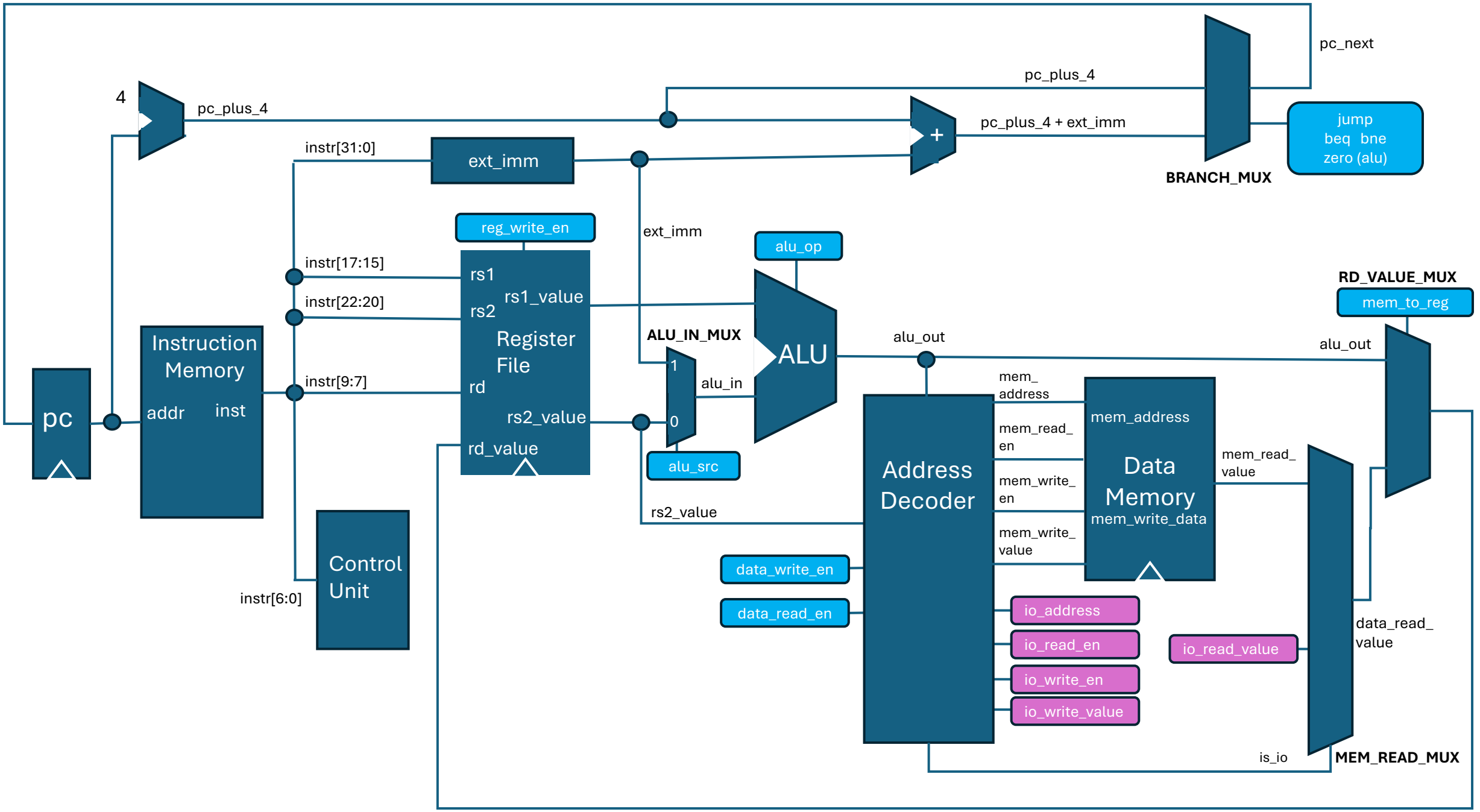
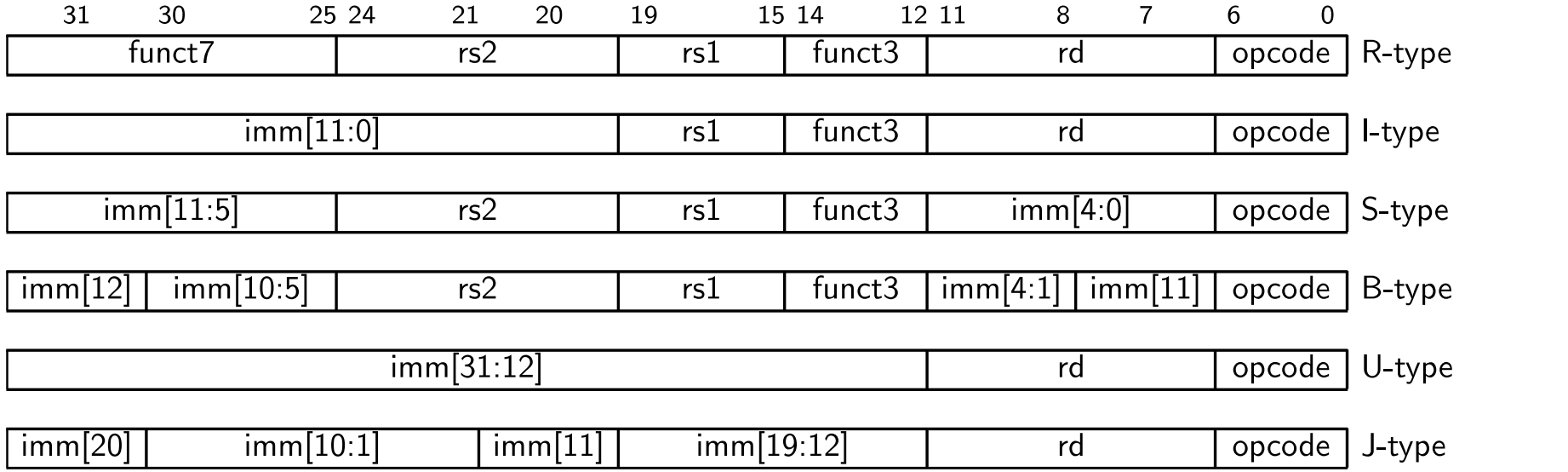
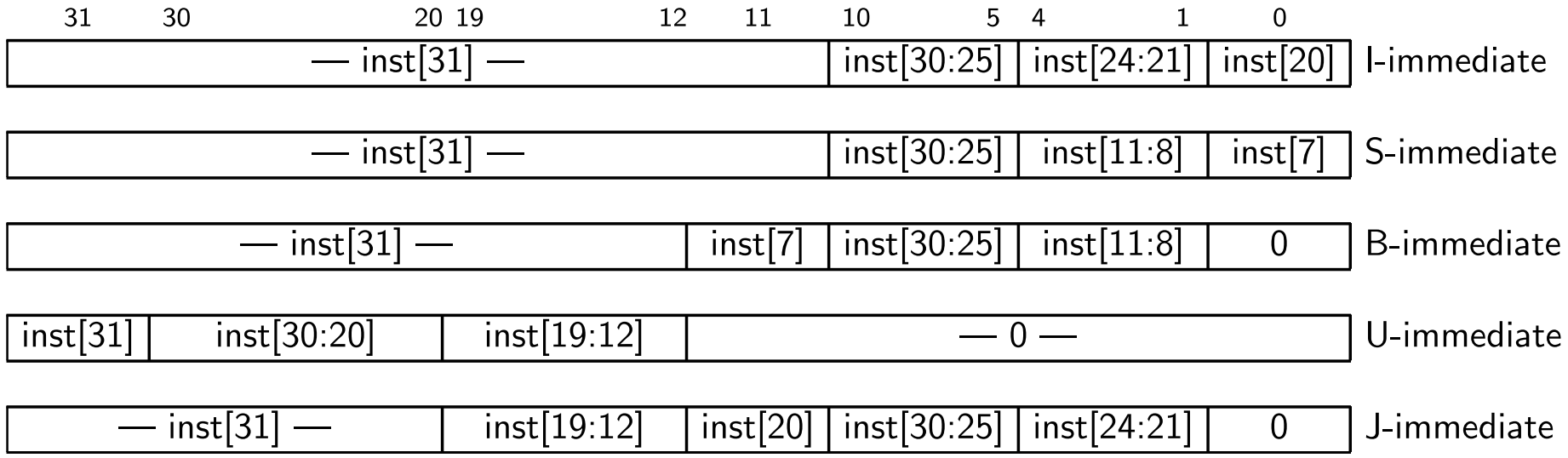
RISC V


