PROCESSOR DESIGN PROJECT
CENG - 3010

COMPUTER ORGANIZATION PROJECT
GROUP MEMBERS

İLKAY TÜRE - 192010020035
FURKAN ALİ TUNÇ - 192010020011
HATİCE BETÜL ESEROĞLU - 192010020100

R - TYPE

Opcode	Rs	Rt	Rd	Shampt
5 bits	3 bits	3 bits	3 bits	2 bits

J - TYPE

Opcode	Target
5 bits	11 bits

I - TYPE

Opcode	Rs	Rt	Immediate
5 bits	3 bits	3 bits	5 bits

REGISTERS

Reg Number	Name
0	$zero
1	$t0
2	$t1
3	$t2
4	$s0
5	$s1
6	$sp
7	$ra
	pc

OPCODES

INSTRUCTION	OPCODE	DESCRIPTION
add	00000	R[rd] = R[rs] + R[rt]
sub	00001	R[rd] = R[rs] - R[rt]
and	00010	R[rd] = R[rs] & R[rt]
or	00011	R[rd] = R[rs] | R[rt]
xor	00100	R[$rd] ← R[$rs] ^ R[$rt]
sll	00101	R[$rd] ← R[$rt] << shamt
srl	00110	R[$rd] ← R[$rt] >> shamt
jr	00111	PC=R[ra]

INSTRUCTION	OPCODE	DESCRIPTION
j	01001	PC=JumpAddr
jal	01010	$ra=PC+2, PC=JumpAddr
lw	01011	R[rt] = M[R[rs]+SignExtImm]
sw	01100	M[R[rs]+SignExtImm] = R[rt]
addi	01101	R[rt] = R[rs] + SignExtImm
andi	01110	R[$rt] ← R[$rs] & {imm}
ori	01111	R[$rt] ← R[$rs] | {imm}
bne	10000	if(R[rs]!=R[rt]) PC=PC+2+BranchAddr
beq	10001	if(R[rs]==R[rt]) PC=PC+2+BranchAddr

ALU CONTROL

ALU Operation	ALU Cntrl
Add	000
Sub	001
And	010
Xor	011
Or	100
Sll	101
Srl	110

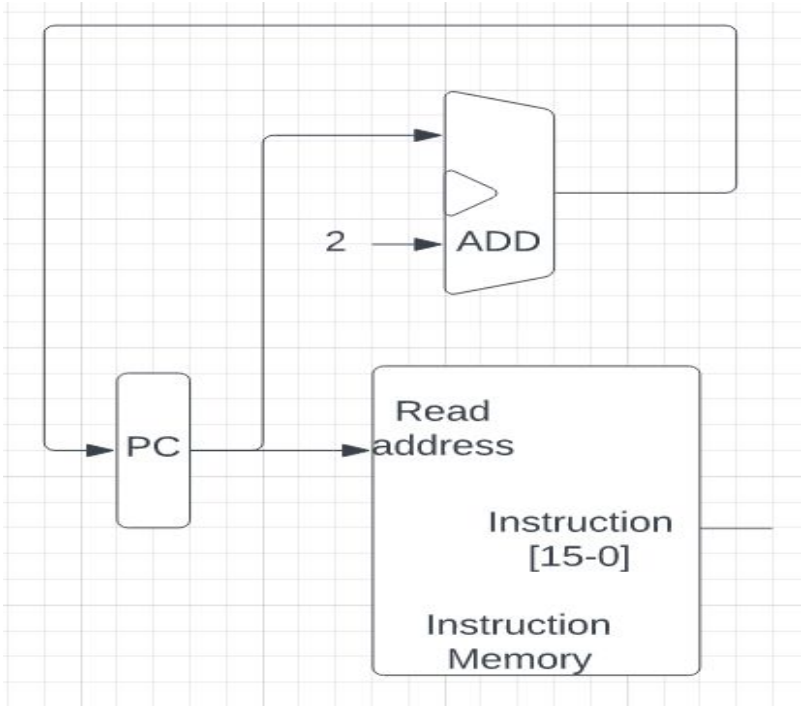
Processor Control Unit Design

Instruction	Reg Dst	ALU Src	Mem to Reg	Reg Write	Mem Read	Mem Write	Branch	ALU Op	Jump Mux	Jal Mux
add	1	0	0	1	0	0	0	000	0	0
sub	1	0	0	1	0	0	0	001	0	0
and	1	0	0	1	0	0	0	010	0	0
or	1	0	0	1	0	0	0	100	0	0
xor	1	0	0	1	0	0	0	011	0	0
sll	1	2	0	1	0	0	0	101	0	0
srl	1	2	0	1	0	0	0	110	0	0
jr	x	x	x	0	0	0	0	111	2	0

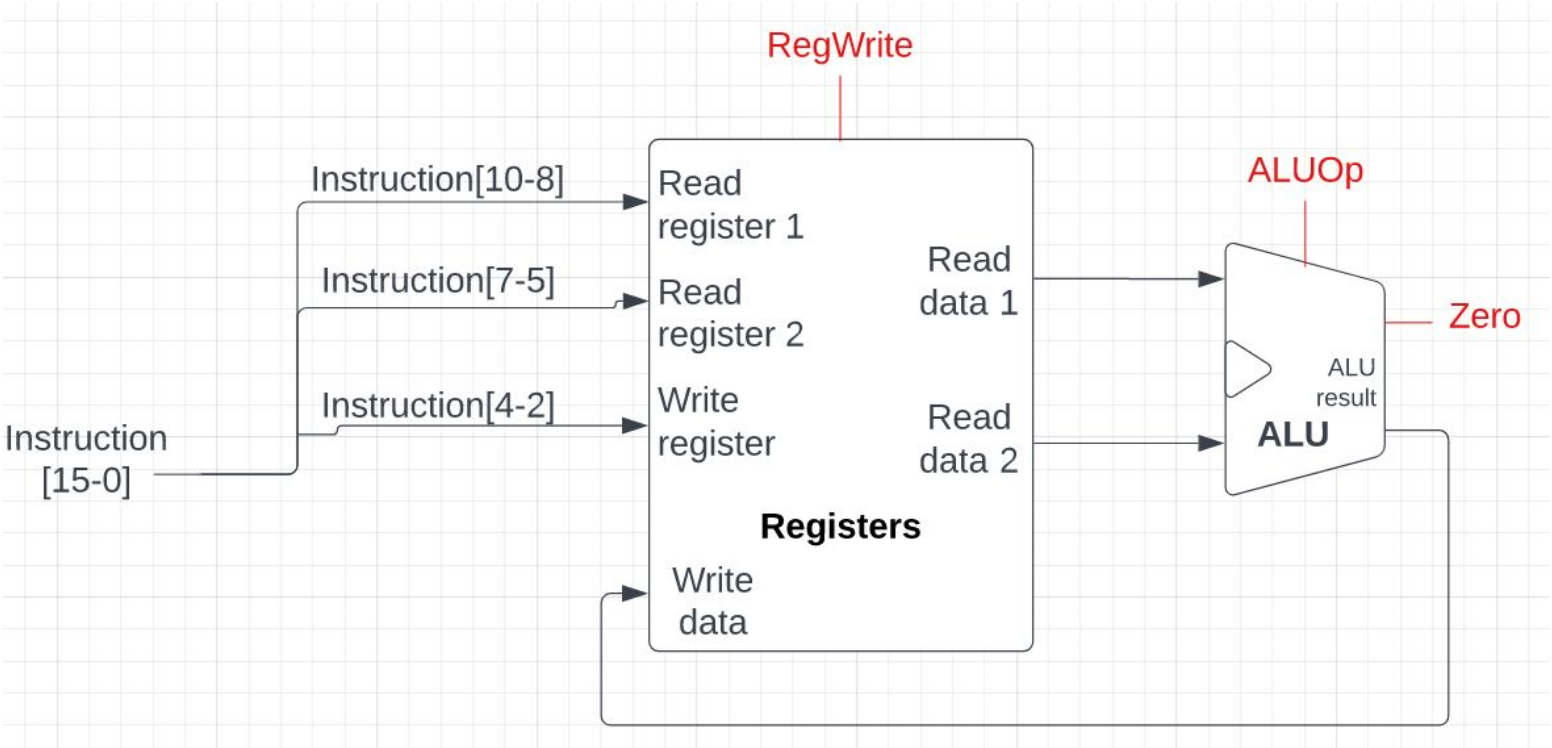
Instruction	Reg Dst	ALU Src	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp	Jump Mux	Jal Mux
addi	0	1	0	1	0	0	0	000	0	0
andi	0	1	0	1	0	0	0	001	0	0
ori	0	1	0	1	0	0	0	010	0	0
lw	0	1	1	1	1	0	0	000	0	0
sw	x	1	x	0	0	1	0	000	0	0
beq	x	0	x	0	0	0	1	001	0	0
bne	x	0	x	0	0	0	1	001	0	0

Instruction	Reg Dst	ALUSrc	Memto Reg	Reg Write	MemRead	Mem Write	Branch	ALUOp	Jump Mux	Jal Mux
j	x	x	x	0	0	0	0	x	1	0
jal	0	x	x	1	0	0	x	x	1	1

