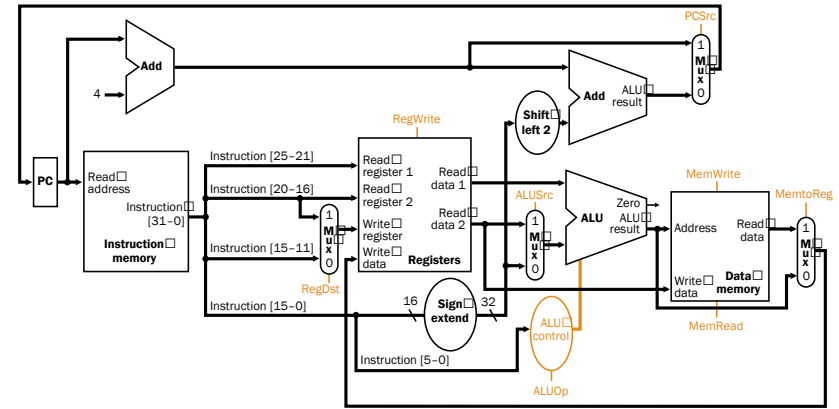


Control Logic for the Single-Cycle CPU

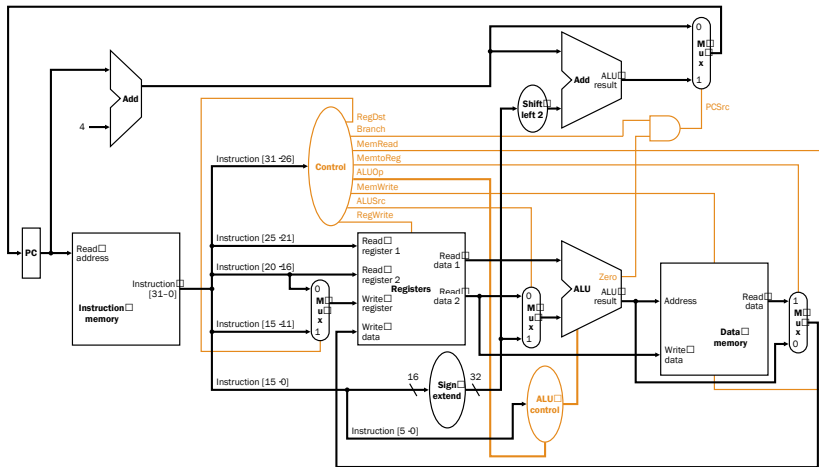
OR
Who's in charge here?

Putting it All Together: A Single Cycle Datapath

- We have everything except



Okay, then, what about those Control Signals?



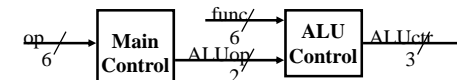
ALU control bits

- Recall: 5-function ALU

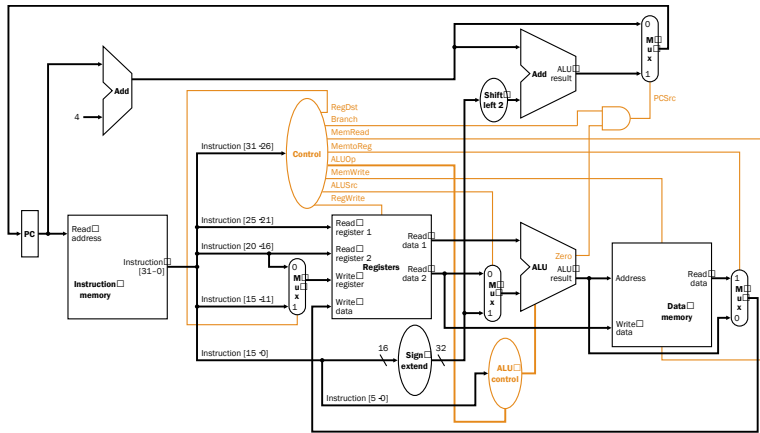
ALU control input	Function	Operations
000	And	and
001	Or	or
010	Add	add, lw, sw
110	Subtract	sub, beq
111	Slt	slt

- based on (bits 31-26) and code (bits 5-0) from instruction
- ALU doesn't need to know all opcodes--we will summarize opcode with ALUOp (2 bits):

00 - lw,sw 01 - beq 10 - R-format

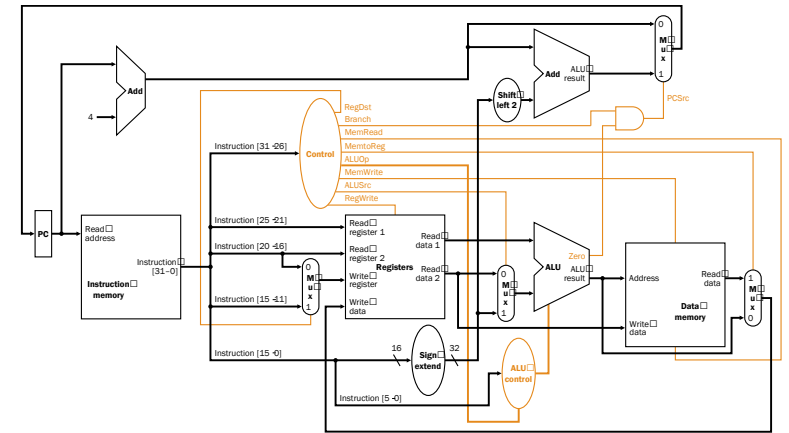


sw Control



Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw								0	0
beq								0	1

beq Control



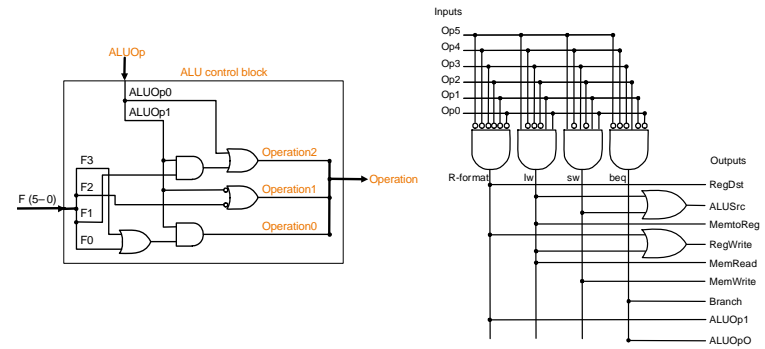
Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq								0	1

Control Truth Table

		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	

Control

- Simple combinational logic (truth tables)



Single-Cycle CPU Summary

- Easy, particularly the control
- Which instruction takes the longest? By how much? Why is that a problem?
- $ET = IC * CPI * CT$
- What else can we do?
- When does a CT make sense?
 - e.g., 70% of instructions take 75 ns, 30% take 200 ns?
 - suppose 20% overhead for extra latches
- Real machines have much *more* variable instruction latencies than this.