Lecture – 14

The Processor: Datapath and Control

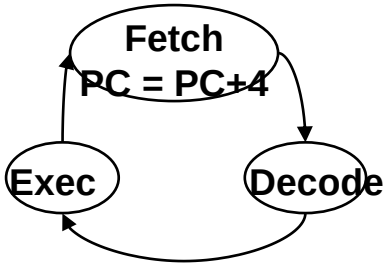
Instruction Execution

For every instruction the first two steps are:

- Send the program counter (PC) to the memory that contains the code and fetch the instruction from that memory.
- Read one or two registers, using the fields of the instruction to select the registers to read.

The Processor: Datapath & Control

- Our implementation of the MIPS is simplified
  - memory-reference instructions: `lw, sw`
  - arithmetic-logical instructions: `add, sub, and, or, slt`
  - control flow / branch instructions: `beq, j`
- Generic implementation
  - use the program counter (PC) to supply the instruction address and fetch the instruction from memory (and update the PC)
  - decode the instruction (and read registers)
  - execute the instruction



- All instructions (except `j`) use the ALU after reading the registers

How? memory-reference? arithmetic? control flow?

Instruction Execution

- All instruction classes, except jump, use the ALU after reading the registers.
- The memory-reference instructions use the ALU for an address calculation.
- The arithmetic-logical instructions use the ALU for operation execution.
- Branch instructions use the ALU for comparison.

Instruction Execution

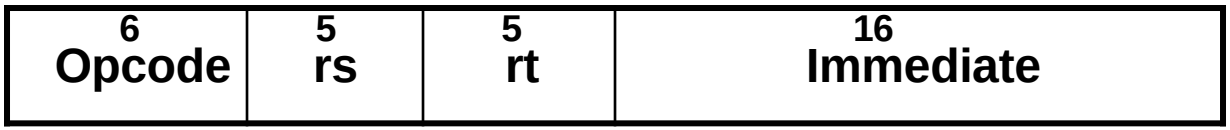
After using the ALU, the actions required to complete various instruction classes differ.

- A memory reference instruction will need to access the memory either to write data for a store or read data for a load.
- Arithmetic-logical instruction must write the data from the ALU back into a register.
- A branch instruction may change the next instruction address based on the comparison, otherwise the PC should be incremented by 4 to get the address of the next instruction.

MIPS Instruction Format

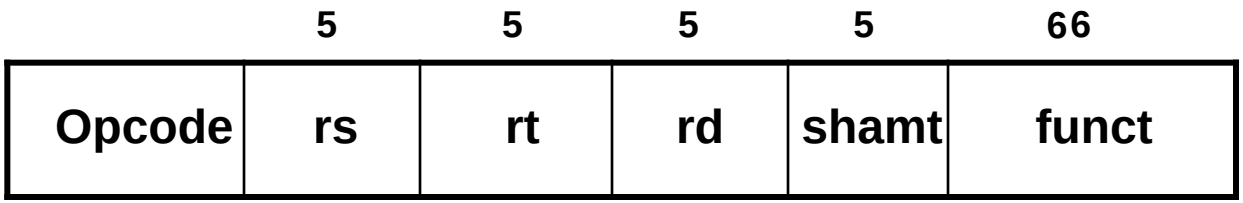
- **All instructions are 32 bits with a 6-bit primary opcode.**

I-type instruction (Transfer, branch, imm. format)



Example: ADDI Rs, Rt, IMM

R-type instruction (Arithmetic instruction format)



