

# CSE 141 – Computer Architecture Spring 2005

## Lecture 7 Single-cycle CPU Part 2

Pramod V. Argade

### Announcements

- **Discussions Sections for 141:**
  - Fridays, 10:00 - 10:50 am, Peterson Hall 104 (Chris)
  - Fridays, 2:00 - 2:50 pm, Peterson Hall 104 (Leo)
- **Reading Assignment**
  - Chapter 5. Processor: Datapath and Control
  - Sec. 5.1 - 5.4
- **Homework 4: Due Mon., April 25<sup>th</sup> in class**
  - 5.2, 5.8, 5.9, 5.10, 5.13, 5.20, 5.22, 5.28
- **Quiz**
  - When:** Mon., April 25th, First 10 minutes of the class
  - Topic:** Single Cycle CPU, Chapter 5   **Need:** Paper, pen

## Academic Honesty

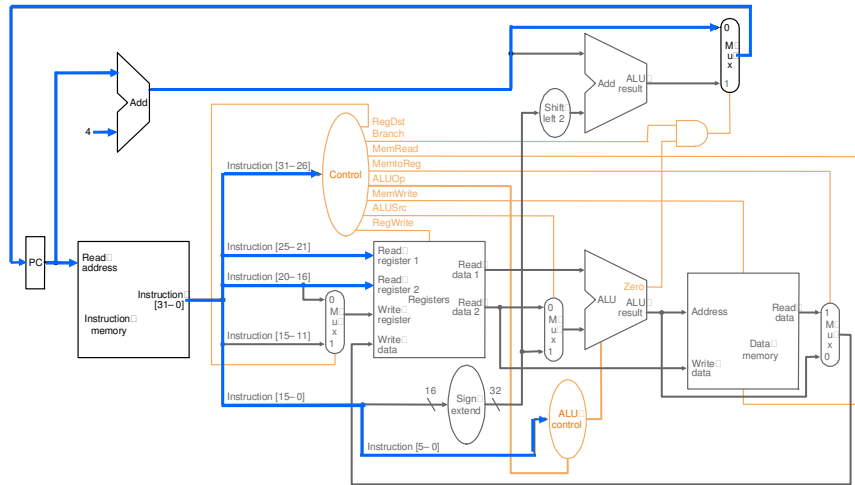
- Academic dishonesty will be taken seriously. If you are caught cheating, it will have serious consequences, including failing grade on the course.
- It is ok to discuss homework in groups
  - However, you cannot copy solution from another student. Each student must complete the homework on his/her own.
- The following is not permitted
  - Looking at a completed homework of another student.
  - Referring or copying solutions to homework from Instructor's solution manual or solutions from any source, including WWW.
  - Referring to any reference material or someone else's solution during a quiz or a test.

## Course Schedule

Lecture #	Date	Day	Lecture Topic	Quiz Topic	Homework Due
1	3/28	Monday	Introduction, Ch. 1	-	-
2	3/30	Wednesday	Performance, Ch. 4	-	-
3	4/4	Monday	ISA, Ch. 2	Performance	#1
4	4/6	Wednesday	Arithmetic, Ch. 3	-	-
5	4/11	Monday	Arithmetic, Ch. 3	ISA	#2
6	4/13	Wednesday	Single cycle CPU, Ch. 5	-	-
7	4/18	Monday	Single cycle CPU, Ch. 5	Arithmetic	#3
8	4/20	Wednesday	Multi-cycle CPU, Ch. 5	-	-
9	4/25	Monday	Multi-cycle CPU, Ch. 5	Single Cycle CPU	#4
10	4/27	Wednesday	Review for the Midterm	-	-
	5/2	Monday	Mid-term Exam	-	-
11	5/4	Wednesday	Exceptions, Ch. 5 and Pipelining, Ch. 6	-	-
12	5/9	Monday	Pipelining, Ch. 6	-	-
13	5/11	Wednesday	Data and control hazards, Ch. 6	-	-
14	5/16	Monday	Data and control hazards, Ch. 6	Pipeline Hazards	#5
15	5/18	Wednesday	Memory & cache design, Ch. 7	-	-
16	5/23	Monday	Memory & cache design, Ch. 7	Cache	#6
17	5/25	Wednesday	Virtual Memory & cache design, Ch. 7	-	-
No Class	5/30	Monday	Memorial Day Holiday	-	-
18	6/1	Wednesday	Course Review	-	-
	6/10	Friday	Final Exam	-	-



