CSCI 210: Computer Architecture
Lecture 27: Control Path

Stephen Checkoway

Oberlin College

Apr. 29, 2022

Slides from Cynthia Taylor

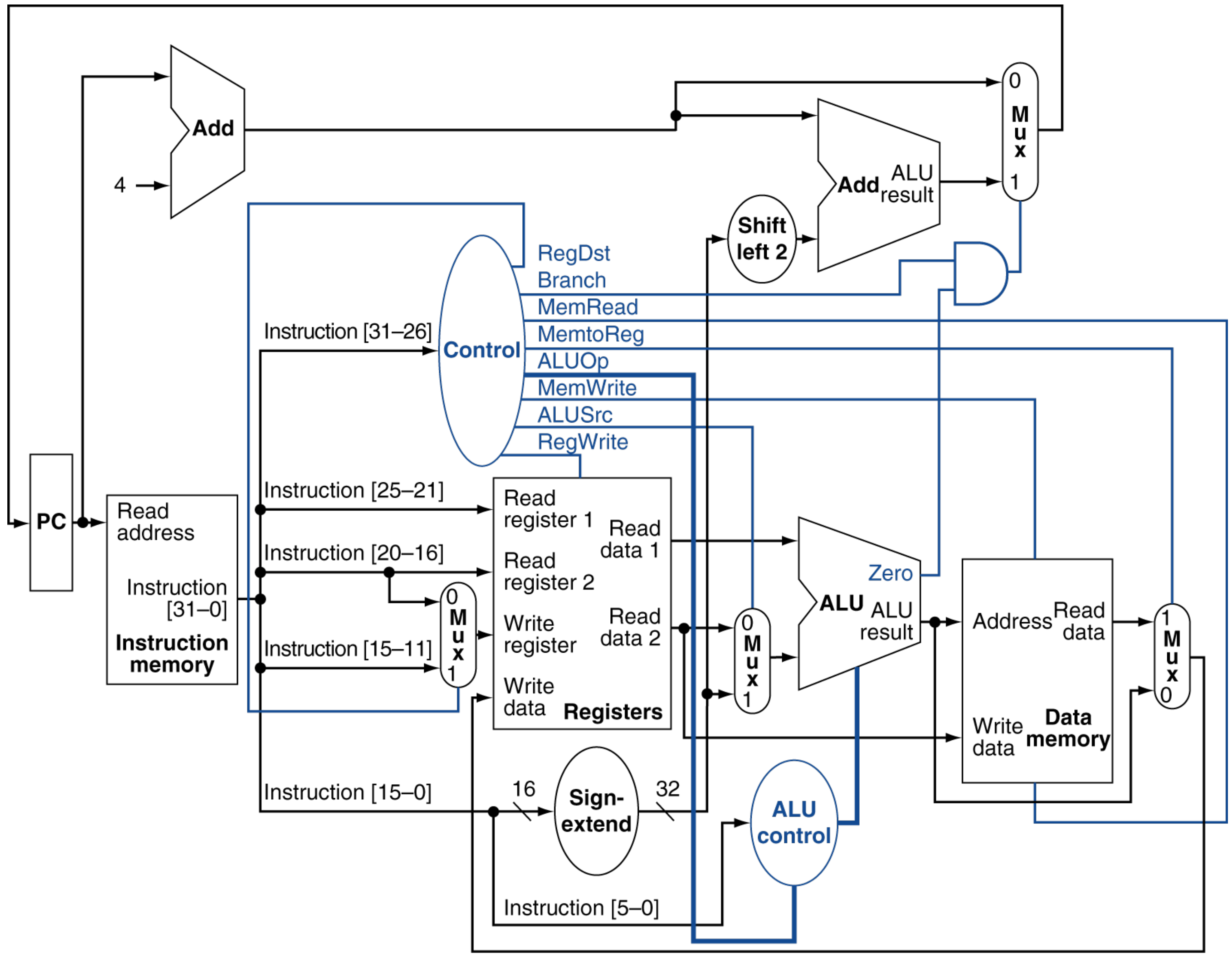
Announcements

- Problem Set 8 due today

