

Computer Architecture: MIPS Datapath Design: Single Cycle

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.



Our Lectur Today



Topics Covered Today

- MIPS ISA: Quick Review
- MIPS Single-Cycle Datapath



Instruction Encoding

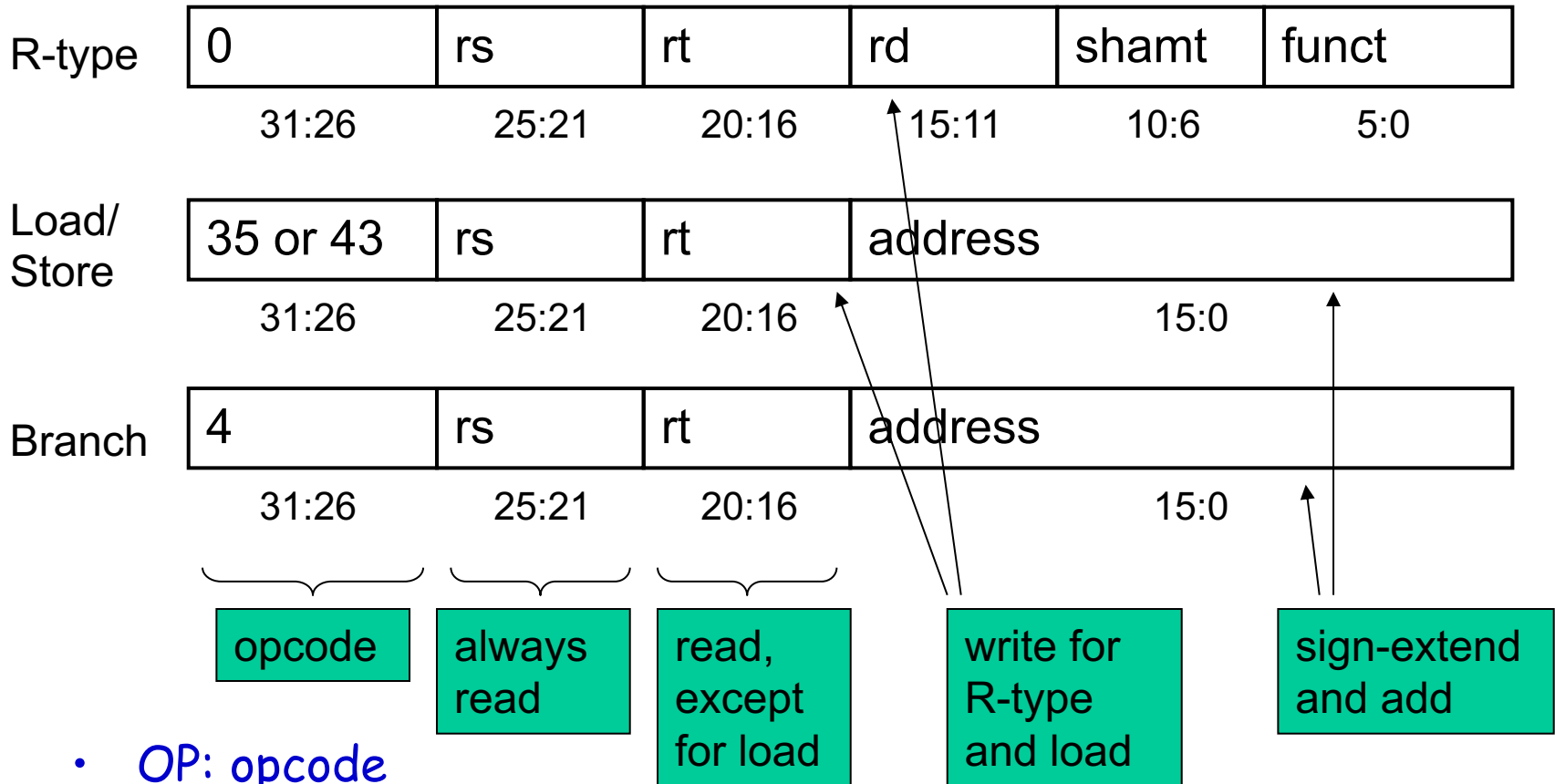
- MIPS Instruction Format

	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
r format	OP	rs	rt	rd	sa	funct
i format	OP	rs	rt	immediate		
j format	OP	target				

- OP: opcode
- rs: first register source operand
- rt: second register source operand
- rd: register destination operand
- sa: shift amount
- funct: function code



MIPS ISA Format



- OP: opcode
- rs: first register source operand
- rt: second register source operand
- rd: register destination operand
- sa: shift amount
- funct: function code

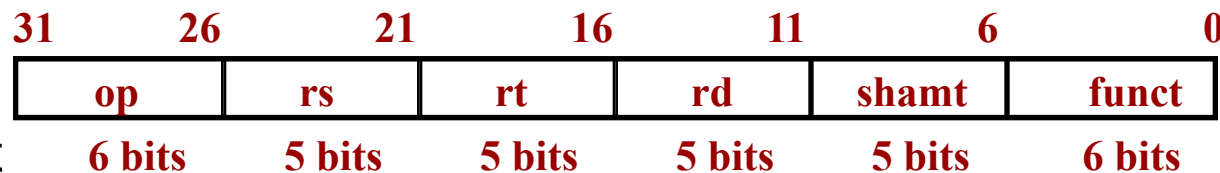


MIPS Instruction Encoding

- R-TYPE

- *add rd, rs, rt*

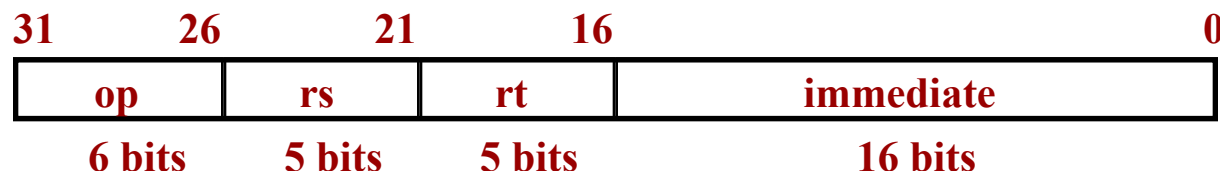
- *sub, and, or, slt*



- LOAD / STORE

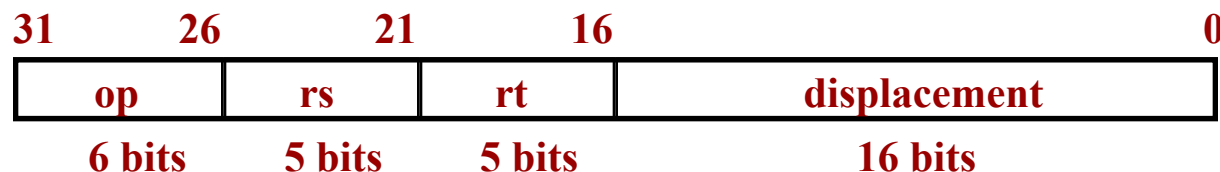
- *lw rt, rs, imm*

- *sw rt, rs, imm*

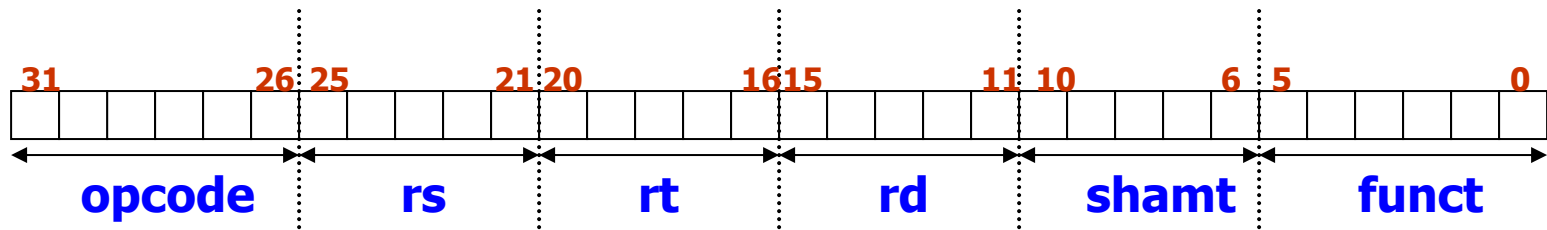


- BRANCH

- *beq rs, rt, imm*



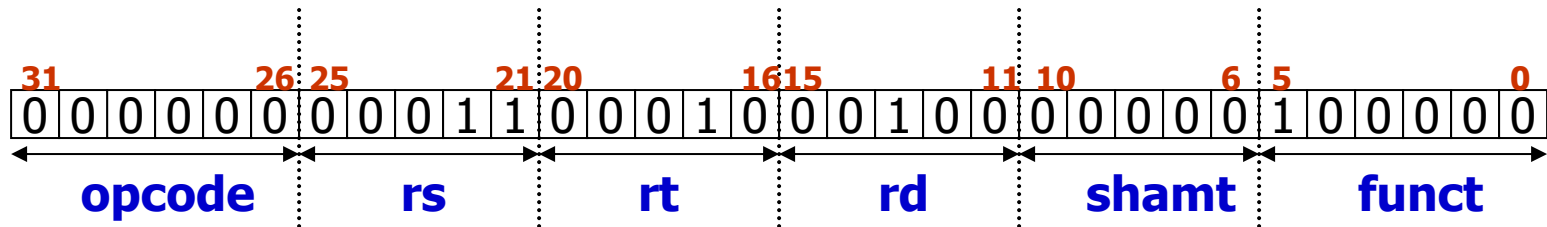
MIPS Instruction Encoding: R-Type



add \$4, \$3, \$2

Diagram showing the mapping of register names to fields:

- rd** (Destination Register)
- rt** (Source Register 2)
- rs** (Source Register 1)



0000000000011000010000010000000000100000

Encoding = 0x00622020

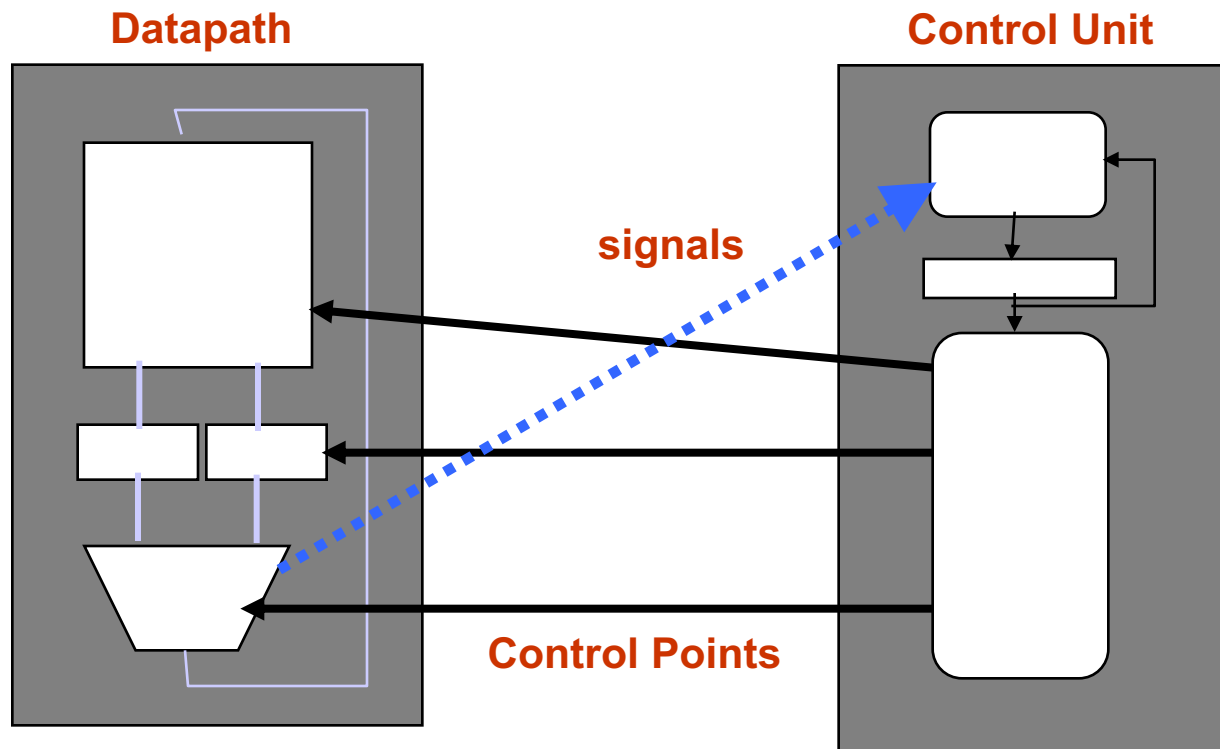


Datapath

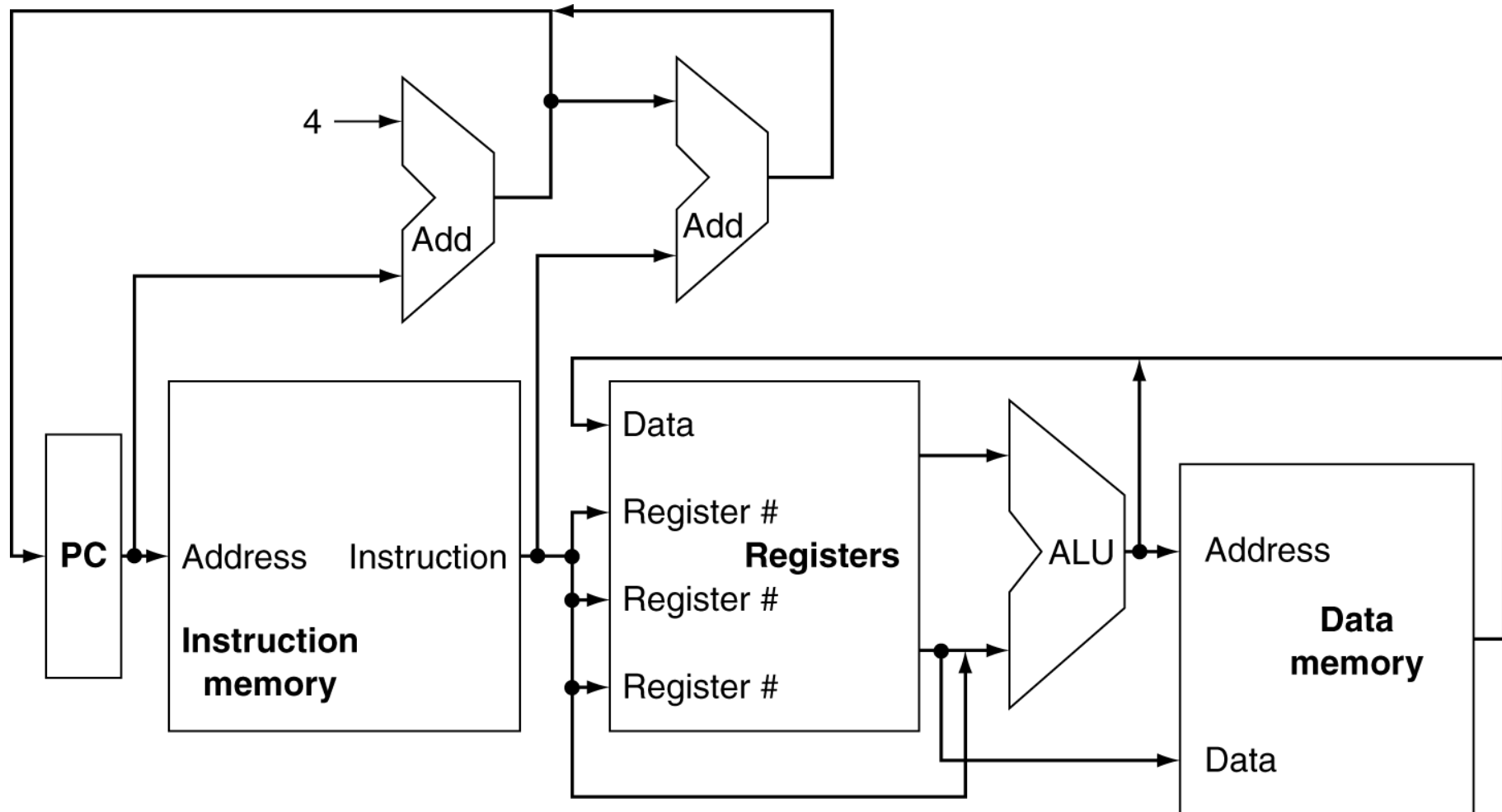
- Datapath?
 - A functional unit used to **operate** on or **hold** data within a processors
- Datapath Elements in MIPS
 - Instruction memory
 - Data memory
 - Register file
 - ALU
 - Adders
- Control Unit
 - Schedule data movements in datapath