4
pc_plus_4
pc_plus_4
pc_next
+
pc_plus_4 + ext_imm
jump
beq bne
zero (alu)
BRANCH_MUX
instr[31:0]
ext_imm
reg_write_en
ext_imm
alu_op
instr[17:15]
instr[22:20]
rs1
rs2
rs1_value
Register
File
rd
rs2_value
rd_value
ALU_IN_MUX
1
0
alu_in
ALU
alu_src
alu_out
RD_VALUE_MUX
mem_to_reg
alu_out
pc
Instruction
Memory
addr
inst
instr[9:7]
instr[6:0]
Control
Unit
rs2_value
Address
Decoder
mem_
address
mem_read_
en
mem_write_
en
mem_write_
value
mem_address
Data
Memory
mem_write_data
mem_read_
value
data_write_en
data_read_en
io_address
io_read_en
io_write_en
io_write_value
io_read_value
data_read_
value
is_io
MEM_READ_MUX

		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	
Load	I	imm[11:0]												rs1					funct3			rd					op								12
Store	S	imm[11:5]							rs2					rs1					funct3			imm[4:0]					op								12
Data	R	funct7							rs2					rs1					funct3			rd					op								
Load imm	U	imm[31:12]																				rd					op								20
Branch	B	i[12]	imm[10:5]						rs2					rs1					funct3			imm[4:1]				i[11]	op								12
Jump	J	i[20]	imm[10:1]										i[11]	imm[19:12]								rd					op								20

