CSCI 341 Machine Organization

–Chapter 4: The Processor

In a major matter, no details are small.

French Proverb

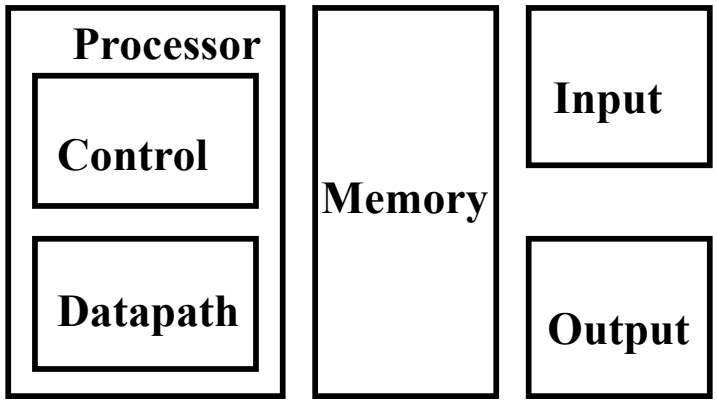
Topics

- ➔ **Single Cycle CPU Design (sections 4.1-4.4)**

- **Pipelining (sections 4.6-4.9)**
  - Overview
  - Pipelined Datapath and control
  - Data hazard
  - Control hazard

The Big Picture

- The Five Classic Components of a Computer



- Performance of a machine is determined by:
  - Instruction count; Clock cycle time; Clock cycles per instruction
- Processor design (datapath and control) will determine:
  - Clock cycle time; Clock cycles per instruction

How to Design a Processor: Step by Step

1. **Analyze instruction set => datapath requirements**
    1. the meaning of each instruction is given by the *register transfers*
    2. datapath must include storage element for registers
    3. datapath must support each register transfer
2. Select the set of datapath components and establish clocking methodology
3. Assemble the datapath meeting the requirements
4. Analyze the implementation of each instruction to determine the settings of the control points that affects the register transfer
5. Assemble the control logic

RISC-V Instruction Formats

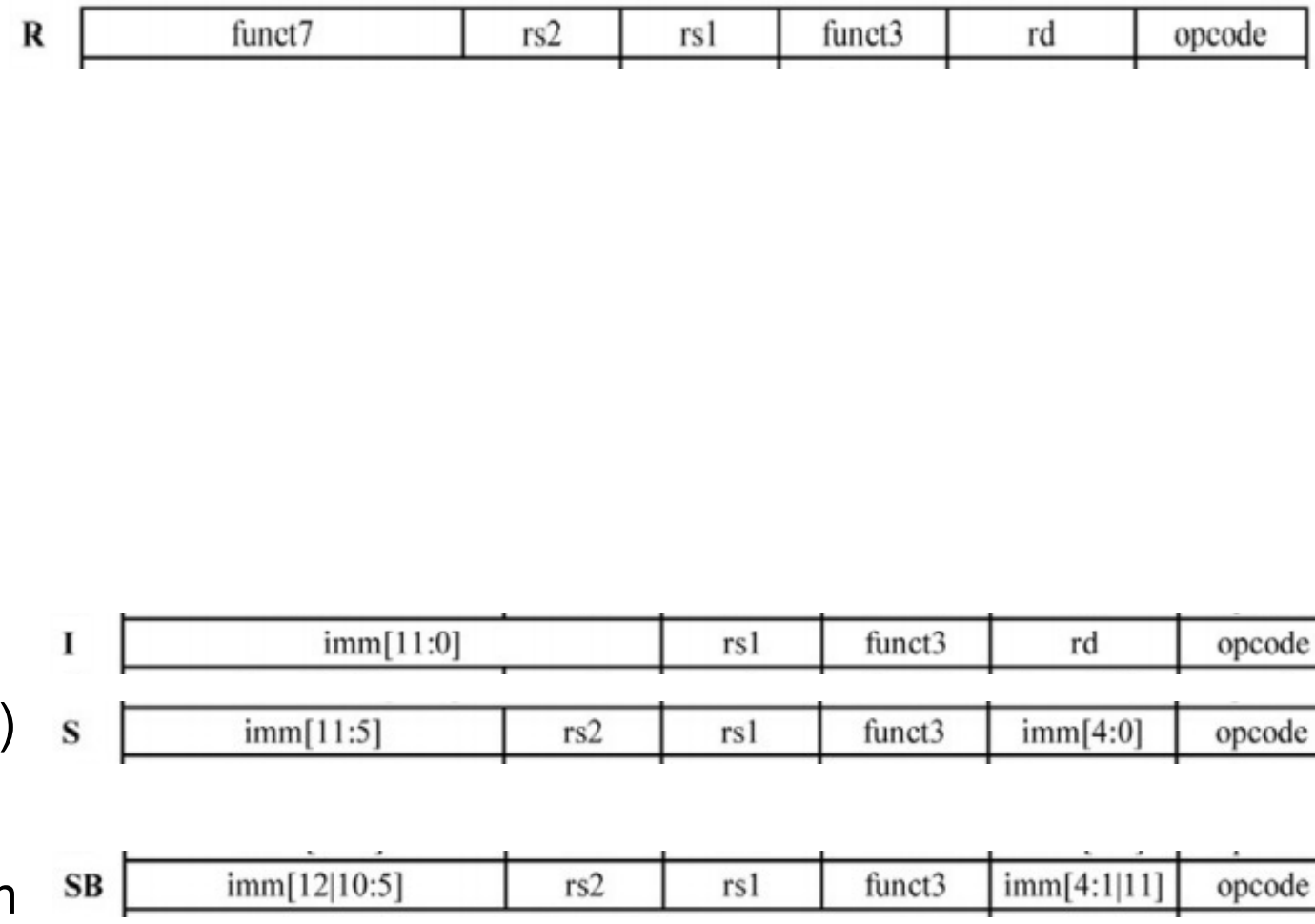
- All RISC-V instructions are 32 bits long. 6 formats:

CORE INSTRUCTION FORMATS

	31 27	26 25	24 20	19 15	14 12	11 7	6 0
R	funct7		rs2	rs1	funct3	rd	opcode
I	imm[11:0]			rs1	funct3	rd	opcode
S	imm[11:5]		rs2	rs1	funct3	imm[4:0]	opcode
SB	imm[12|10:5]		rs2	rs1	funct3	imm[4:1|11]	opcode
U	imm[31:12]					rd	opcode
UJ	imm[20|10:1|11|19:12]					rd	opcode

Step 1a: The RISC-V-lite Subset

- **ADD, SUB, AND, OR**
  - add rd, rs1, rs2
  - sub rd, rs1, rs2
  - and rd, rs1, rs2
  - or rd, rs1, rs2
- **LOAD and STORE Word**
  - lw rd, imm (rs1)
  - sw rs2, imm (rs1)
- **BRANCH:**
  - beq rs1, rs2, imm



Register Transfer Language

- **RTL gives the meaning of the instructions**
- **First step is to fetch the instruction from memory**

  funct7|rs2|rs1|funct3|rd|opcode= MEM[PC]