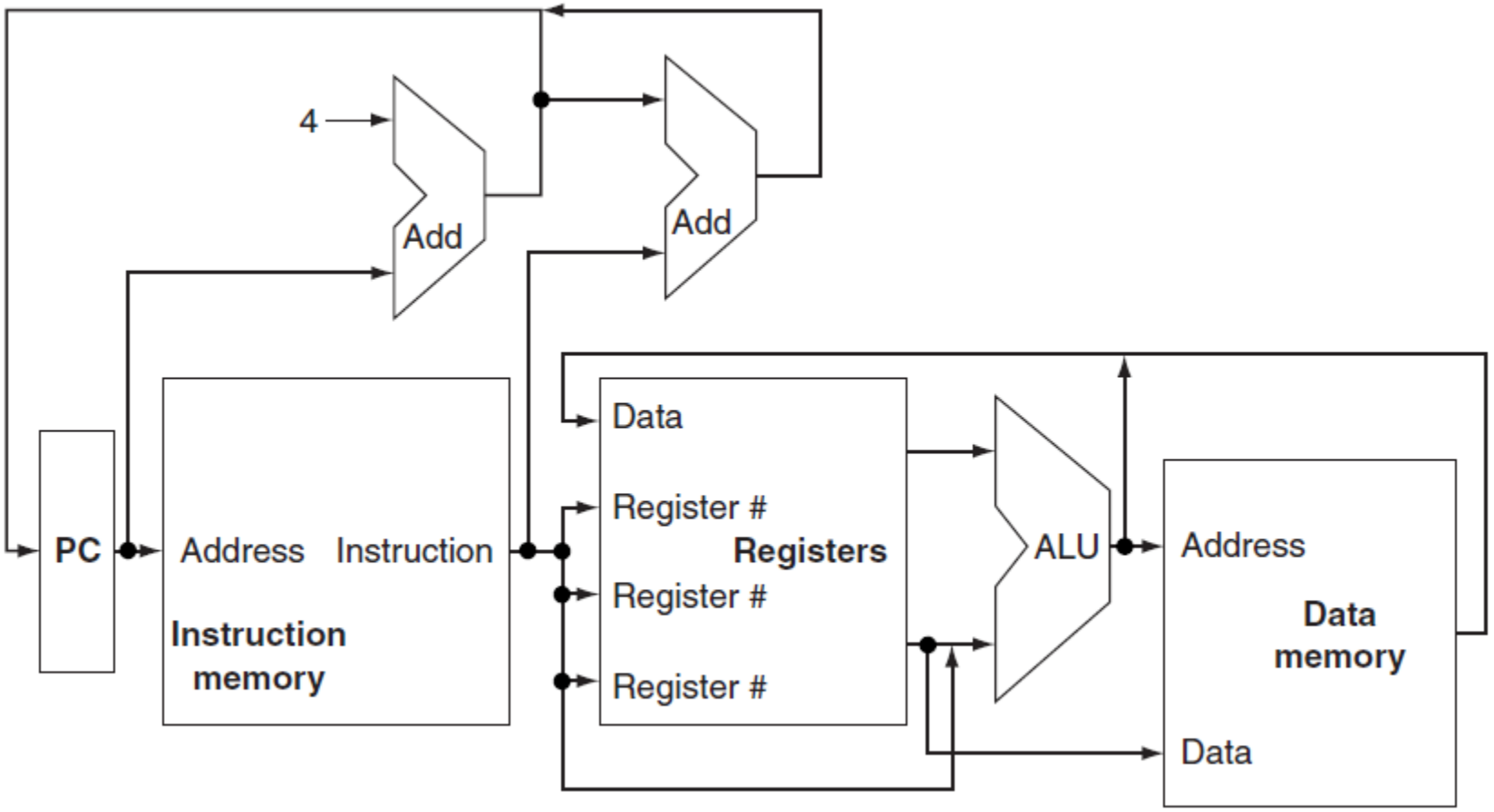
Example: ADD Rd, Rs, Rt
SLL Rd, Rt, Shamt