## The first phase for R-Type Instruction Instruction Fetch, Increment PC

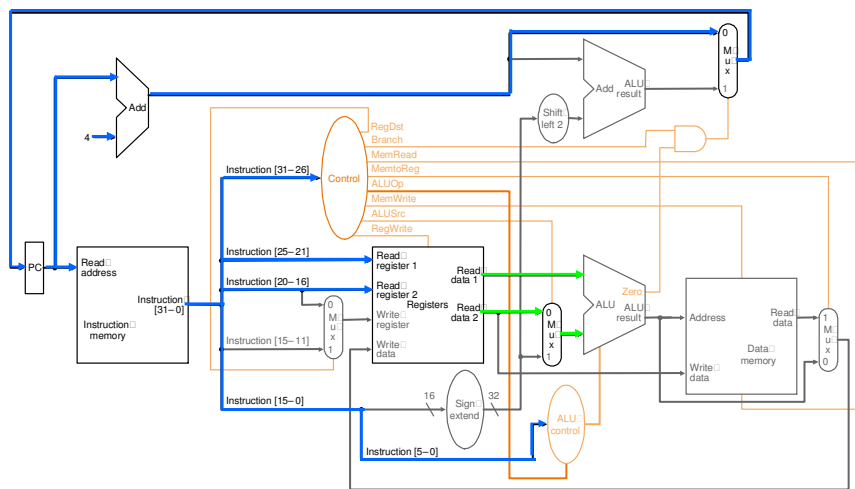


Pramod Argade

UCSD CSE 141, Spring 2005

Slide 7-7

## Second Phase for R-Type Instruction Read two Source Registers from RegFile

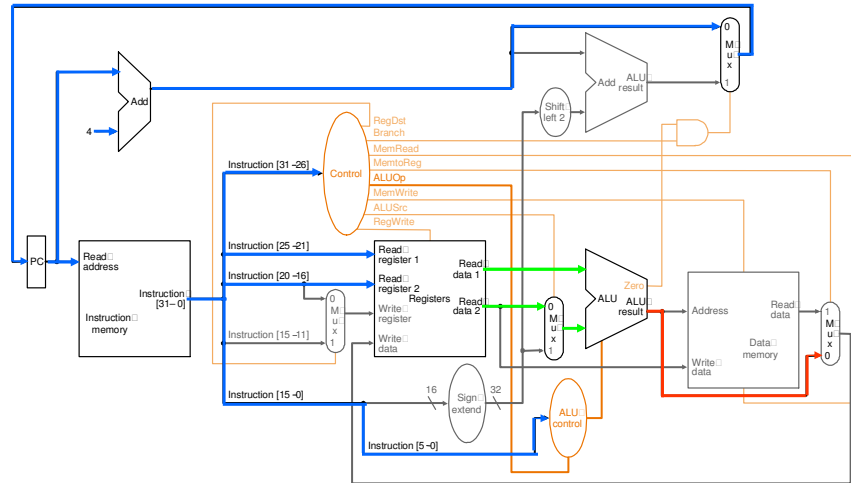


Pramod Argade

UCSD CSE 141, Spring 2005

Slide 7-8

## The Third Phase for R-Type Instruction ALU Operation

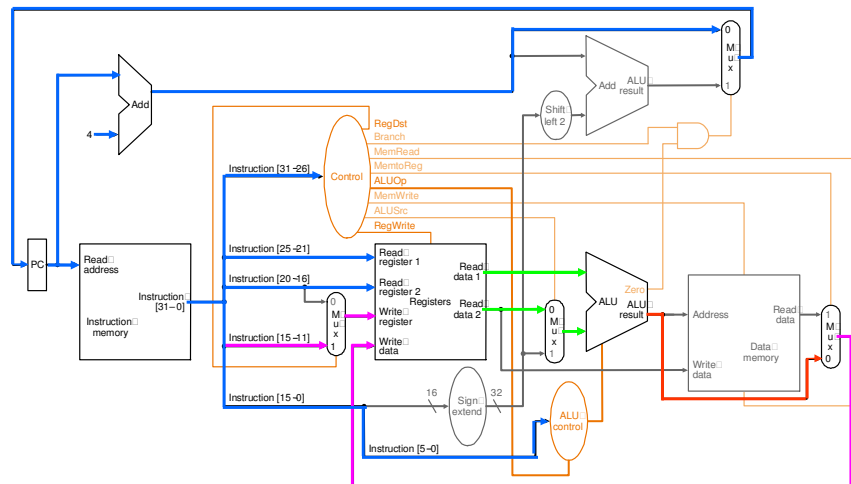


Pramod Argade

UCSD CSE 141, Spring 2005

Slide 7-9

## The Fourth (Final) Phase for R-Type Instruction Write the result



**You should trace other instructions through above datapath!**

Pramod Argade

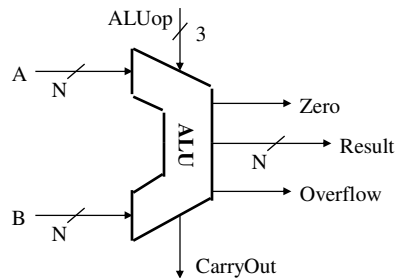
UCSD CSE 141, Spring 2005

Slide 7-10

## Control Logic

- A variety of control signals are needed
  - ALU Operation code
  - Mux controls
  - Register file write
  - Memory read/write
- How to generate various control signals?
  - From instruction
    - Opcode bits
    - Function code
  - From result of ALU operation
    - Zero bit for BEQ

## ALU Designed Before



- | <u>ALU Control Lines (ALUop)</u> | <u>Function</u>  |
|----------------------------------|------------------|