		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
Load	I	inst[31]																					inst[30:25]						inst[24:21]				i[20]
Store	S	inst[31]																					inst[30:25]						inst[11:8]				i[7]
Load imm	U	i[31]	inst[30:20]											inst[19:12]								0												
Branch	B	inst[31]																				i[7]	inst[30:25]						inst[11:8]				0
Jump	J	inst[31]												imm[19:12]								i[20]	inst[30:25]						inst[24:21]				0


31 30 25 24 21 20 19 15 14 12 11 8 7 6 0
funct7 rs2 rs1 funct3 rd opcode R-type
imm[11:0] rs1 funct3 rd opcode I-type
imm[11:5] rs2 rs1 funct3 imm[4:0] opcode S-type
imm[12] imm[10:5] rs2 rs1 funct3 imm[4:1] imm[11] opcode B-type
imm[31:12] rd opcode U-type
imm[20] imm[10:1] imm[11] imm[19:12] rd opcode J-type


31 30 20 19 12 11 10 5 4 1 0
— inst[31] — inst[30:25] inst[24:21] inst[20] I-immediate
— inst[31] — inst[30:25] inst[11:8] inst[7] S-immediate
— inst[31] — inst[7] inst[30:25] inst[11:8] 0 B-immediate
inst[31] inst[30:20] inst[19:12] — 0 — U-immediate
— inst[31] — inst[19:12] inst[20] inst[30:25] inst[24:21] 0 J-immediate

Inst Type	Bits	Hex max	Unsigned max	Bit Range	Instruction Range
I	12	0x00 00 0F FF	4095	-2048 to 2047	-2048 to 2047
S	12	0x00 00 0F FF	4095	-2048 to 2047	-2048 to 2047
U	20	0x00 0F FF FF	1 048 575	-524 288 to 524 287 0 to 1 048 575	-2 147 483 648 to 2 147 483 647 in steps of 4096 0 to 4294967295 in steps of 4096
B	12	0x00 00 0F FF	4095	-2048 to 2047	-4096 to 4095 in steps of 2
J	20	0x00 0F FF FF	1 048 575	-524 288 to 524 287	-1 048 576 to 1 048 575 in steps of 2

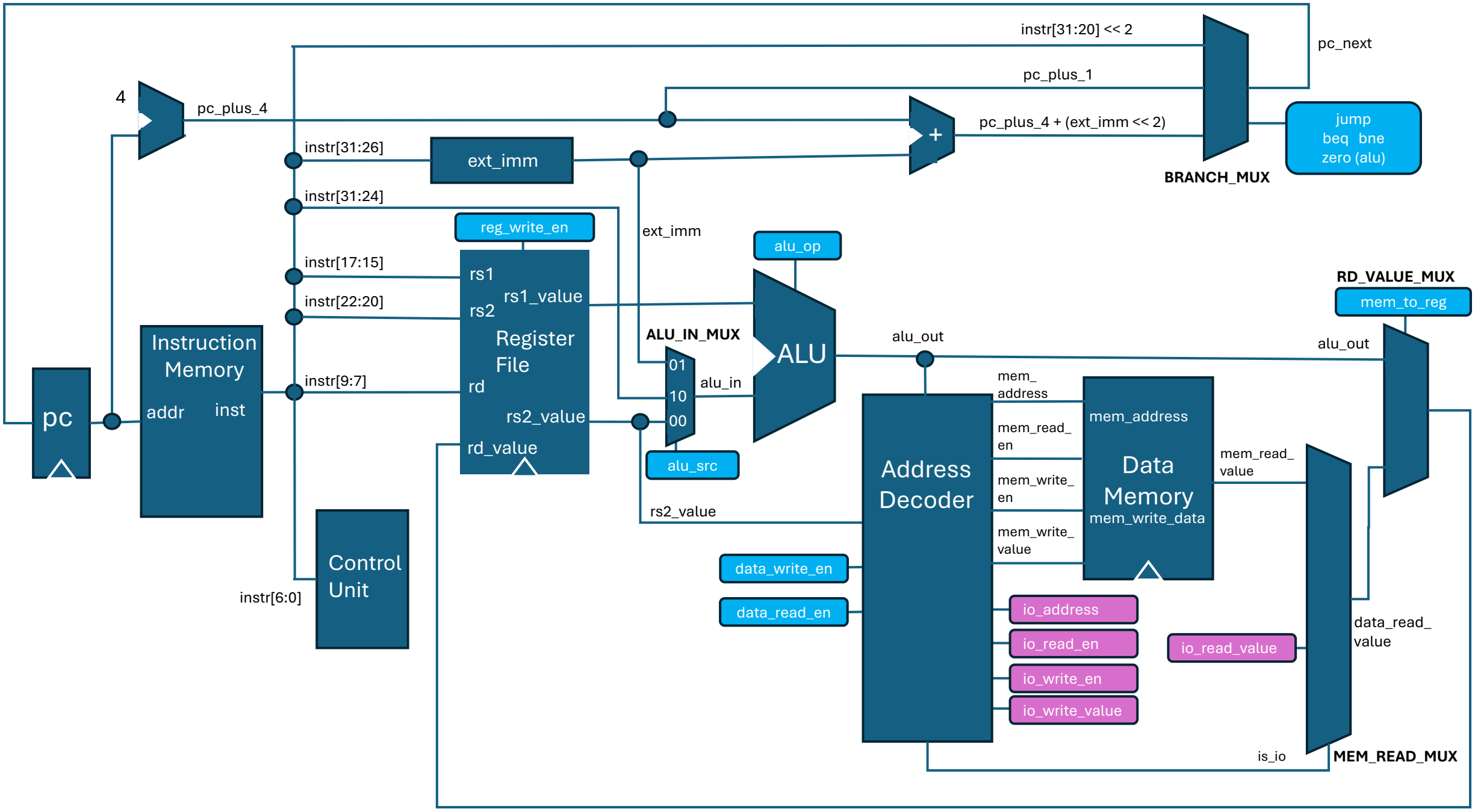
Instruction	Mnemonic	Action	Instruction Type
Load word	LD rd, offset6(rs1)	rd := mem [rs1 + imm]	S
Store word	ST rs2, offset6(rs1)	mem [rs1 + imm] := rs2	S
Add	ADD rd, rs1, rs2	rd := rs1 + rs2	R
Subtract	SUB rd, rs1, rs2	rd := rs1 - rs2	R
Invert (1s complement)	INV rd, rs1	rd := !rs1	R
Logical Shift Left	LSL rd, rs1, rs2	rd := rs1 << rs2	R
Logical Shift Right	LSR rd, rs1,rs2	rd := rs1 >> rs2	R
Bitwise AND	AND rd, rs1, rs2	rd := rs1 & rs2	R
Bitwise OR	OR rd, rs1, rs2	rd := rs1 | rs2	R
Set on Less Than	SLT rd, rs1, rs2	rd := 1 if rs1 < rs2 rd := 0 if rs1 >= rs2	R
Branch on Equal	BEQ rs1, rs2, offset6	pc := pc + 4 + imm if rs1 == rs2	B
Branch on Not Equal	BNE rs1, rs2, offset6	pc := pc + 4 + imm if rs1 != rs2	B
Jump	JMP offset12	pc := pc + 4 + imm	J
Load upper	LUI rd, imm	rd := {imm, 12'b0}	U

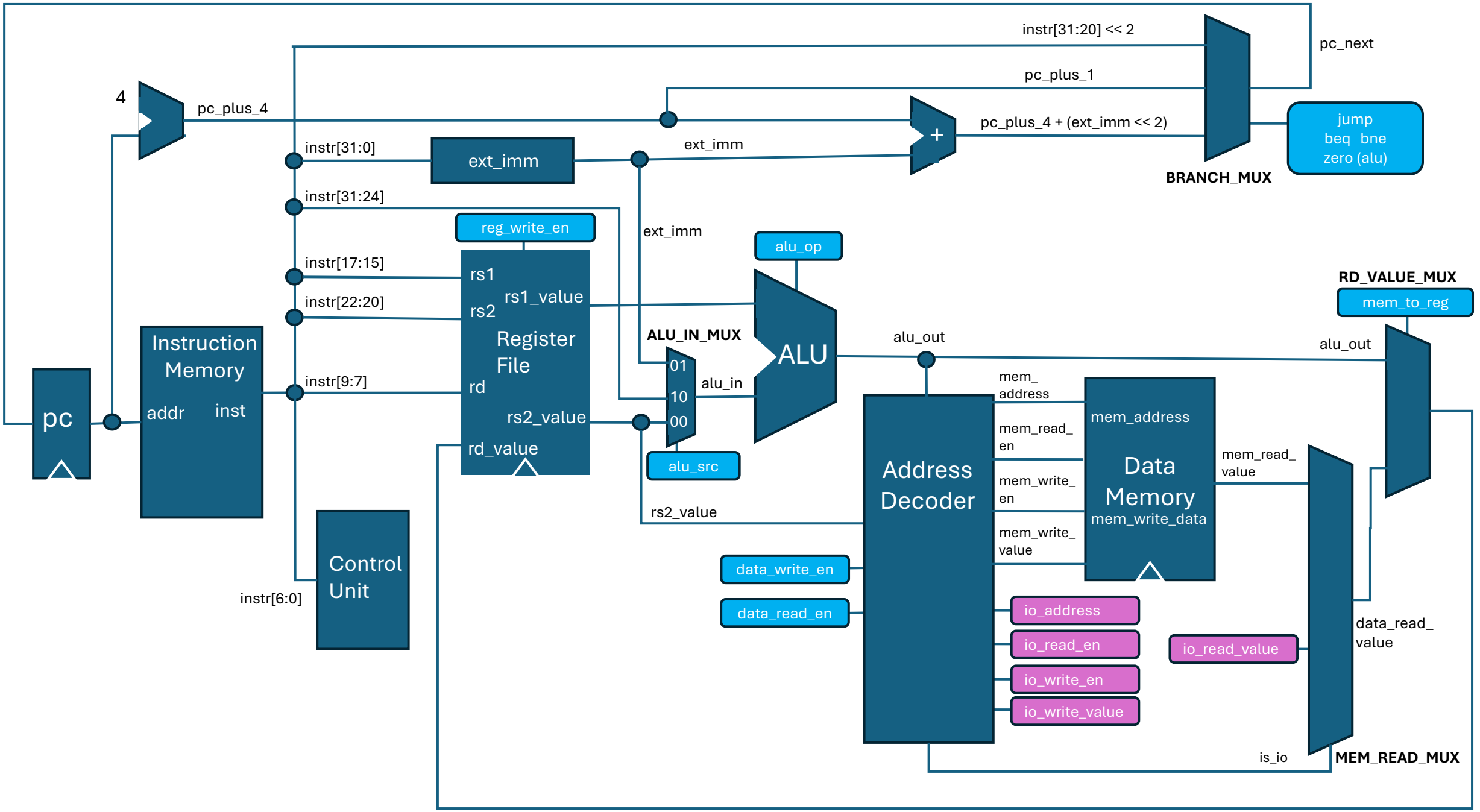
Control signals								
Instruction	**ALUSrc**	**Memto Reg**	**Reg Write**	**Mem Read**	**Mem Write**	**Branch**	**ALUOp**	**Jump**
Data-processing	00	0	1	0	0	0	see below	0
LW	01	1	1	1	0	0	0000	0
SW	01	0	0	0	1	0	0000	0
BEQ,BNE	00	0	0	0	0	1	0001	0
J	00	0	0	0	0	0	0000	1
LUI	10	0	1	0	0	0	see below	0

