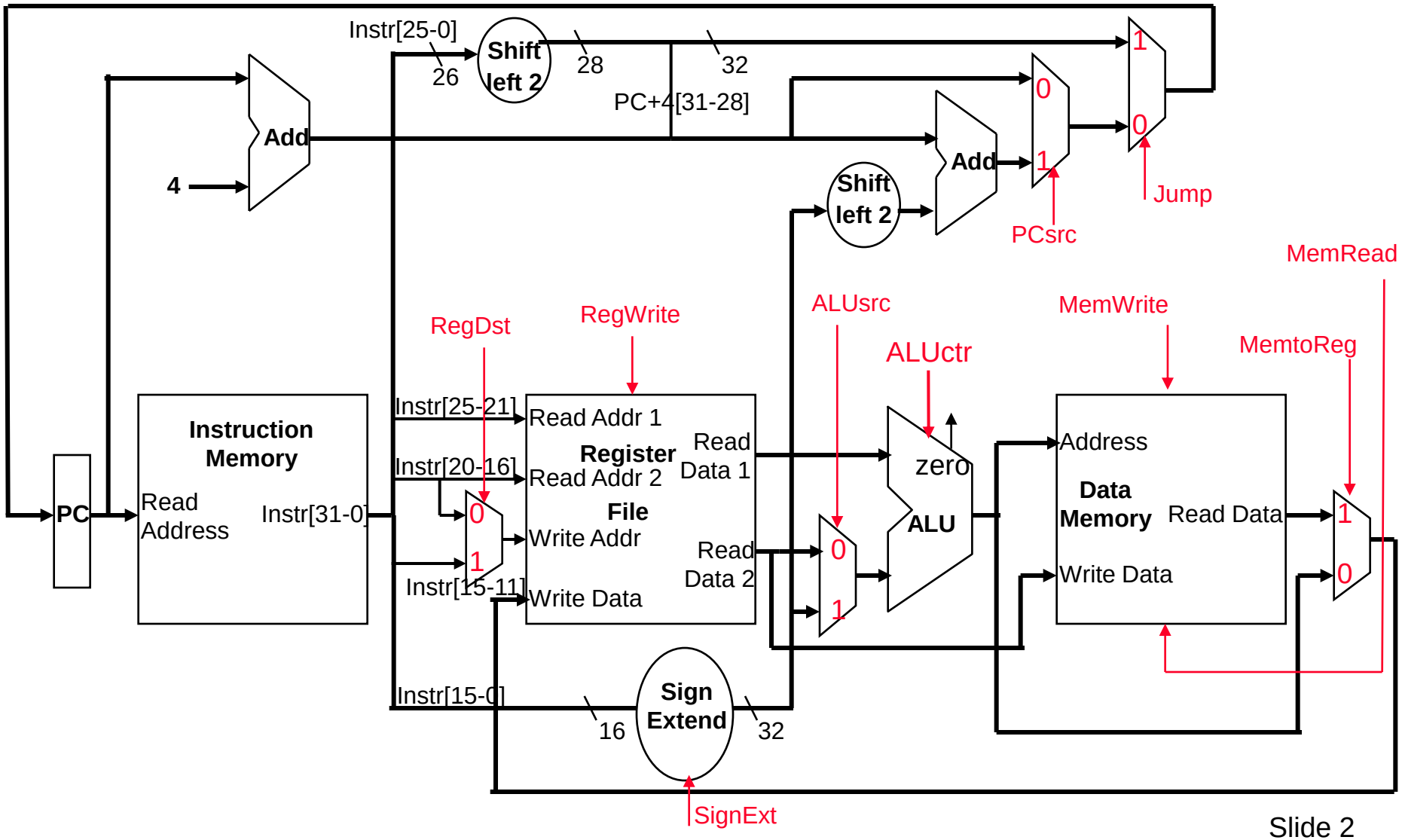
Principles of Computer Organization

Lecture 8: Designing Control for Single-cycle Processor

Yanyan Liang
Faculty of Information Technology
Macau University of Science and Technology

