Computer Architecture: MIPS Control Unit Design

Hossein Asadi (asadi@sharif.edu)
Department of Computer Engineering
Sharif University of Technology
Spring 2024

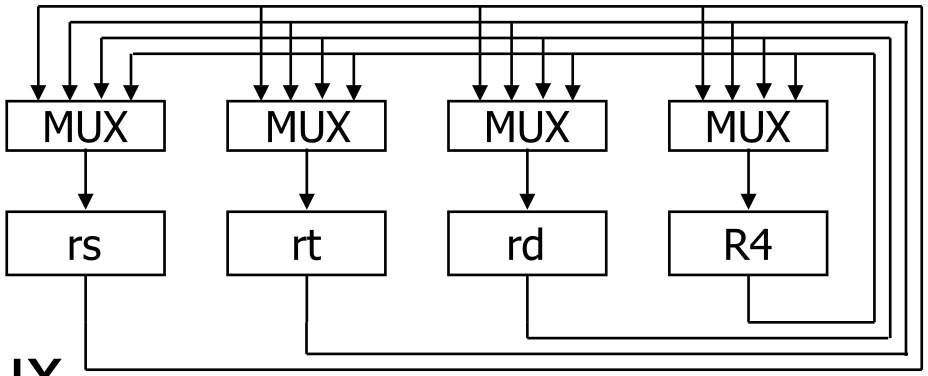
Copyright Notice

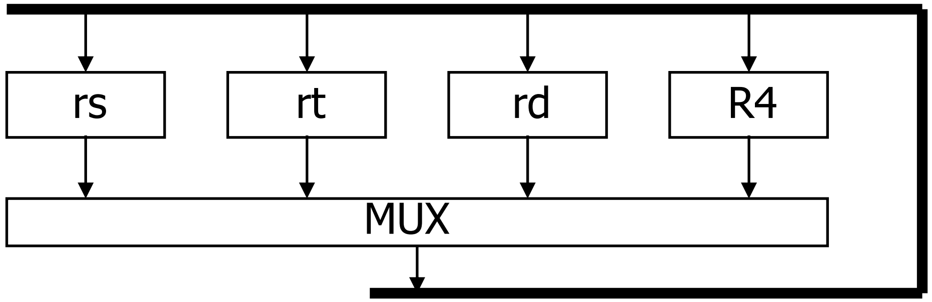
- Some Parts (text & figures) of this Lecture adopted from following:
  - Computer Organization & Design, The Hardware/Software Interface, 3rd Edition, by D. Patterson and J. Hennessey, MK publishing, 2005.
  - “Intro to Computer Architecture” handouts, by Prof. Hoe, CMU, Spring 2009.
  - “Computer Architecture & Engineering” handouts, by Prof. Kubiatowicz, UC Berkeley, Spring 2004.
  - “Intro to Computer Architecture” handouts, by Prof. Hoe, UWisc, Spring 2021.
  - “Computer Arch I” handouts, by Prof. Garzarán, UIUC, Spring 2009.
  - “Intro to Computer Organization” handouts, by Prof. Mahlke & Prof. Narayanasamy, Winter 2008.

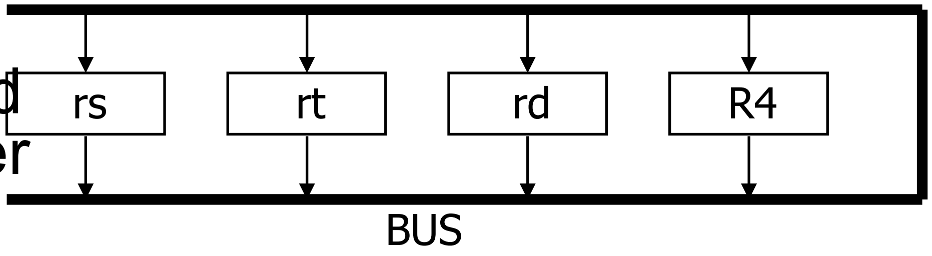
Quick Reminder from Previous Lecture

Register Interconnects

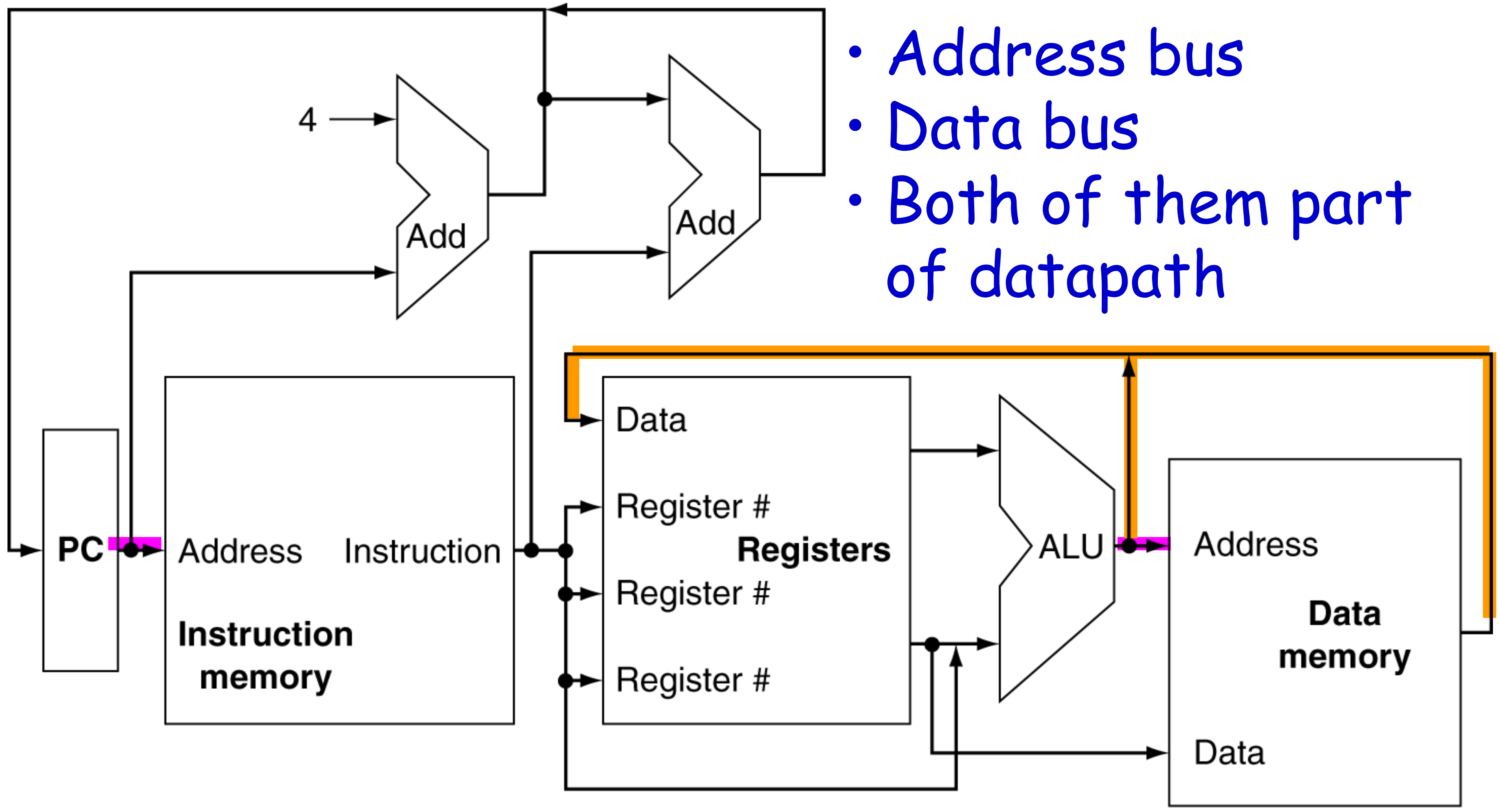
- Point-to-Point Connection
  - Dedicated wires
  - Muxes on inputs of each register
- Common Input from mux
  - Load enables for each register
  - Control signals for multiplexer
- Common Bus with Output Enables
  - Output enables and load enables for each register





