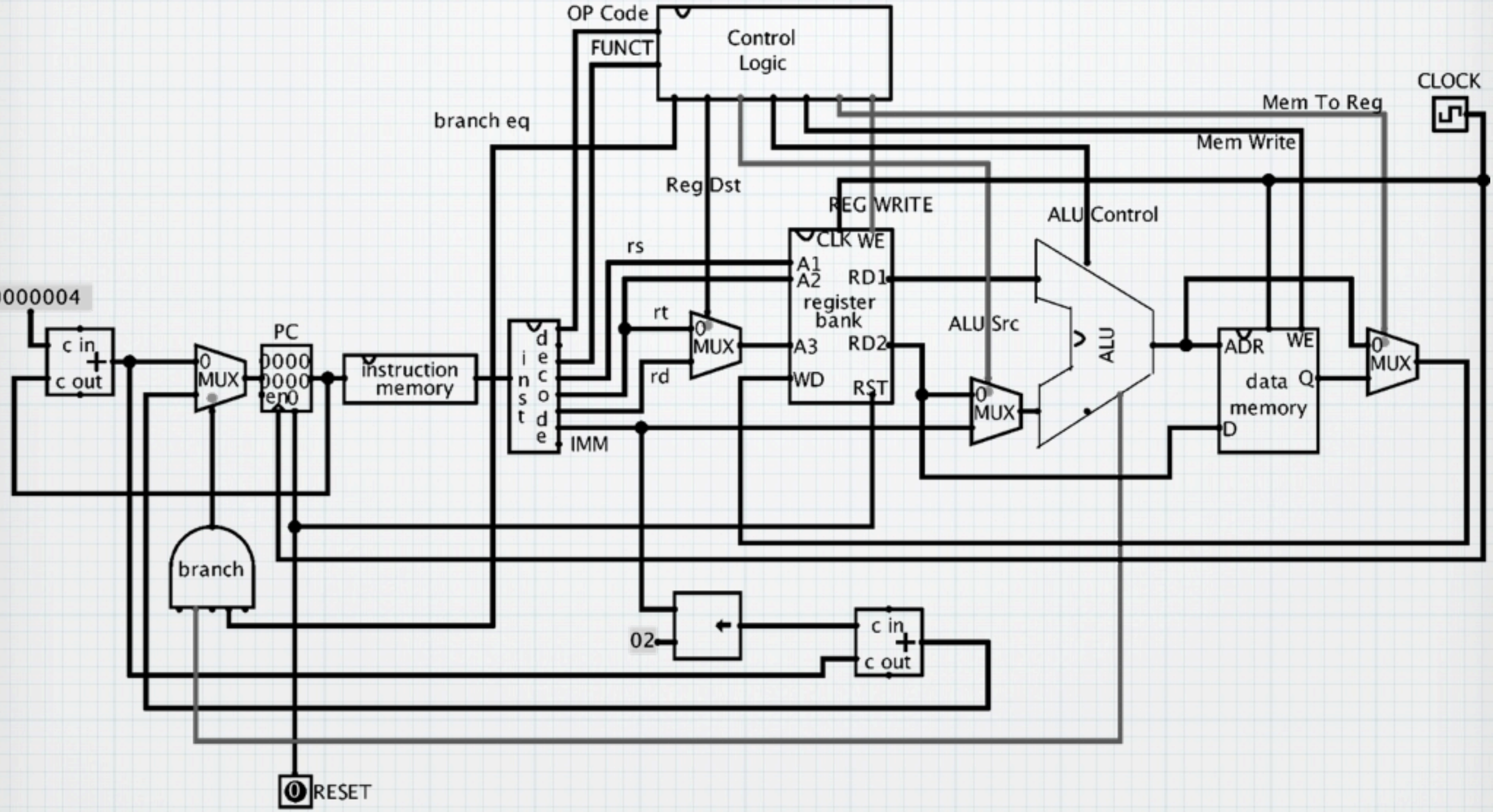
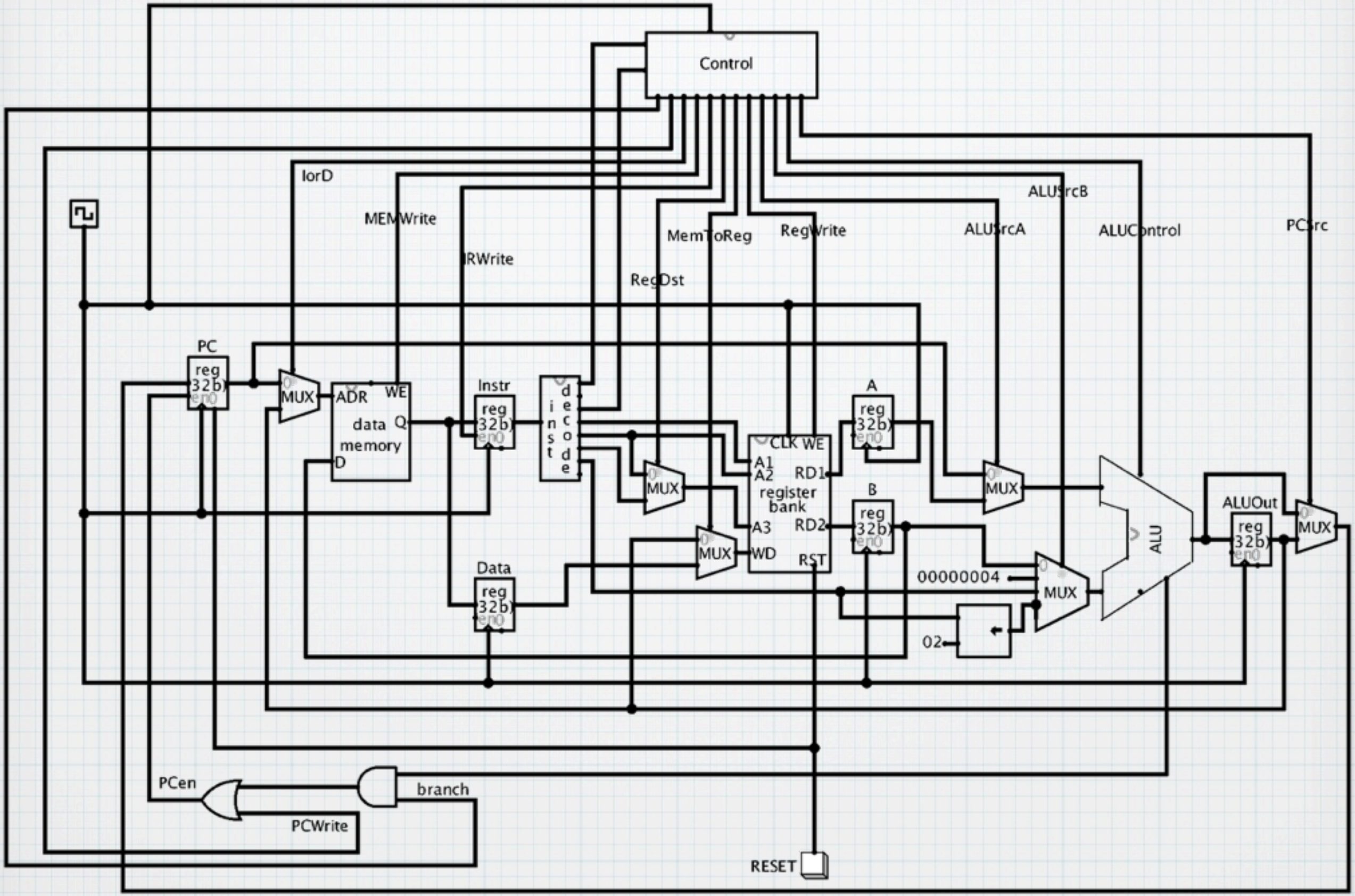
Announcements

TA Application Time

TEPID Handout Updated

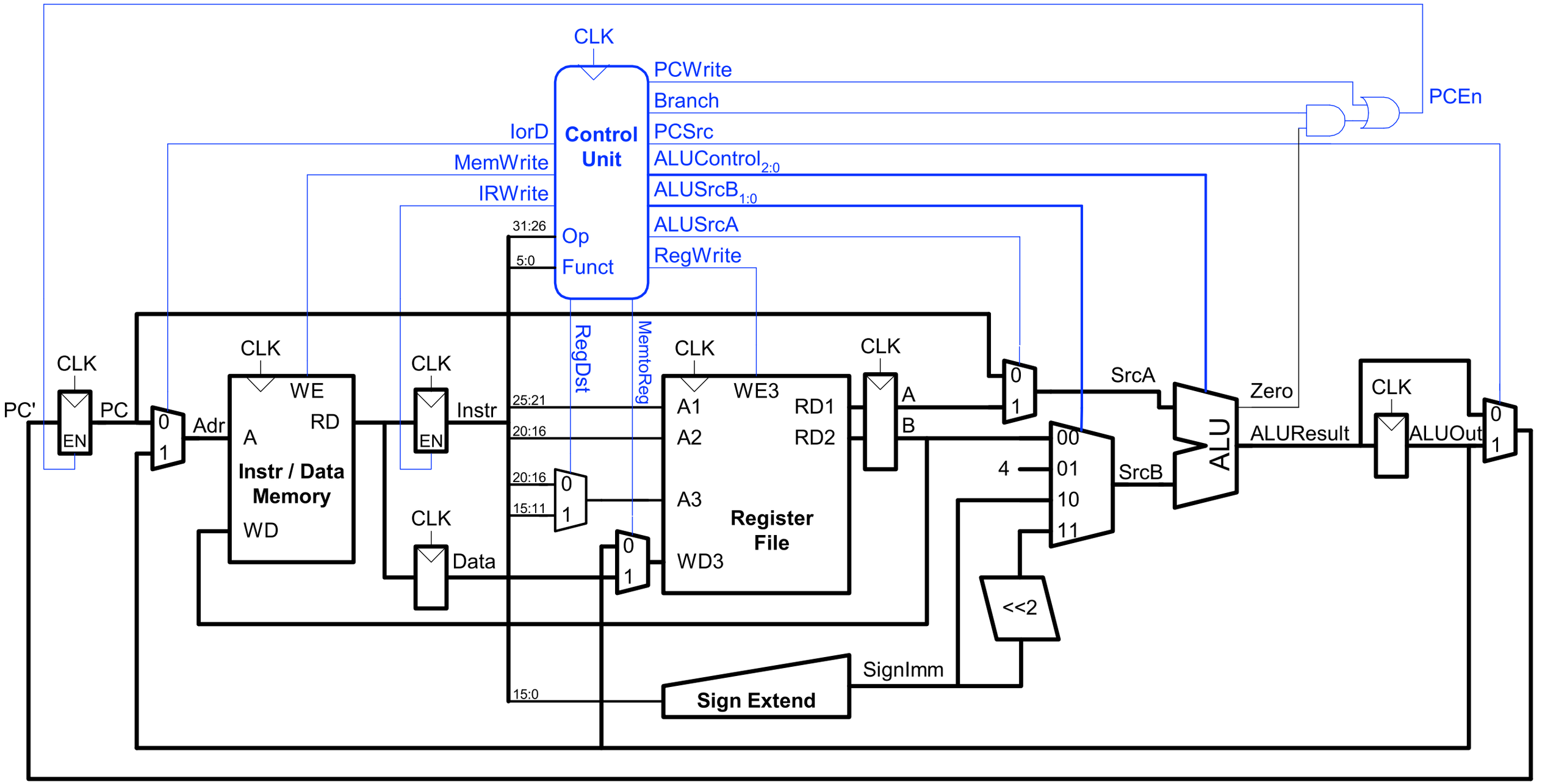

OP Code
FUNCT
Control Logic
branch eq
Mem To Reg
CLOCK
Mem Write
Reg Dst
REG WRITE
ALU Control
rs
CLK WE
A1
A2
RD1
register bank
A3
RD2
WD
RST
rt
rd
0
MUX
ALU Src
ALU
ADR
WE
data
Q
memory
D
PC
c in
c out
instruction memory
inst decode
IMM
branch
02
RESET


Control
IorD
MEMWrite
IRWrite
RegDst
MemToReg
RegWrite
ALUSrcA
ALUSrcB
ALUControl
PCSrc
PC
reg
32b
MUX
ADR
WE
data
Q
memory
D
Instr
inst
decode
A
B
CLK
WE
A1
A2
RD1
register
bank
A3
RD2
WD
RST
Data
00000004
02
ALU
ALUOut
PCen
branch
PCWrite
RESET