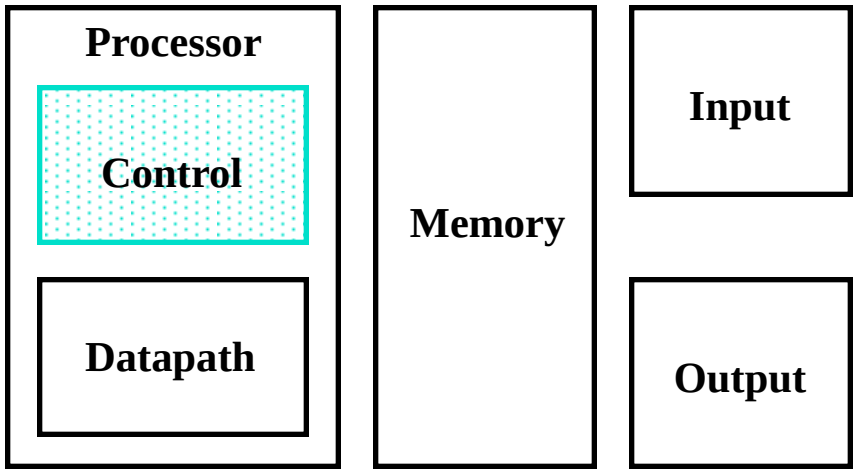
Recap: A Single Cycle Datapath

- We have everything except control signals, this lecture will show you how to generate the control signals.



The Big Picture: Where are we now?

- **The Five Classic Components of a Computer**



- **Today's Topic: Designing the Control for the Single Cycle Datapath**

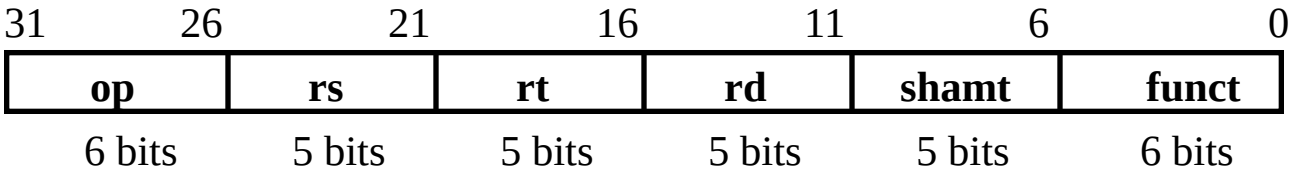
Step 4: Designing control signals

- **Control signals are used to control the datapath to perform necessary operations.**
- **Need to decide what control signals are required such that we can fully control our processor to perform calculation.**
- **Example: add t0, t1, t2**
  - **Need the ALU to perform addition.**
  - **As a result, we need a control signal "ALUctr" to control the ALU to perform necessary operation.**
- **Will be addressed in next lecture.**

Meaning of the Control Signals

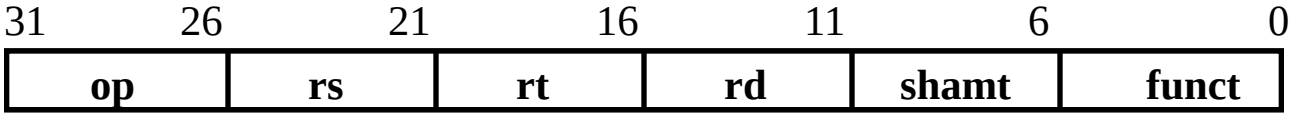
- **RegDst: Register destination. 0 → rt, 1 → rd.**
- **RegWrite: Write register file or not. 0 → no, 1 → yes.**
- **ALUctr: Operation of ALU, “add”, “sub”, “or”, etc.**
- **ALUsrc: Select data for ALU. 0 → Register file, 1 → Immediate.**
- **MemWrite: Write data memory or not. 0 → no, 1 → yes.**
- **MemRead: Read data memory or not. 0 → no, 1 → yes.**
- **MemtoReg: Select data for writing register file. 0 → ALU, 1 → data memory.**
- **PCsrc: Branch or not. 0 → no, 1 → yes.**
- **Jump: Jump or not. 0 → no, 1 → yes.**
- **SignExt: Perform sign or zero extend. “sign” or “zero”.**

