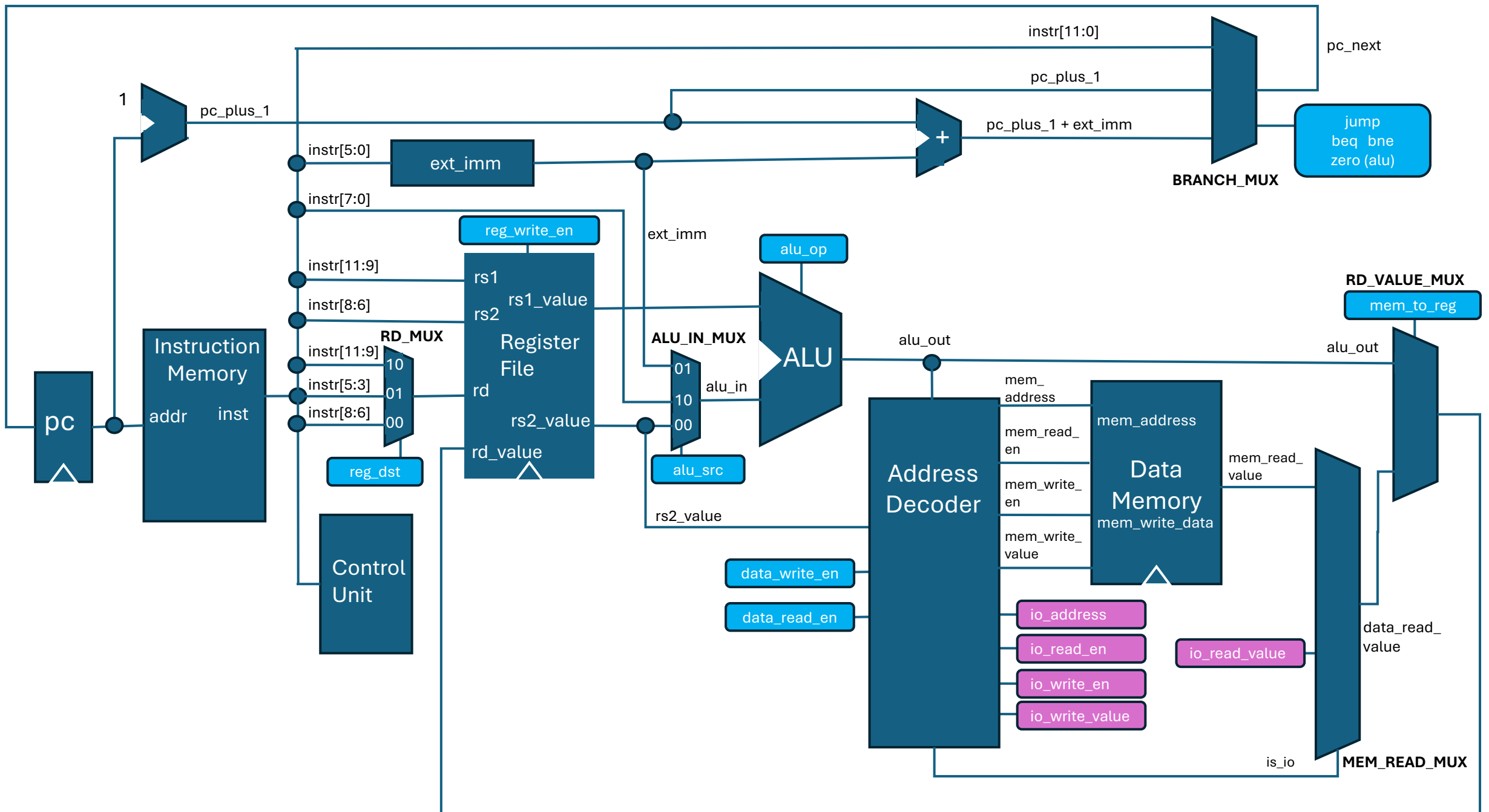


RISC v7



Instruction	Mnemonic	Action
Load word	LD rd, offset6(rs1)	$rd := \text{mem} [rs1 + \text{offset6}]$
Store word	ST rs2, offset6(rs1)	$\text{mem} [rs1 + \text{offset6}] := rs2$
Add	ADD rd, rs1, rs2	$rd := rs1 + rs2$
Subtract	SUB rd, rs1, rs2	$rd := rs1 - rs2$
Invert (1s complement)	INV rd, rs1	$rd := !rs1$
Logical Shift Left	LSL rd, rs1, rs2	$rd := rs1 \ll rs2$
Logical Shift Right	LSR rd, rs1, rs2	$rd := rs1 \gg rs2$
Bitwise AND	AND rd, rs1, rs2	$rd := rs1 \& rs2$
Bitwise OR	OR rd, rs1, rs2	$rd := rs1 rs2$
Set on Less Than	SLT rd, rs1, rs2	$rd := 1 \text{ if } rs1 < rs2$ $rd := 0 \text{ if } rs1 \geq rs2$
Branch on Equal	BEQ rs1, rs2, offset6	$PC := PC + 1 + \text{offset6} \quad \text{if } rs1 == rs2$
Branch on Not Equal	BNE rs1, rs2, offset6	$PC := PC + 1 + \text{offset6} \quad \text{if } rs1 \neq rs2$
Jump	JMP offset12	$PC := \text{offset12}$
Load upper	LUI rd, imm8	$rd := \{\text{imm8}, rd[7:0]\}$
