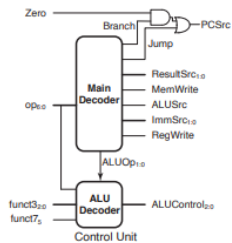


31:2	5	24:20	19:1	5	14:12	11:7	6:0	
funct7	rs2	rs1	funct3	rd	op			R-Type
imm11,12								I-Type
imm11,13	rs2	rs1	funct3	imm4,0	op			S-Type
imm12,10,5	rs2	rs1	funct3	imm4,1,1	op			B-Type
imm10,12				rd	op			U-Type
imm20,10:1,11,10,12				rd	op			J-Type
fs3	funct2	fs2	fs1	funct3	fd	op		R4-Type
5 bits	2 bits	5 bits	5 bits	5 bits	5 bits	7 bits		

RISC-V 32-bit instruction formats  
R-type: register-type, I-type: immediate,  
S-type: store, B-type: branch,  
U-type: upper immediate, J-type: jump  
R4-type: 4 reg floating point operation

Name	Reg. No	Use
zero	x0	Constant value 0
ra	x1	Return address
sp	x2	Stack pointer
gp	x3	Global pointer
tp	x4	Thread pointer
x0-2	x5-7	Temporary registers
x0fp	x8	Saved reg/Frame pointer
s1	x9	Saved register
a0-1	x10-11	Function arg/Return values
a2-7	x12-17	Function arguments
s2-11	x18-27	Saved registers
t3-6	x28-31	Temporary registers



ALU Decoder truth table

ALUOp	funct3	op5	funct7	op5	funct7	ALUControl	Instruction
00	00	00	00	00	00	00	lw, sw
01	x	x	001	(sub)act	beq	000	beq
10	000	00,01,10	000	(add)	add	000	add
	000	11	001	(sub)act	sub	000	sub
	010	x	101	(set less than)	slt	000	slt
	110	x	011	(or)	or	000	or
	111	x	010	(and)	and	000	and
10	101	110	110	110	110	110	sra

op	funct3	funct7	Type	Instruction	Description	Operation
0000011 (3)	000	-	I	lb rd, imm(rs1)	load byte	rd = SignExt(Address[rs1], 7)
0000011 (3)	001	-	I	lh rd, imm(rs1)	load half	rd = SignExt(Address[rs1], 15)
0000011 (3)	010	-	I	lw rd, imm(rs1)	load word	rd = Address[rs1]
0000011 (3)	100	-	I	lbu rd, imm(rs1)	load byte unsigned	rd = ZeroExt(Address[rs1], 7)
0000011 (3)	101	-	I	lhu rd, imm(rs1)	load half unsigned	rd = ZeroExt(Address[rs1], 15)
0010011 (19)	000	-	I	addi rd, rs1, imm	add immediate	rd = rs1 + SignExt(imm)
0010011 (19)	001	0000000	I	slli rd, rs1, uimm	shift left logical immediate	rd = rs1 << uimm
0010011 (19)	010	-	I	slti rd, rs1, imm	set less than immediate	rd = (rs1 < SignExt(imm))
0010011 (19)	011	-	I	sltiu rd, rs1, imm	set less than imm. unsigned	rd = (rs1 < SignExt(imm))
0010011 (19)	100	-	I	xori rd, rs1, imm	xor immediate	rd = rs1 ^ SignExt(imm)
0010011 (19)	101	0000000	I	srai rd, rs1, uimm	shift right logical immediate	rd = rs1 >> uimm
0010011 (19)	101	0100000	I	srai rd, rs1, uimm	shift right arithmetic imm.	rd = rs1 >> uimm
0010011 (19)	110	-	I	ori rd, rs1, imm	or immediate	rd = rs1   SignExt(imm)
0010011 (19)	111	-	I	andi rd, rs1, imm	and immediate	rd = rs1 & SignExt(imm)
0010111 (23)	-	-	U	auipc rd, upimm	add upper immediate to PC	rd = (upimm, 12'b0) + PC
0100011 (35)	000	-	S	sb rs2, imm(rs1)	store byte	(Address[rs1] = rs2, 7)
0100011 (35)	001	-	S	shw rs2, imm(rs1)	store half	(Address[rs1] = rs2, 15)
0100011 (35)	010	-	S	sw rs2, imm(rs1)	store word	(Address[rs1] = rs2, 31)
0100011 (51)	000	0000000	R	add rd, rs1, rs2	add	rd = rs1 + rs2
0100011 (51)	000	0100000	R	sub rd, rs1, rs2	sub	rd = rs1 - rs2
0100011 (51)	001	0000000	R	sll rd, rs1, rs2	shift left logical	rd = rs1 << rs2 40
0100011 (51)	010	0000000	R	slt rd, rs1, rs2	set less than	rd = (rs1 < rs2)
0100011 (51)	011	0000000	R	sltu rd, rs1, rs2	set less than unsigned	rd = (rs1 < rs2)
0100011 (51)	100	0000000	R	xor rd, rs1, rs2	xor	rd = rs1 ^ rs2
0100011 (51)	101	0000000	R	srl rd, rs1, rs2	shift right logical	rd = rs1 >> rs2 40
0100011 (51)	101	0100000	R	sra rd, rs1, rs2	shift right arithmetic	rd = rs1 >> rs2 40
0100011 (51)	110	0000000	R	or rd, rs1, rs2	or	rd = rs1   rs2
0100011 (51)	111	0000000	R	and rd, rs1, rs2	and	rd = rs1 & rs2
0101011 (55)	-	-	U	lui rd, upimm	load upper immediate	rd = (upimm, 12'b0)
1100011 (99)	000	-	B	beq rs1, rs2, label	branch if =	if (rs1 == rs2) PC = BTA
1100011 (99)	001	-	B	bne rs1, rs2, label	branch if ≠	if (rs1 ≠ rs2) PC = BTA
1100011 (99)	100	-	B	blt rs1, rs2, label	branch if <	if (rs1 < rs2) PC = BTA
1100011 (99)	101	-	B	bge rs1, rs2, label	branch if ≥	if (rs1 ≥ rs2) PC = BTA
1100011 (99)	110	-	B	bltu rs1, rs2, label	branch if < unsigned	if (rs1 < rs2) PC = BTA
1100011 (99)	111	-	B	bgeu rs1, rs2, label	branch if ≥ unsigned	if (rs1 ≥ rs2) PC = BTA
1100111 (103)	000	-	J	jalr rd, rs1, imm	jump and link register	PC = rs1 + SignExt(imm), rd = PC + 4
1101111 (111)	-	-	J	jal rd, label	jump and link	PC = JTA, rd = PC + 4

\*Encoded in instr[7:25], the upper seven bits of the immediate field

Instruction	Opcode	RegWrite	ImmSrc	ALUSrc	MemWrite	ResultSrc	Branch	ALUOp	Jump
lw	0000011	1	00	1	0	01	0	00	0
sw	0100011	0	01	1	1	xx	0	00	0
R-type	0110011	1	xx	0	0	00	0	10	0
beq	1100011	0	10	0	0	xx	1	01	0
I-type ALU	0010011	1	00	1	0	00	0	10	0
jal	1101111	1	11	x	0	10	0	xx	1

ALUControl02	Function
000	Add
001	Subtract
010	AND
011	OR
101	SLT
110	sra

Soru 2 de sra tasarımı için alu decoder ve alu function listte mavi yazıların olduğu şekilde ekleme yapmayı tasarladım. Block şemasında bir değişiklik yapmadım.