Steps in add instruction



- **add rd, rs, rt**
  - **mem[PC]** — **Fetch the instruction from memory**
  - **R[rd] <- R[rs] + R[rt]** — **The actual operation**
  - **PC <- PC + 4** — **Calculate the next instruction's address**

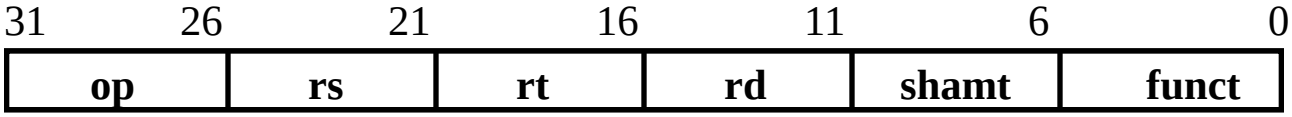
The Single Cycle Datapath during add



- R[rd] <- R[rs] + R[rt]

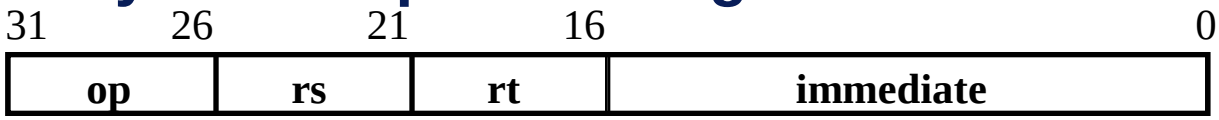
x: means we don't care

The Single Cycle Datapath during sub??



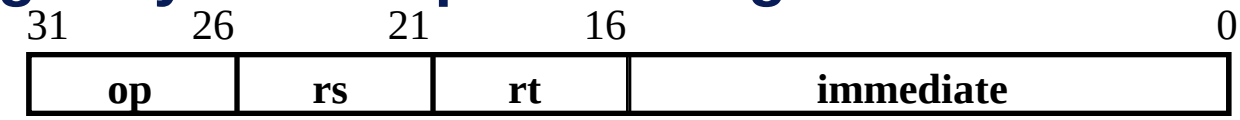
- R[rd] <- R[rs] - R[rt] (sub rd, rs, rt)

The Single Cycle Datapath during addi??



- R[rt] <- R[rs] or SignExt[imm16] (addi t0, t1, -10)

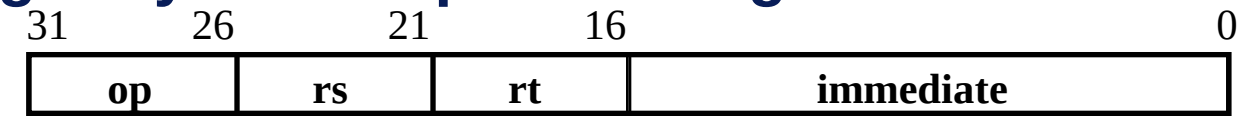
The Single Cycle Datapath during addi



- R[rt] <- R[rs] or SignExt[imm16] (addi t0, t1, -10)

Instr[25-0]
26
Shift left 2
28
32
PC+4[31-28]
Add
4
Add
Shift left 2
0
1
0
1
Jump=0
PCsrc=0
MemRead=x
ALUsrc=1
MemWrite=0
RegDst=0
RegWrite=1
ALUctr=add
MemtoReg=0
PC
Instruction Memory
Read Address
Instr[31-0]
Instr[25-21]
Instr[20-16]
Instr[15-11]
Instr[15-0]
Register File
Read Addr 1
Read Addr 2
Write Addr
Write Data
Read Data 1
Read Data 2
zero
ALU
Address
Data Memory
Read Data
Write Data
16
Sign Extend
32
SignExt=sign