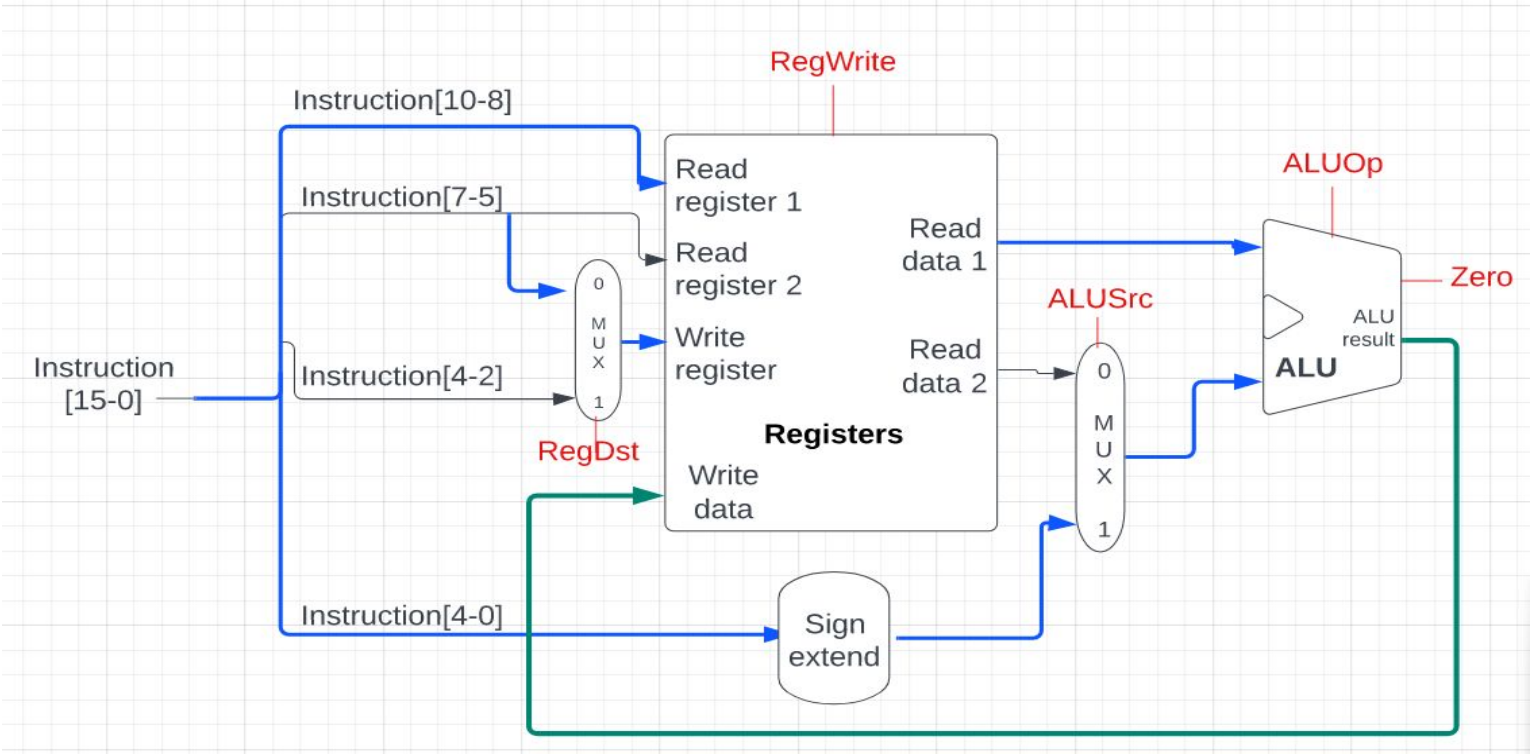
PC



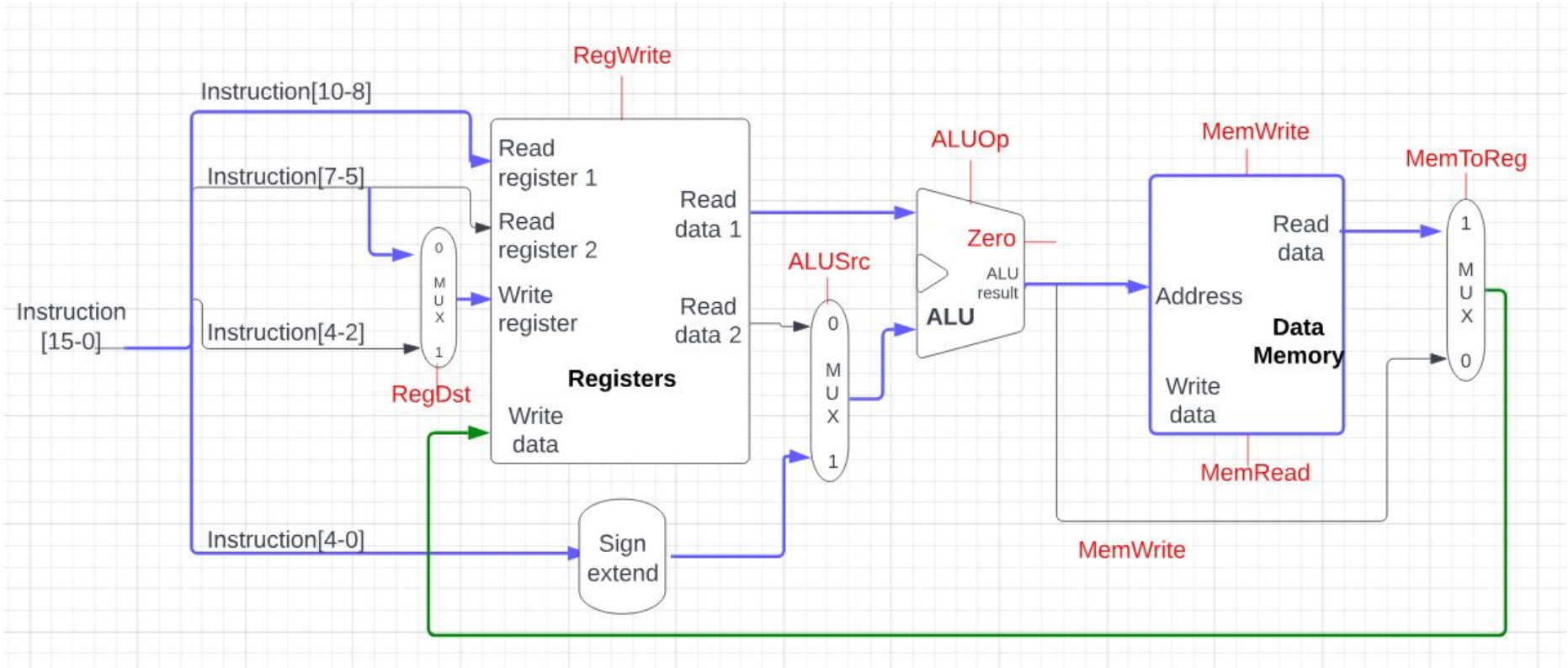
Arithmetic (add $rd, $rs, $rt)



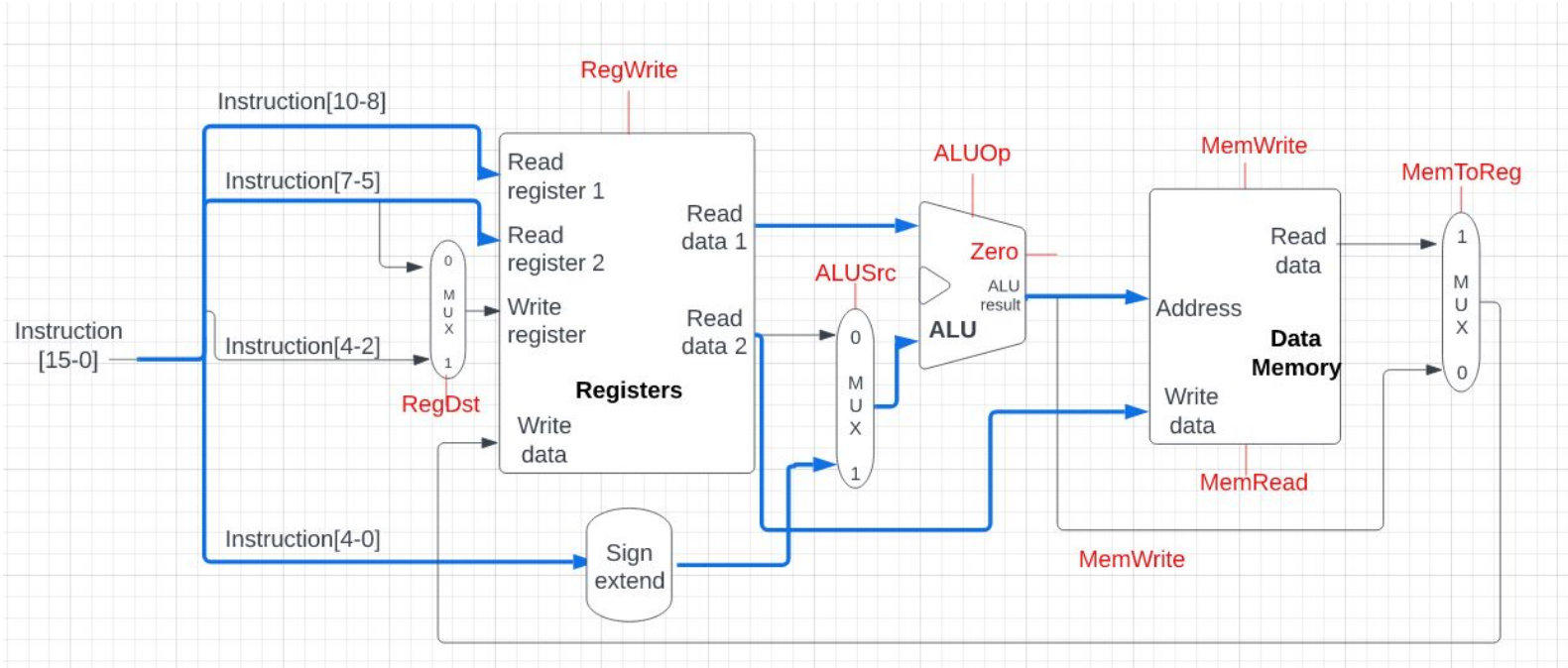
IMMEDIATE (addi $rt, $rs, imm)