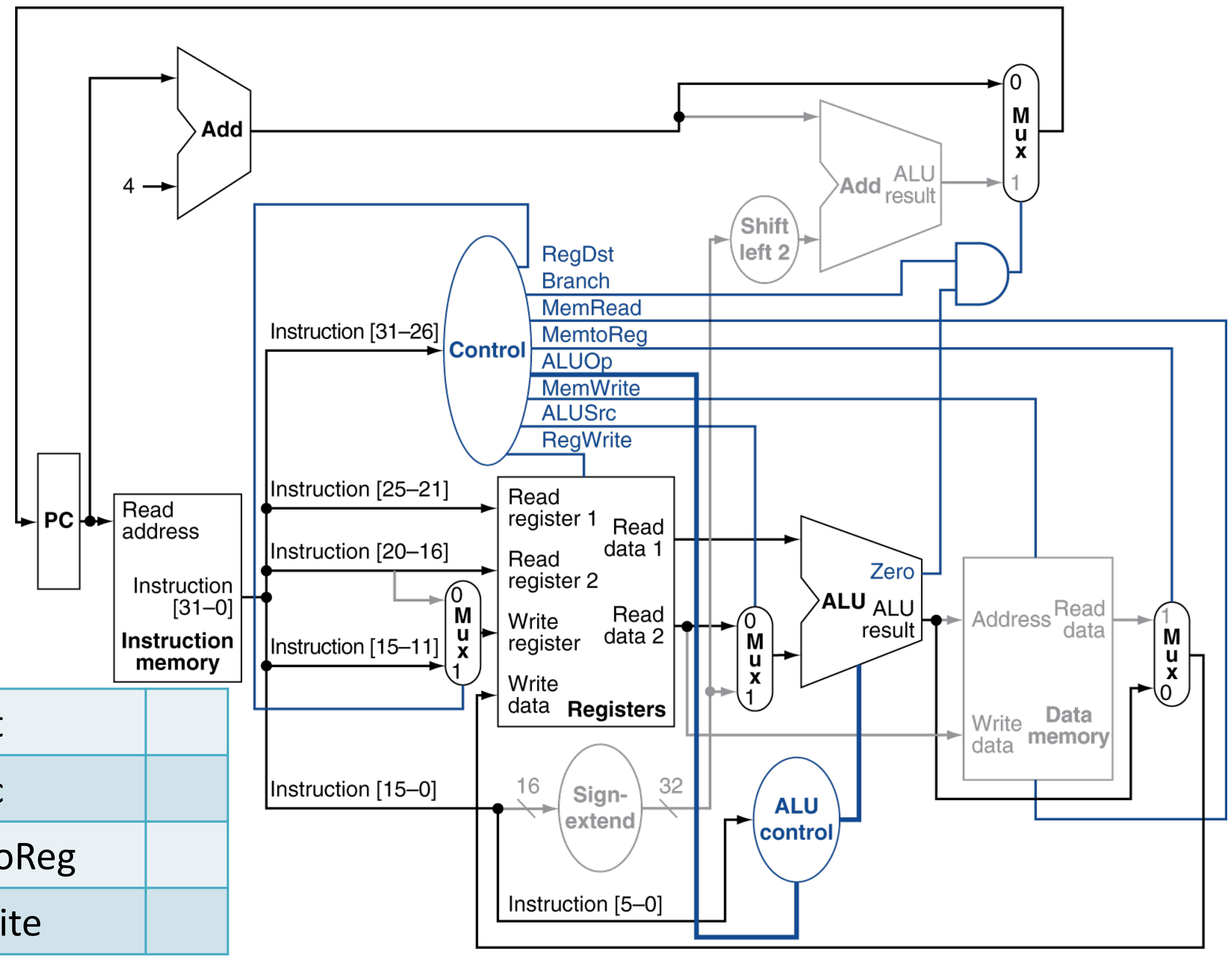
- Lab 7 due Monday

- Office Hours today 13:30 – 14:30

Data & Control Path



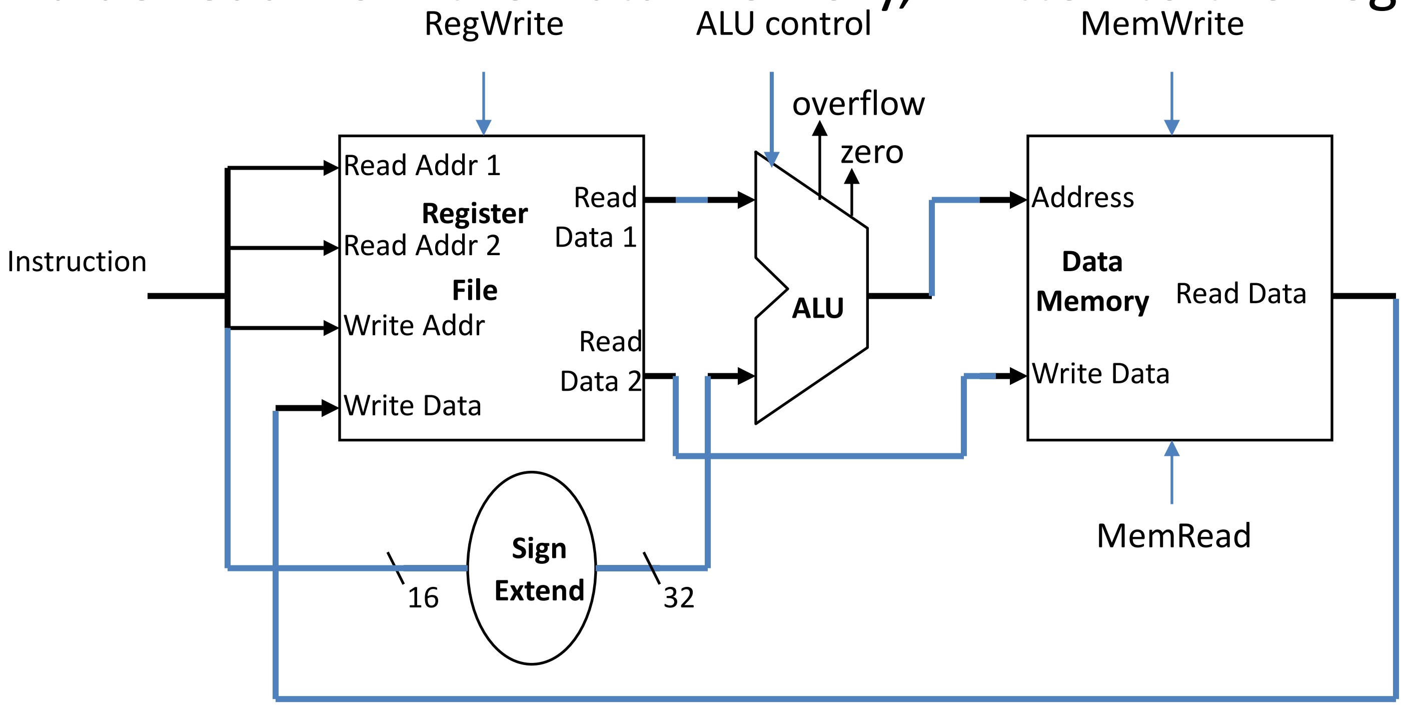
R-Type Instruction



RegDst	
ALUSrc	
MemToReg	
RegWrite	

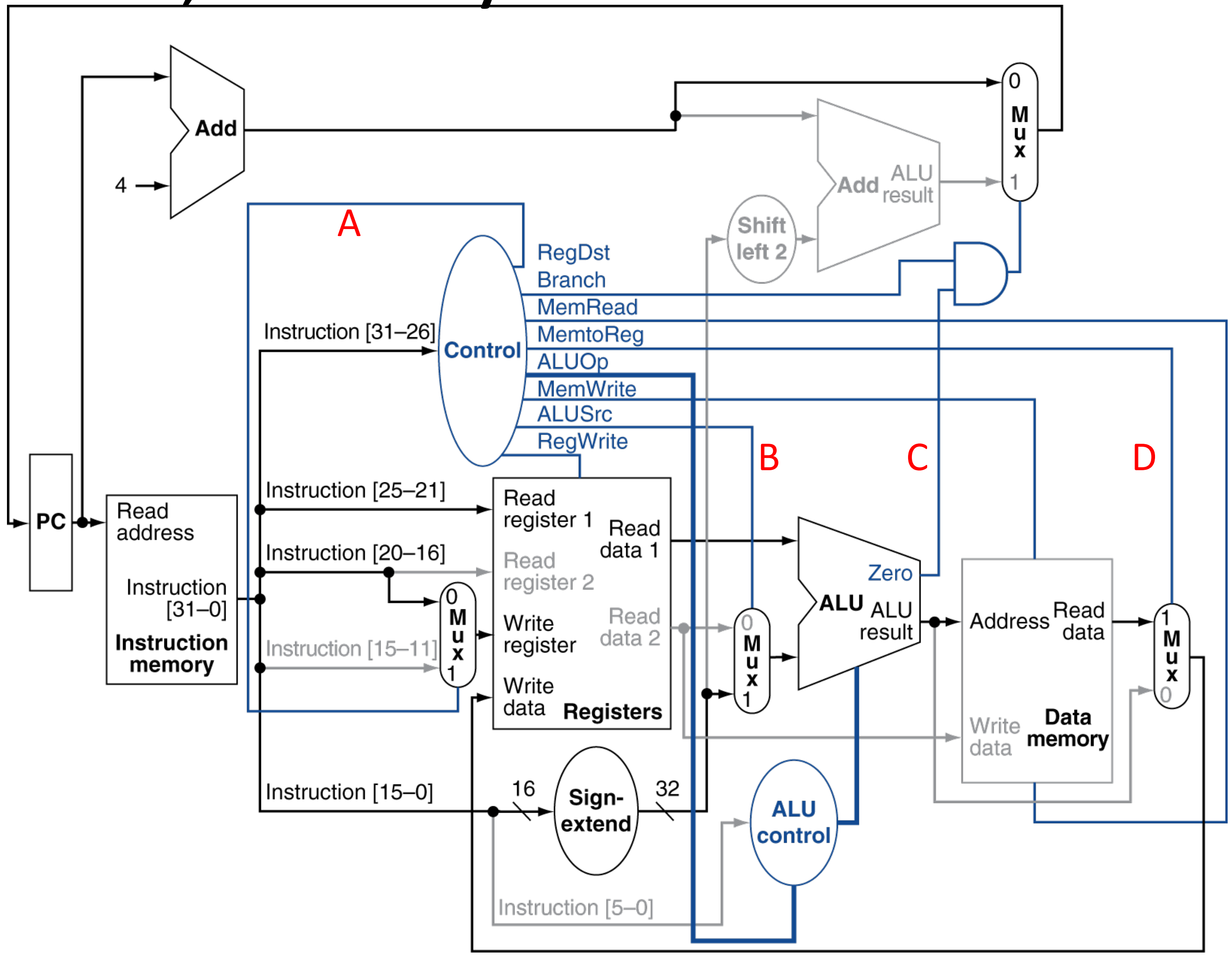
Executing Load and Store Operations

- compute memory address by adding base register to 16-bit sign-extended offset field
- store value written to the Data Memory
- load value read from the Data Memory, written to the Register File