  Imm[11:0]|rs1|funct3|rd|opcode = MEM[PC]

  Imm[11:5]|rs2|rs1|funct3|imm[4:0]|opcode= MEM[PC]

  Imm[12][10:5]|rs2|rs1|funct3|imm[4:1|11]|opcode= MEM[PC]

- **Each instruction involves register transfers**

Step 1a. Register Transfers by Instruction

instruction	register transfers	
ADD	R[rd]←R[rs1] + R[rs2]	PC←PC+4
SUB	R[rd] ←R[rs1] - R[rs2]	PC←PC+4
AND	R[rd] ←R[rs1] & R[rs2]	PC←PC+4
OR	R[rd] ←R[rs1] | R[rs2]	PC←PC+4
LOAD	R[rd] ←MEM[R[rs1]+ SignExtImm]	PC←PC+4
STORE	MEM[R[rs1]+ SignExtImm] ←R[rs2]	PC←PC+4
BEQ	if (R[rs1]==R[rs2]) PC←PC+(SignExtImm<<1) else PC←PC+4	

Step 1b: Requirements of the Instruction Set

- **Memory**
  - instruction & data: instruction=MEM[PC]
- **Registers (32 x 32)**
  - read rs1; read rs2; Write rd
- **PC, what is the new PC?**
  - Add 4 or extended immediate to PC
- **Extender: sign-extension**
- **Add: rs1+rs2, rs1+SignExtImm**
  - Which operations
- **Sub, And, Or: rs1 op rs2**
  - Which operations

How to Design a Processor: Step by Step

1. Analyze instruction set => datapath requirements
    1. the meaning of each instruction is given by the *register transfers*
    2. datapath must include storage element for registers
    3. datapath must support each register transfer
2. **Select the set of datapath components and establish clocking methodology**
3. Assemble the datapath meeting the requirements
4. Analyze the implementation of each instruction to determine the settings of the control points that affects the register transfer
5. Assemble the control logic

Step 2: Components of the Datapath

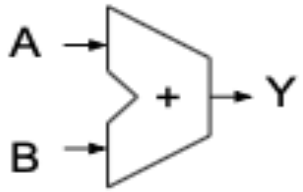
- **Combinational Elements**
  - Elements that operate on data values
  - Output depends only on the current inputs
- **State/sequential Elements**
  - Elements that contain state
  - Output depends on both inputs and the internal state (previous input)
  - Clocking methodology
    - Defines when signals can be read and written
    - Wouldn't want to read a signal while it is being written

Basic Building Blocks

(Combinational Logic Elements)

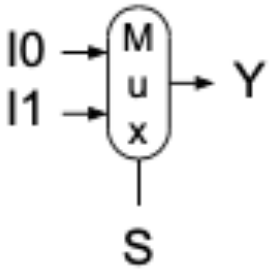
- **Adder**

Y=A+B



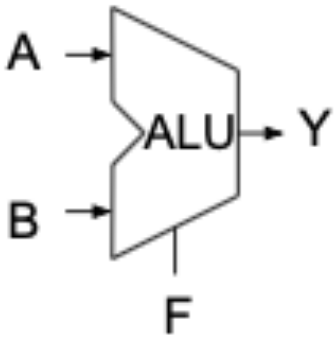
- **MUX**

Y=S?11:10



- **ALU**

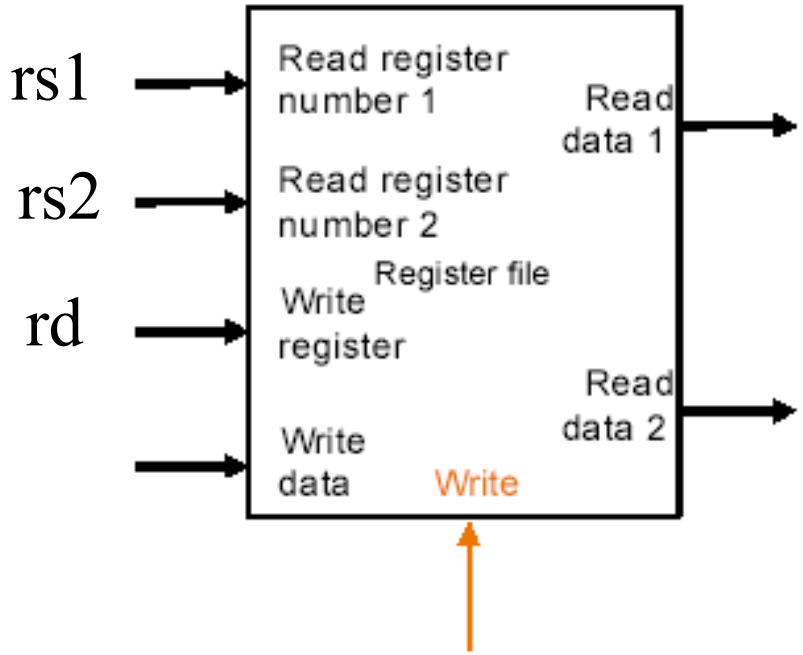
Y=F(A,B)



- **All blocks have two 32-bit inputs and one 32-bit output**
- **Binary encoded in voltage on wires**
  - Low voltage = 0, high voltage = 1
  - One wire per bit
  - multi-bit data encoded on multi-wire buses

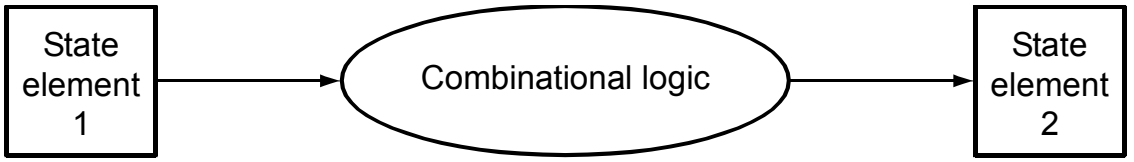
Basic Building Blocks
(State Element: Register File)

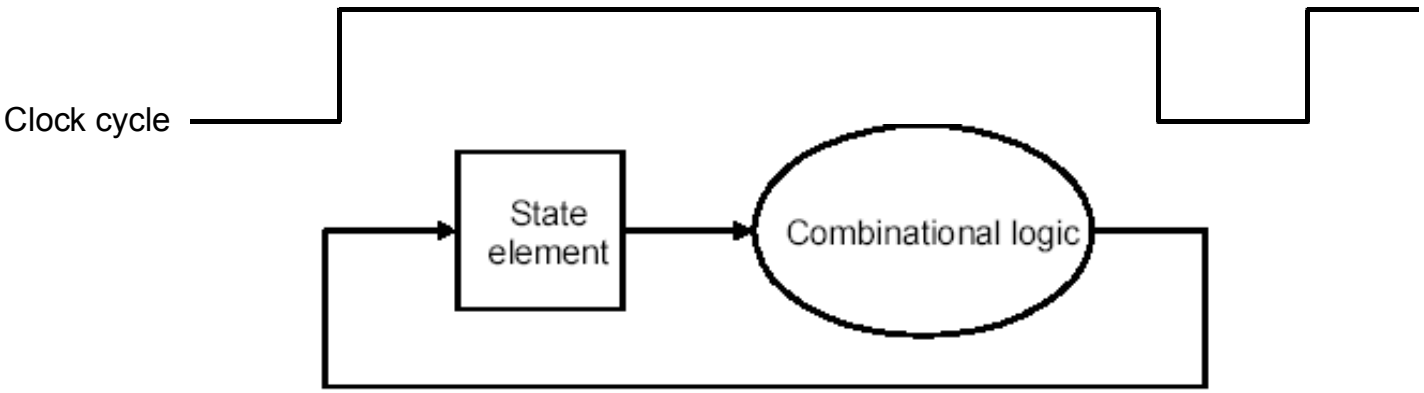
- **Consist of a set of registers built from registers**
  - Accessed by specifying a register number
- **Example**
  - A register file with 2 read ports and 1 write port