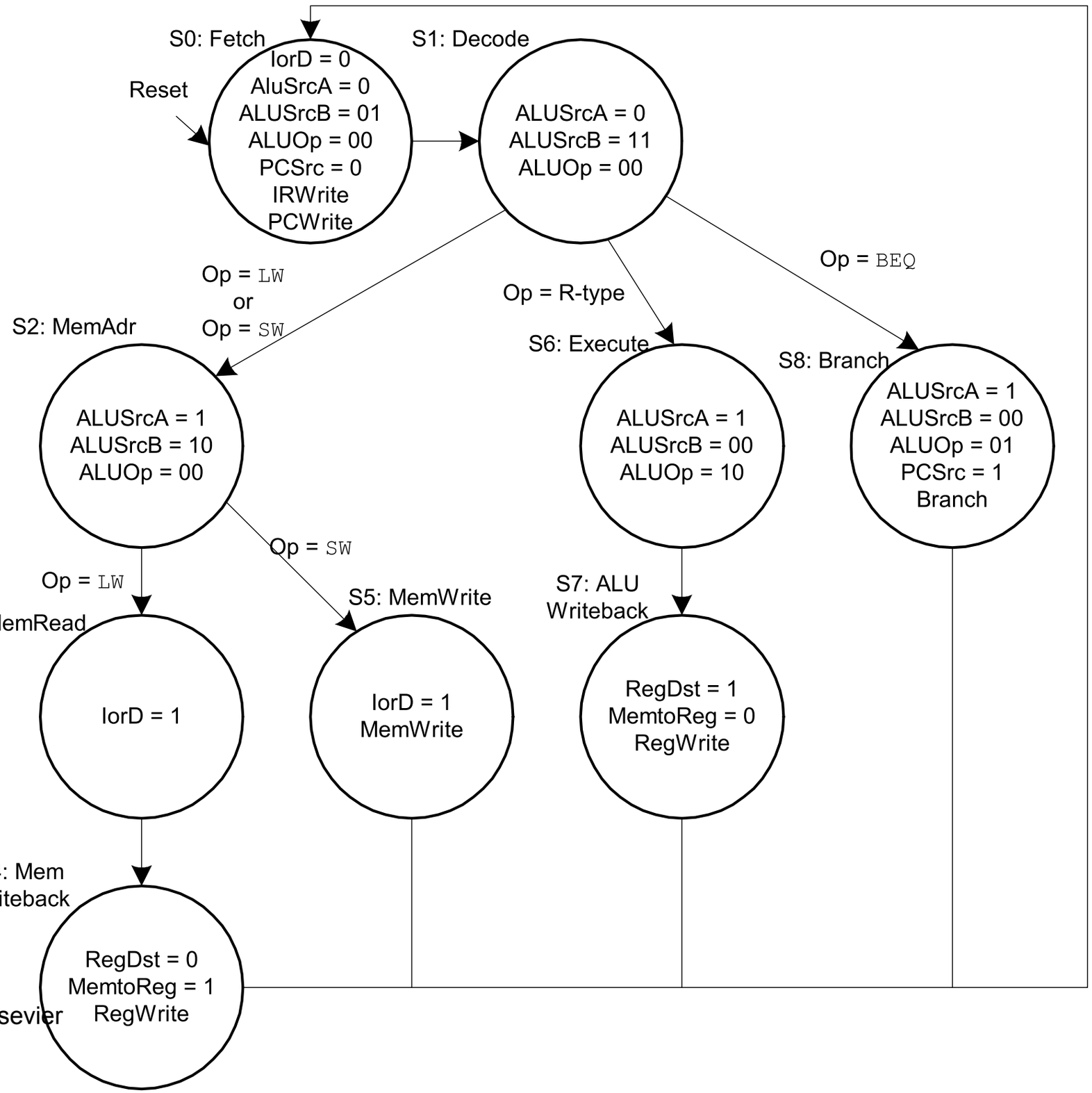
The Five Cycles of MIPS

* Instruction Fetch
* Instruction decode + Register fetch
* Execute | Memory address | Branch
* Memory access | R-type completion
* Writeback

Complete Multicycle Processor



Complete Multicycle Controller FSM



CYCLE 0

IR:= Memory[PC]
PC:= PC+4

CYCLE 1

A:= Reg[IR[25:21]]
B:= Reg[IR[20:16]]
ALUout := PC + sign-extend(IR[15:0]]

CYCLE 2

ALUout:= A + IR[15:0]

if A=B then
PC := ALUout

ALUout:= A op B

CYCLE 3

SW:
Memory[ALUout]:= B

LW:
MDR:= Memory[ALUout]