MIPS Instruction Format

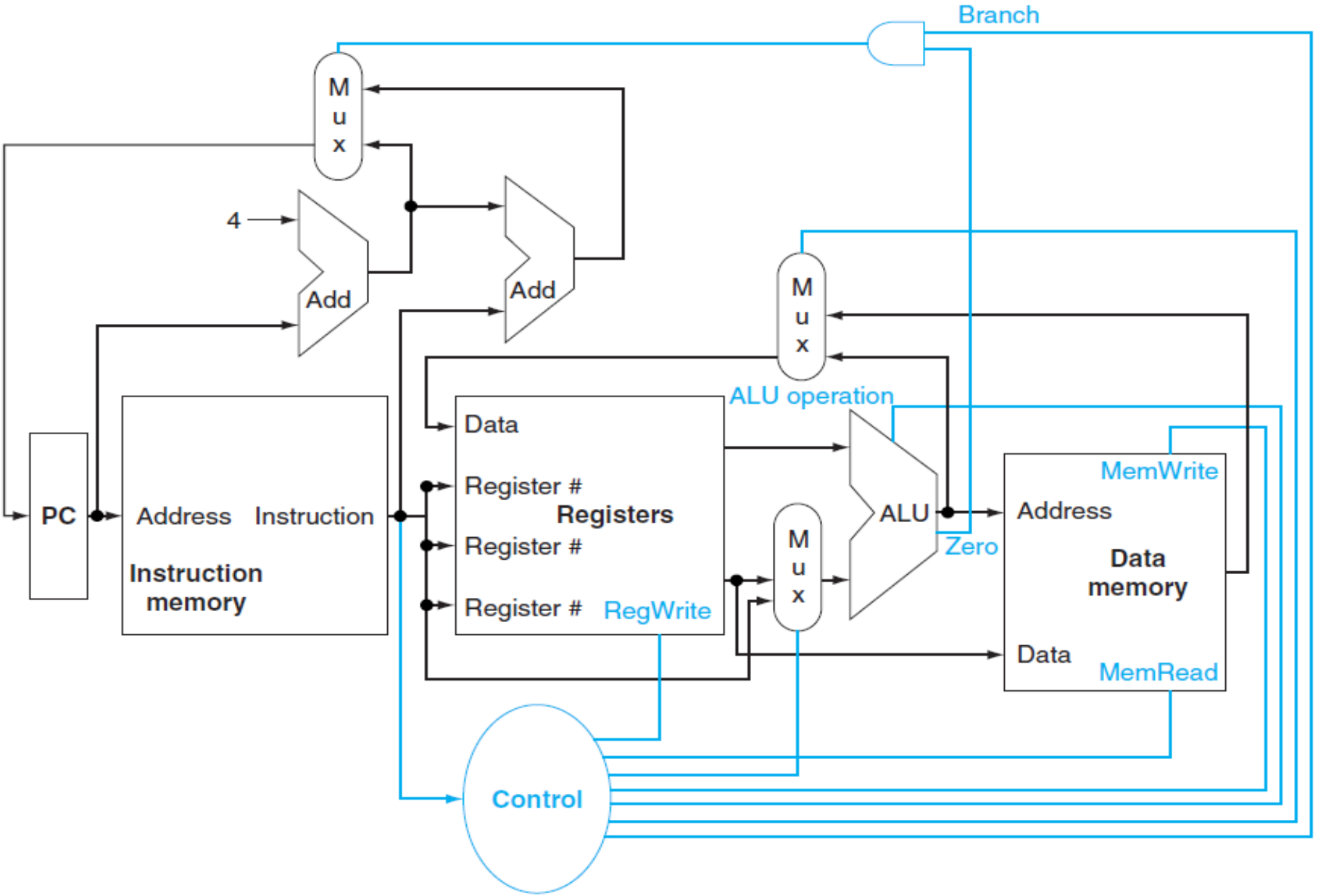
J-type instruction (Jump instruction format)

Example: J ADR

An abstract view of the implementation of the MIPS subset



The basic implementation of the MIPS subset



The Single Cycle Datapath

- Takes a single long clock cycle for every instruction.
- Every instruction begins execution on one clock edge and completes execution on the next clock edge.