Load lower	LLI rd, imm8	$rd := \{rd[15:8], \text{imm8}\}$

OP	Instruction
0000	Load word
0001	Store word
0002	Add
0003	Subtract
0004	Invert (1s complement)
0005	Logical Shift Left
0006	Logical Shift Right
0007	Bitwise AND
0008	Bitwise OR
0009	Set on Less Than
0011	Branch on Equal
0012	Branch on Not Equal
0013	Jump
0014	Load upper immediate
0015	Load lower immed

Load word

Op	rs1	rd	offset6
4	3	3	6

Store word

Op	rs1	rs2	offset6
4	3	3	6

Data
processing

Op	rs1	rs2	rd	
4	3	3	3	3

Load imm

Op	rd		imm8
4	3	1	8

Branch

Op	rs1	rs2	offset6
4	3	3	6

Jump

Op	offset12
4	12

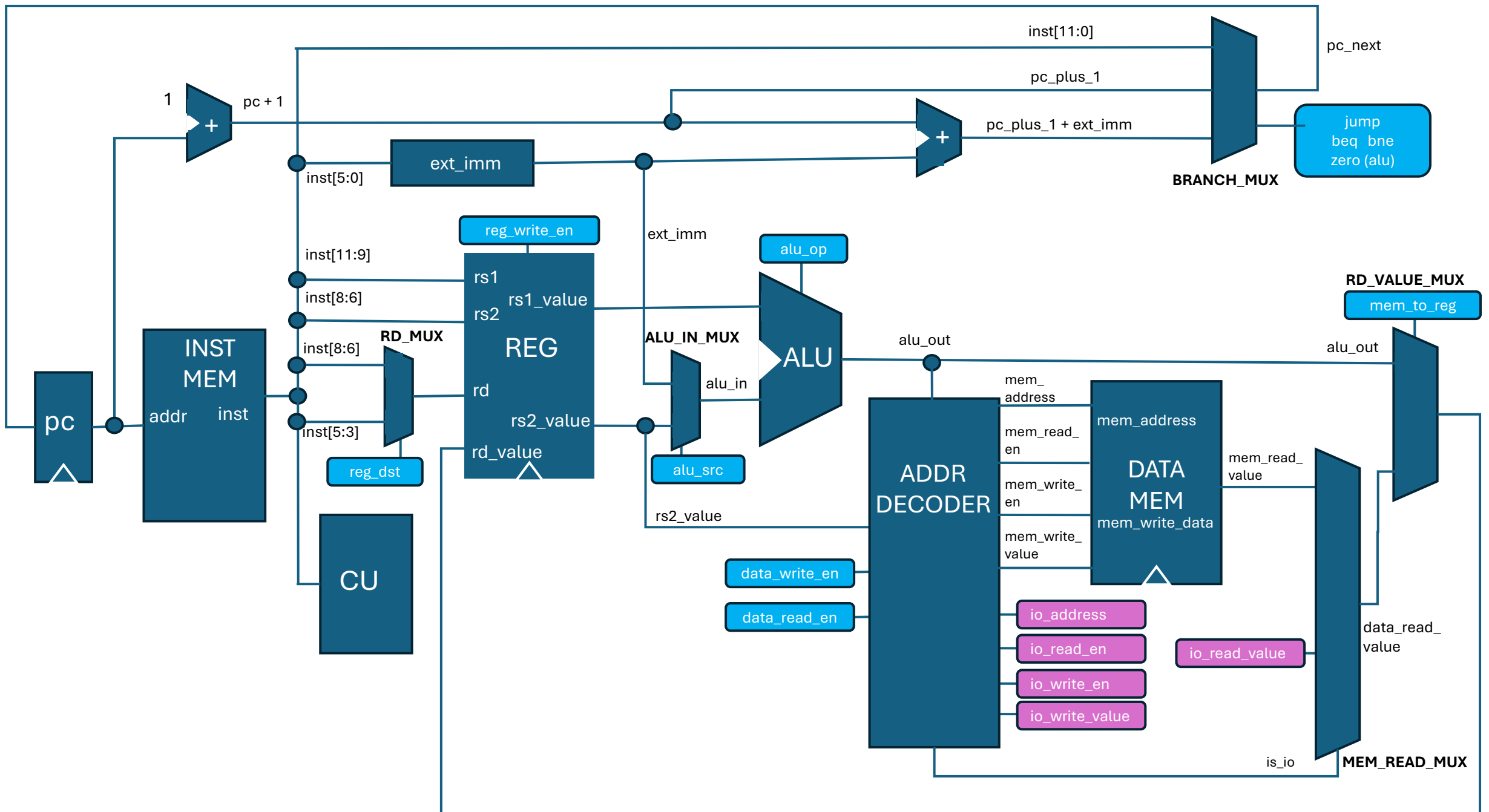
Control signals									
Instruction	Reg Dst	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp	Jump
Data-processing	01	00	0	1	0	0	0	see below	0
LW	00	01	1	1	1	0	0	0000	0
SW	00	01	0	0	0	1	0	0000	0
BEQ,BNE	00	00	0	0	0	0	1	0001	0
J	00	00	0	0	0	0	0	0000	1
LUI, LLI	01	10	0	1	0	0	0	see below	0