| – 000                            | And              |
| – 001                            | Or               |
| – 010                            | Add              |
| – 110                            | Subtract         |
| – 111                            | Set-on-less-than |

Note: Appendix B shows NOR operation also. We will not use it in further discussion.

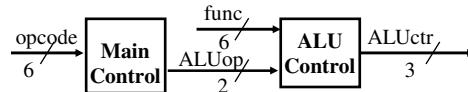
## ALU control bits

- Recall: 5-function ALU

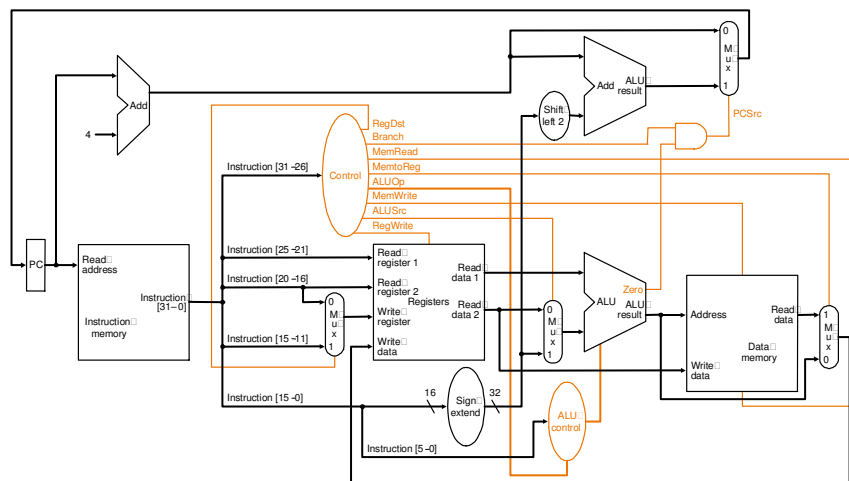
ALU control input	Function	Operations
000	And	and
001	Or	or
010	Add	add, lw, sw
110	Subtract	sub, beq
111	Slt	slt

- 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

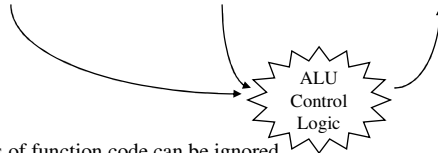


## Simple Datapath with the Control



## 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



Note: Upper 2 bits of function code can be ignored

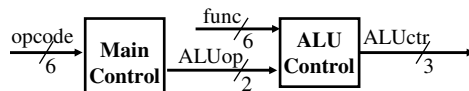
## Generating individual ALU signals

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

$$\text{ALUctr}_2 = \text{ALUop}_0 \mid (\text{ALUop}_1 \ \& \ \text{F}_1)$$

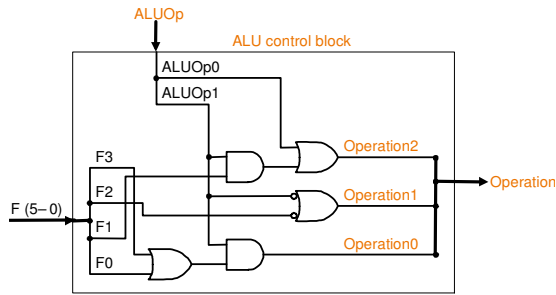
$$\text{ALUctr}_1 = \text{ALUop}_1 \mid \text{F}_2$$