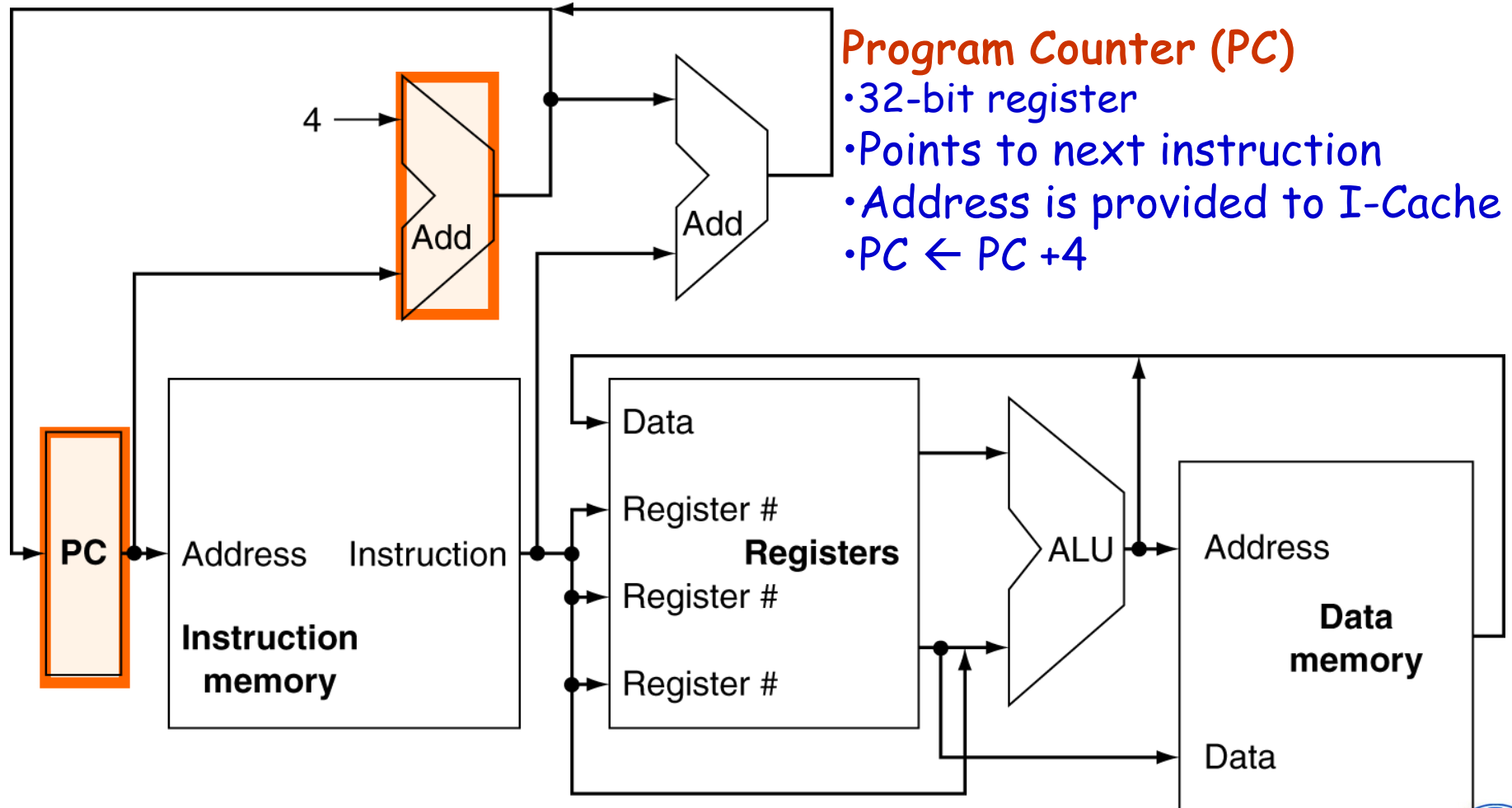
Datapath (cont.)



Single Cycle MIPS Datapath

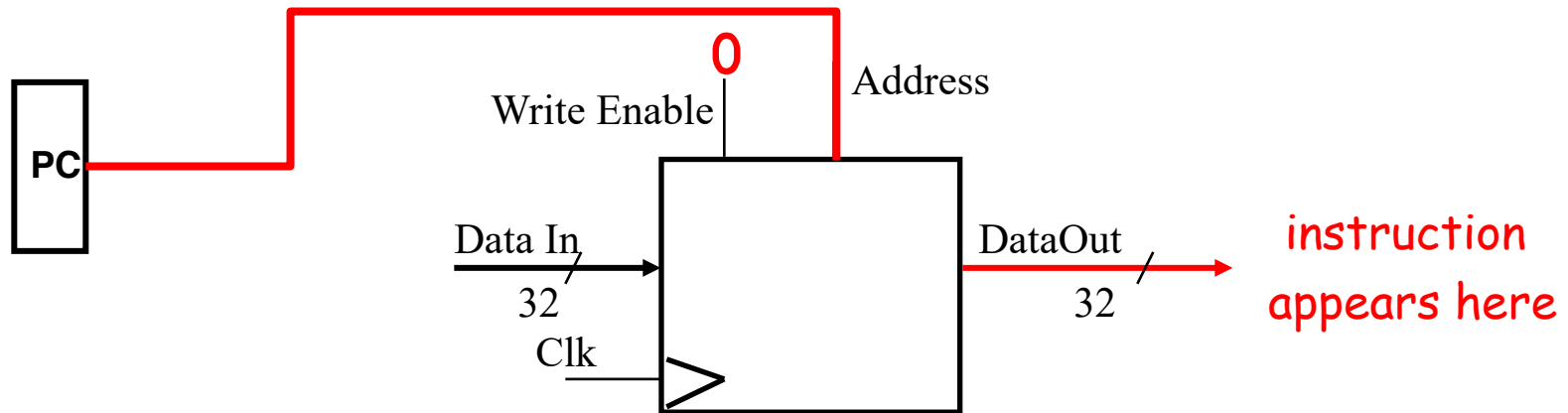


Single Cycle MIPS Datapath: PC

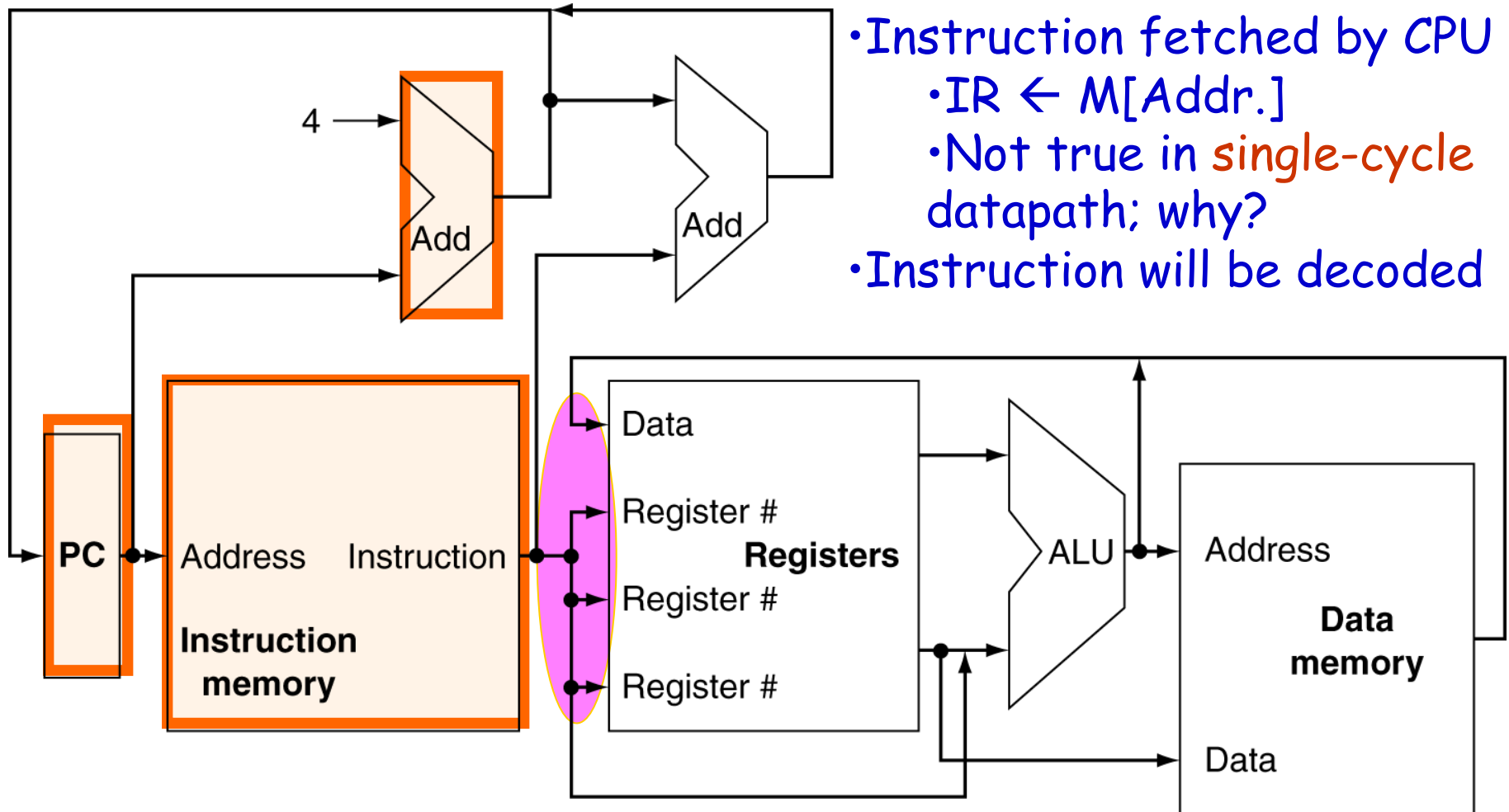


Instruction Fetch

- Program Counter (PC)
 - Supplies instruction address
 - Get instructions from memory

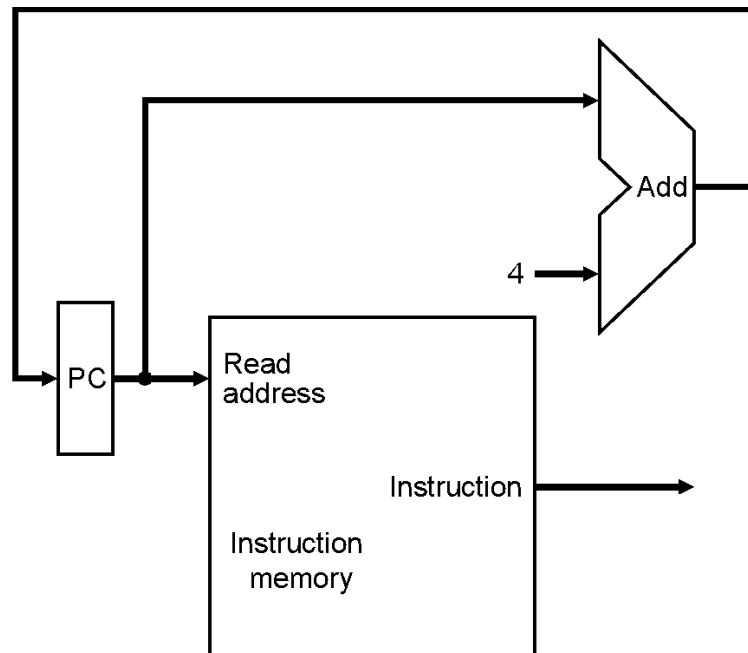


Instruction Memory (I-Cache)

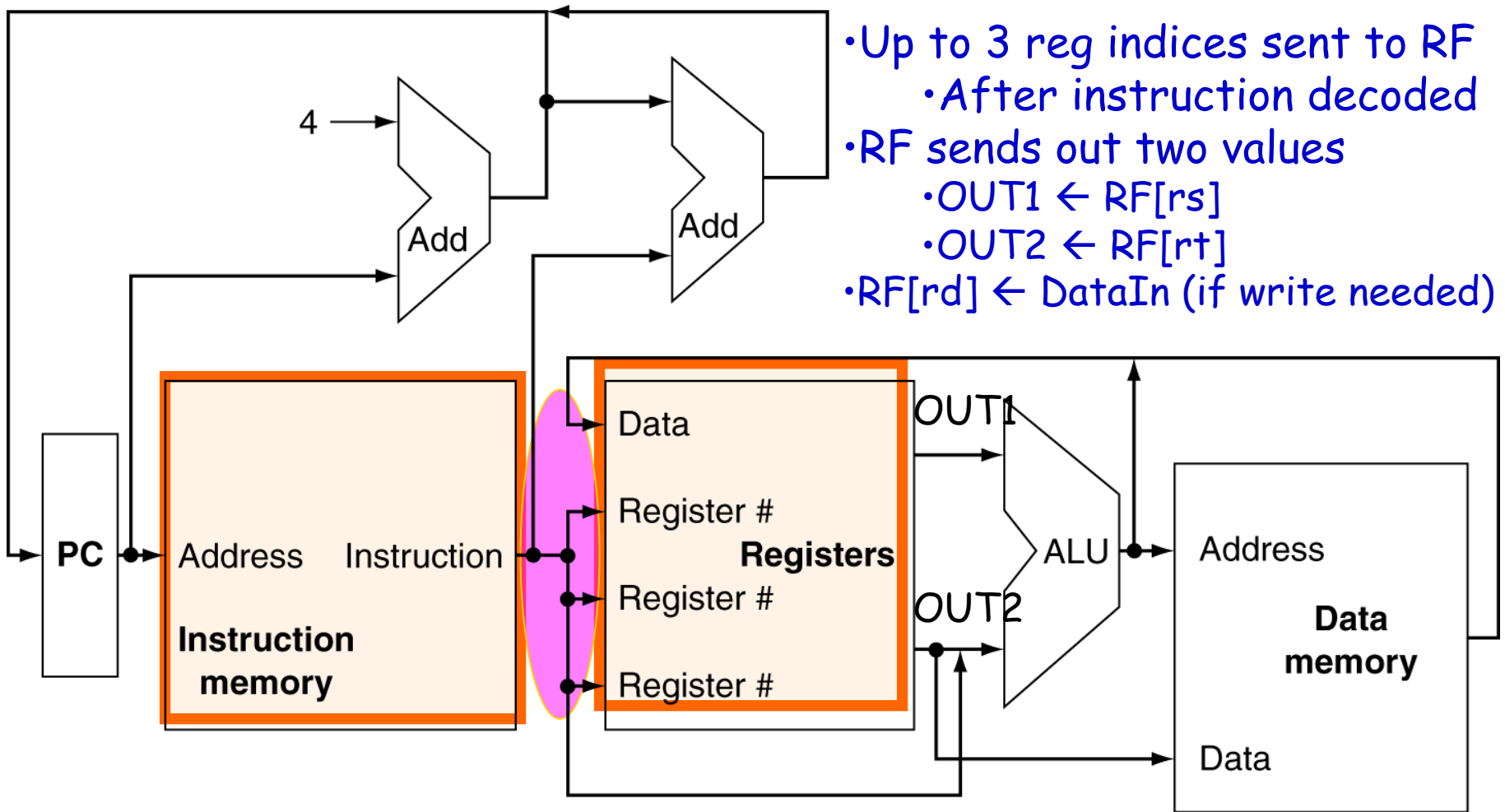


Instruction Fetch Unit

- Updating PC for Next Instruction
 - Sequential Code: $PC \leftarrow PC + 4$
 - Branch and Jump: $PC \leftarrow \text{New Address}$
 - we'll worry about this later