Reg[IR[15:11]]:= ALUout

CYCLE 4

Reg[[20:16]]:= MDR

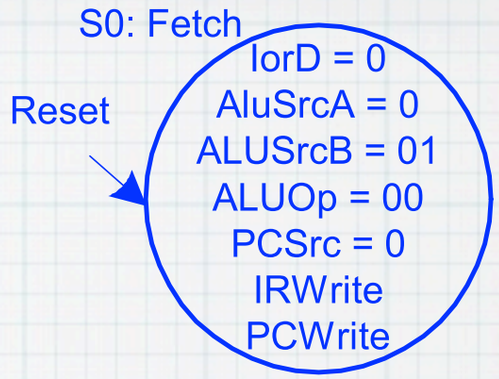
LW AND SW

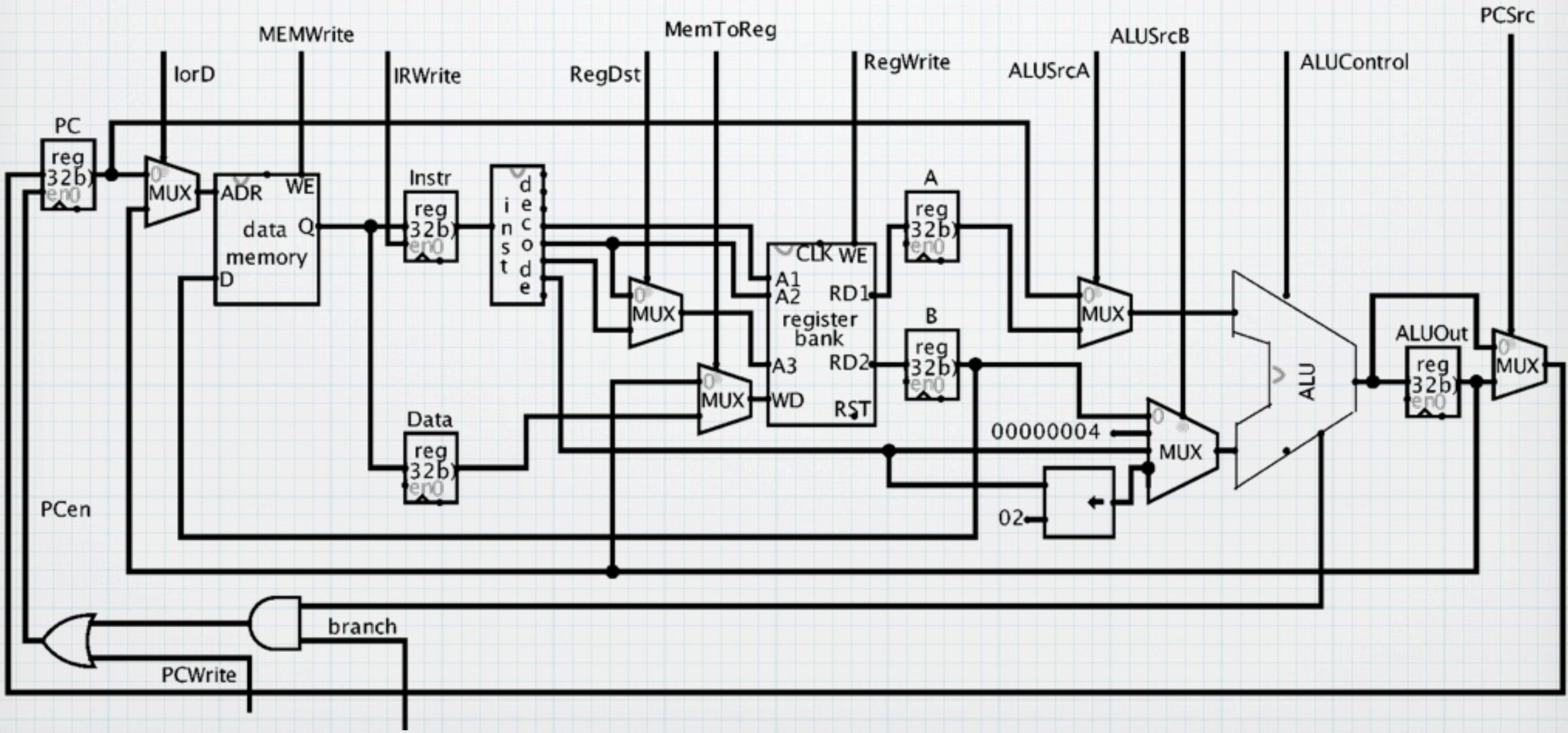
BRANCH EQ

R-TYPE

* Instruction Fetch

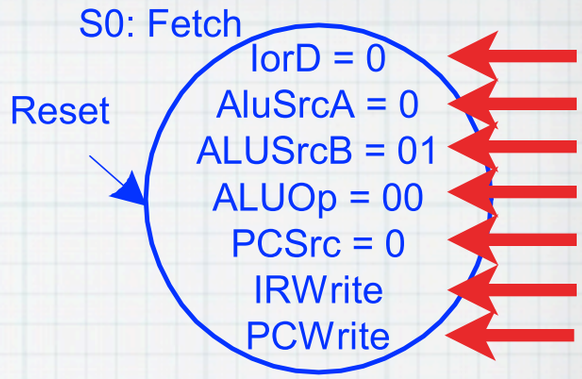
IR:= Memory[PC]
PC:= PC+4

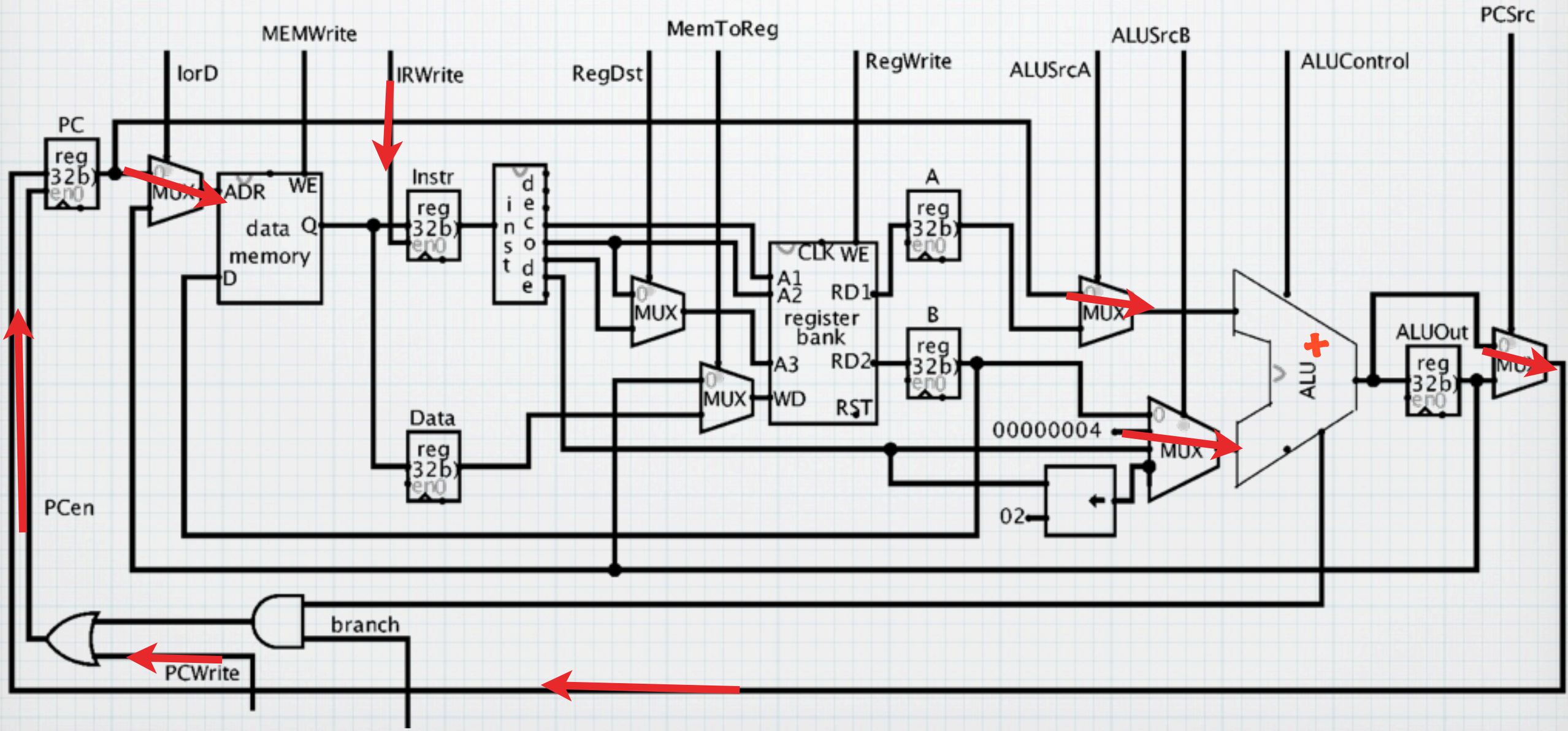




* Instruction Fetch

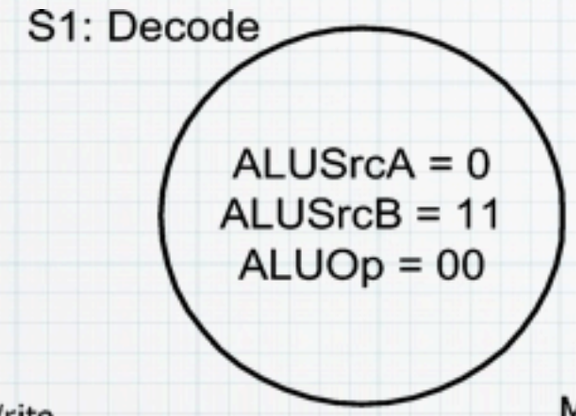
IR:= Memory[PC]
PC:= PC+4

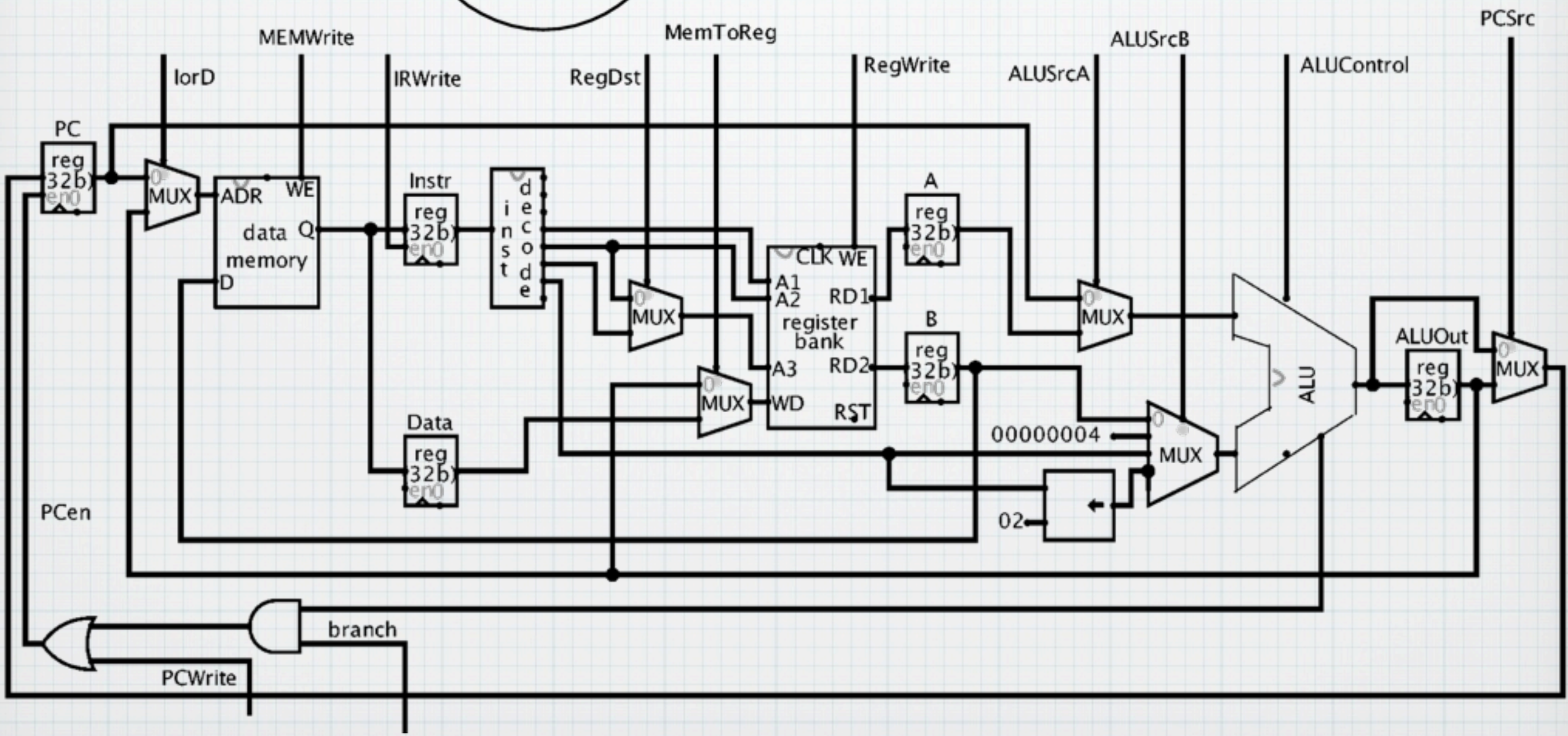




Instruction decode and Register fetch

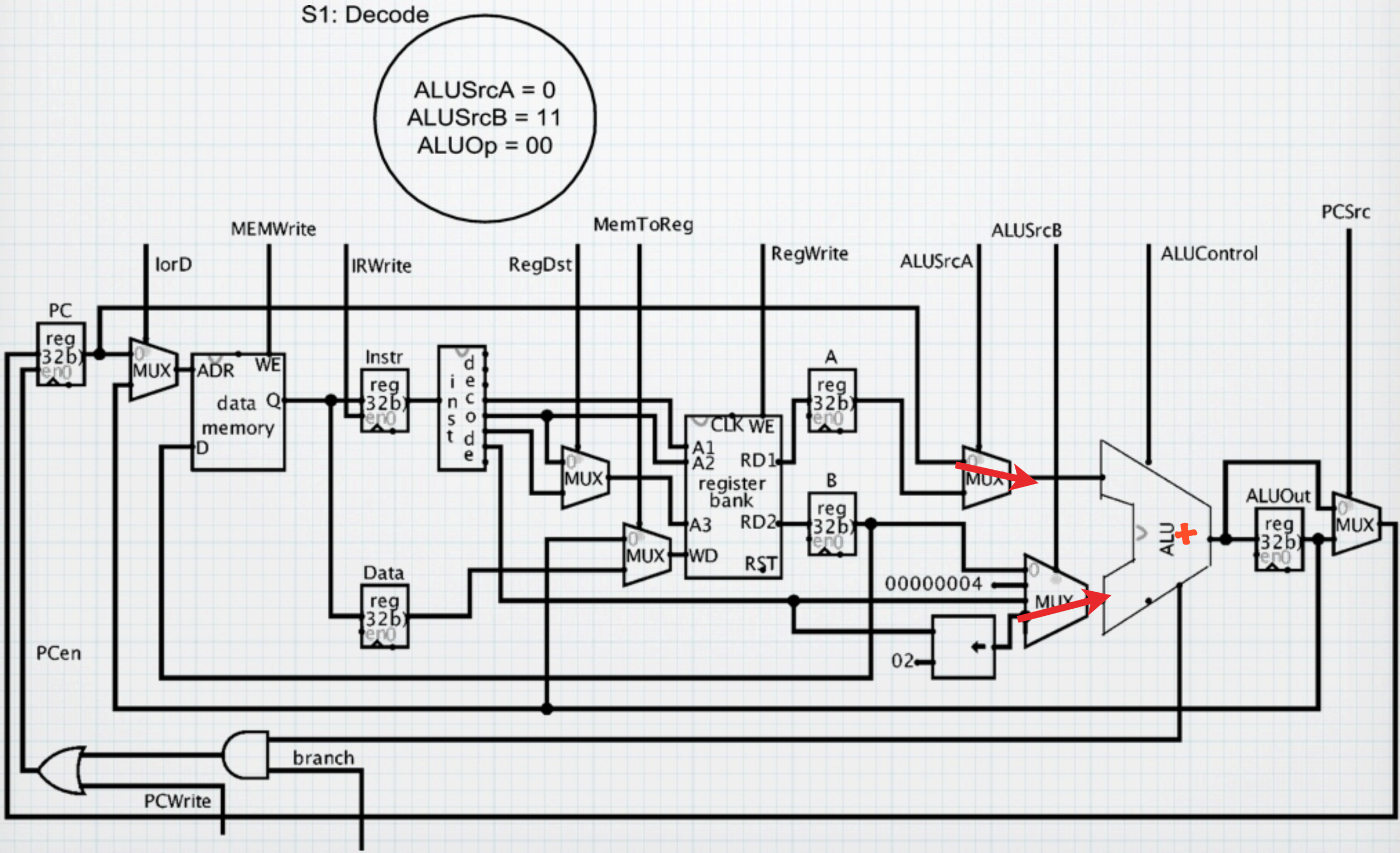
A:= Reg[IR[25:21]]
B:= Reg[IR[20:16]]
ALUout := PC + sign-extend(IR[15:0]]





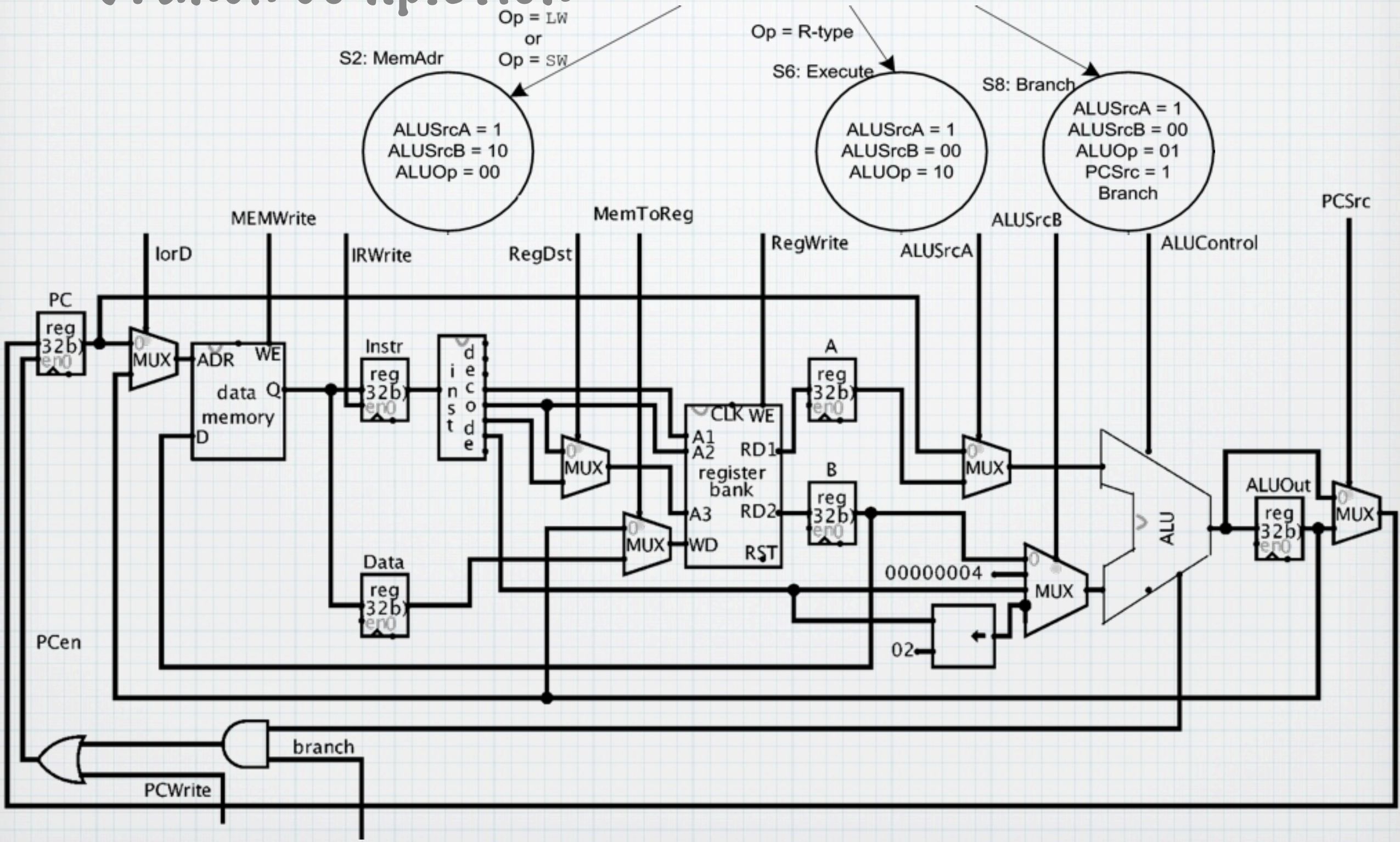
Instruction decode and Register fetch

A:= Reg[IR[25:21]]
B:= Reg[IR[20:16]]
ALUout := PC + sign-extend(IR[15:0]]