$$\text{ALUctr}_0 = \text{ALUop}_1 \ \& \ (\text{F}_0 \mid \text{F}_3)$$

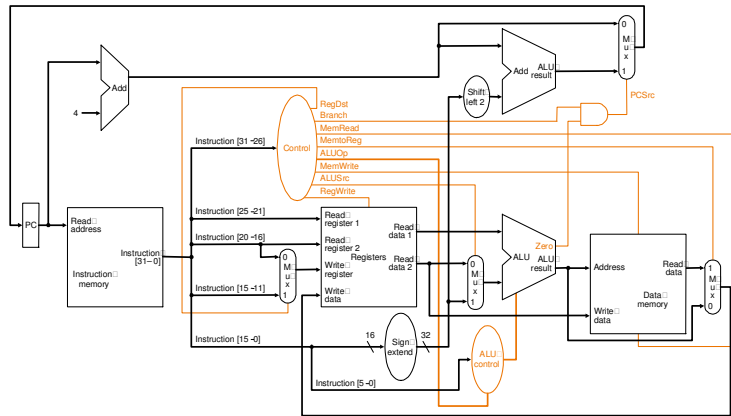


# ALU Control Logic

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

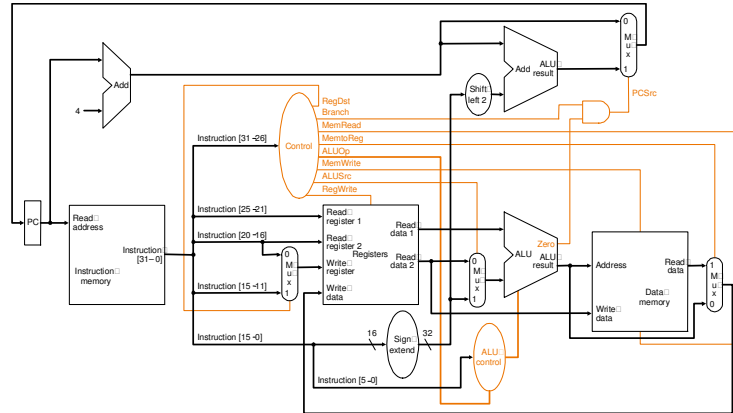


# Logic for Control signals



Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUop1	ALUop0
R-format								1	0
lw								0	0
sw								0	0
beq								0	1

# R Format Instruction Control Logic



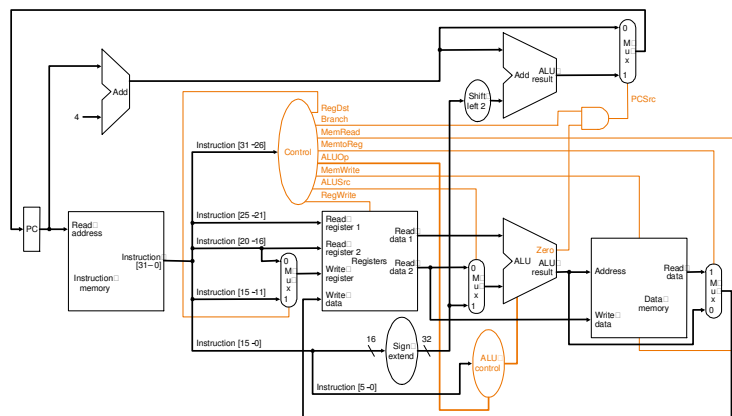
Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw								0	0
sw								0	0
beq								0	1

Pramod Argade

UCSD CSE 141, Spring 2005

Slide 7-19

# LW Instruction Control Logic



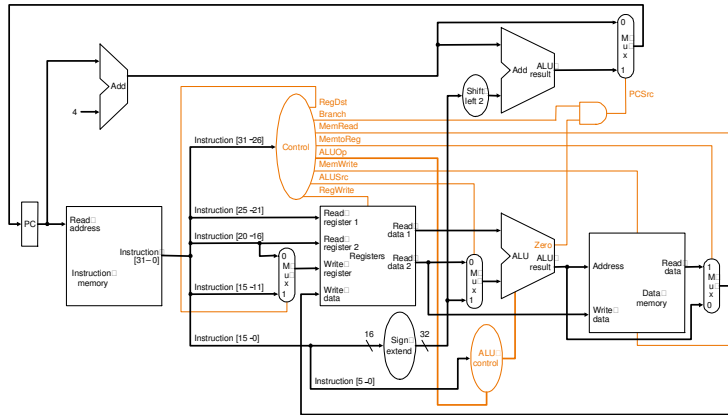
Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw								0	0
beq								0	1