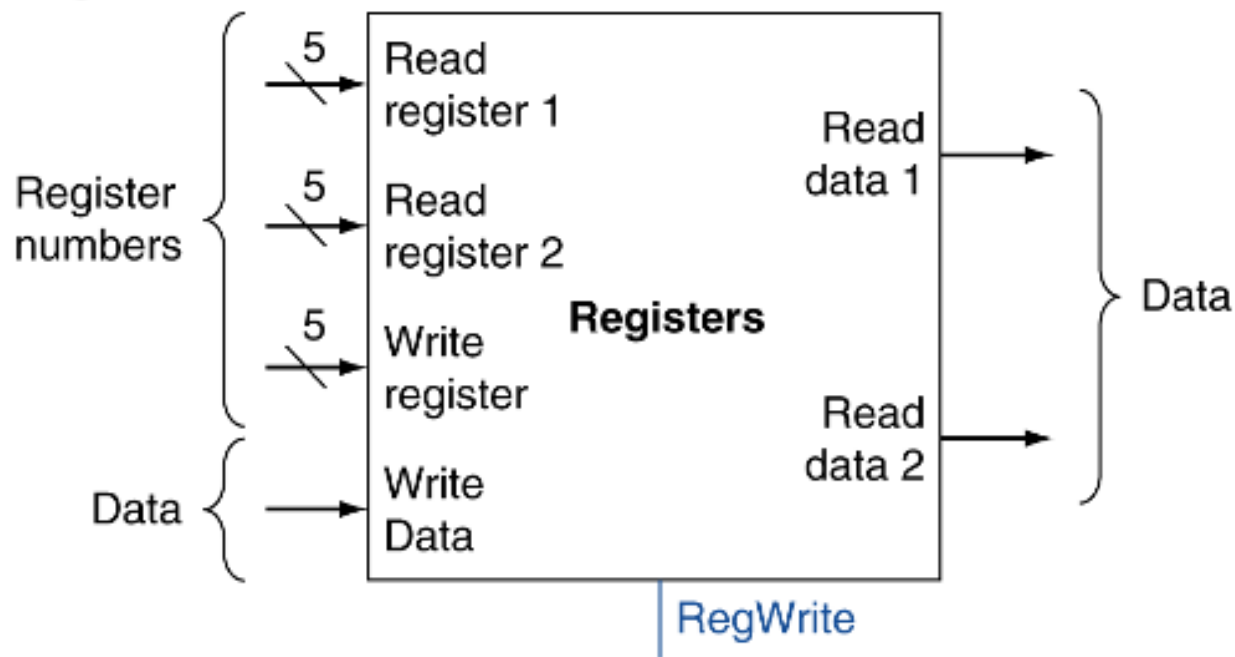


Register File



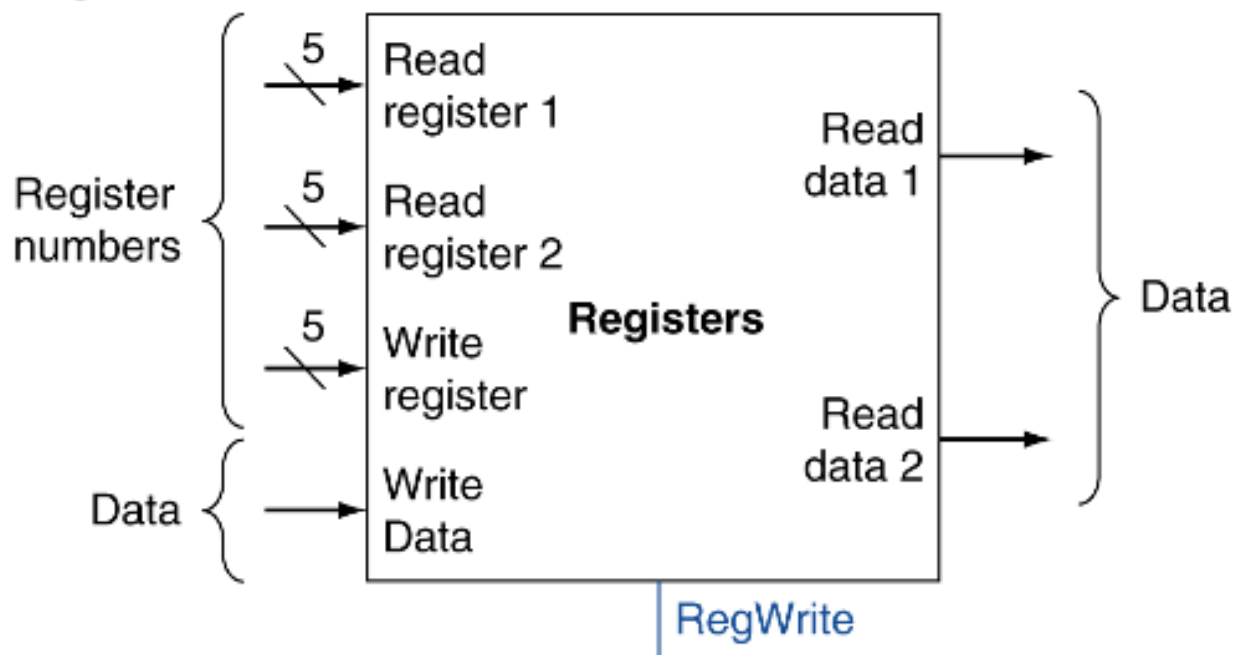
Datapath Elements

- Register File (RF)
 - Also called, General Purpose Registers (GPR)
 - Up to three indices as inputs
 - 32-bit write input data
 - Two 32-bit outputs



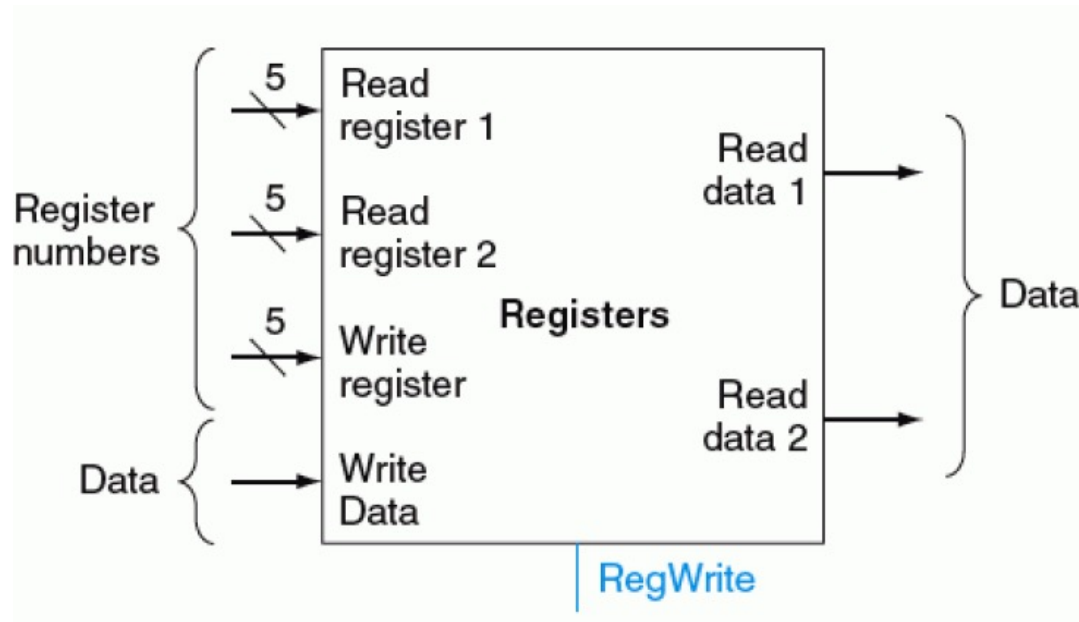
Datapath Elements (cont.)

- Question 1:
 - How read & write can be accomplished in one clock cycle without data contention?

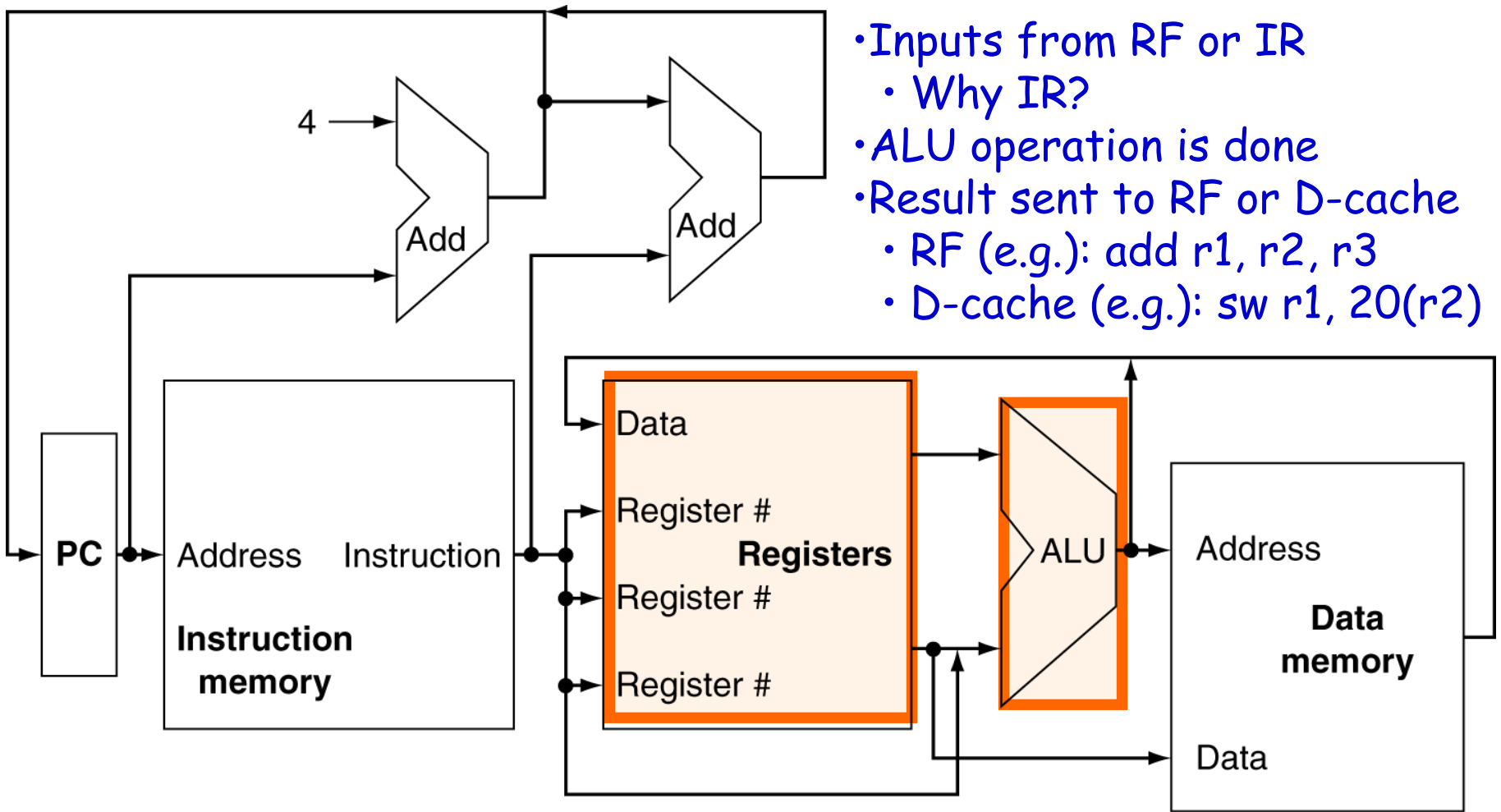


Datapath Elements (cont.)

- Question 2:
 - How two simultaneous read operations possible?



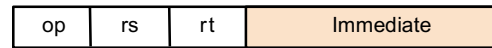
Arithmetic Logical Unit (ALU)



MIPS Addressing Modes

Operand is constant

1. Immediate addressing



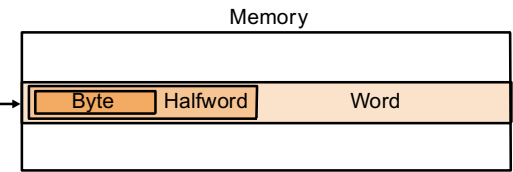
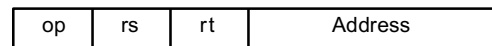
Operand is in register

2. Register addressing



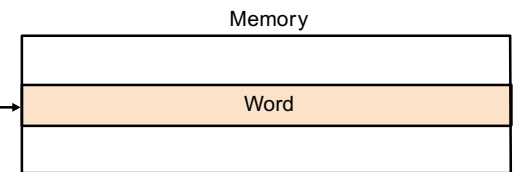
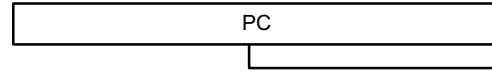
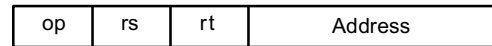
lb \$t0, 48(\$s0)

3. Base addressing



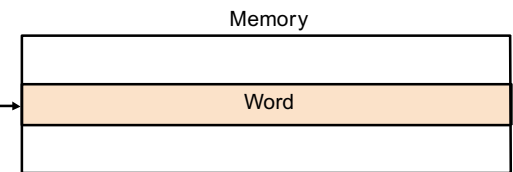
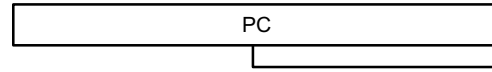
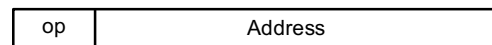
bne \$4, \$5, Label
(label will be assembled into
a distance)

4. PC-relative addressing



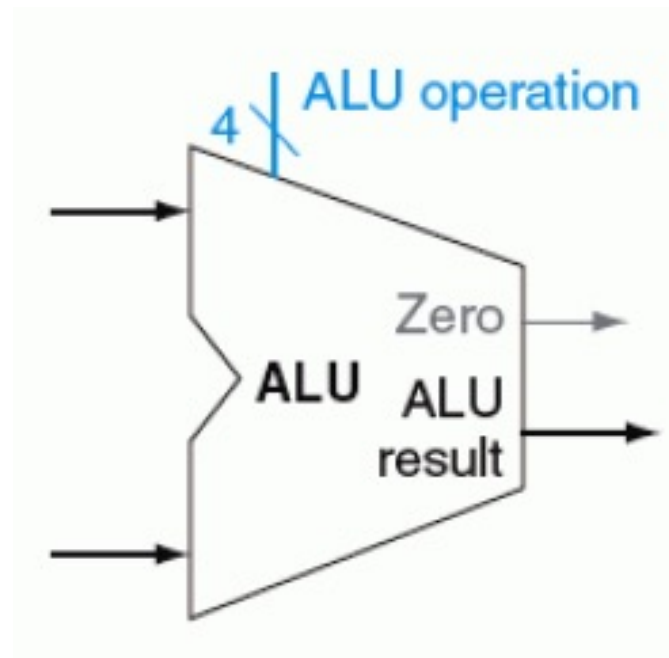
j Label

5. Pseudodirect addressing

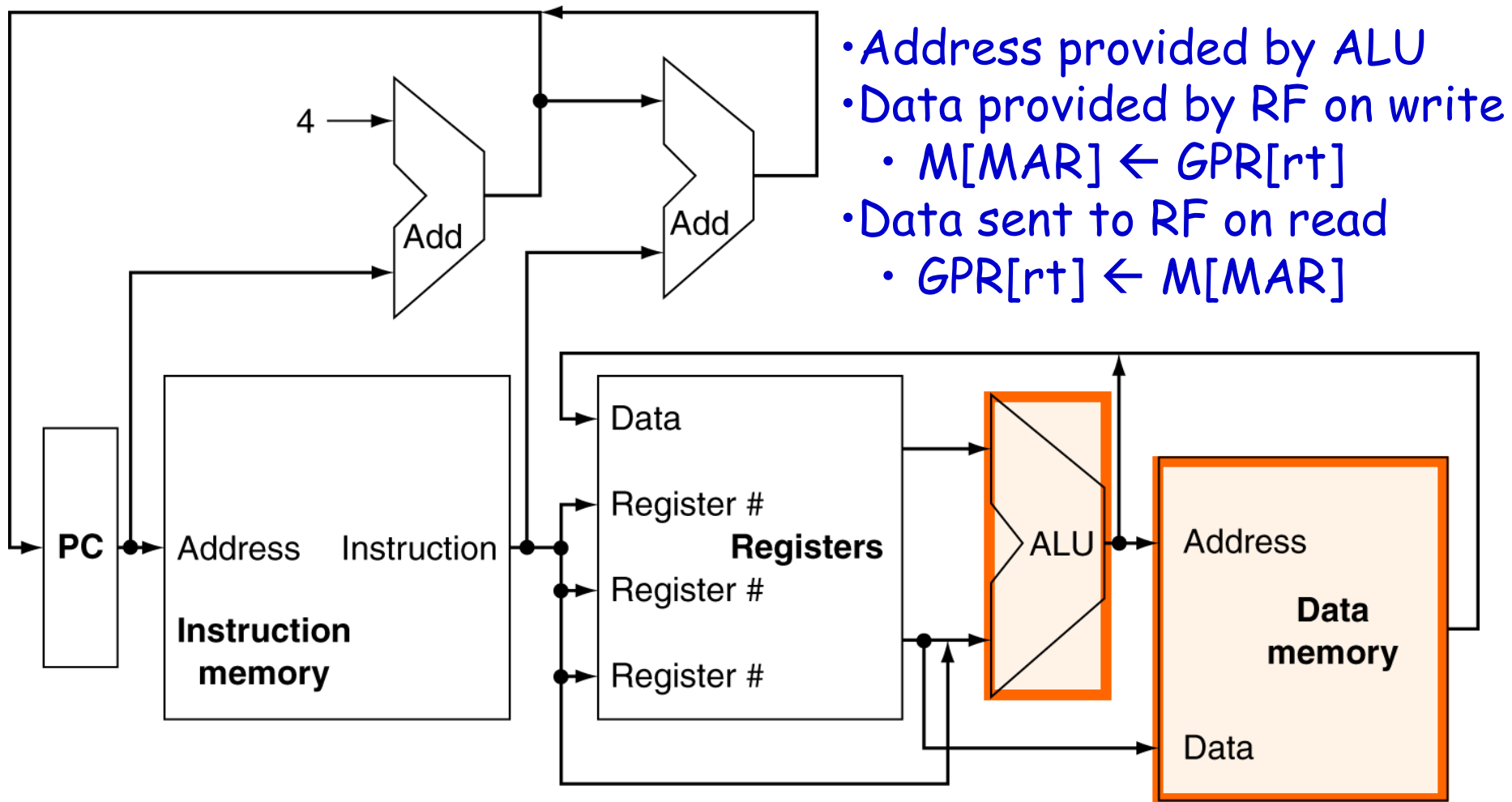


Datapath Elements (cont.)