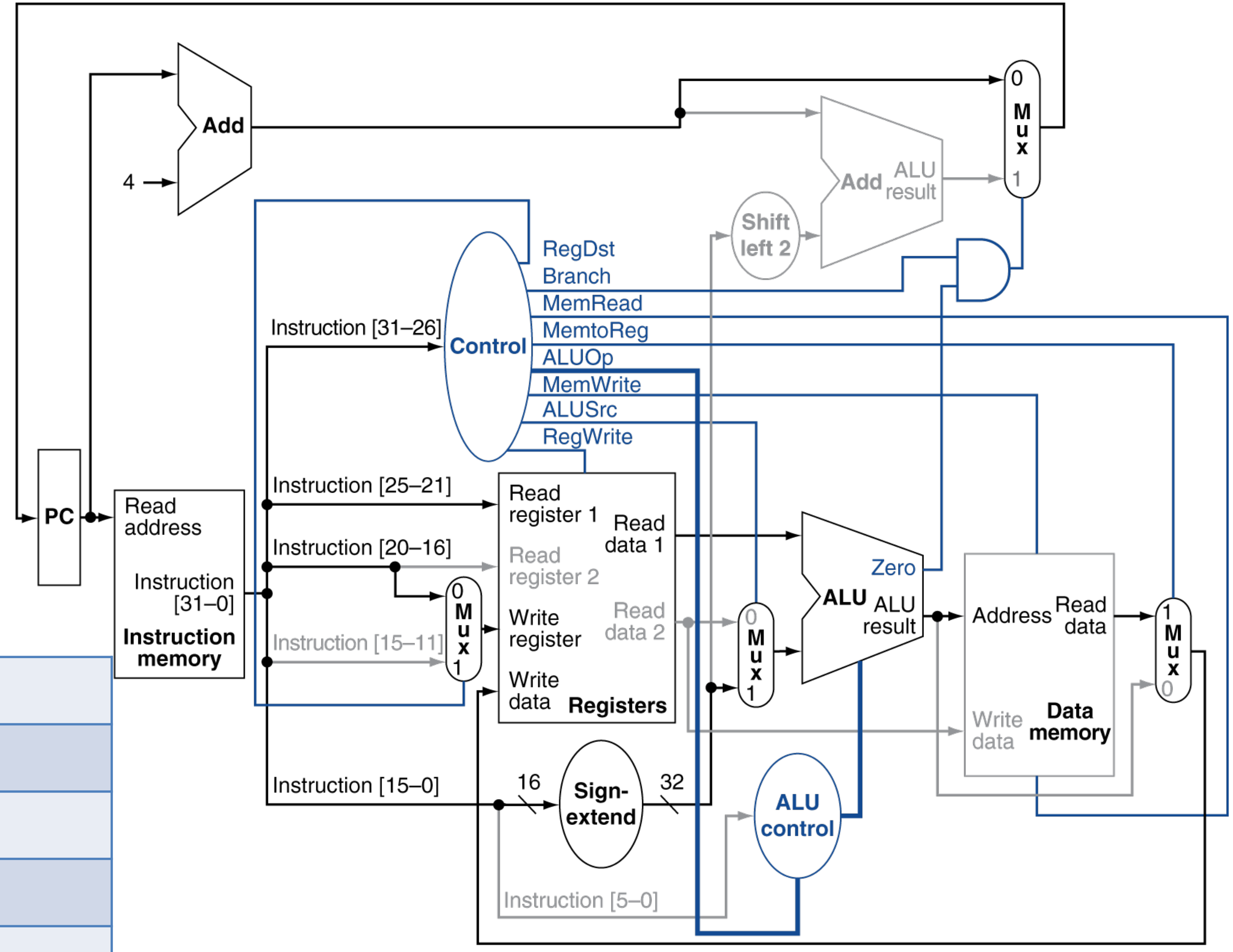


Load/ Store	opcode	rs	rt	offset
	31:26	25:21	20:16	15:0

Which wire, if always 1 would break lw?



Load Instruction



RegDest	
MemWrite	
MemRead	
MemtoReg	
RegWrite	

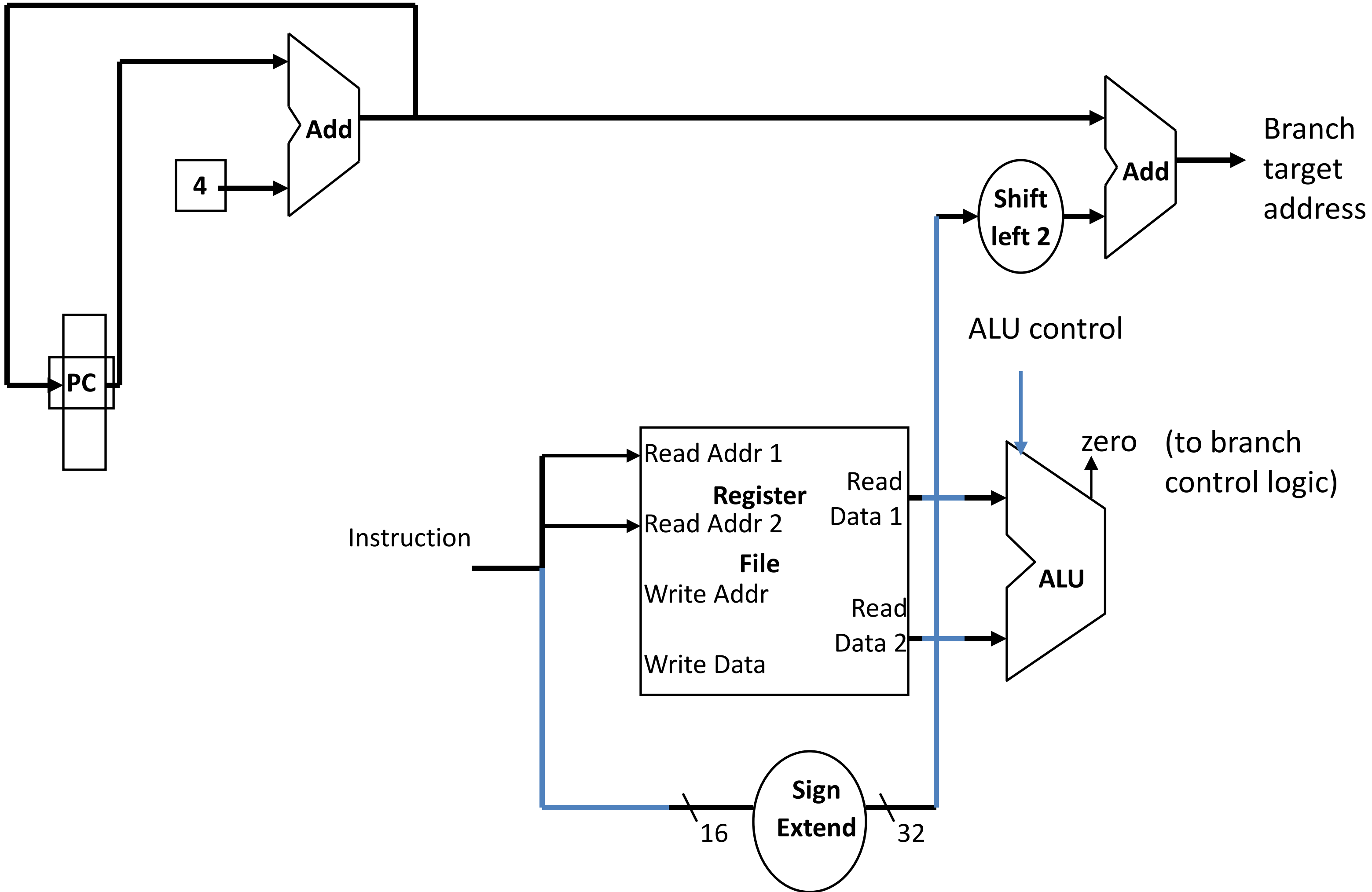
Executing Branch Operations

- compare the operands read from the Register File during decode for equality (zero ALU output)