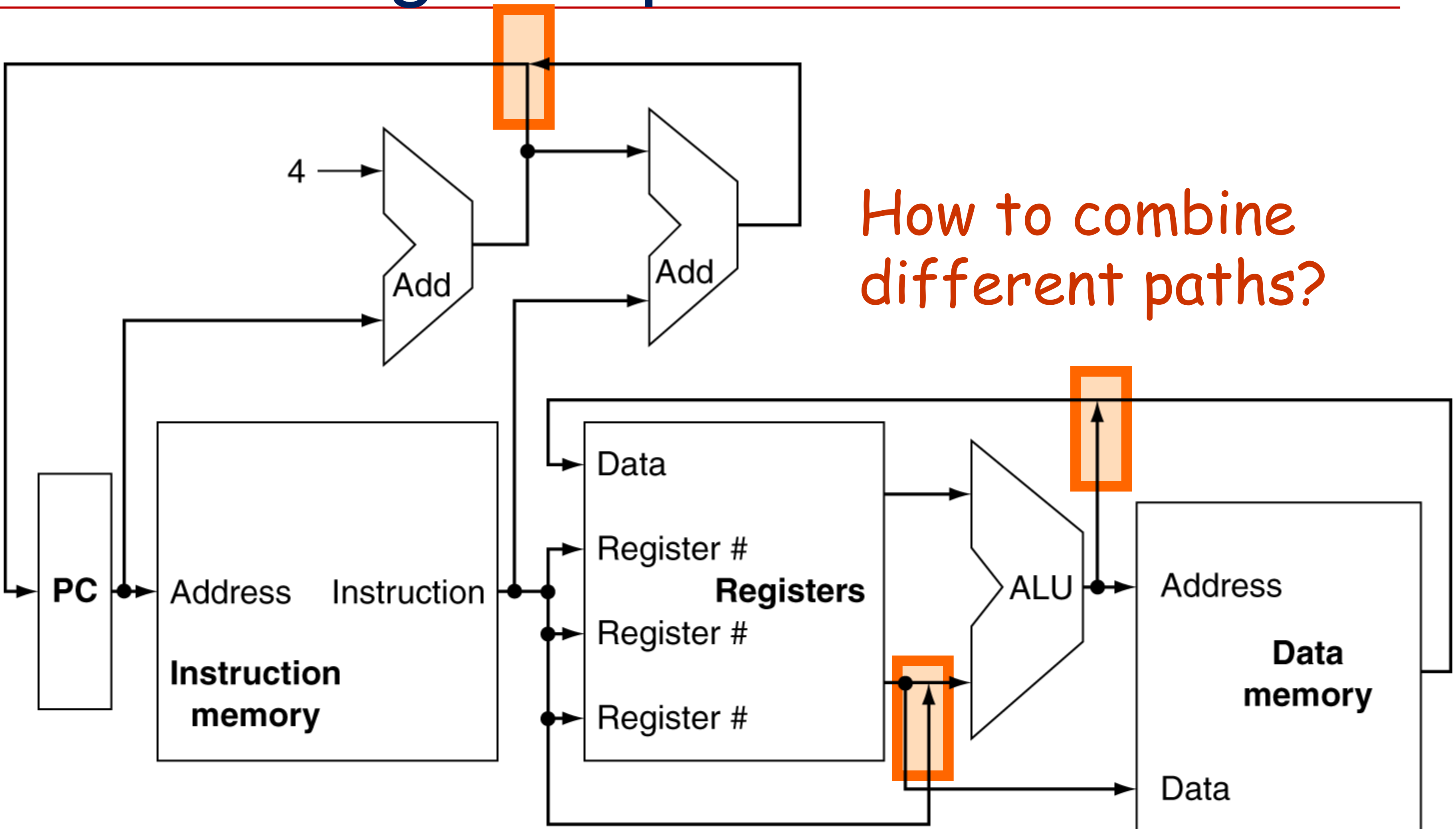


Address Bus & Data Bus



- Address bus
- Data bus
- Both of them part of datapath

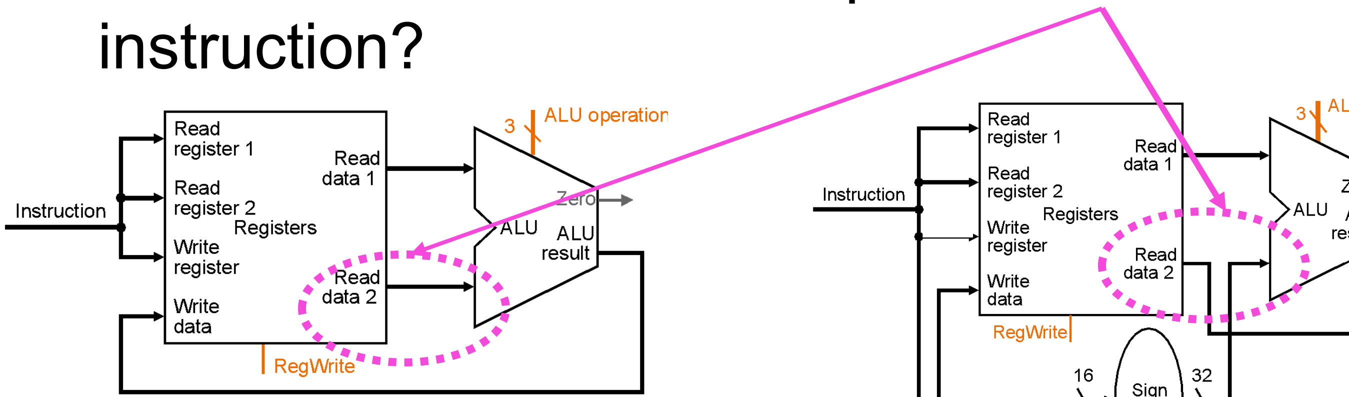
Combining Datapaths

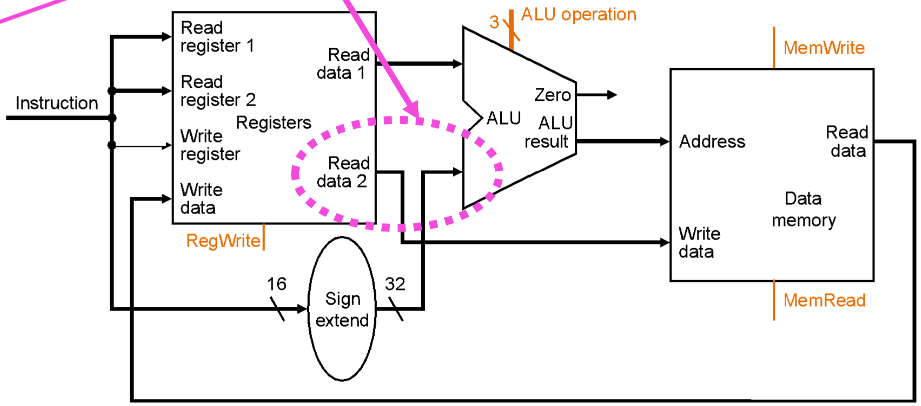


How to combine different paths?

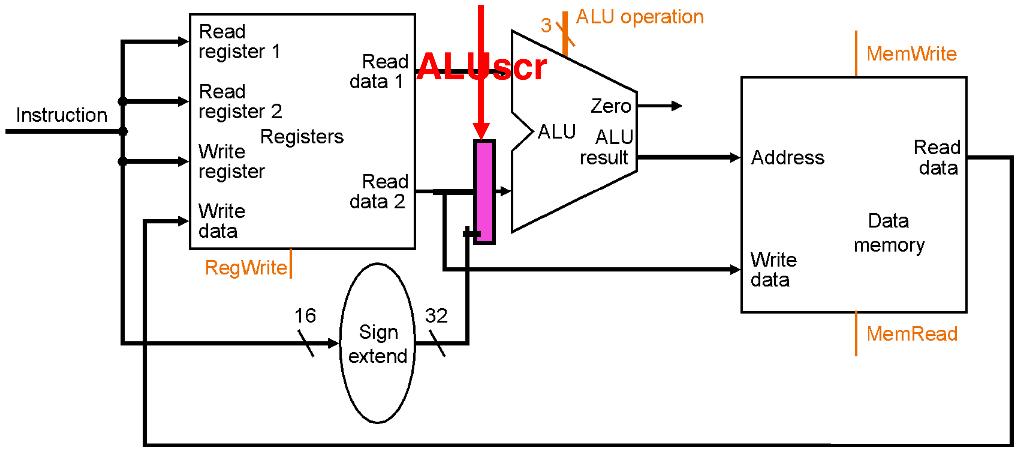
Combining Datapaths (cont.)

- Question:
  - How to have different datapaths for different instruction?