- ALU
 - Two 32-bit inputs
 - One 32-bit output
 - 4-bit operation selector
 - Zero?



Data Memory (D-Cache)

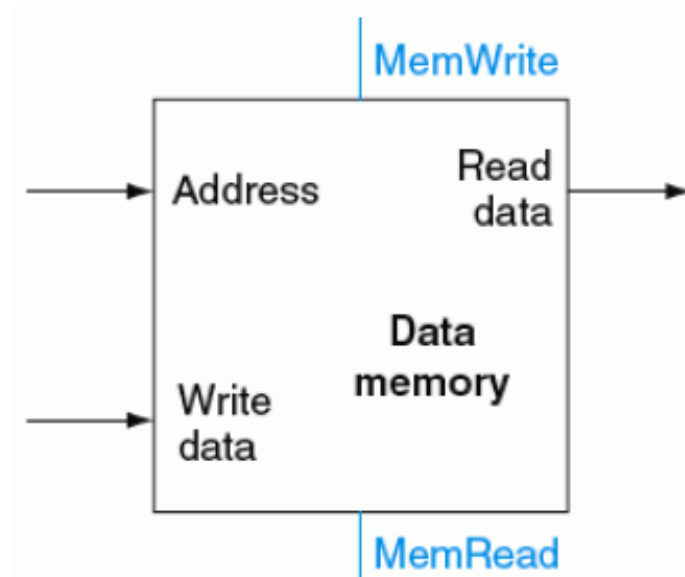


- Address provided by ALU
- Data provided by RF on write
 - $M[MAR] \leftarrow GPR[rt]$
- Data sent to RF on read
 - $GPR[rt] \leftarrow M[MAR]$



Datapath Elements (cont.)

- Data Memory Unit
 - Address
 - Loads/store: $MAR = GPR[rs] + imm16$
 - 32-bit write data
 - $Write\ data \leftarrow GPR[rt]$
 - 32-bit read data
 - $GPR[rt] \leftarrow Read\ data$
 - MemRead signal
 - MemWrite signal

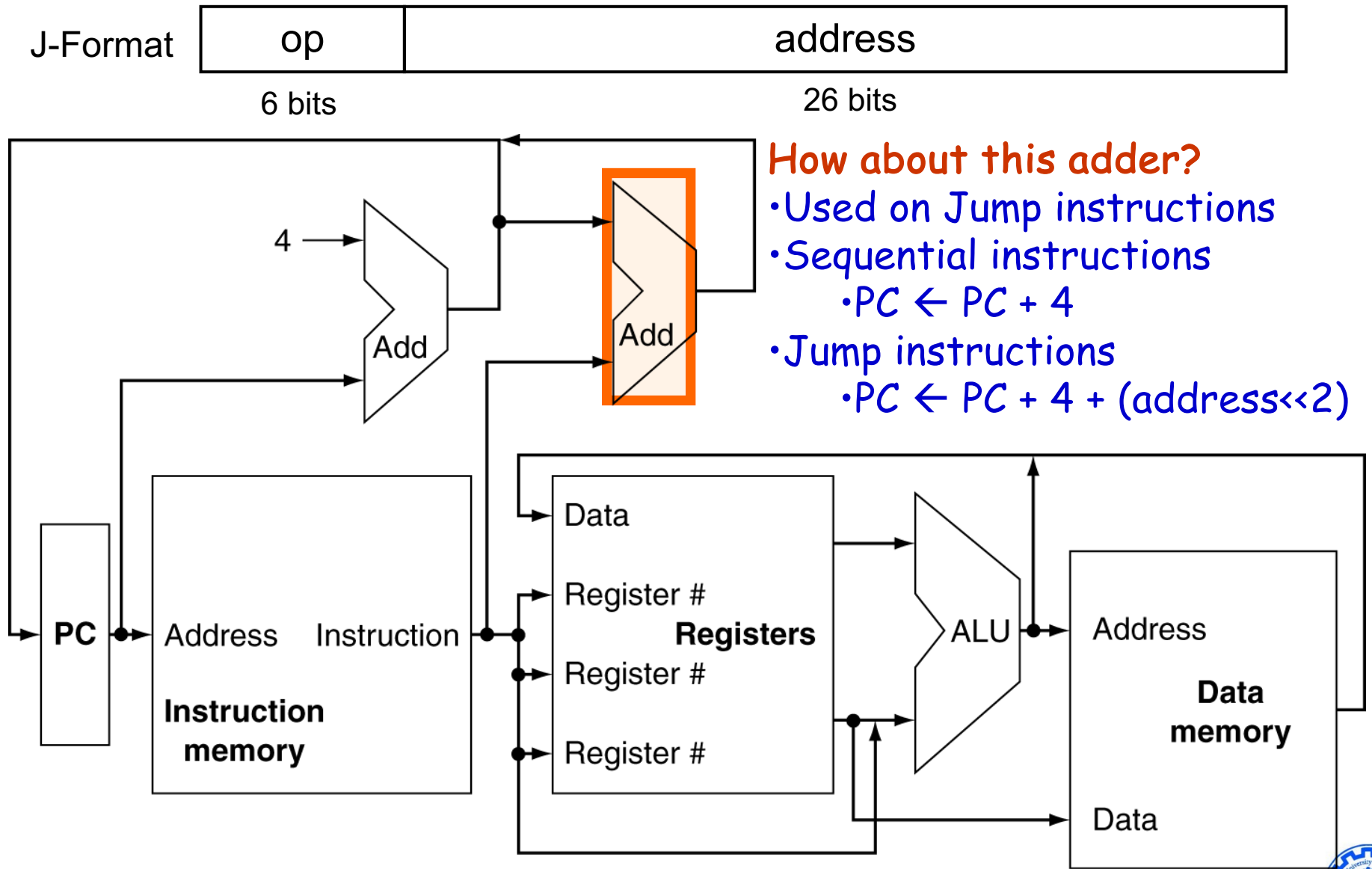


Datapath Elements: Discussion

- Compare **MemRead** and **MemWrite** Activation Process in Following Cases
 - Case A: separate I-cache & D-cache with separate data bus
 - Case B: separate I-cache & D-cache with common data bus
 - Case C: unified I-cache & D-cache

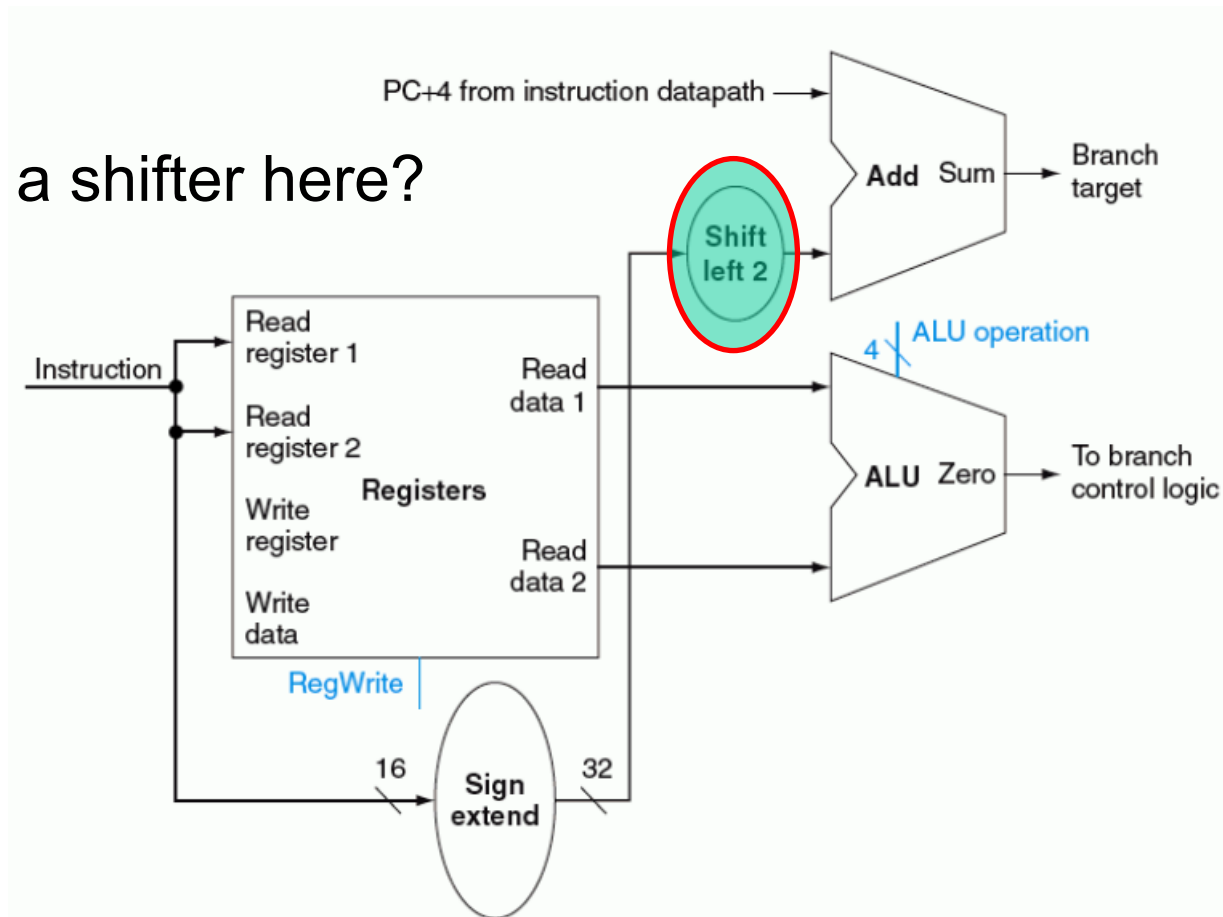


Abstract View of MIPS Implementation



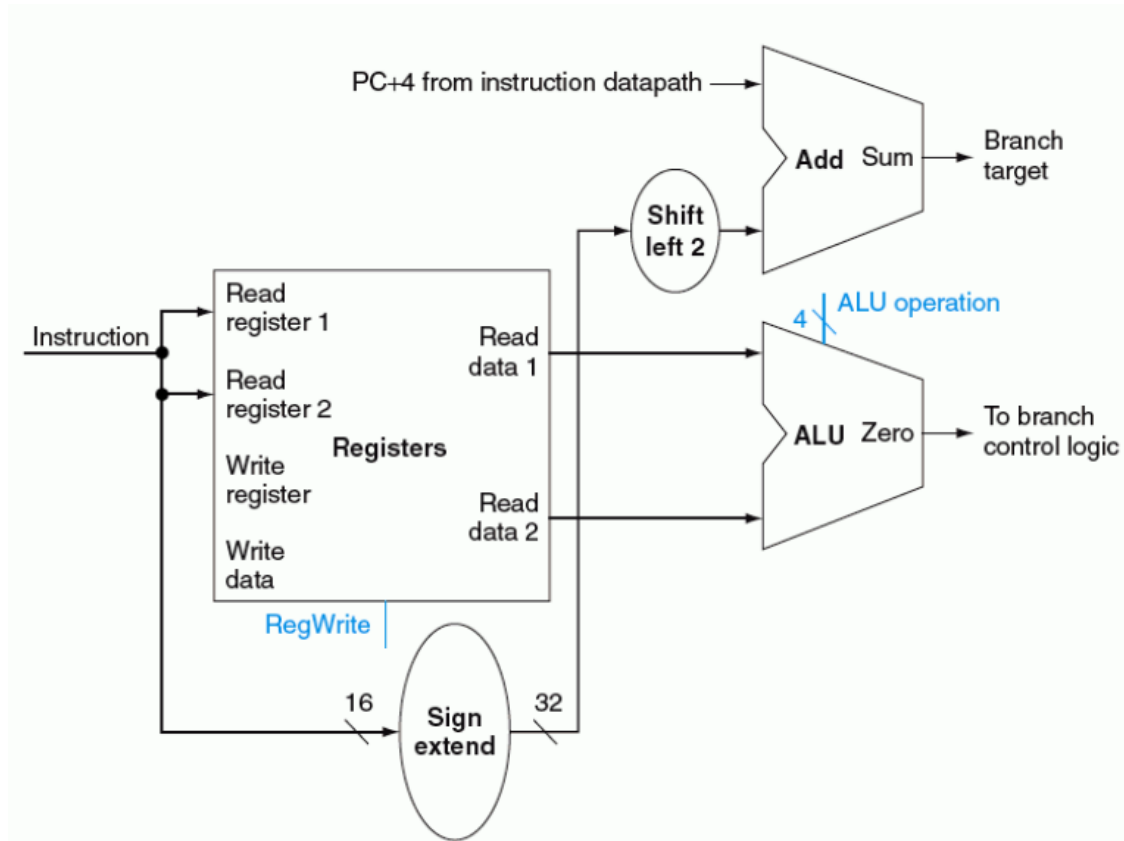
Datapath Elements (cont.)

- Branch Unit
 - Sign extend unit
 - Adder
 - Shifter
 - Do we need a shifter here?

