

Control Logic for the Single-Cycle CPU

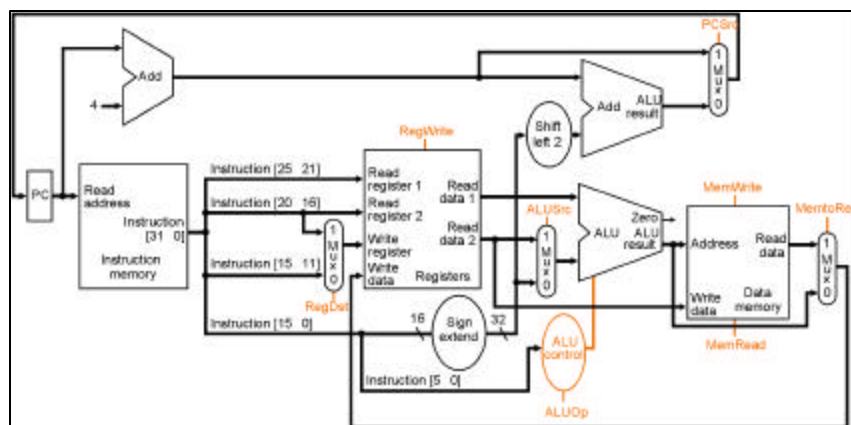
or
Who's in charge here?

CSE 141

Allan Snively

Putting it All Together: A Single Cycle Datapath

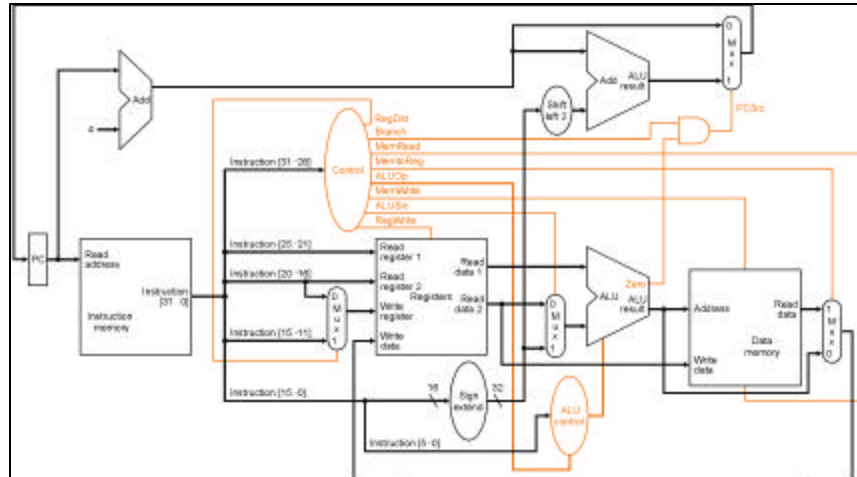
- We have everything except control signals



CSE 141

Allan Snively

Okay, then, what about those Control Signals?



CSE 141

Allan Snively

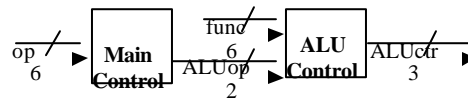
ALU control bits

- 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

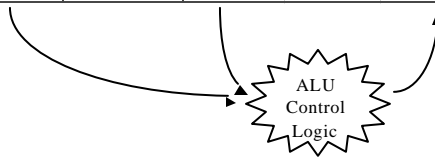
- based on opcode (bits 31-26) and function code (bits 5-0) from instruction
- ALU doesn't need to know all opcodes--we will summarize opcode with ALUOp (2 bits):

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



Generating ALU control

Instruction opcode	ALUOp	Instruction operation	Function code	Desired ALU action	ALU control input
lw	00	load word	xxxxxx	add	010
sw	00	store word	xxxxxx	add	010
beq	01	branch eq	xxxxxx	subtract	110
R-type	10	add	100000	add	010
R-type	10	subtract	100010	subtract	110
R-type	10	AND	100100	and	000
R-type	10	OR	100101	or	001
R-type	10	slt	101010	slt	111



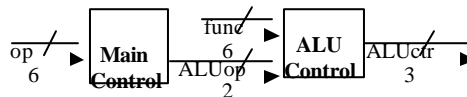
CSE 141

Allan Snively

Generating individual ALU signals

ALUop	Function	ALU signals
00	xxxx	010
01	xxxx	110
10	0000	010
10	0010	110
10	0100	000
10	0101	001
10	1010	111