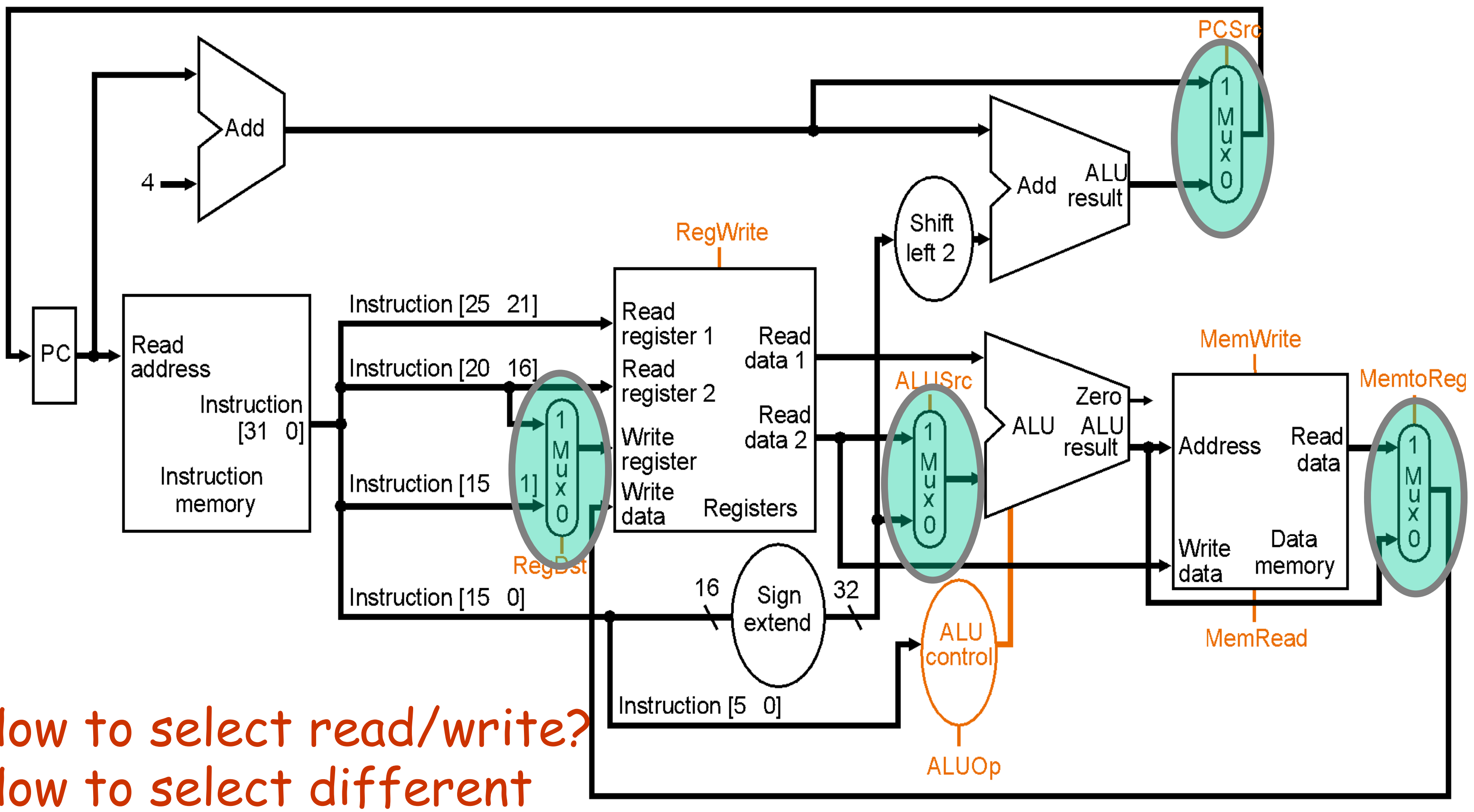




- Use a multiplexer

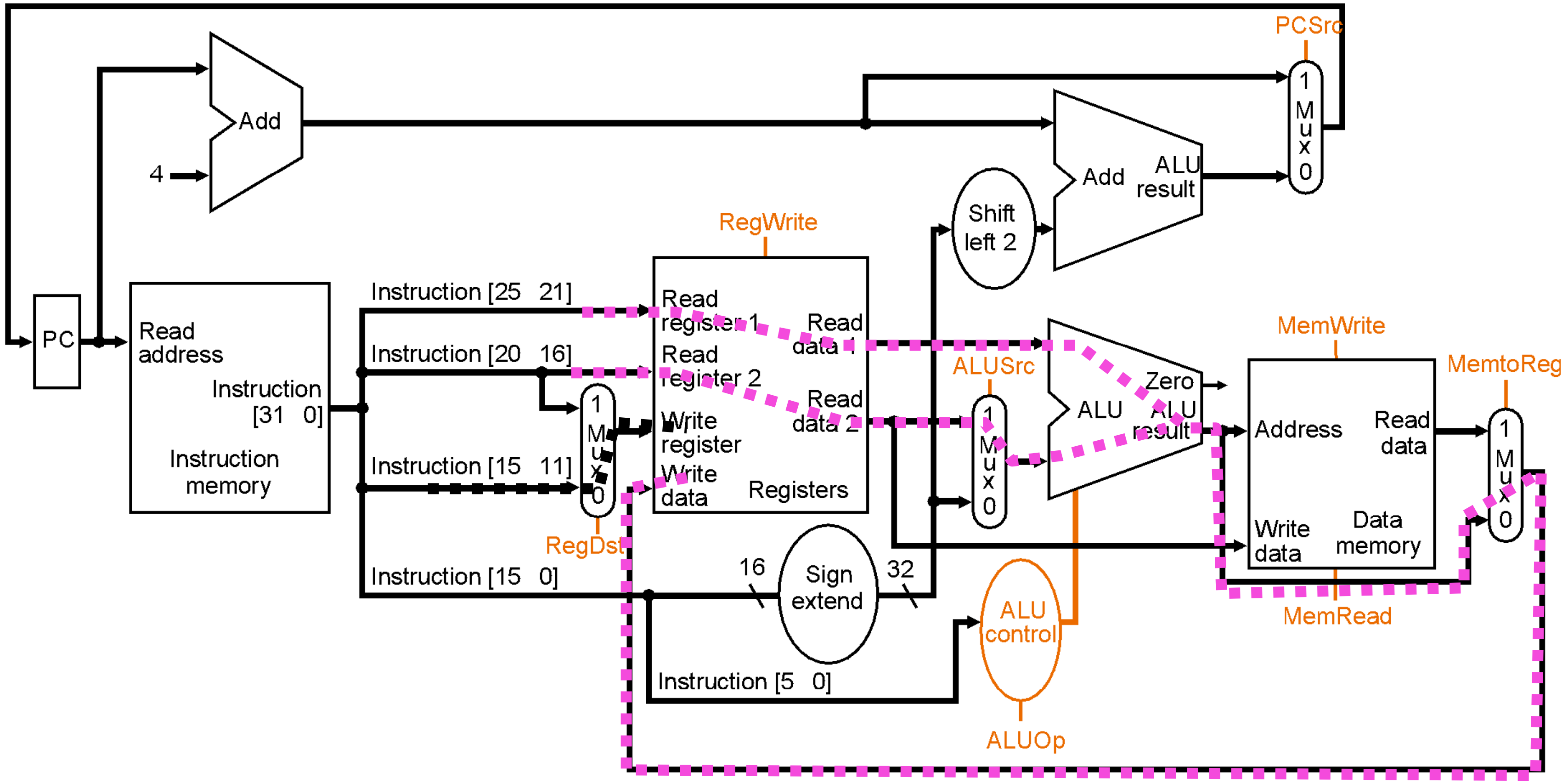


All Together: Single Cycle Datapath



How to select read/write?
How to select different datapaths?

R-Format Datapath (e.g. *add*)



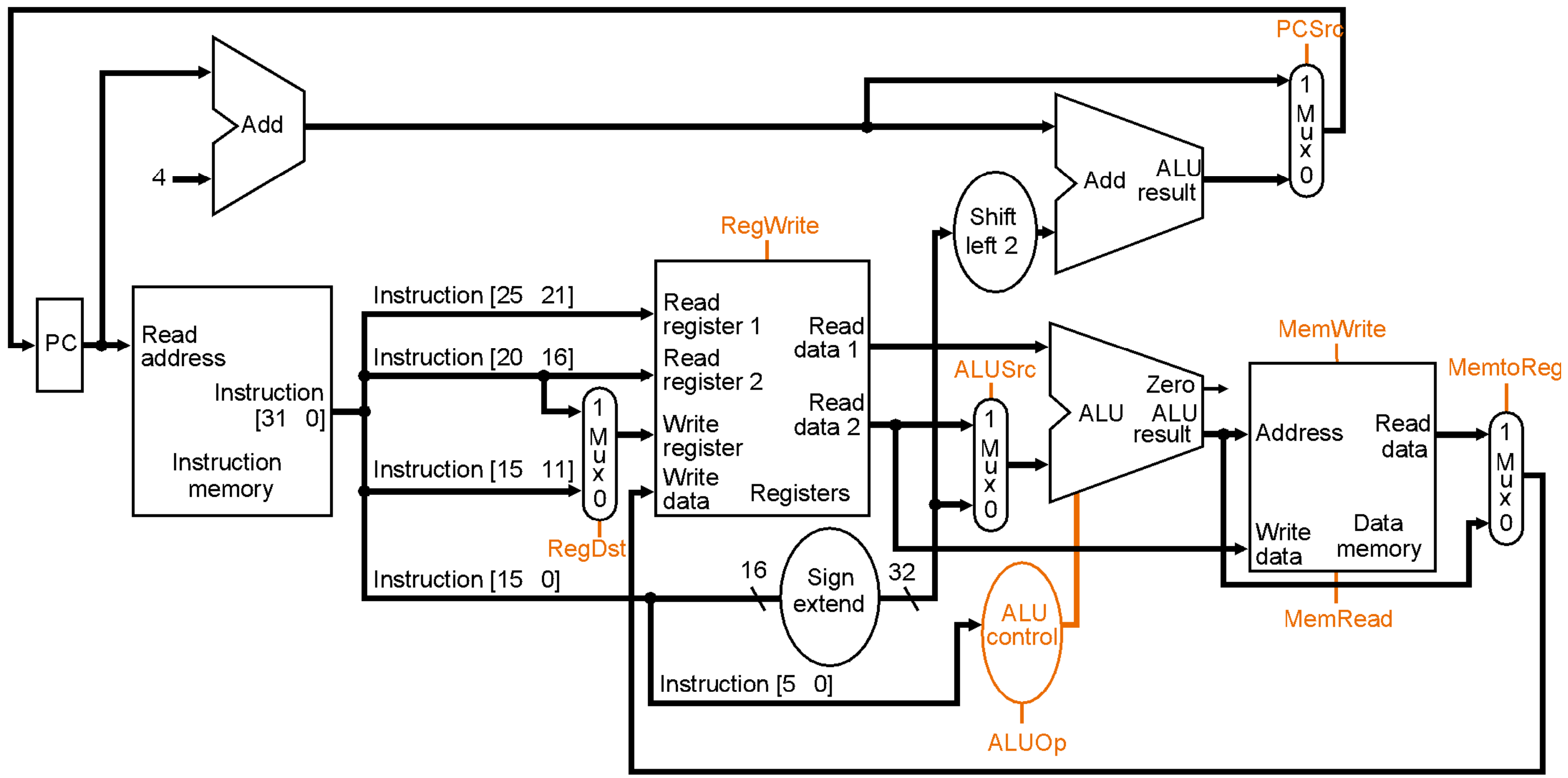
Need ALUsrc=1, ALUop="add", MemWrite=0, MemToReg=0,
RegDst = 0, RegWrite=1 and PCsrc=1.

Our Lectur Today

Topics Covered Today

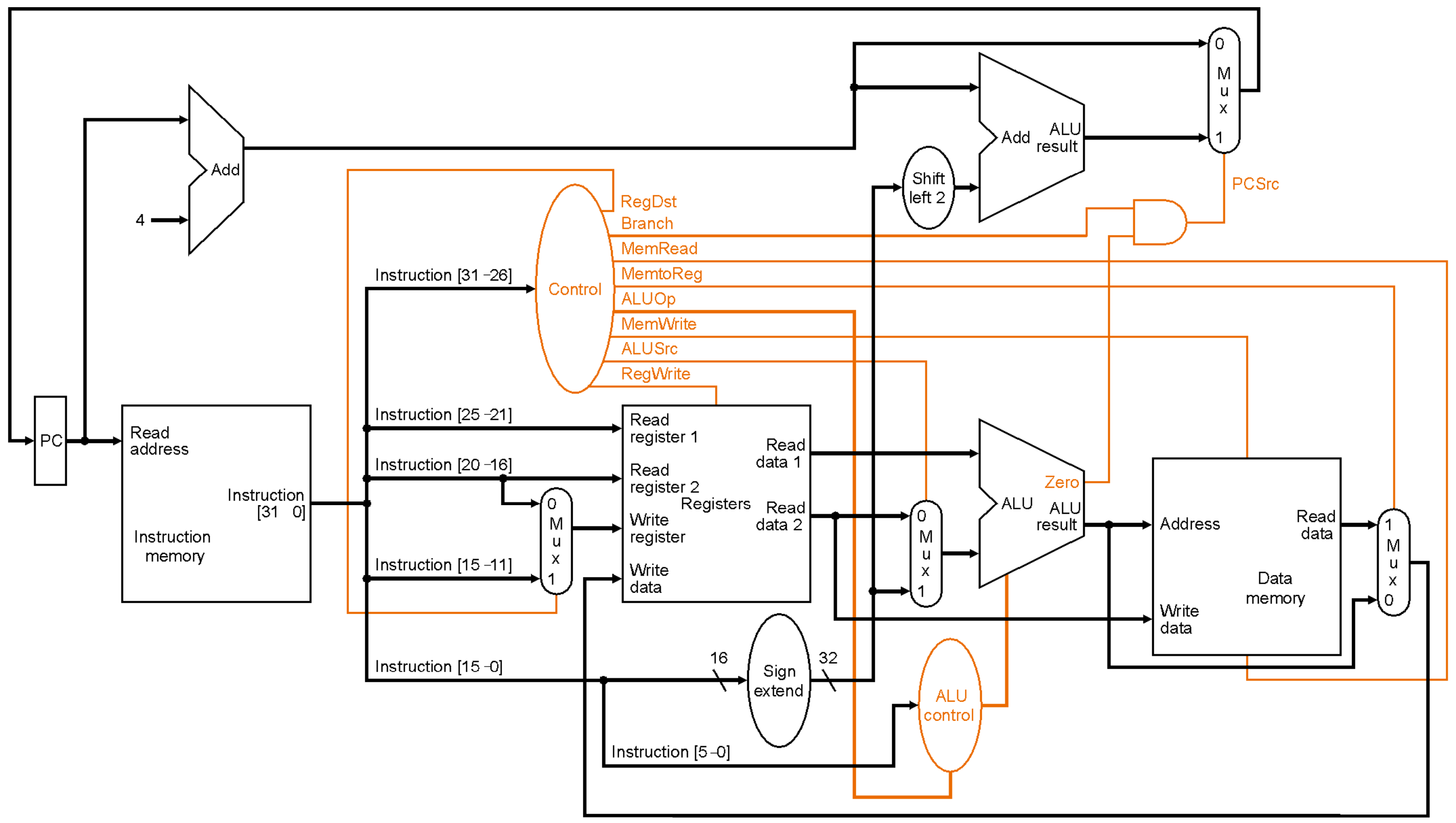
- MIPS Control Unit Design
  - Main control unit
  - ALU control signals

MIPS Single Cycle Datapath



We have everything except details for generating control signals

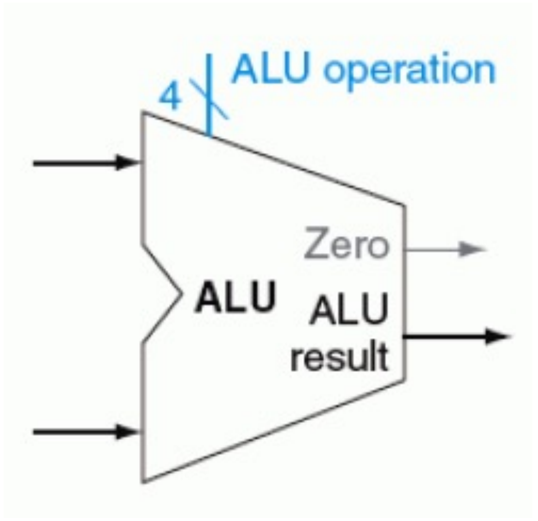
Adding Control Signals



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



ALU Control (cont.)

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

ALU Control (cont.)

