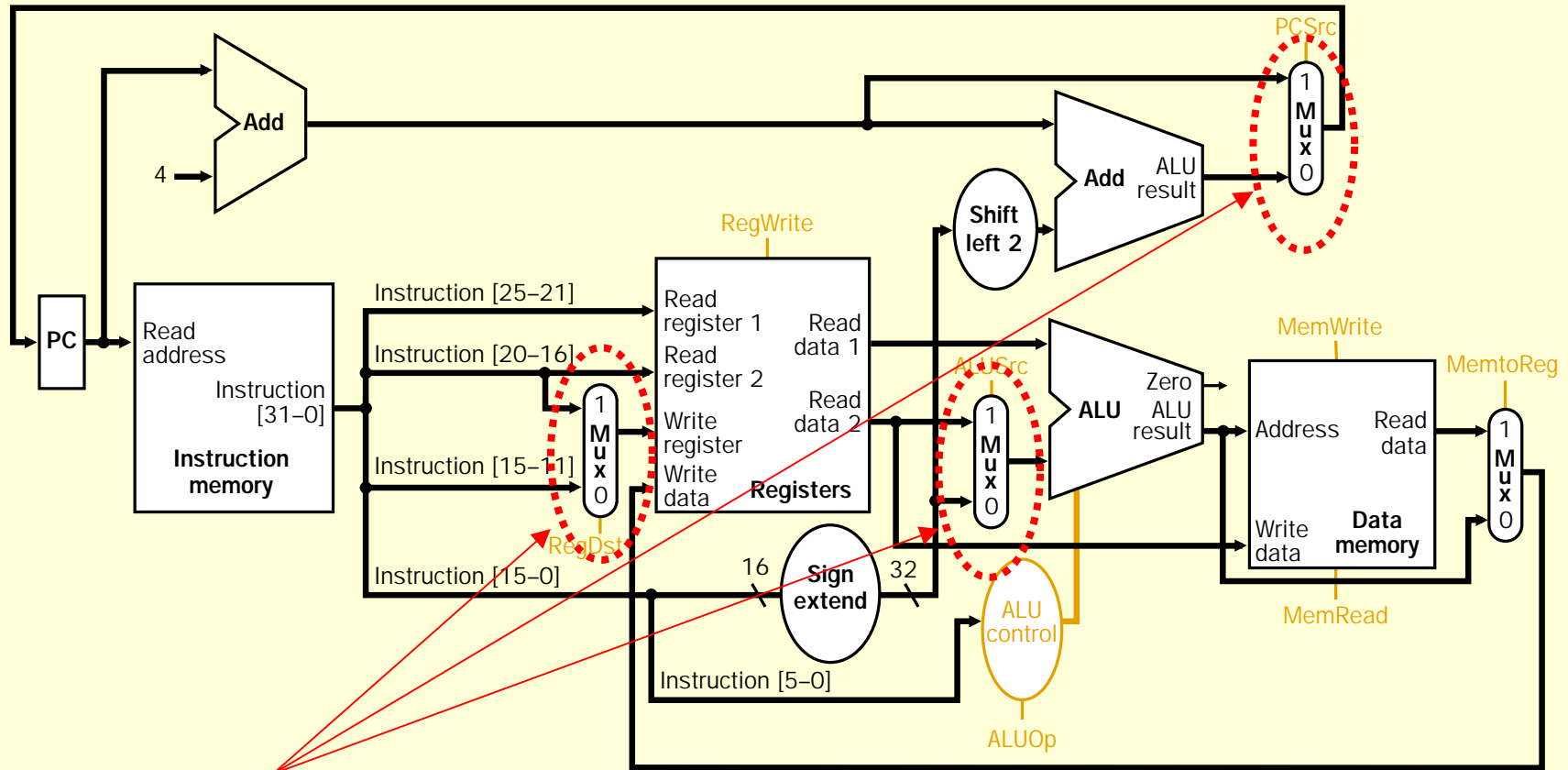


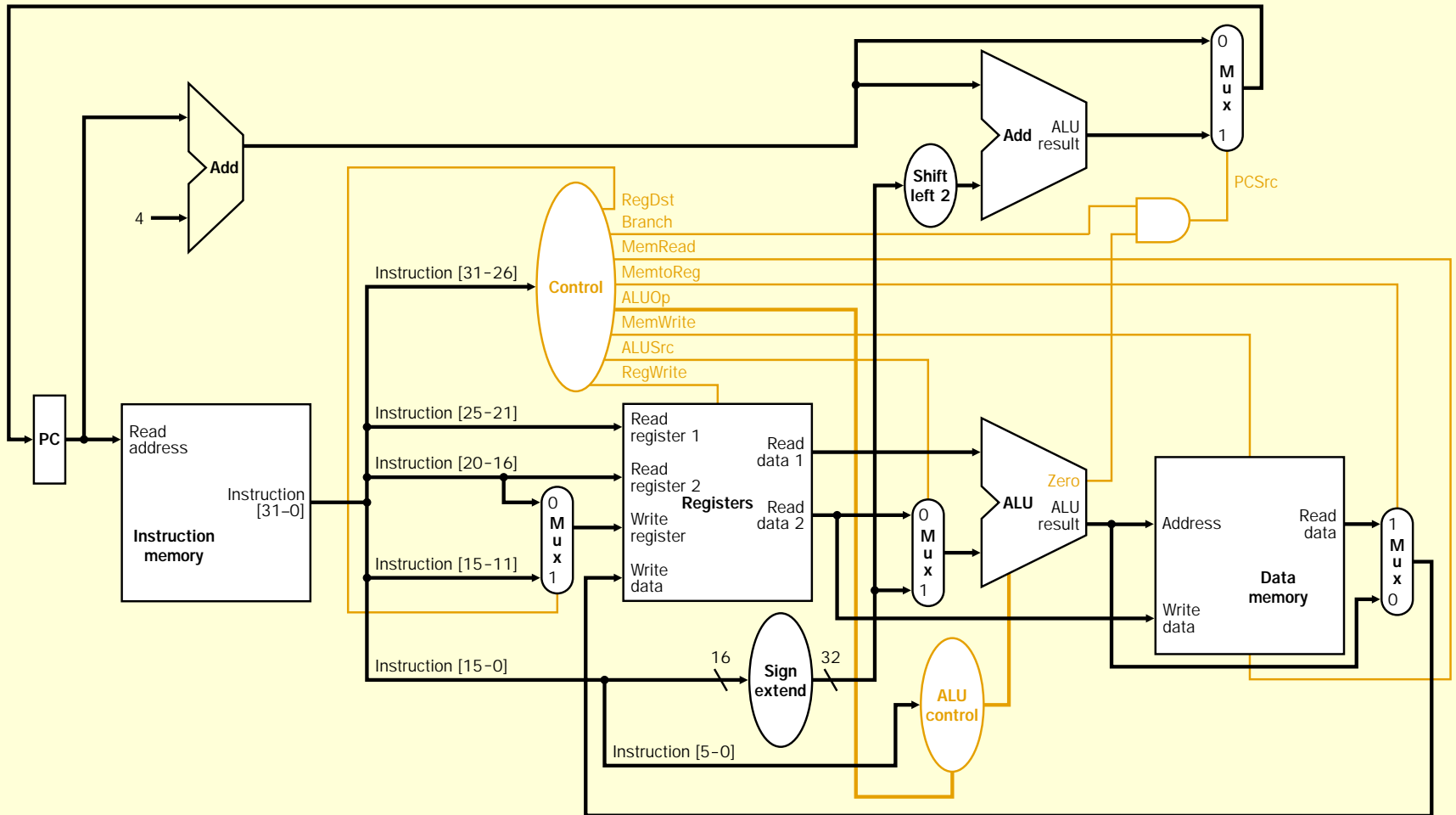
Single Cycle Datapath

- We have everything except control signals



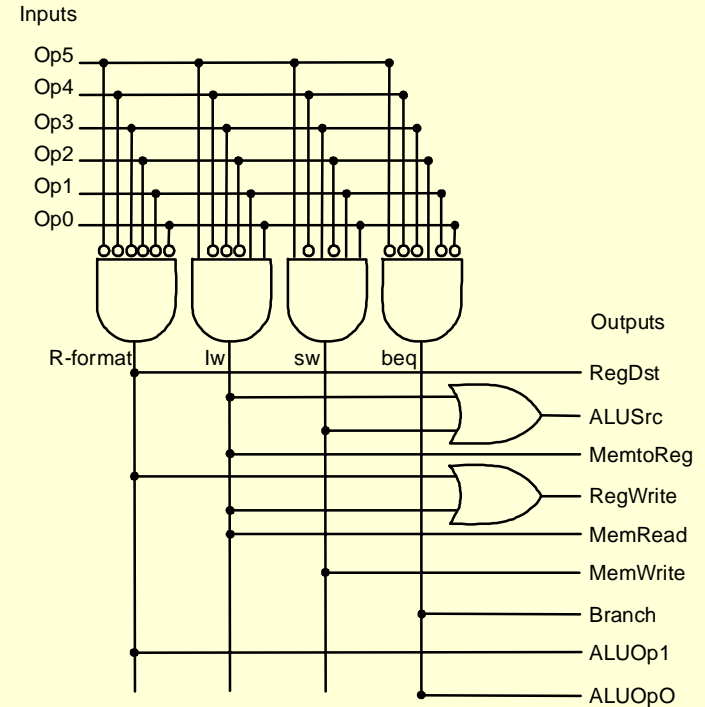
Warning! Text is inconsistent. MUX control signals sometimes have "1" is on top, sometimes "0". On exercises&test, look carefully!

Adding Control Signals