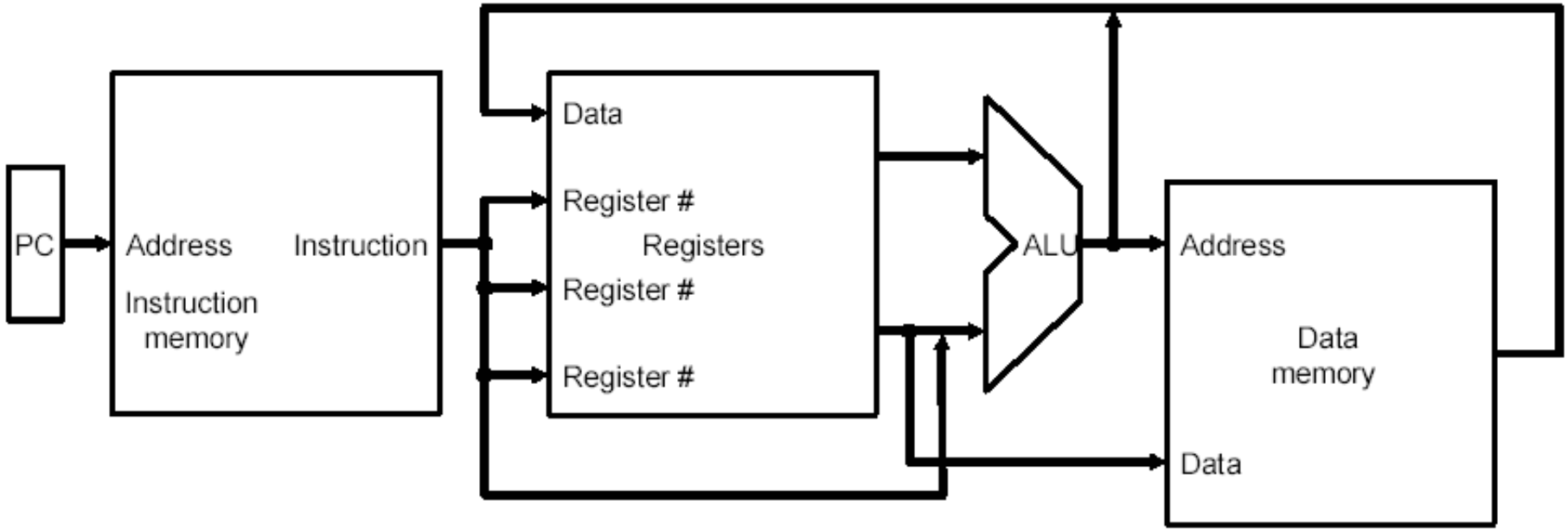
Our Implementation

- **A rising-edge-triggered methodology**
- **Typical execution:**
  - read contents of some state elements,
  - send values through some combinational logic
  - write results to one or more state elements





Abstract/Simplified View of Datapath



- **Two types of functional units**
  - combinational element: ALU
  - state elements: Instruction memory, data memory, register files, PC
- **Single cycle processor**
  - Pros: one clock cycle per instruction
  - Cons: long cycle time

How to Design a Processor: Step by Step

1. Analyze instruction set => datapath requirements
   1. the meaning of each instruction is given by the *register transfers*
   2. datapath must include storage element for registers
   3. datapath must support each register transfer
2. Select the set of datapath components and establish clocking methodology
3. **Assemble the datapath meeting the requirements**
4. Analyze the implementation of each instruction to determine the settings of the control points that affects the register transfer
5. Assemble the control logic

Step 3: Assemble DataPath

Register Transfer Requirements

⇒Datapath Assembly

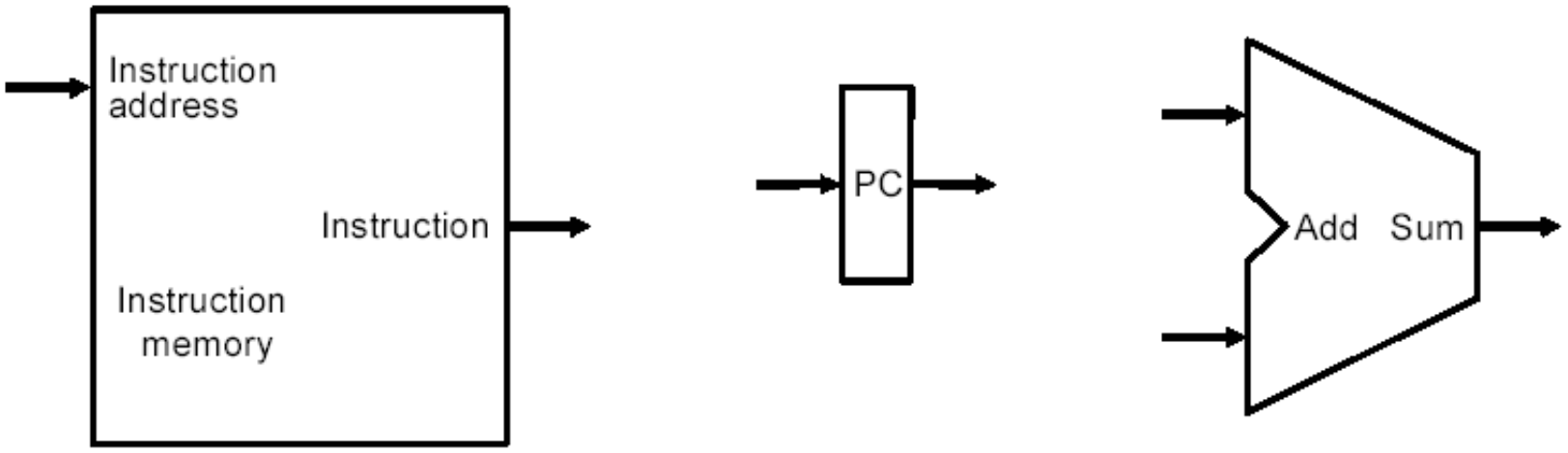
- **Instruction Fetch**
  - Instruction = MEM[PC]
  - Update PC

- **Read Operands and Execute Operation**
  - Read one or two registers
  - Execute operation
  - Write register