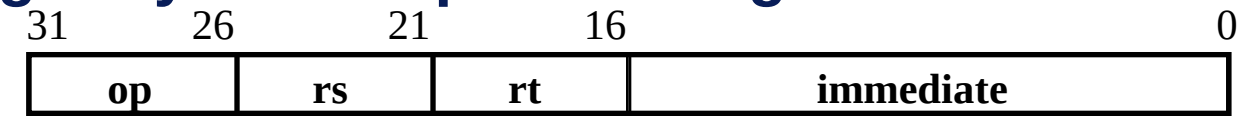
The Single Cycle Datapath during Load



- R[rt] <- Data Memory {R[rs] + SignExt[imm16]} lw rt, imm16(rs)

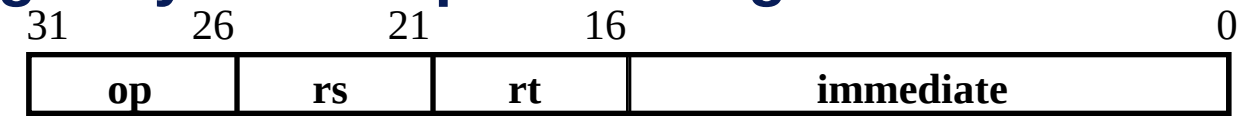
Instr[25-0]
26
Shift left 2
28
32
PC+4[31-28]
Add
4
Add
Shift left 2
0
1
1
0
Jump=0
PCsrc=0
MemRead=1
ALUsrc=1
MemWrite=0
RegDst=0
RegWrite=1
ALUctr=add
MemtoReg=1
PC
Instruction Memory
Read Address
Instr[31-0]
Instr[25-21]
Instr[20-16]
Instr[15-11]
Instr[15-0]
Register File
Read Addr 1
Read Addr 2
Write Addr
Write Data
Read Data 1
Read Data 2
0
1
zero
ALU
0
1
Address
Data Memory
Read Data
Write Data
1
0
16
Sign Extend
32
SignExt=sign

The Single Cycle Datapath during store??



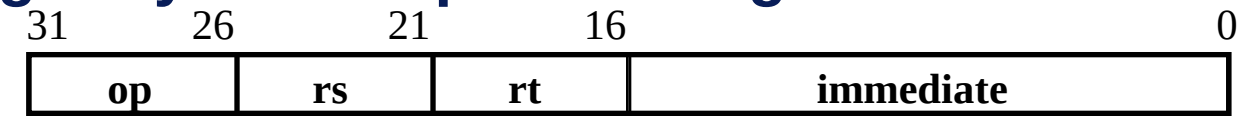
- Data Memory {R[rs] + SignExt[imm16]} <- R[rt] sw rt, imm16(rs)

The Single Cycle Datapath during store



- Data Memory {R[rs] + SignExt[imm16]} <- R[rt] sw rt, imm16(rs)