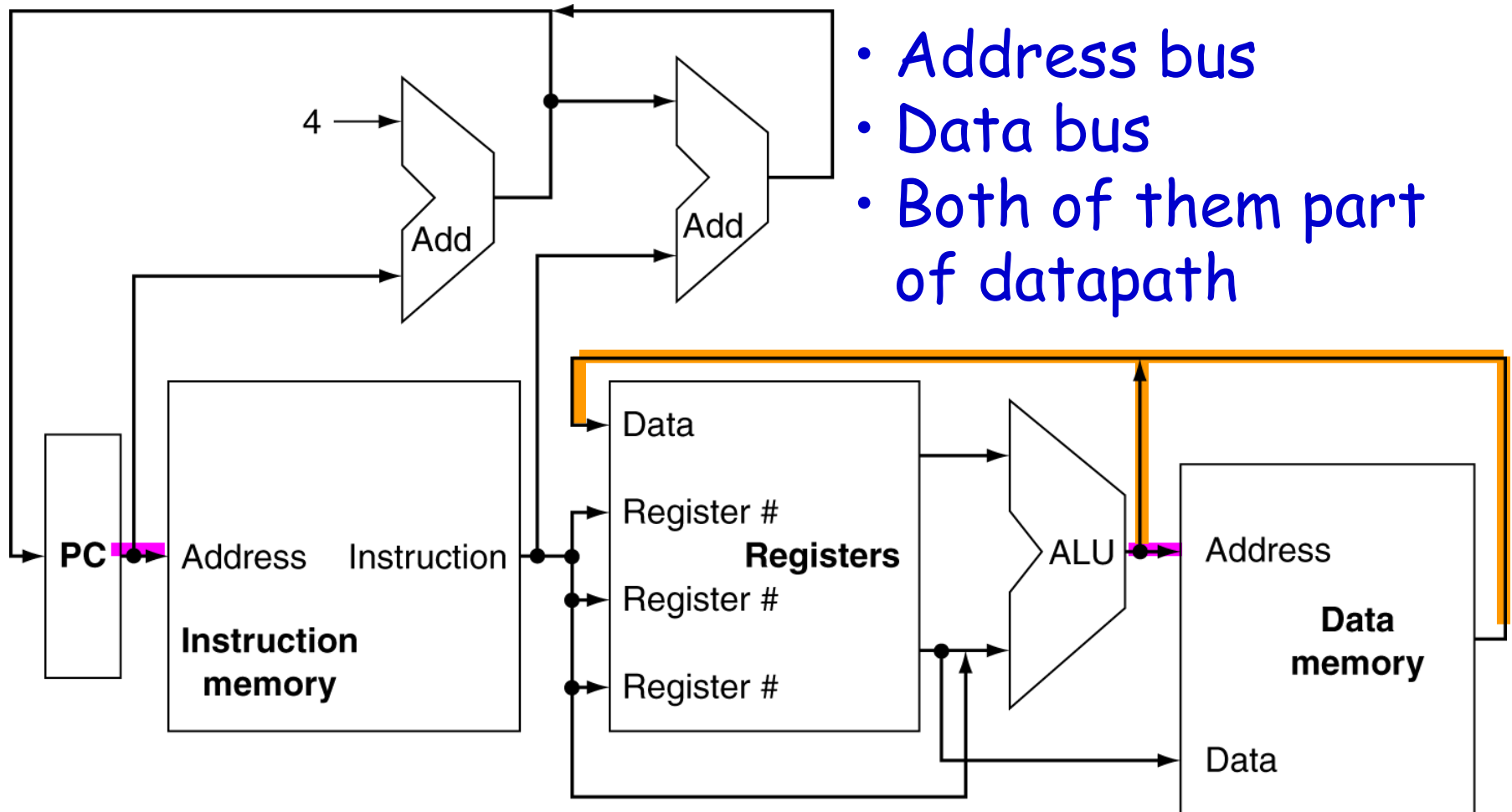


Practice

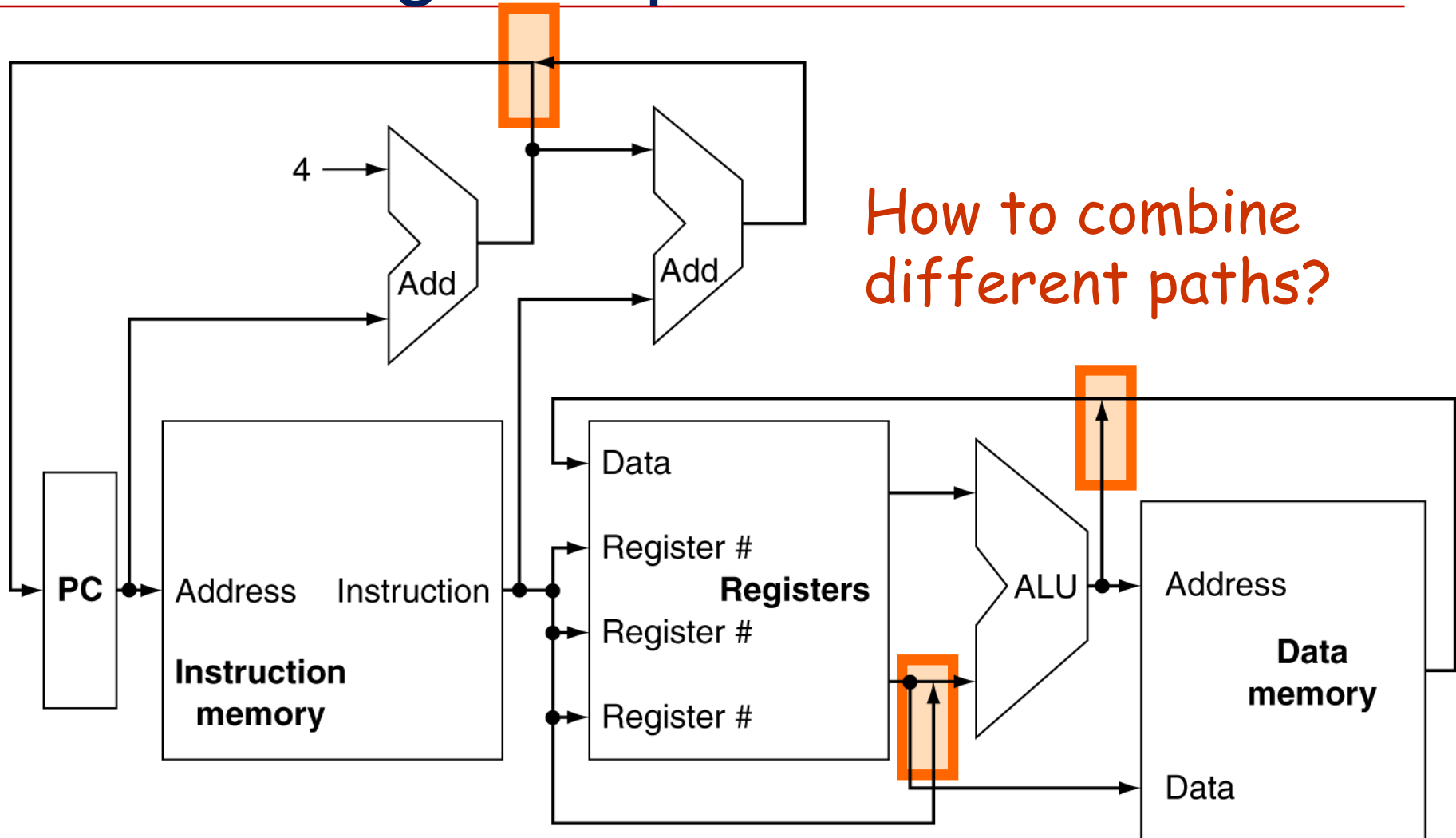
- In Following Circuit:
 - Draw sign-extend & shift logic



Address Bus & Data Bus

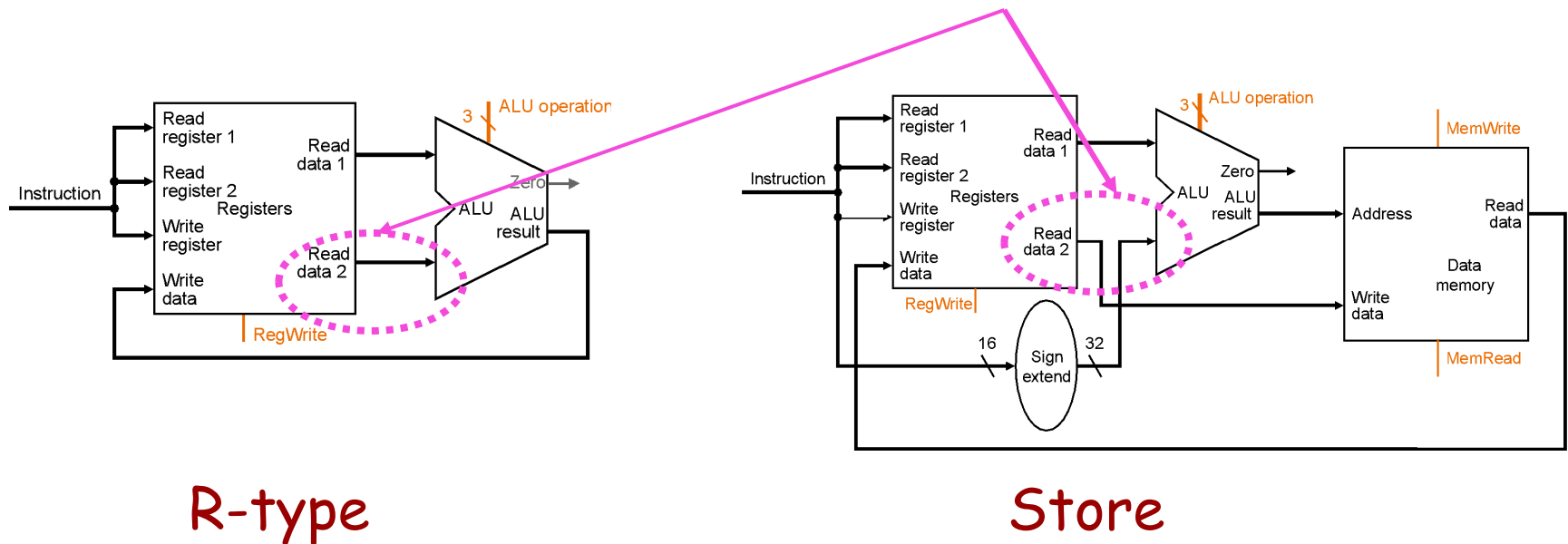


Combining Datapaths



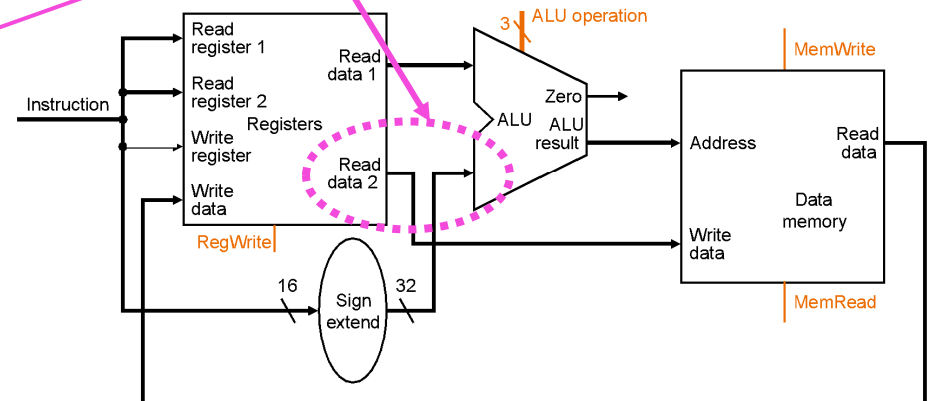
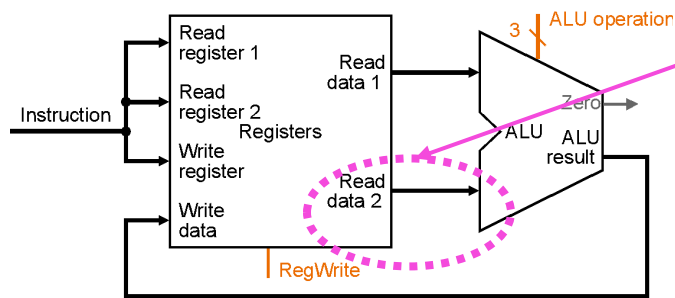
Combining Datapaths (cont.)

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

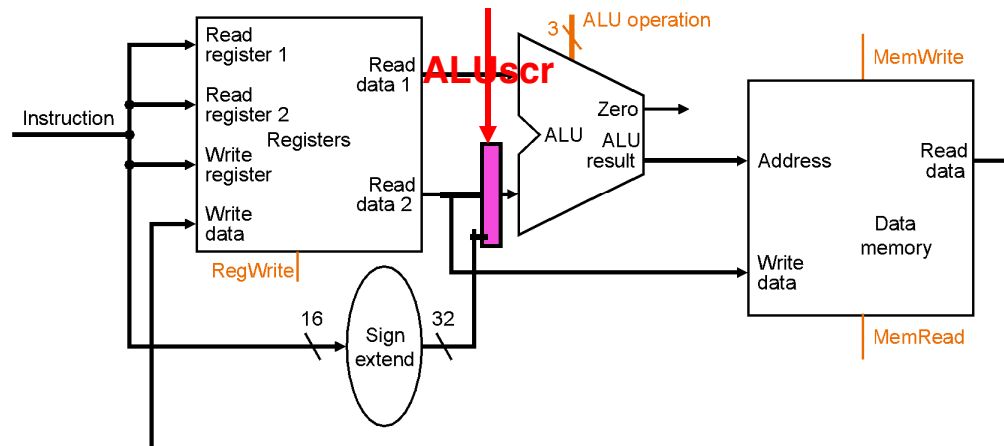


Combining Datapaths (cont.)

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

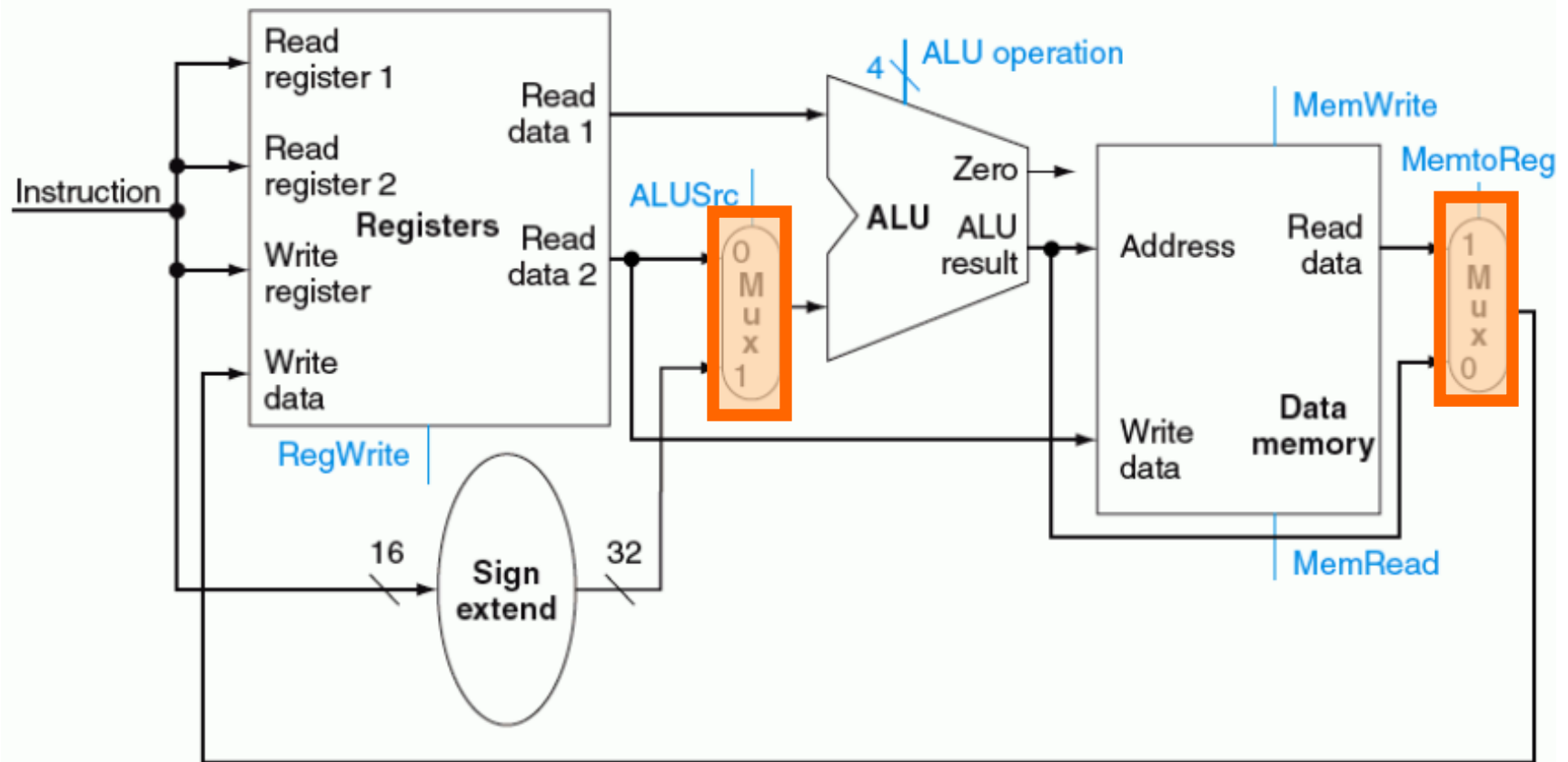


- Use a multiplexer

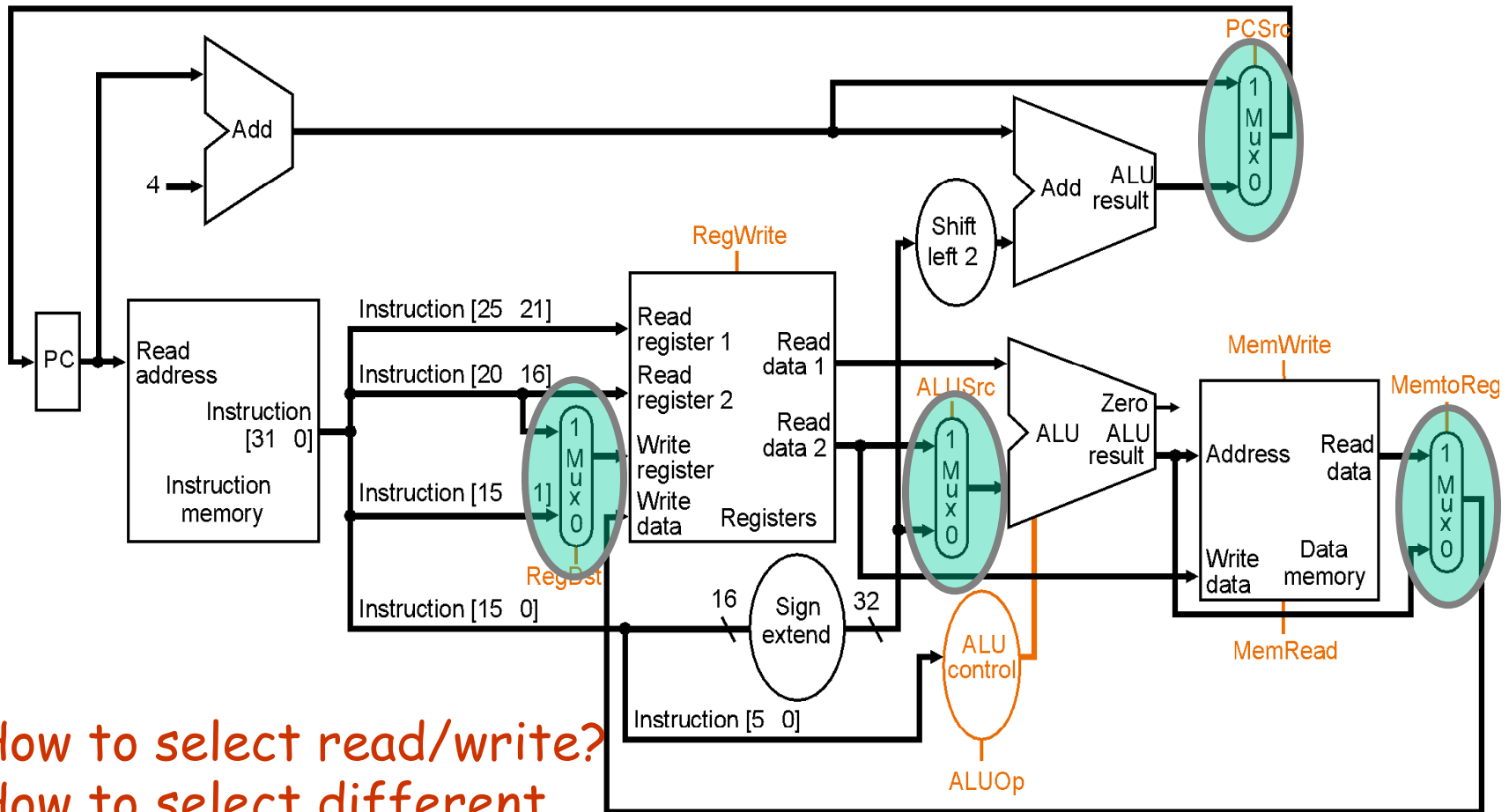


Combining Datapaths (cont.)