RISC v3

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
15	LLI	1001

ALU Op			
Opcode	Instruction	ALU Operation	ALUop
00	LD	ADD	0000
01	ST	SUB	0001
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	LLI	LLI	1001

RISC v6



RISC v5

Instruction	Mnemonic	Action
Load word	LD rd, offset(rs1)	rd := mem [rs1 + offset]
Store word	ST rs2, offset(rs1)	mem [rs1 + offset] := rs2
Add	ADD rd, rs1, rs2	rd := rs1 + rs2
Subtract	SUB rd, rs1, rs2	rd := rs1 - rs2
Invert (1s complement)	INV rd, rs1	rd := !rs1
Logical Shift Left	LSL rd, rs1, rs2	rd := rs1 << rs2
Logical Shift Right	LSR rd, rs1, rs2	rd := rs1 >> rs2
Bitwise AND	AND rd, rs1, rs2	rd := rs1 & rs2
Bitwise OR	OR rd, rs1, rs2	rd := rs1 rs2
Set on Less Than	SLT rd, rs1, rs2	rd := 1 if rs1 < rs2 rd := 0 if rs1 >= rs2
Branch on Equal	BEQ rs1, rs2, offset	PC := PC + 1 + offset if rs1 == rs2
Branch on Not Equal	BNE rs1, rs2, offset	PC := PC + 1 + offset if rs1 != rs2
Jump	JMP offset	PC := PC[15:12], inst[11:0]}

OP	Instruction
0000	Load word
0001	Store word
0002	Add
0003	Subtract
0004	Invert (1s complement)
0005	Logical Shift Left
0006	Logical Shift Right
0007	Bitwise AND
0008	Bitwise OR
0009	Set on Less Than
0011	Branch on Equal
0012	Branch on Not Equal
0013	Jump