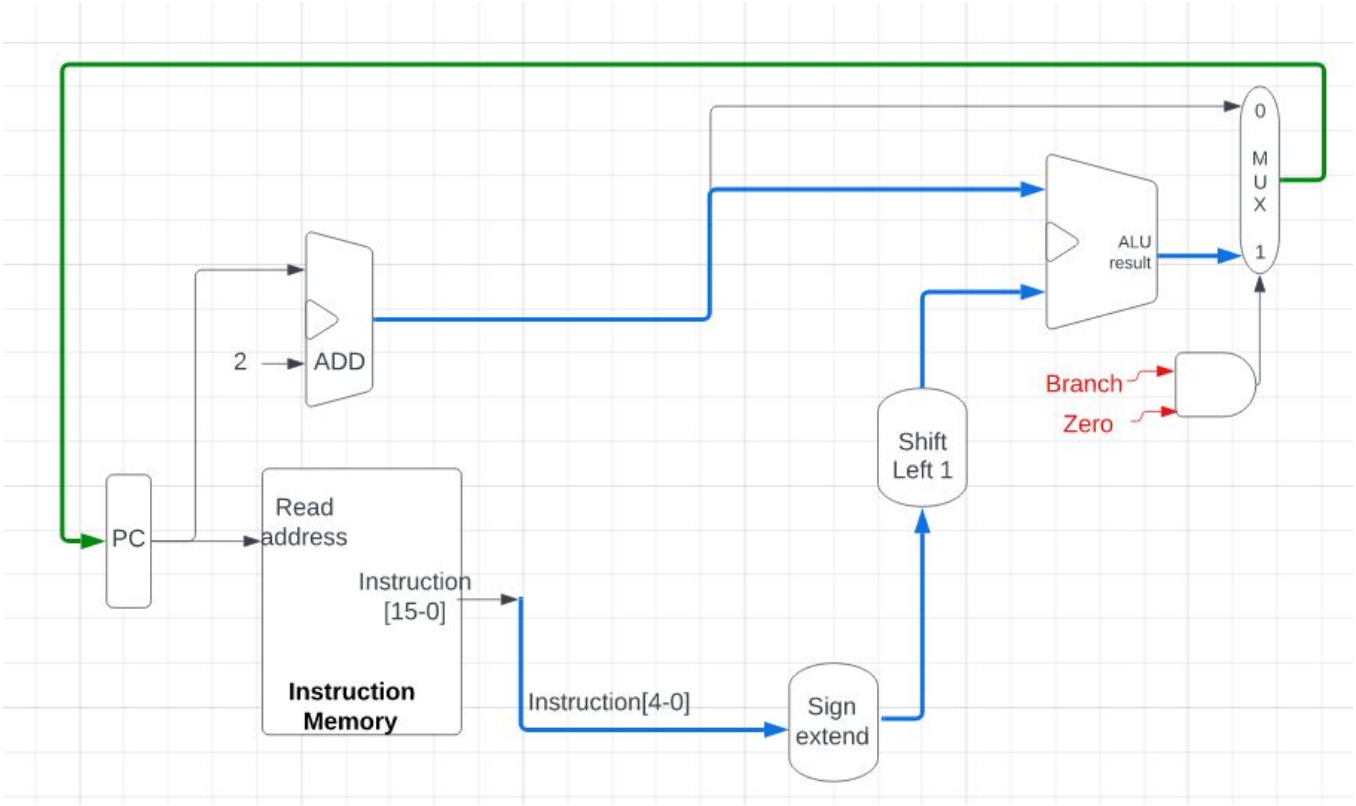
LOAD (lw $rt, imm($rs))



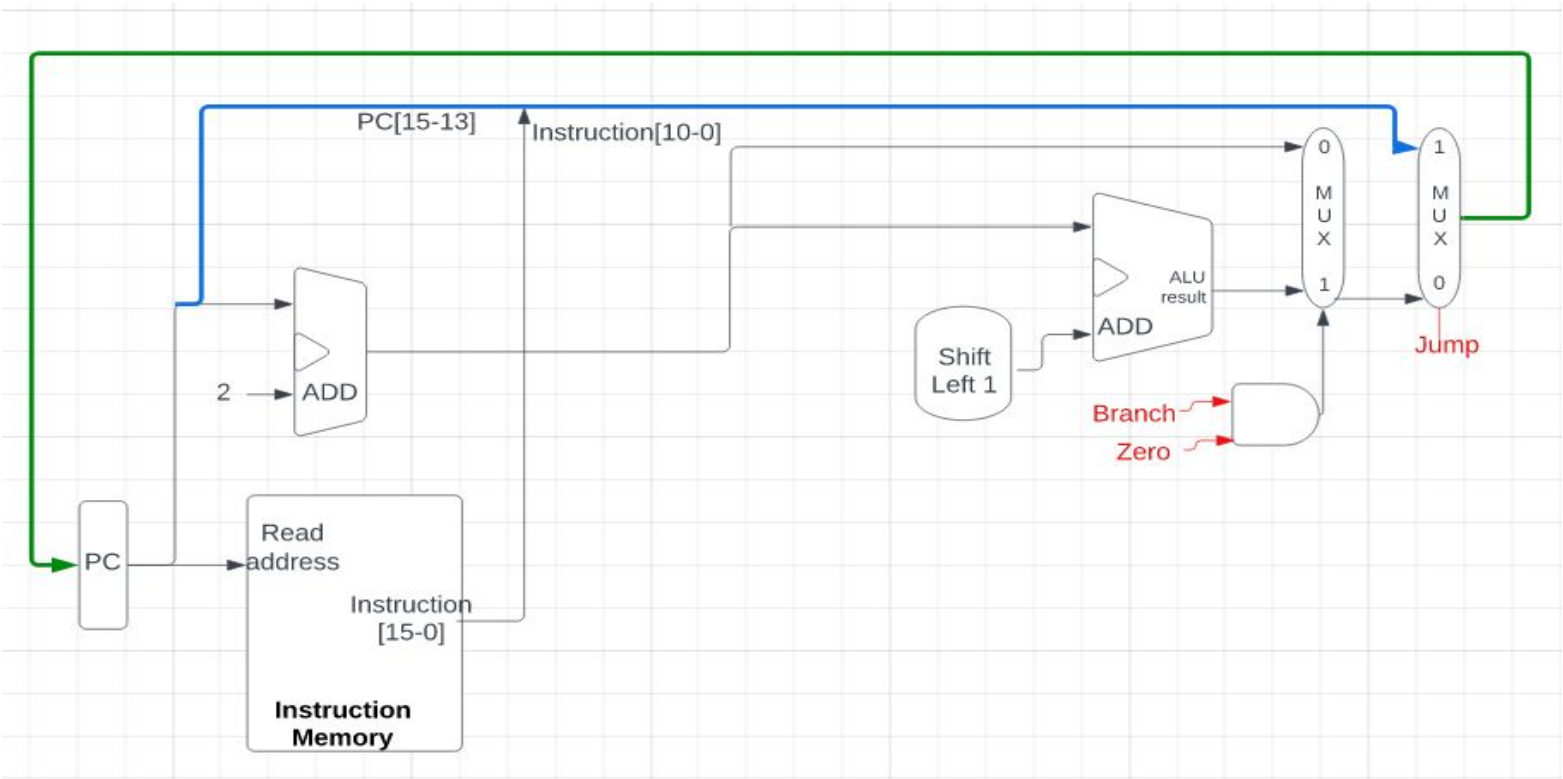
STORE (sw $rt, imm($rs))