- Merging R-type datapath with load/store datapath using multiplexer



All Together: Single Cycle Datapath



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



Practice

- Question:
 - Could we swap rs, rt, & rd bits to have easier datapath design?
 - In other words, can we remove RegDst MUX?



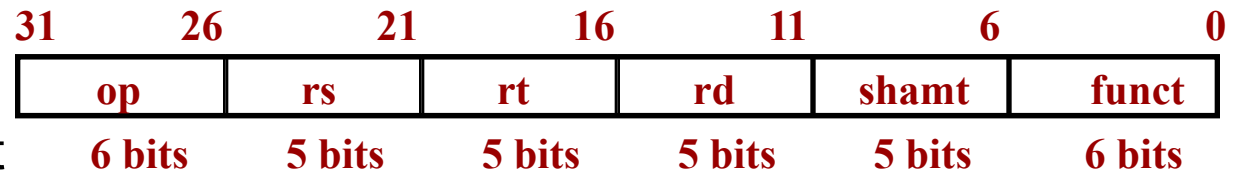
Practice: Hint

- R-type
 - rs & rt: read index bits
 - rd: write index bits

- lw:
 - rs: read index bits
 - rt: write index bits
- SW:
 - rs & rt: read index bits

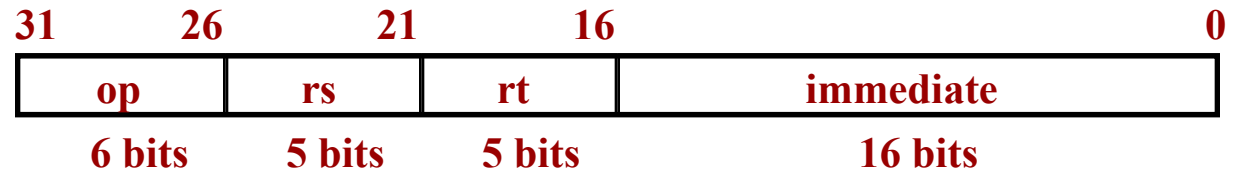
- R-TYPE

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



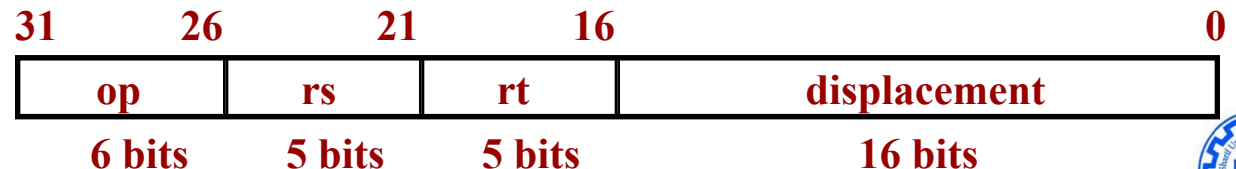
- LOAD / STORE

- *lw rt, rs, imm*
 - *sw rt, rs, imm*

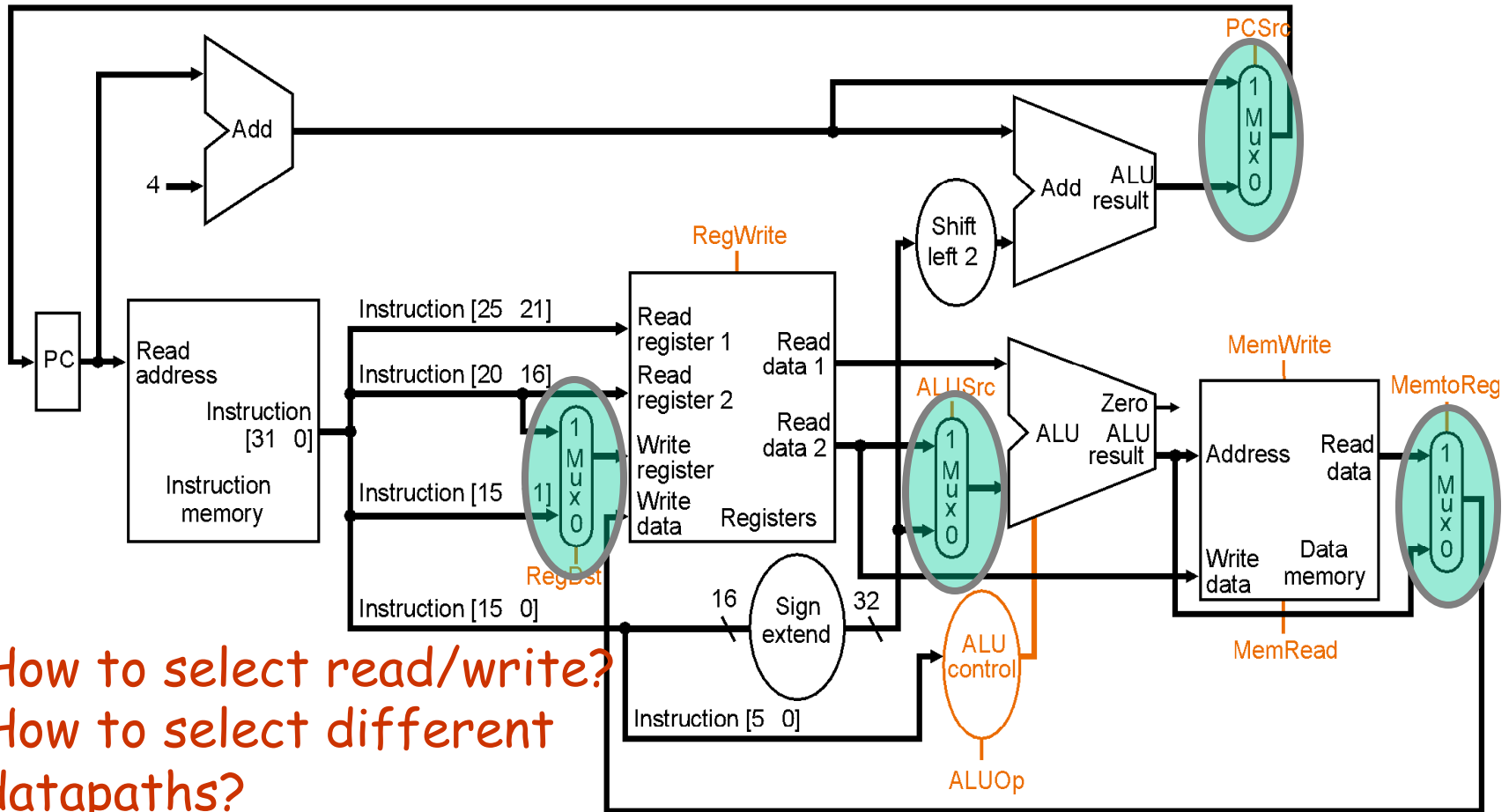


- BRANCH

- *beq rs, rt, imm*



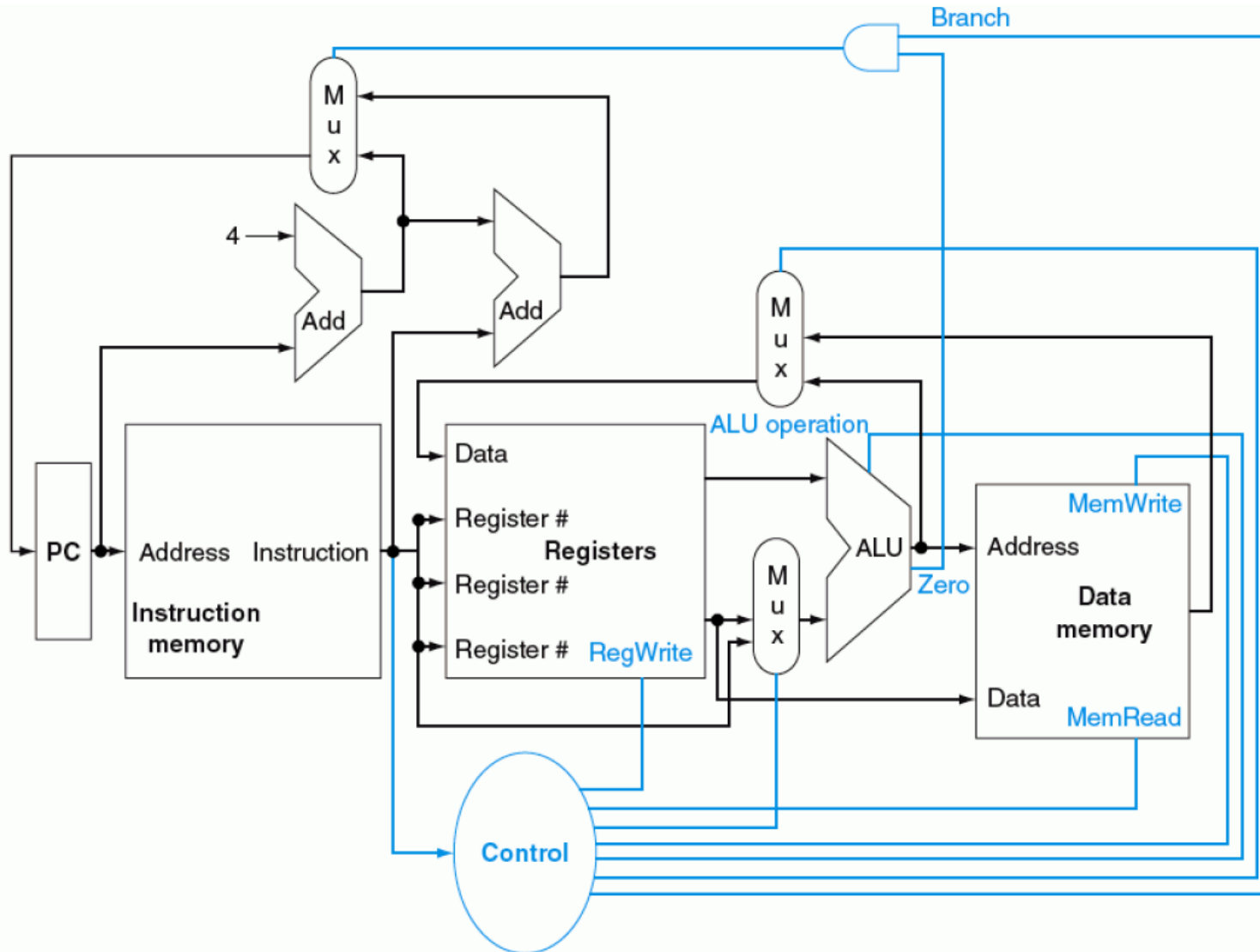
All Together: Single Cycle Datapath



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

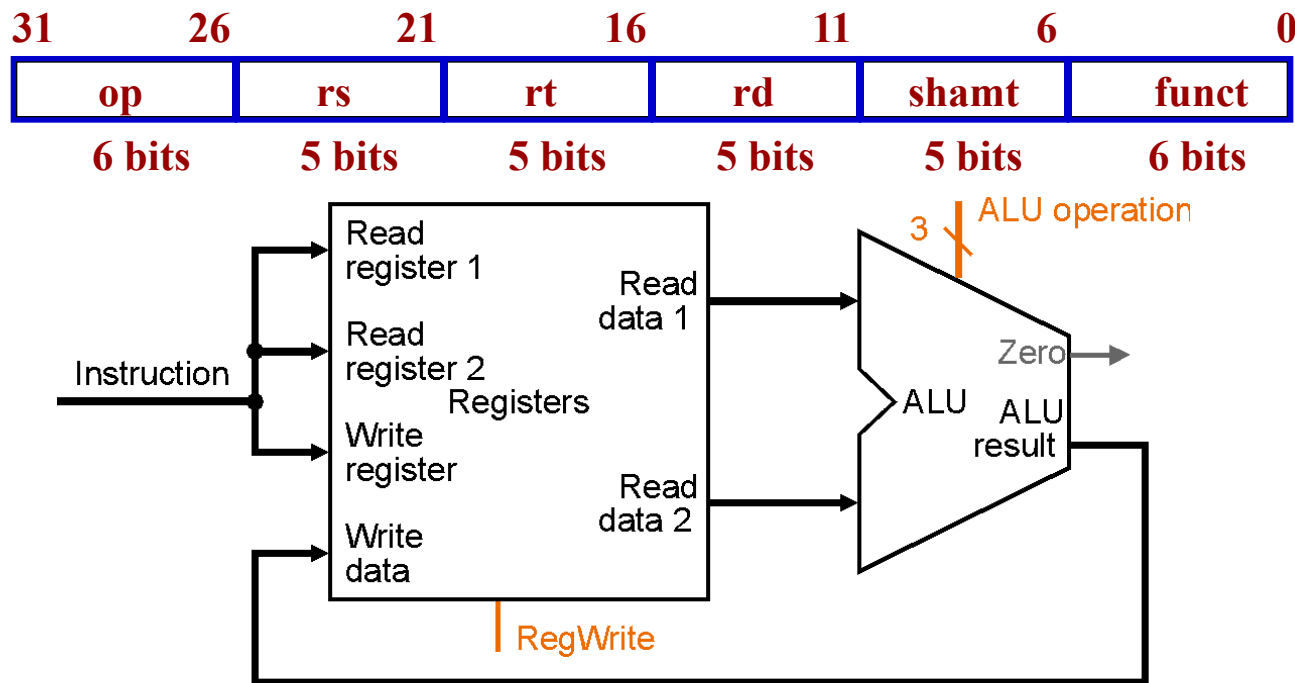


Abstract View of MIPS Implementation: Control



Datapath for Reg-Reg Operations

- $GPR[rd] \leftarrow GPR[rs] \text{ op } GPR[rt]$
 - Example: *add rd, rs, rt*
 - ALUoperation signal depends on op and funct