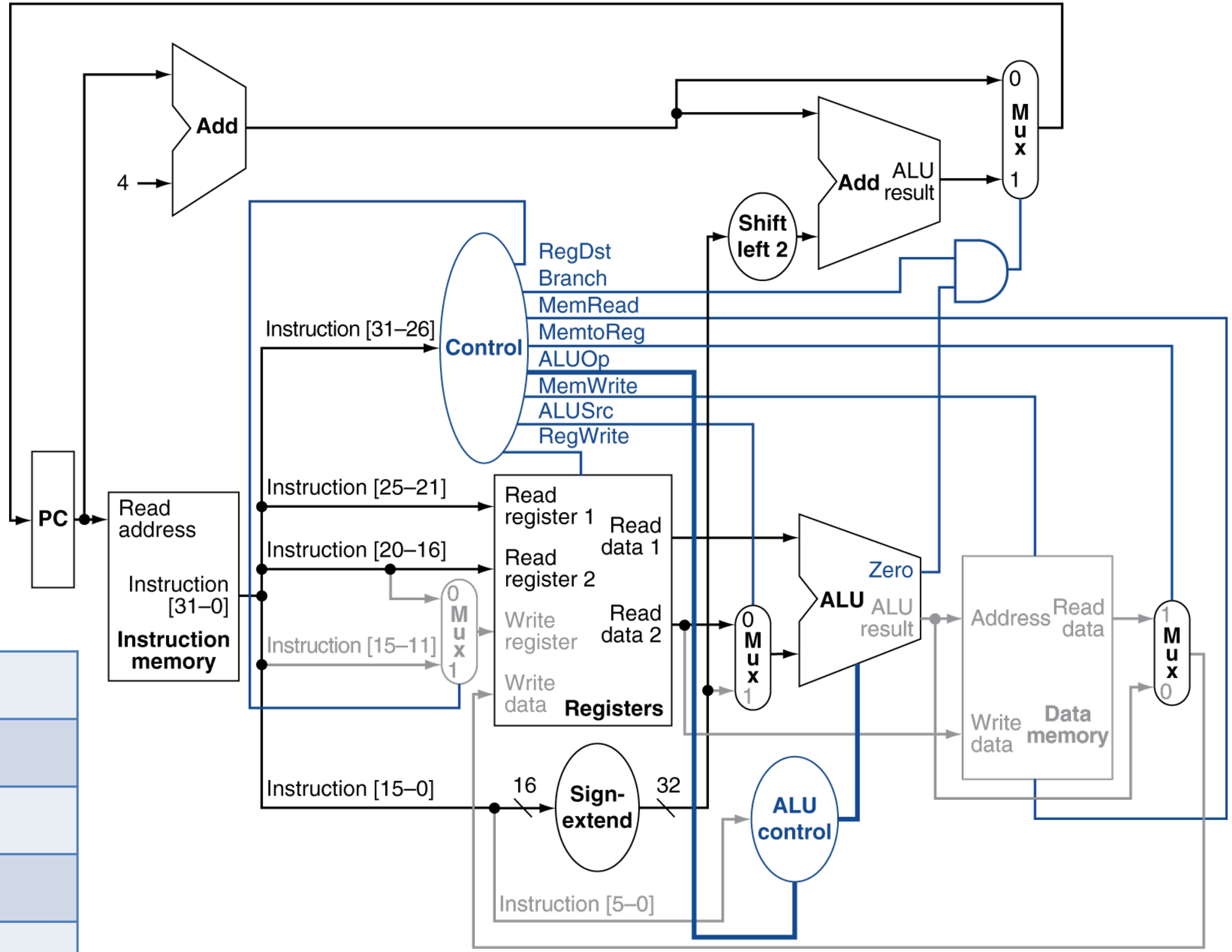
- compute the branch target address by adding the updated PC to the 16-bit sign-extended offset field in the instruction