Pramod Argade

UCSD CSE 141, Spring 2005

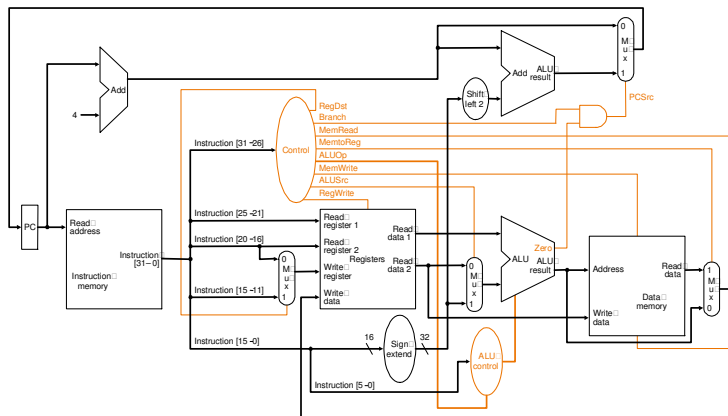
Slide 7-20

# SW Instruction Control Logic



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							1	0	1

# BEQ Instruction Control Logic



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	X	0	X	0	0	0	1	0	1

## Control Truth Table

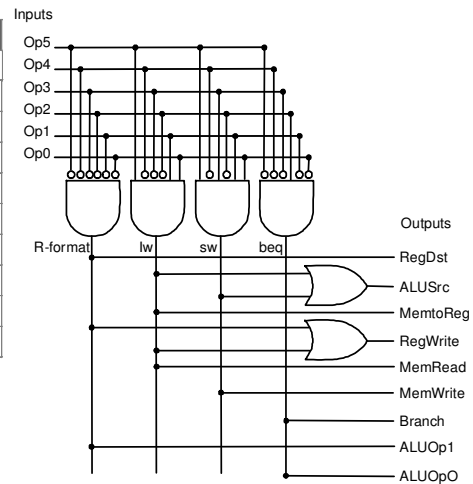
		R-format	lw	sw	beq
Opcode		000000	100011	101011	000100
Outputs	RegDst	1	0	x	x
	ALUSrc	0	1	1	0
	MemtoReg	0	1	x	x
	RegWrite	1	1	0	0
	MemRead	0	1	0	0
	MemWrite	0	0	1	0
	Branch	0	0	0	1
	ALUOp1	1	0	0	0
	ALUOp0	0	0	0	1

How to read this table? Example:

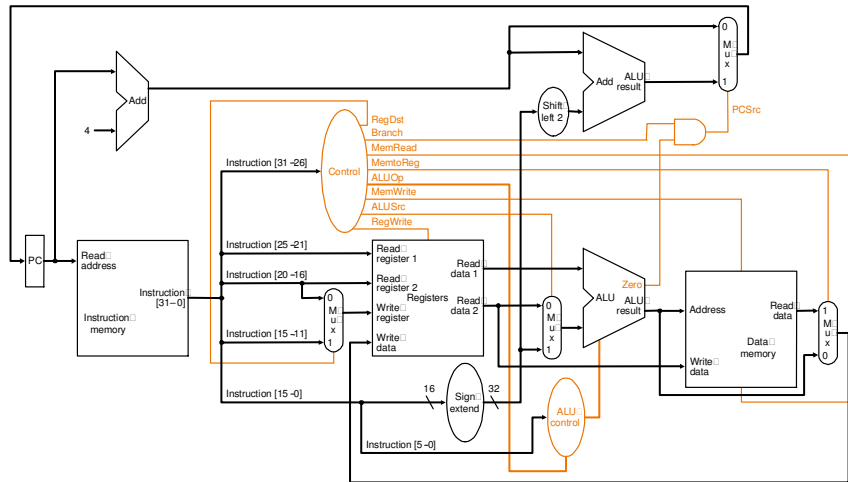
- $LW = Op5 \cdot Op4 \cdot Op3 \cdot Op2 \cdot Op1 \cdot Op0$
- $RegWrite = R\text{-format} + LW$

## Main Control Logic

Opcode	R-format	lw	sw	beq
000000	100011	101011	000100	
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