Load word

Op	rs1	rd	offset
4	3	3	6

Store word

Op	rs1	rs2	offset
4	3	3	6

Data
processing

Op	rs1	rs2	rd	
4	3	3	3	3

Branch

Op	rs1	rs2	offset
4	3	3	6

Jump

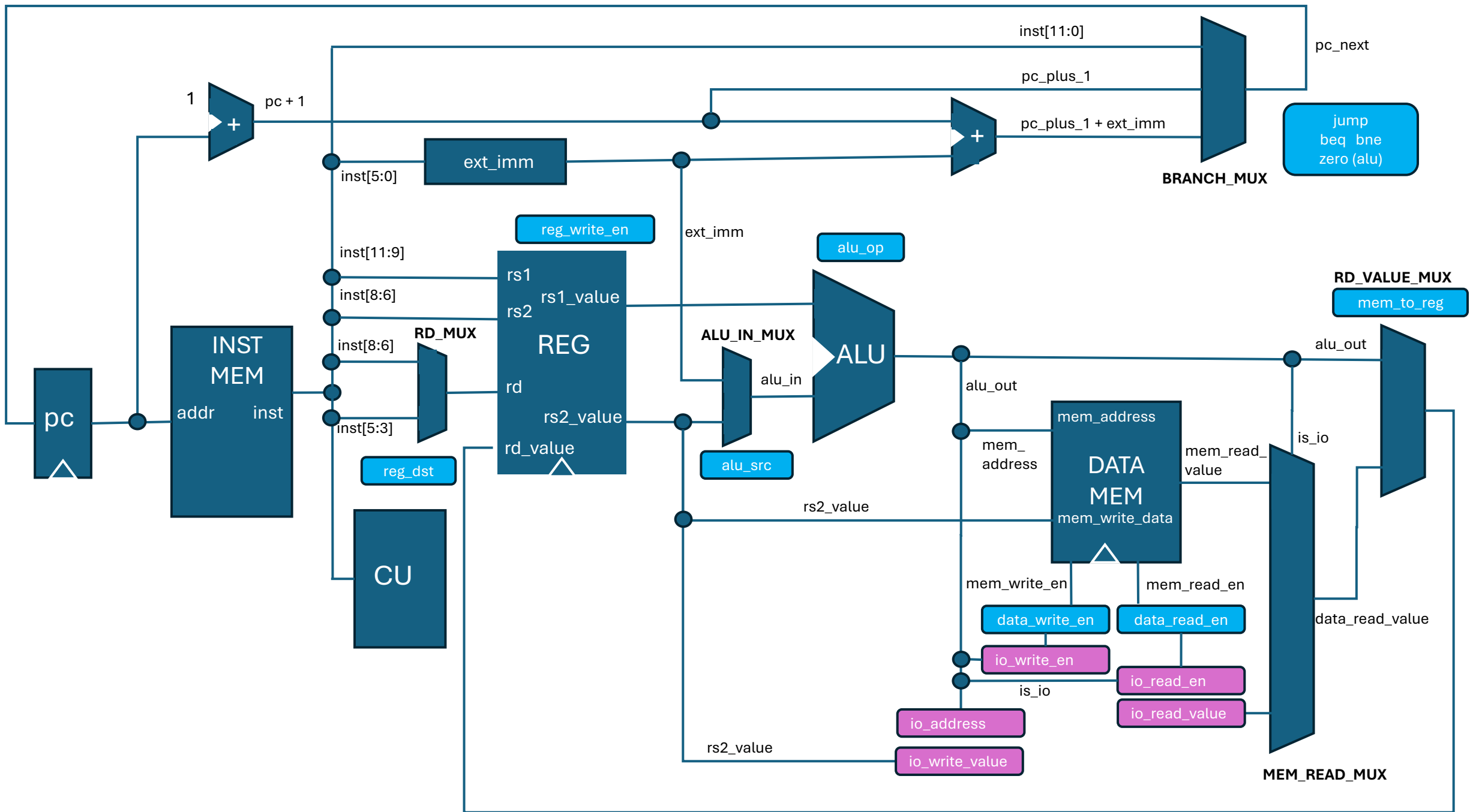
Op	offset
4	12

Control signals									
Instruction	Reg Dst	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp	Jump
Data-processing	1	0	0	1	0	0	0	see below	0
LW	0	1	1	1	1	0	0	000	0
SW	0	1	0	0	0	1	0	000	0
BEQ,BNE	0	0	0	0	0	0	1	001	0
J	0	0	0	0	0	0	0	000	1

RISC v3

ALU Op		
Opcode	ALU Operation	ALU Op
02	ADD	000
03	SUB	001
04	INVERT	010
05	LSL	011
06	LSR	100
07	AND	101
08	OR	110
09	SLT	111

ALU Op			
Opcode	Instruction	ALU Operation	ALUop
00	LD	ADD	000
01	ST	SUB	001
02	ADD	ADD	000
03	SUB	SUB	001
04	INV	INVERT	010
05	LSL	LSL	011
06	LSR	LSR	100
07	AND	AND	101
08	OR	OR	110
09	SLT	SLT	111
11	BEQ	SUB	001
12	BNE	SUB	001
13	JMP	ADD	000



RISC v3

Data memory is 16 bit words and word addressed

Instruction memory is 16 bit words, word addressed but the code uses byte addresses and converts to words – which is inconsistent

Changing it to use word addresses for instructions too – so PC increments by 1 each cycle

instruction_memory.v

```
14      wire [3 : 0] rom_addr = pc[3 : 0];
```

datapath.v

```
26      wire [11:0] jump_shift;  
37      assign pc2 = pc_current + 16'd1;  
44      assign jump_shift = instr[11:0];  
75      assign PC_beq = pc2 + ext_im[15:0];  
76      assign PC_bne = pc2 + ext_im[15:0];  
91      assign PC_j = {pc2[15:12],jump_shift};
```