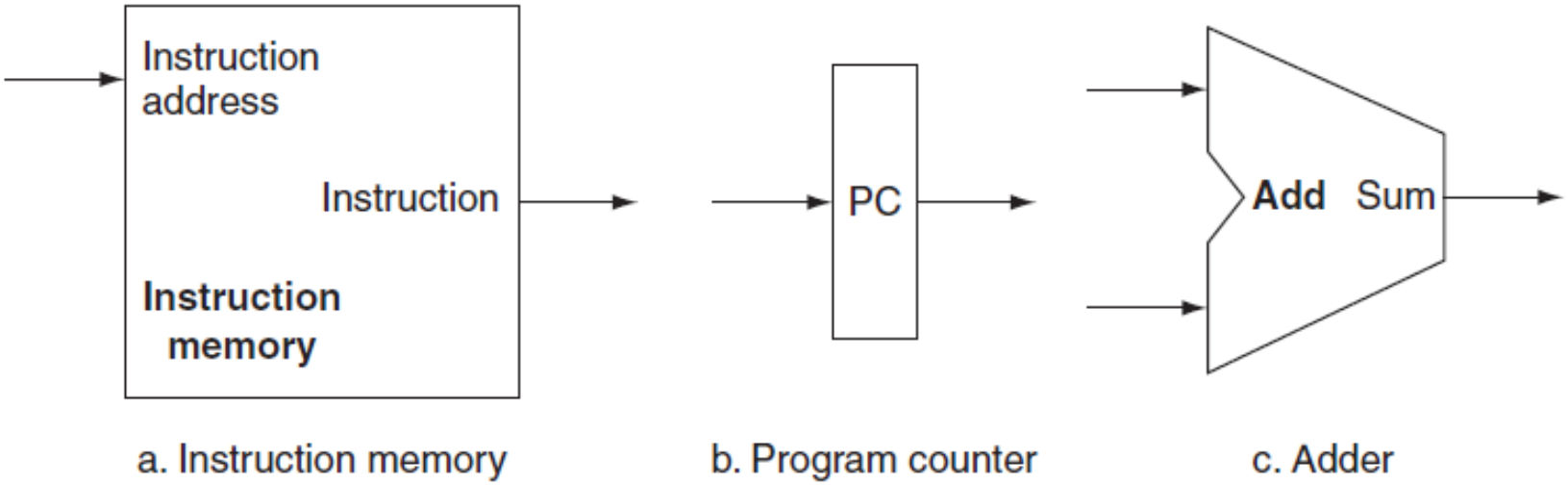
Reasons for separate Memory and Data Unit:

- The format of data and instruction is different in MIPS.
- Less Expensive.
- The processor operates in one clock cycle and cannot use a single-ported memory for two different accesses within that cycle.