The Single Cycle Datapath during Branch

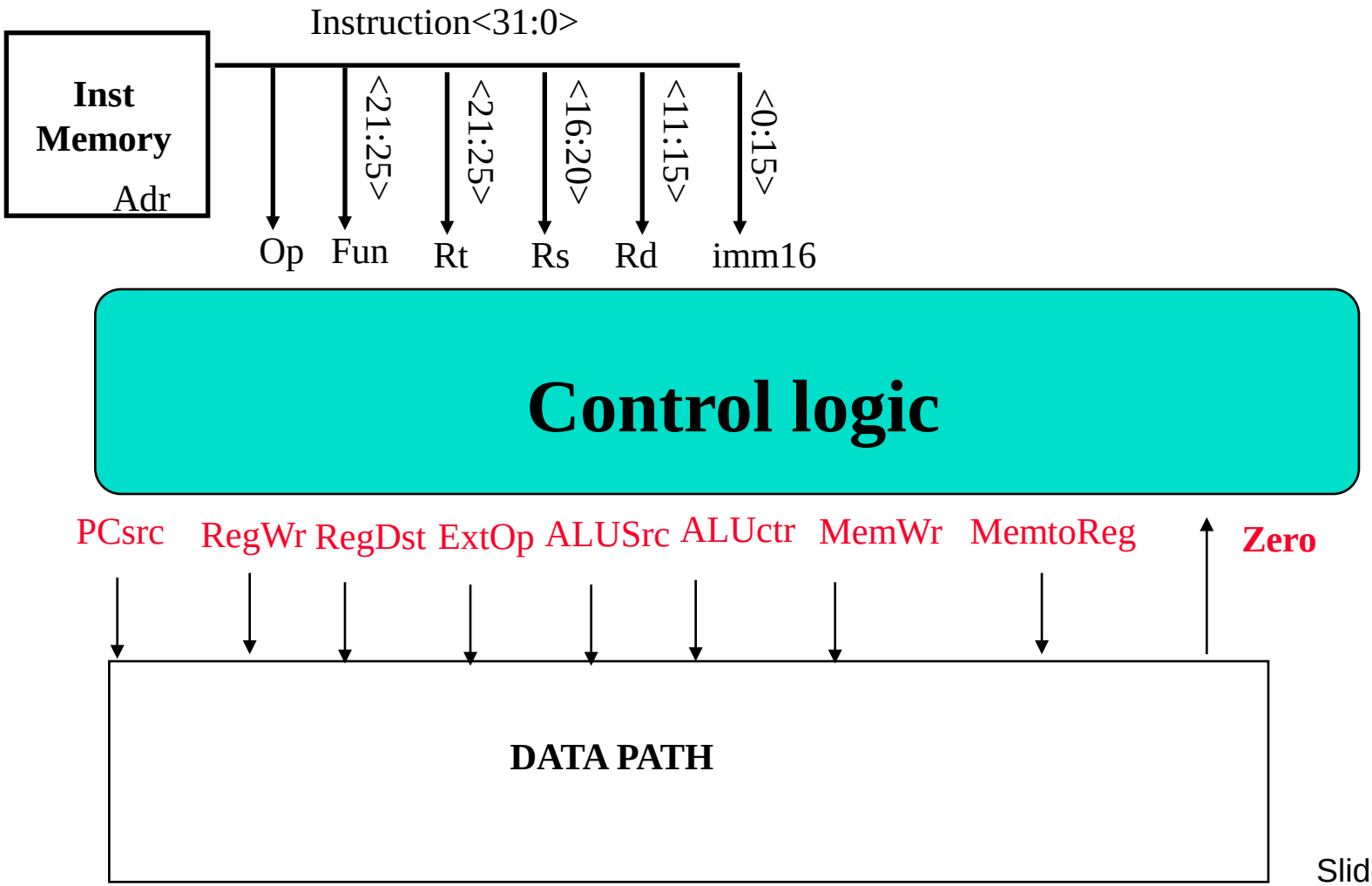


- if (R[rs] - R[rt] == 0) then zero = 1; else zero = 0 beq rs, rt, Label

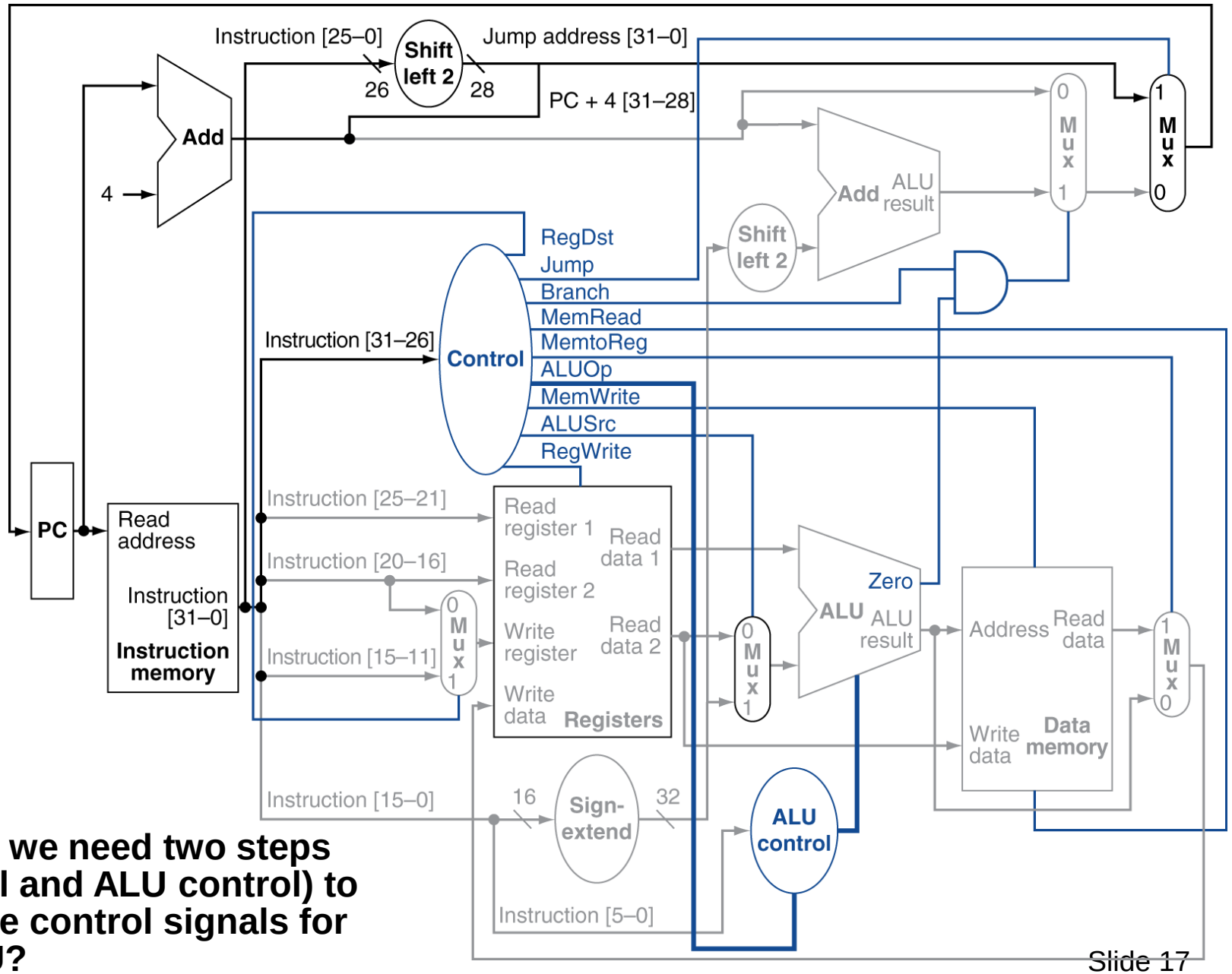
Instr[25-0]
26
Shift left 2
28
32
PC+4[31-28]
Add
4
Add
Shift left 2
1
0
0
1
Jump=0
PCsrc=1
MemRead=x
RegDst=x
RegWrite=0
ALUsrc=0
ALUctr=sub
MemWrite=0
MemtoReg=x
Instruction Memory
Read Address
Instr[31-0]
PC
Instr[25-21]
Instr[20-16]
Instr[15-11]
Instr[15-0]
Read Addr 1
Register
Read Addr 2
File
Write Addr
Write Data
Read Data 1
Read Data 2
zero
ALU
Address
Data Memory
Read Data
Write Data
16
Sign Extend
32
SignExt=sign

Step 5: Adding control logic

- **Control logic is a component used to determine the value of control signals based on instruction type.**
  - **E.g. ALUctr, RegDst, ……**