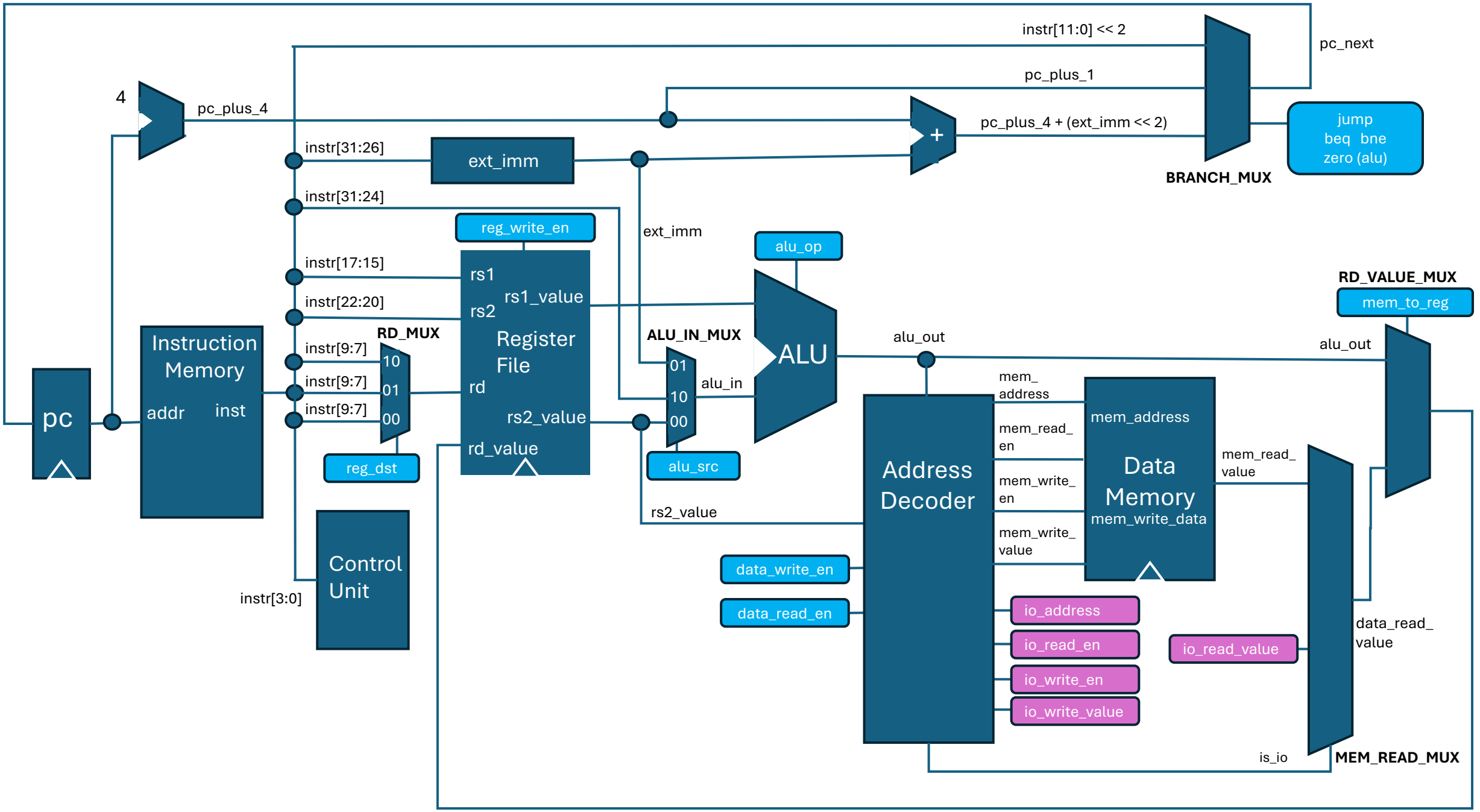
ALU Op		
Opcode	**ALU Operation**	**ALU Op**
02	ADD	0000
03	SUB	0001
04	INV	0010
05	LSL	0011
06	LSR	0100
07	AND	0101
08	OR	0110
09	SLT	0111
14	LUI	1000