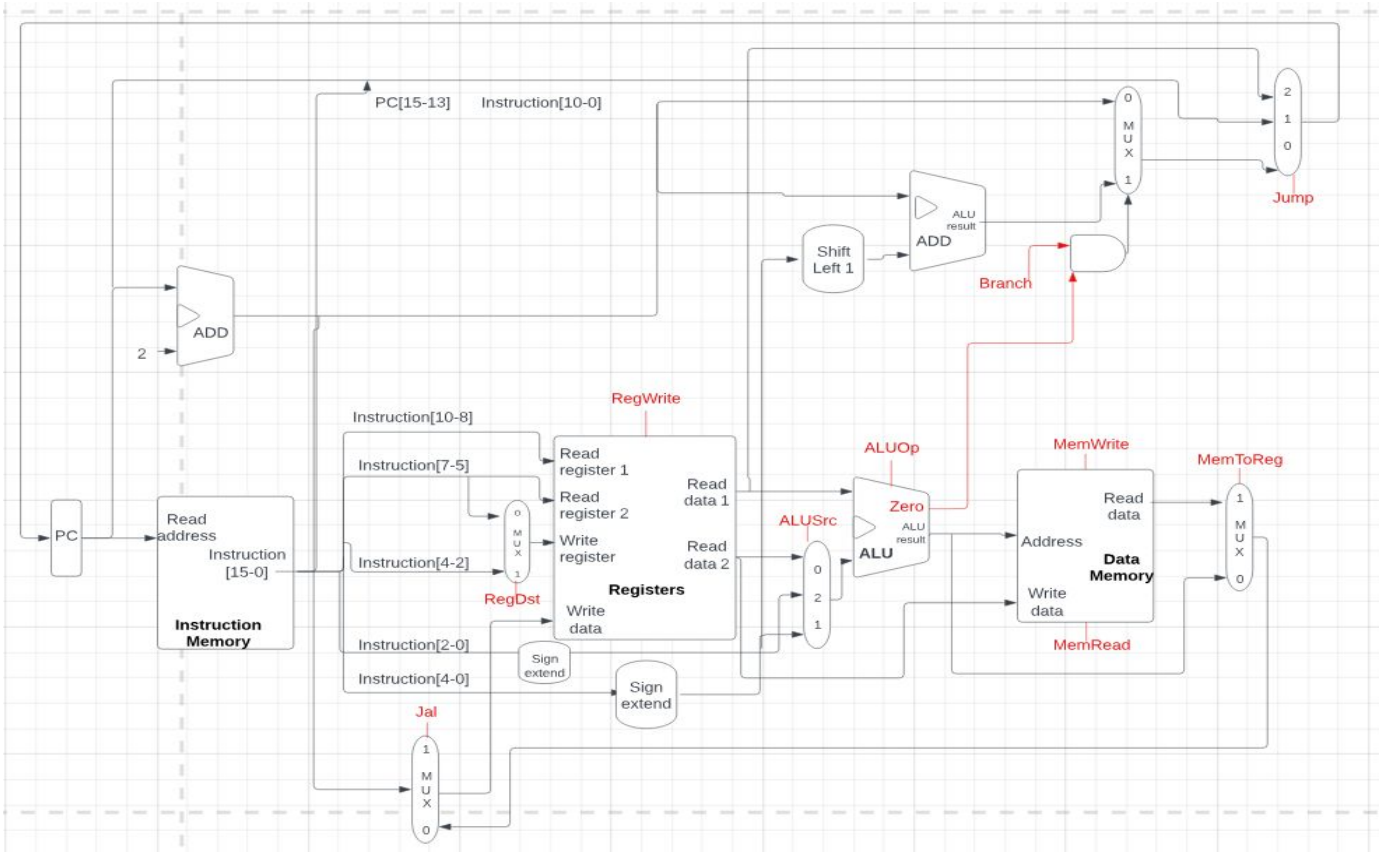
BRANCH (bne $rs, $rt, imm)



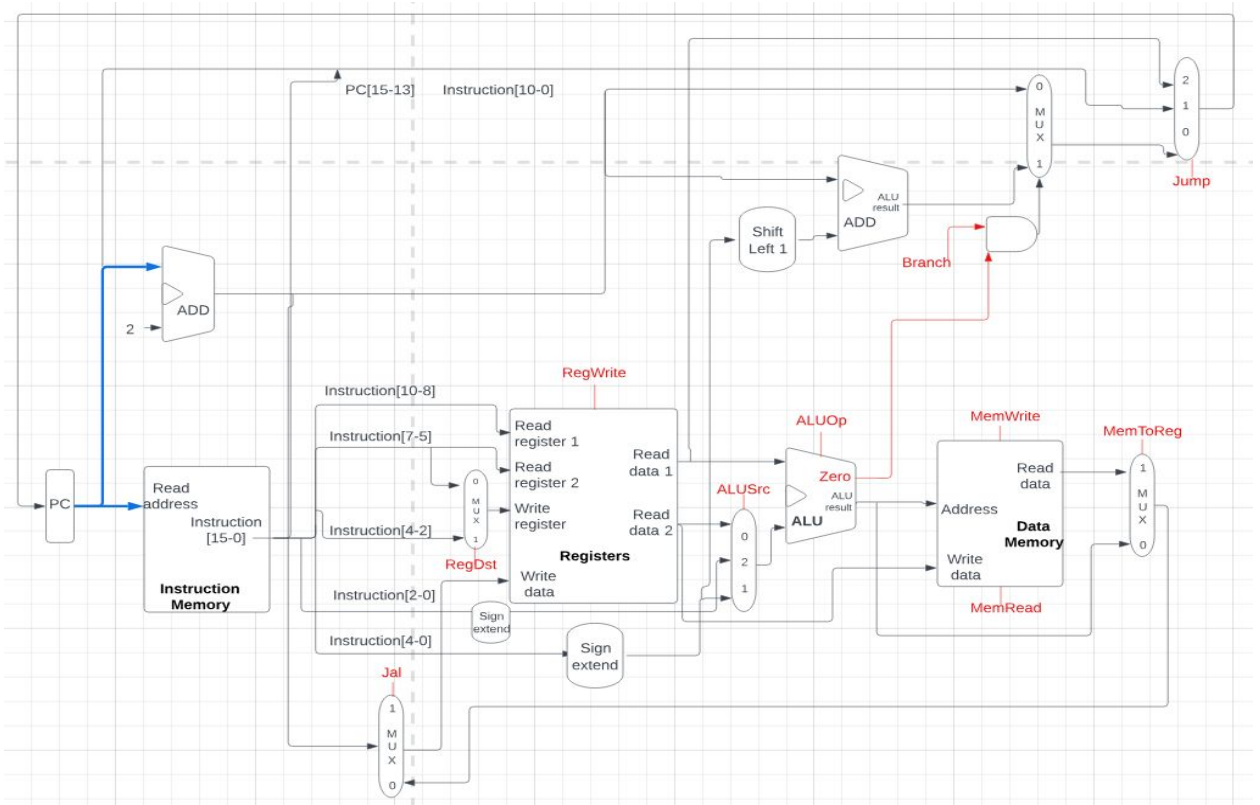
JUMP (j address)



