## (Assembled by jinho.rho)

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	TYPE	MNEMONIC	NAME	Descript	Note
Funct7 (7)	Regitster Source 2 (5) rs2	Regitster Source 1 (5)	Funct3 (3) 0 0 0	Register Destination (5)	Opcode (7)	R-TYPE	ADD	ADD		
		rs1		rd	0 1 1 0 0 1 1		ADD		rd = rs1 + rs2	
	rs2	rs1	0 0 0	rd			SUB	SUB	rd = rs1 - rs2	
0 0 0 0 0 0 0	rs2	rs1	0 0 1	rd	0 1 1 0 0 1 1		SLL	Shift Left Logical	rd = rs1 << rs2	
0 0 0 0 0 0 0	rs2	rs1	1 0 1	rd	0 1 1 0 0 1 1		SRL	Shift Right Logical	rd = rs1 >> rs2	
0 1 0 0 0 0 0	rs2	rs1	1 0 1	rd	0 1 1 0 0 1 1		SRA	Shift Right Arith*	rd = rs1 >>> rs2	msb-extends
0 0 0 0 0 0 0	rs2	rs1	0 1 0	rd	0 1 1 0 0 1 1		SLT	Set Less Than	rd = (rs1 < rs2) ? 1:0	
0 0 0 0 0 0 0	rs2	rs1	0 1 1	rd	0 1 1 0 0 1 1		SLTU	Set Less Than (U)	rd = (rs1 < rs2) ? 1:0	zero-extends
0 0 0 0 0 0 0	rs2	rs1	1 0 0	rd	0 1 1 0 0 1 1		XOR	XOR	rd = rs1 ^ rs2	
0 0 0 0 0 0 0	rs2	rs1	1 1 0	rd	0 1 1 0 0 1 1		OR	OR	rd = rs1   rs2	
0 0 0 0 0 0 0	rs2	rs1	1 1 1	rd	0 1 1 0 0 1 1		AND	AND	rd = rs1 & rs2	
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					
immediate (7)	shift amount (5)	Regitster Source 1 (5)	Funct3 (3)	Register Destination (5)	Opcode (7)	TYPE	MNEMONIC	NAME	Descript	Note
imm[11:0]	5 (b)	rs1	0 0 0	rd	0 0 0 0 0 1 1		LB	Load Byte	rd = M[rs1+imm][0:7]	
imm[11:0]		rs1	0 0 1	rd	0 0 0 0 0 1 1		LH	Load Half	rd = M[rs1+imm][0:15]	
imm[11:0]		rs1	0 1 0	rd	0 0 0 0 0 1 1	L-TYPE	LW	Load Word	rd = M[rs1+imm][0:31]	
imm[11:0]		rs1	1 0 0	rd	0 0 0 0 0 1 1		LBU	Load Byte (U)	rd = M[rs1+imm][0:7]	zero-extends
imm[11:0]		rs1	1 0 1	rd	0 0 0 0 0 1 1		LHU	Load Half (U)	rd = M[rs1+imm][0:15]	zero-extends
imm[11:0]		rs1	0 0 0	rd	0 0 1 0 0 1 1		ADDI	ADD Immediate	rd = rs1 + imm	
imm[11:0]		rs1	0 1 0	rd	0 0 1 0 0 1 1		SLTI	Set Less Than Imm	rd = (rs1 < imm) ? 1 : 0	
imm[11:0]		rs1	0 1 1	rd	0 0 1 0 0 1 1		SLTIU	Set Less Than Imm (U)	rd = (rs1 < imm) ? 1 : 0	
imm[11:0]		rs1	1 0 0	rd	0 0 1 0 0 1 1		XORI	XOR Immediate	rd = rs1 ^ imm	
imm[11:0]		rs1	1 1 0	rd	0 0 1 0 0 1 1	I-TYPE	ORI	OR Immediate	rd = rs1   imm	
imm[11:0]		rs1	1 1 1	rd	0 0 1 0 0 1 1		ANDI	AND Immediate	rd = rs1 & imm	