- 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

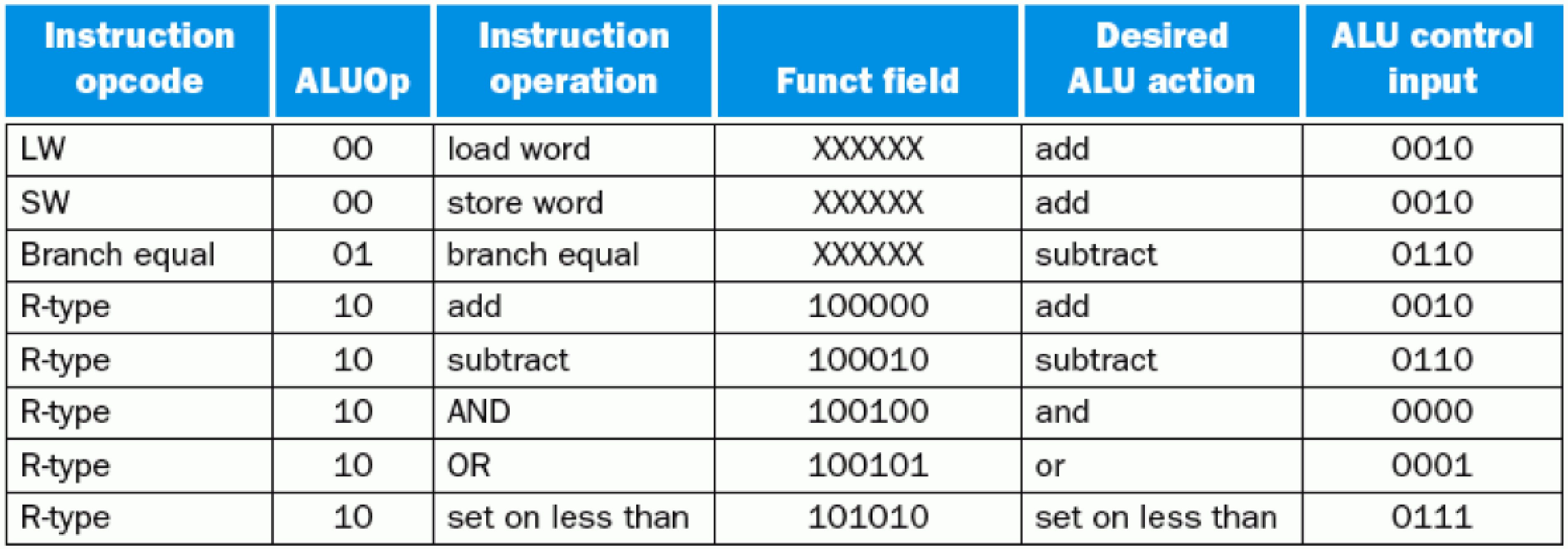
ALU Control (cont.)

- 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 (cont.)

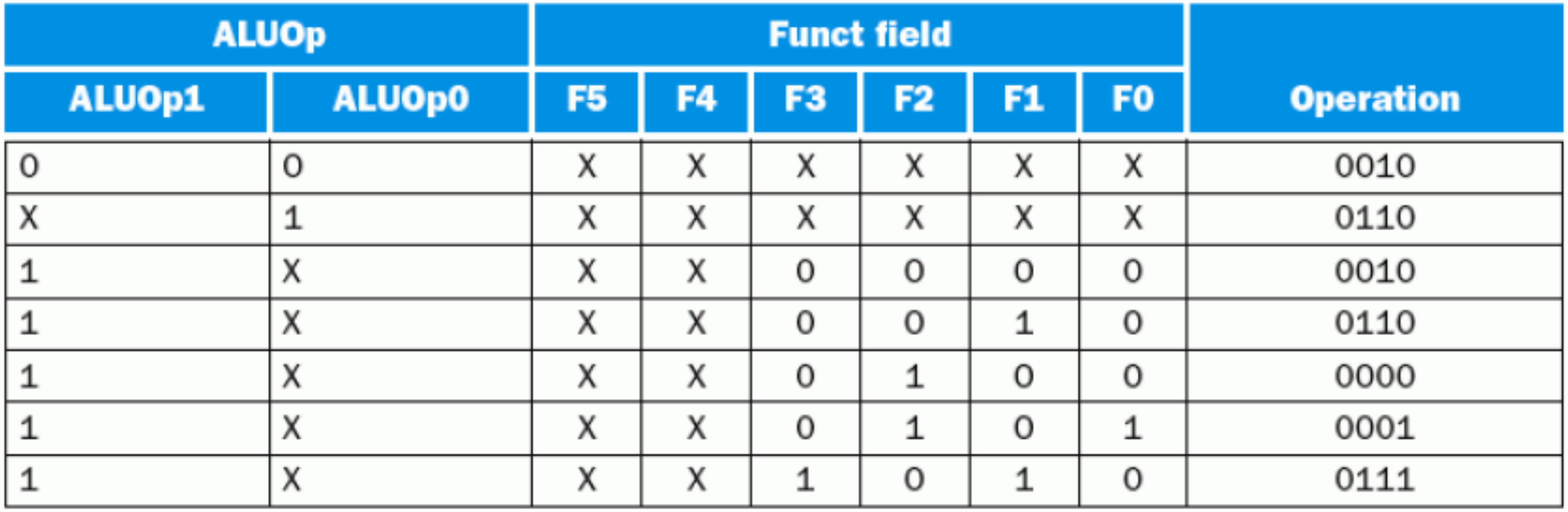
- ALU Control Inputs in terms of:
  - ALUop, funct field



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

ALU Control (cont.)

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



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

ALU Control (cont.)

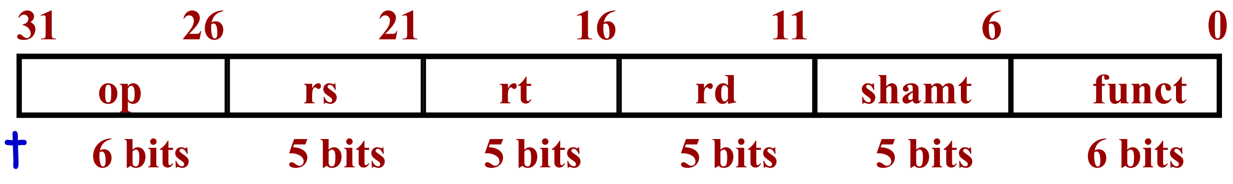
- How to Generate Control Logic?
  - Use Karnaugh Map
  - Or use intuitive logic design
    - Example for ALUcnt0
      - G1 = AND(ALUOp1, ~F3, F2, ~F1, F0)
      - G2 = AND(ALUOp1, F3, ~F2, F1, ~F0)
      - ALUcnt0 = OR(G1, G2)

Designing Main Control Unit

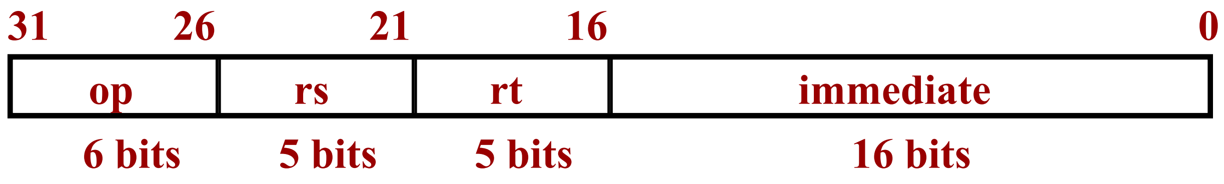
- Steps
  - Identify fields of instructions
  - Identify control lines needed for datapath
  - Figure out how to generate control lines from fields of instructions

MIPS Instruction Formats

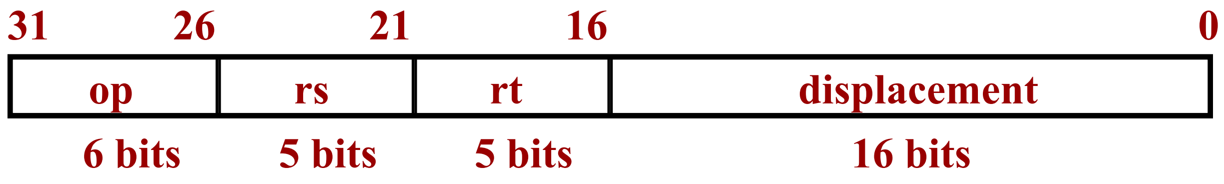
- R-TYPE
  - *add rd, rs, rt*
  - sub, and, or, slt



- LOAD / STORE
  - lw rt, rs, imm
  - sw rt, rs, imm



- BRANCH
  - beq rs, rt, imm



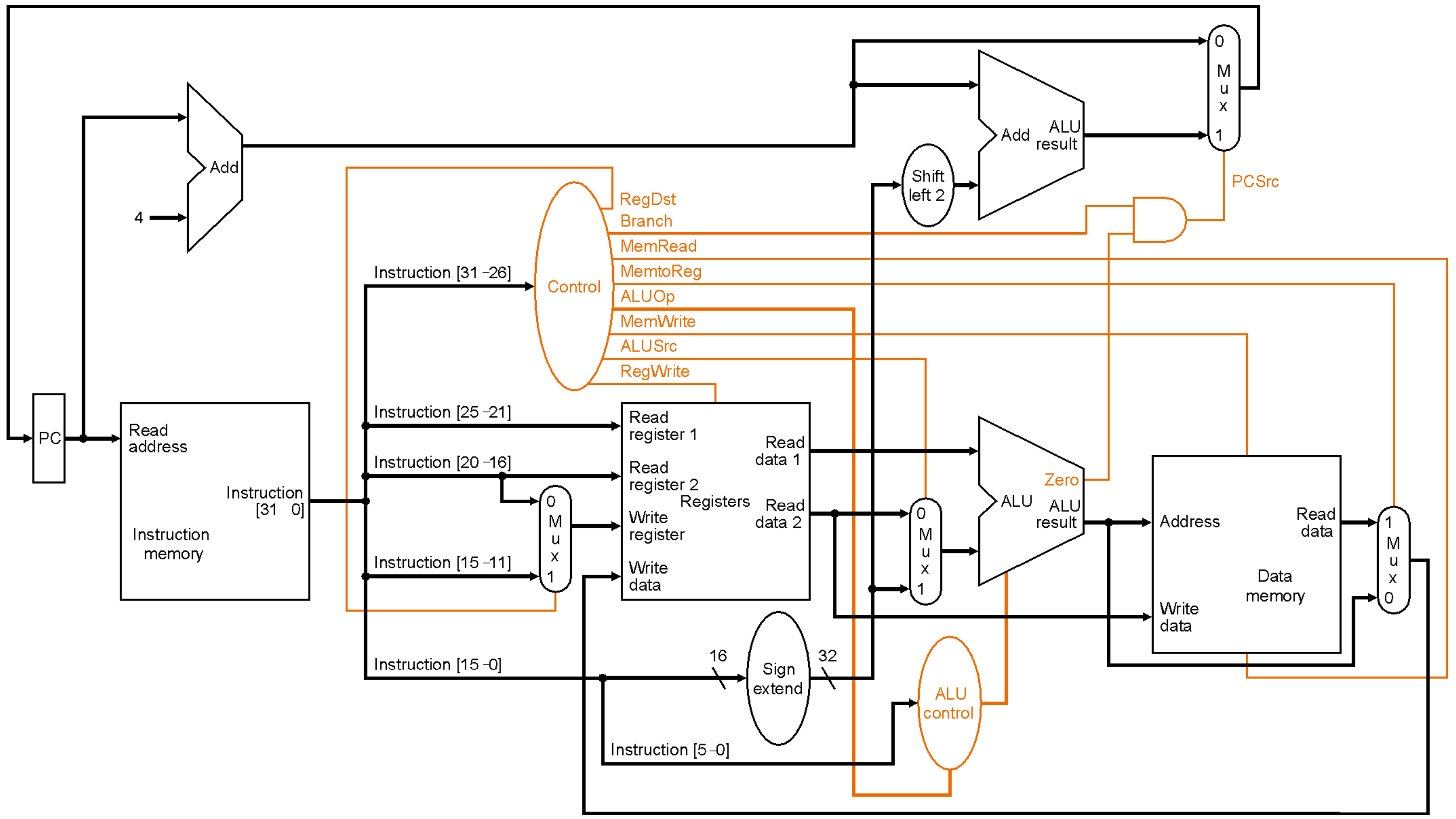
Designing Main Control Unit (cont.)