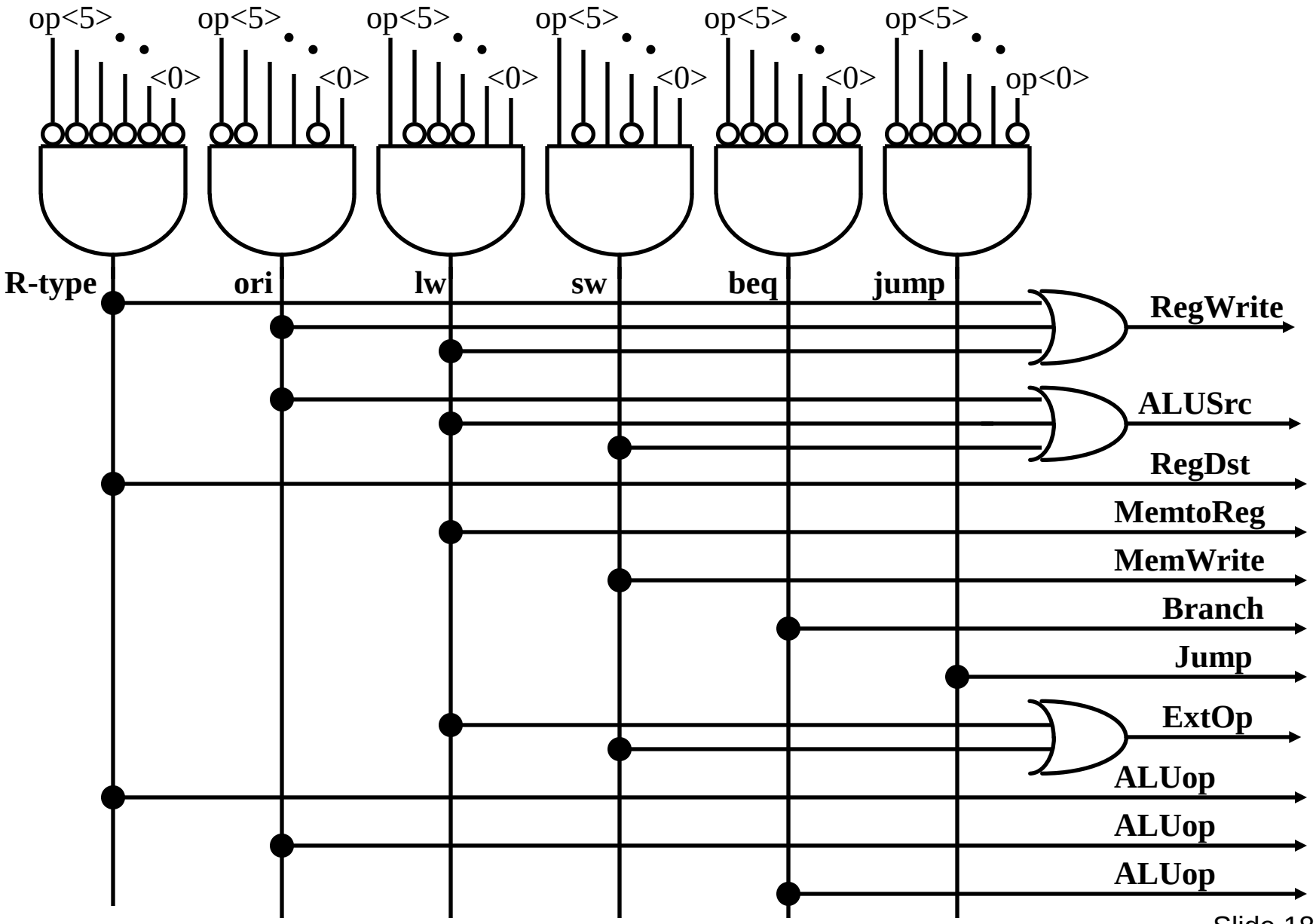


Datapath after adding control logic: Control, ALU control, and



- **Why do we need two steps (Control and ALU control) to generate control signals for the ALU?**

PLA Implementation of the Main Control



ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	AND	0000
R-type	10	OR	100101	OR	0001
R-type	10	set on less than	101010	set on less than	0111

ALUOp		Funct field						Operation
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	0010
0	1	X	X	X	X	X	X	0110
1	0	X	X	0	0	0	0	0010
1	X	X	X	0	0	1	0	0110
1	0	X	X	0	1	0	0	0000
1	0	X	X	0	1	0	1	0001
1	X	X	X	1	0	1	0	0111

Summary

- **Single cycle datapath => CPI=1**
  - **Long cycle time**
  - **Cycle time for load is much longer than needed for all other instructions**
- **5 steps to design a processor**
  - **1. Analyze instruction set => datapath requirements**
  - **2. Select set of datapath components**
  - **3. Assemble datapath meeting the requirements**
  - **4. Analyze implementation of each instruction to determine setting of control signals**
  - **5. Assemble the control logic**
- **Usually control is the hard part**
  - **Complex control => long cycle time**