Generating the control signals

		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



PLA for control signals

Last detail - ALU control

- Don't worry too much about the details of the following three slides
 - The main point is that we control the ALU using bits both from opcode and funct field.

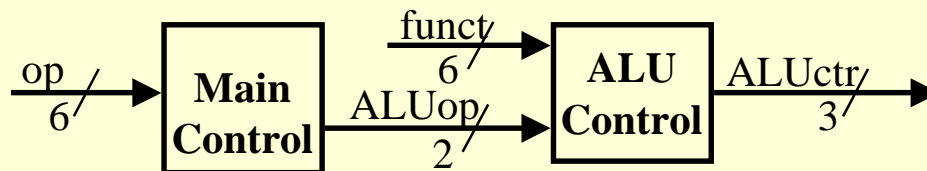
ALU control bits

- Suppose ALU has 5 functions with control bits

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

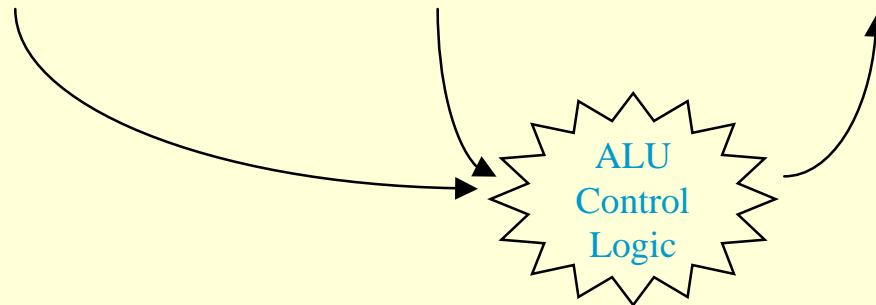
- We'll generate ALU control from opcode (bits 31-26) and funct field (bits 5-0) of instruction
- ALU doesn't need to know all opcodes--we will summarize opcode with ALUOp (2 bits):