- 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.)

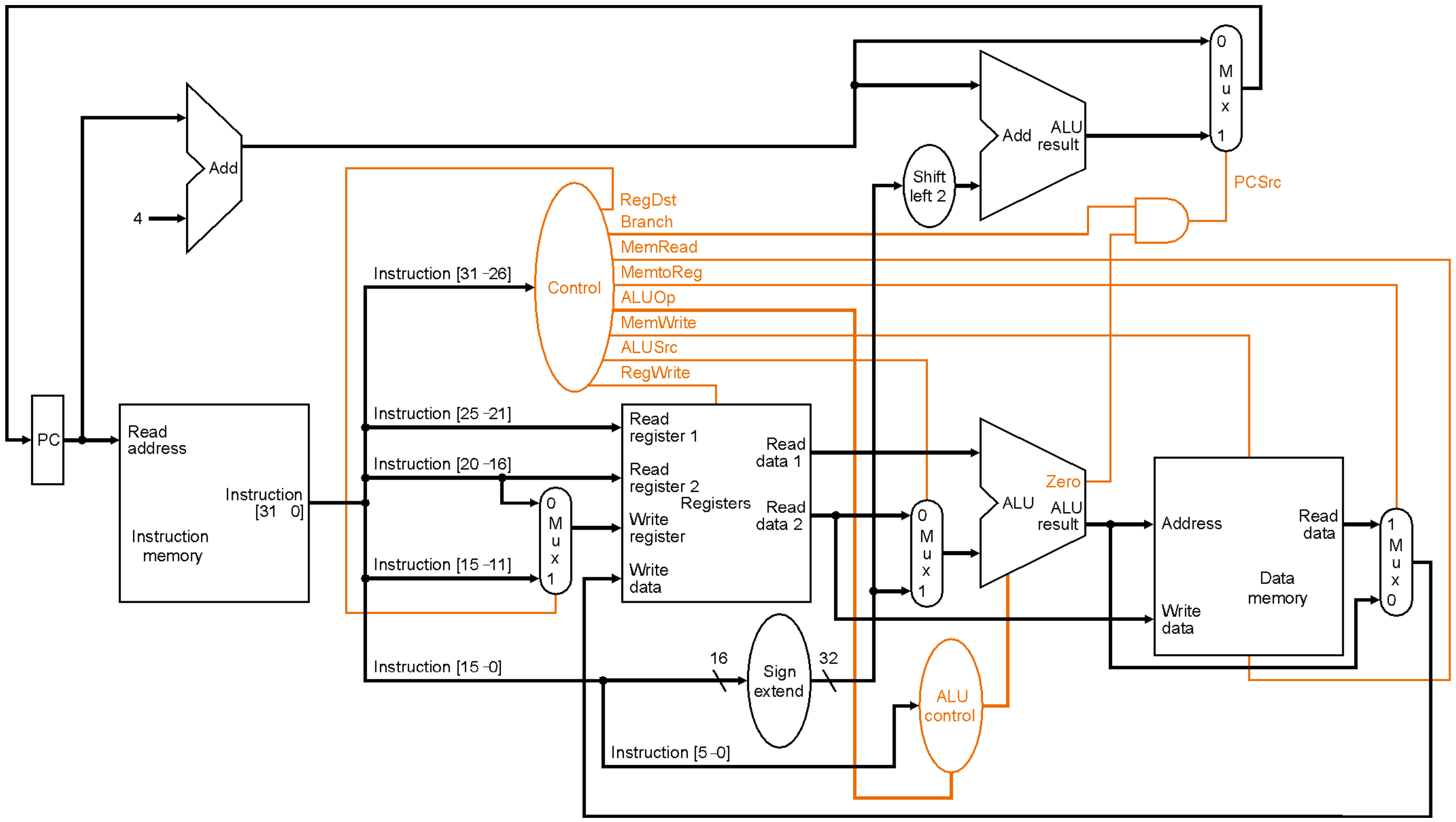
- Observations
  - Base reg for load & store always in bits 25:21 (rs)
  - 16-bit offset for branch equal, load, & store always in bits 15:0
  - Dest. reg specified either by rd or rt
    - R-type : in bits 15:11 (rd)
    - Load: in bits 20:16 (rt)

R-Format Instruction



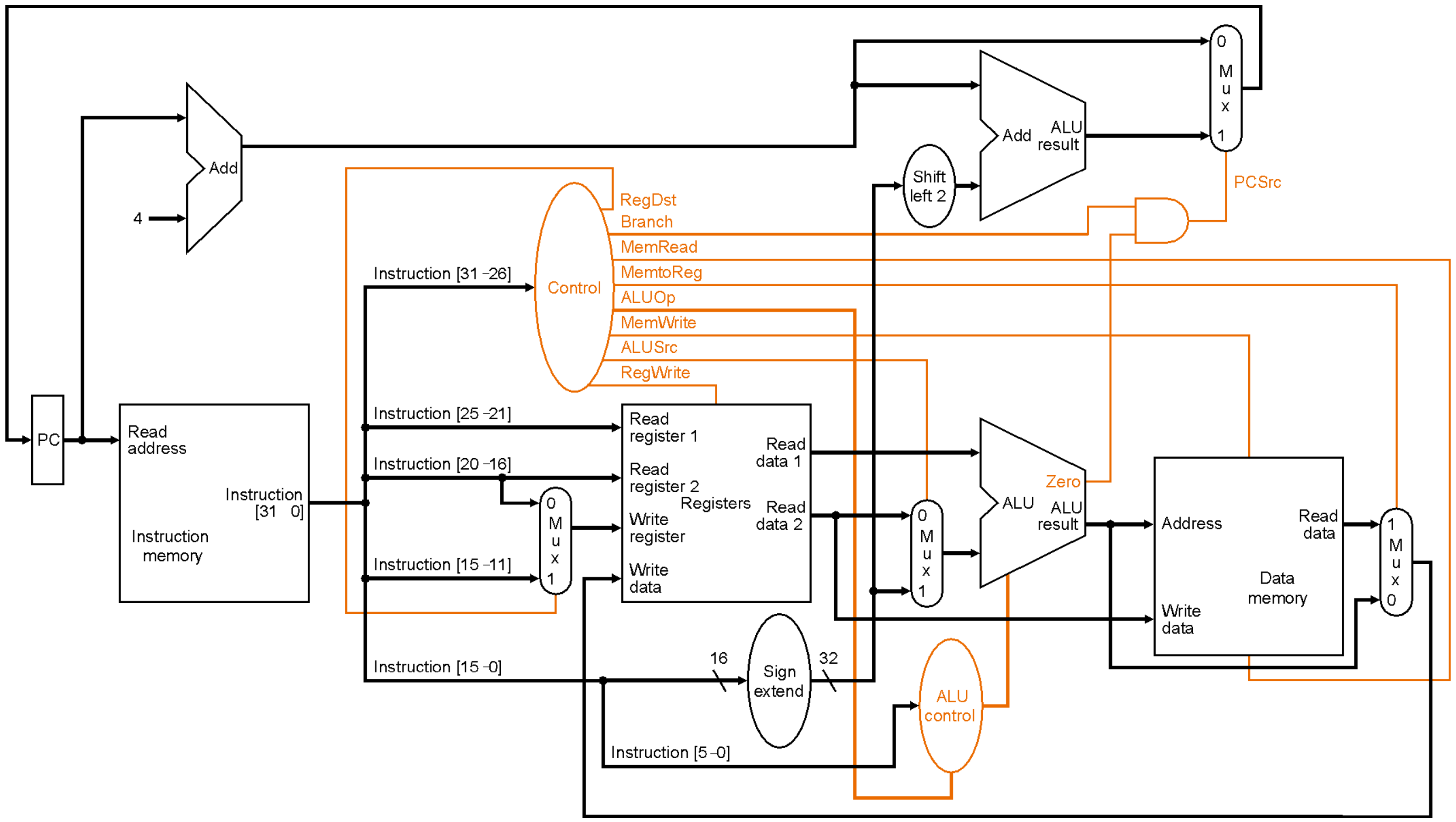
Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUp0
R-format								1	0
lw								0	0
sw								0	0
beq								0	1

LW Control



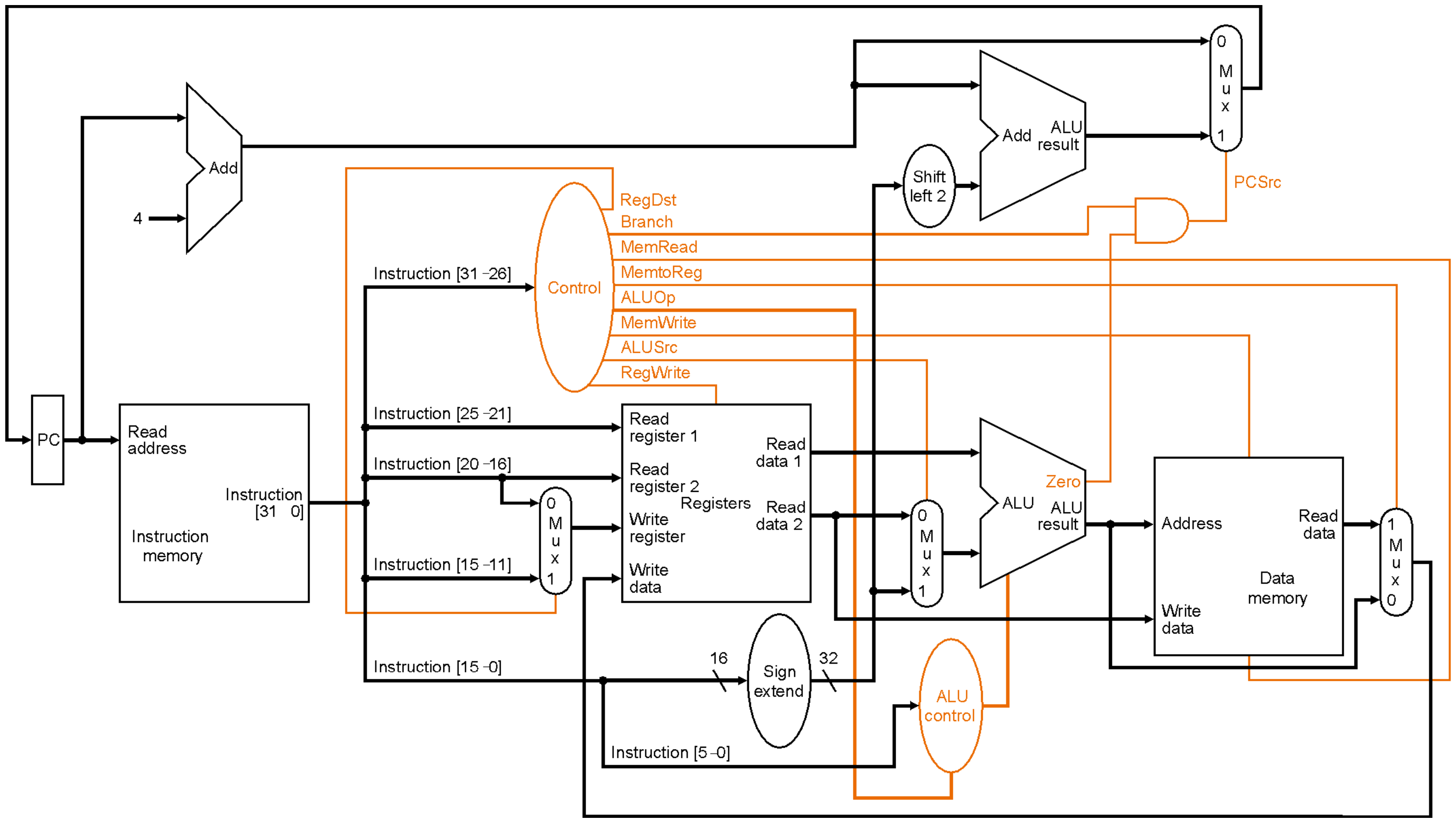
Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUp0
R-format	1	0	0	1	0	0	0	1	0
`lw`								0	0
`sw`								0	0
`beq`								0	1