Instruction	Mnemonic	Action
Load word	LD ws, offset(rs1)	ws := mem16[rs1 + offset]
Store word	ST rs2, offset(rs1)	mem16[rs1 + offset] := rs2
Add	ADD ws, rs1, rs2	ws := rs1 + rs2
Subtract	SUB ws, rs1, rs2	ws := rs1 - rs2
Invert (1s complement)	INV ws, rs1	ws := !rs1
Logical Shift Left	LSL ws, rs1, rs2	ws := rs1 << rs2
Logical Shift Right	LSR ws, rs1, rs2	ws := rs1 >> rs2
Bitwise AND	AND ws, rs1, rs2	ws := rs1 & rs2
Bitwise OR	OR ws, rs1, rs2	ws := rs1 rs2
Set on Less Than	SLT ws, rs1, rs2	ws := 1 if rs1 < rs2 ws := 0 if rs1 >= rs2
Branch on Equal	BEQ rs1, rs2, offset	PC := PC + 1 + offset if rs1 != rs2
Branch on Not Equal	BNE rs1, rs2, offset	PC := PC + 1 + offset if rs1 == rs2
Jump	JMP offset	PC := PC[15:12], inst[11:0]}

OP	Instruction
0000	Load word
0001	Store word
0002	Add
0003	Subtract
0004	Invert (1s complement)
0005	Logical Shift Left
0006	Logical Shift Right
0007	Bitwise AND
0008	Bitwise OR
0009	Set on Less Than
0011	Branch on Equal
0012	Branch on Not Equal
0013	Jump