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

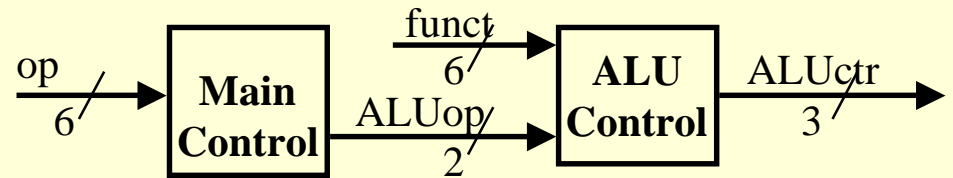


Generating ALU control

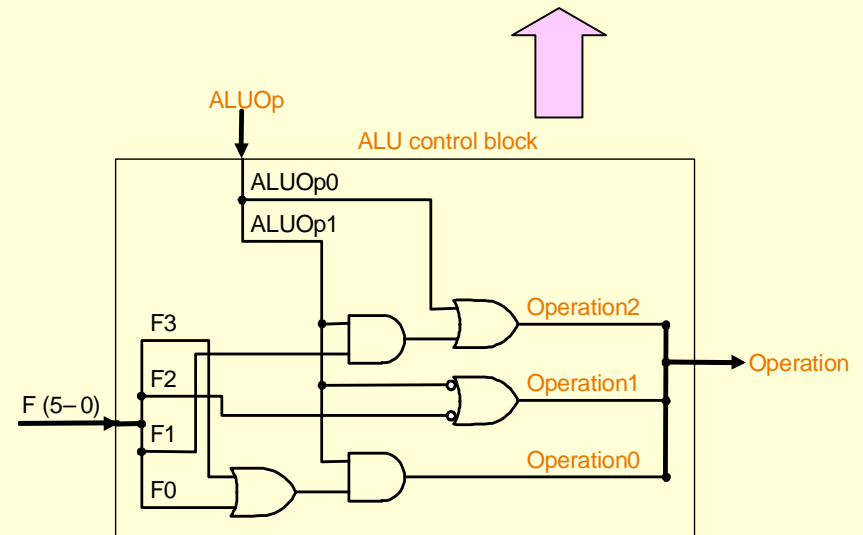
Instruction opcode	ALUOp	Instruction operation	Function code	Desired ALU action	ALU control input
lw	00	load word	xxxxxx	add	010
sw	00	store word	xxxxxx	add	010
beq	01	branch eq	xxxxxx	subtract	110
R-type	10	add	100000	add	010
R-type	10	subtract	100010	subtract	110
R-type	10	AND	100100	and	000
R-type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



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



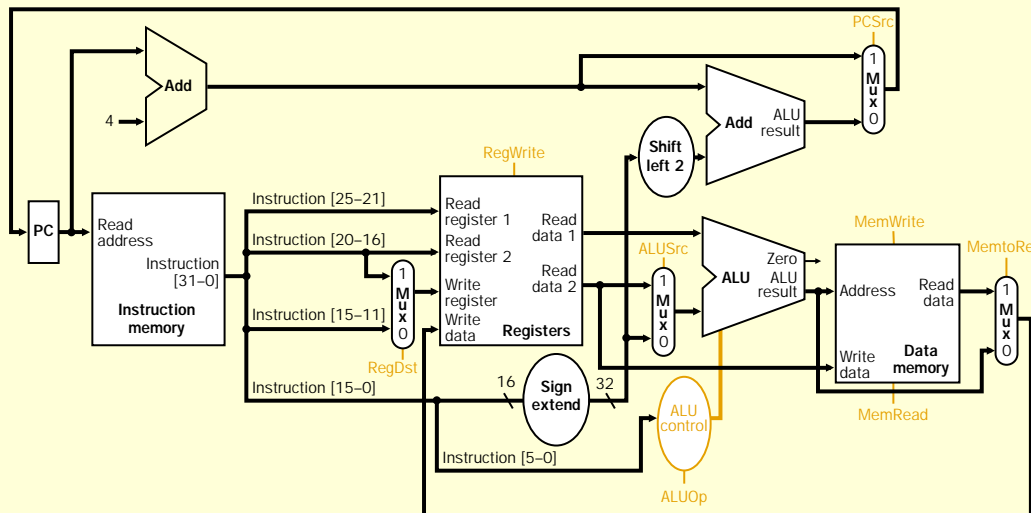
Random logic for ALU control

Don't worry about the details of ALUctr
 But DO worry about other control signals!

Single-Cycle CPU clock cycle time

Critical path: a path through combinational circuit that takes as long or longer than any other.

	I cache	Decode, R-Read	ALU	PC update	D cache	R-Write	Total
R-type	1	1	.9	-	-	.8	3.7
Load	1	1	.9	-	1	.8	4.7
Store	1	1	.9	-	1	-	3.9
beq	1	1	.9	.1	-	-	3.0



Clock cycle time = 4.7 + setup + hold

Single-Cycle CPU Summary

- Easy, particularly the control
- Which instruction takes the longest? By how much? Why is that a problem?
- Execution time = $\text{insts} * \text{cpi} * \text{cycle time}$
- Real machines have much *more* variable instruction latencies than this small subset.