SW Control



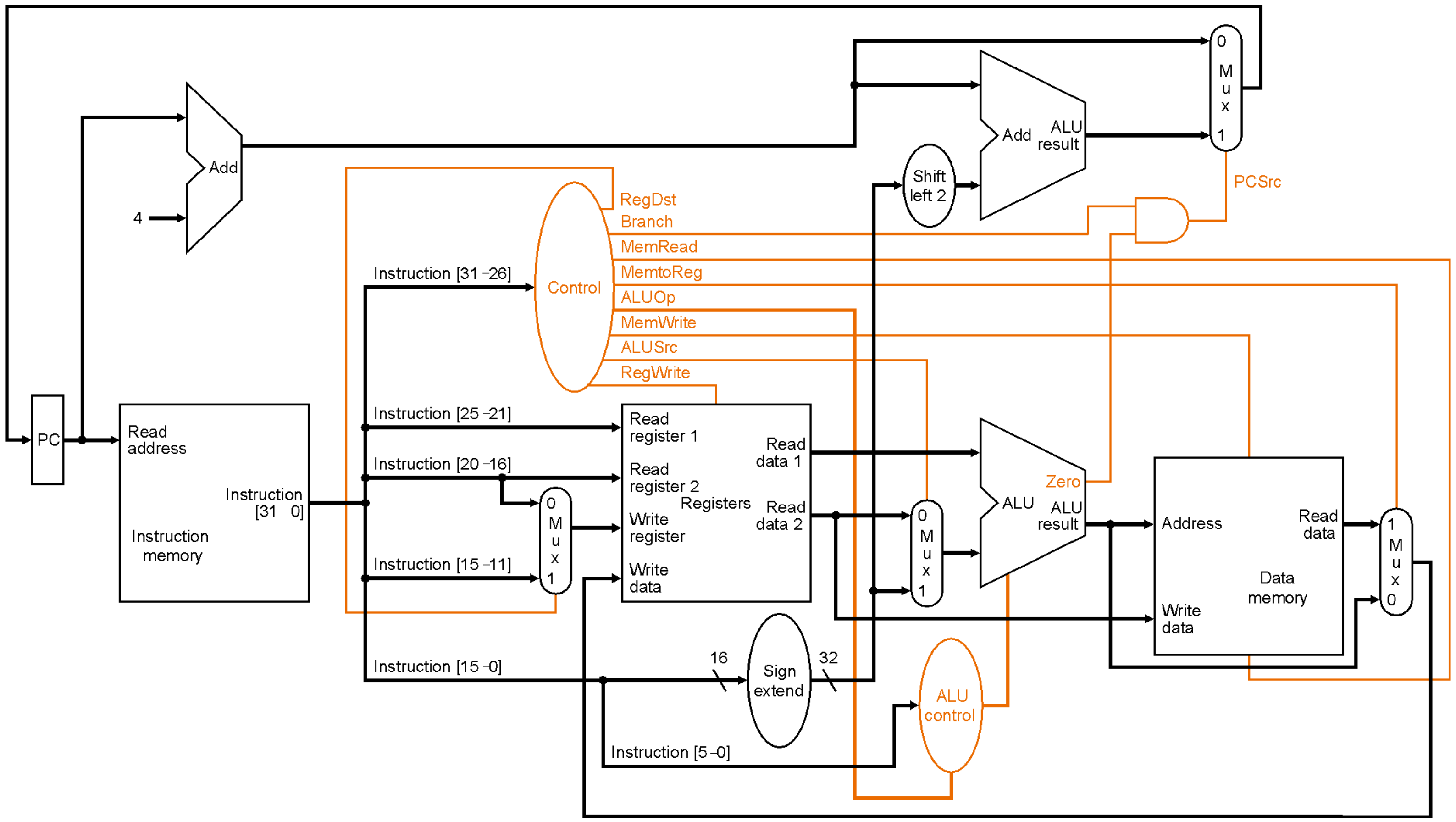
Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw								0	0
beq								0	1

Beq 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								0	1

Beq Control

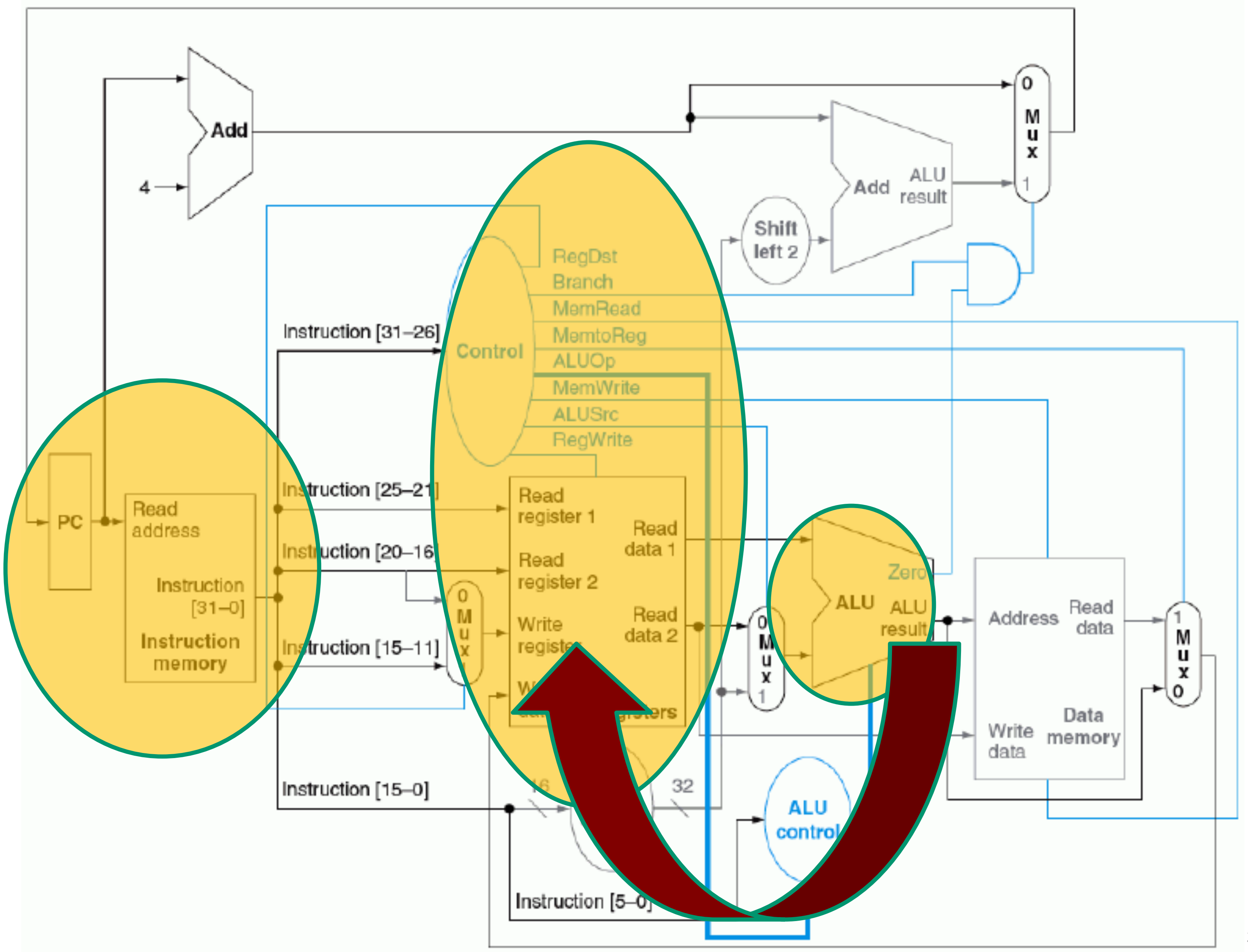


Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUp0
R-format	1	0	0	1	0	0	0	1	0
`lw`	0	1	1	1	1	0	0	0	0
`sw`	X	1	X	0	0	1	0	0	0
`beq`	X	0	X	0	0	0	1	0	1