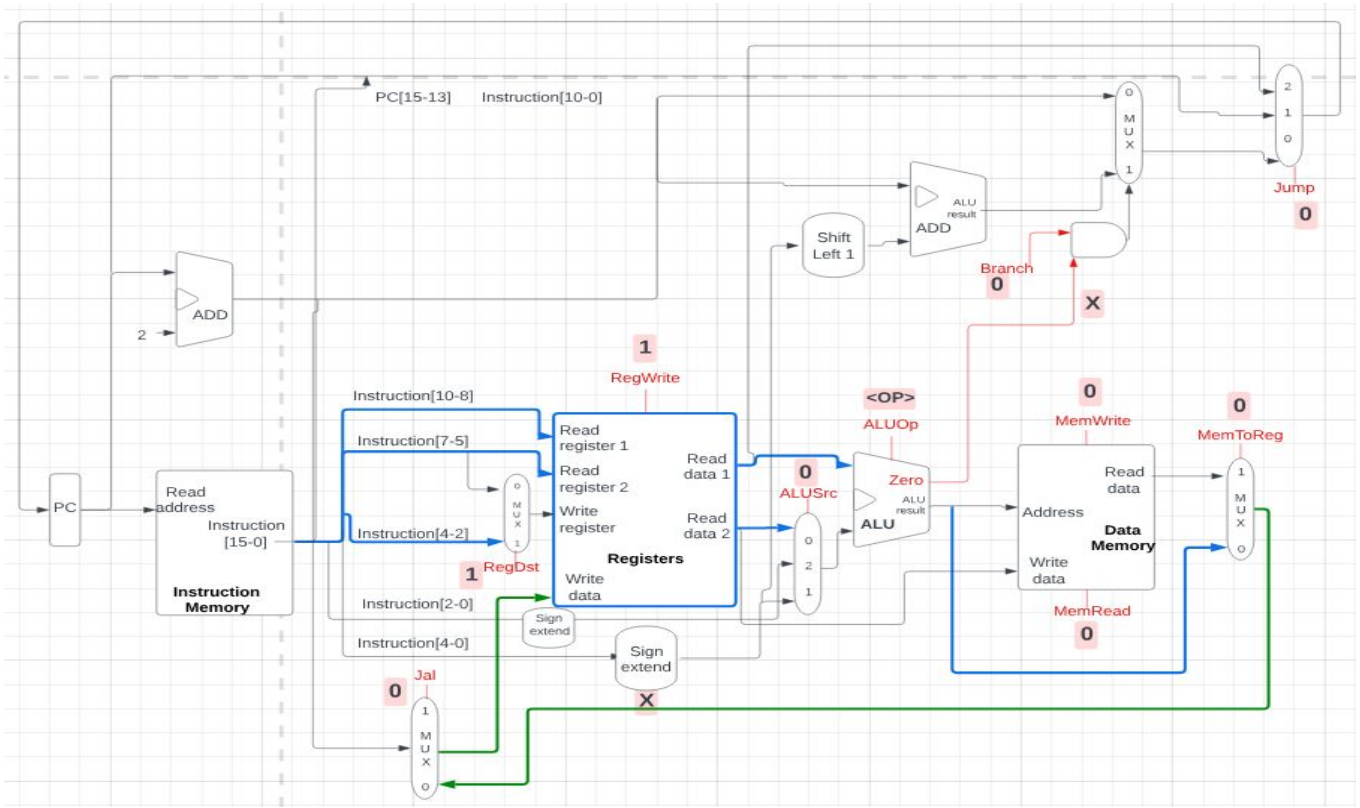
All Together



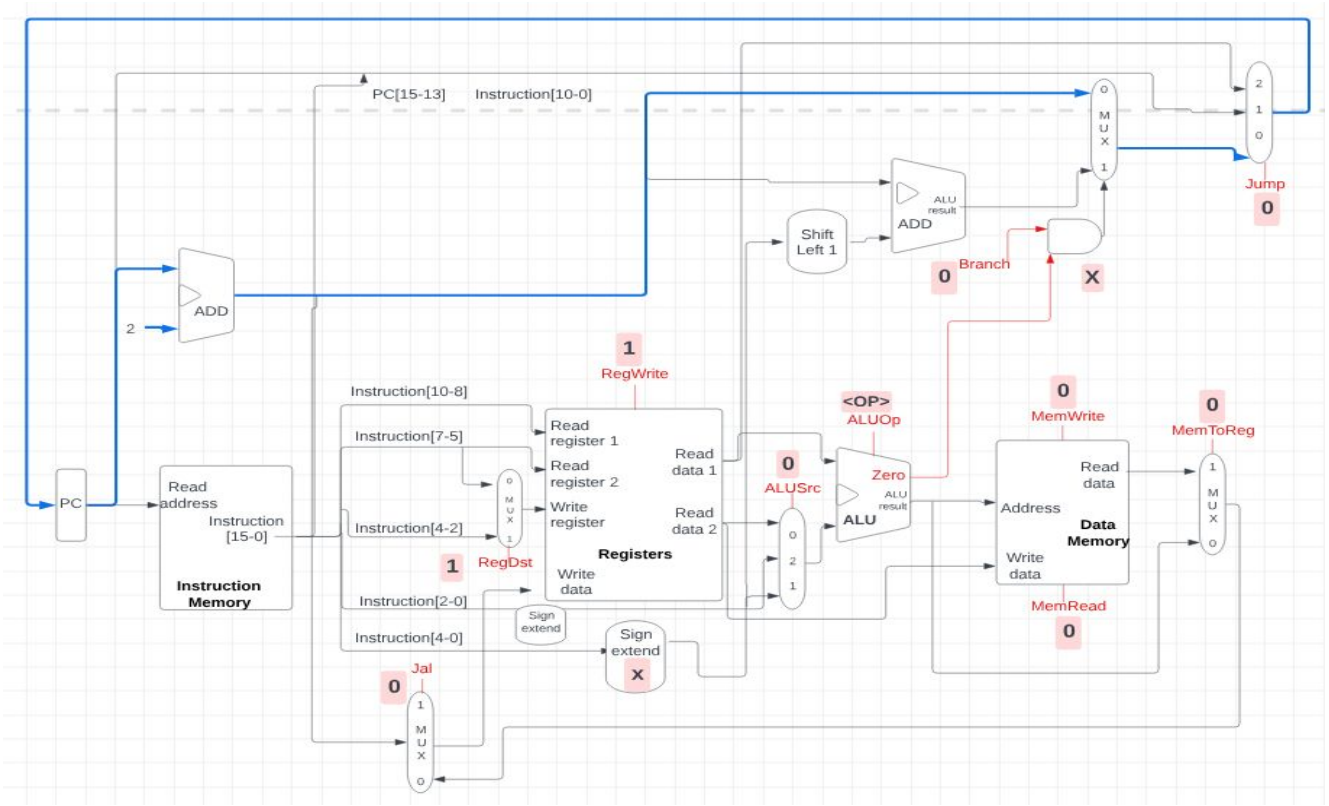
Control for Instruction Fetch



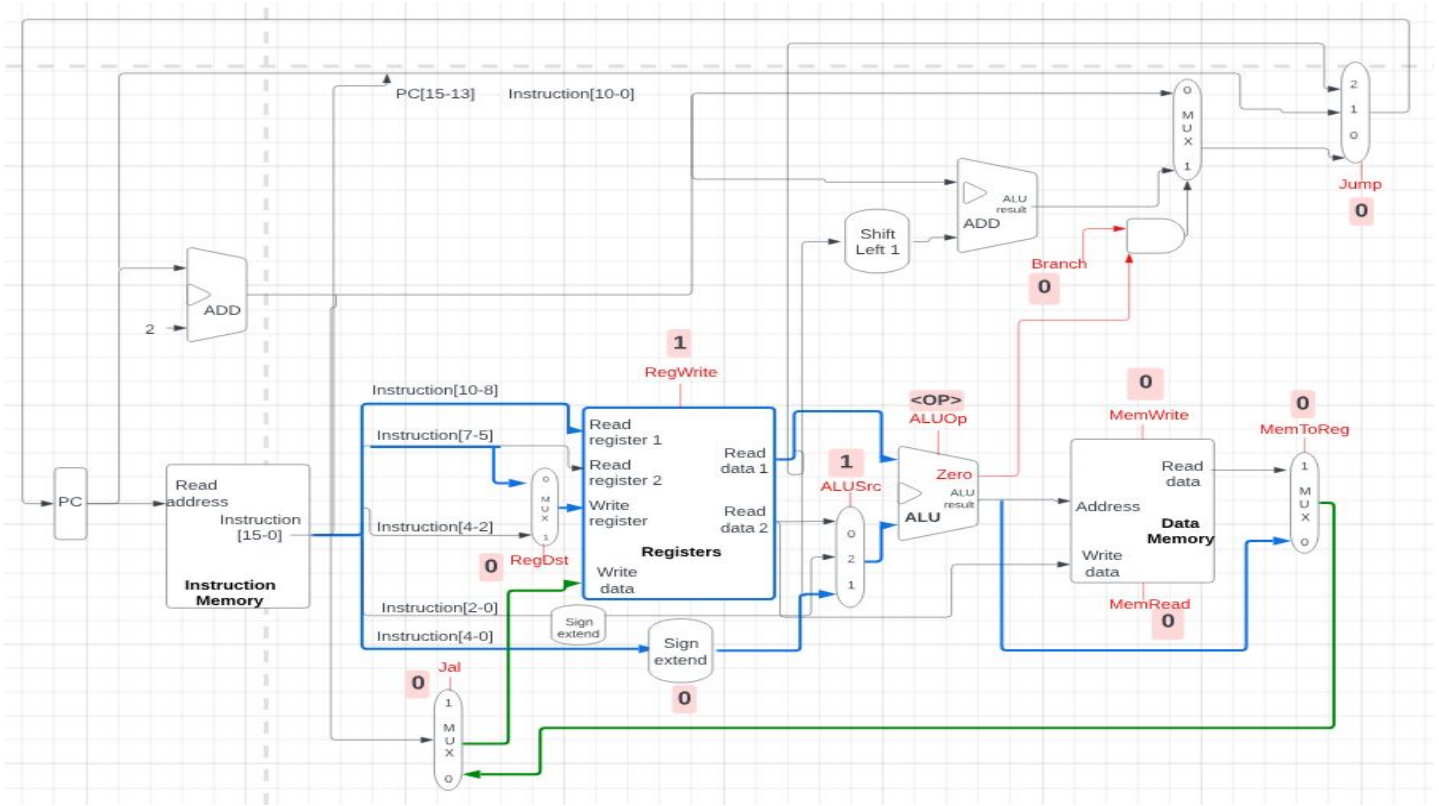
Control for Arithmetic (add $rd, $rs, $rt)



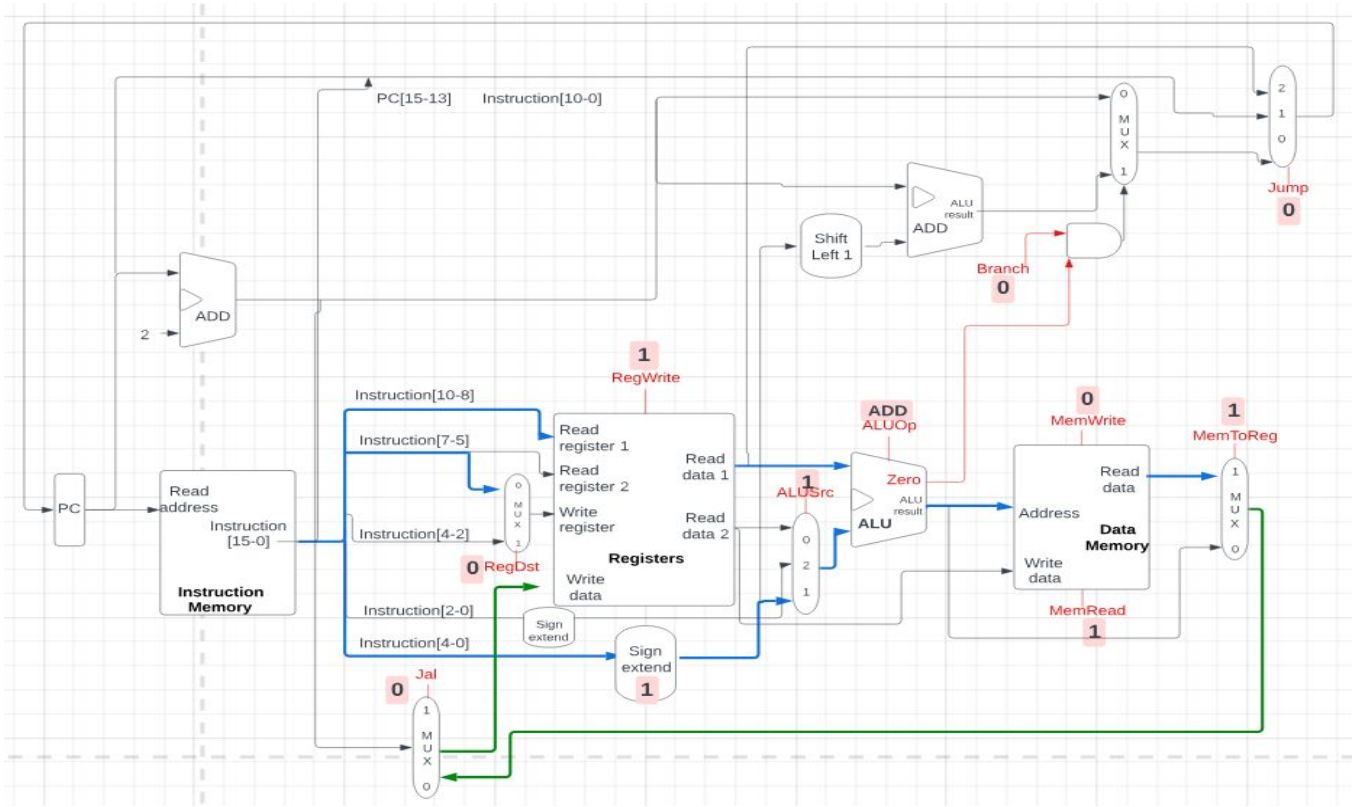
Instruction Fetch at End



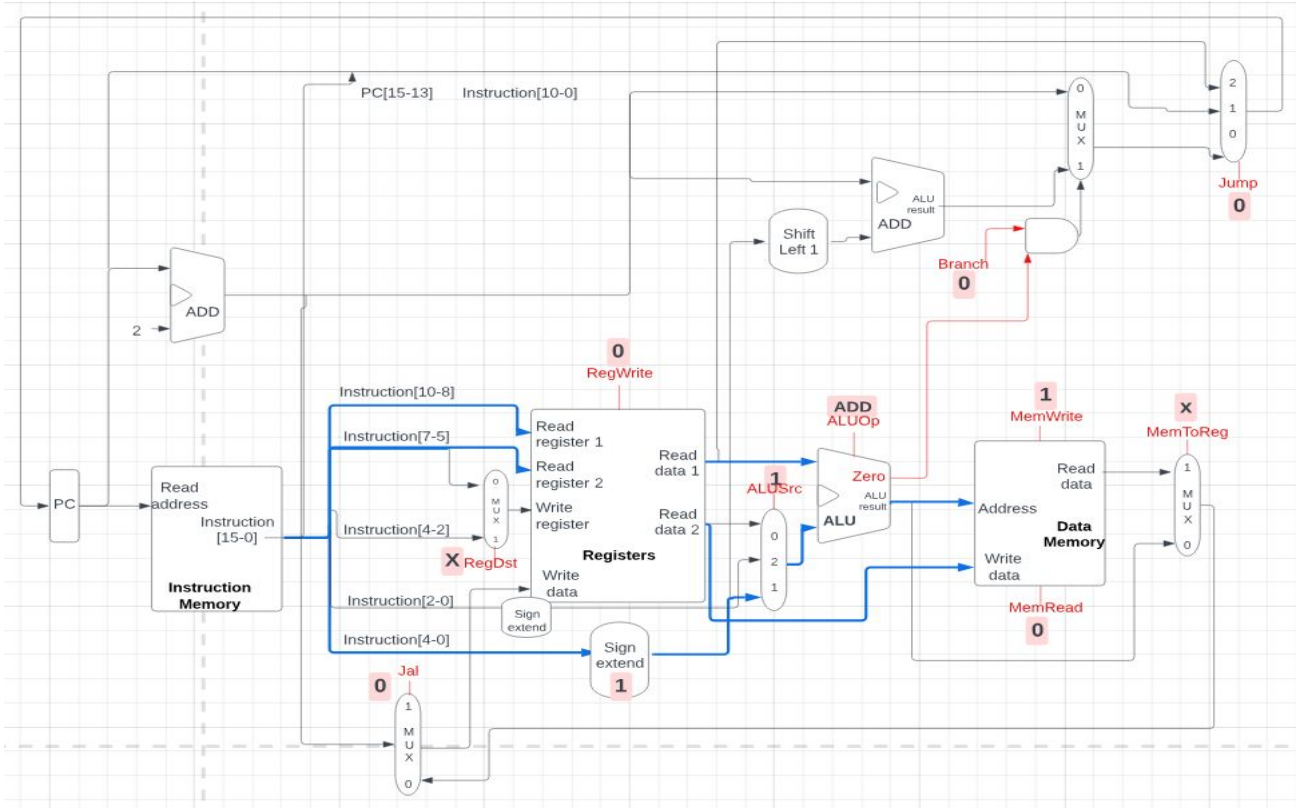
Control for Arithmetic Immediate (addi $rt, $rs, imm)



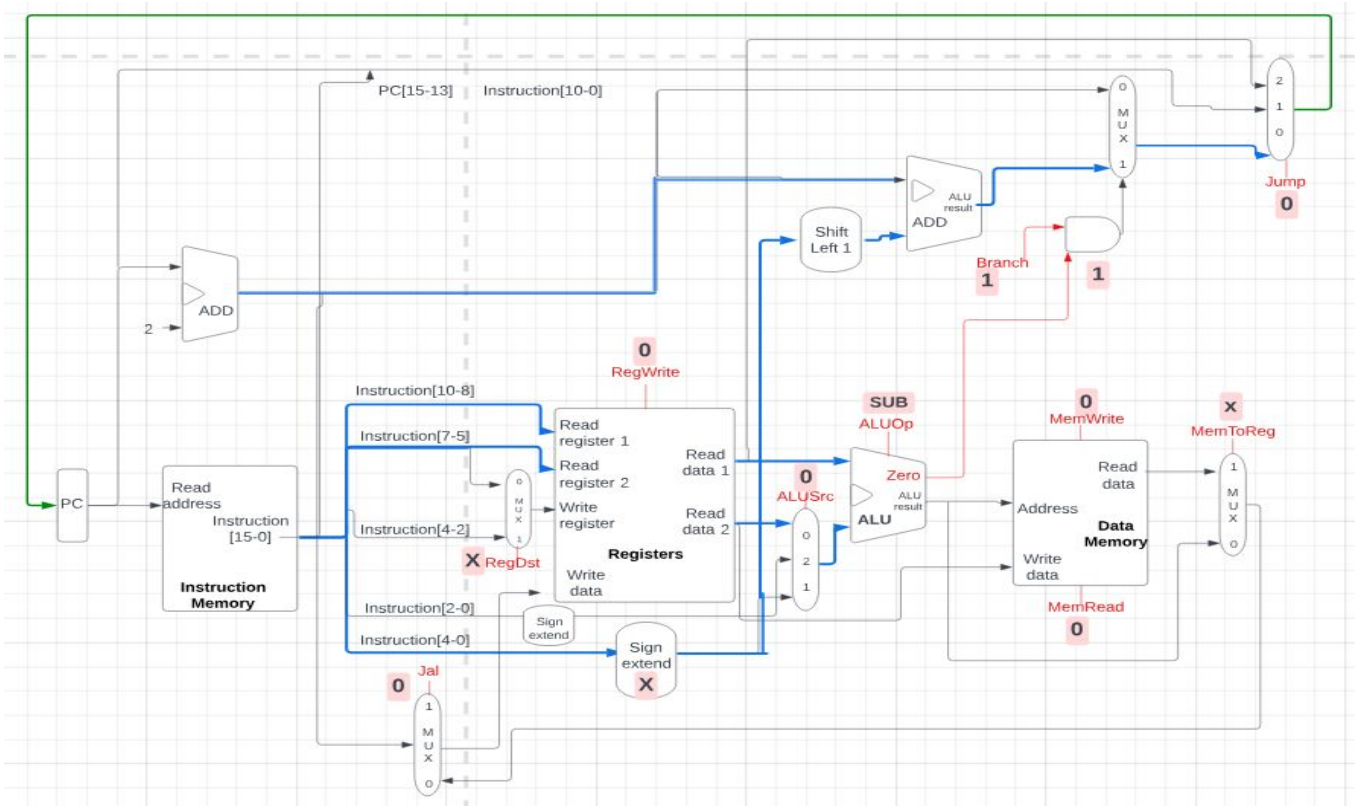
Control for Load (lw $rt, imm($rs))



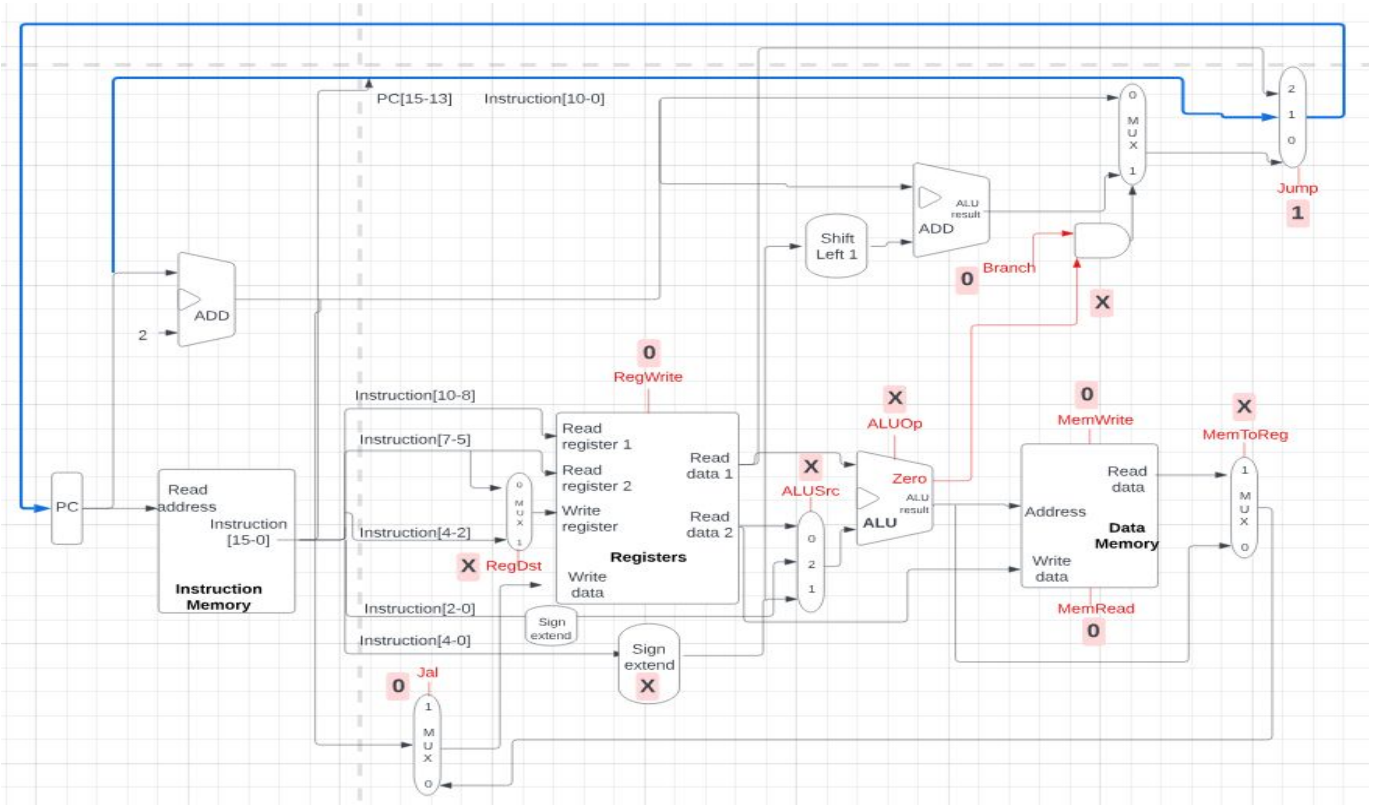
Control for Store (sw $rt, imm($rs))



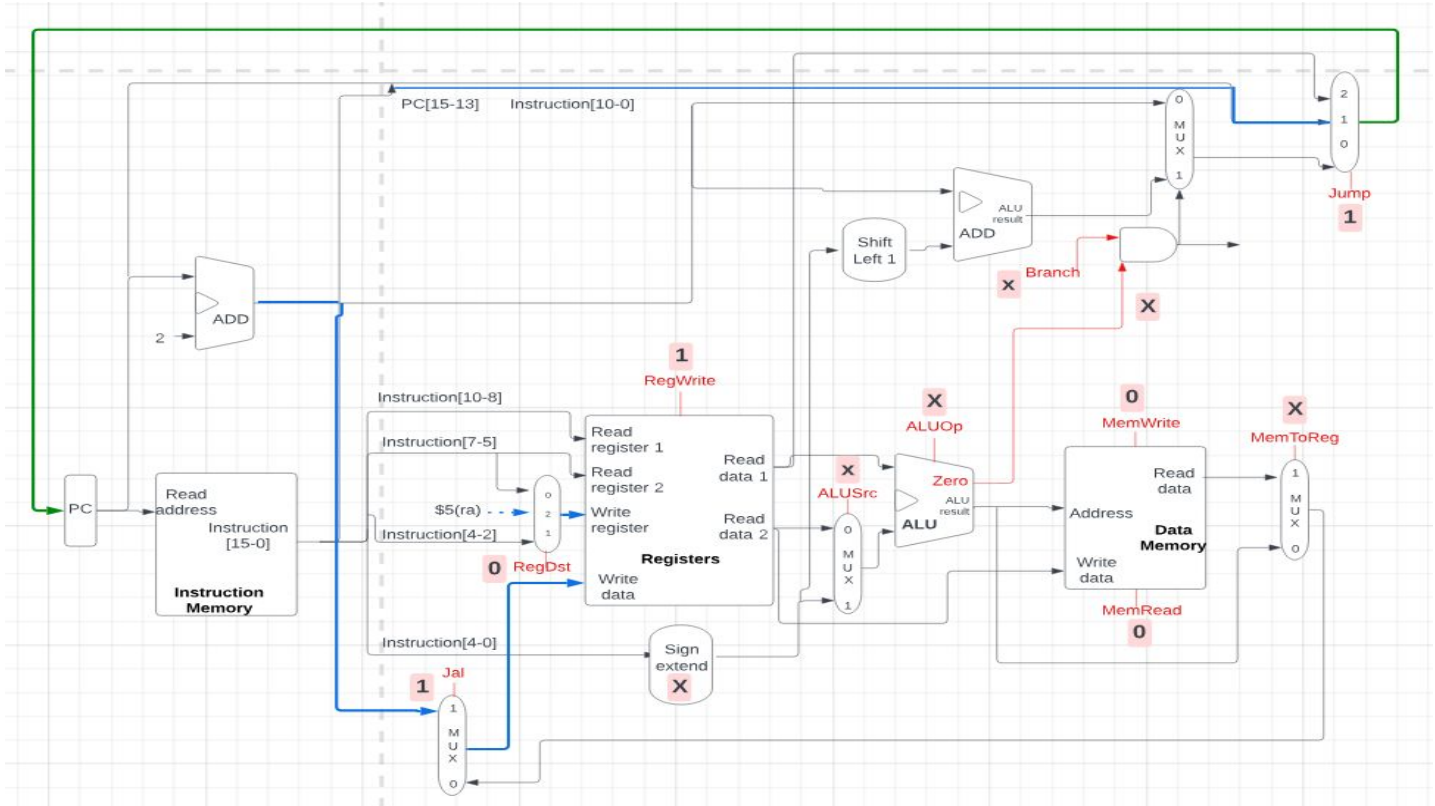
Control for Branch (bne $rs, $rt, imm)



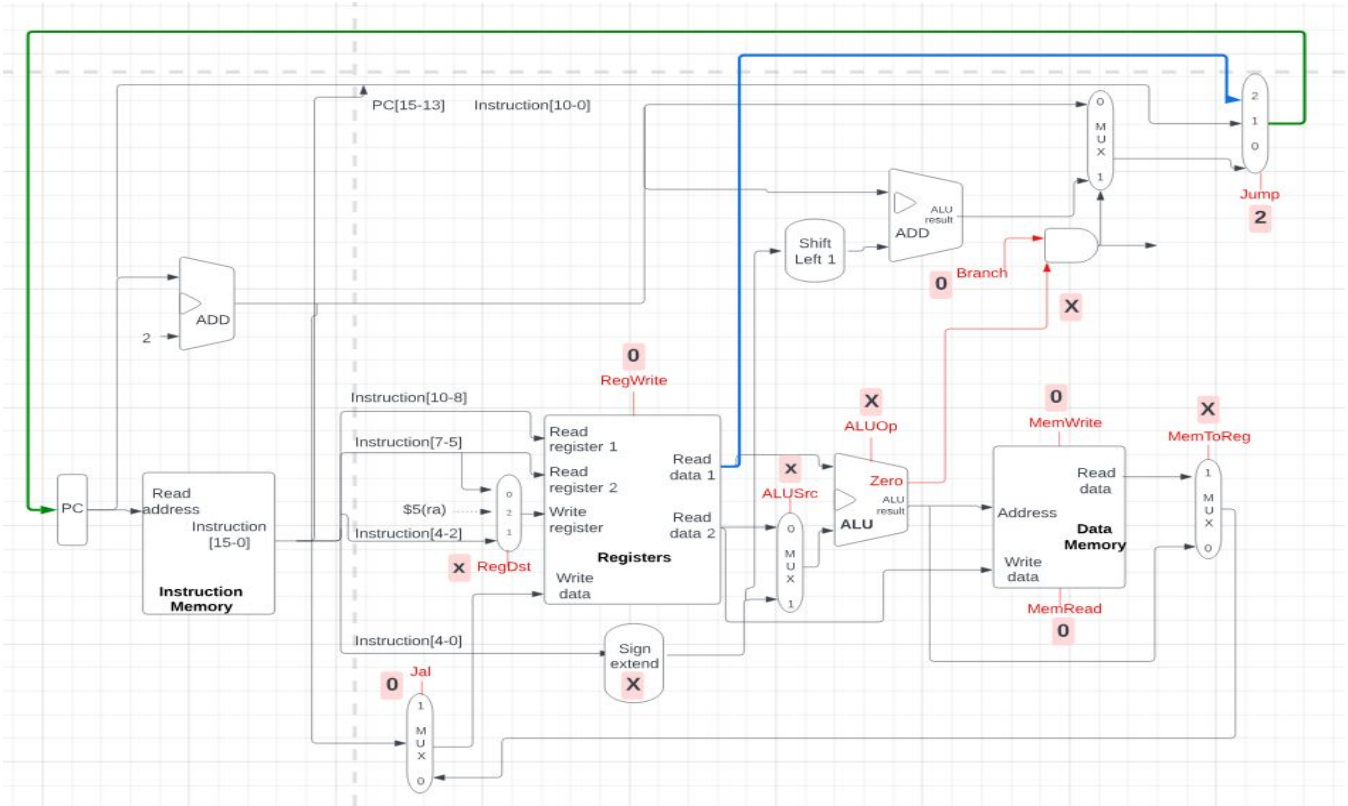
Control for Jump (j address)