Why use multicycle design?

- Problem: In single-cycle design, cycle time must be long enough for longest instruction
- Solution: break execution into smaller tasks
 - each task takes a cycle;
 - different instructions require different numbers of cycles
- Another advantage: May need fewer logic blocks
 - One ALU instead of ALU plus 2 adders
 - Only need one unified (instruction + data) cache

Multicycle implementation

Goal: balance amount of work done each cycle.

	I cache	Decode, R-Read	ALU	PC update	D cache	R- Write	Total
R-type	1	1	.9	-	-	.8	3.7
Load	1	1	.9	-	1	.8	4.7
Store	1	1	.9	-	1	-	3.9
beq	1	1	.9	.1	-	-	3.0

- Load needs 5 cycles
- Store and R-type need 4
- beq needs 3

Will multicycle design be faster?

	I cache	Decode, R-read	ALU	PC update	D cache	R-write	Total
R-type	1	1	.9	-	-	.8	3.7
Load	1	1	.9	-	1	.8	4.7
Store	1	1	.9	-	1	-	3.9
beq	1	1	.9	.1	-	-	3.0

Let's assume setup + hold time = 0.1 ns

Single cycle design:

Clock cycle time = 4.7 + 0.1 = 4.8 ns

time/inst = 1 cycle/inst * 4.8 ns/cycle = 4.8 ns/inst

Multicycle design:

Clock cycle time = 1.0 + 0.1 = 1.1

time/inst = CPI * 1.1 ns/cycle

It depends on the program!

	Cycles needed	Instruction frequency
R-type	4	60%
Load	5	20%
Store	4	10%
beq	3	10%

What is CPI assuming this instruction mix???

Let's assume setup + hold time = 0.1 ns

Single cycle design:

$$\text{Clock cycle time} = 4.7 + 0.1 = 4.8 \text{ ns}$$

$$\text{time/inst} = 1 \text{ cycle/inst} * 4.8 \text{ ns/cycle} = 4.8 \text{ ns/inst}$$

Multicycle design:

$$\text{Clock cycle time} = 1.0 + 0.1 = 1.1$$

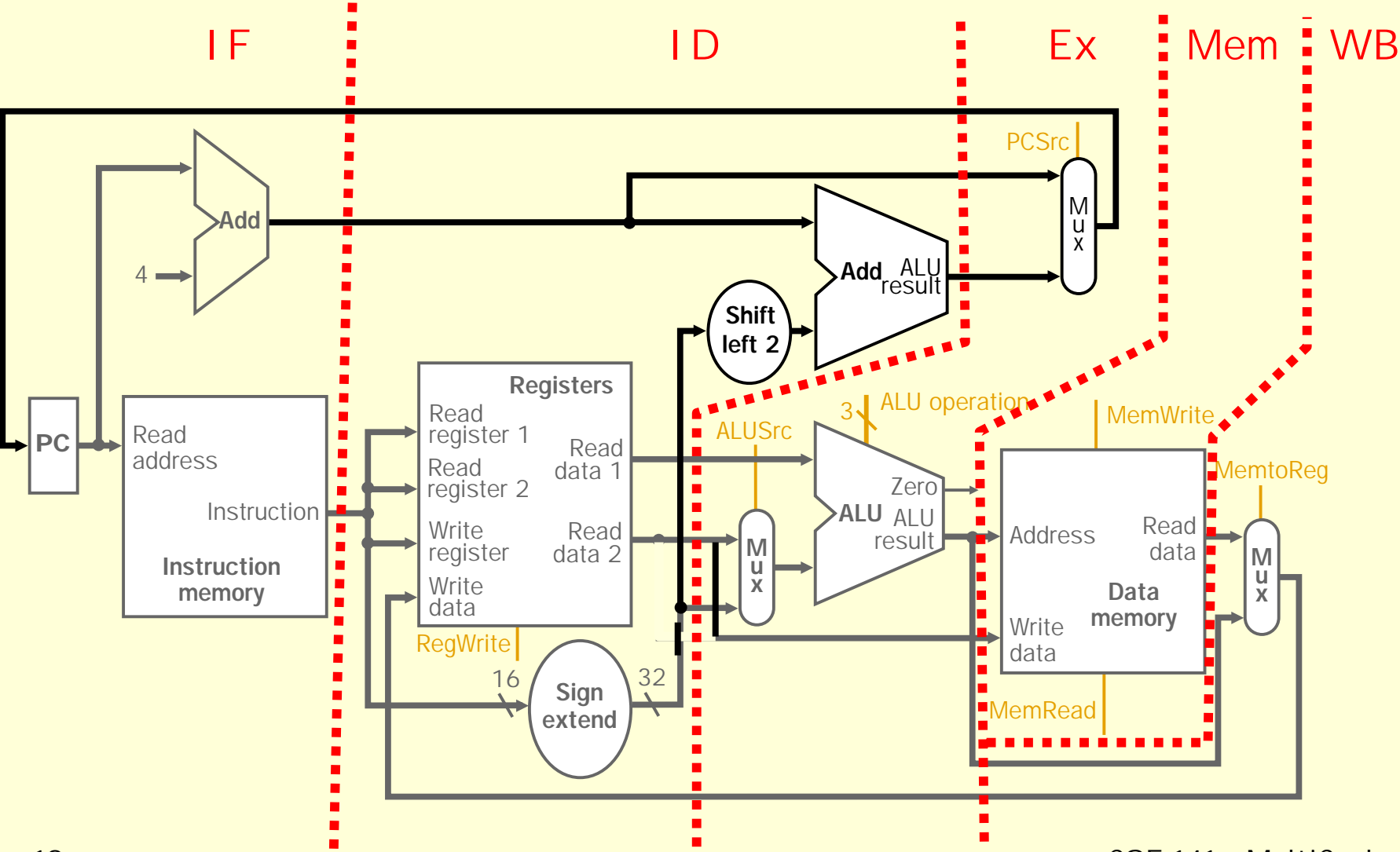
$$\text{time/inst} = \text{CPI} * 1.1 \text{ ns/cycle} = ???$$

The Five Cycles

- Five execution steps (some instructions use fewer)
 - IF: Instruction Fetch
 - ID: Instruction Decode (& register fetch & add PC+immed)
 - EX: Execute
 - Mem: Memory access
 - WB: Write-Back into registers

	IF	ID	EX	Mem	WB	Total	
	I cache	Decode, R-Read	ALU	PC update	D cache	R-Write	
R-type	1	1	.9	-	-	.8	3.7
Load	1	1	.9	-	1	.8	4.7
Store	1	1	.9	-	1	-	3.9
beq	1	1	.9	.1	-	-	3.0