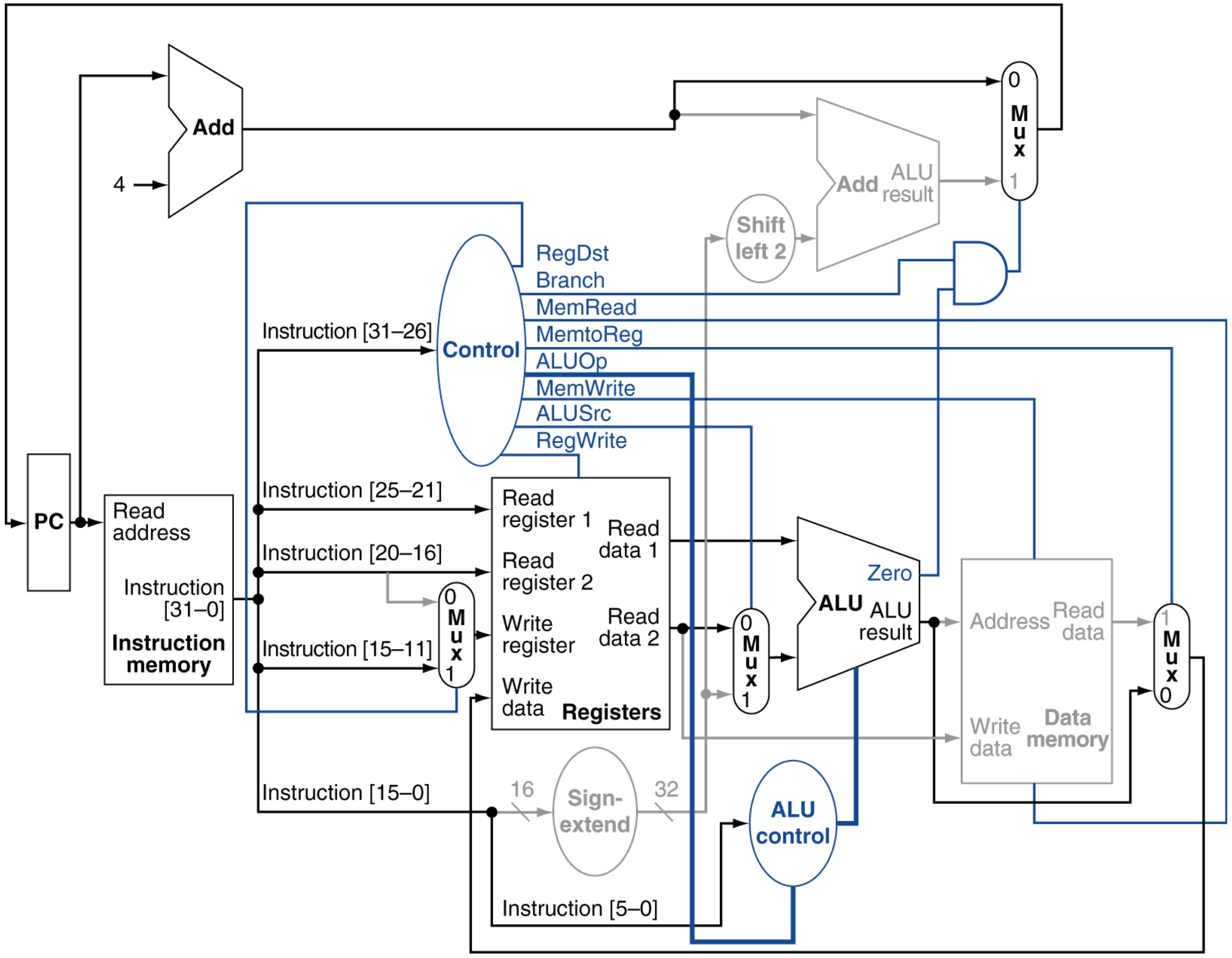
Operation of Datapath: R-Type

- 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 (cont.)



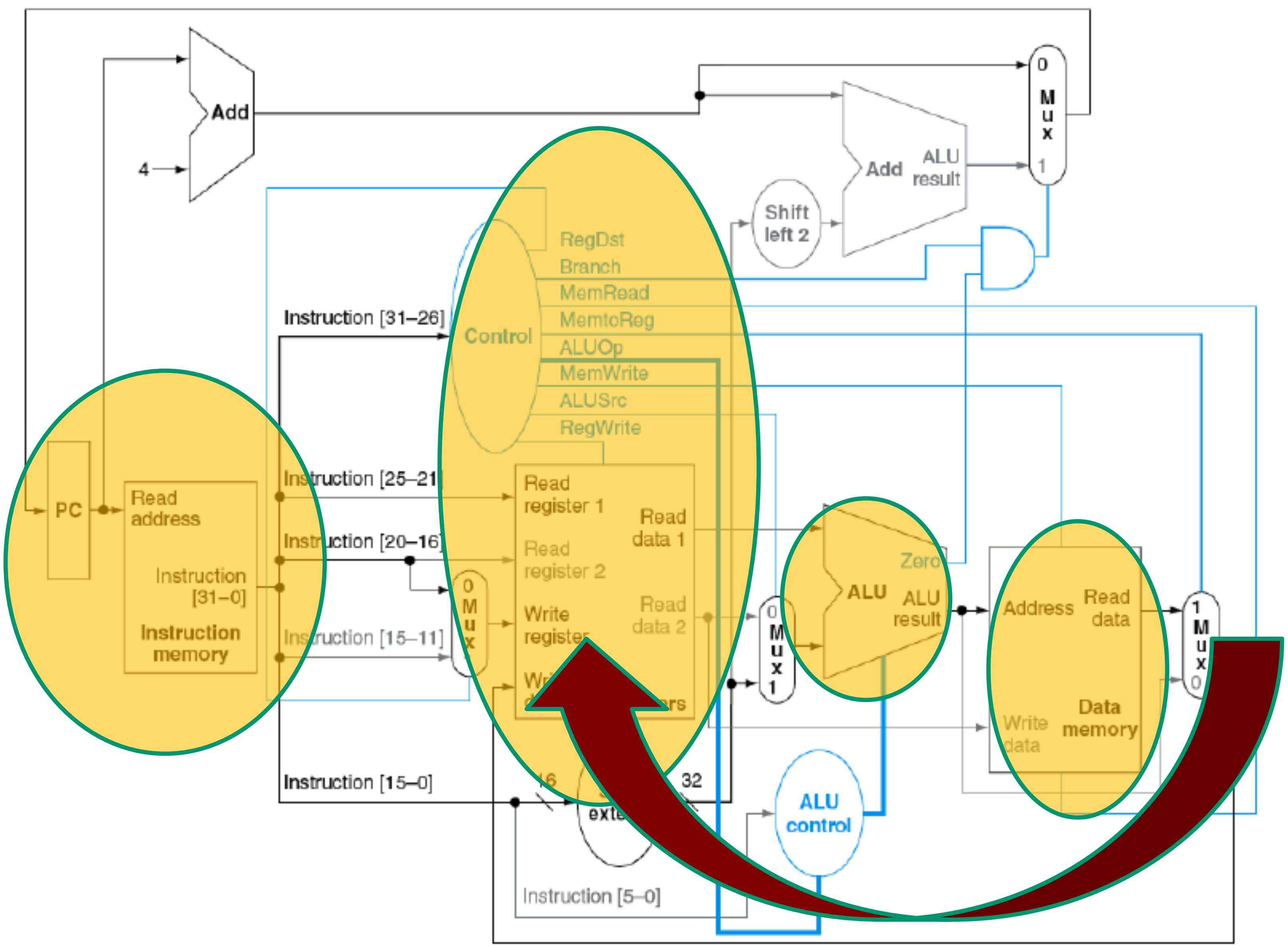
R-Type Datapath + Control Signals



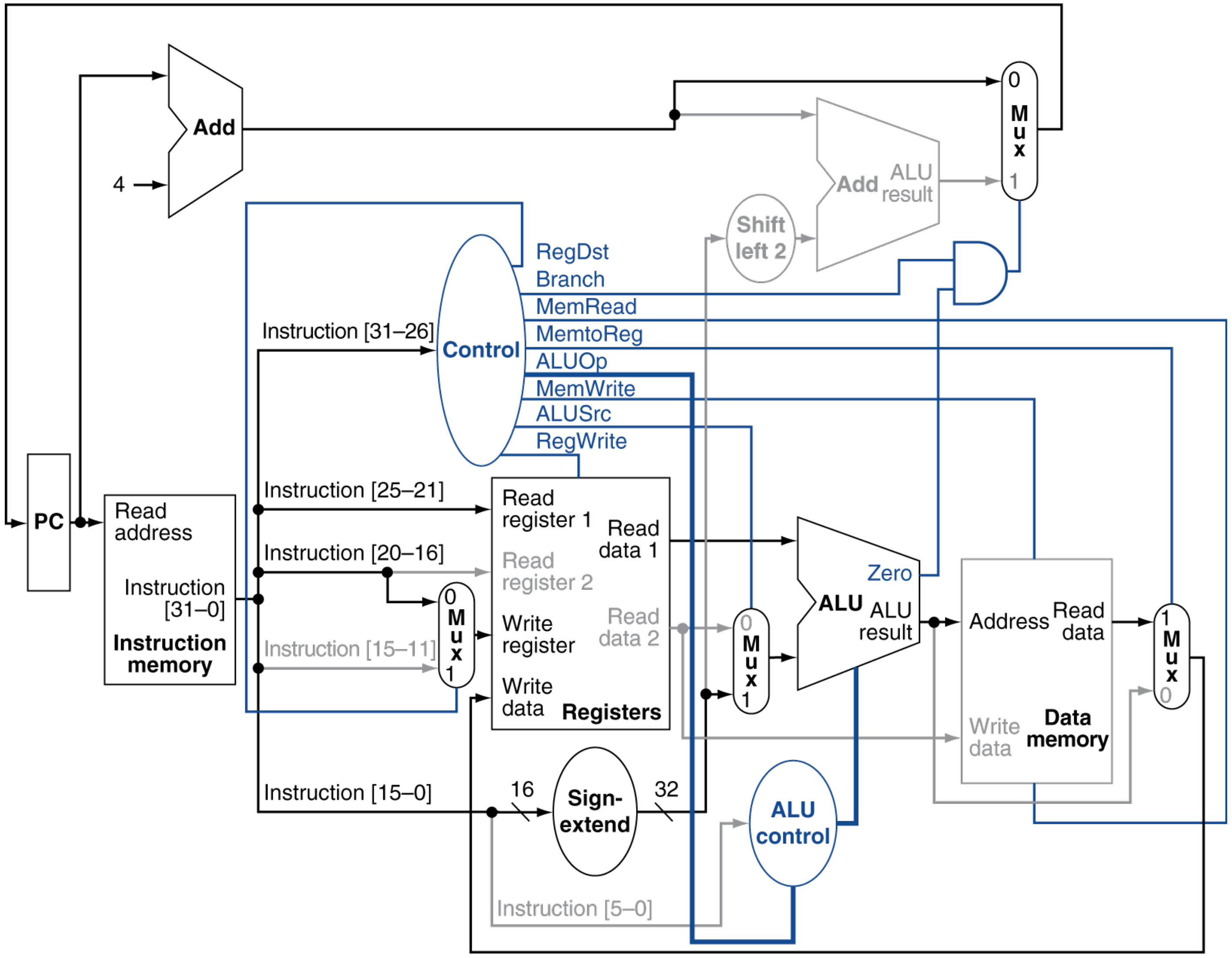
Operation of Datapath: Load

- 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 $t1 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.)