Executing Branch Operations

opcode	rs	rt	address
31:26	25:21	20:16	15:0



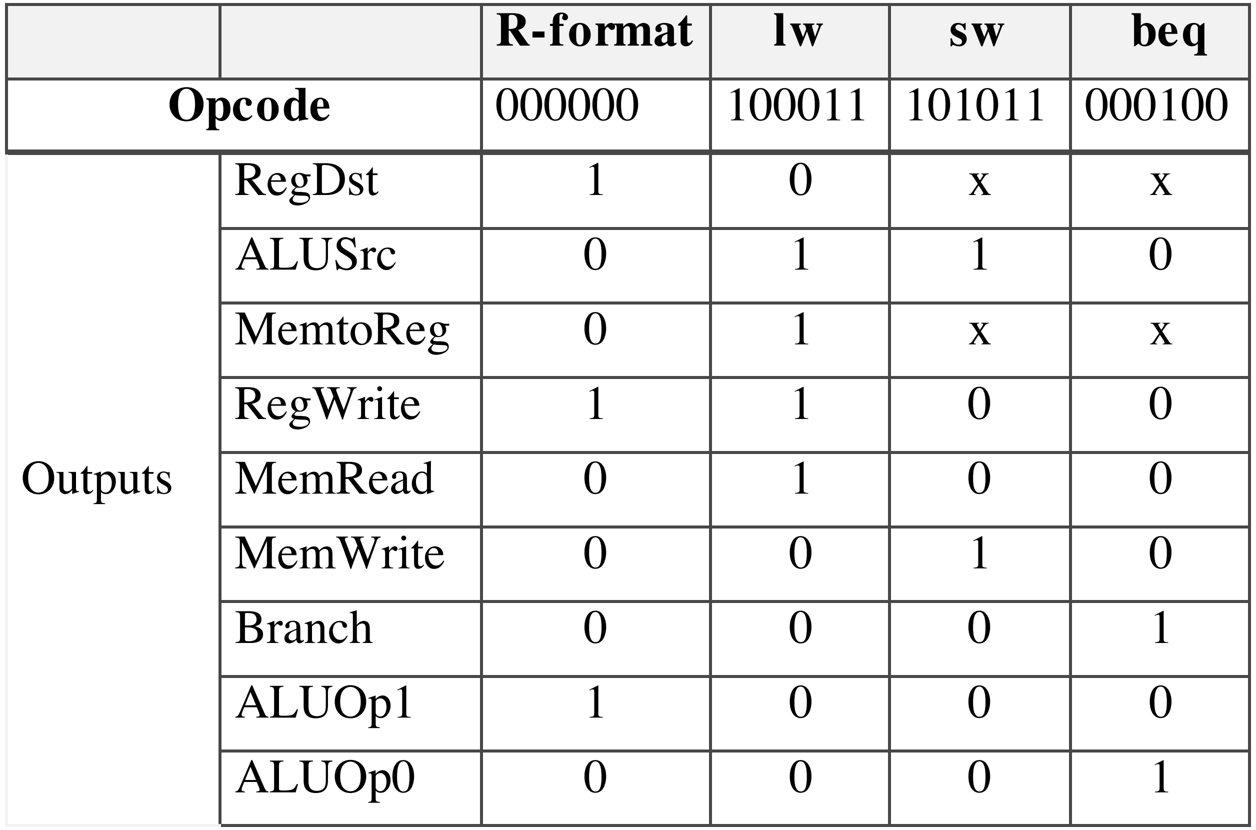
Branch-on-Equal Instruction



Branch	
MemWrite	
MemRead	
AluSrc	
RegWrite	

Control Truth Table

Main control takes the 6 opcode bits and produces the control signals using combinatorial logic

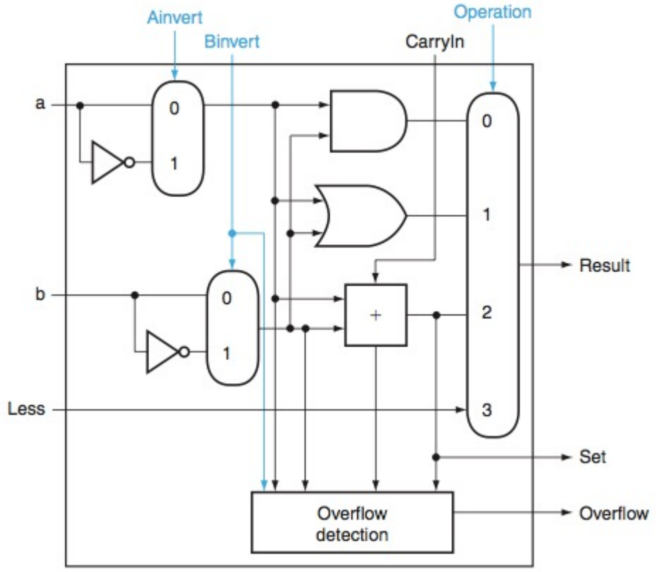


		R-format	**lw**	**sw**	**beq**
Opcode		000000	100011	101011	000100
Outputs	RegDst	1	0	x	x
	ALUSrc	0	1	1	0
	MemtoReg	0	1	x	x
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

Recall: PLAs

- Derived from truth table using sum of products
- Allow us to encode arbitrary functions
- Used to derive control signals in the datapath
  - Each control signal is a function of the 6 opcode bits