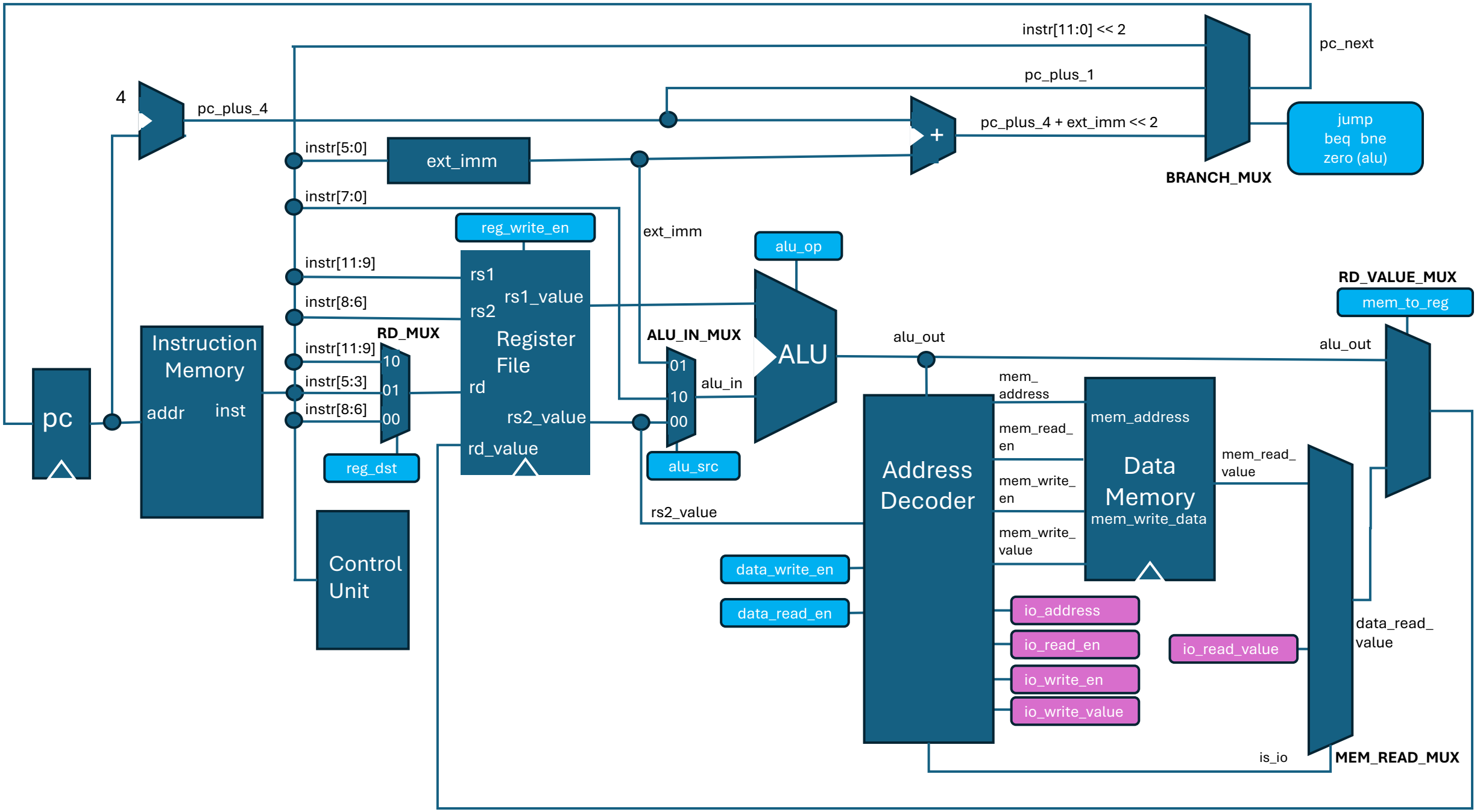
ALU Op			
Opcode	**Instruction**	**ALU Operation**	**ALUop**
00	LD	ADD	0000
01	ST	ADD	0000
02	ADD	ADD	0000
03	SUB	SUB	0001
04	INV	INV	0010
05	LSL	LSL	0011
06	LSR	LSR	0100
07	AND	AND	0101
08	OR	OR	0110
09	SLT	SLT	0111
10			
11	BEQ	SUB	0001
12	BNE	SUB	0001
13	JMP	ADD	0000
14	LUI	LUI	1000
15			