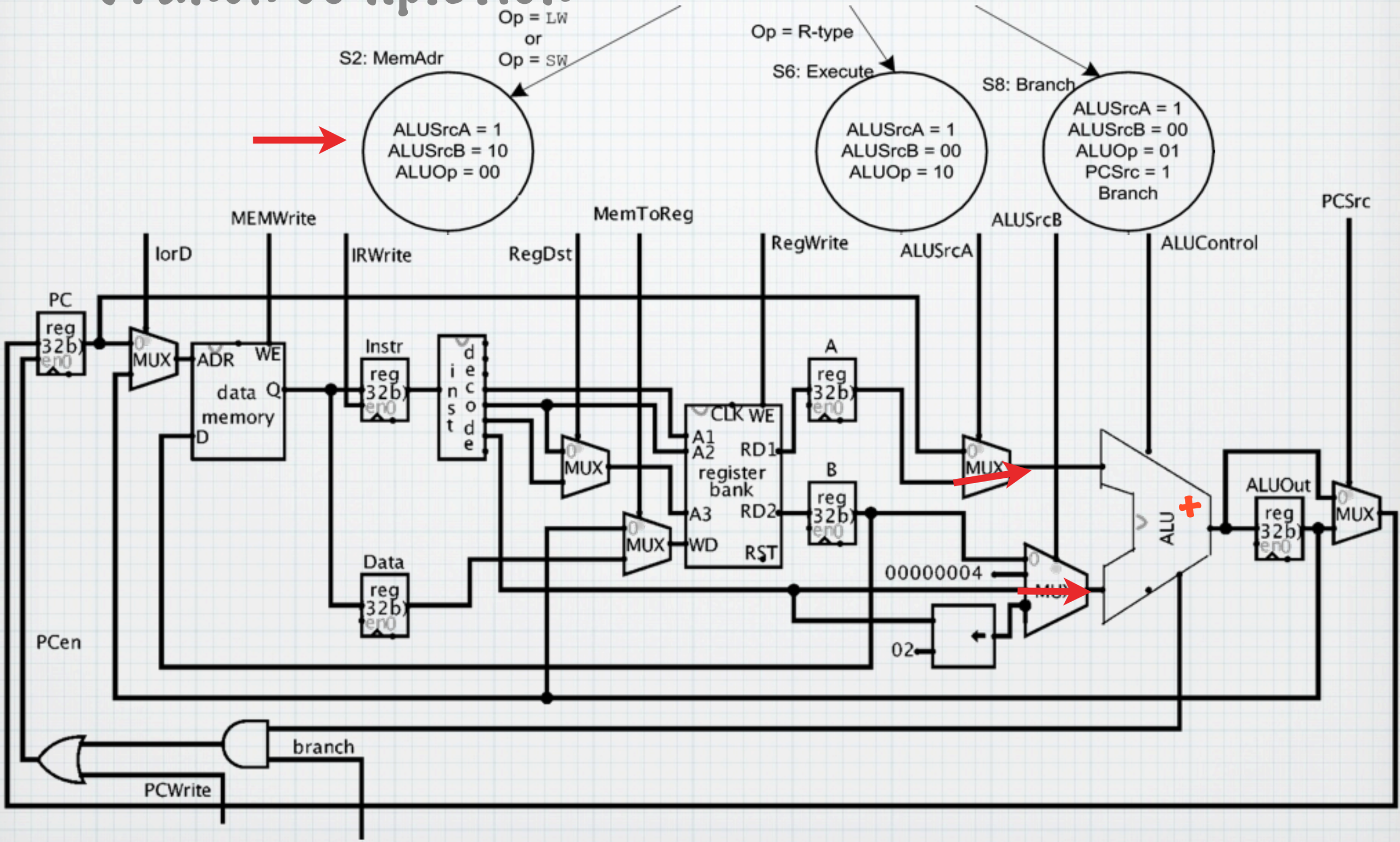
* Execute, Memory address, Branch completion

ALUout:= A op B
ALUout:= A+ IR[15:0]
if A=B then PC := ALUout



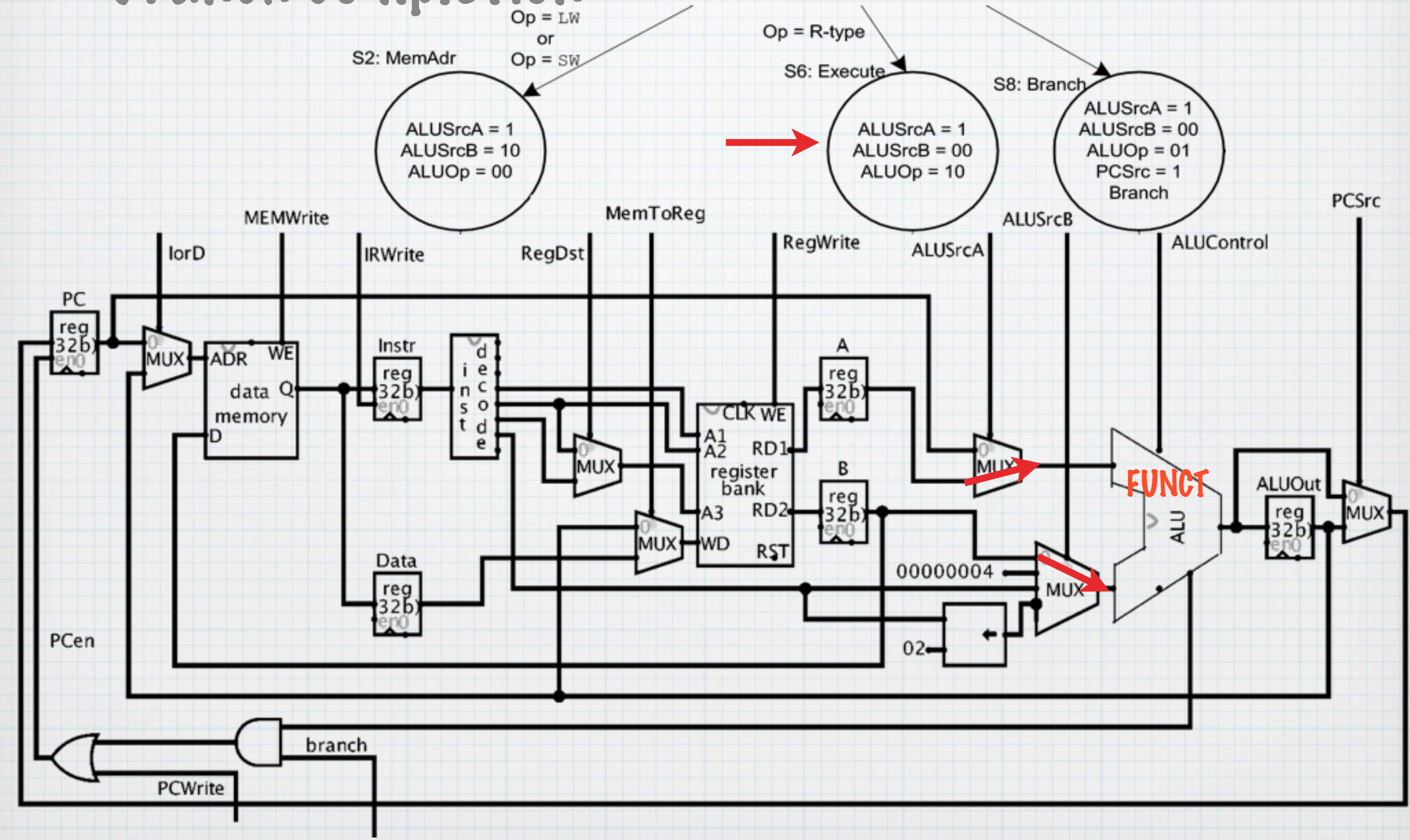
* Execute,
→ Memory address,
Branch completion

ALUout:= A op B
ALUout:= A+ IR[15:0]
if A=B then PC := ALUout



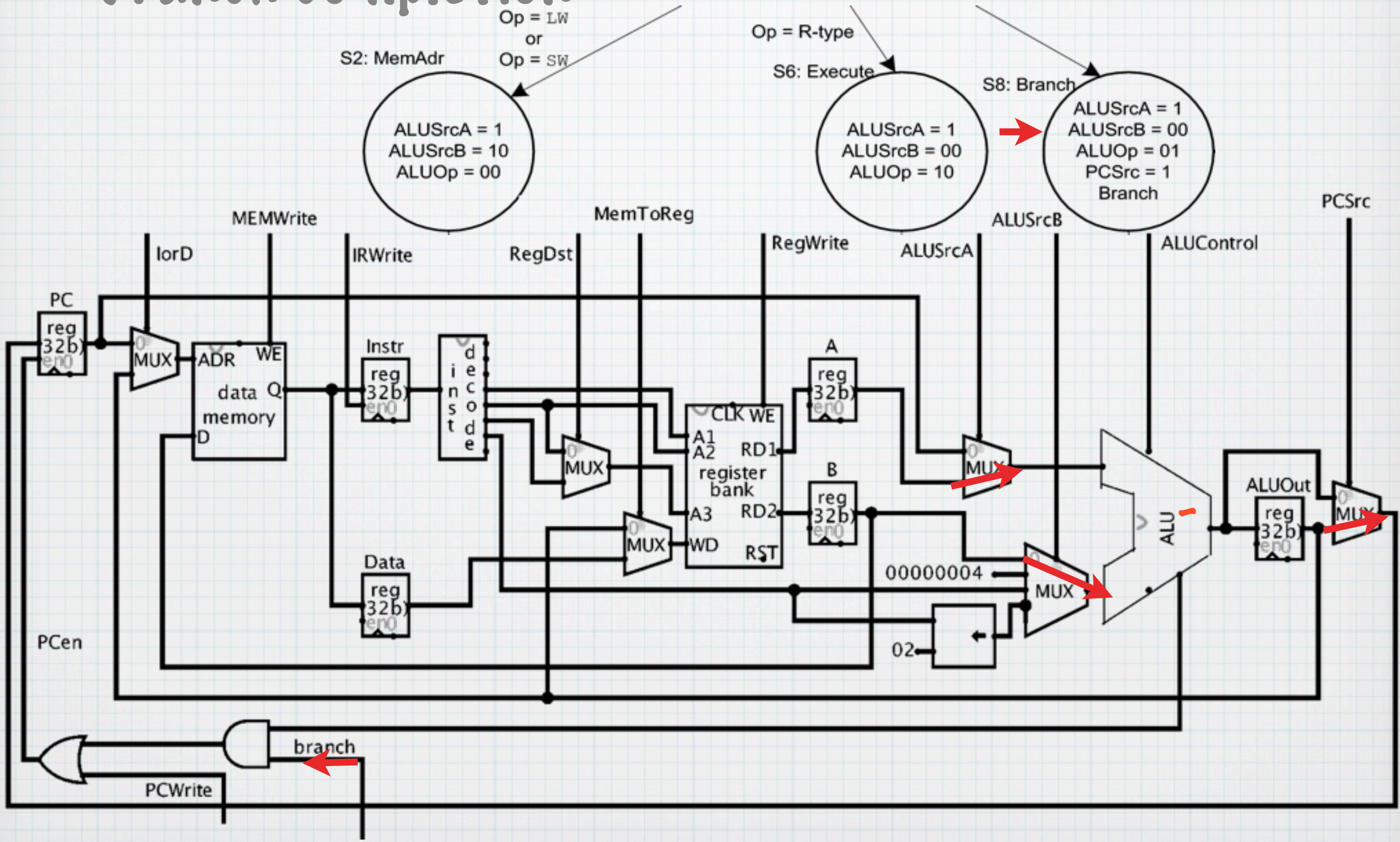
Execute, Memory address, Branch completion

