

# Control

---

- Selecting the operations to perform (ALU, read/write, etc.)
- Controlling the flow of data (multiplexor inputs)
- Information comes from the 32 bits of the instruction
- Example:

**add \$8, \$17, \$18**

**Instruction Format:**

000000	10001	10010	01000	00000	100000
op	rs	rt	rd	shamt	funct

- ALU's operation based on instruction type and function code

# Control

---

- e.g., what should the ALU do with this instruction
- Example: `lw $1, 100($2)`

35	2	1	100
----	---	---	-----

op	rs	rt	16 bit offset
----	----	----	---------------

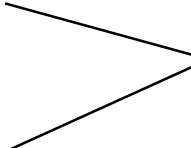
- ALU control input

000	AND
001	OR
010	add
110	subtract
111	set-on-less-than

- Why is the code for subtract 110 and not 011?

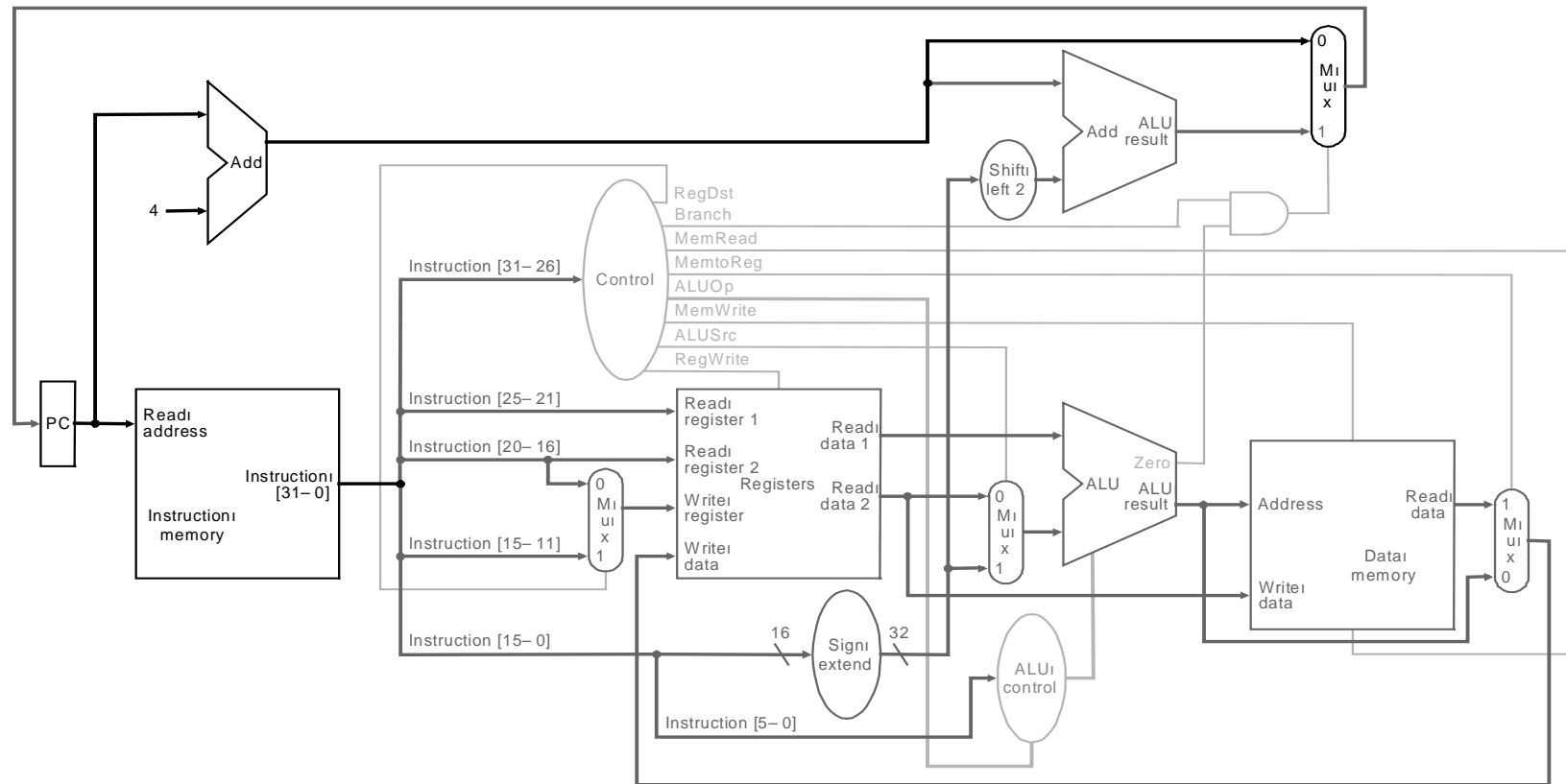
# Control

---

- Must describe hardware to compute 3-bit ALU control input
    - given instruction type
      - 00 = lw, sw
      - 01 = beq,
      - 11 = arithmetic
    - function code for arithmetic
-  **ALUOp**  
computed from instruction type
- Describe it using a truth table (can turn into gates):

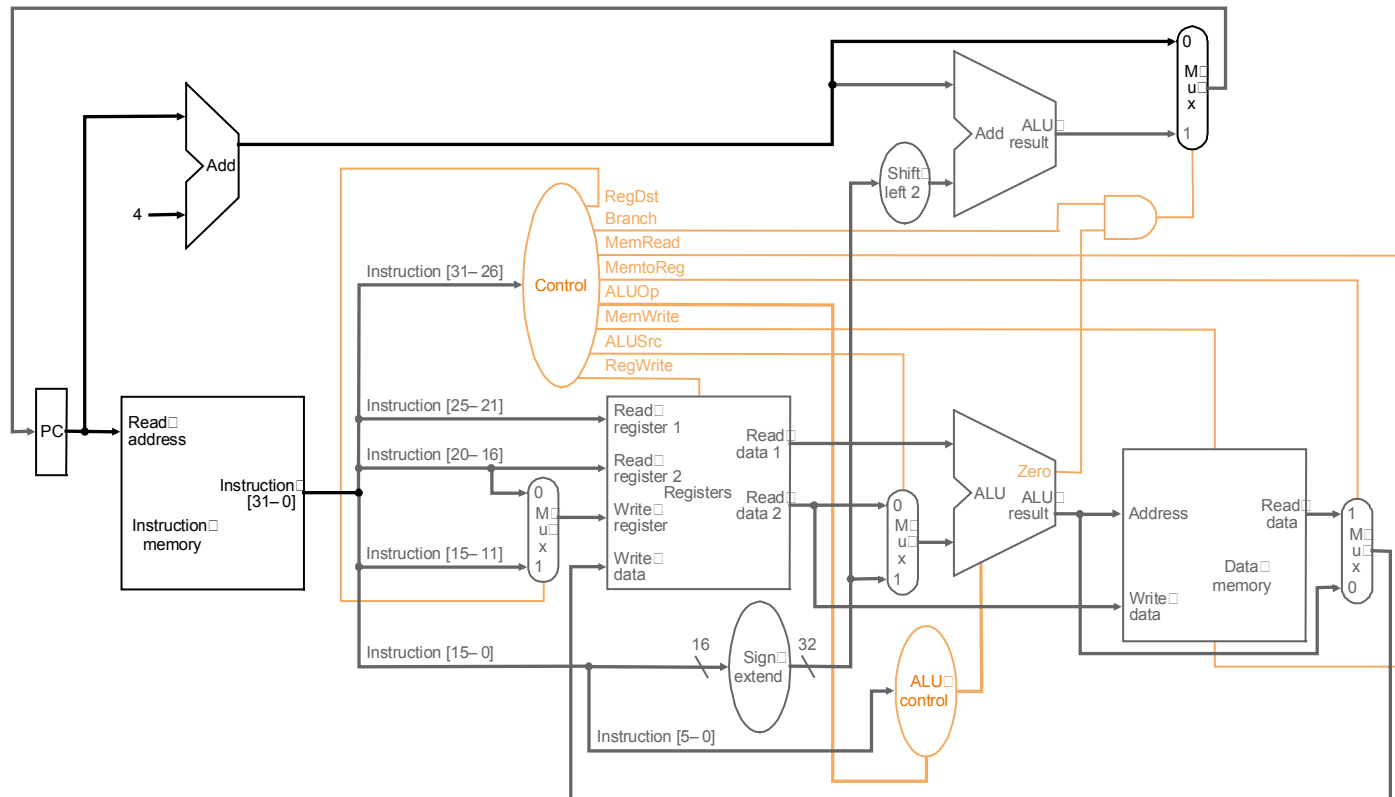
ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	010
X	1	X	X	X	X	X	X	110
1	X	X	X	0	0	0	0	010
1	X	X	X	0	0	1	0	110
1	X	X	X	0	1	0	0	000
1	X	X	X	0	1	0	1	001
1	X	X	X	1	0	1	0	111