Step 3a: Instruction Fetch Unit

- **The common RTL operations**
  - Fetch the Instruction: mem[PC]
  - Update the program counter:
    - Sequential Code: PC ←PC + 4
    - Branch and Jump: PC ← "something else"
    - all instructions initially increment the PC
      - Don't know if instruction is a Branch/Jump or one of the other instructions until we have fetched and interpreted the instruction from memory (may need to change PC if so)

Basic Components for Instruction Fetch

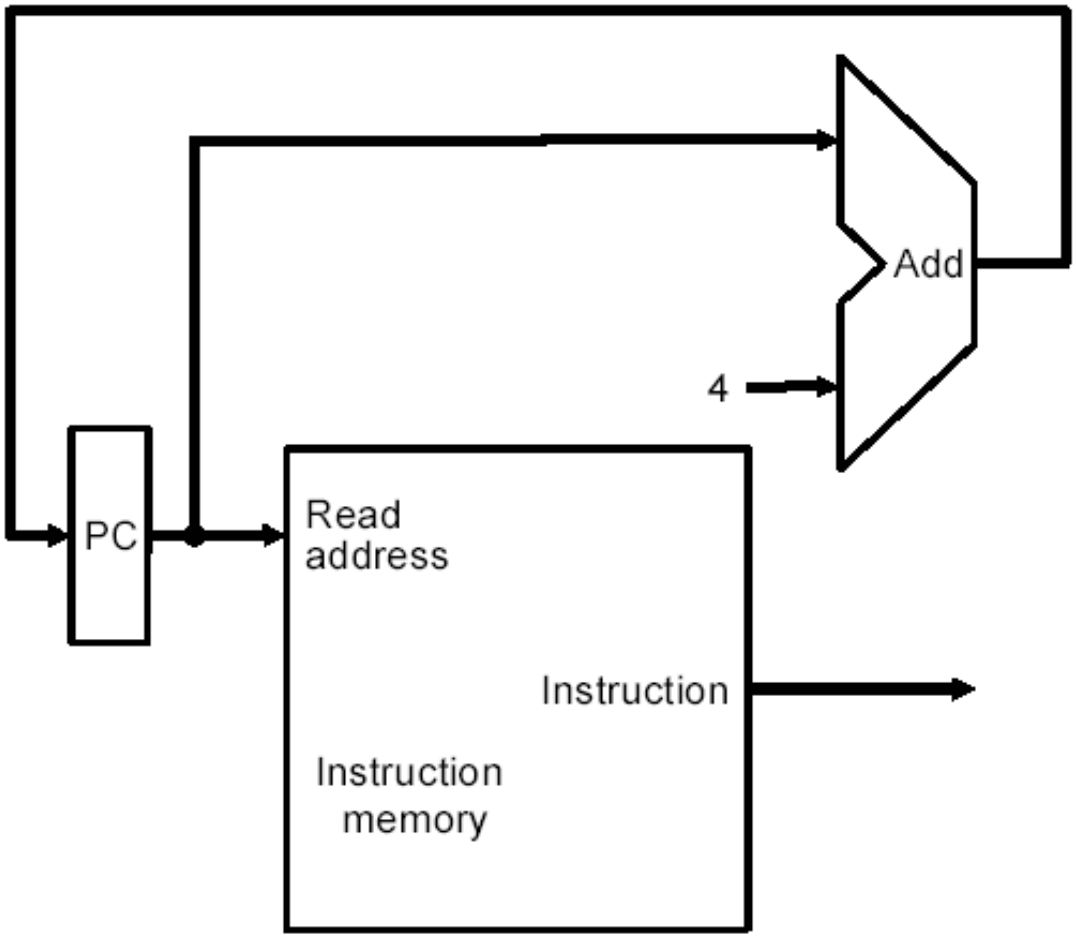


a. Instruction memory

b. Program counter

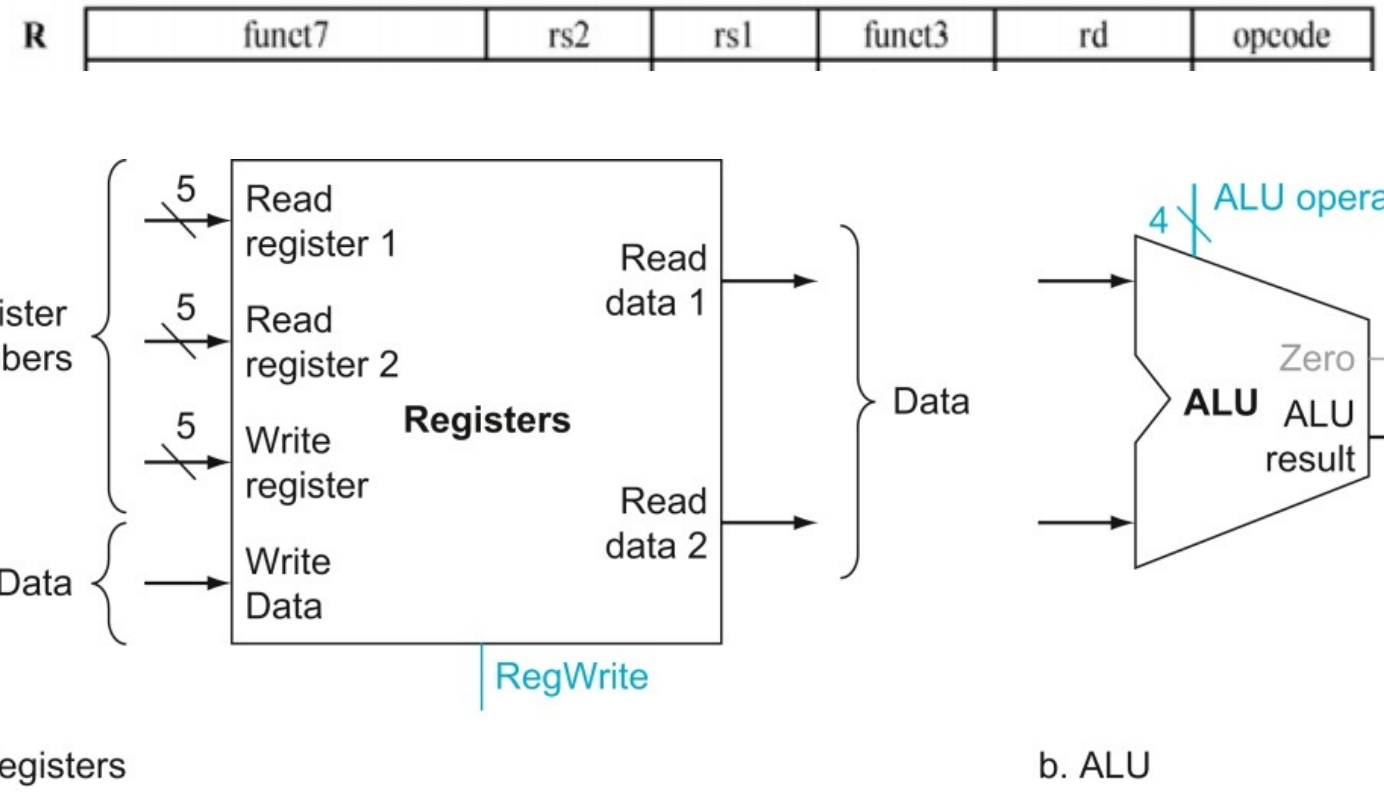
c. Adder

Datapath for Instruction Fetch

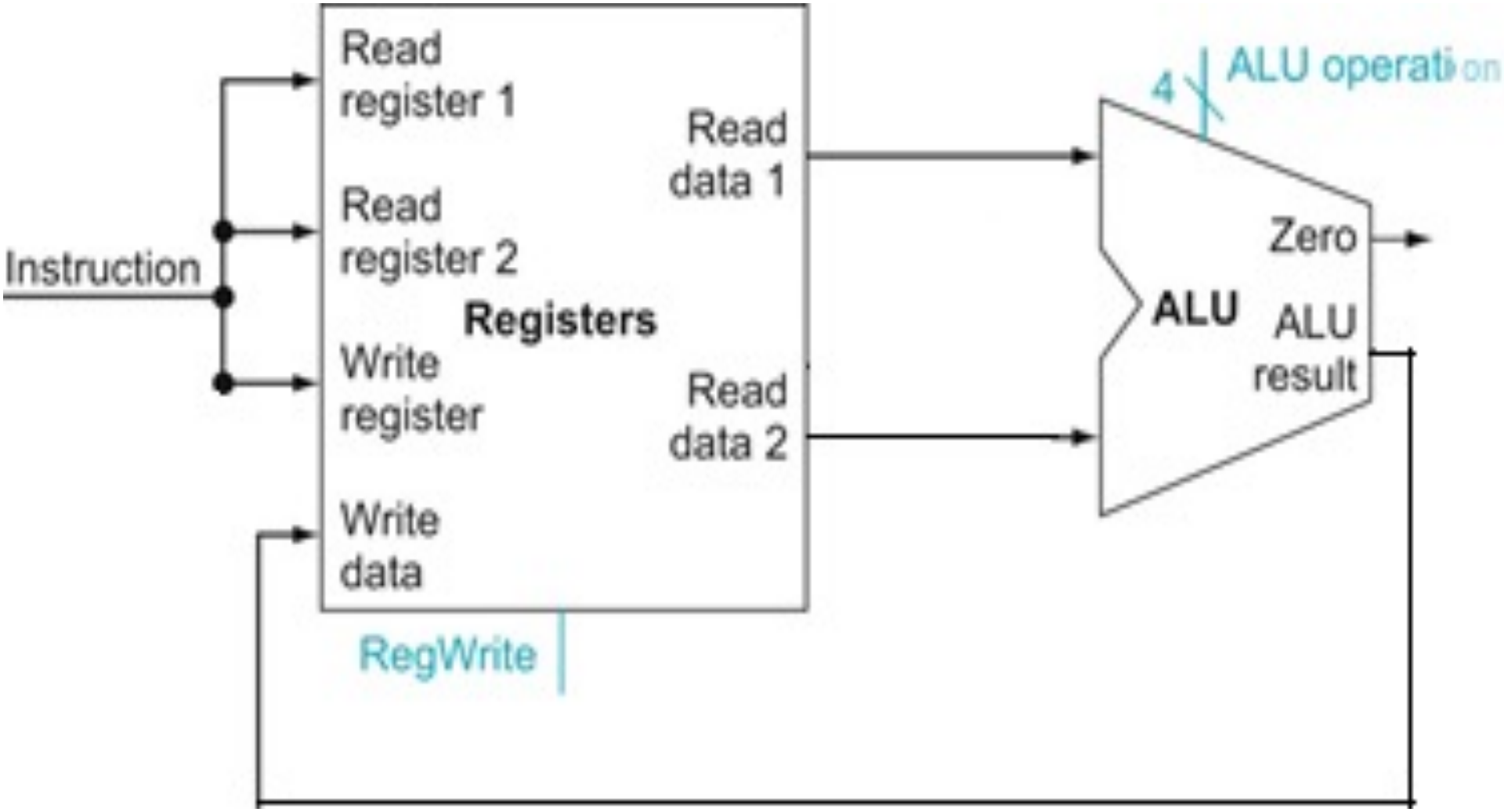