Load word

Op	rs1	ws	offset
4	3	3	6

Store word

Op	rs1	rs2	offset
4	3	3	6

Data
processing

Op	rs1	rs2	ws	
4	3	3	3	3

Branch

Op	rs1	rs2	offset
4	3	3	6

Jump

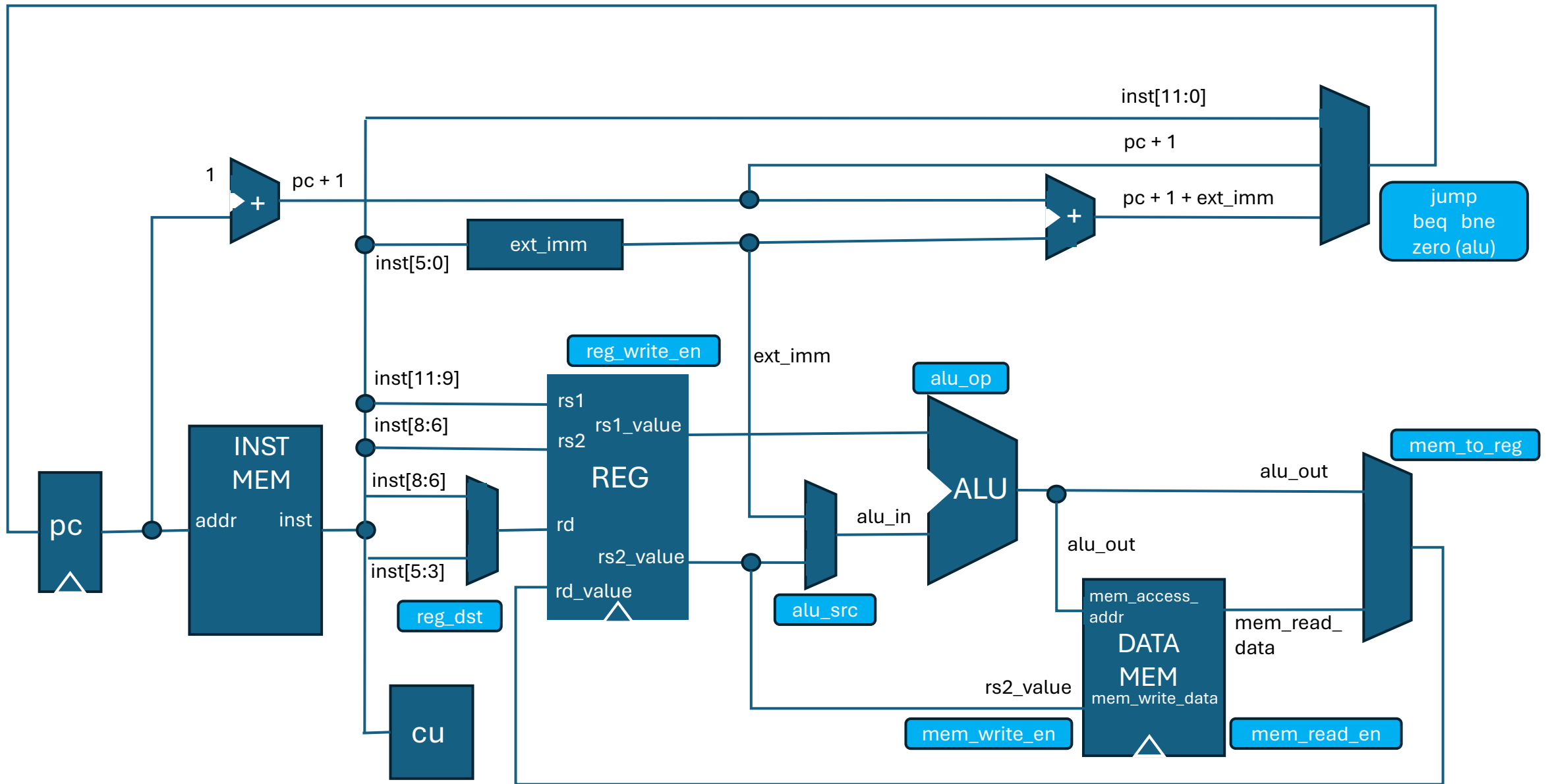
Op	offset
4	12

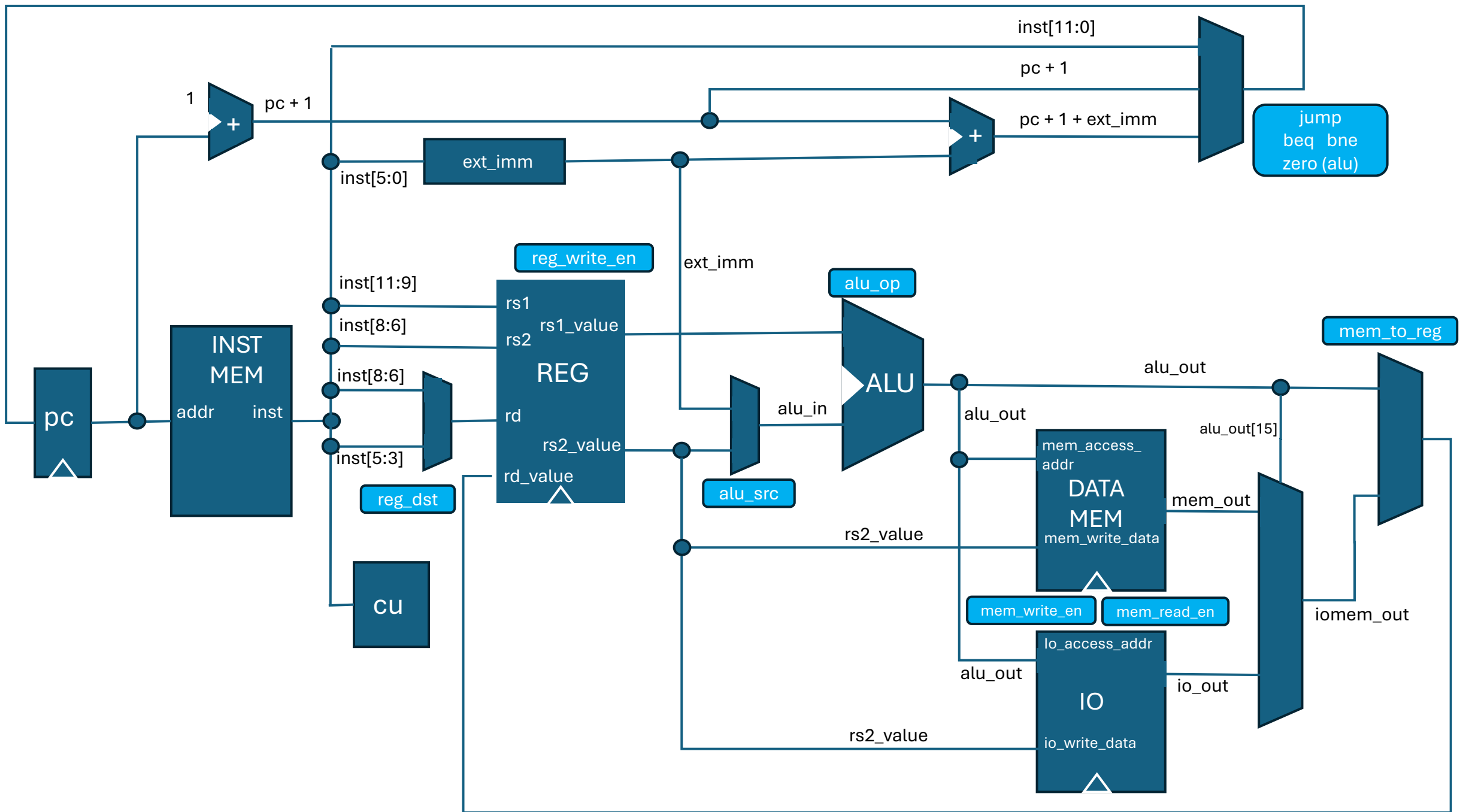
Control signals									
Instruction	Reg Dst	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp	Jump
Data-processing	1	0	0	1	0	0	0	see below	0
LW	0	1	1	1	1	0	0	000	0
SW	0	1	0	0	0	1	0	000	0
BEQ,BNE	0	0	0	0	0	0	1	001	0
J	0	0	0	0	0	0	0	000	1