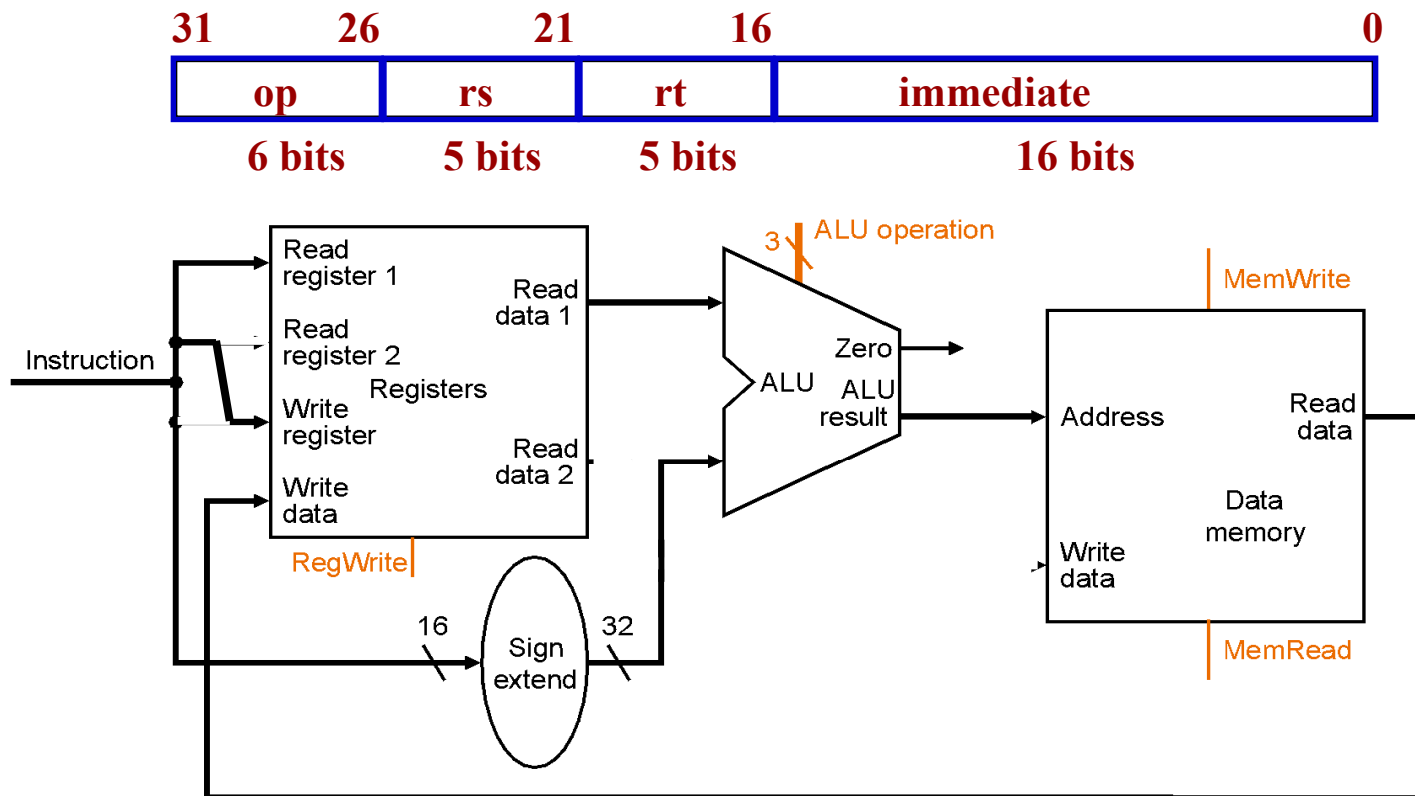


ALU operation and *RegWrite*: control logic after decoding instruction



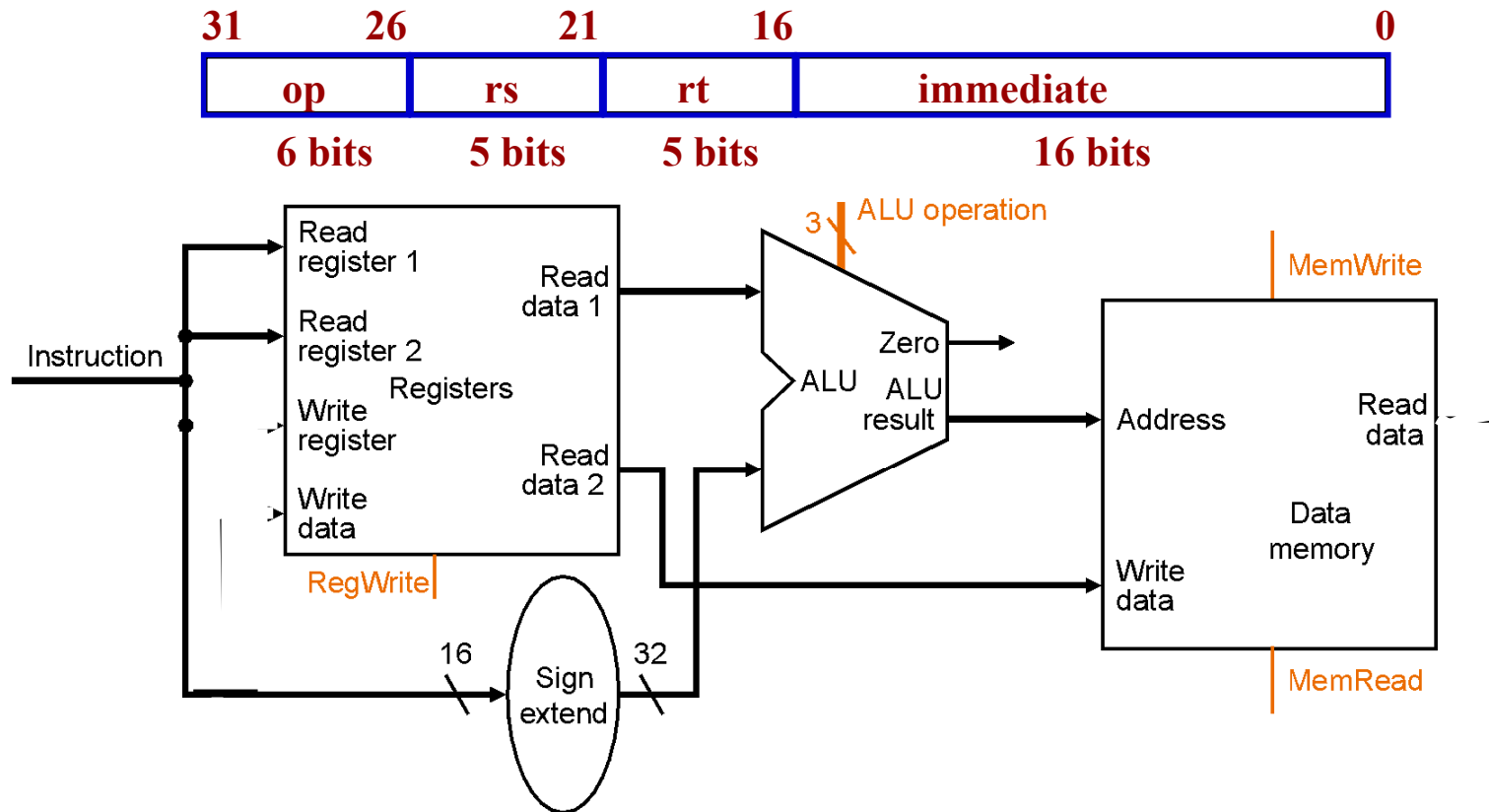
Datapath for Load Operations

- $GPR[rt] \leftarrow Mem[GPR[rs] + SignExt[imm16]]$
 - Example: *lw rt, rs, imm16*



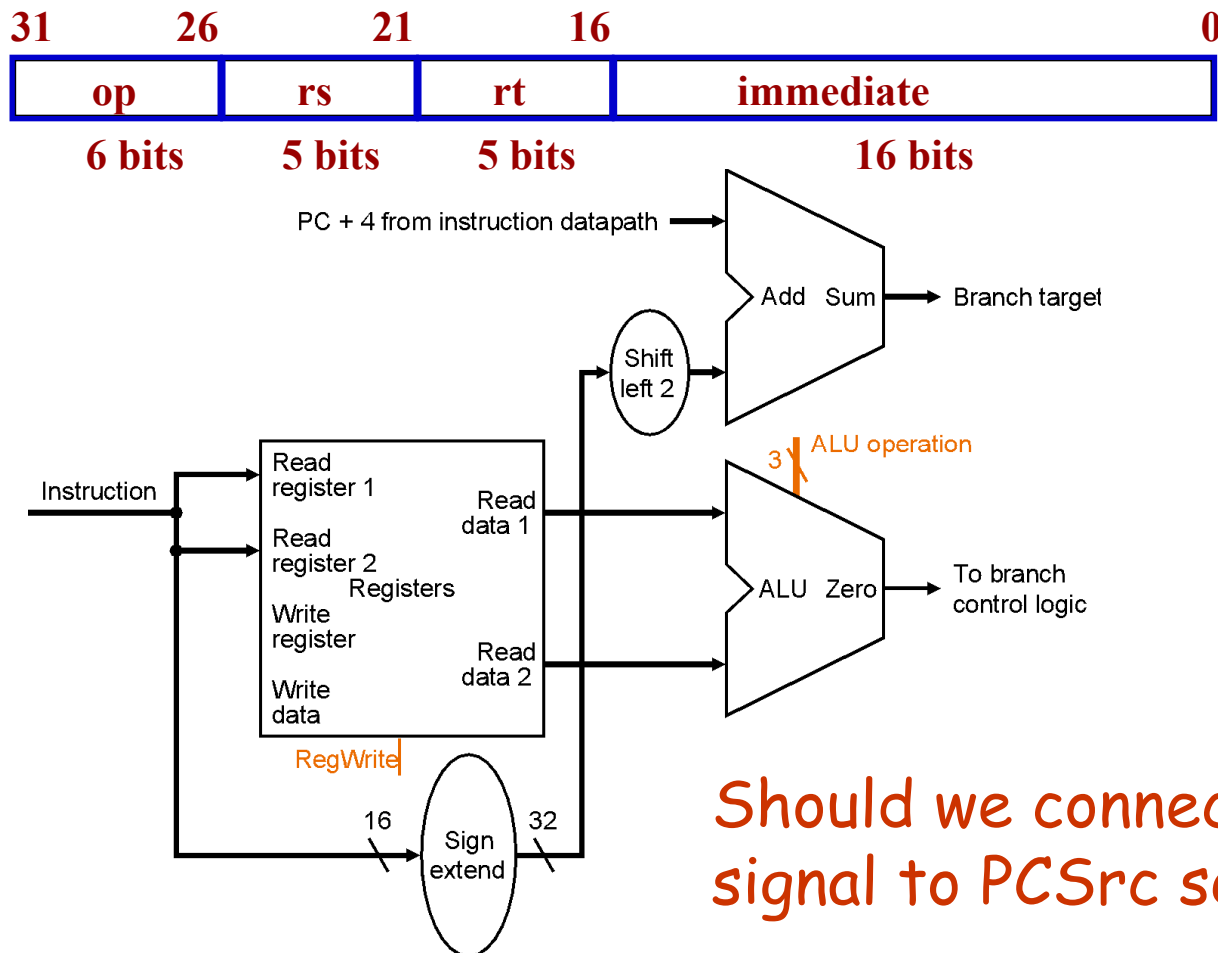
Datapath for Store Operations

- $\text{Mem}[\text{GPR}[\text{rs}] + \text{SignExt}[\text{imm16}]] \leftarrow \text{GPR}[\text{rt}]$
 - Example: *sw rt, rs, imm16*



Datapath for Branch Operations

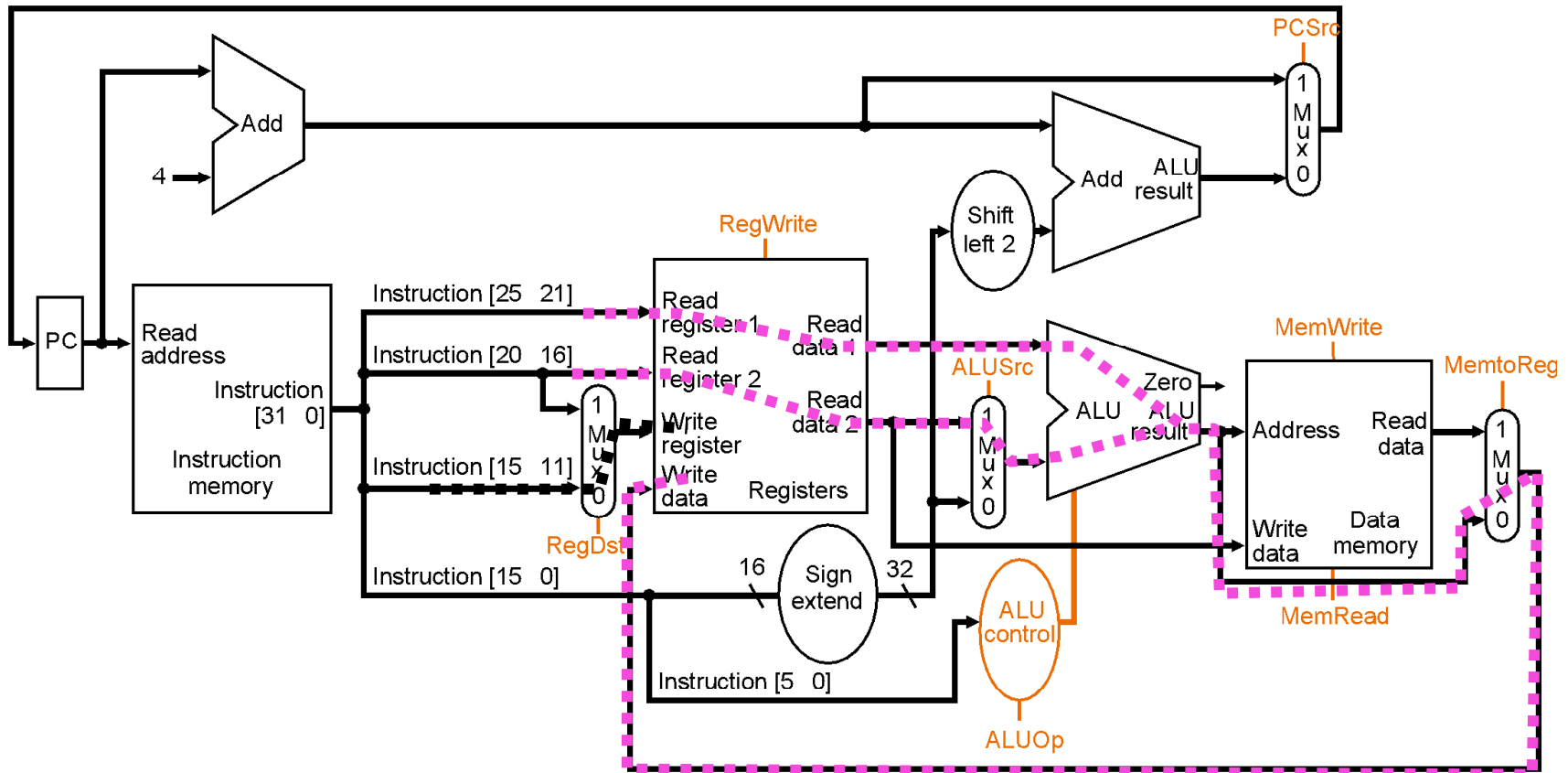
- beq rs, rt, imm16*



Should we connect "Zero" signal to PCSrc selector?



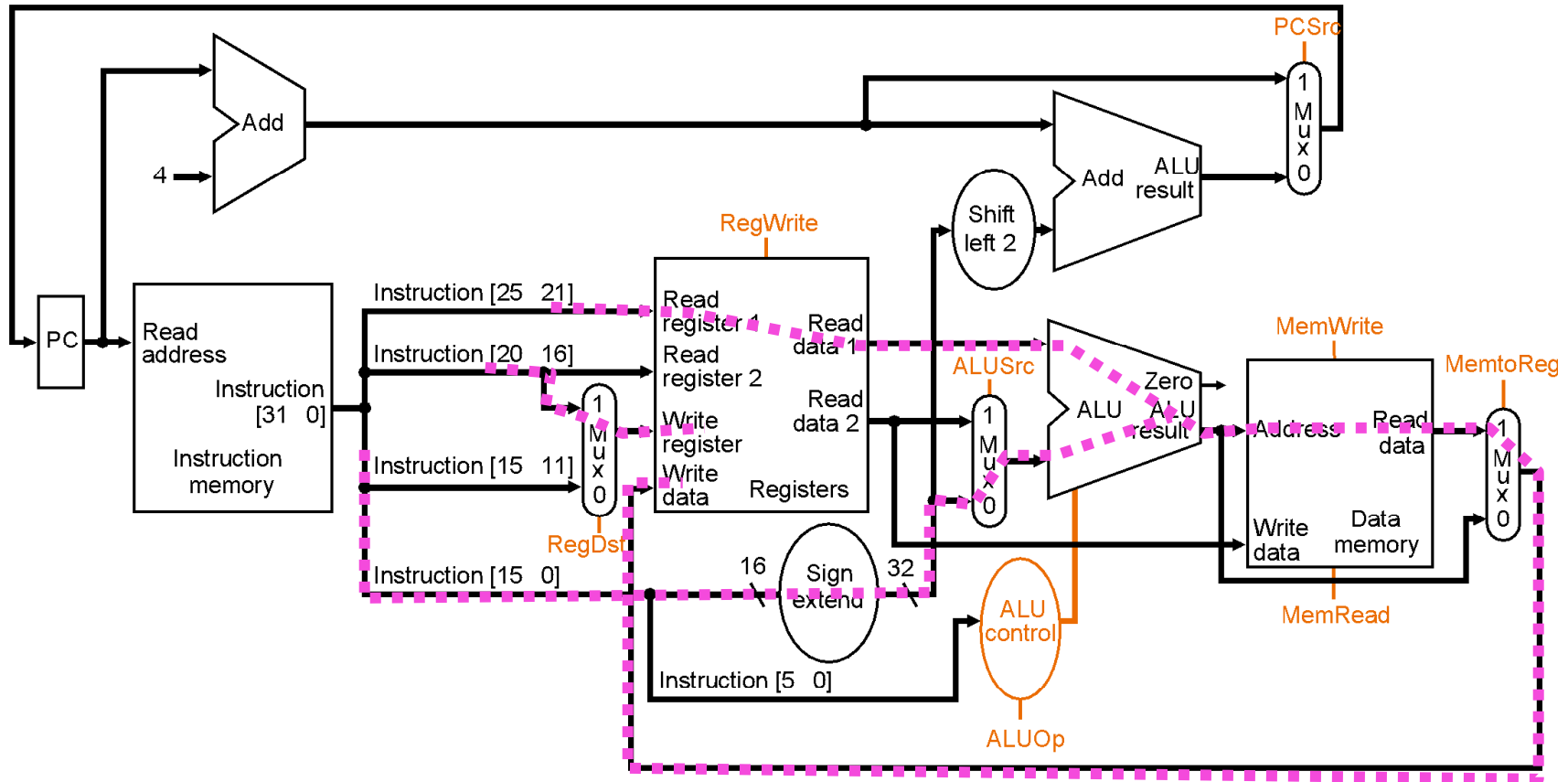
R-Format Datapath (e.g. *add*)