3b: R-format Instructions *(add, sub, and, or)*

- **R[rd] ← R[rs1] op R[rs2] Example: add rd, rs1, rs2**
  - Read register 1, Read register 2, and Write register come from instruction's rs1, rs2, and rd fields
  - ALU control and RegWrite: control logic after decoding the instruction

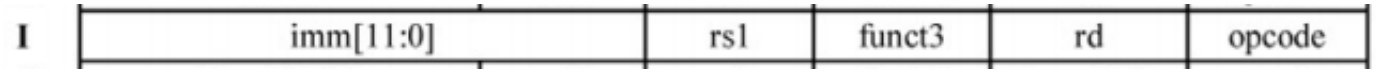


Datapath for R-format Instructions

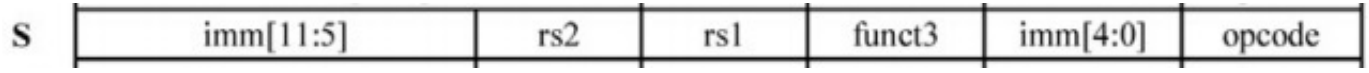


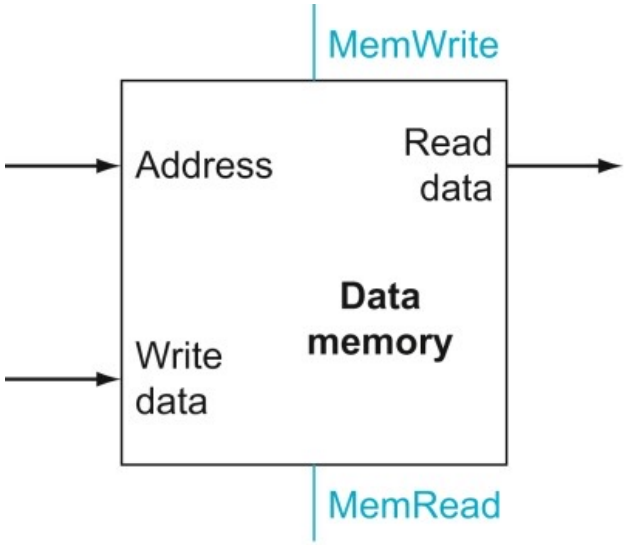
3c: Load and Store Operations

- R[rd] ← Mem[R[rs1] + SignExt[imm]] Example: lw rd, imm (rs1)

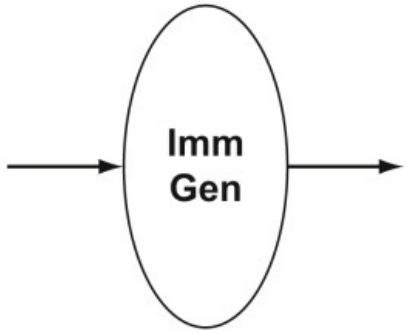


- Mem[R[rs1] + SignExt[imm] ← R[rs2]] Example: sw rs2, imm (rs1)



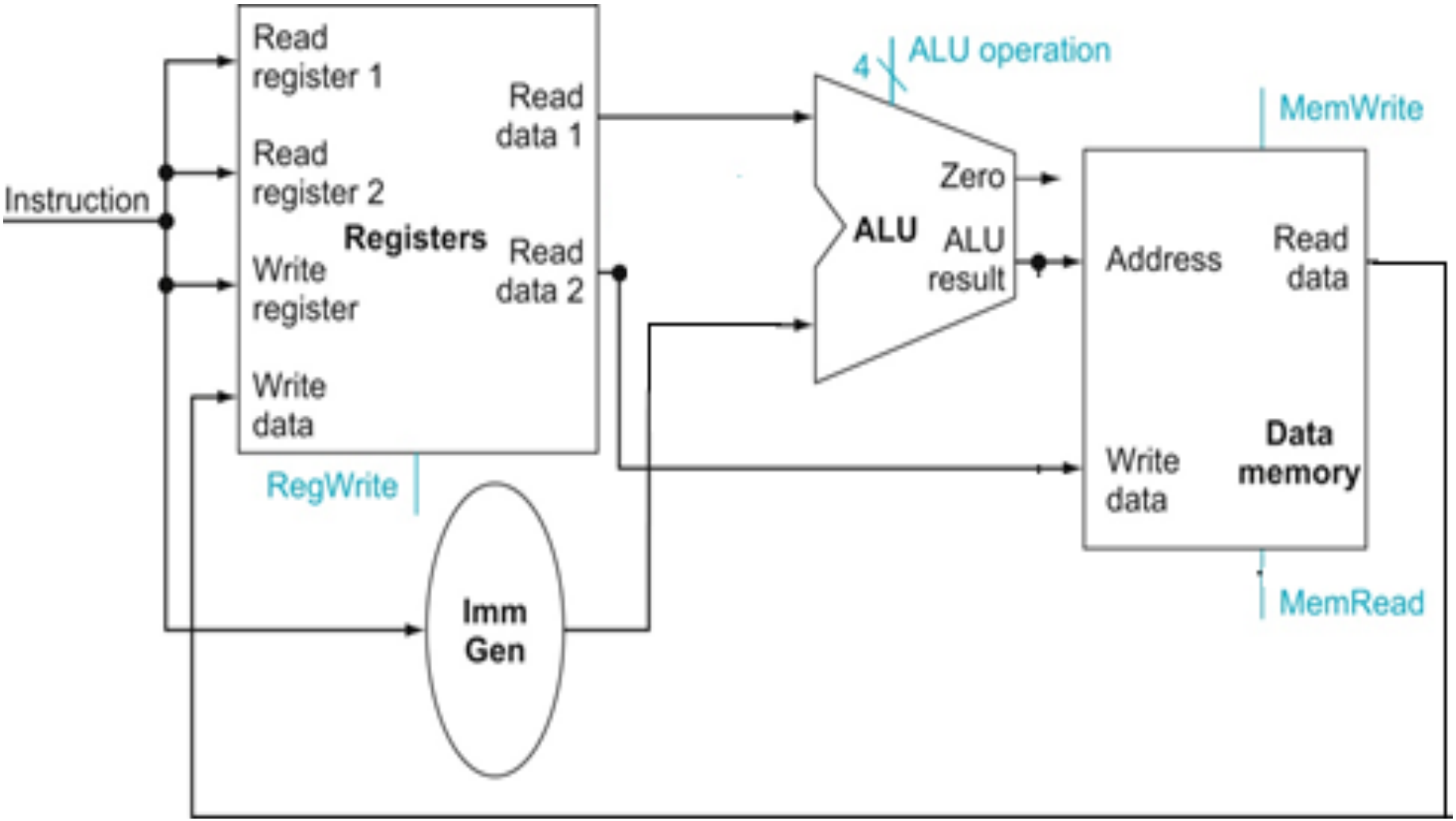


a. Data memory unit

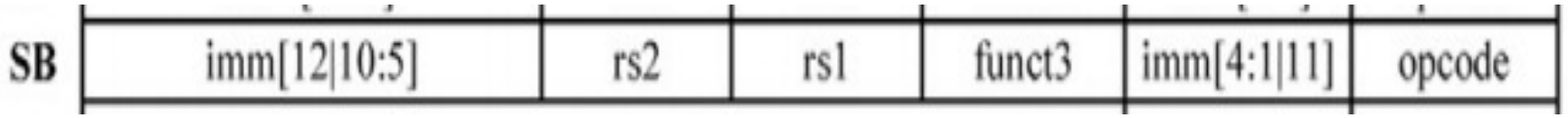


b. Immediate generation unit

Datapath for Load and Store Instructions

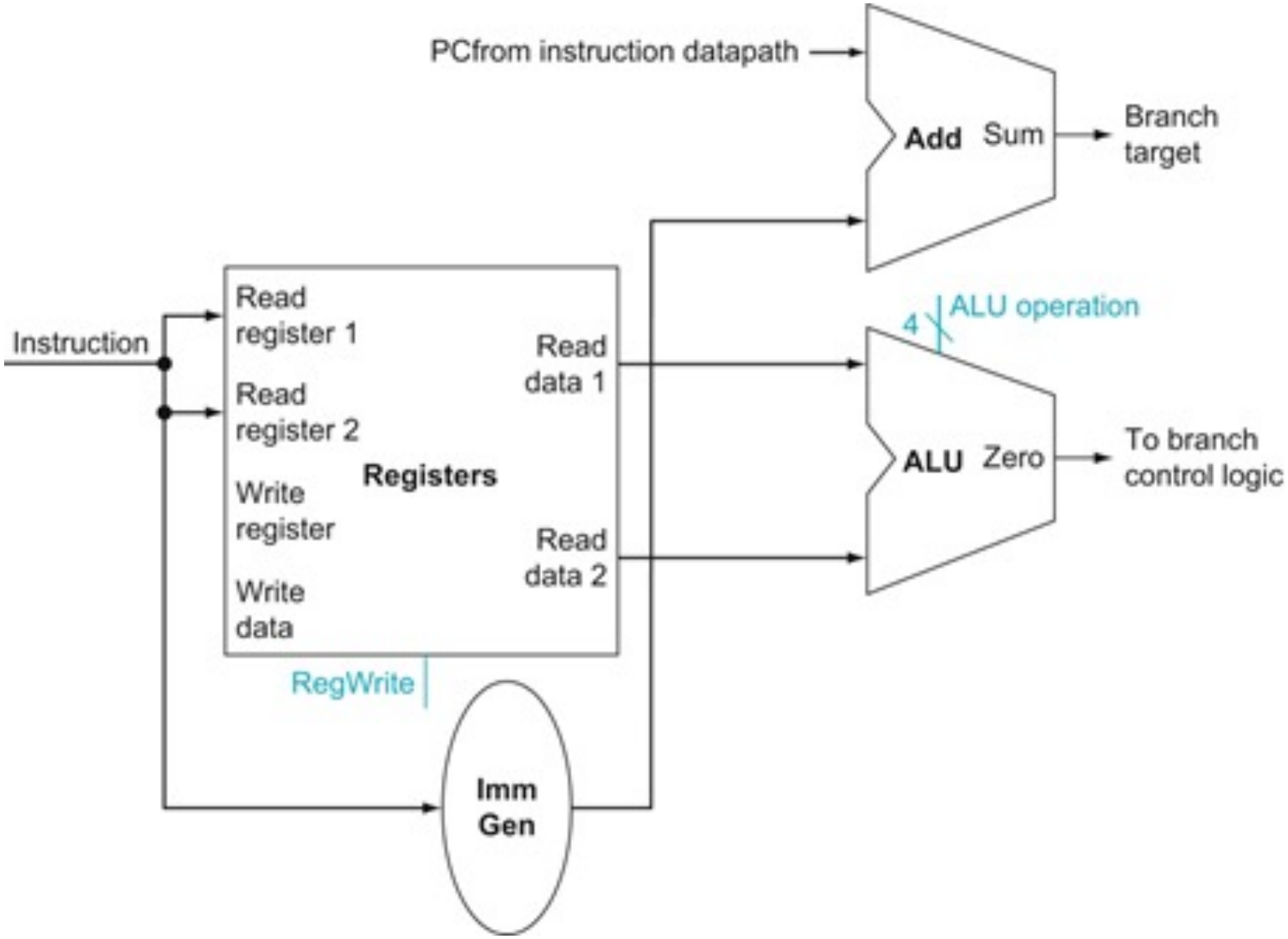


3d: The Branch Instruction



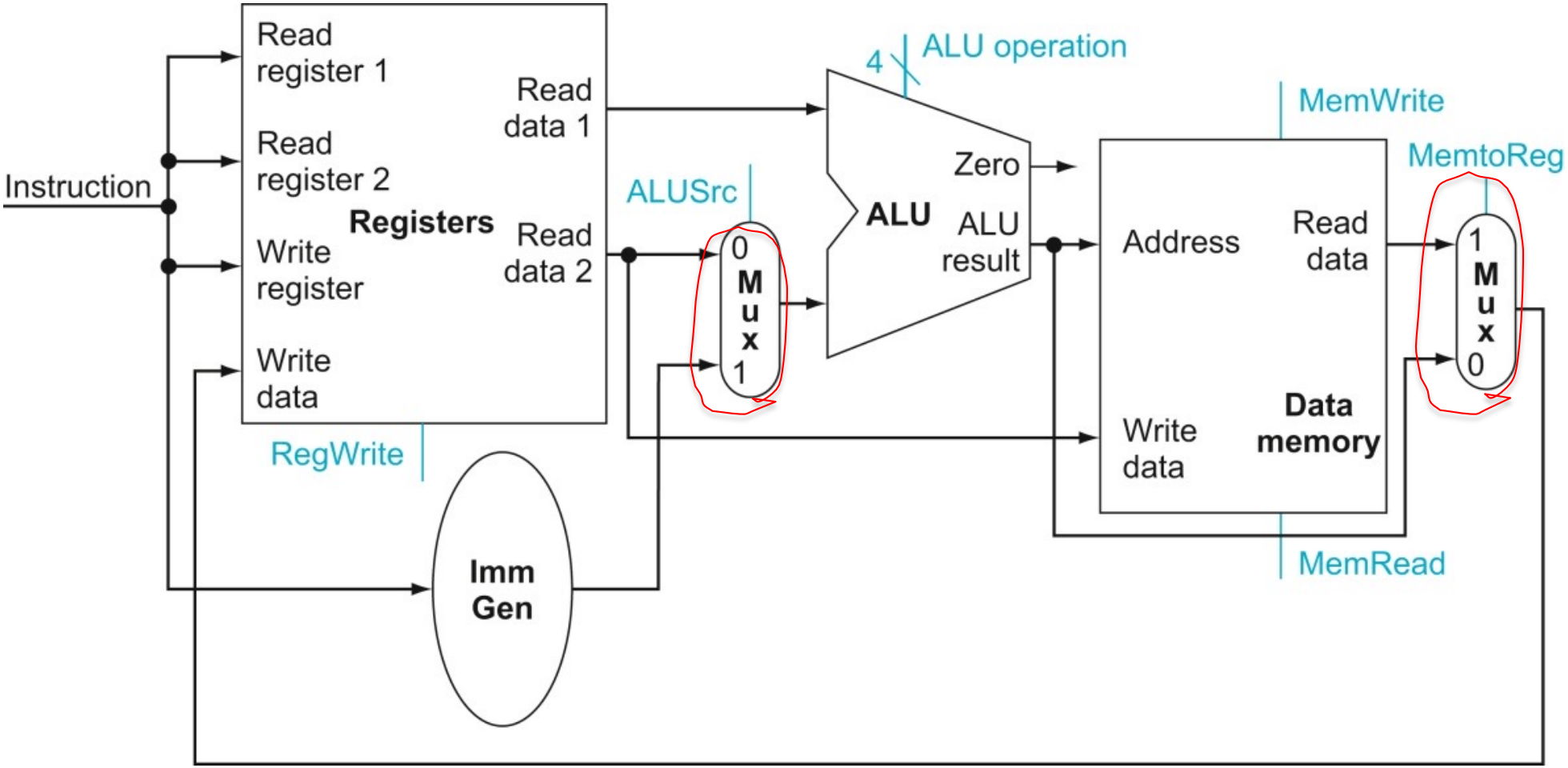
- **beq rs1, rs2, imm**
  - **mem[PC]** **Fetch the instruction from memory**
  - **Zero ← R[rs] == R[rt]** **Calculate the branch condition**
  - **if (Zero)** **Calculate the next instruction's address**
    - **PC ← PC + (SignExt(imm) << 1)**
  - **else**
    - **PC ← PC + 4**