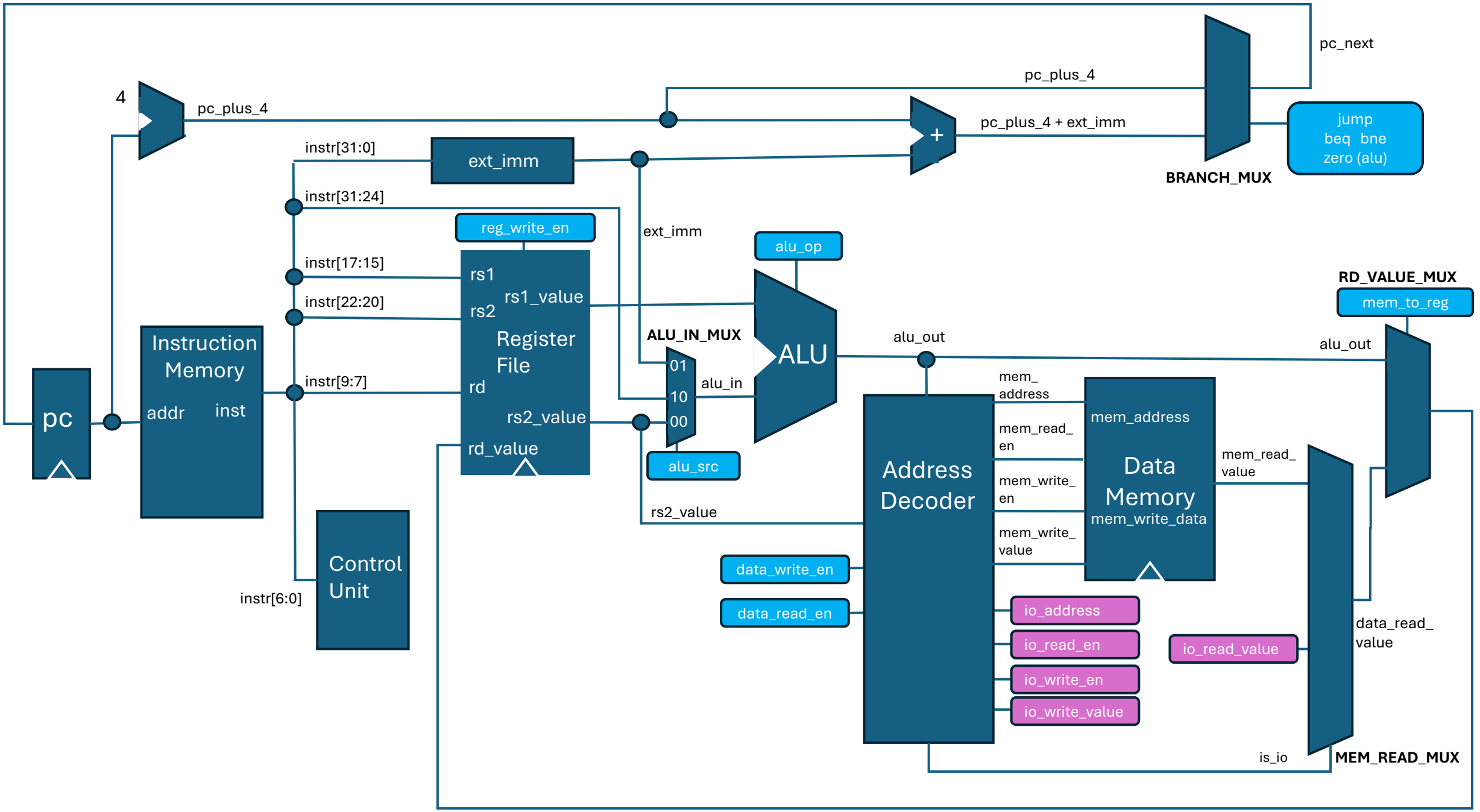
OP	**Instruction**
0000 11	Load word
0001 11	Store word
0010 11	Add
0011 11	Subtract
0100 11	Invert (1s complement)
0101 11	Logical Shift Left
0110 11	Logical Shift Right
0111 11	Bitwise AND
1000 11	Bitwise OR
1001 11	Set on Less Than
1010 11	
1011 11	Branch on Equal
1100 11	Branch on Not Equal
1101 11	Jump
1110 11	Load upper immediate
1111 11	