ALU Control



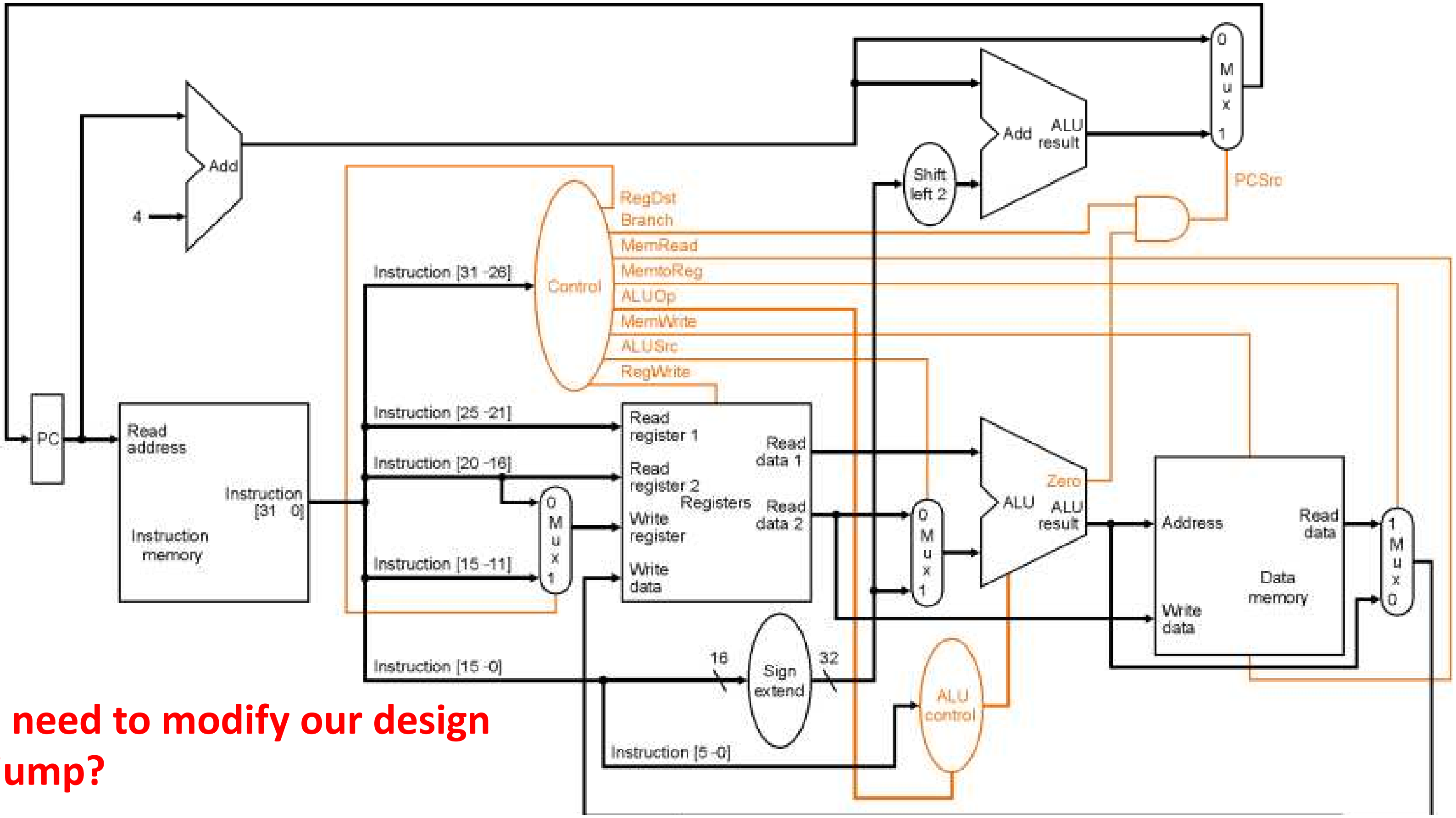
Takes as input 2-bit ALUop (derived from opcode) and 6-bit funct field; outputs 4 bits

Instruction	ALUOp	funct	ALU function	Ainvert	Binvert	ALU operation
load word	00 (add)	XXXXXX	add	0	0	10 (add)
store word	00 (add)	XXXXXX	add	0	0	10 (add)
branch equal	01 (subtract)	XXXXXX	subtract	0	1	10 (add)
add	10 (r-type)	100000	add	0	0	10 (add)
subtract		100010	subtract	0	1	10 (add)
AND		100100	AND	0	0	00 (and)
OR		100101	OR	0	0	01 (or)
NOR		100111	NOR	1	1	00 (and)
set-on-less-than		101010	set-on-less-than	0	1	11 (less)

Implementing Jumps

Jump	2	address
	31:26	25:0

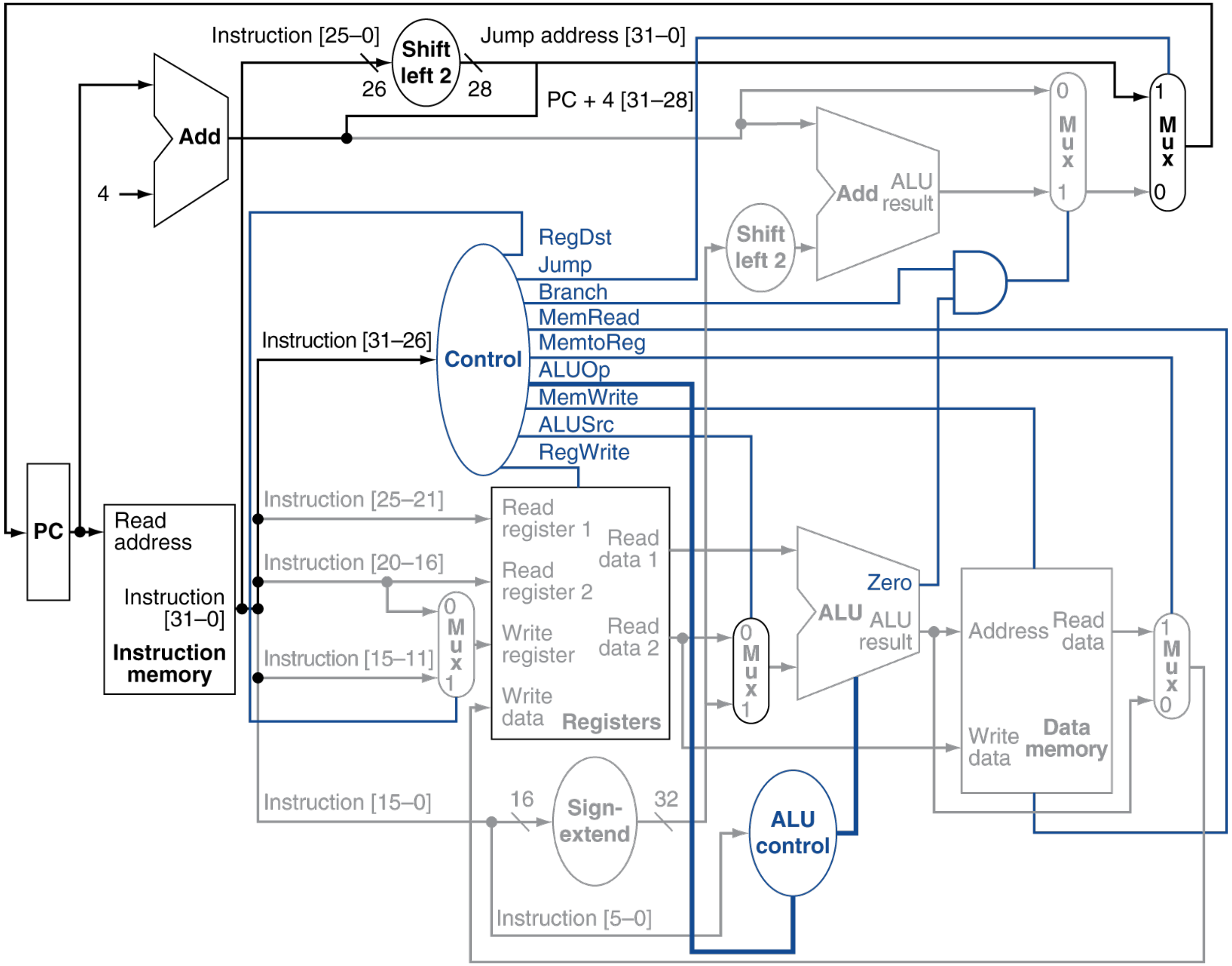
- Jump uses word address
- Update PC with concatenation of
  - Top 4 bits of PC + 4
  - 26-bit jump address
  - 00