ALUout:= A op B
ALUout:= A+ IR[15:0]
if A=B then PC := ALUout



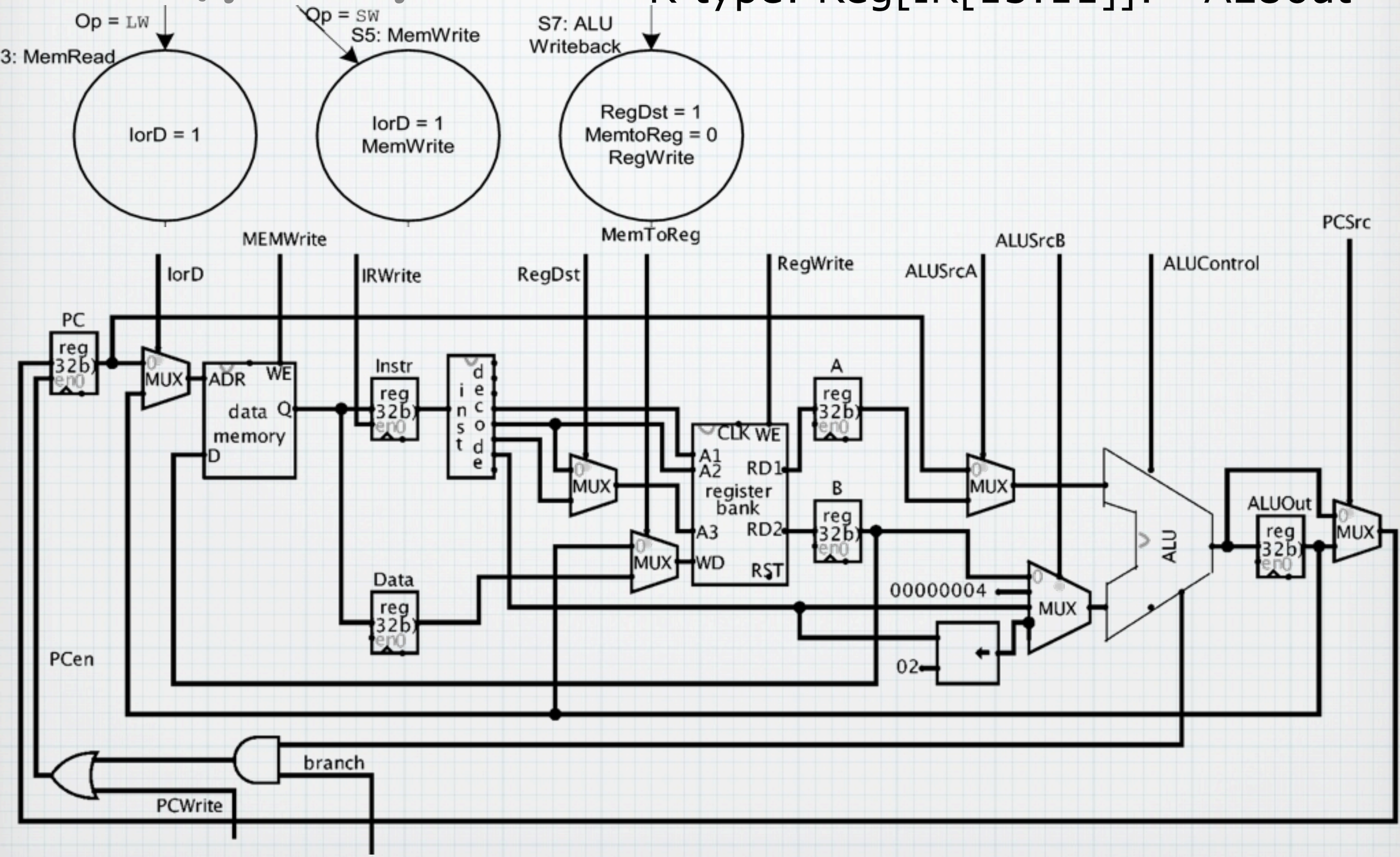
* Execute,
Memory address,
→ Branch completion