Partitioning the Single-Cycle Design



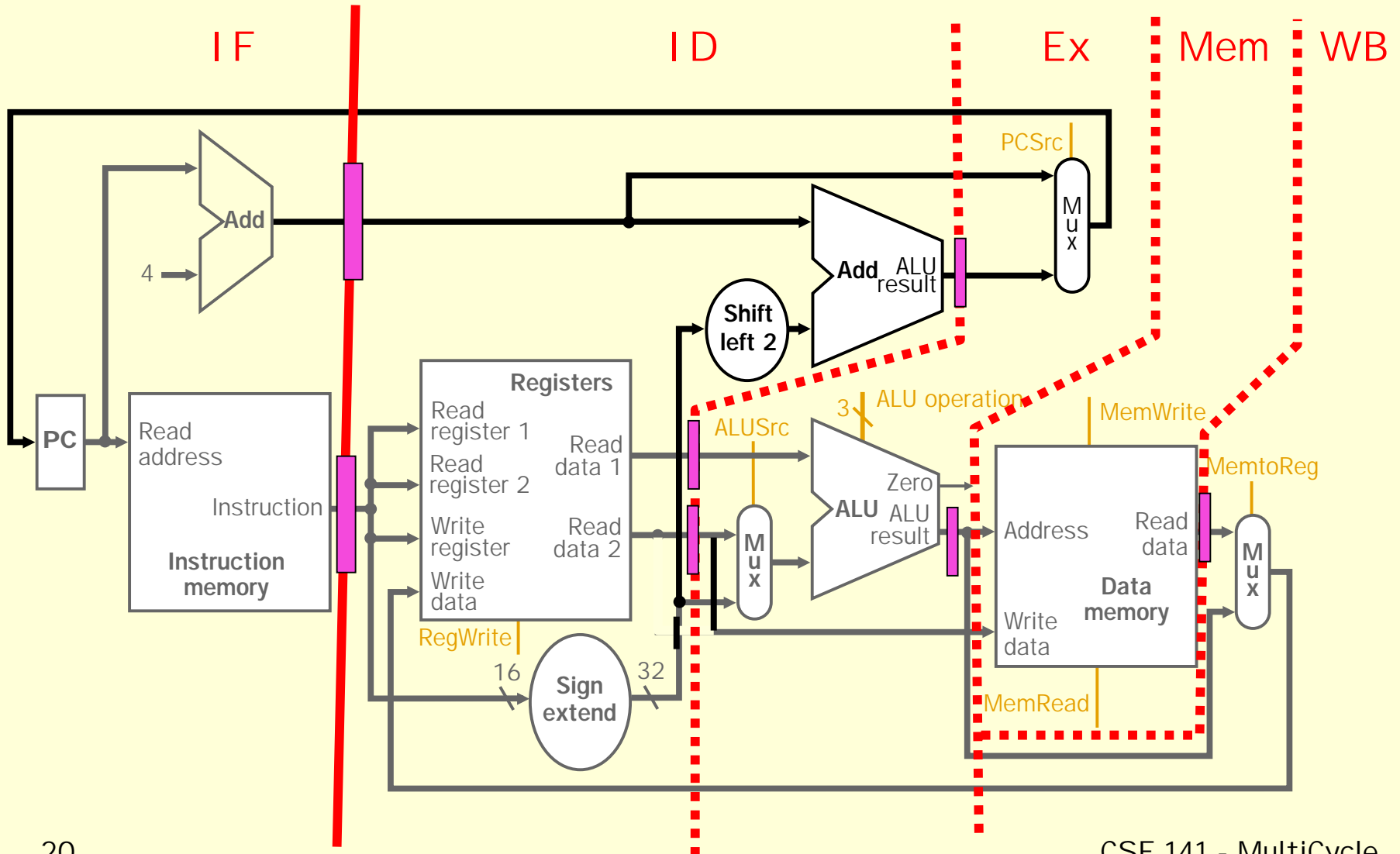
Adding State Elements

Since we reuse logic (e.g. ALU), we need to store results between states

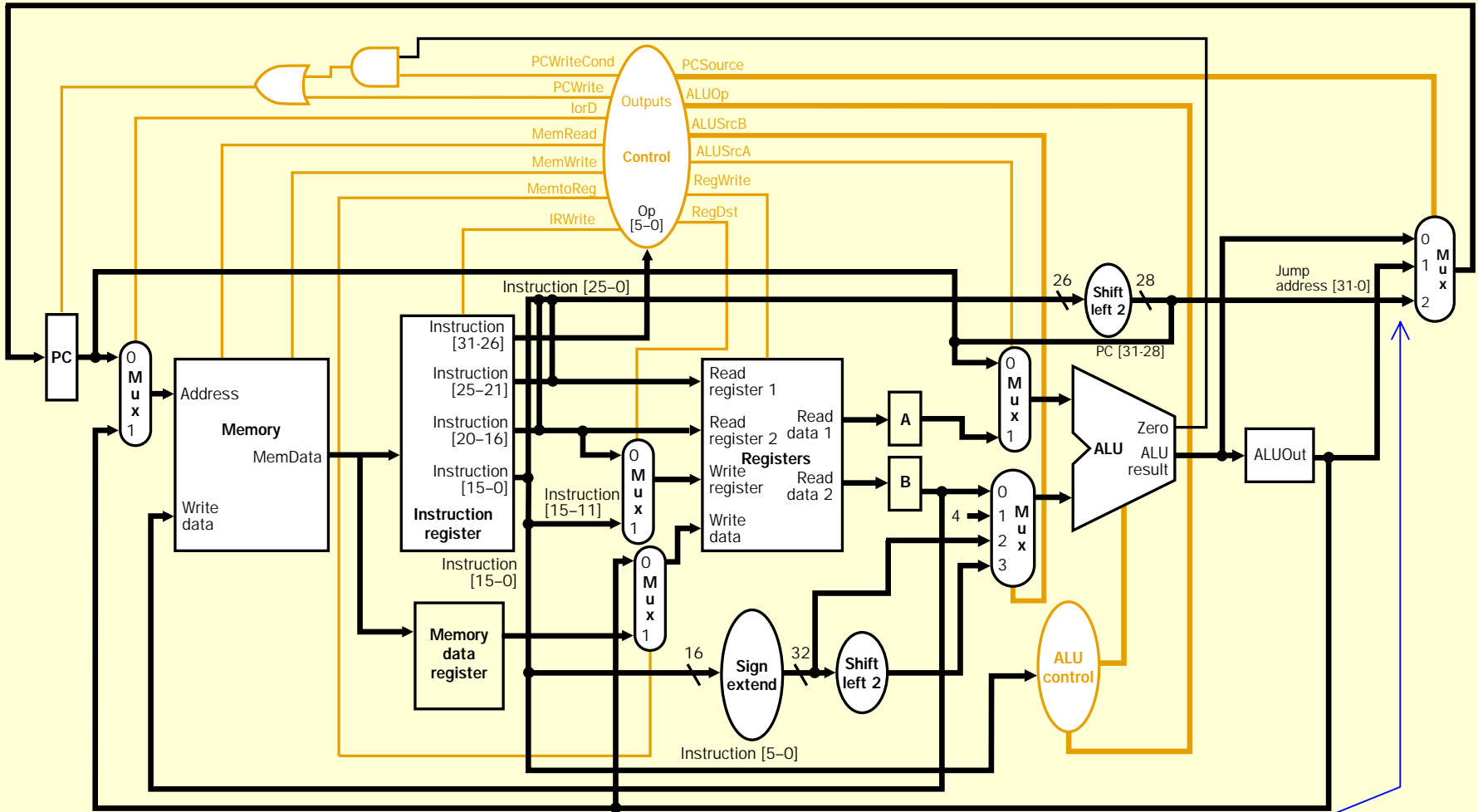
Need extra registers when:

- signal is computed in one clock cycle and used in another, AND
- the inputs to the combinational circuit can change before the signal is written into a state element.

Where to add registers (more or less)



Complete Multicycle Datapath



(note: logic for jump instruction now included)

Summary of execution steps

Step	R-type	Memory	Branch
Instruction Fetch	$IR = Mem[PC]$ $PC = PC + 4$		
Instruction Decode/ register fetch	$A = Reg[IR[25-21]]$ $B = Reg[IR[20-16]]$ $ALUout = PC + (sign-extend(IR[15-0]) \ll 2)$		
Execution, address computation, branch completion	$ALUout = A \text{ op } B$	$ALUout = A +$ sign- extend($IR[15-0]$)	if ($A==B$) then $PC=ALUout$
Memory access or R- type completion	$Reg[IR[15-11]] =$ $ALUout$	memory-data = $Mem[ALUout]$ <i>or</i> $Mem[ALUout]=$ B	
Write-back		$Reg[IR[20-16]] =$ memory-data	

We'll go through these in exacting detail

Computer of the Day

The LGP 30 (Made by the Royal McBee Computer Corp.)

A first generation computer (1956)

- 113 vacuum tubes, 1450 diodes.

Accumulator ISA, 16 instructions - 4 OP, 12(?) operand bits

- A = add, B = ring bell, C = clear, D = divide, ..., Z = stop
- Self modifying code needed (see "Olden Days" on class page)

Magnetic drum memory - 4096 31-bit words

- 120 KHz clock, but you had to wait for drum to rotate
- Good placement of data on drum was important

First "desk computer". Cost ~\$40,000.

- 800 pounds, 1500 watts.
- Paper tape I/O. Read or punch 10 6-bit characters/sec.

Most important rule: wait 45 minutes after turning off.

- Otherwise, you may scrape the drum memory