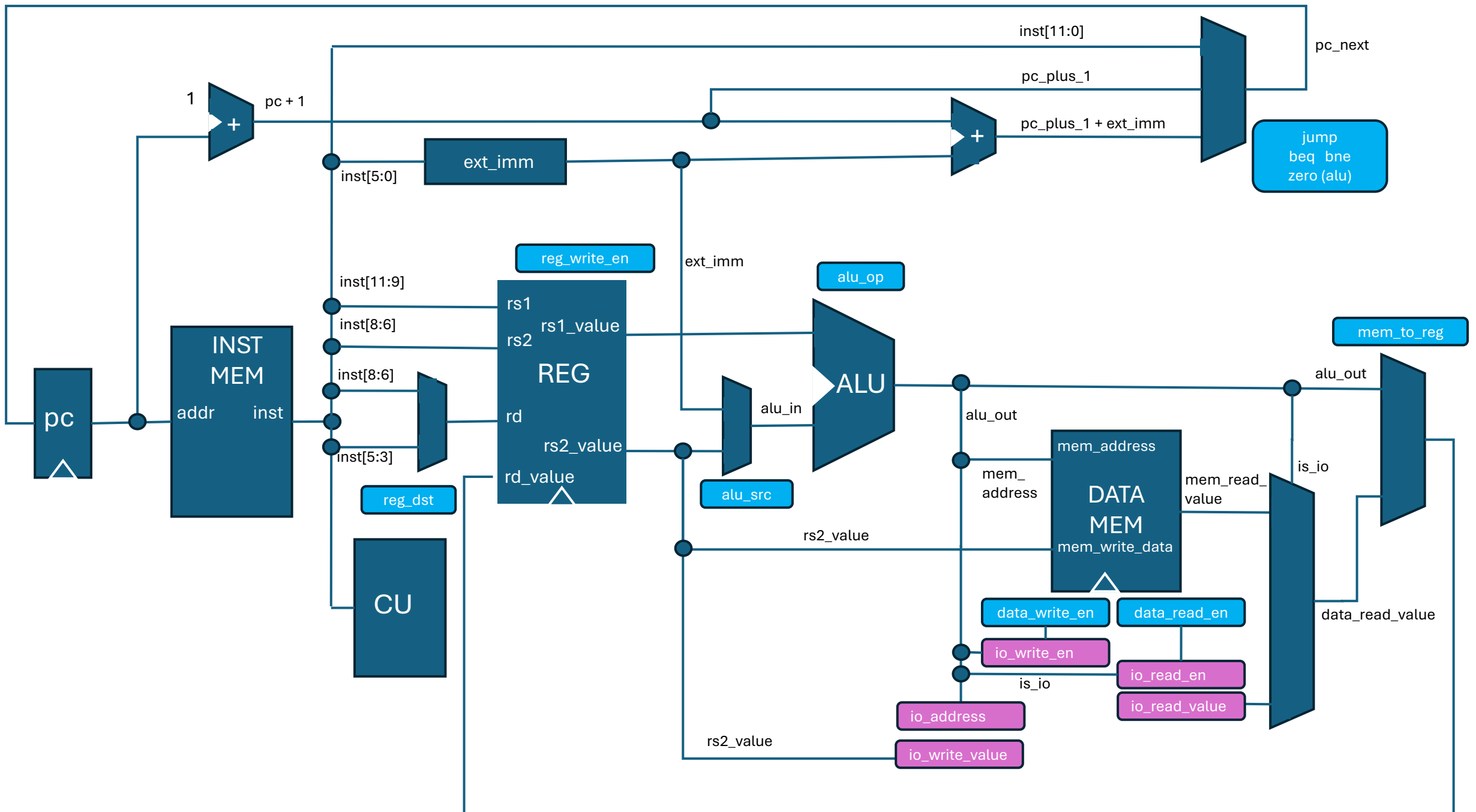
RISC v3

ALU Op		
Opcode	ALU Operation	ALU Op
02	ADD	000
03	SUB	001
04	INVERT	010
05	LSL	011
06	LSR	100
07	AND	101
08	OR	110
09	SLT	111