# 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	X	0	X	0	0	0	1	0	1

# Control

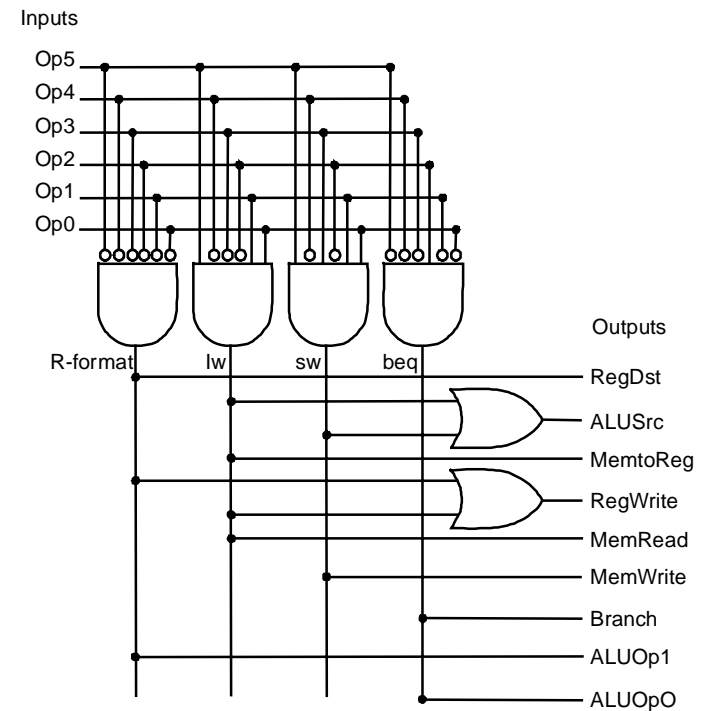
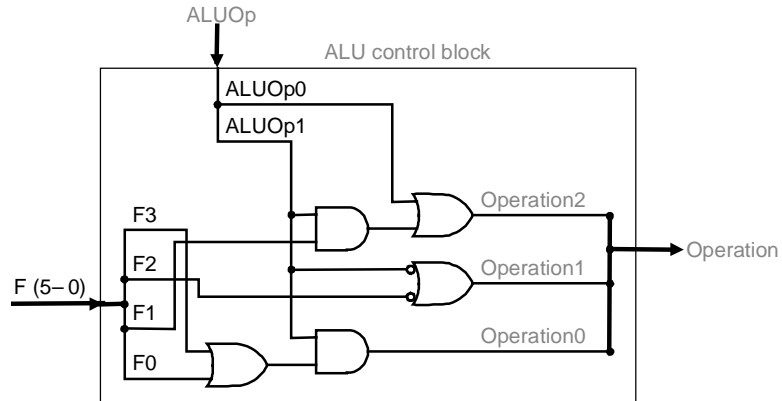


ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	010
X	1	X	X	X	X	X	X	110
1	X	X	X	0	0	0	0	010
1	X	X	X	0	0	1	0	110
1	X	X	X	0	1	0	0	000
1	X	X	X	0	1	0	1	001
1	X	X	X	1	0	1	0	111

Instruction	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
Rformat	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	X	0	X	0	0	0	1	0	1

# Control

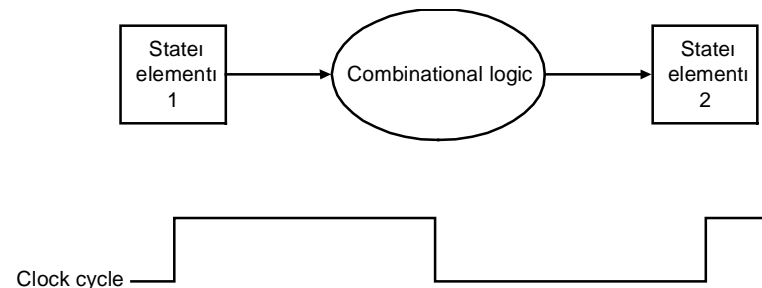
- Simple combinational logic (truth tables)