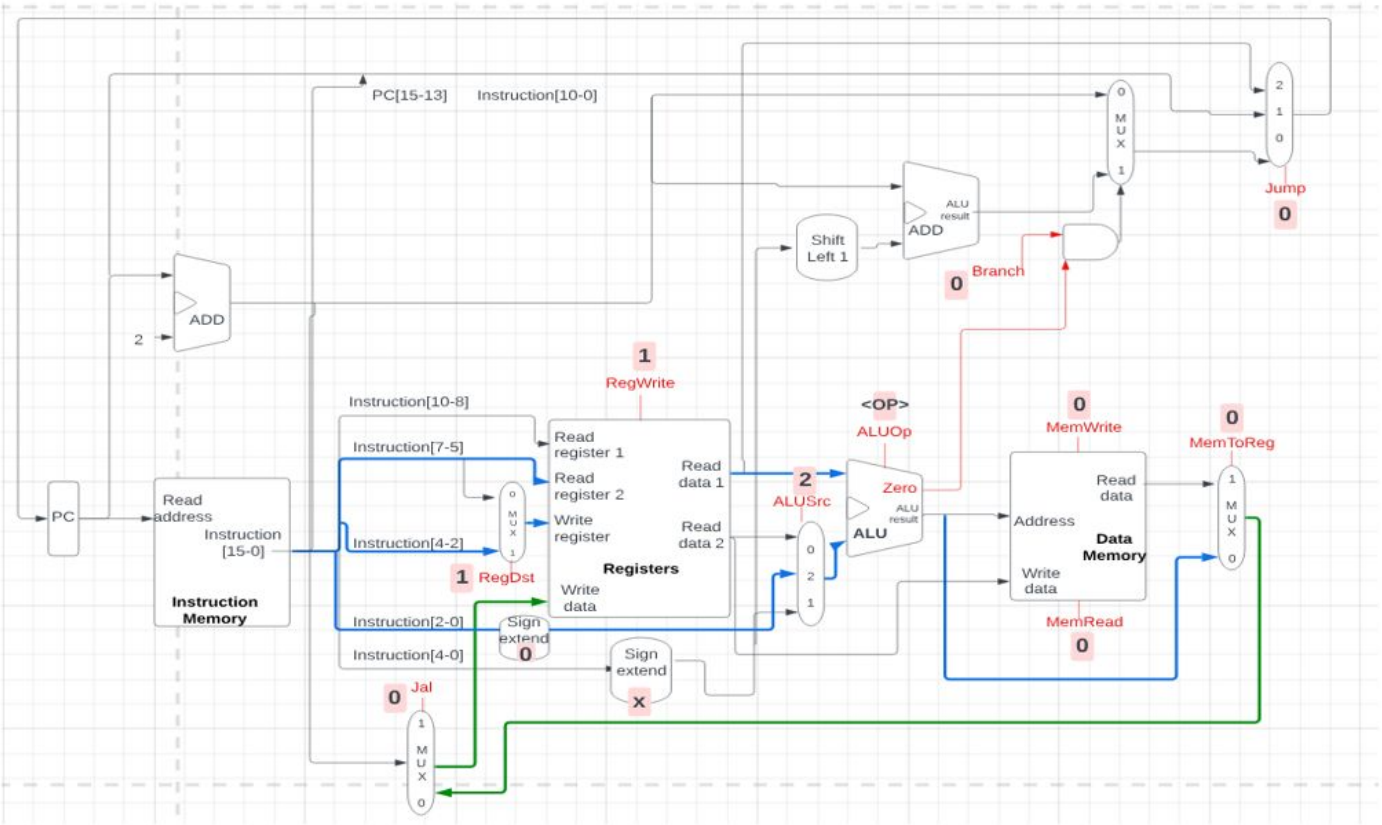
JAL (jal address)

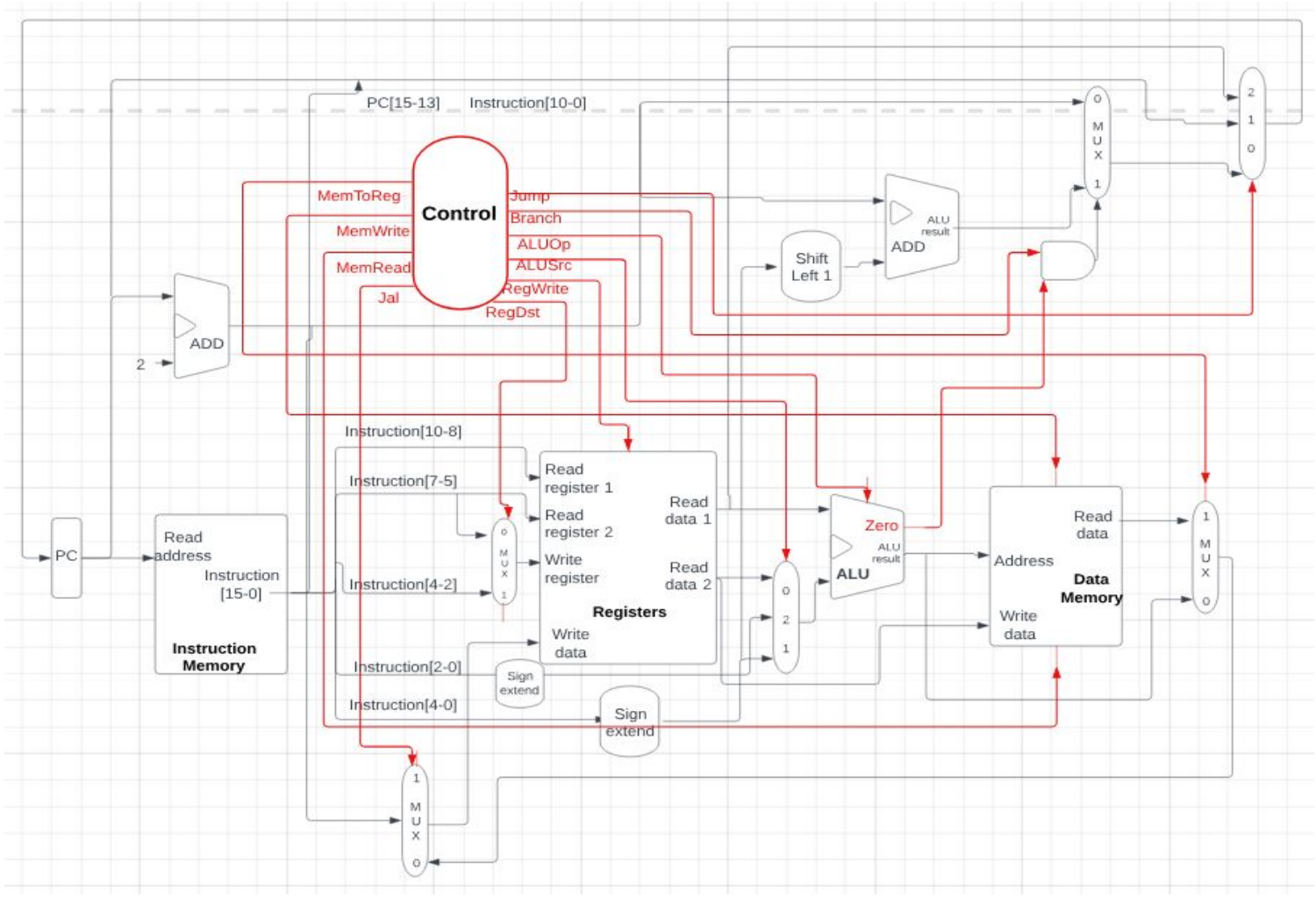


JR (jr $ra)


PC[15-13]
Instruction[10-0]
Jump
2
ADD
2
Shift Left 1
ADD
ALU result
0
Branch
X
0
RegWrite
Instruction[10-8]
Instruction[7-5]
Instruction[4-2]
$5(ra)
X
RegDst
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
X
ALUOp
X
ALUSrc
Zero
ALU result
ALU
0
MemWrite
X
MemToReg
Read data
Address
Data Memory
Write data
MemRead
0
PC
Read address
Instruction [15-0]
Instruction Memory
Instruction[4-0]
Sign extend
X
0
Jal
MUX

SHIFT AMOUNT (sll $rd, $rt, shamt)




PC[15-13]
Instruction[10-0]
Control
MemToReg
MemWrite
MemRead
Jal
Jump
Branch
ALUOp
ALUSrc
RegWrite
RegDst
ADD
2
Shift Left 1
ALU result
ADD
M U X
0
1
2
1
0
PC
Read address
Instruction [15-0]
Instruction Memory
Instruction[10-8]
Instruction[7-5]
Instruction[4-2]
Instruction[2-0]
Instruction[4-0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
Sign extend
Sign extend
Zero
ALU result
ALU
Address
Read data
Write data
Data Memory

THANK YOU!