Datapath for Branch Operations

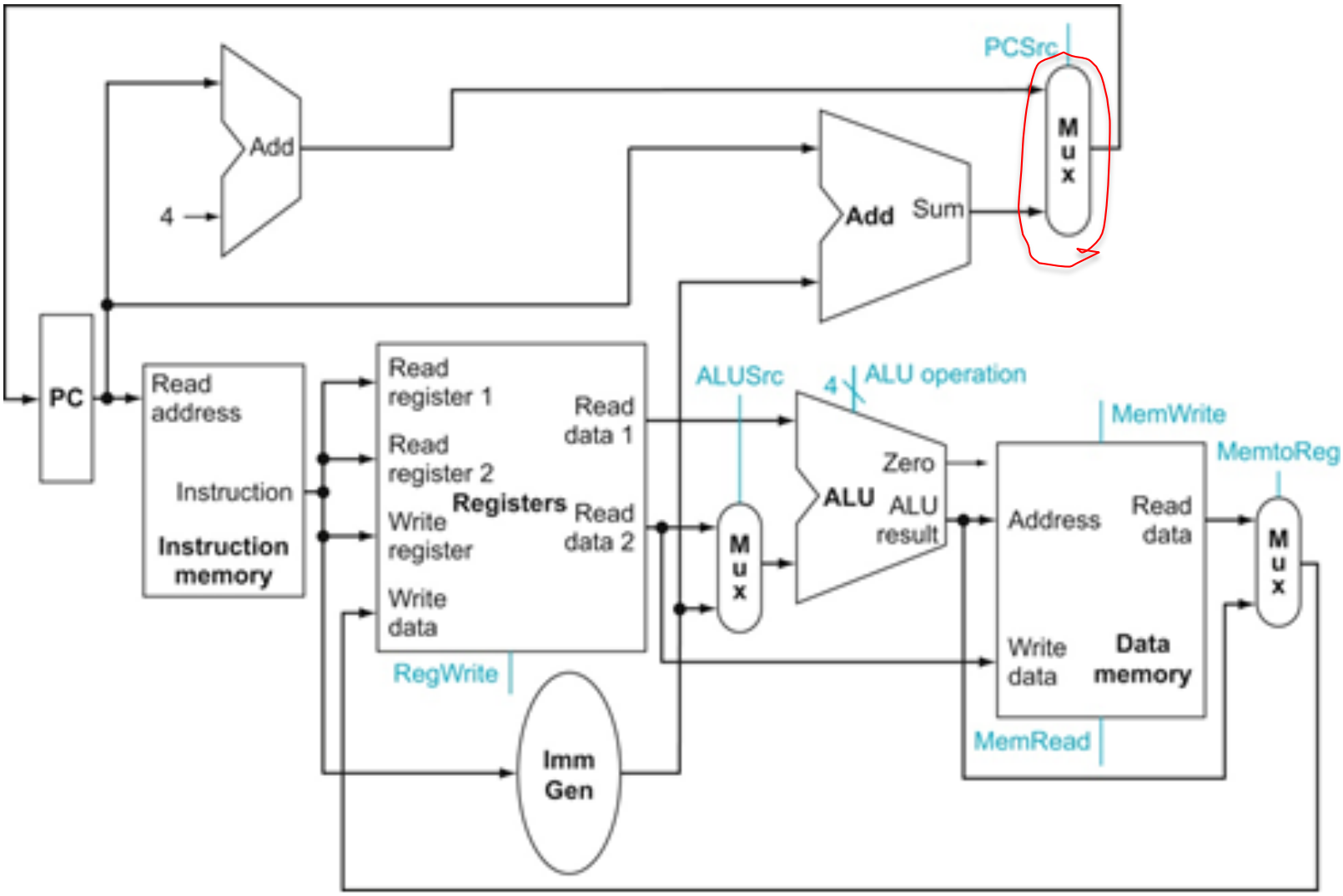


Multiplexor: Stitch Datapath

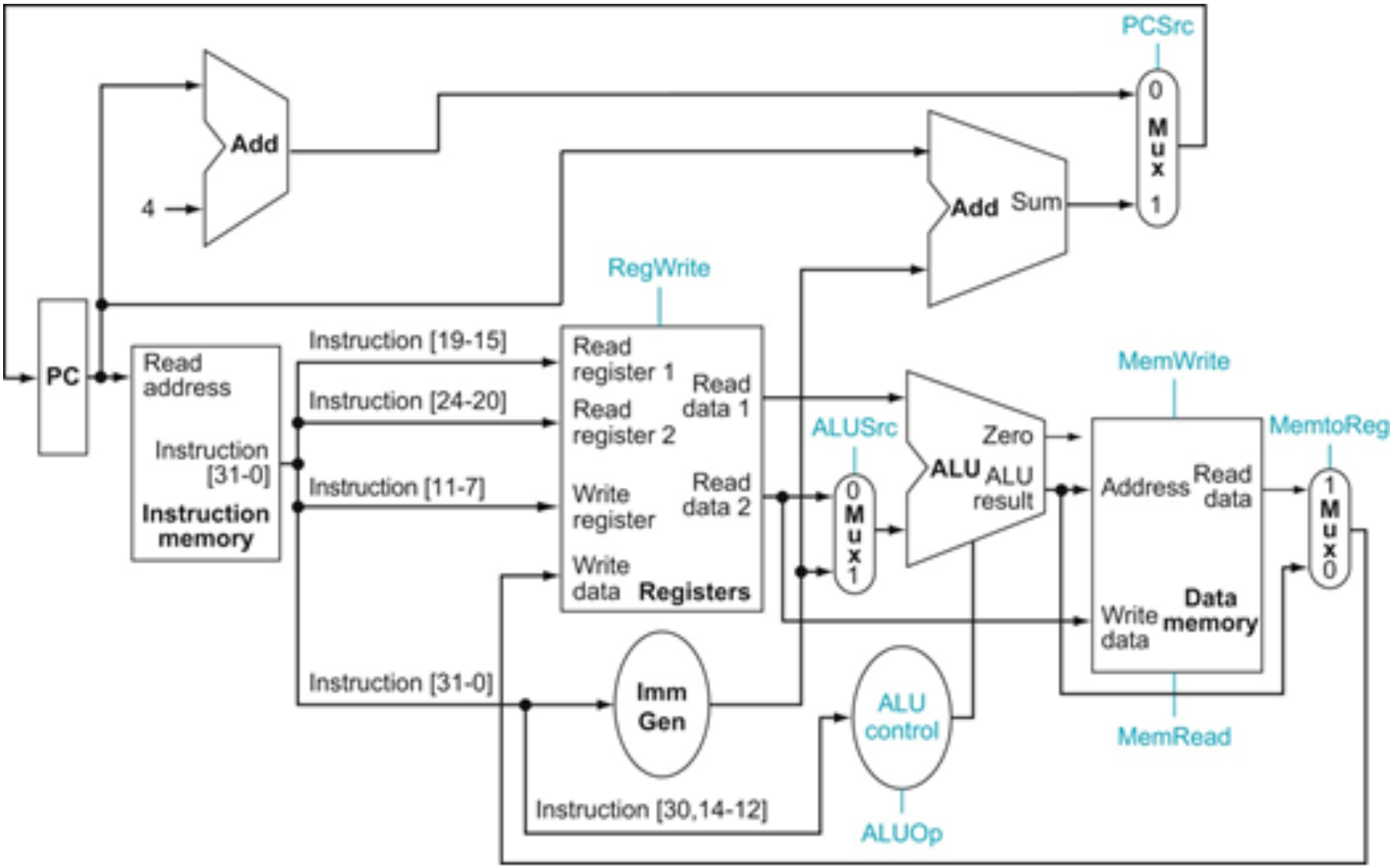
(R-instructions & Memory Access)



Putting it All Together
(with addition of Branch)



Final Datapath



Five Phases

- Create an interconnection of building blocks such that an instruction can flow through the datapath until completion of execution.
- Five steps from start to finish
  1. Fetch the instruction from IM. Get ready for next instruction
  2. Read the register file

Phase	R-type	Data-Transfer	Branch
3. Perform operation	Operate on rs1 and rs2	Address calculation R[rs1] + offset	Compare rs1 and rs2 Calculate BTA
4. Memory Access		Read from or write to data memory	
5. Wrap-Up	Update register file	Update register file for lw	

The Five Phases

1. Fetch instruction from IM, calculate PC+4
2. Read register file
3. Perform operation
4. Memory access
5. Wrap-up