# Our Simple Control Structure

---

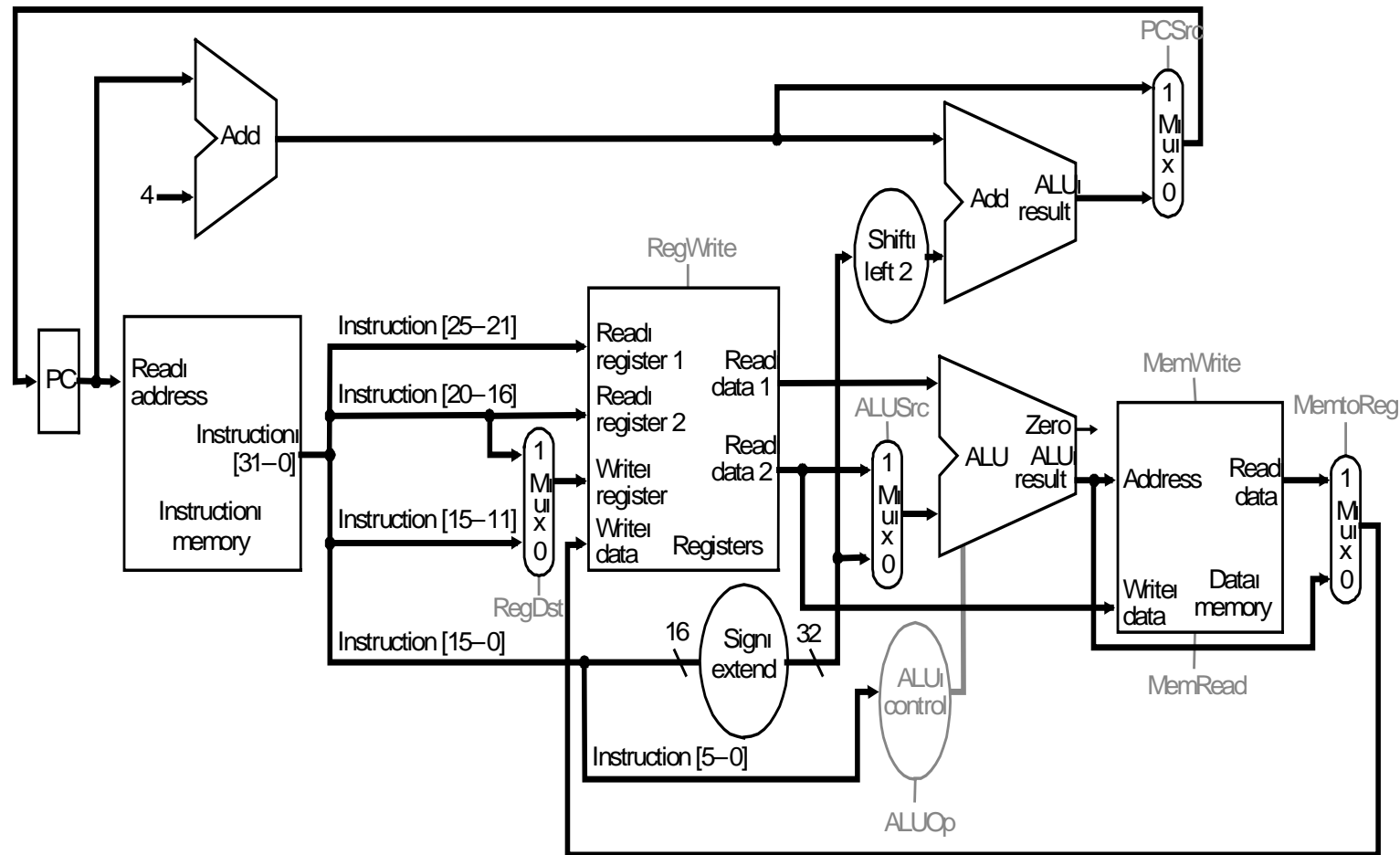
- All of the logic is combinational
- We wait for everything to settle down, and the right thing to be done
  - ALU might not produce “right answer” right away
  - we use write signals along with clock to determine when to write
- Cycle time determined by length of the longest path



*We are ignoring some details like setup and hold times*

# Single Cycle Implementation

- Calculate cycle time assuming negligible delays except:
  - memory (2ns), ALU and adders (2ns), register file access (1ns)





# Where we are headed

---

- **Single Cycle Problems:**
  - what if we had a more complicated instruction like floating point?
  - wasteful of area
- **One Solution:**
  - use a “smaller” cycle time
  - have different instructions take different numbers of cycles
  - a “multicycle” datapath:

