

Computer Architecture

Hossein Asadi
Department of Computer Engineering
Sharif University of Technology
asadi@sharif.edu



Today's Topics

- · Control Logic for Single-Cycle CPU
 - ALU control
 - Processor control unit
- Implementing Unconditional Branch

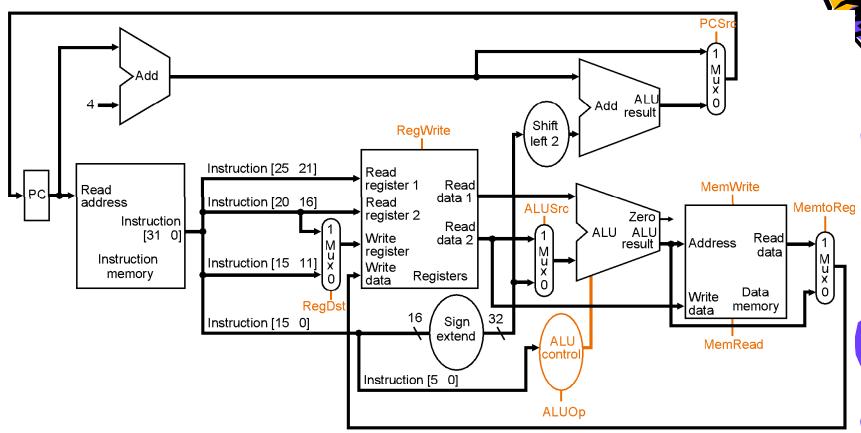


Copyright Notice

- Parts (text & figures) of this lecture adopted from:
 - Computer Organization & Design, The Hardware/Software Interface, 3rd Edition, by D. Patterson and J. Hennessey, Morgan Kaufmann publishing, 2005.
 - "Intro to Computer Architecture" handouts, by Prof. Hoe, CMU, Spring 2009.
 - "Computer Architecture & Engineering" handouts, by Prof. Kubiatowicz, UC Berkeley, Spring 2004.
 - "Intro to Computer Architecture" handouts, by Prof. Hoe, UWisc, Fall 2009.
 - "Computer Arch I" handouts, by Prof. Garzarán, UIUC, Spring 2009.

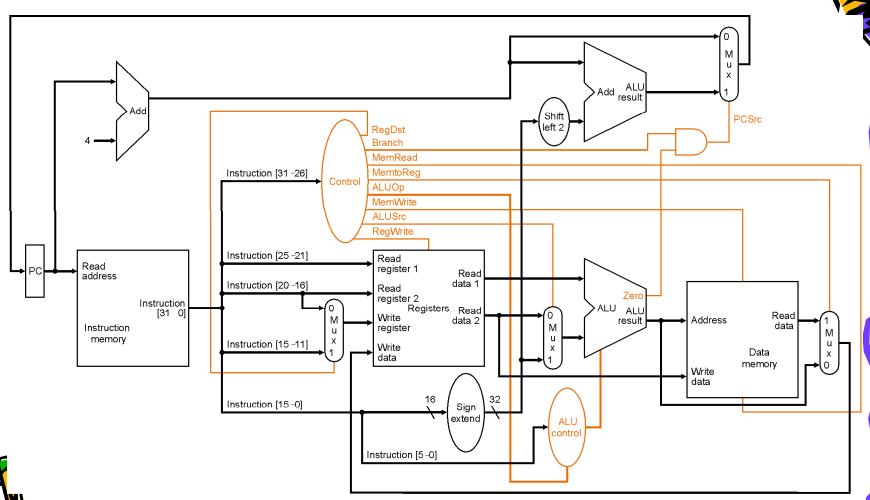


Datapath



We have everything except details for generating control signals

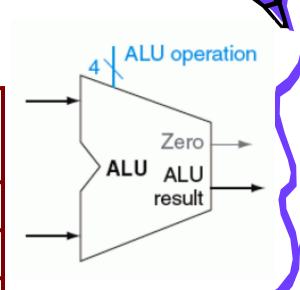
Adding Control Signals



ALU Control

- · ALU Control Lines
 - Four control lines

ALU Control Lines	Function
0000	AND
0001	OR
0010	add
0110	sub
0111	set on less than
1100	NOR





- · Load/Store
 - Memory address computed by addition
- R-Type Instructions
 - AND, OR, sub, add, set on less than
- Branch Equal
 - subtraction



· Question:

- A MIPS designer wants to design ALU controller. What signals should be used as inputs to this controller?

· Answer:

- 6-bit function code
 - F5, F4, ..., F0
- Signals to distinguish R-type, lw/sw, beq
 - Lets called it ALUop0 and ALUop1



- ALUop
 - Used to distinguish R-type, lw/sw, beq

ALUop	Instruction	ALU Operation
00	Load/Store	Add
01	Beq	Sub
10	R-type	Determined by funct. Code (F5~F0)



- ALU Control Inputs in terms of:
 - ALUop, funct field

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





- · Truth Table of ALU Control Inputs
 - 8 inputs
 - 4 outputs

AL	UOp			Funct				
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	FO	Operation
0	0	Х	Х	Х	Х	Х	Х	0010
X	1	Х	Х	Х	Х	Х	Х	0110
1	Х	Х	Х	0	0	0	0	0010
1	X	Х	Х	0	0	1	0	0110
1	X	Χ	Х	0	1	0	0	0000
1	X	Х	Х	0	1	0	1	0001
1	X	Χ	Х	1	0	1	0	0111



- How to Generate Control Logic?
 - Use Karnaugh Map
 - Or use intuitive logic design
 - Example for ALUcnt0
 - $-G1 = AND(ALUOp1, \sim F3, F2, \sim F1, F0)$
 - G2 = AND(ALUOp1, F3, ~F2, F1, ~F0)
 - ALUcnt0 = OR(G1, G2)



Designing Main Control Unit

- Steps
 - Identify fields of instructions
 - Identify control lines needed for datapath
 - Figure out how to generate control lines from fields of instructions



MIPS Instruction Formats



	31 26	21	10	11	0	
- add rd, rs, rt	op	rs	rt	rd	shamt	funct

- sub, and, or, slt 6 bits 5 bits 5 bits 5 bits 5 bits 6 bits

LOAD / STORE

- 26 21 **16** - lw rt, rs, imm
- immediate rt op rs - sw rt, rs, imm 6 bits 5 bits 5 bits 16 bits

BRANCH



beg rs, rt, imm

5 bits

16 bits

Designing Main Control Unit (cont.)

- Observations
 - Opcode always contained in bits 31:26
 - Op[5:0]
 - Two regs to be read always specified by rs & rt
 - rs in bits 25:21
 - rt in bits 20:16
 - · R-type, branch equal, store

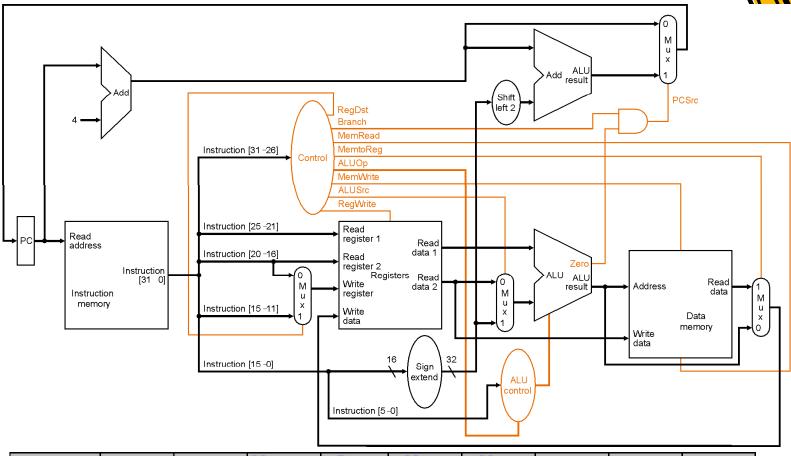


Designing Main Control Unit (cont.)

- Observations
 - Base reg for load & store always in bits 25:21 (rs)
 - 16-bit offset for branch equal, load, & store always in bits 15:0
 - Dest. reg specified either by rd or rt
 - R-type: in bits 15:11 (rd)
 - Load: in bits 20:16 (rt)

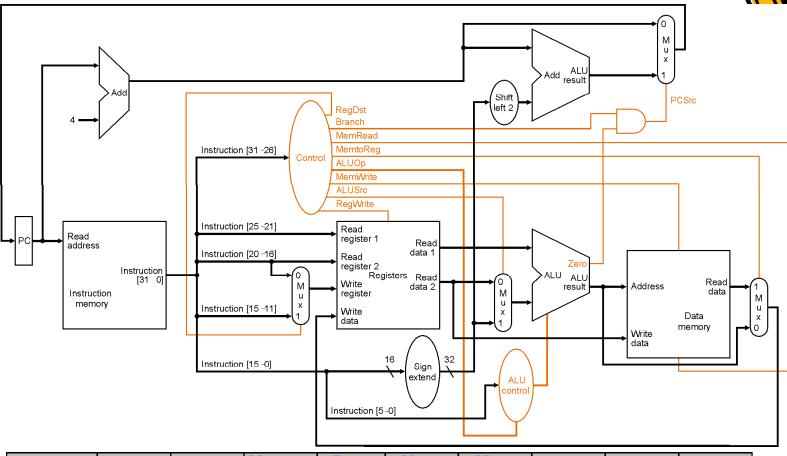


R-Format Instruction



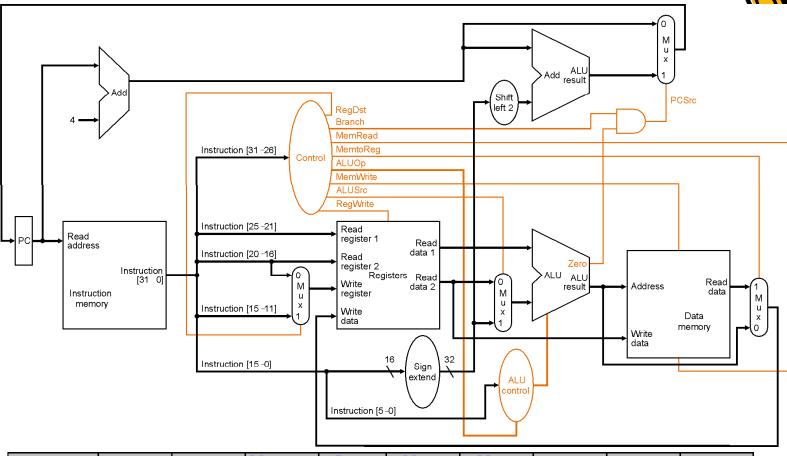
			Memto-	Reg	Mem	Mem			
Instruction	RegDst	ALUSrc	Reg	Write	Read	Write	Branch	ALUOp1	ALUp0
R-format								1	0
lw								0	0
SW								0	0
beq								0	1

Iw Control



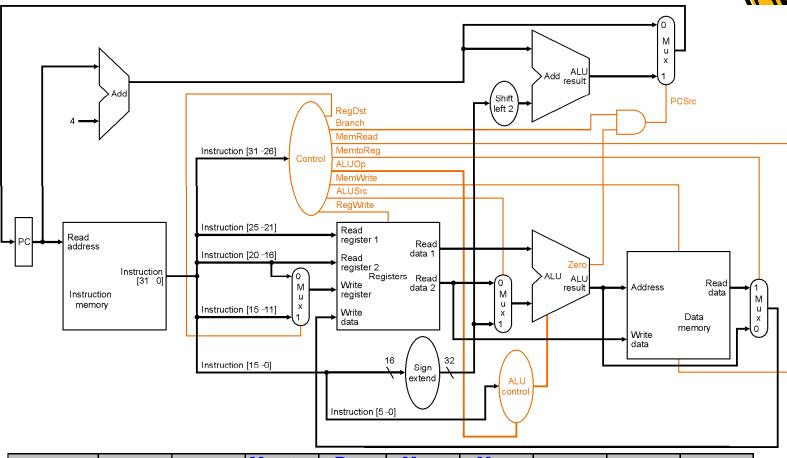
Instruction	RegDst		Memto- Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUp0
R-format	1	0	0	1	0	0	0	1	0
lw								0	0
SW								0	0
beq								0	1

sw Control



			-						
			Memto-	•	Mem	Mem			
Instruction	RegDst	ALUSTC	Reg	Write	Read	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								0	0
beq								0	1

beq Control

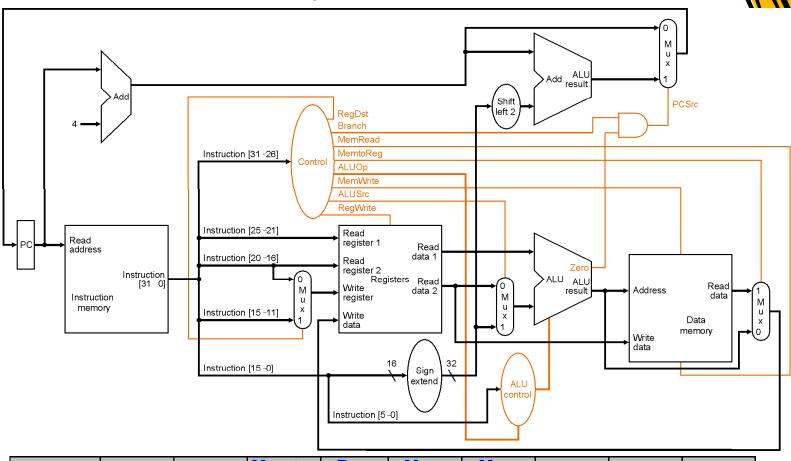


			-						
Instruction	RegDst		Memto- Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq								0	1

Lecture 6

Sharif University of Technology, Spring 2021

beq Control



Instruction	RegDst		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

Lecture 6

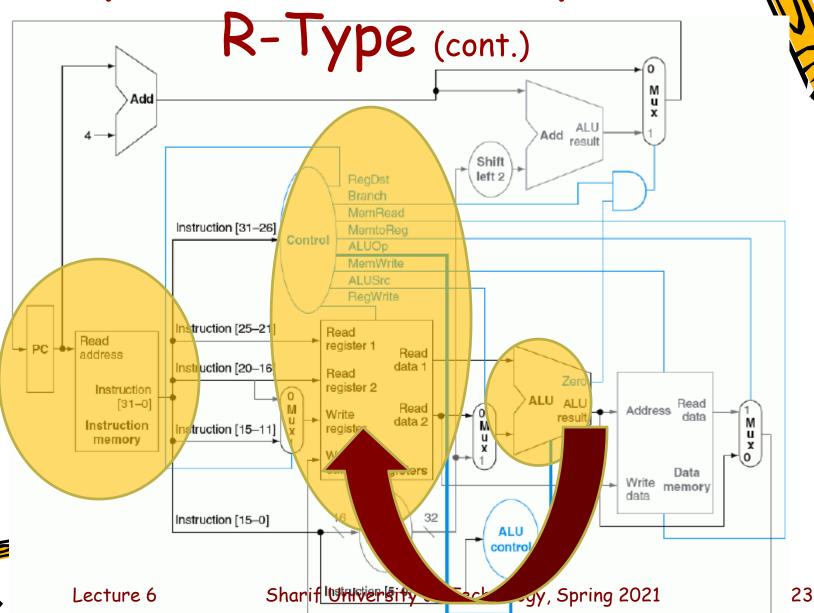
Sharif University of Technology, Spring 2021

Operation of Datapath: R-Type

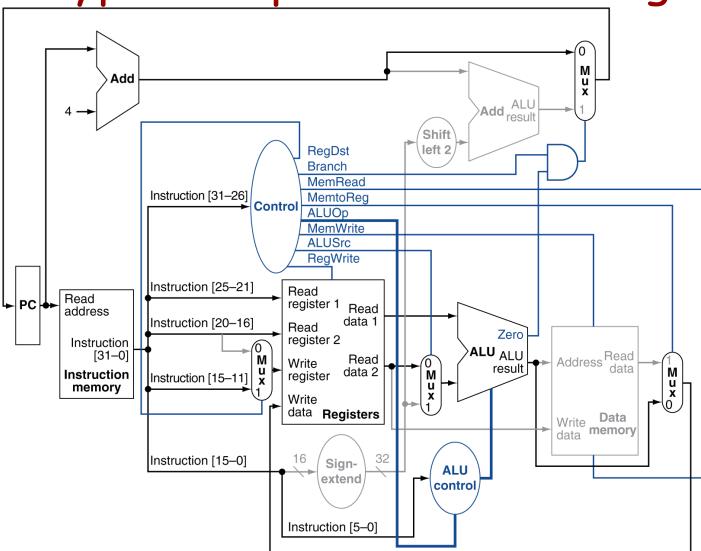
- Step 1:
 - Instruction fetched
 - PC incremented
- Step 2:
 - Two regs read from GPR
 - Main CU computes setting of control lines
- Step 3:
 - ALU control determined by funct. Code
 - Then, ALU operates on data read from GPR
- Step 4:
 - Results from ALU written into RF using bits 15:11



Operation of Datapath:



R-Type Datapath + Control Signal



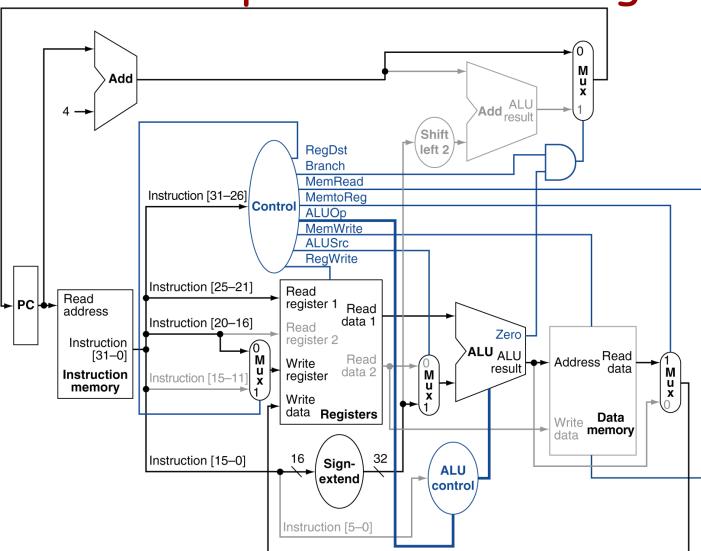


Operation of Datapath: Load

- Step 1:
 - Instruction fetched
 - PC incremented
- Step 2:
 - A reg read from GPR (e.g. \$t1)
 - CU computes setting of control lines
- Step 3:
 - ALU computes target memory address
 - Based on \$11 and sign-extended value in bits 15:0
- Step 4:
 - 32-bit data read from Memory based on calculated addr.
- Step 5:
 - Data written into GPR (destination reg: bits 20:16)



Operation of Datapath: Load (cont.) Add Add Shift left 2 MemRead Instruction [31-26] MemtoReg Control ALUOp **ALUSrc** RegWrite Instruction [25-21 Read Read register 1 Read Instruction [20-16] data 1 Zero register 2 Instruction [31-0]Read Address Write М Instruction Instruction [15-11] register memory Х Data memory Instruction [15-0] ALU exte control Sharif University of Tecm. Lecture 6 26 Load Datapath + Control Signals





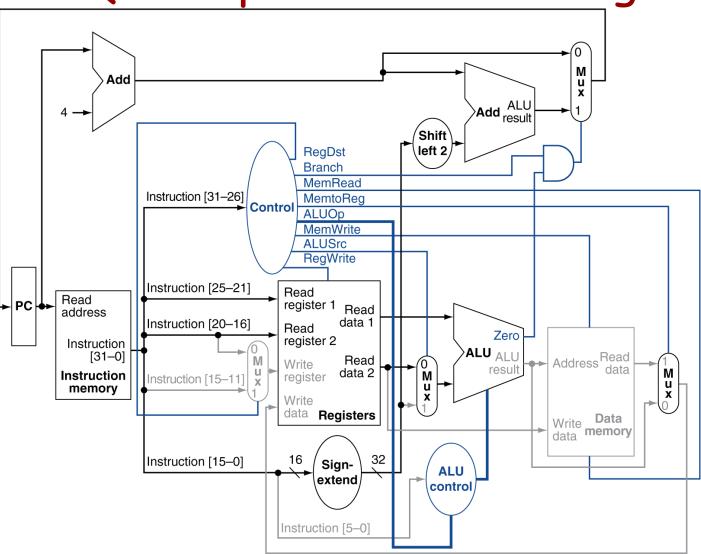
Operation of Datapath: Branch Equal

- Step 1:
 - Instruction fetched
 - PC incremented
- Step 2:
 - Two regs read from GPR (e.g., \$t0, \$t1)
 - CU computes setting of control lines
- Step 3:
 - ALU performs subtract on ALU inputs (\$t0-\$t1)
 - Branch target address computed: PC+4+(addr<<2)
- Step 4:
 - Zero results from ALU used to update PC



Operation of Datapath: Branch Equal (cont.) Add Add Shift left 2 Branch Instruction [31-26] MemtoReg Control **ALUO**p ALUSrc Instruction [25–21] Read Read PC reaister 1 Read address data 1 Instruction [20-16 Read Zero register 2 Instruction [31-0]Read Read Address data data 2 M Instruction struction [15-11] register memory Х х Write Registers Data Write memory 32 Instruction [15-0] ALU extend control Sharif University of Technology, Spring 2021 29 Lecture 6

BEQ Datapath + Control Signals





Implementing Unconditional Branch

Jump Instruction Formation

31 26	25 0
000010	address
6 bits	26 bits

- Target Address
 - $Addr[1:0] = 00_{two}$
 - Addr[27:2] = IR[25:0]
 - · Immediate field in instruction
 - $Addr[31:28] = PC_{new}[31:28]$
 - $PC_{new} = PC + 4$



Jump Datapath Jump address [31-0] Instruction [25-0] Shift left 2 PC + 4 [31-28] М Add X ALU Add result ReaDst Shift left 2 Jump Branch MomPeau Instruction [31-26] MemtoReg Control **ALUOp** MemWrite **ALUSrc** RegWrite Instruction [25-21] Read register 1 Read data 1 Instruction [20-16] Read Zero register 2 Instruction ALU [31-0] ALU Read Read Address Write **►**/0 result data 2 Instruction M Instruction [15-11] register u memory Х Write Registers data Data Write memory data Instruction [15-0] 16 32 Sign ALU extend control Sharif University of Technology, Spring 2021



Read

address

Practice

- · Question:
 - Why (PC+4) is used instead of PC to provide upper four bits of new instruction address (Addr[31:28])?



Backup