instr[31:20] << 2
pc_next
pc_plus_1
4
pc_plus_4
+
pc_plus_4 + (ext_imm << 2)
jump
beq bne
zero (alu)
BRANCH_MUX
instr[31:26]
ext_imm
instr[31:24]
reg_write_en
ext_imm
alu_op
instr[17:15]
rs1
rs1_value
instr[22:20]
rs2
Register
File
rd
rs2_value
rd_value
ALU_IN_MUX
01
10
00
alu_in
ALU
alu_src
RD_VALUE_MUX
mem_to_reg
alu_out
alu_out
pc
Instruction
Memory
addr
inst
instr[9:7]
rs2_value
Address
Decoder
mem_
address
mem_read_
en
mem_write_
en
mem_write_
value
mem_address
Data
Memory
mem_write_data
mem_read_
value
data_write_en
data_read_en
io_address
io_read_en
io_write_en
io_write_value
io_read_value
data_read_
value
Control
Unit
instr[6:0]
is_io
MEM_READ_MUX


instr[31:20] << 2
pc_next
pc_plus_1
4
pc_plus_4
pc_plus_4 + (ext_imm << 2)
jump
beq bne
zero (alu)
BRANCH_MUX
instr[31:0]
ext_imm
ext_imm
instr[31:24]
reg_write_en
ext_imm
alu_op
instr[17:15]
rs1
rs1_value
instr[22:20]
rs2
Register
File
rd
rs2_value
rd_value
ALU_IN_MUX
01
10
00
alu_in
ALU
alu_out
RD_VALUE_MUX
mem_to_reg
alu_out
alu_src
pc
Instruction
Memory
addr
inst
instr[9:7]
rs2_value
Address
Decoder
mem_
address
mem_read_
en
mem_write_
en
mem_write_
value
mem_address
Data
Memory
mem_write_data
mem_read_
value
data_write_en
data_read_en
io_address
io_read_en
io_write_en
io_write_value
io_read_value
data_read_
value
instr[6:0]
Control
Unit
is_io
MEM_READ_MUX


instr[11:0] << 2
pc_next
pc_plus_1
4
pc_plus_4
+
pc_plus_4 + (ext_imm << 2)
jump
beq bne
zero (alu)
BRANCH_MUX
instr[31:26]
ext_imm
instr[31:24]
reg_write_en
ext_imm
alu_op
instr[17:15]
rs1
rs1_value
instr[22:20]
rs2
Register
File
rd
rs2_value
rd_value
RD_VALUE_MUX
mem_to_reg
RD_MUX
ALU_IN_MUX
ALU
alu_out
alu_out
instr[9:7]
10
01
00
alu_in
pc
Instruction
Memory
addr
inst
reg_dst
alu_src
mem_
address
mem_address
mem_read_
en
Data
Memory
mem_write_data
mem_read_
value
mem_write_
en
mem_write_
value
Address
Decoder
rs2_value
Control
Unit
instr[3:0]
data_write_en
data_read_en
io_address
io_read_en
io_write_en
io_write_value
io_read_value
data_read_
value
is_io
MEM_READ_MUX


instr[11:0] << 2
pc_next
pc_plus_1
4
pc_plus_4
pc_plus_4 + ext_imm << 2
jump
beq bne
zero (alu)
BRANCH_MUX
instr[5:0]
ext_imm
instr[7:0]
reg_write_en
ext_imm
alu_op
instr[11:9]
instr[8:6]
rs1
rs2
rs1_value
Register
File
rd
rs2_value
rd_value
RD_MUX
ALU_IN_MUX
ALU
alu_out
RD_VALUE_MUX
mem_to_reg
alu_out
Instruction
Memory
addr
inst
pc
instr[11:9]
instr[5:3]
instr[8:6]
10
01
00
reg_dst
01
10
00
alu_in
alu_src
rs2_value
Address
Decoder
mem_
address
mem_read_
en
mem_write_
en
mem_write_
value
mem_address
Data
Memory
mem_write_data
mem_read_
value
data_write_en
data_read_en
io_address
io_read_en
io_write_en
io_write_value
io_read_value
data_read_
value
is_io
MEM_READ_MUX
Control
Unit


4
pc_plus_4
pc_next
pc_plus_4
+
pc_plus_4 + ext_imm
jump
beq bne
zero (alu)
BRANCH_MUX
instr[31:0]
ext_imm
instr[31:24]
reg_write_en
ext_imm
alu_op
instr[17:15]
instr[22:20]
rs1
rs2
rs1_value
Register
File
rd
rs2_value
rd_value
ALU_IN_MUX
01
10
00
alu_in
ALU
alu_out
RD_VALUE_MUX
mem_to_reg
alu_out
pc
Instruction
Memory
addr
inst
instr[9:7]
alu_src
rs2_value
Address
Decoder
mem_
address
mem_read_
en
mem_write_
en
mem_write_
value
mem_address
Data
Memory
mem_write_data
mem_read_
value
data_write_en
data_read_en
io_address
io_read_en
io_write_en
io_write_value
io_read_value
data_read_
value
Control
Unit
instr[6:0]
is_io
MEM_READ_MUX