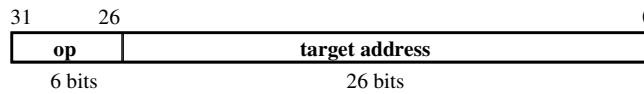
# Putting it all together



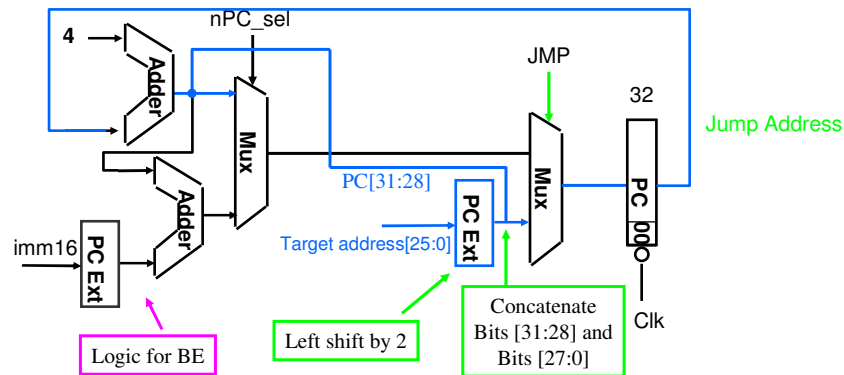
Note: PCSrc is generated by ANDing "Branch" and "Zero"

## CPU: Datapath for Jump

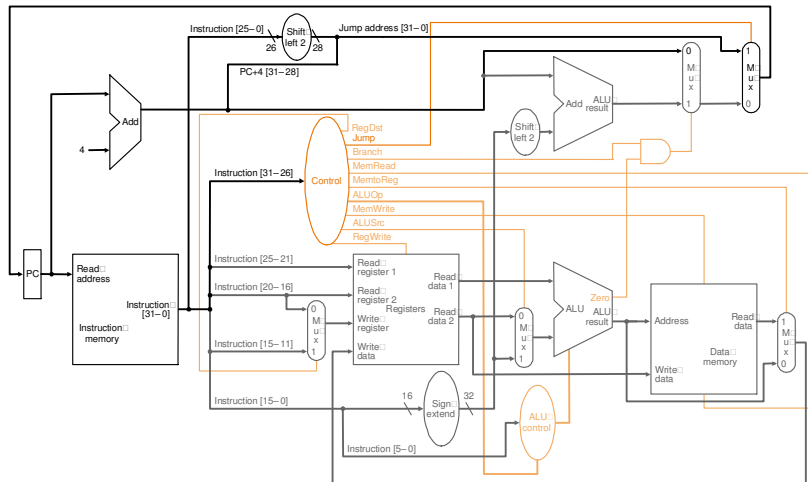
- J label



Instruction Address



## Jump Datapath and Control



Pramod Argade

UCSD CSE 141, Spring 2005

Slide 7-27

## Single Cycle Execution: Observations

- MIPS makes control easier
  - All the instructions are the same size (32-bits)
  - Source registers always in same place
  - Immediates same size, location
  - Operations always on registers/immediates
- Long cycle time:
  - Cycle time must be long enough for the load instruction:
    - PC's Clock -to-Q +
    - Instruction Memory Access Time +
    - Register File Access Time +
    - ALU Delay (address calculation) +
    - Data Memory Access Time +
    - Register File Setup Time +
    - Clock Skew
- Cycle time for load is much longer than needed for all other instructions

Pramod Argade

UCSD CSE 141, Spring 2005

Slide 7-28

## Single-Cycle CPU Summary

- Simple Design
  - Particularly the control logic
- CPU is just a collection of state and combinational logic
- Is our design efficient?
- Execution time = Instructions \* CPI \* Cycle time
- What did we sacrifice for simplicity?
  - **Cycle time**
- What is inefficient in our design?

## Announcements

- **Discussions Sections for 141:**
  - Fridays, 10:00 - 10:50 am, Peterson Hall 104 (Chris)
  - Fridays, 2:00 - 2:50 pm, Peterson Hall 104 (Leo)
- **Reading Assignment**
  - Chapter 5. Processor: Datapath and Control
  - Sec. 5.1 - 5.4
- **Homework 4: Due Mon., April 25<sup>th</sup> in class**
  - 5.2, 5.8, 5.9, 5.10, 5.13, 5.20, 5.22, 5.28
- **Quiz**
  - When:** Mon., April 25th, First 10 minutes of the class
  - Topic:** Single Cycle CPU, Chapter 5    **Need:** Paper, pen