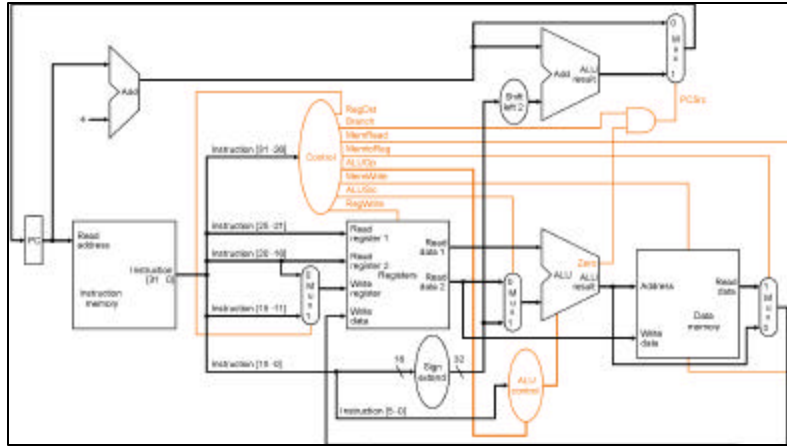
$$\begin{aligned} \text{ALUctr2} &= \overline{\text{ALUop0}} \mid (\overline{\text{ALUop1}} \ \& \ \text{F1}) \\ \text{ALUctr1} &= \text{ALUop1} \mid \text{F2} \\ \text{ALUctr0} &= \text{ALUop1} \ \& \ (\text{F0} \mid \text{F3}) \end{aligned}$$



CSE 141

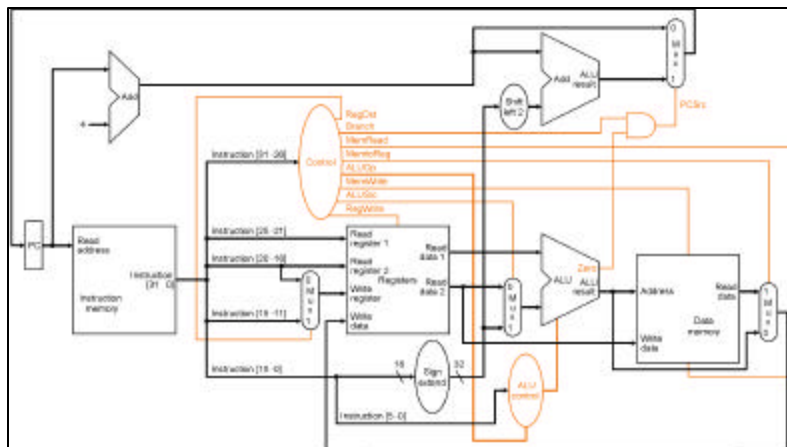
Allan Snively

sw Control



Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0
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	0	1

beq Control



Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0
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	0	1

Control Truth Table

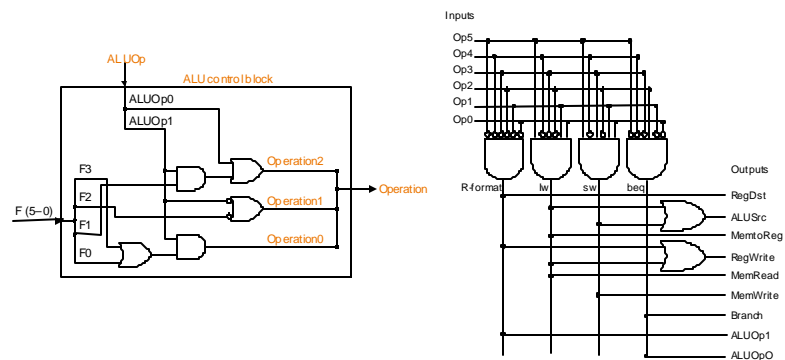
		R-format	lw	sw	beq
	Opcode	000000	100011	101011	000100
Outputs	RegDst	1	0	x	x
	ALUSrc	0	1	1	0
	MemtoReg	0	1	x	x
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
ALUOp0	0	0	0	1	

CSE 141

Allan Snively

Control

- Simple combinational logic (truth tables)



CSE 141

Allan Snively

Single-Cycle CPU Summary

- Easy, particularly the control
- Which instruction takes the longest? By how much? Why is that a problem?
- performance = insts * cpi * cycle time
- What else can we do?
- When does a multi-cycle implementation 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.