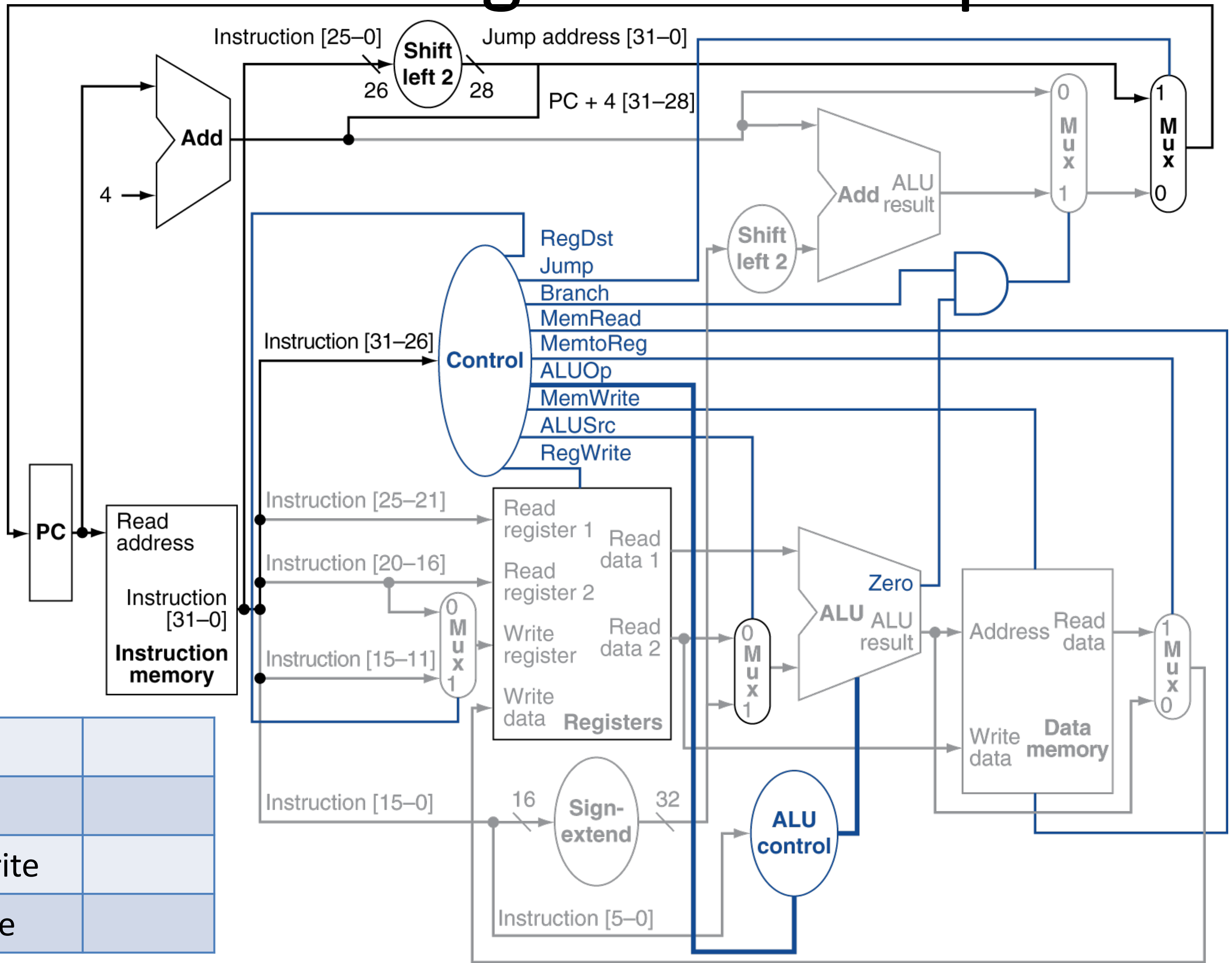
Do we need to modify our design to do jump?

Select	Best Answer
A	Yes – we need both new control and datapath.
B	Yes – we need just datapath.
C	No – but we should for better performance.
D	No – just changing control signals is fine.
E	Single cycle can't do jump register.

Datapath With Jumps Added



What will the Signals for Jump be?



Jump	
Branch	
MemWrite	
RegWrite	

Questions on the Data & Control Path?

Reading

- Next lecture: Pipeline
  - – Section 5.7