s frd frs frs Psue do FNEG.S frd, frs1 Single-precision negate fsgnjn.s frd, frs, frs Psuedo FMV.D frd, frs1 Double-precision move fsgnj.d frd, frs, frs Psue do FABS.D frd, frs1 Double-precision absolute value fsgnjx.d frd, frs, frs Psue do FNEG.D frd, frs1 Double-precision negate fsgnjn.d frd, frs, frs Psue do BEQZ rs1, offset Branch if = zero bears x0 offset Psue do BNEZ rs1, offset Branch if ≠ zero bne rs. x0. offset Psuedo BLEZ rs1, offset Branch if ≤ zero bge x0, rs, offset Psuedo BGEZ rs1, offset Branch if  $\geq$  zero bge rs, x0, offset Psuedo BLTZ rs1, offset blt rs, x0, offset Branch if < zero Psue do BGTZ rs1, offset Branch if > zero blt x0, rs, offset Psue do BGT rs, rt, offset blt rt rs offset Branch if > Psuedo BLE rs, rt, offset Branch if ≤ bge rt, rs, offset Psuedo BGTU rs, rt, offset Branch if >, unsigned bltu rt, rs, offset Psue do BLEU rs, rt, offset Branch if  $\leq$ , unsigned bltu rt, rs, offset Psuedo **J** offset Jump jal x0, offset Psuedo JR offset jal x1, offset Jump register Psue do RET Return from subroutine jalr x0, x1, 0 Syntax: Imm [12][10:5] = Bits 12 & 10-5 of immediate (other bits in other part)