ALUout:= A op B
ALUout:= A+ IR[15:0]
if A=B then PC := ALUout



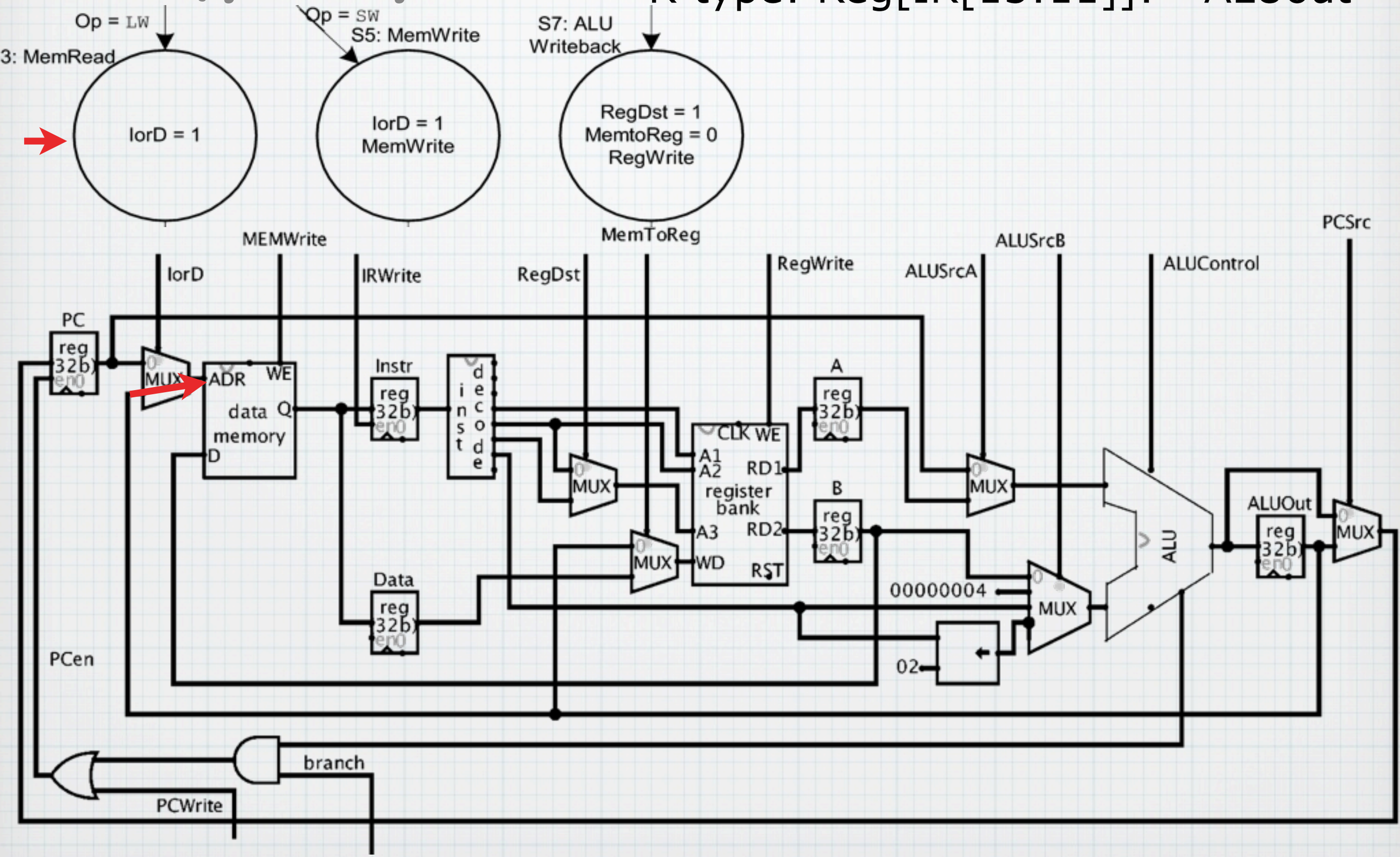
* Memory access
R-type completion

LW: MDR:= Memory[ALUout]
SW: Memory[ALUout]:= B
R-type: Reg[IR[15:11]]:= ALUout



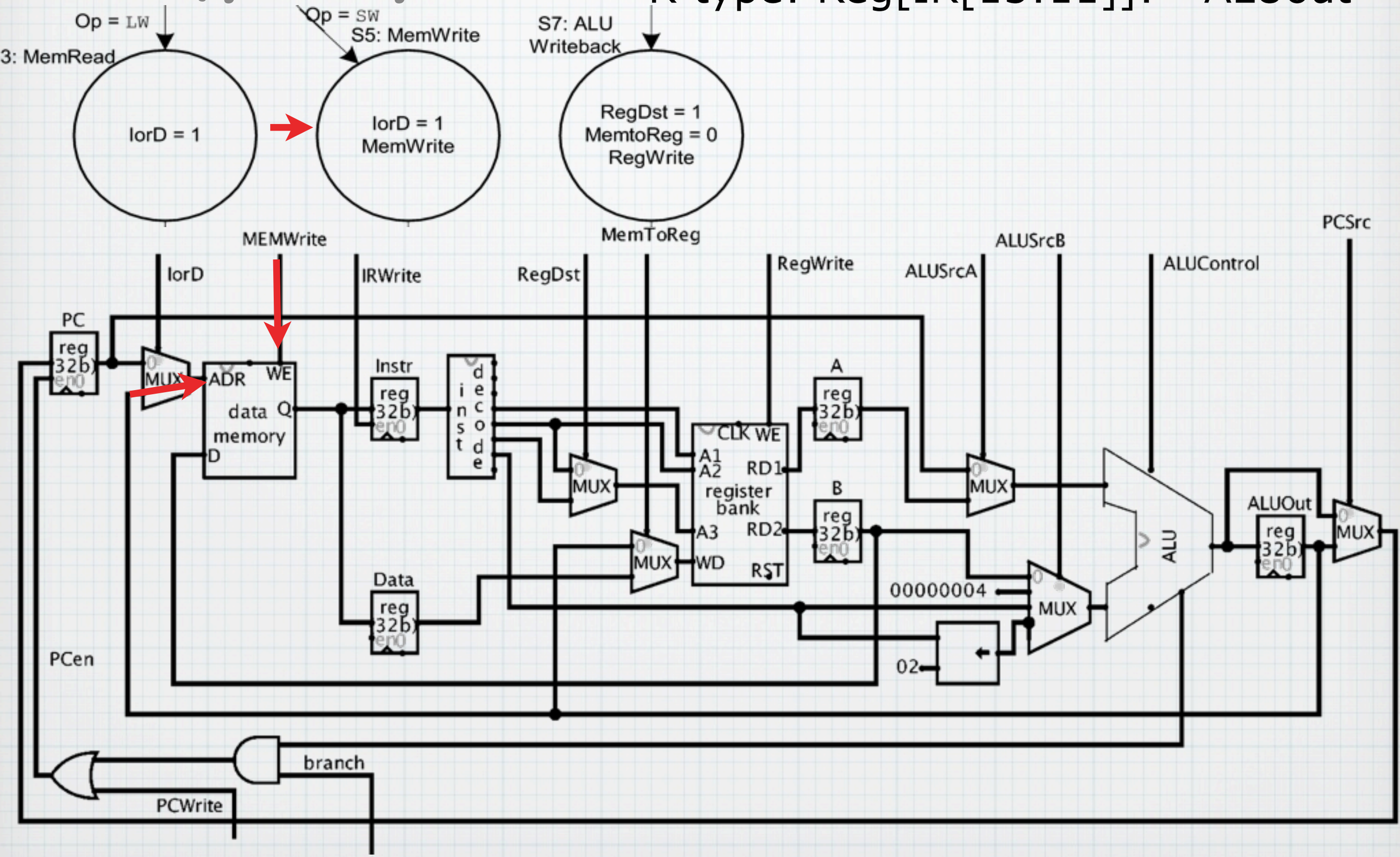
* Memory access
R-type completion

LW: MDR:= Memory[ALUout]
SW: Memory[ALUout]:= B
R-type: Reg[IR[15:11]]:= ALUout



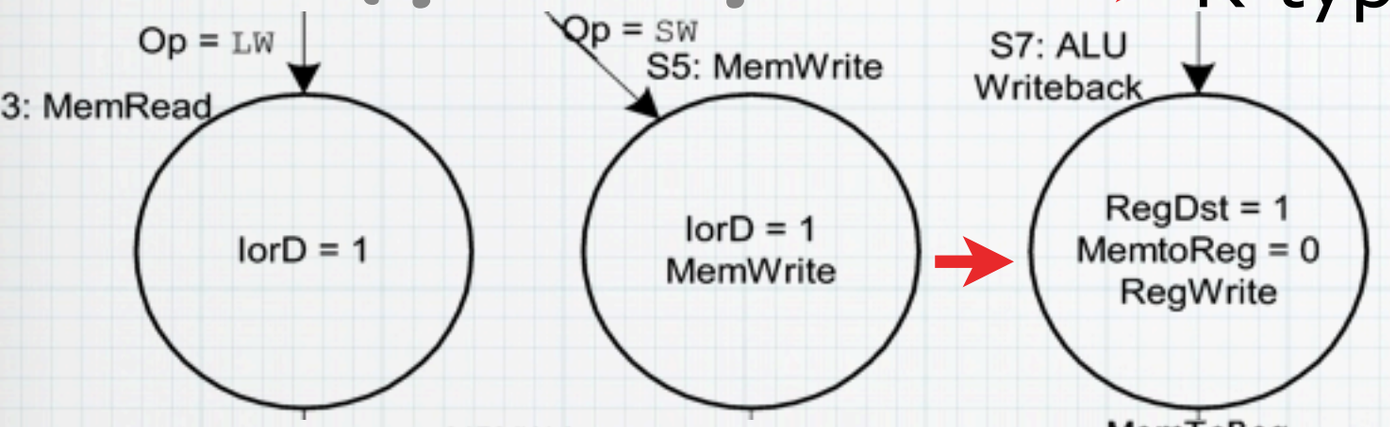
* Memory access
R-type completion

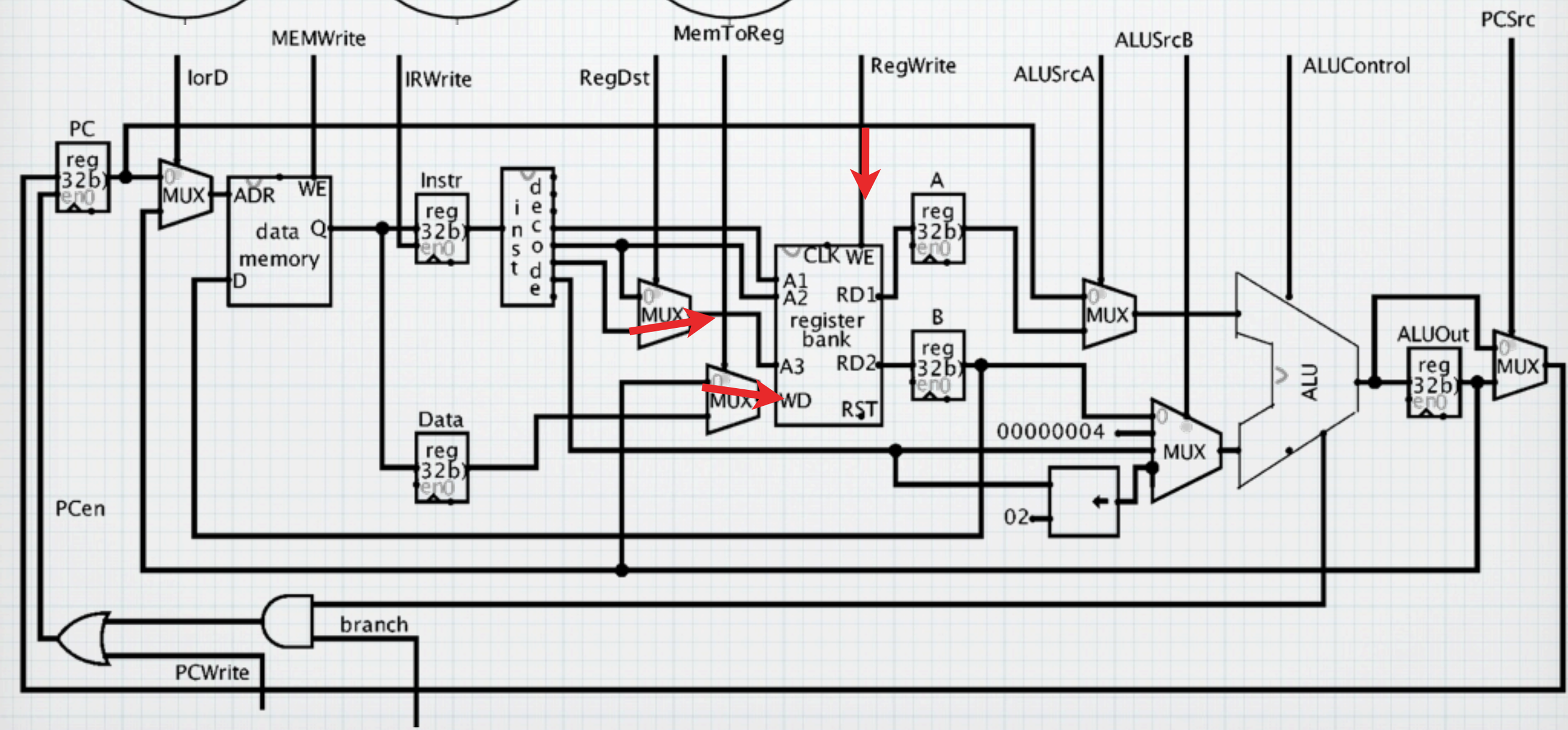
LW: MDR:= Memory[ALUout]
SW: Memory[ALUout]:= B
R-type: Reg[IR[15:11]]:= ALUout



* Memory access
R-type completion

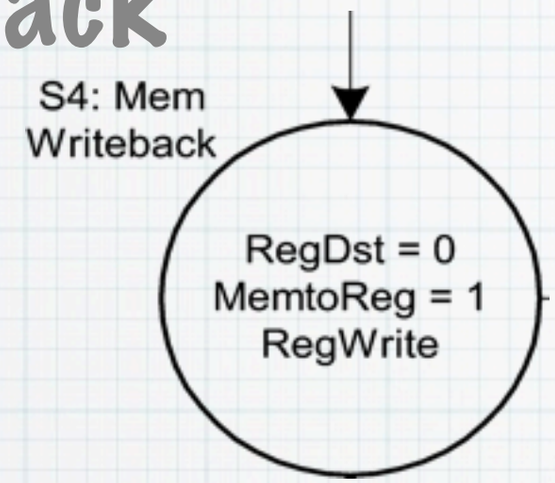
LW: MDR:= Memory[ALUout]
SW: Memory[ALUout]:= B
R-type: Reg[IR[15:11]]:= ALUout