0 0 0 0 0 0 0	shamt	rs1	0 0 1	rd	0 0 1 0 0 1 1		SLLI	Shift Left Logical Imm	rd = rs1 << shamt[0:4]	
0 0 0 0 0 0 0	shamt	rs1	1 0 1	rd	0 0 1 0 0 1 1		SRLI	Shift Right Logical Imm	rd = rs1 >> shamt[0:4]	
0 1 0 0 0 0 0	shamt	rs1	1 0 1	rd	0 0 1 0 0 1 1		SRAI	Shift Right Arith Imm	rd = rs1 >>> shamt[0:4]	
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	TYPE	MNEMONIC	NAME	Descript	Note
immediate (7)	Regitster Source 2 (5)	Regitster Source 1 (5)	Funct3 (3)	immediate (5)	Opcode (7)		milemonie	TWITTE	Descript	11010
imm[11:5]	rs2	rs1	0 0 0	imm[4:0]	0 1 0 0 0 1 1		SB	Store Byte	M[rs1+imm][0:7] = rs2[0:7]	
imm[11:5]	rs2	rs1	0 0 1	imm[4:0]	0 1 0 0 0 1 1	S-TYPE	SH	Store Half	M[rs1+imm][0:15] = rs2[0:15]	
imm[11:5]	rs2	rs1	0 1 0	imm[4:0]	0 1 0 0 0 1 1		SW	Store Word	M[rs1+imm][0:31] = rs2[0:31]	
									T	
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	TYPE	MNEMONIC	NAME	Descript	Note
immediate (7)	Regitster Source 2 (5)	Regitster Source 1 (5)	Funct3 (3)	immediate (5)	Opcode (7)		250	D I	·	
imm[12][10:5]	rs2	rs1	0 0 0	imm[4:1][11]	1 1 0 0 0 1 1		BEQ	Branch ==	if(rs1 == rs2) PC += imm	
imm[12][10:5]	rs2	rs1	0 0 1	imm[4:1][11]	1 1 0 0 0 1 1	B-TYPE	BNE	Branch !=	if(rs1 != rs2) PC += imm	
imm[12][10:5]	rs2 rs2	rs1	1 0 0	imm[4:1][11]	1 1 0 0 0 1 1		BLT	Branch <	if(rs1 < rs2) PC += imm	
imm[12][10:5]	rs2	rs1 rs1	1 0 1	imm[4:1][11]	1 1 0 0 0 1 1		BGE BLTU	Branch ≥	if(rs1 >= rs2) PC += imm	
imm[12][10:5] imm[12][10:5]	rs2	rs1	1 1 0	imm[4:1][11] imm[4:1][11]	1 1 0 0 0 1 1		BGEU	Branch < (U)  Branch ≥ (U)	if(rs1 < rs2) PC += imm if(rs1 >= rs2) PC += imm	
11111[12][10.5]	132	131		mmi[ <del>4</del> .1][11]			DGEU	DIGHCH ≥ (U)	11(131 /- 132) FC += 1111111	I
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	TYPE	MNEMONIC	NAME	Descript	Note
immediate (20)			Register Destination (5)	Opcode (7)				·		
imm[31:12]			rd	0 1 1 0 1 1 1	LU-TYPE	LUI	Load Upper Imm	rd = imm << 12		
imm[31:12]			rd	0 0 1 0 1 1 1	AU-TYPE	AUIPC	Add Upper Imm to PC	rd = PC + (imm << 12)		
imm[20][10:1][11][19:12]			rd	1 1 0 1 1 1 1	J-TYPE	JAL	Jump And Link	rd = PC + 4; PC+= imm		
imm[11:0]		rs1	0 0 0	rd	1 1 0 0 1 1 1	JL-TYPE	JALR	Jump And Link Reg	rd = PC + 4; $PC = rs1 + imm$	