Computer Architecture

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

Today's Topics

- Data Path Design

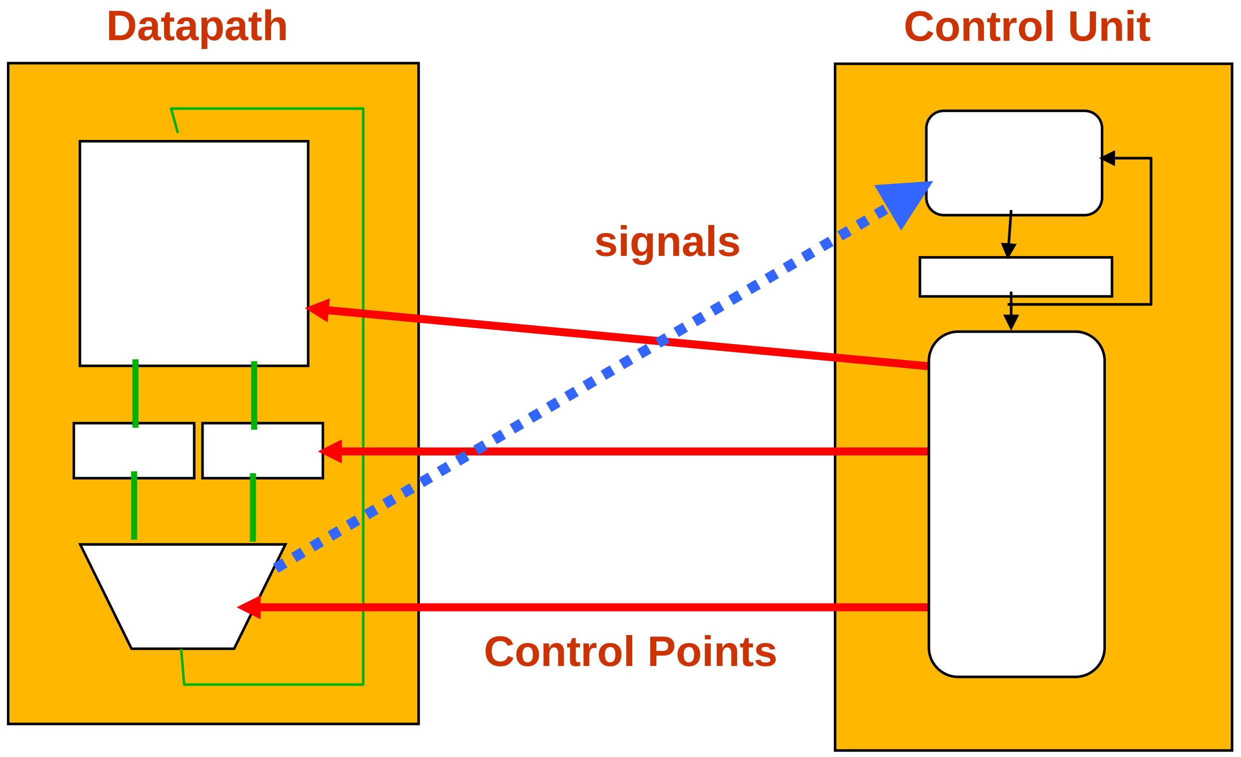
Copyright Notice

- Parts (text & figures) of this lecture adopted from:
  - Computer Organization & Design, The Hardware/Software Interface, 3rd Edition, by D. Patterson and J. Hennessey, 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, Fall 2009.
  - "Computer Arch I" handouts, by Prof. Garzarán, UIUC, Spring 2009.

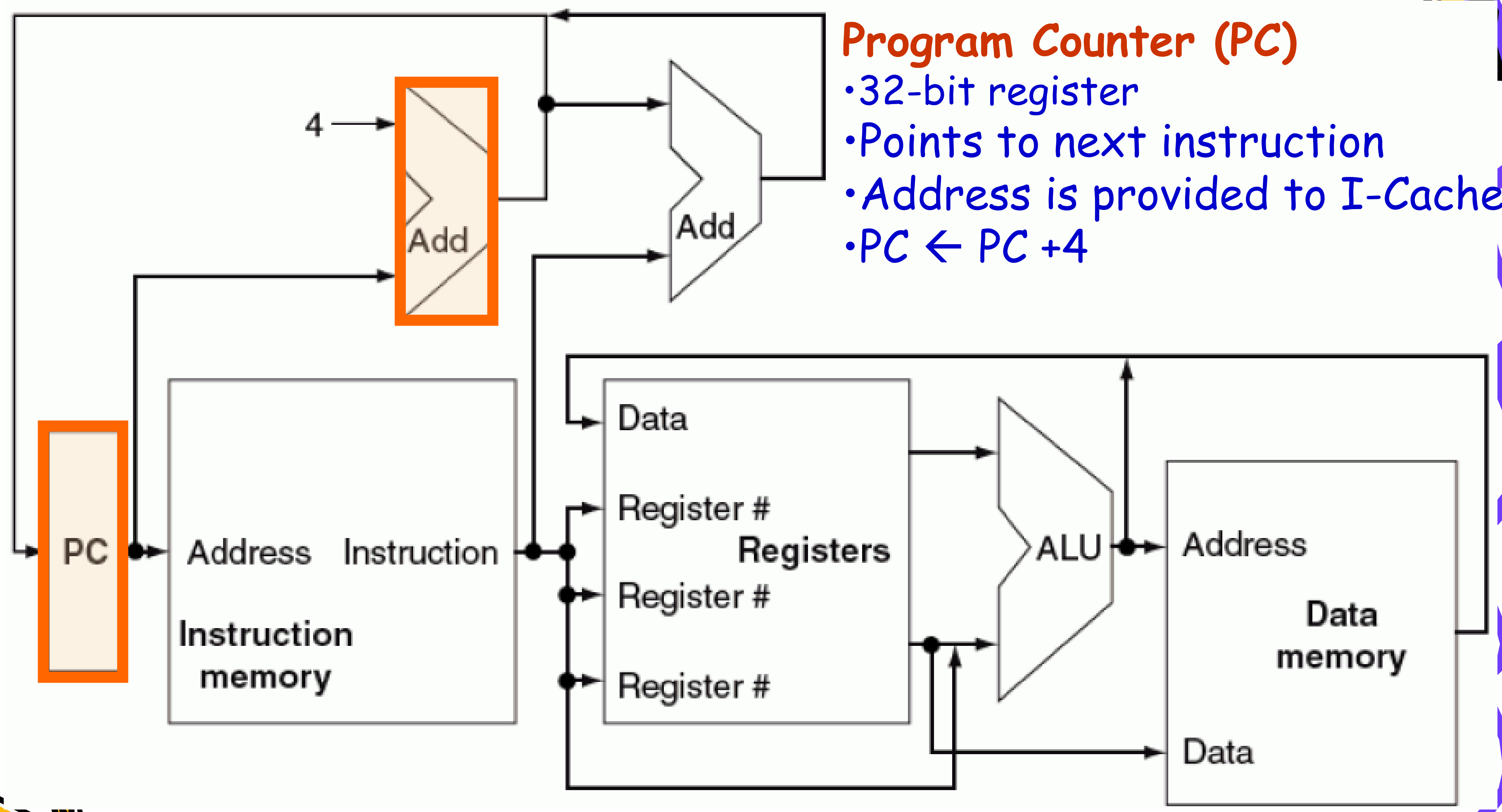
Datapath

- 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.)



Abstract View of MIPS Implementation

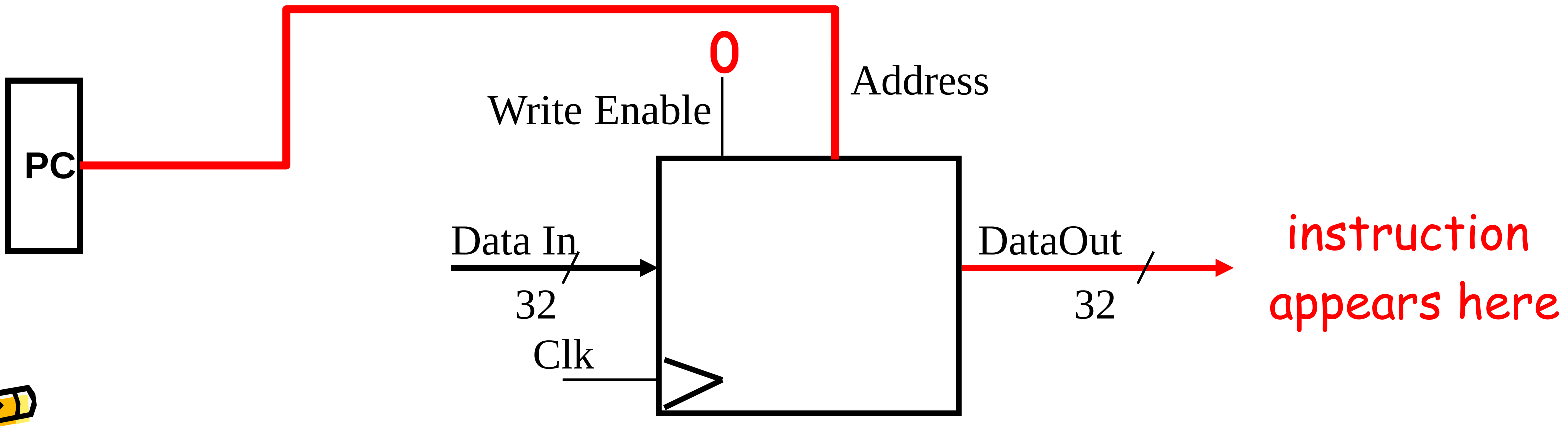


Program Counter (PC)

- 32-bit register
- Points to next instruction
- Address is provided to I-Cache
- PC ← PC +4

Instruction Fetch

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

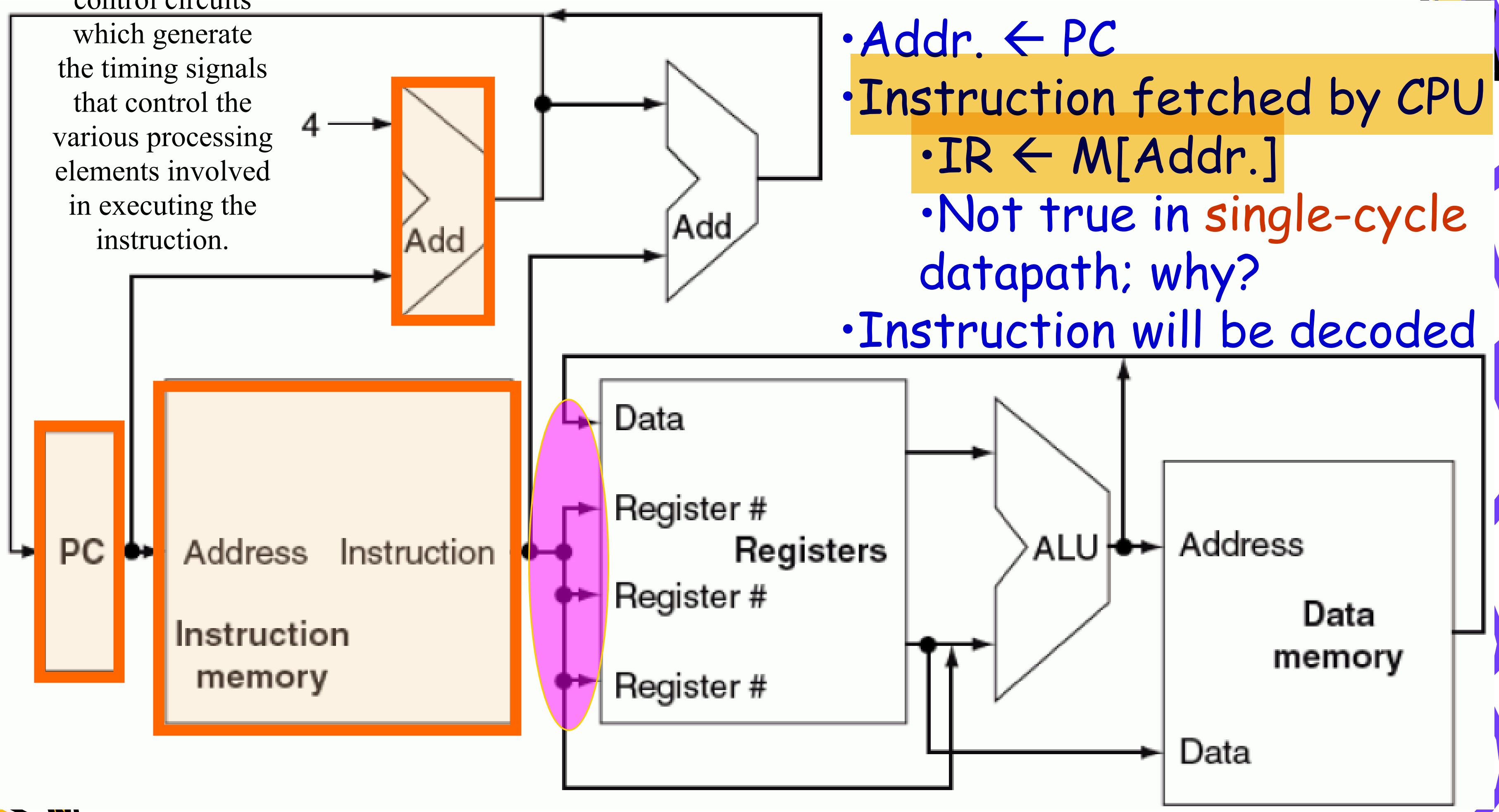


Instruction Memory (I-Cache)

چون داخل طراحی single cycle چیزی به عنوان IR نداریم

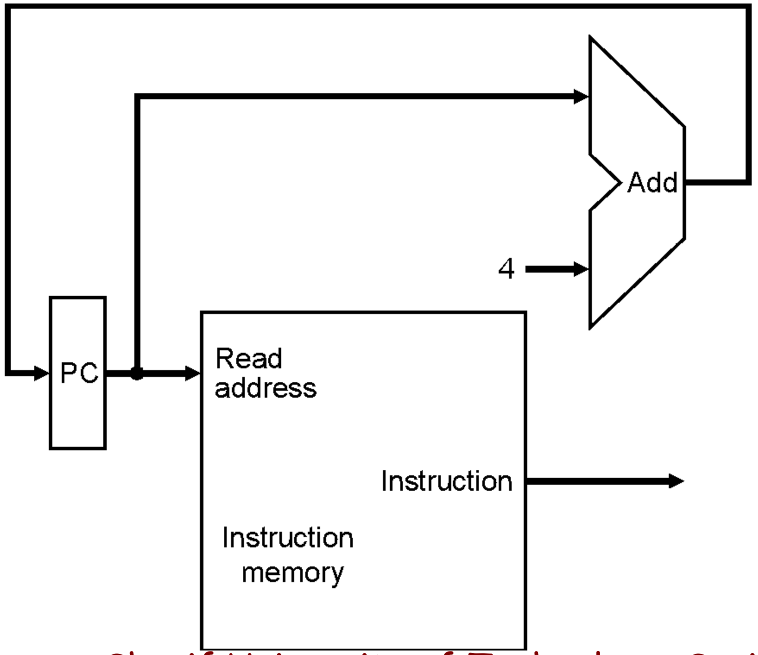
The output of the IR is available to control circuits which generate the timing signals that control the various processing elements involved in executing the instruction.

- Addr. ← PC
- Instruction fetched by CPU
    - IR ← M[Addr.]
    - Not true in single-cycle datapath; why?
- Instruction will be decoded



Instruction Fetch Unit

- Updating PC for Next Instruction
  - Sequential Code: PC ← PC + 4
  - Branch and Jump: PC ← New Address
    - we'll worry about this later

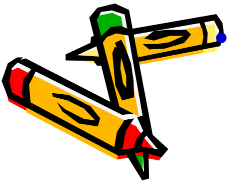


Instruction Encoding

- 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



Instruction Encoding (cont.)

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

31 26	21	16	11	6	0
op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

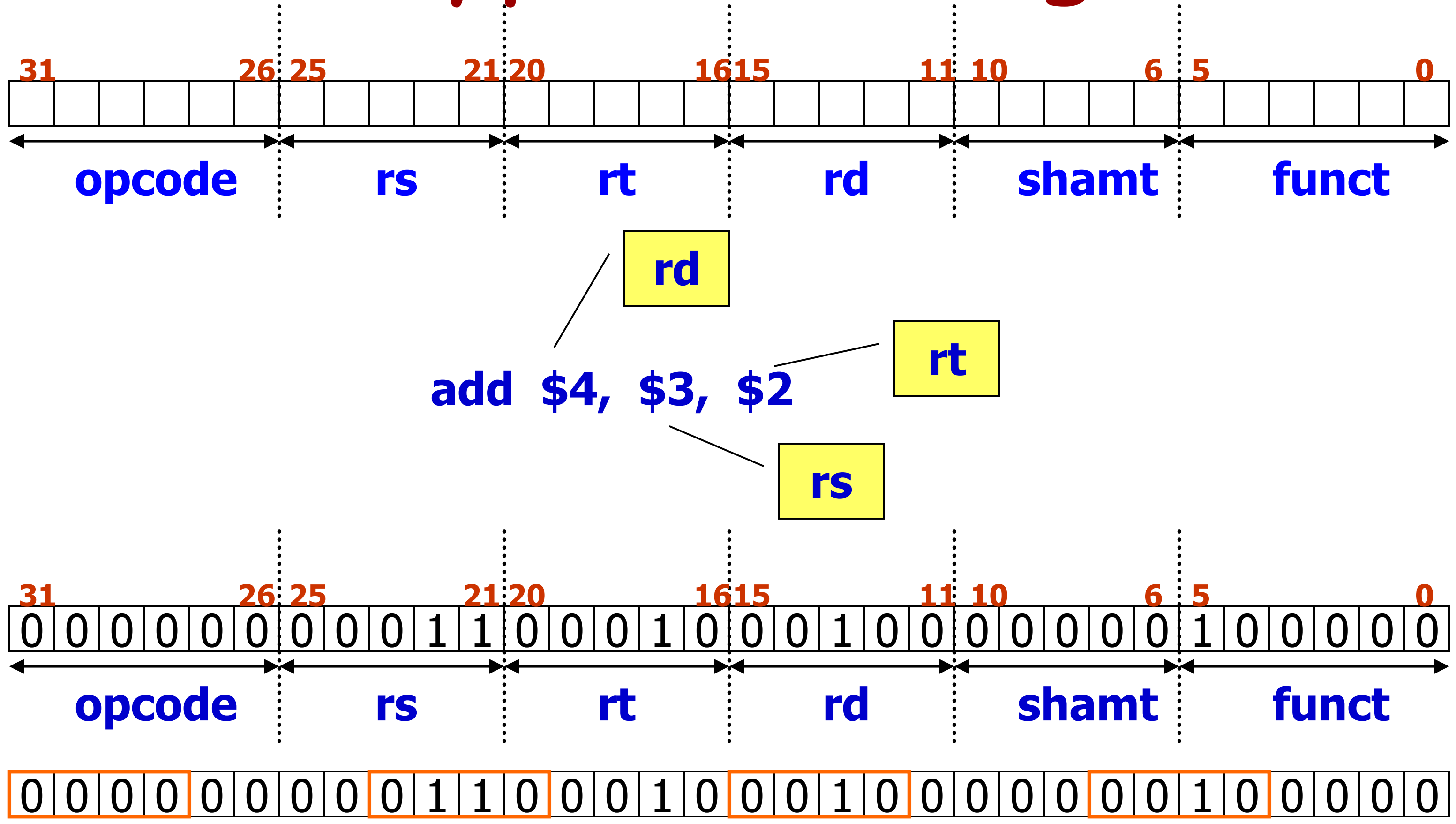
- LOAD / STORE
  - lw rt, rs, imm
  - sw rt, rs, imm

31 26	21	16	0
op	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits

- BRANCH
  - beq rs, rt, imm

31 26	21	16	0
op	rs	rt	displacement
6 bits	5 bits	5 bits	16 bits

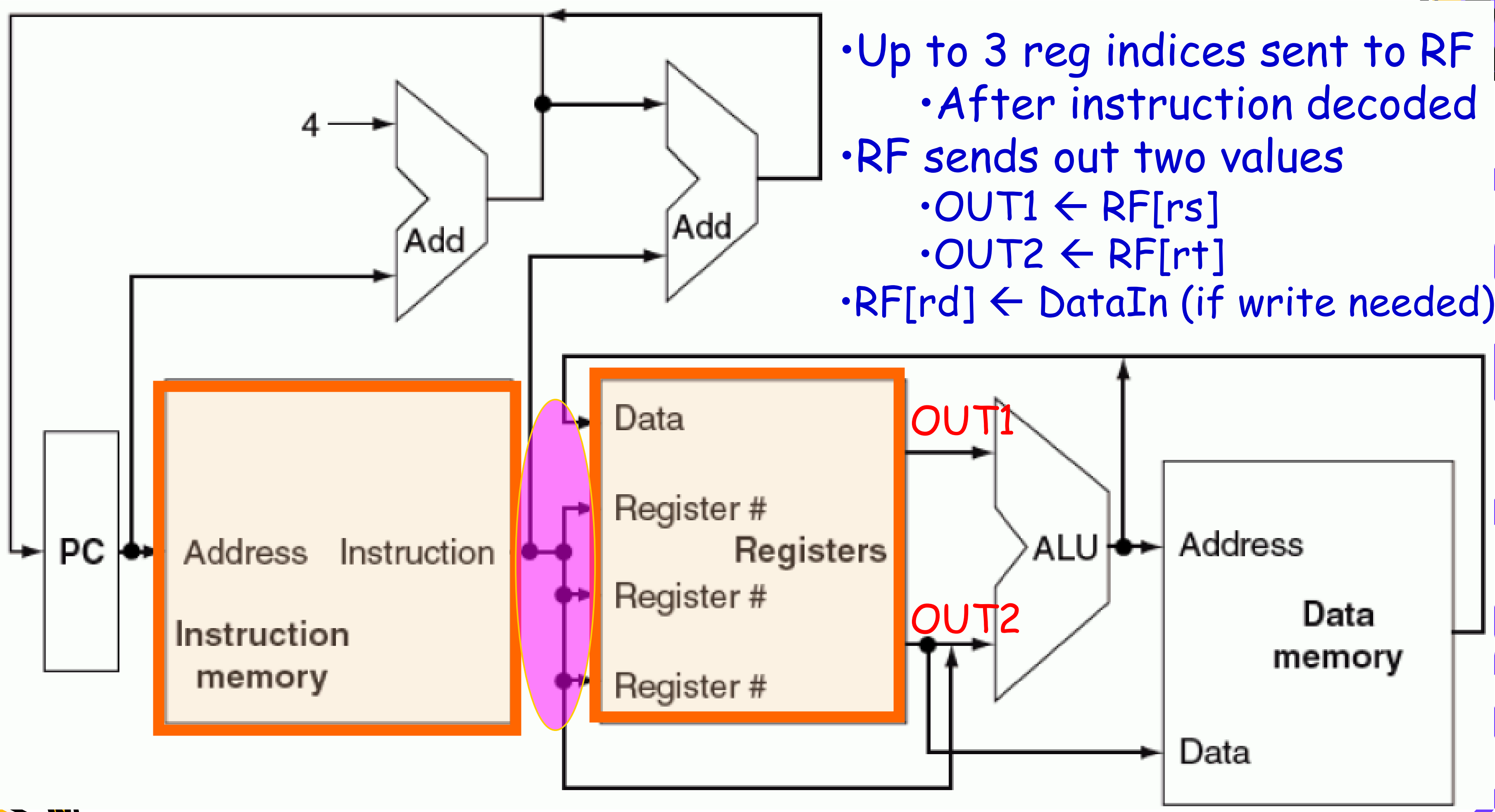
R-Type Encoding



Encoding = 0x00622020

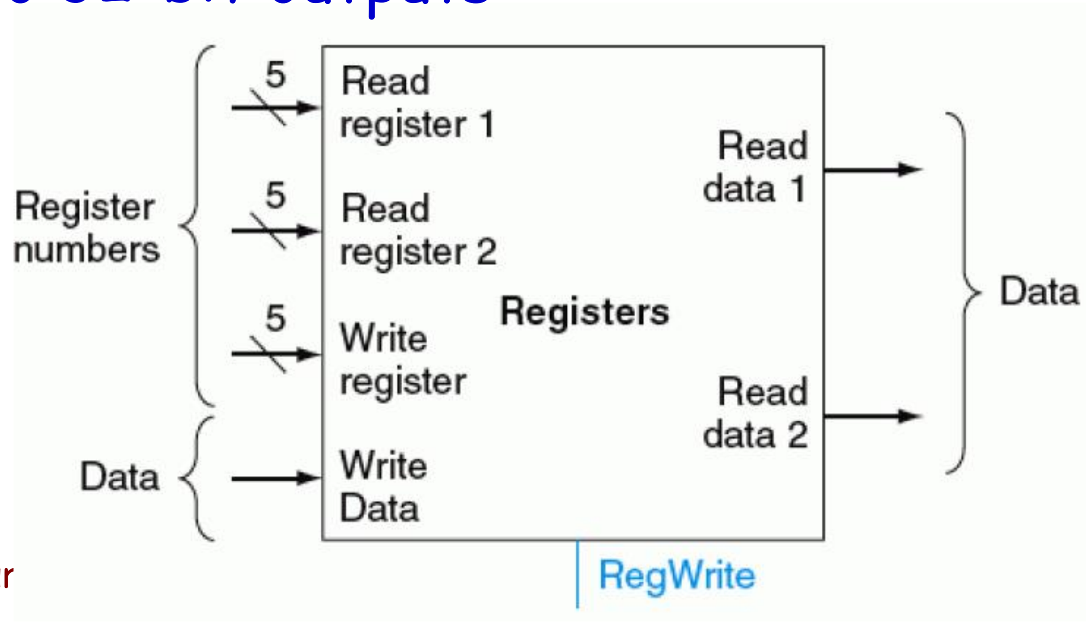
Register File

- Up to 3 reg indices sent to RF
    - After instruction decoded
- RF sends out two values
    - OUT1 ← RF[rs]
    - OUT2 ← RF[rt]
- RF[rd] ← DataIn (if write needed)



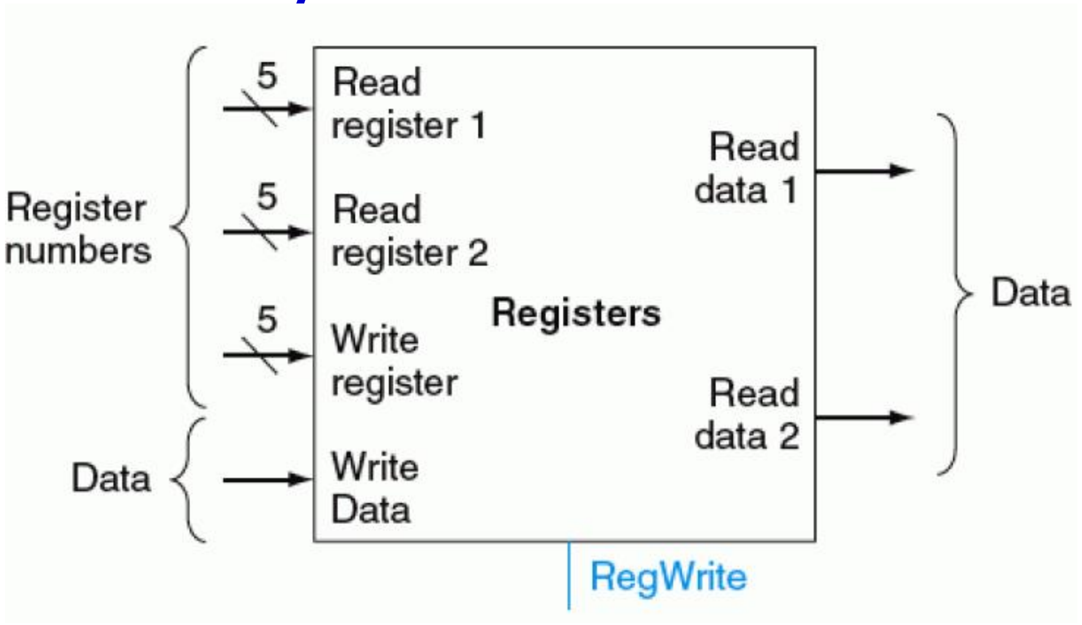
Datapath Elements

- 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.)

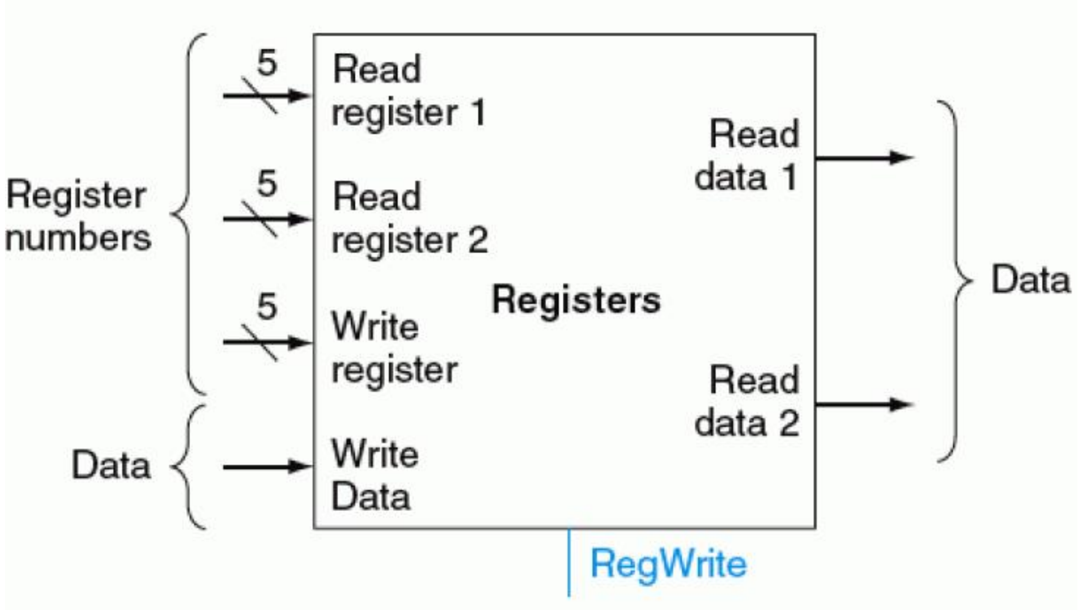
- Question 1:
  - How read & write can be accomplished in one clock cycle without data contention?



Datapath Elements (cont.)

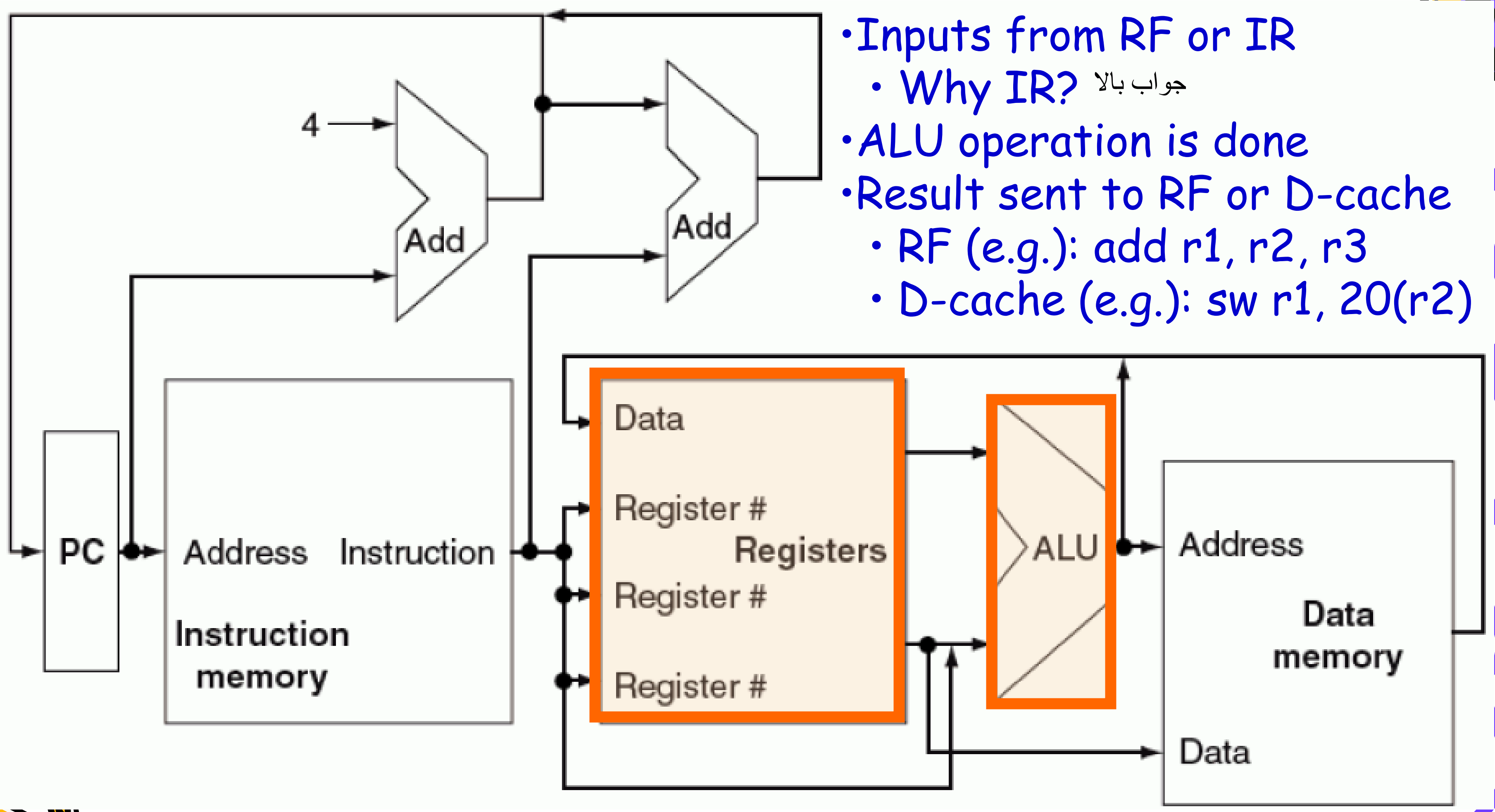
- Question 2:
  - How two simultaneous read operations possible?

به ازای هر ورودی Read Register یک mux قرار دارد که سیگنال select آن برای خواندن از هر رجیستر از ورودی های Read Register می آید



Mosttemporaries(exceptIR)areupdatedonevery cycle, so no write control is required (always write)
این موضوع برای multi cycle data path درست است.

ALU



- Inputs from RF or IR
  - Why IR? جواب بالا
- ALU operation is done
- Result sent to RF or D-cache
  - RF (e.g.): add r1, r2, r3
  - D-cache (e.g.): sw r1, 20(r2)

Reminder: Addressing Modes

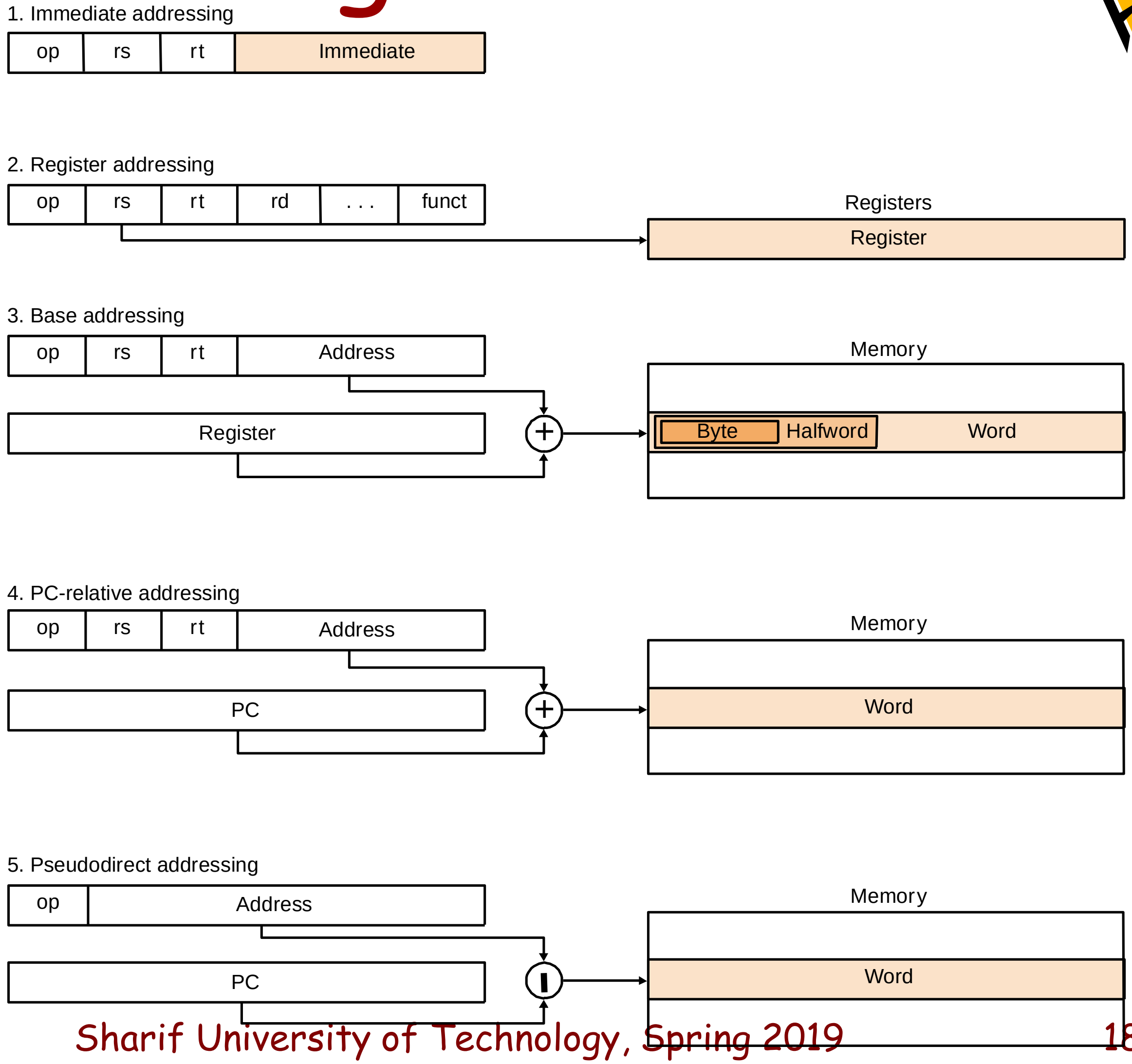
Operand is constant

Operand is in register

lb $t0, 48($s0)

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

j Label



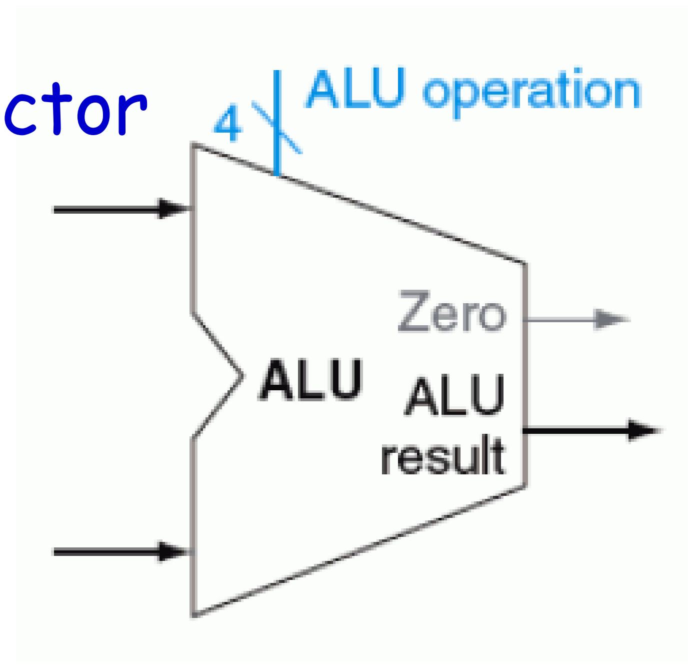
Datapath Elements (cont.)

از خروجی zero برای دستورات beq و bne استفاده می شود :

Zero = 0 => $r1 != $r2

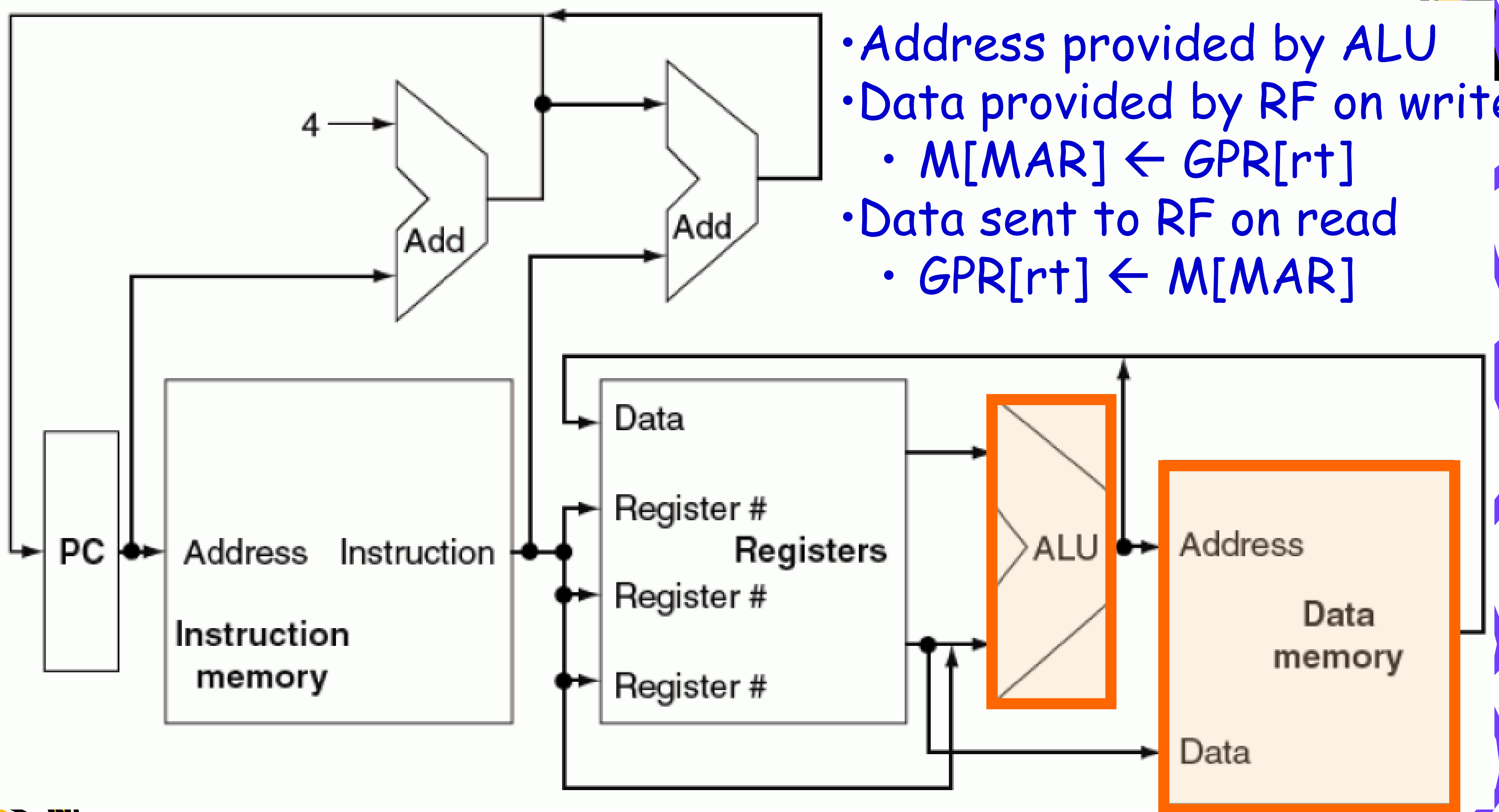
Zero = 1 => $r1 = $r2

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



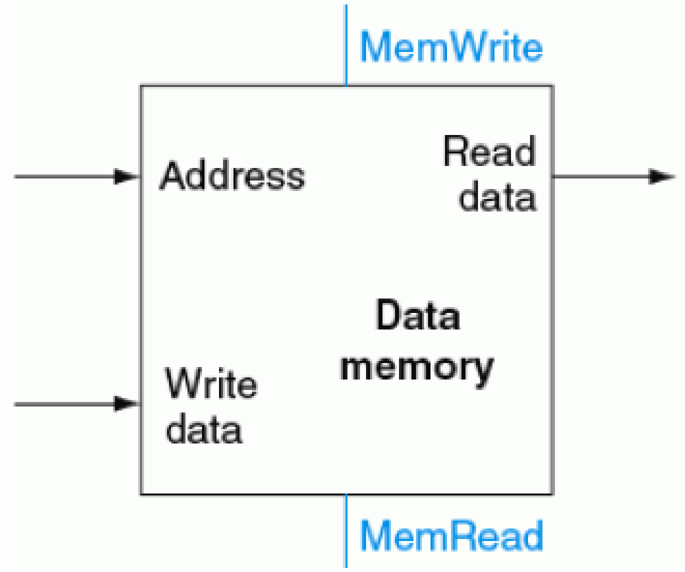
Data Memory (D-Cache)

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



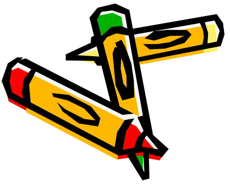
Datapath Elements (cont.)

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

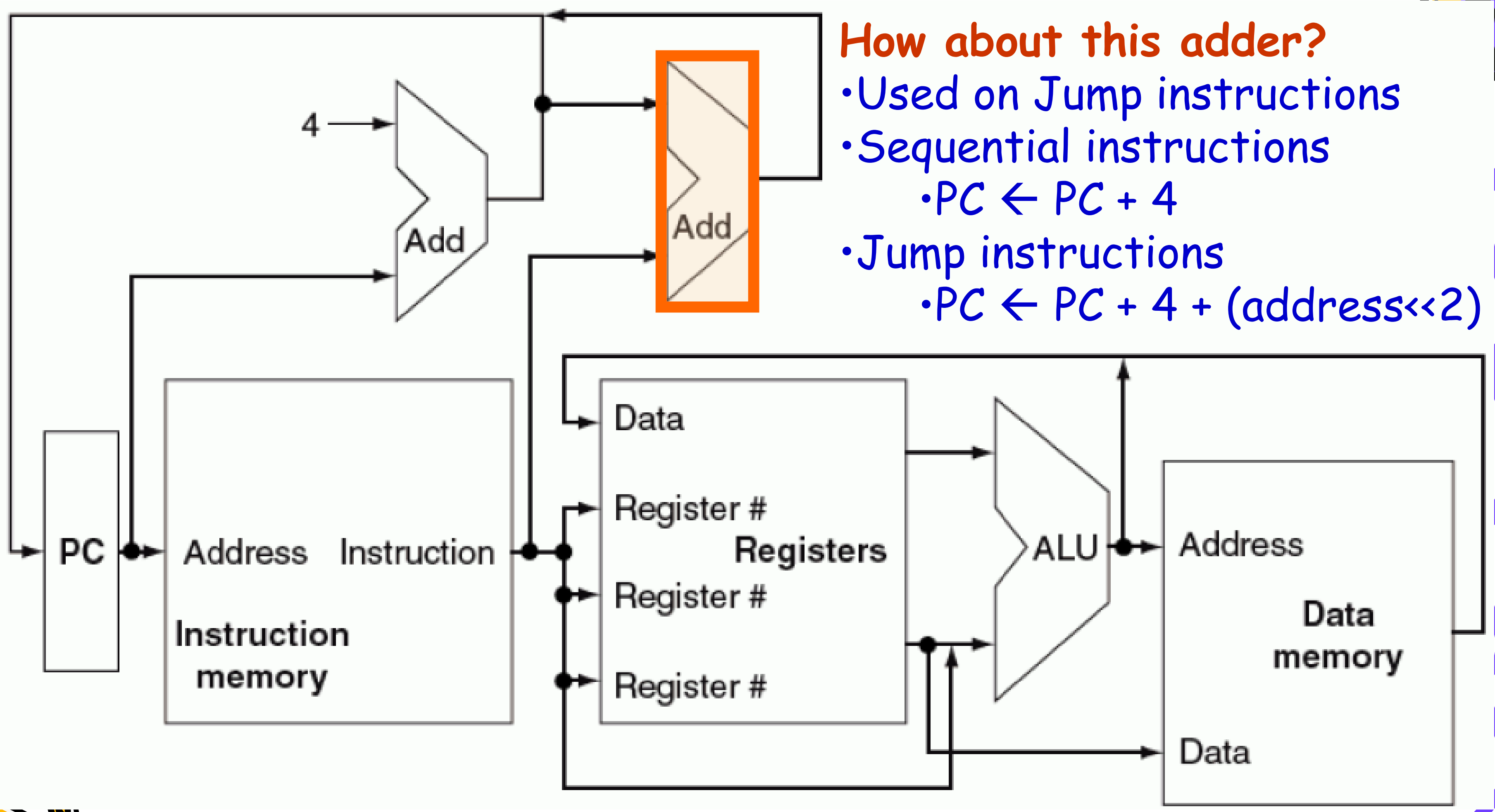


Discussion

- 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

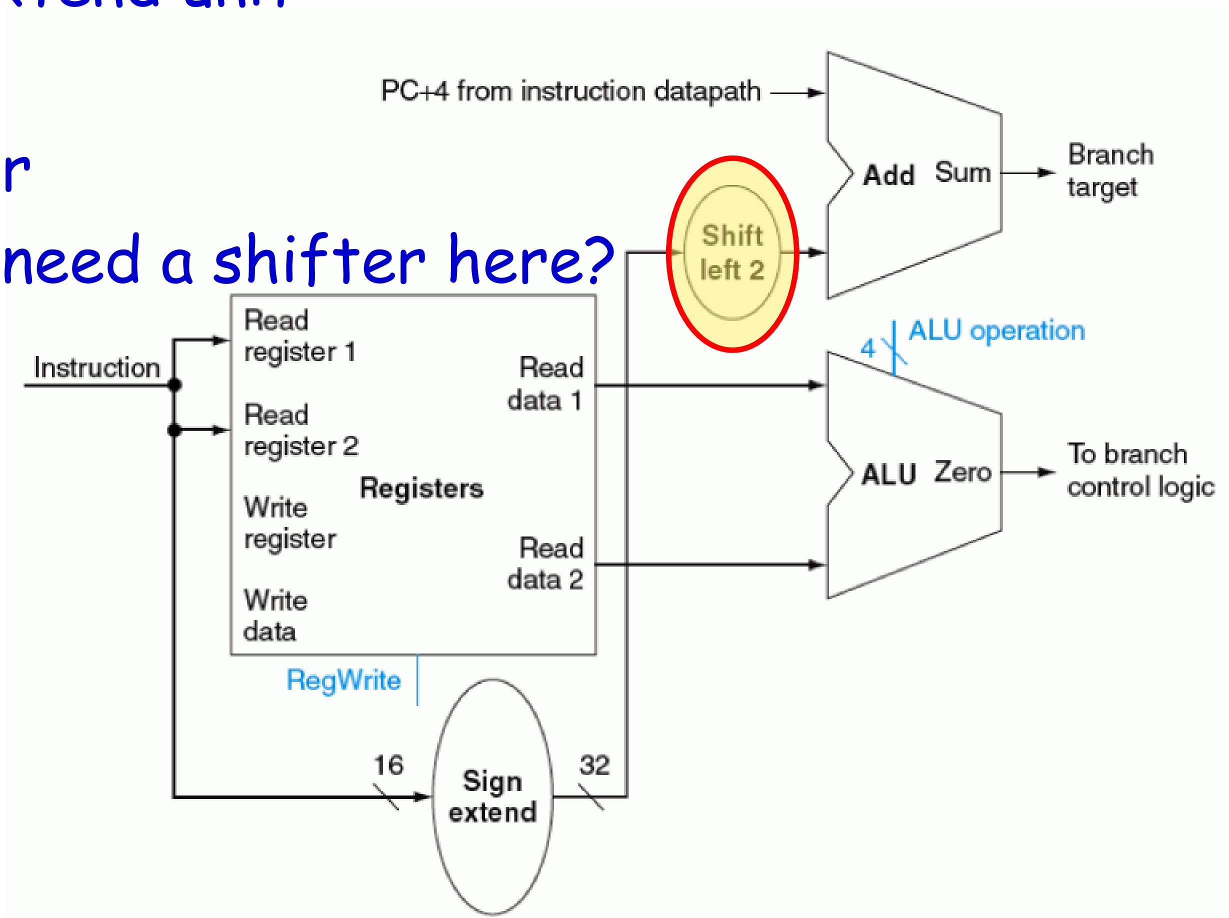


How about this adder?

- Used on Jump instructions
- Sequential instructions
    - PC ← PC + 4
- Jump instructions
    - PC ← PC + 4 + (address<<2)

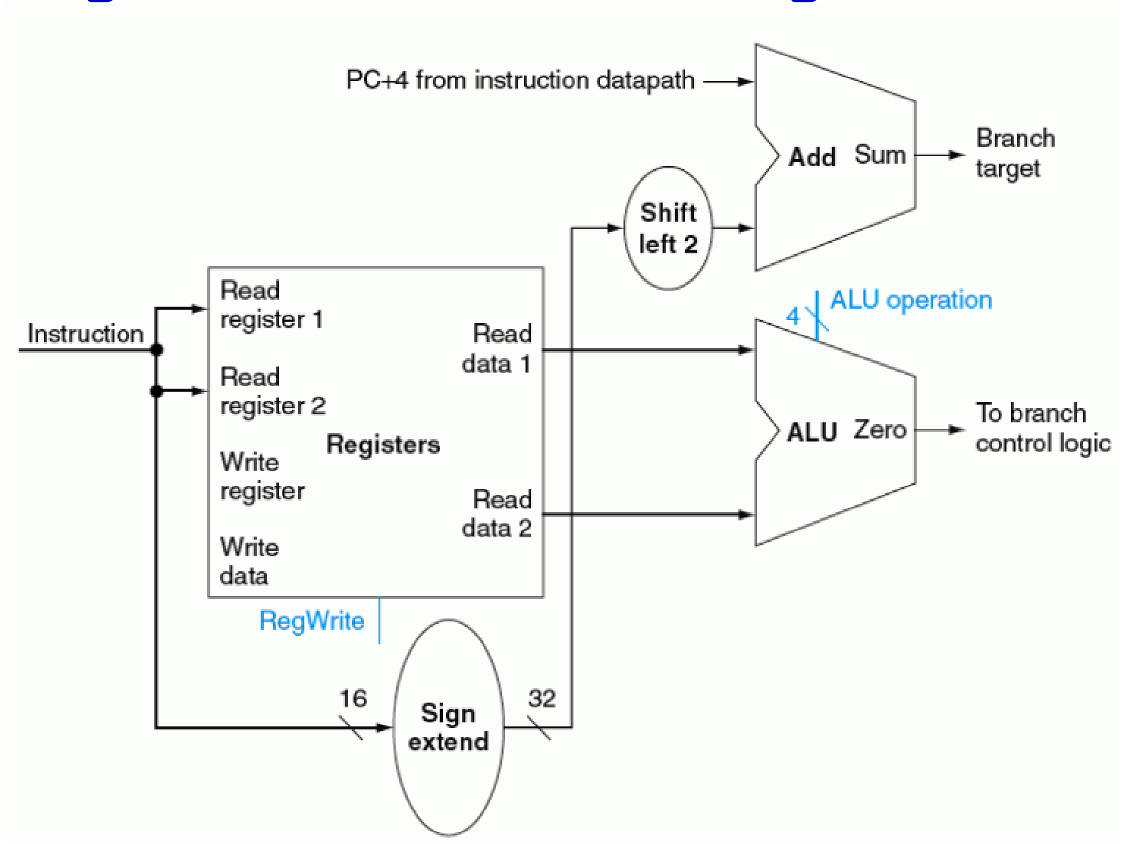
Datapath Elements (cont.)

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

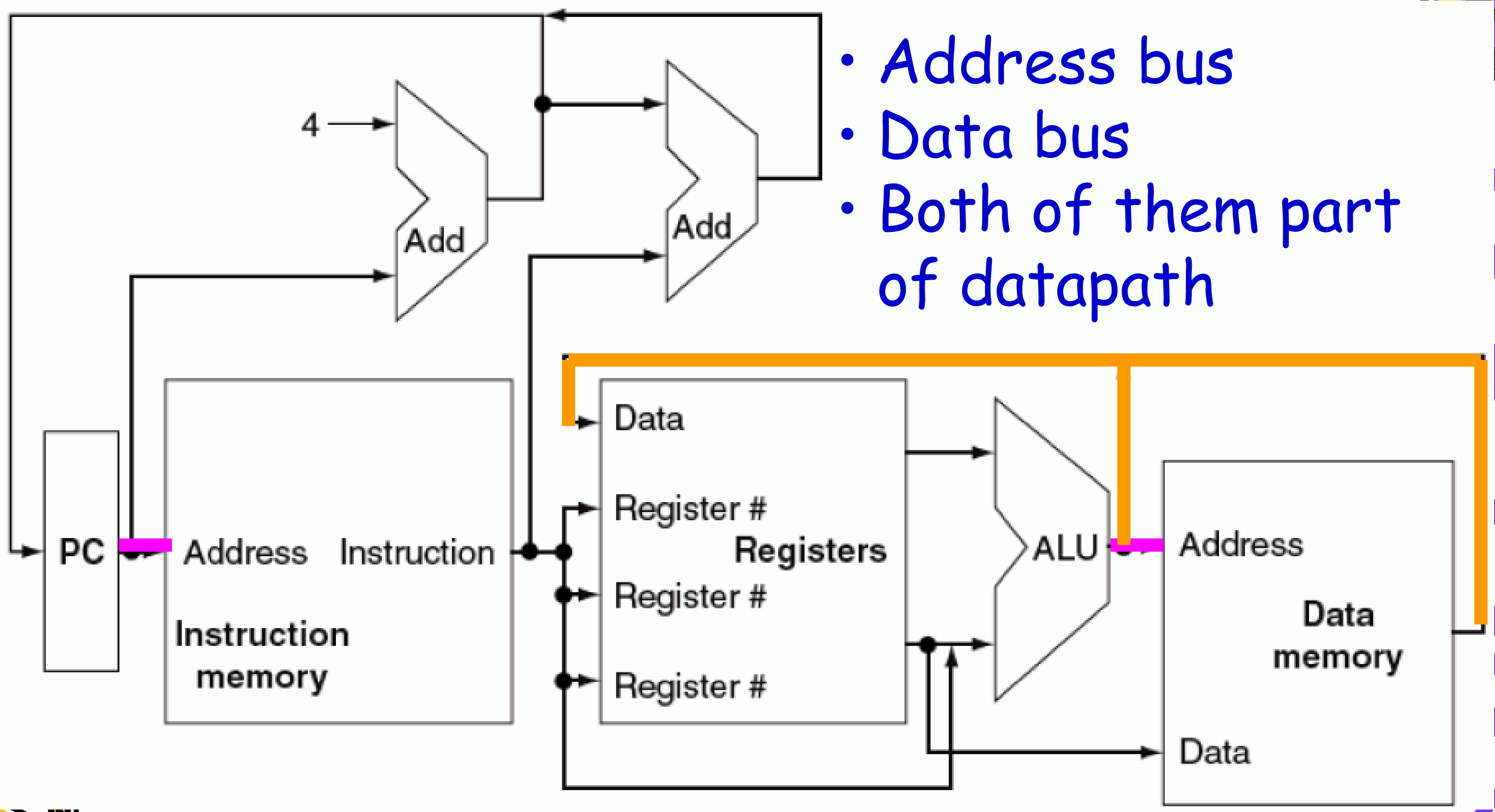


Practice

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

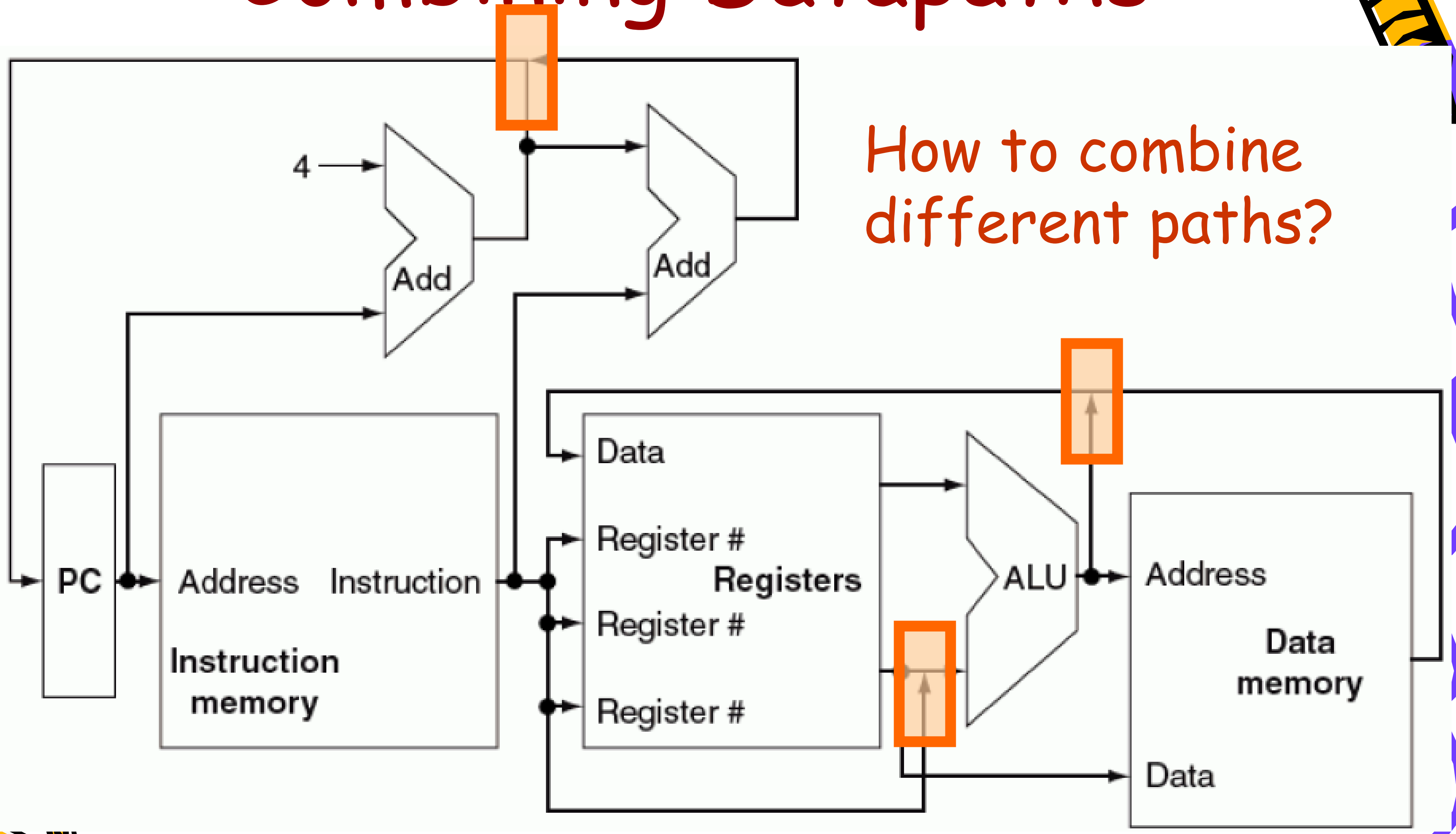


Address Bus & Data Bus



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

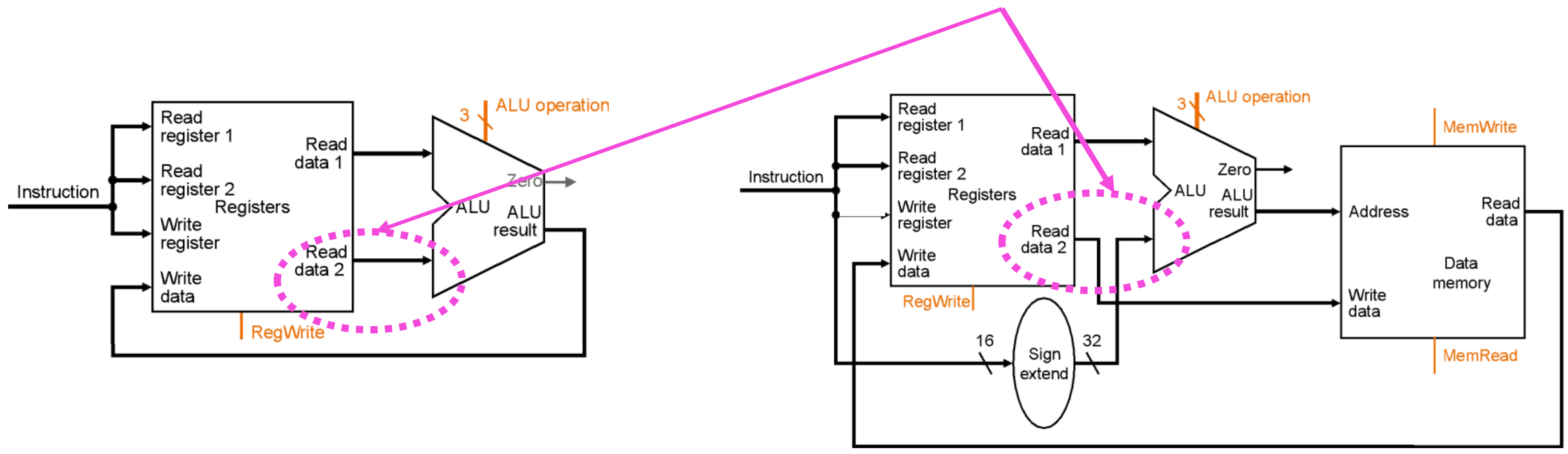
Combining Datapaths



How to combine different paths?

Combining Datapaths (cont.)

- How to have different datapaths for different instruction?

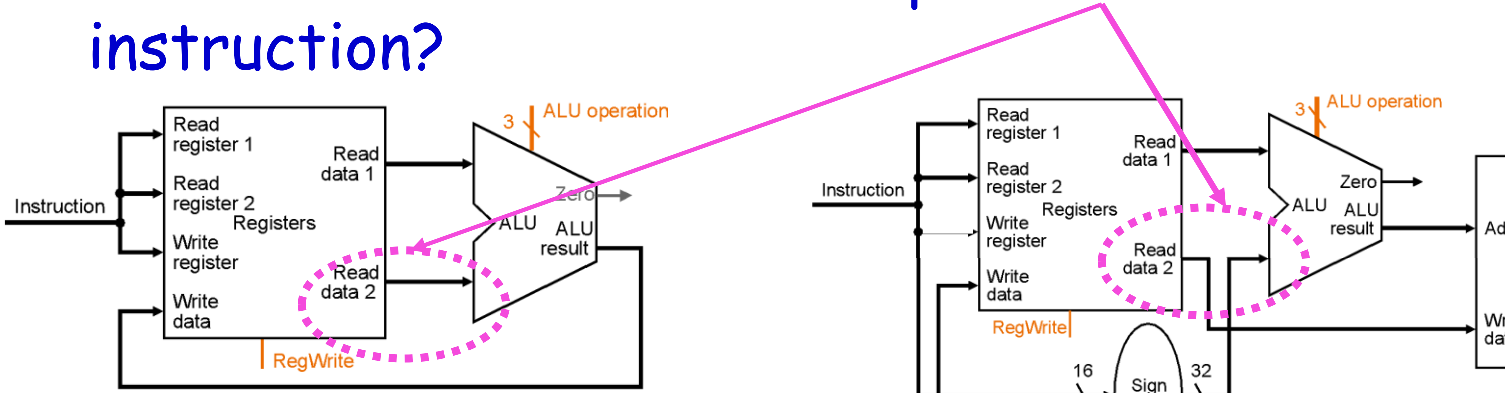


R-type

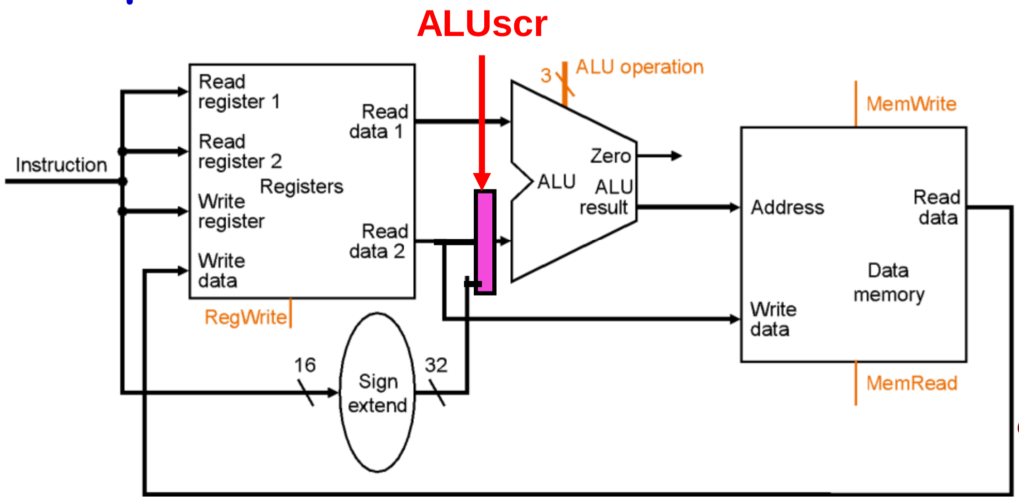
Store

Combining Datapaths (cont.)

- How to have different datapaths for different instruction?

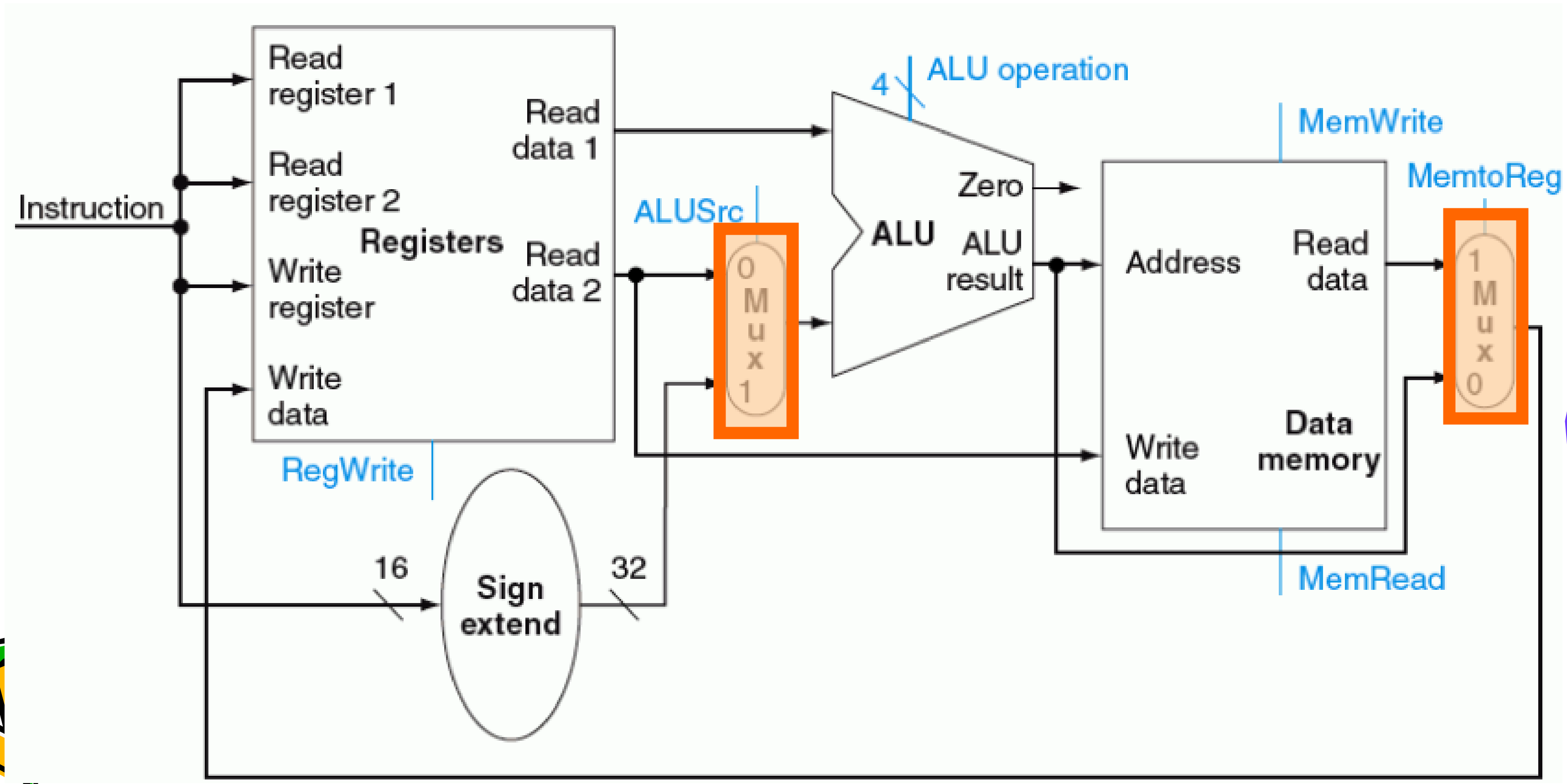


- Use a multiplexer

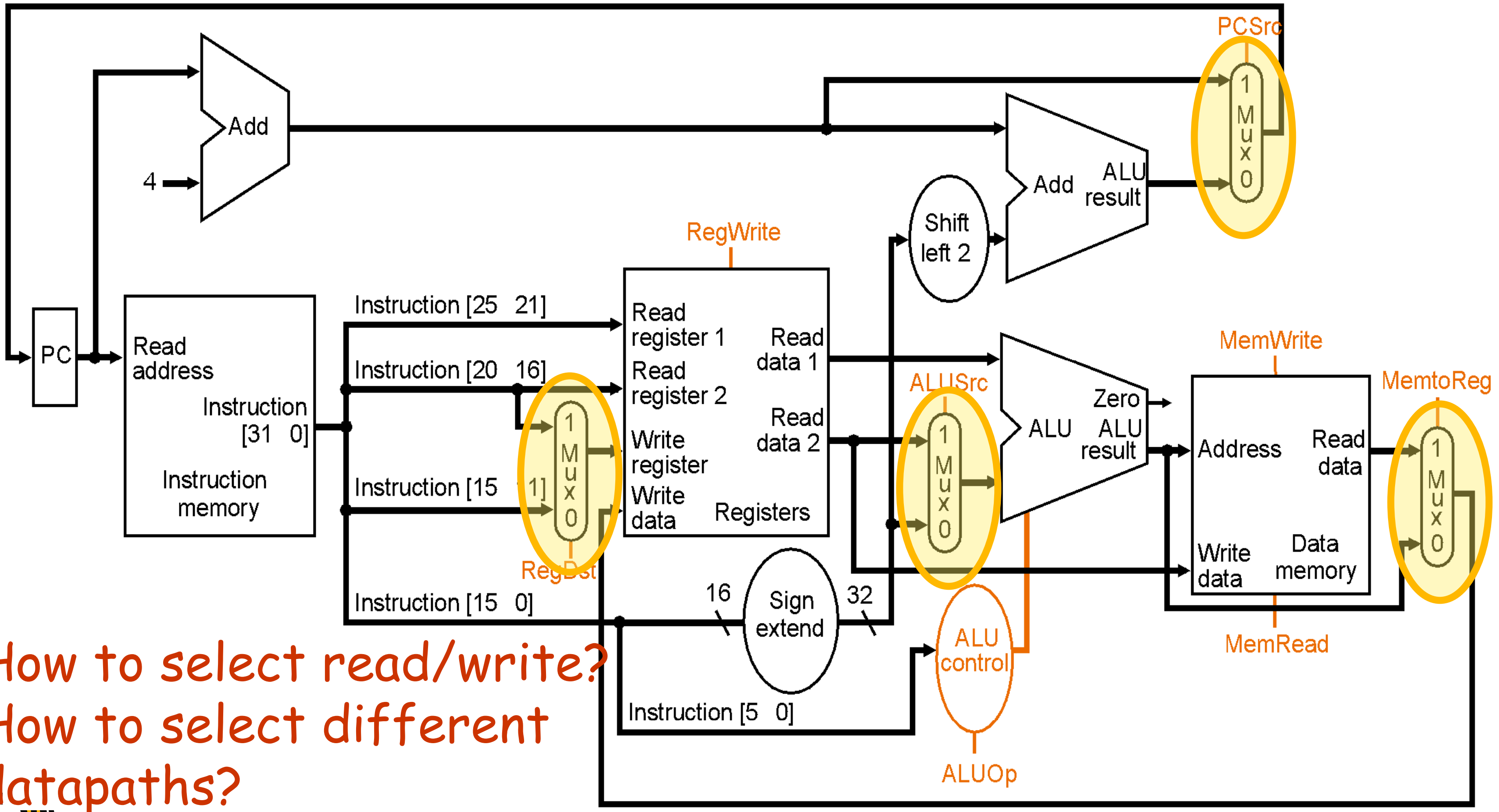


Combining Datapaths (cont.)

- 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: Answer

- 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

Reminder: Instruction Encoding

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

31 26	21	16	11	6	0
op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

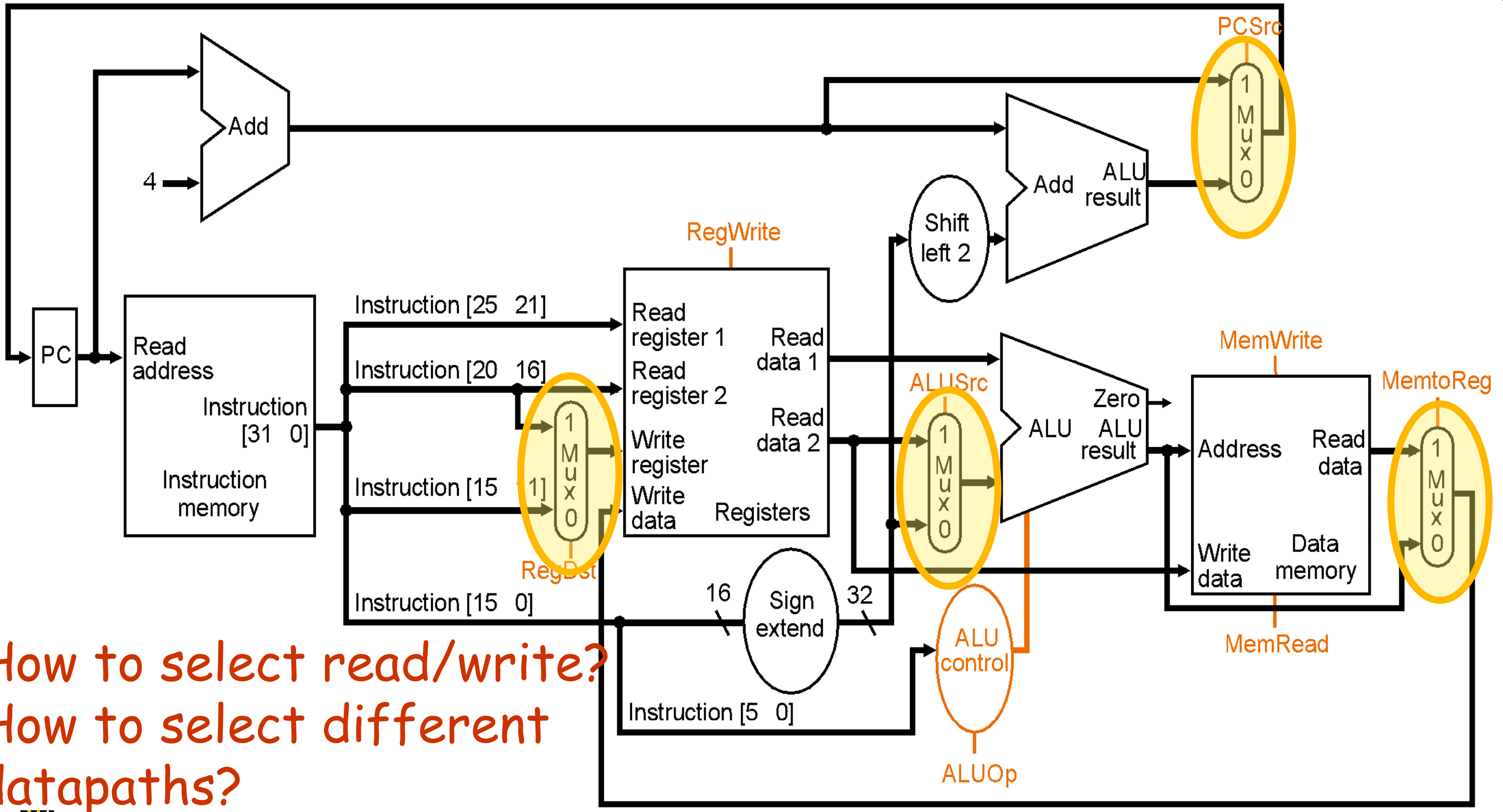
- LOAD / STORE
  - lw rt, rs, imm
  - sw rt, rs, imm

31 26	21	16	0
op	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits

- BRANCH
  - beq rs, rt, imm

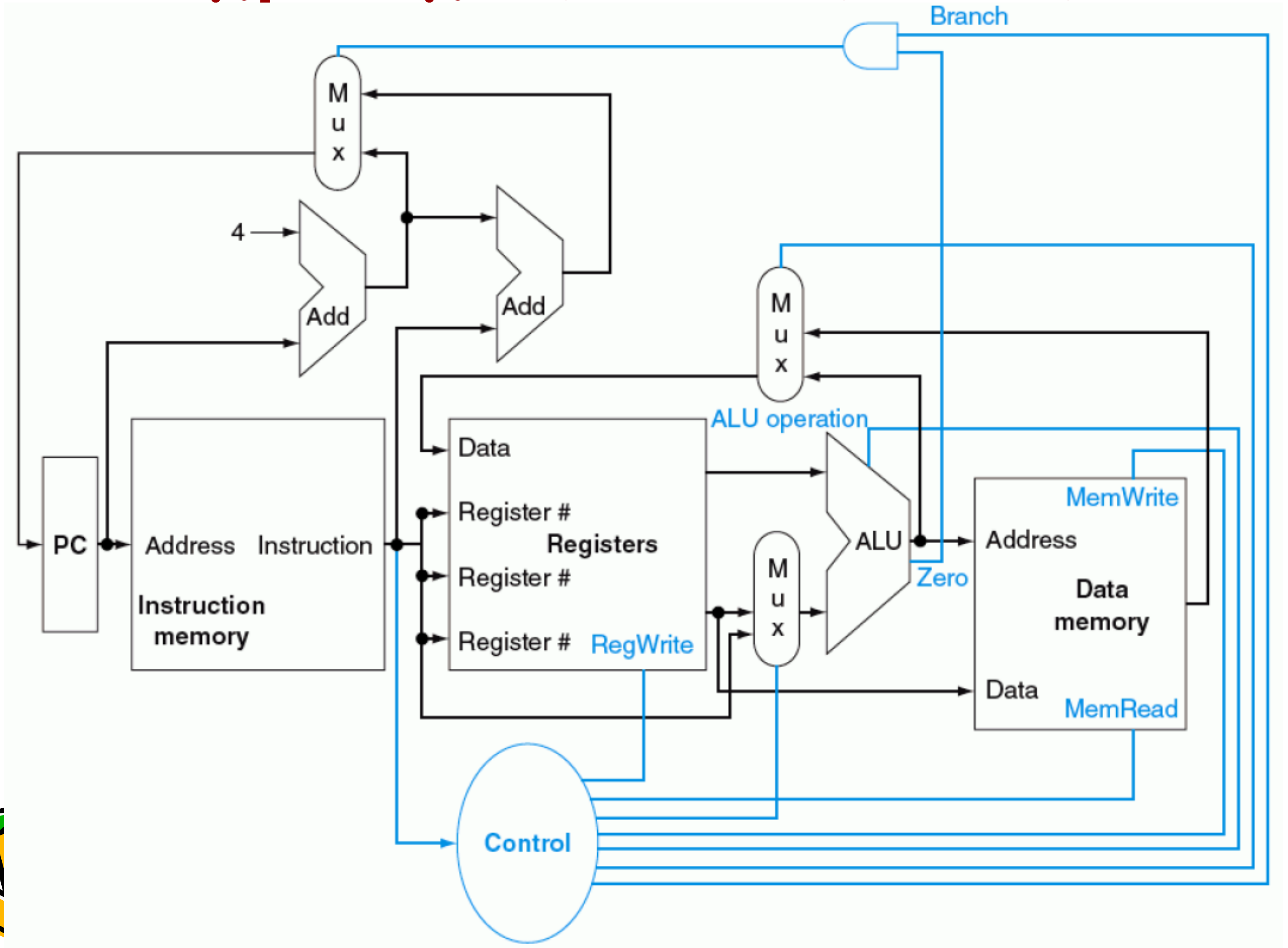
31 26	21	16	0
op	rs	rt	displacement
6 bits	5 bits	5 bits	16 bits

All Together: Single Cycle Datapath



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

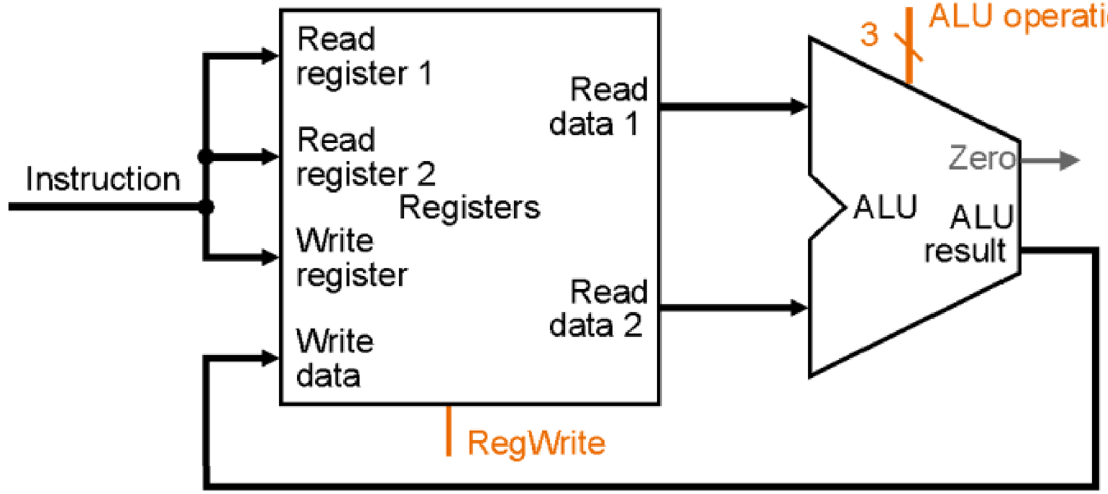
Abstract View of MIPS Implementation: Control



Datapath for Reg-Reg Operations

- GPR[rd] ← GPR[rs] op GPR[rt]
  - Example: *add rd, rs, rt*
  - ALUoperation signal depends on op and funct

31–26	26–21	21–16	16–11	11–6	6–0
op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

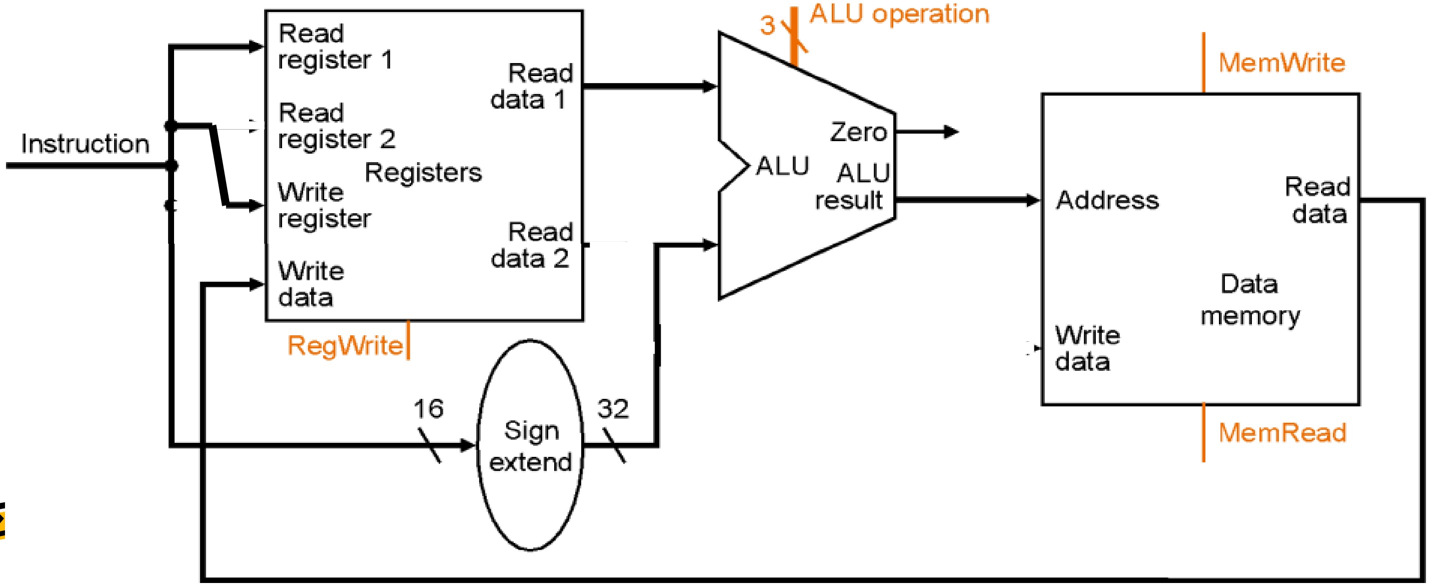


ALU operation **and** ***RegWrite*****: control logic after decoding instruction**

Datapath for Load Operations

- GPR[rt] ← Mem[GPR[rs] + SignExt[imm16]]
  - Example: *lw rt, rs, imm16*

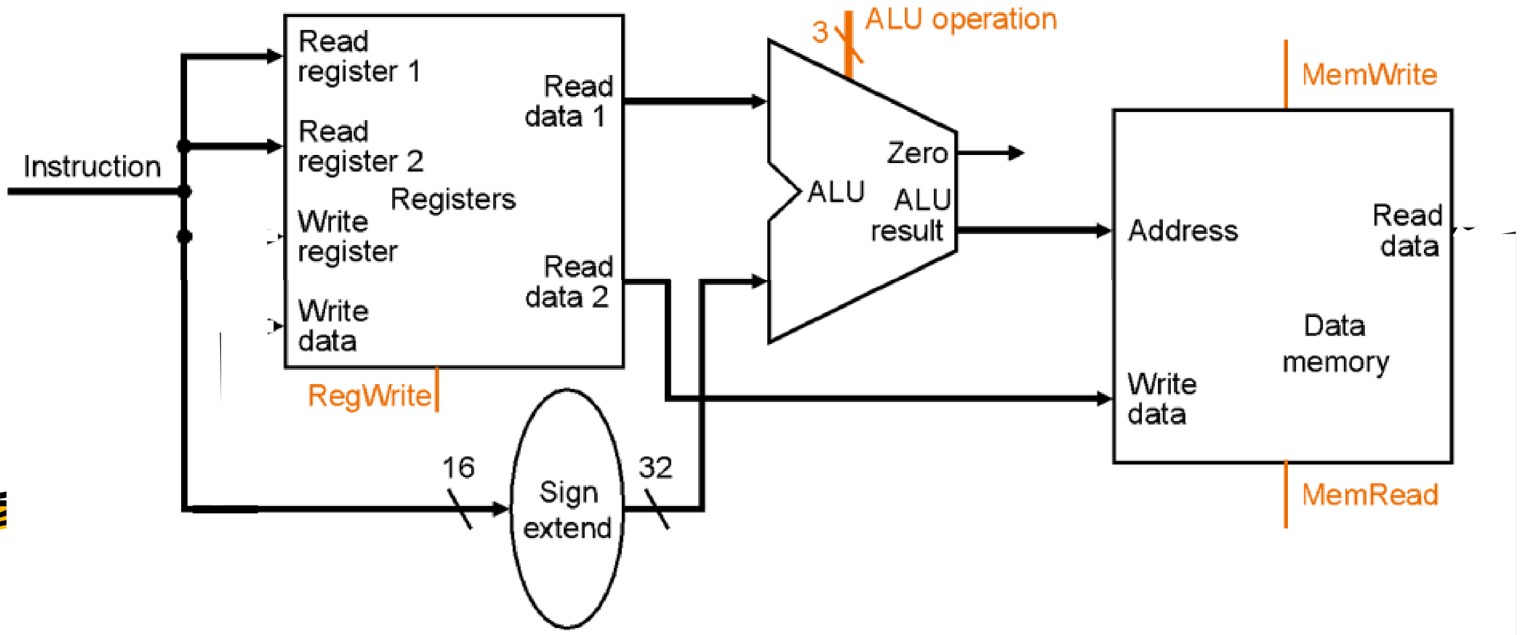
31 26	21	16	0
op	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits



Datapath for Store Operations

- Mem[GPR[rs] + SignExt[imm16]] ← GPR[rt]
  - Example: *sw rt, rs, imm16*

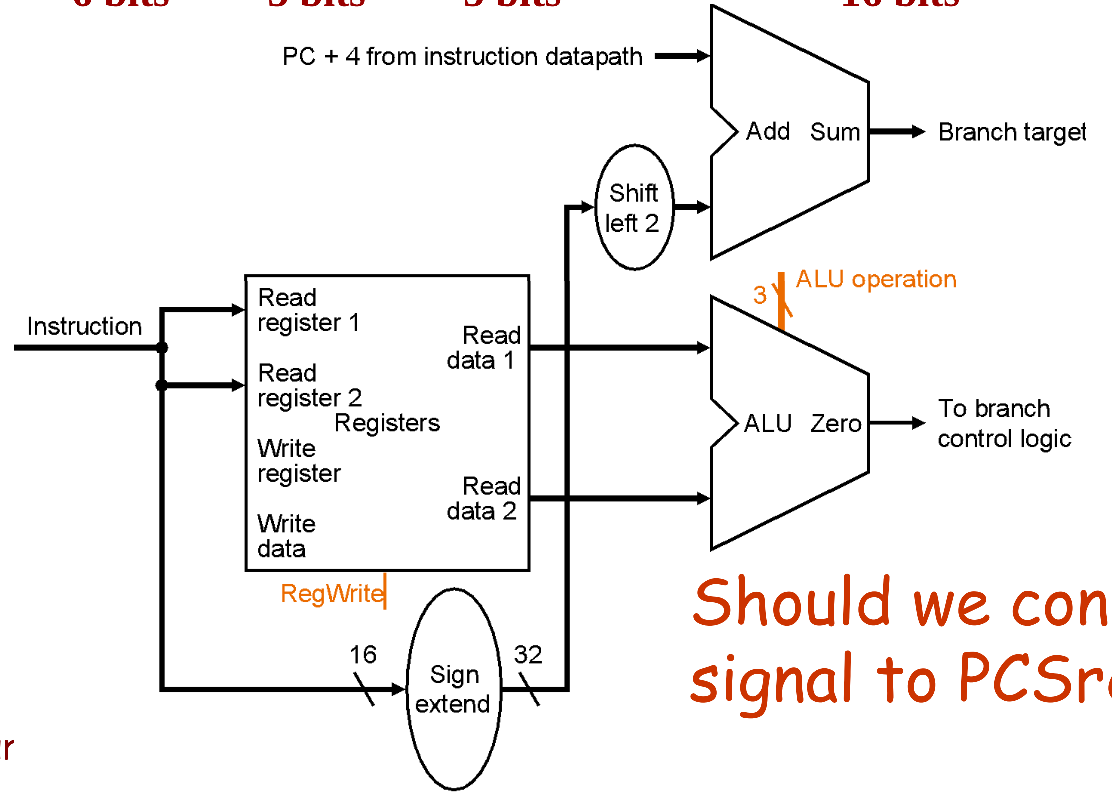
31 26	21	16	0
op	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits



Datapath for Branch Operations

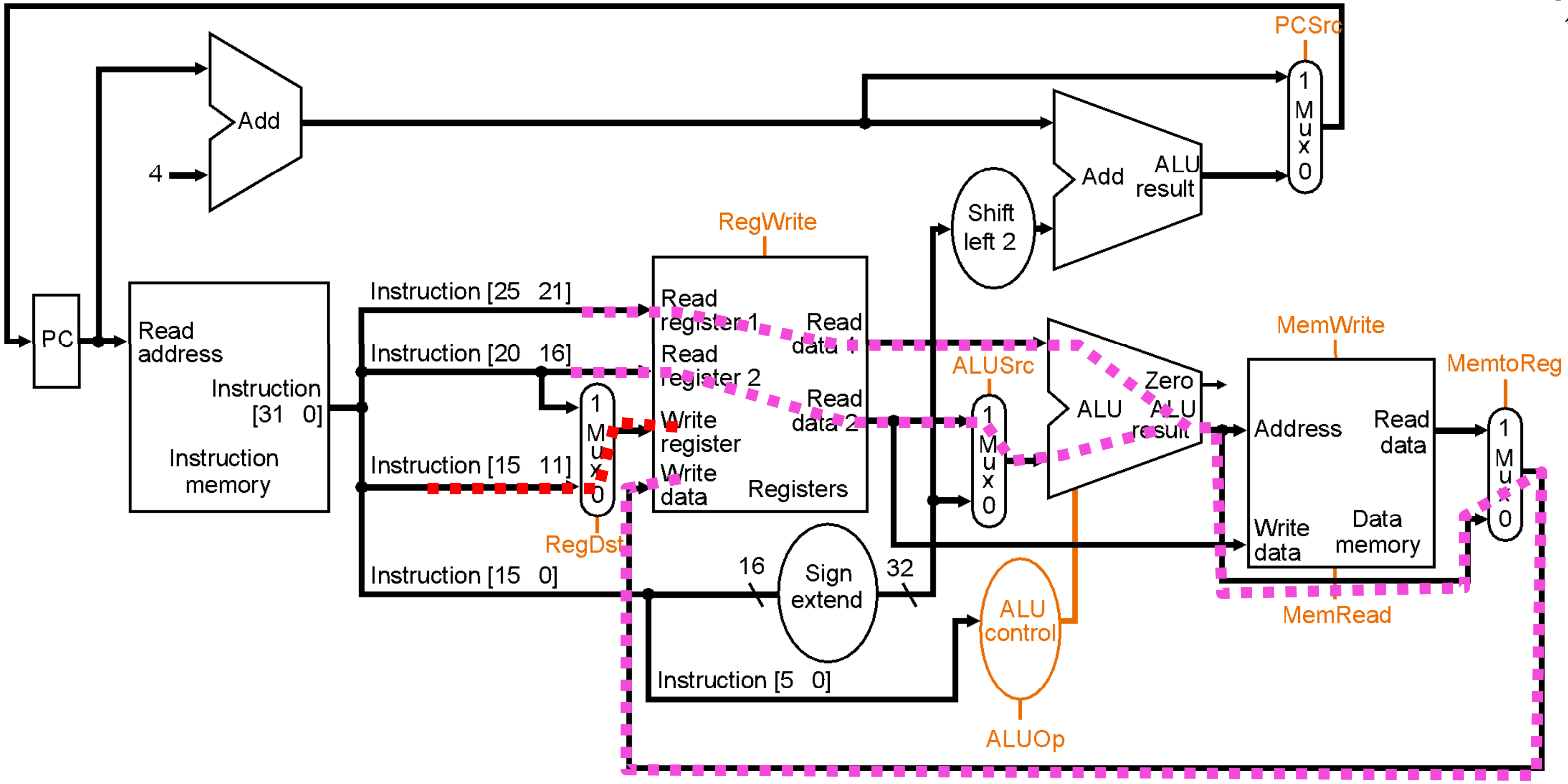
- *beq rs, rt, imm16*

31 — 26	26 — 21	21 — 16	16 — 0
op	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits



Should we connect "Zero" signal to PCSrc selector?

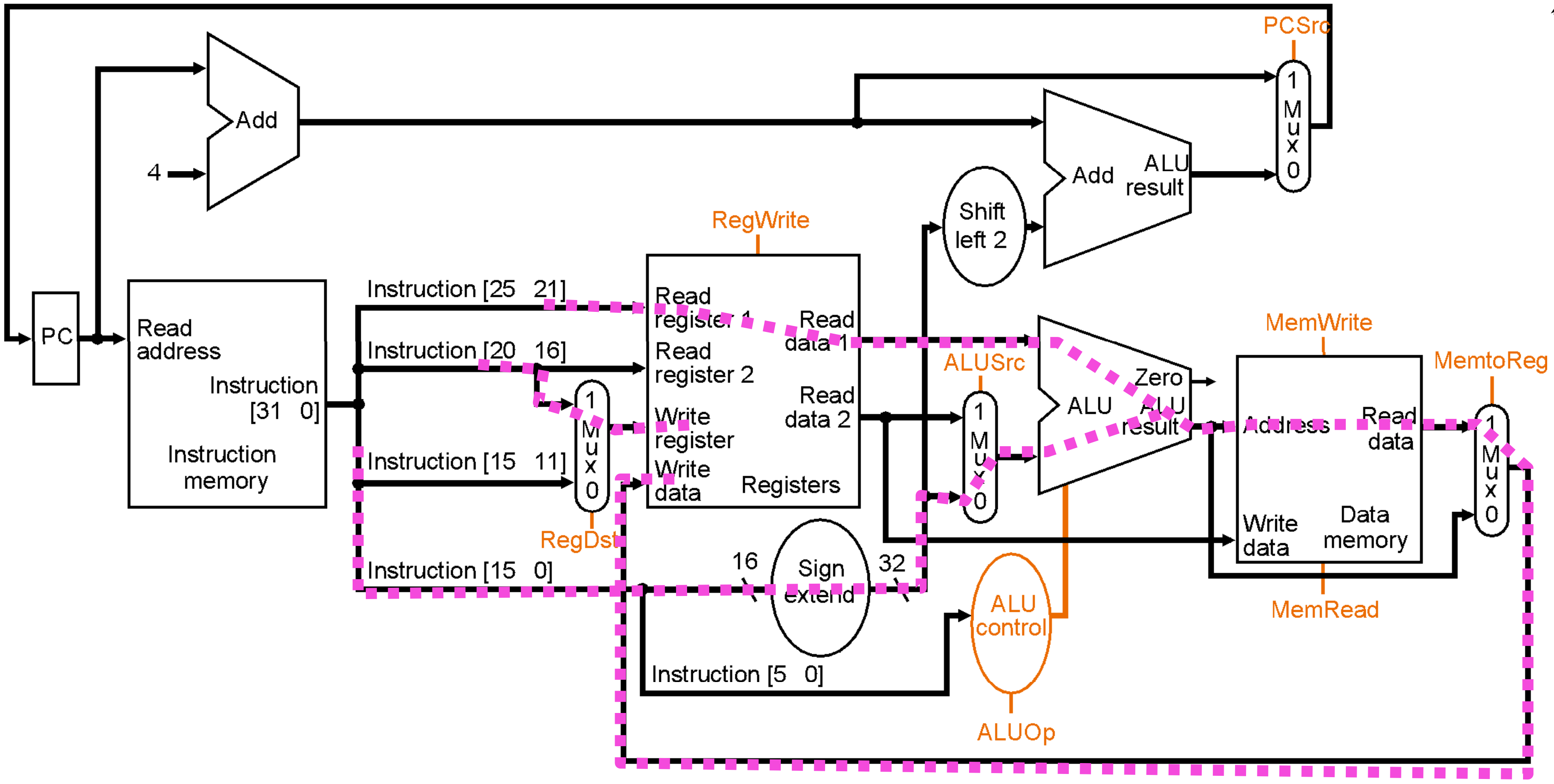
R-Format Datapath (e.g. *add*)



Need ALUsrc=1, ALUop="add", MemWrite=0, MemToReg=0,

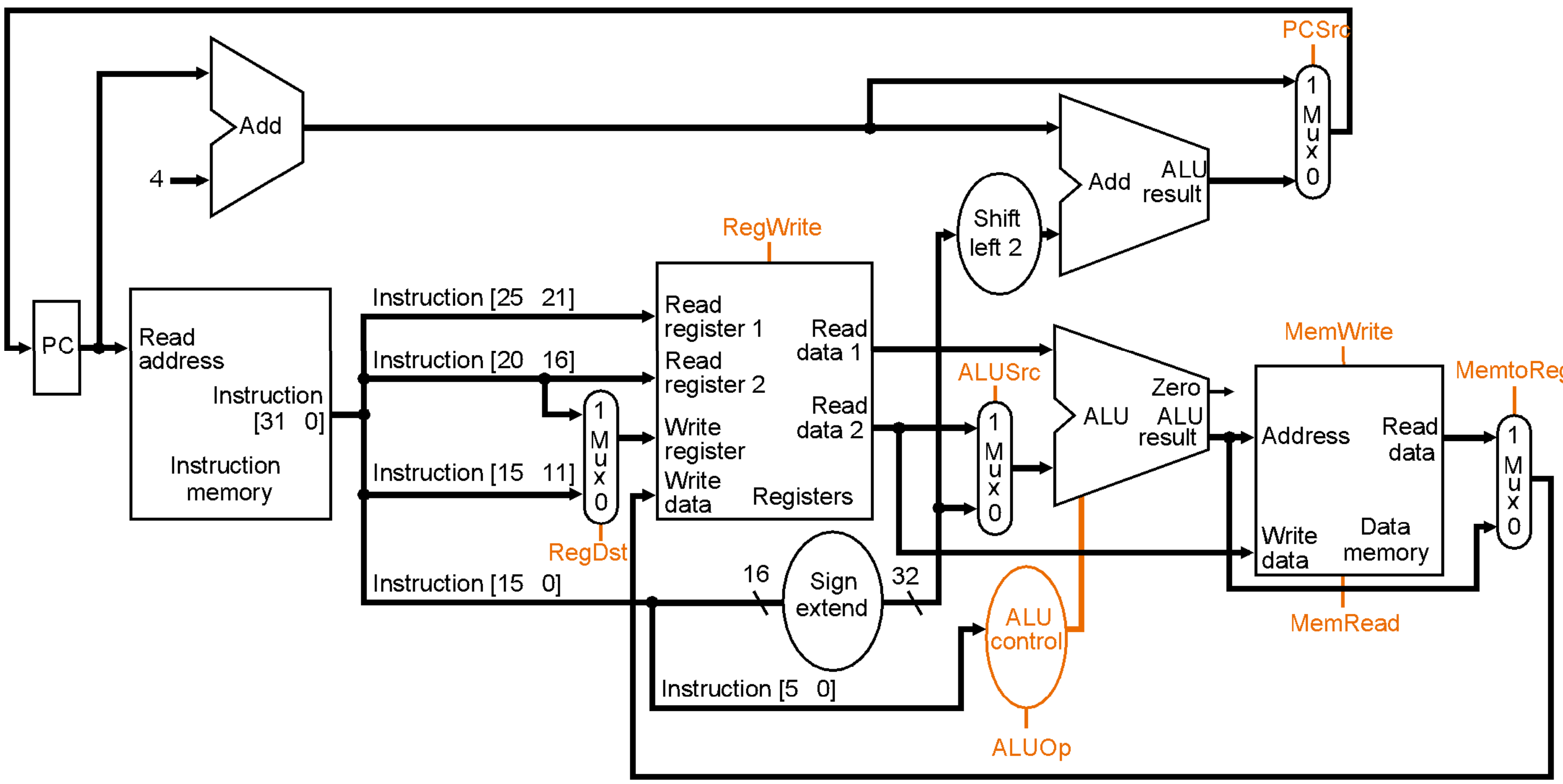
RegDst = 0, RegWrite=1 and PCsrc=1.

Load Datapath

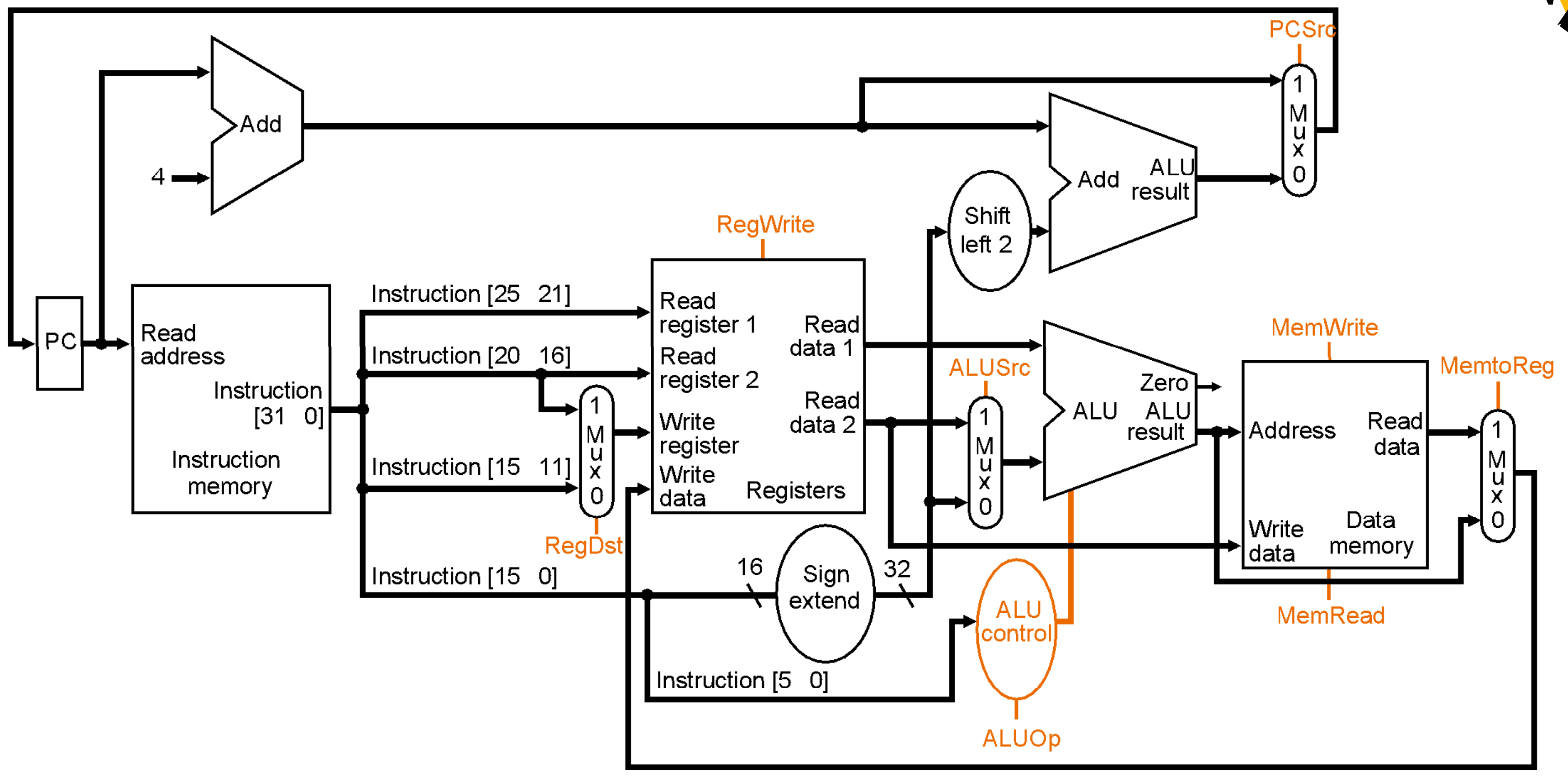


What control signals do we need for load??

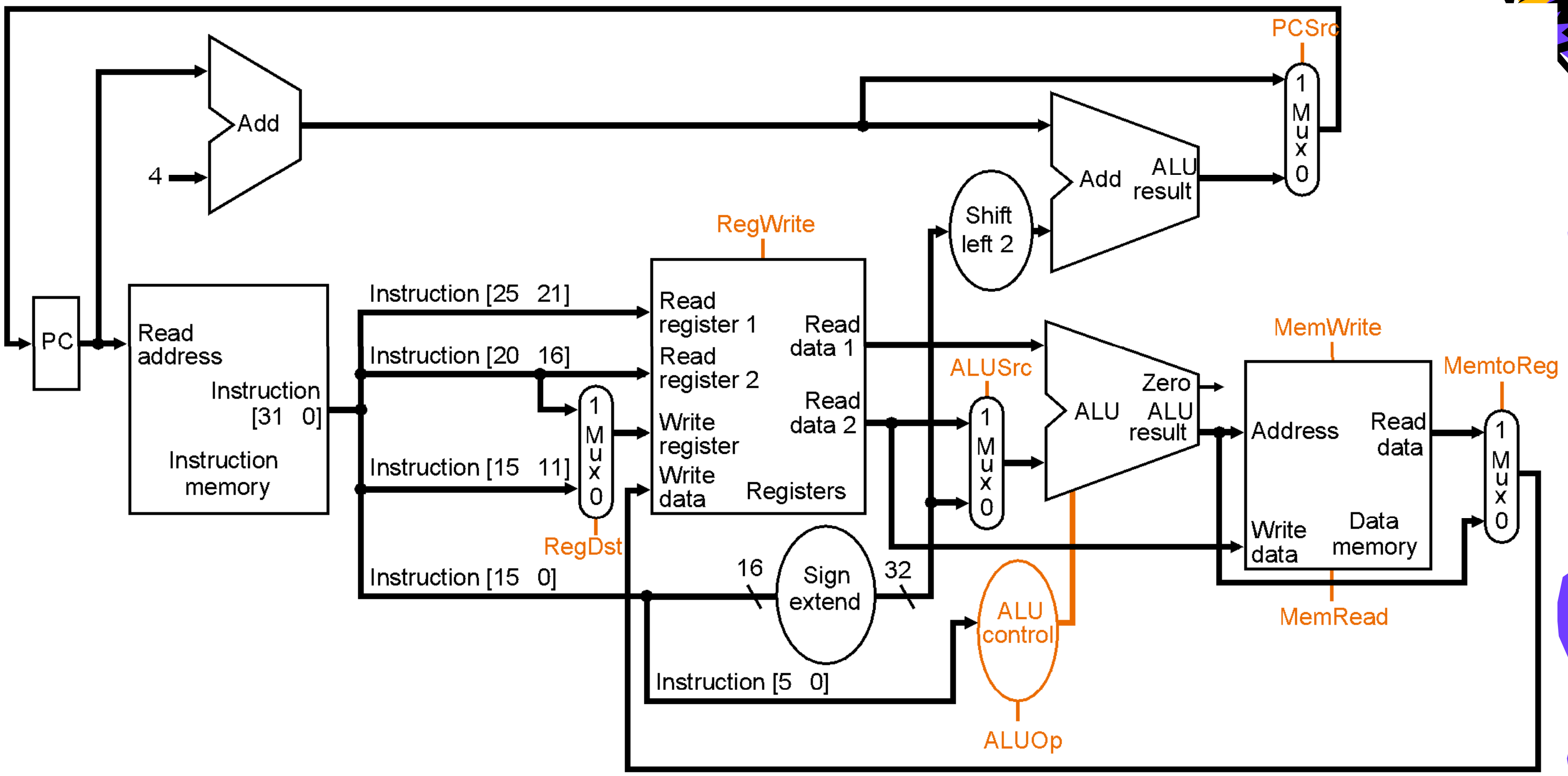
Store Datapath



beq Datapath



Putting it All Together



We have everything except details for generating **control signals**

Backup