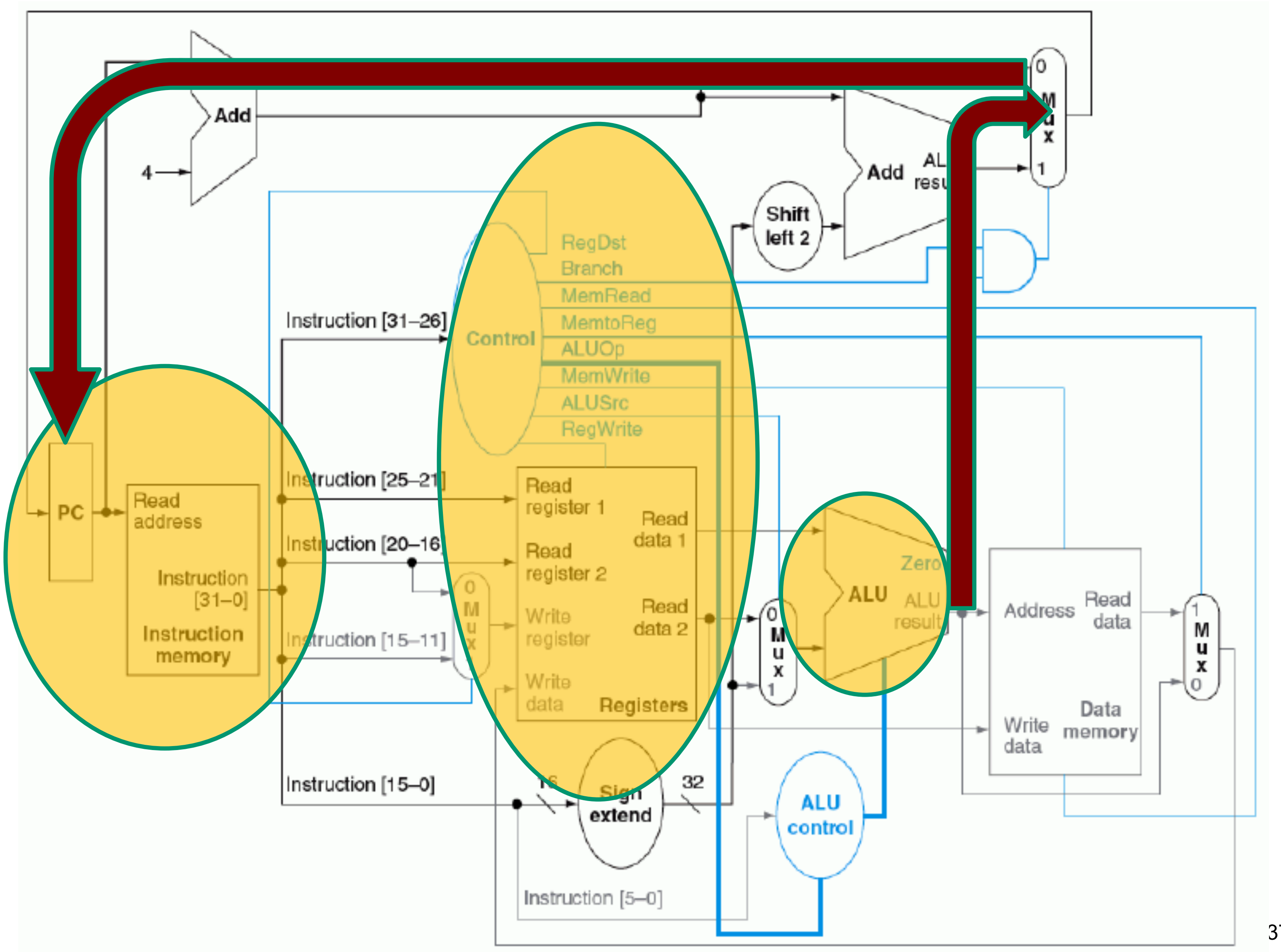
Load Datapath + Control Signals



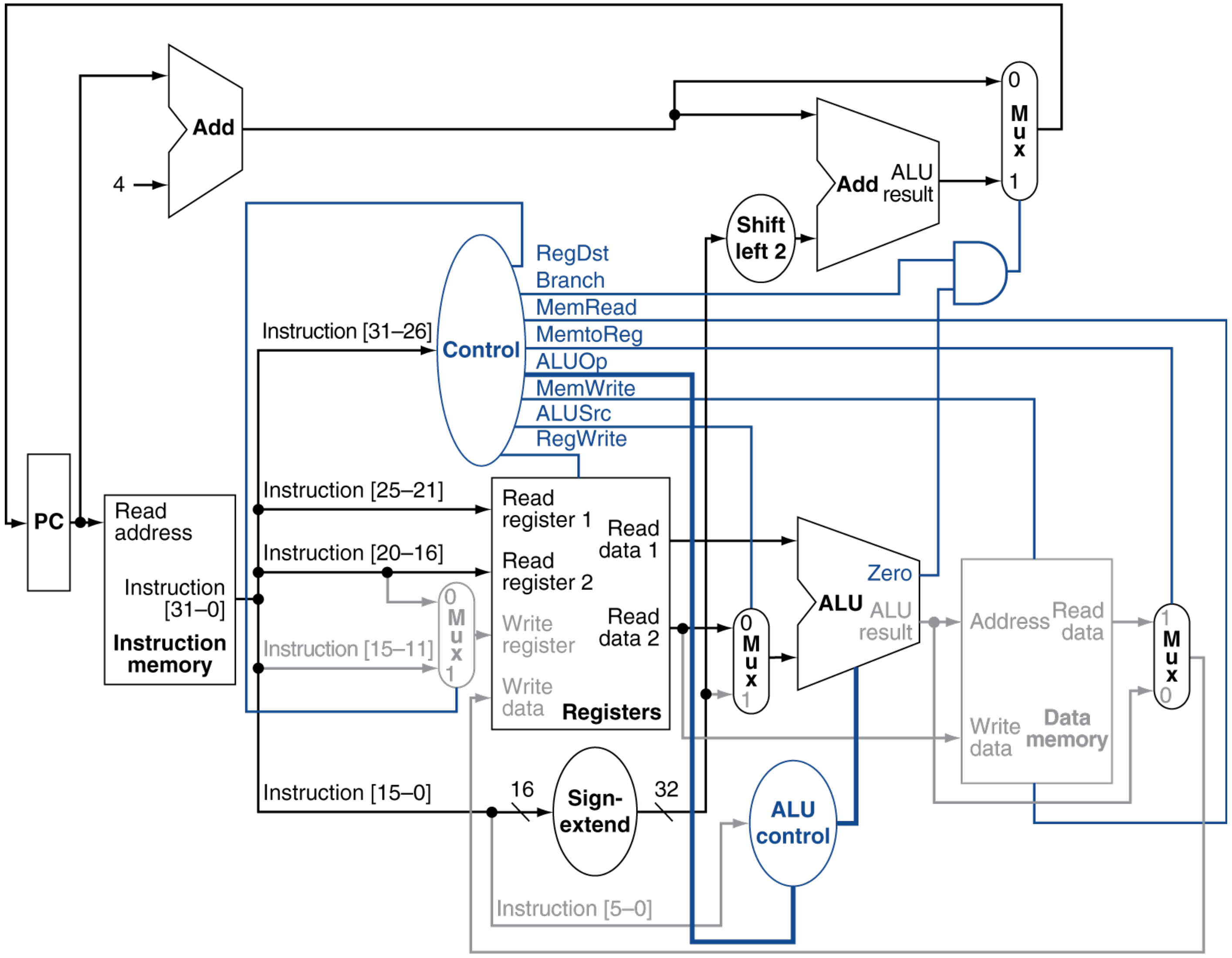
Operation of Datapath: Branch Equal

- 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.)

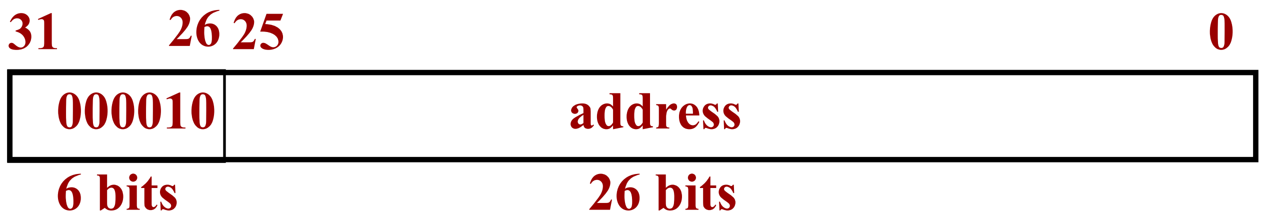


BEQ Datapath + Control Signals



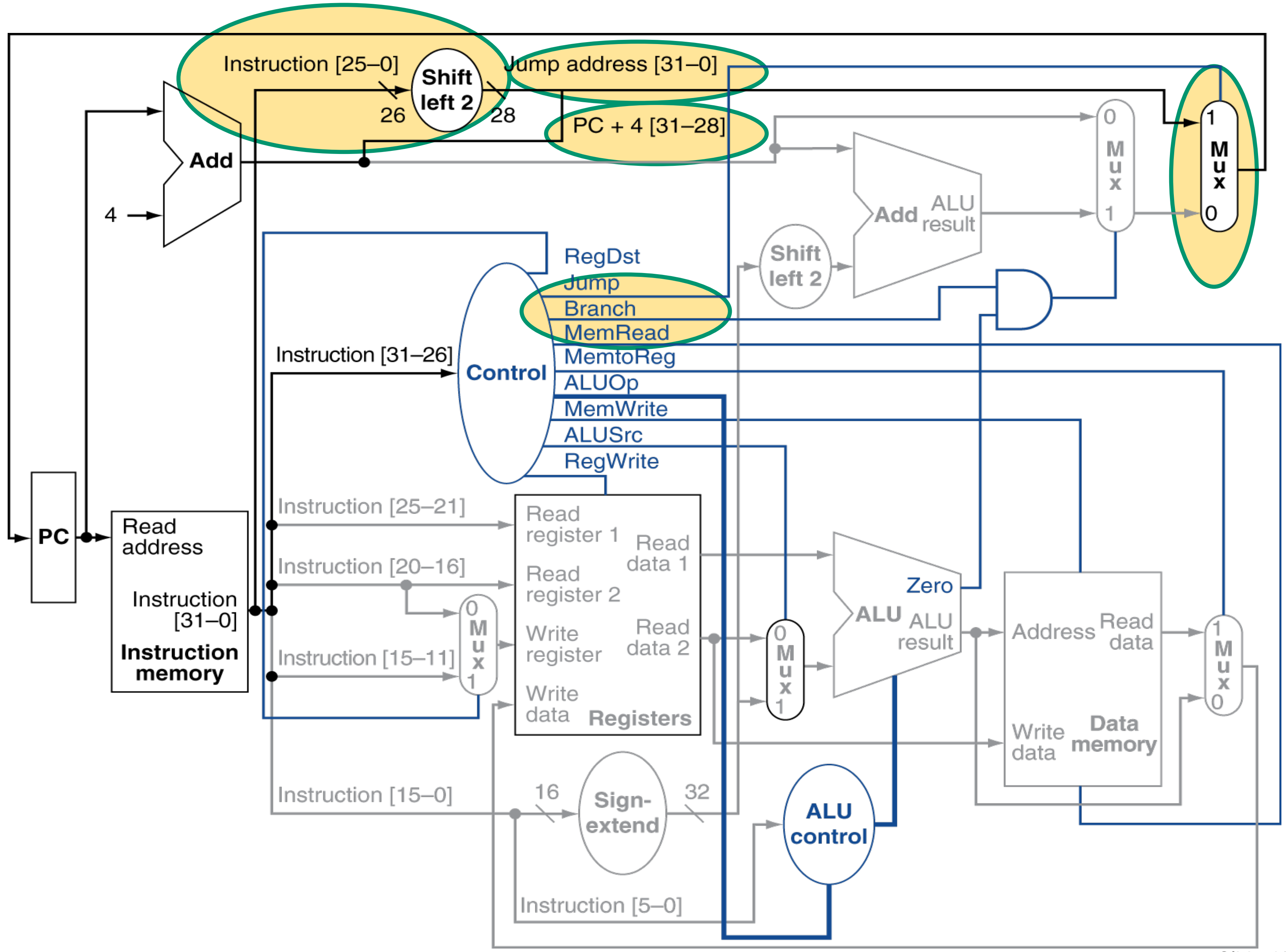
Implementing Unconditional Branch

- Jump Instruction Formation



- 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



Thanks for Your Attention!