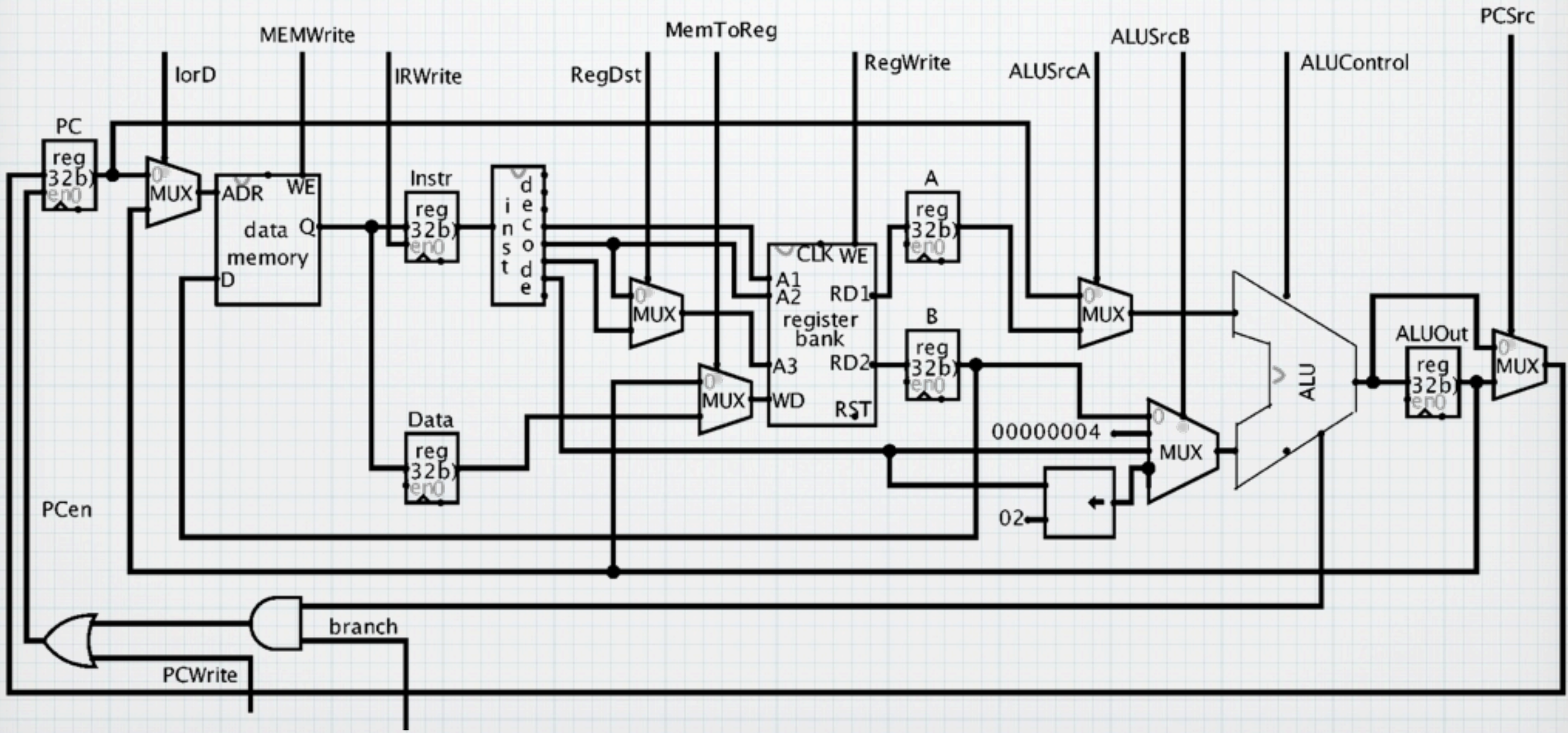




* Writeback

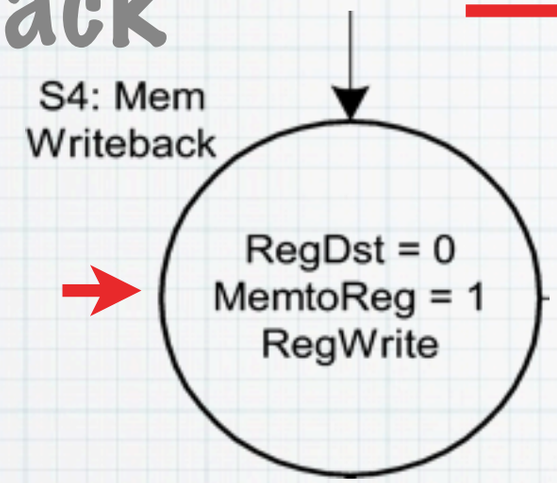
LW: Reg[[20:16]]:= MDR

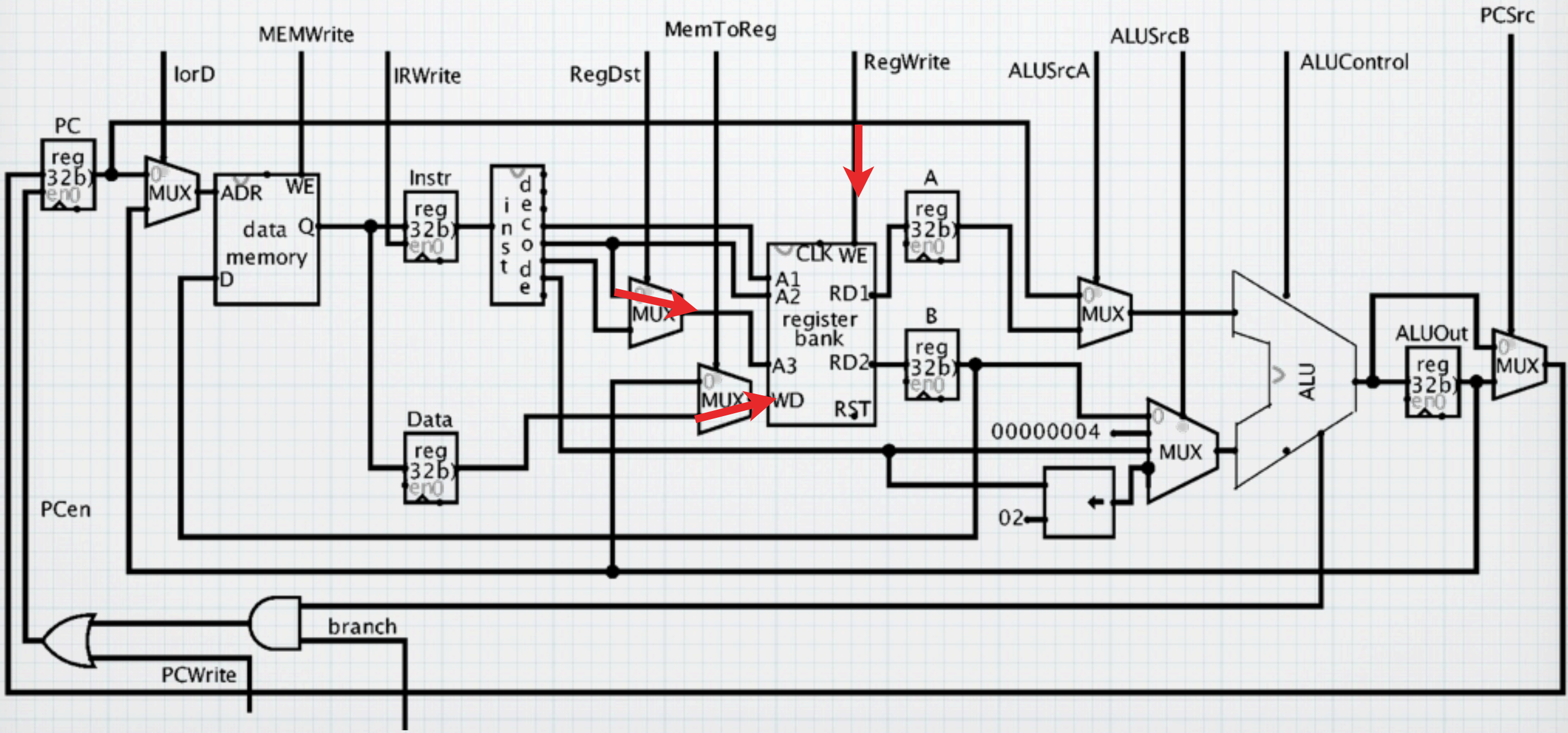




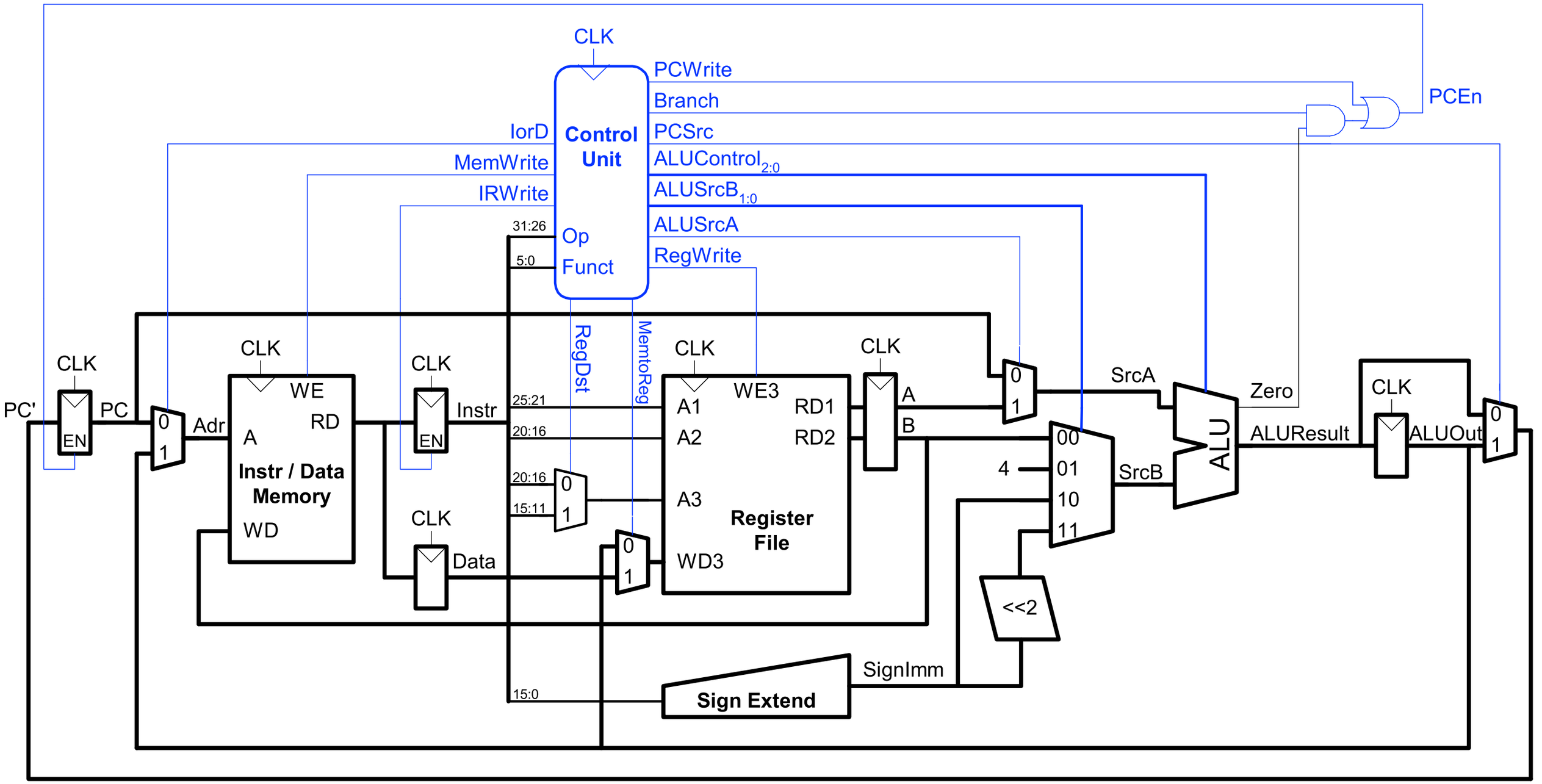
* Writeback

→ LW: Reg[[20:16]]:= MDR

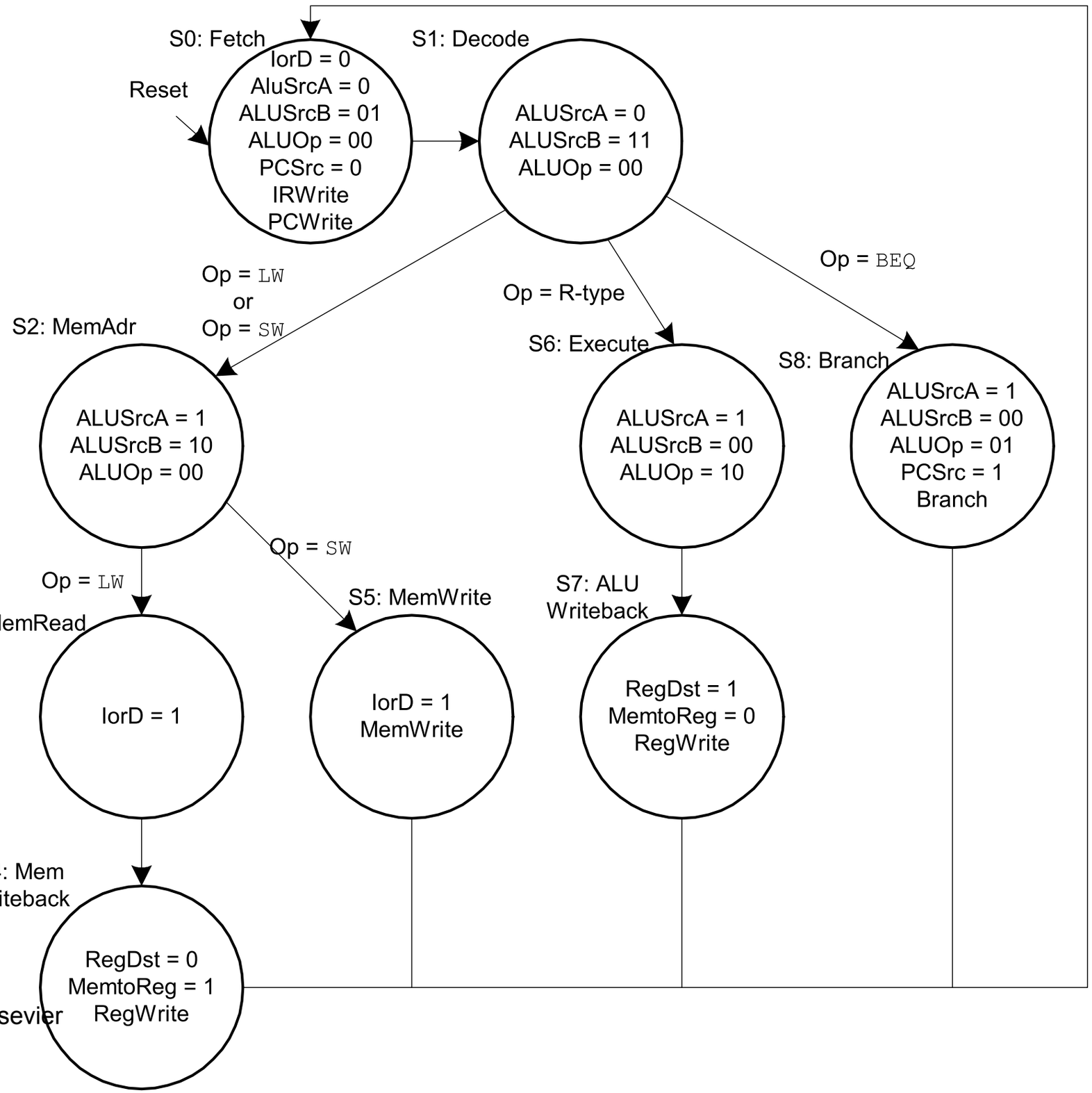


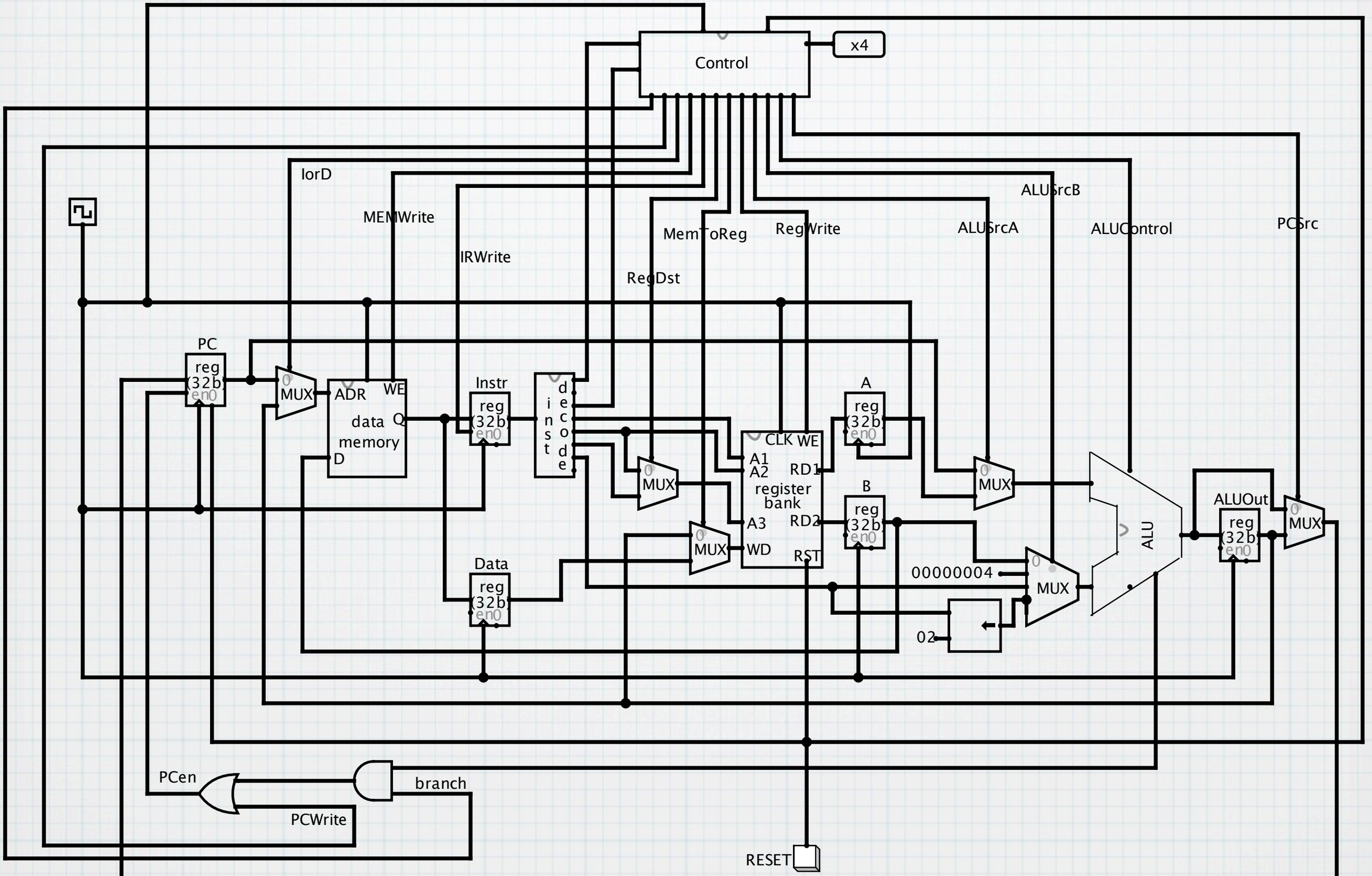


Complete Multicycle Processor

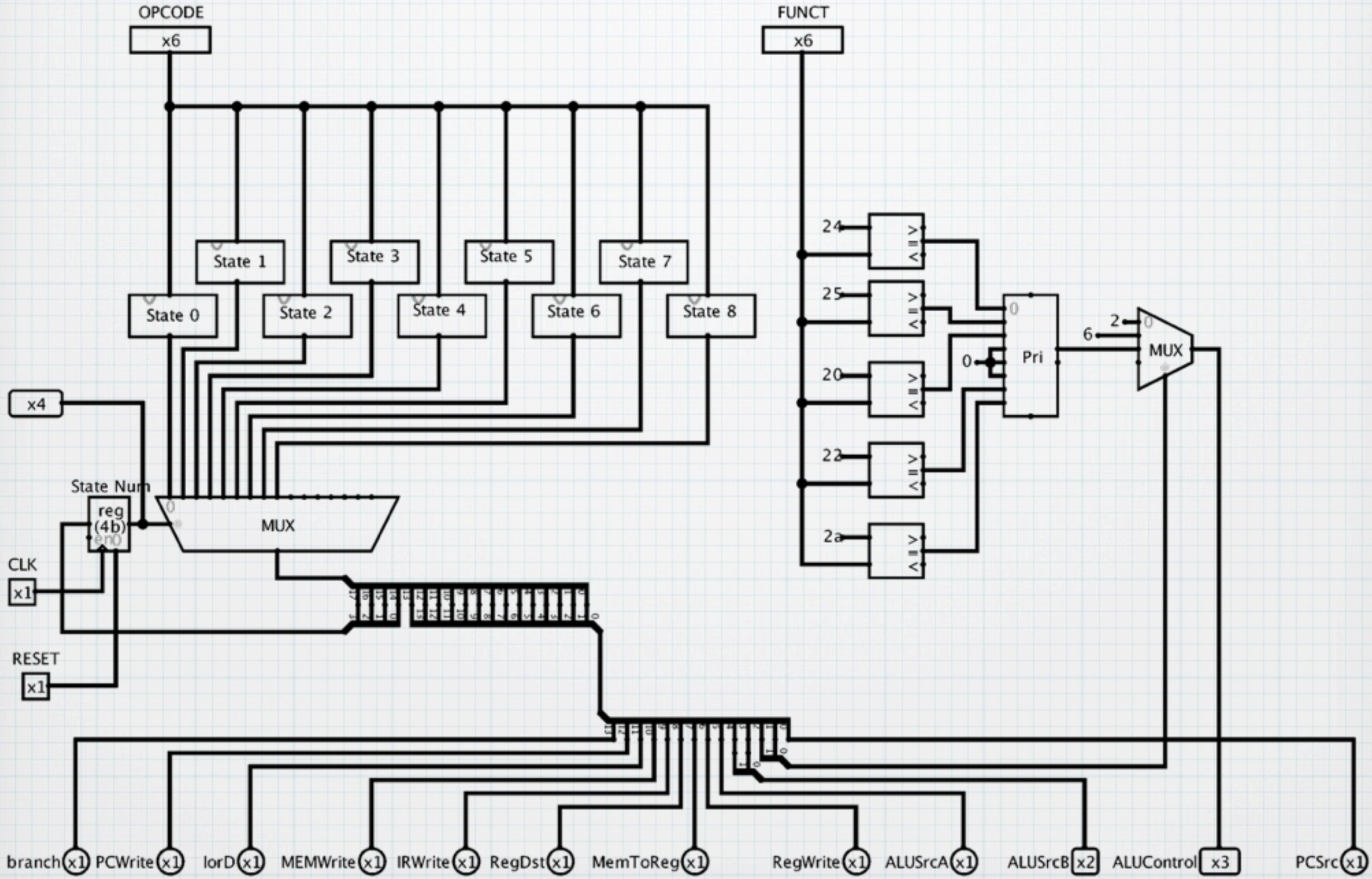


Complete Multicycle Controller FSM

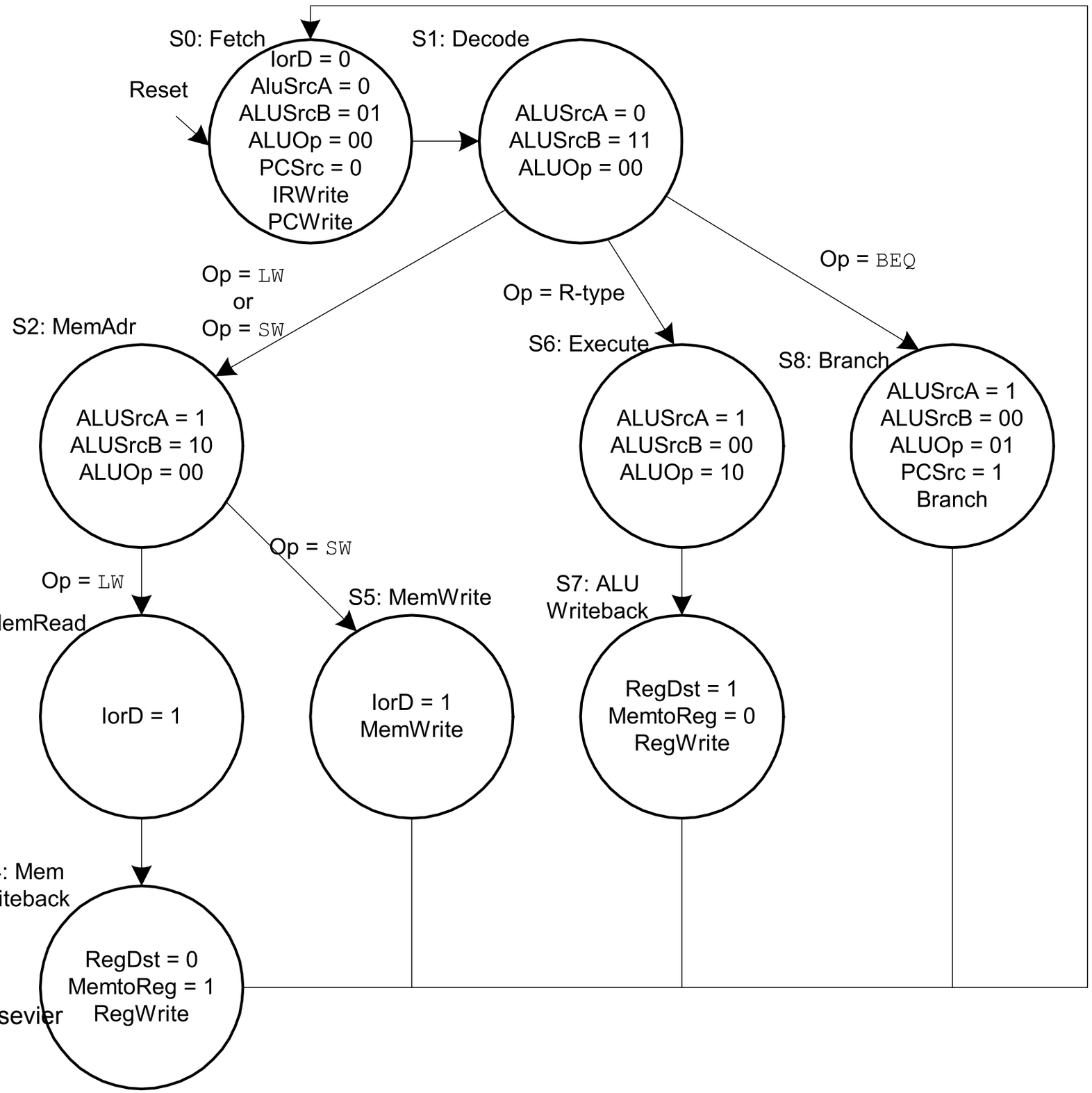



Control
x4
IorD
MEMWrite
IRWrite
MemToReg
RegDst
RegWrite
ALUSrcA
ALUSrcB
ALUControl
PCSrc
PC
reg
(32b)
en0
MUX
ADR
WE
data
memory
D
Q
Instr
reg
(32b)
en0
d
e
c
o
d
e
Data
reg
(32b)
en0
CLK
WE
A1
A2
RD1
register
bank
A3
RD2
WD
RST
A
reg
(32b)
en0
B
reg
(32b)
en0
00000004
02
ALU
ALUOut
reg
(32b)
en0
PCen
branch
PCWrite
RESET

Implementing Multi-state Contro



Complete Multicycle Controller FSM



Complete Multicycle Controller FSM

ALUSrcB = 01
ALUOp = 00
PCSrc = 0
IRWrite
PCWrite

ALUS
ALUSr
ALUO

Op = R

S6: M

Op = LW
or
Op = SW

S2: MemAdr

ALUSrcA = 1
ALUSrcB = 10
ALUOp = 00

Op = SW

Op = LW

S3: MemRead

S5: MemWrite

S
Wr

IorD = 1

Implementing Multi-state Contro