ALU Op			
Opcode	Instruction	ALU Operation	ALUop
00	LW	ADD	000
01	SW	SUB	001
02	ADD	ADD	000
03	SUB	SUB	001
04	INVERT	INVERT	010
05	LSL	LSL	011
06	LSR	LSR	100
07	AND	AND	101
08	OR	OR	110
09	SLT	SLT	111
11	BEQ	SUB	001
12	BNE	SUB	001
13	JMP	ADD	000







RISC v2

Instruction	Mnemonic	Action
Load word	LD ws, offset(rs1)	$ws := \text{mem16}[rs1 + \text{offset}]$
Store word	ST rs2, offset(rs1)	$\text{mem16}[rs1 + \text{offset}] := rs2$
Add	ADD ws, rs1, rs2	$ws := rs1 + rs2$
Subtract	SUB ws, rs1, rs2	$ws := rs1 - rs2$
Invert (1s complement)	INV ws, rs1	$ws := !rs1$
Logical Shift Left	LSL ws, rs1, rs2	$ws := rs1 \ll rs2$
Logical Shift Right	LSR ws, rs1, rs2	$ws := rs1 \gg rs2$
Bitwise AND	AND ws, rs1, rs2	$ws := rs1 \& rs2$
Bitwise OR	OR ws, rs1, rs2	$ws := rs1 rs2$
Set on Less Than	SLT ws, rs1, rs2	$ws := 1 \text{ if } rs1 < rs2$ $ws := 0 \text{ if } rs1 \geq rs2$
Branch on Equal	BEQ rs1, rs2, offset	$PC := PC + 2 + \text{offset} \ll 1$ if $rs1 \neq rs2$
Branch on Not Equal	BNE rs1, rs2, offset	$PC := PC + 2 + \text{offset} \ll 1$ if $rs1 == rs2$
Jump	JMP offset	$PC := \{PC[15:13], (\text{offset} \ll 1)\}$

OP	Instruction
0000	Load word
0001	Store word
0002	Add
0003	Subtract
0004	Invert (1s complement)
0005	Logical Shift Left
0006	Logical Shift Right
0007	Bitwise AND
0008	Bitwise OR
0009	Set on Less Than
0011	Branch on Equal
0012	Branch on Not Equal
0013	Jump

Load word

Op	rs1	ws	offset
4	3	3	6

Store word

Op	rs1	rs2	offset
4	3	3	6

Data
processing

Op	rs1	rs2	ws	
4	3	3	3	3

Branch

Op	rs1	rs2	offset
4	3	3	6

Jump

Op	offset
4	12

Control signals									
Instruction	Reg Dst	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp	Jump
Data-processing	1	0	0	1	0	0	0	see below	0
LW	0	1	1	1	1	0	0	000	0
SW	0	1	0	0	0	1	0	000	0
BEQ,BNE	0	0	0	0	0	0	1	001	0
J	0	0	0	0	0	0	0	000	1

RISC v2

ALU Op		
Opcode	ALU Operation	ALU Op
02	ADD	000
03	SUB	001
04	INVERT	010
05	LSL	011
06	LSR	100
07	AND	101
08	OR	110
09	SLT	111

ALU Op			
Opcode	Instruction	ALU Operation	ALUop
00	LW	ADD	000
01	SW	SUB	001
02	ADD	ADD	000
03	SUB	SUB	001
04	INVERT	INVERT	010
05	LSL	LSL	011
06	LSR	LSR	100
07	AND	AND	101
08	OR	OR	110
09	SLT	SLT	111
11	BEQ	SUB	001
12	BNE	SUB	001
13	JMP	ADD	000

RISC v1

ALU Control				
ALUOp	Opcode (hex)	ALUcnt	ALU Operation	Instruction
10	xxxx	000	ADD	LW,SW
01	xxxx	001	SUB	BEQ,BNE
00	0002	000	ADD	D-type: ADD
00	0003	001	SUB	D-type: SUB
00	0004	010	INVERT	D-type: INVERT
00	0005	011	LSL	D-type: LSL
00	0006	100	LSR	D-type: LSR
00	0007	101	AND	D-type: AND
00	0008	110	OR	D-type: OR
00	0009	111	SLT	D-type: SLT