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



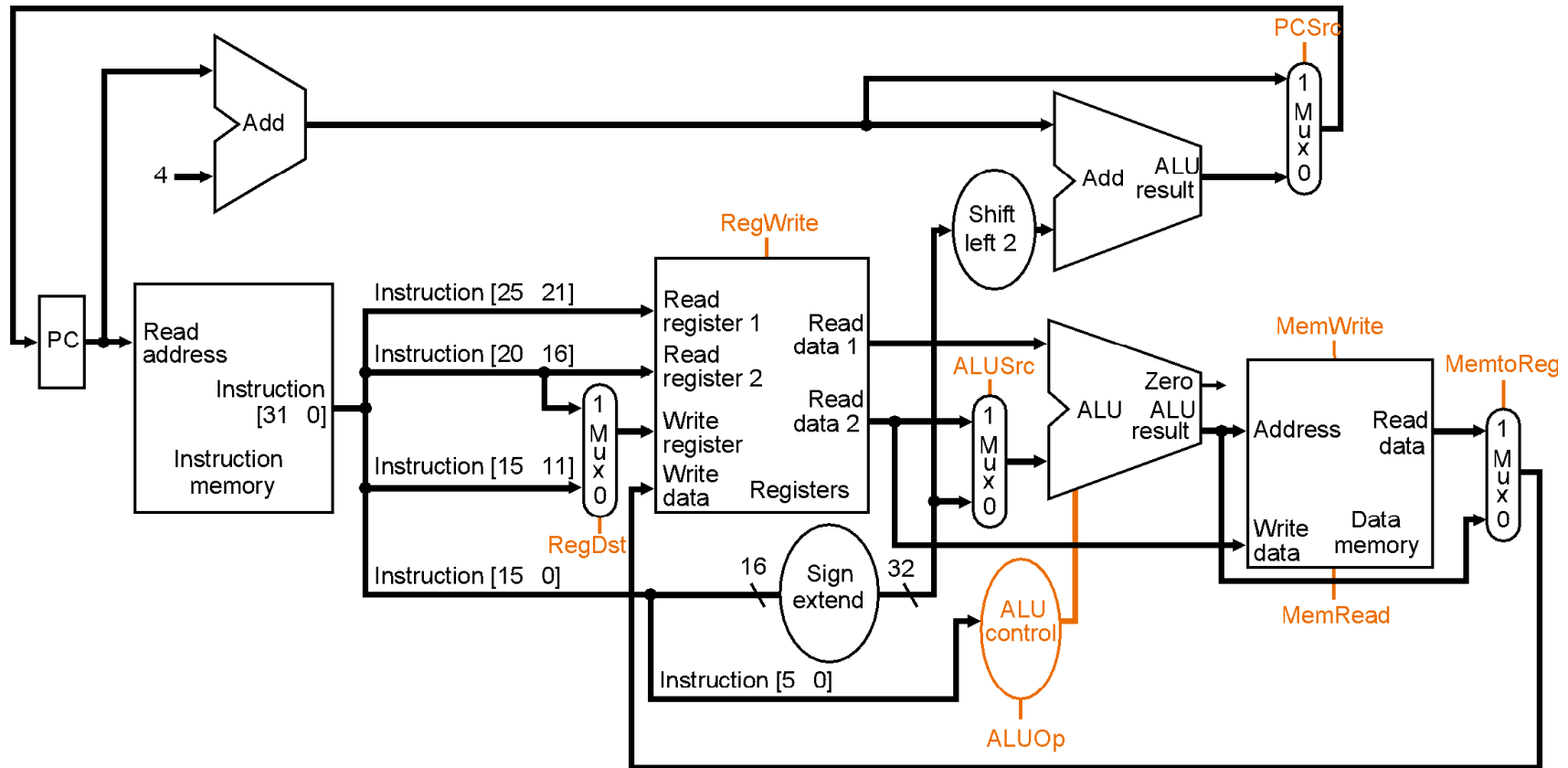
Load Datapath



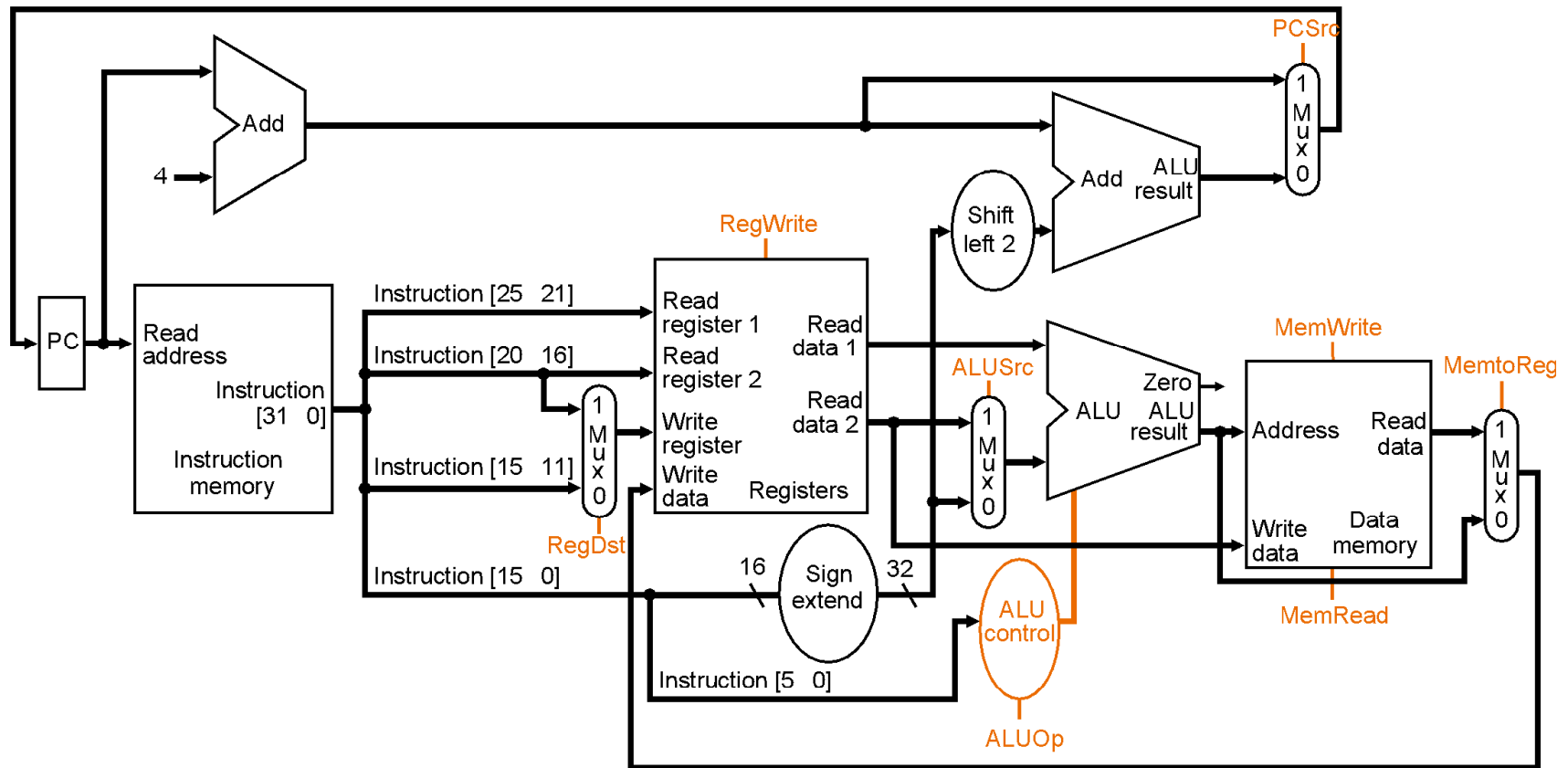
What control signals do we need for load??



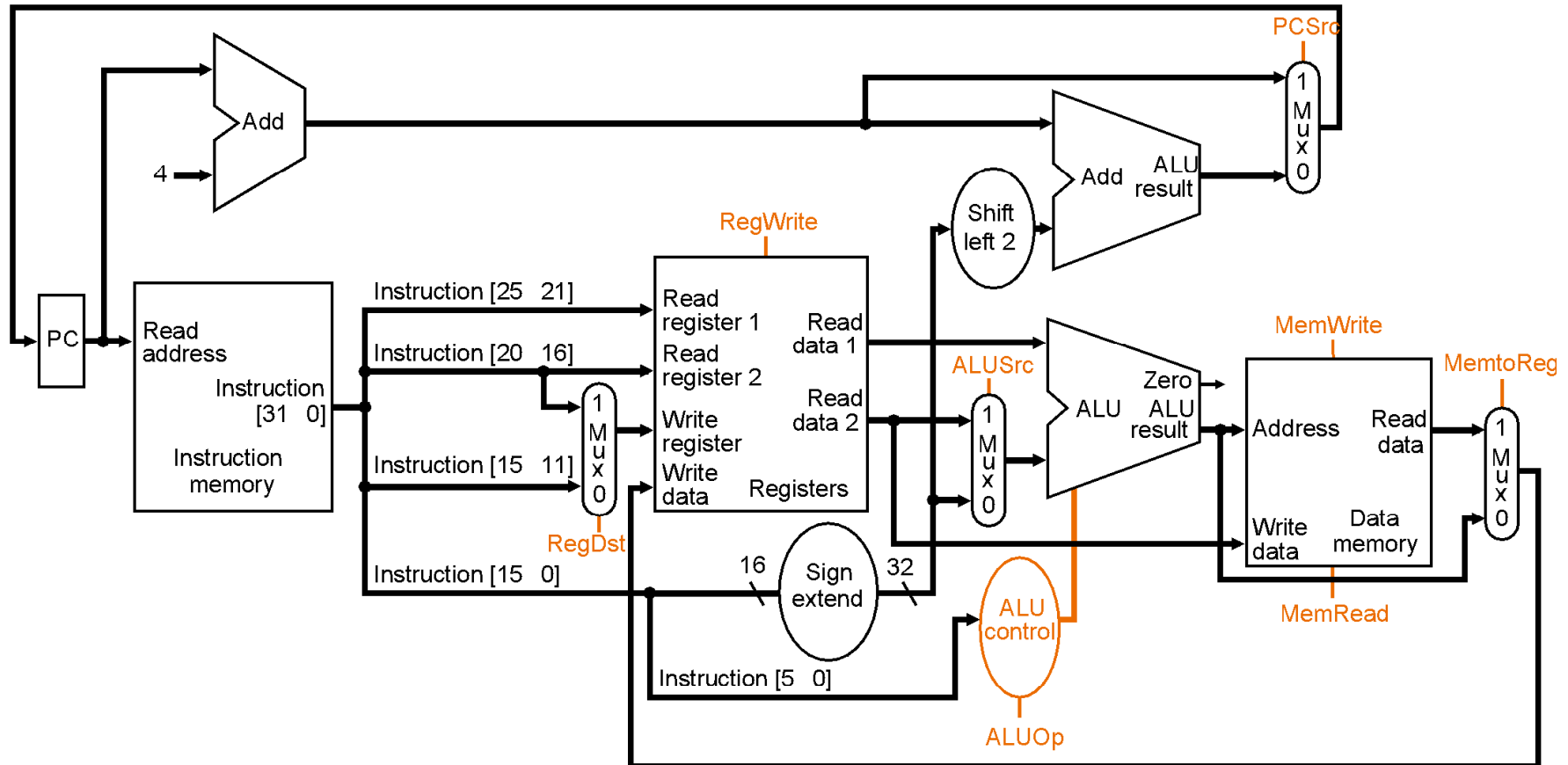
Store Datapath



beq Datapath



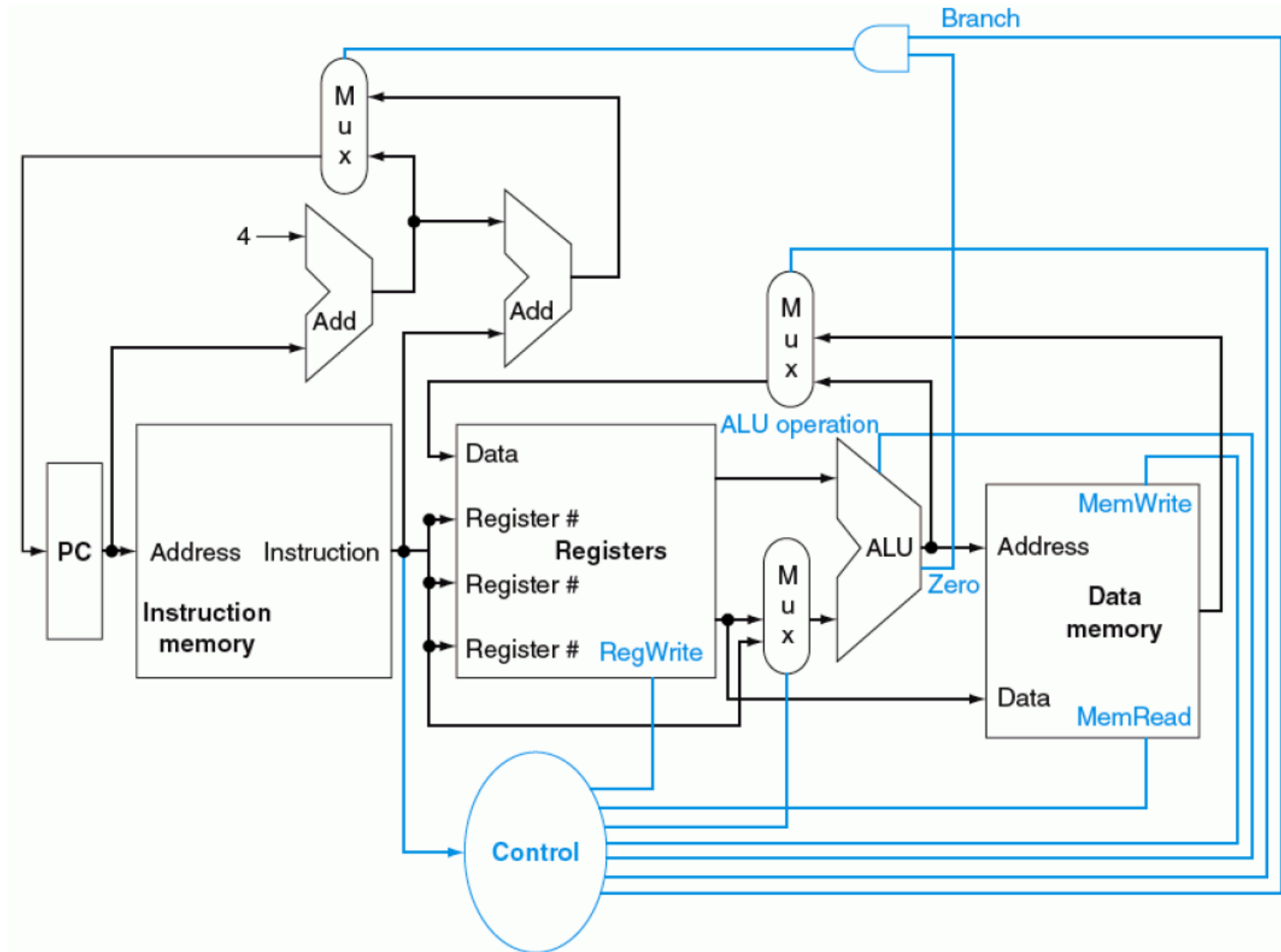
Putting it All Together: Datapath



We have everything except details for generating control signals



Next Step: MIPS Control Unit





Thanks for Your Attention!