Logic Design Conventions

Read Yourself

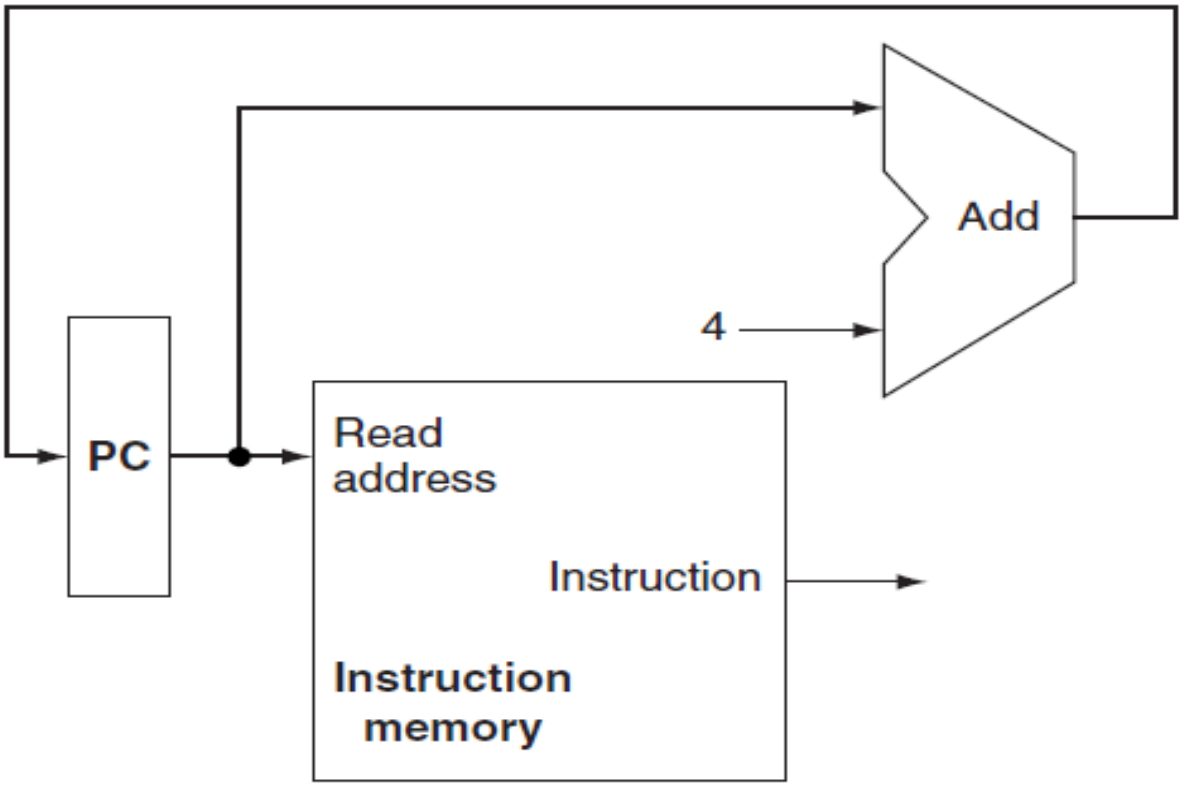
Building a Datapath



a. Instruction memory b. Program counter c. Adder

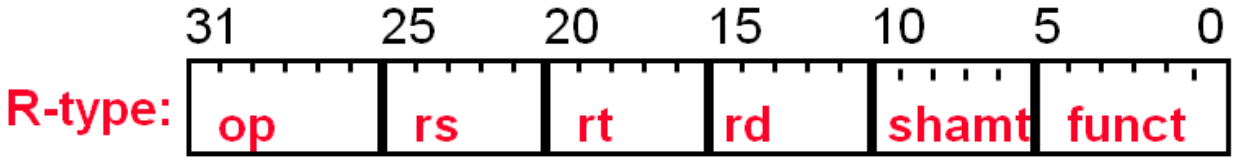
- Two state elements (instruction memory and program counter) are needed to store and access instructions.
- An adder is needed to compute the next instruction address.

Building a Datapath

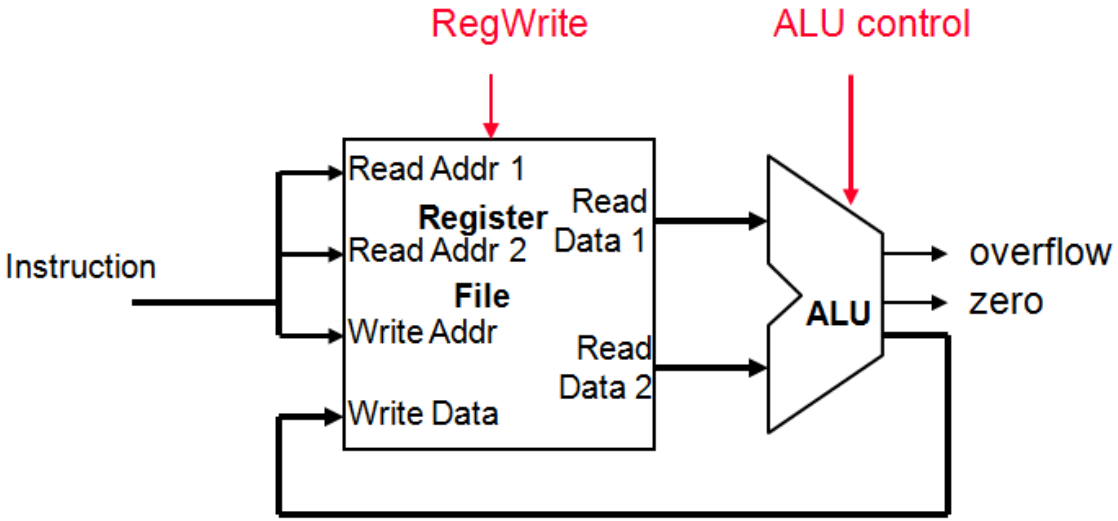


- It fetches instructions and increments the PC to obtain the address of the next sequential instruction.

R format operations (`add, sub, slt, and, or`)

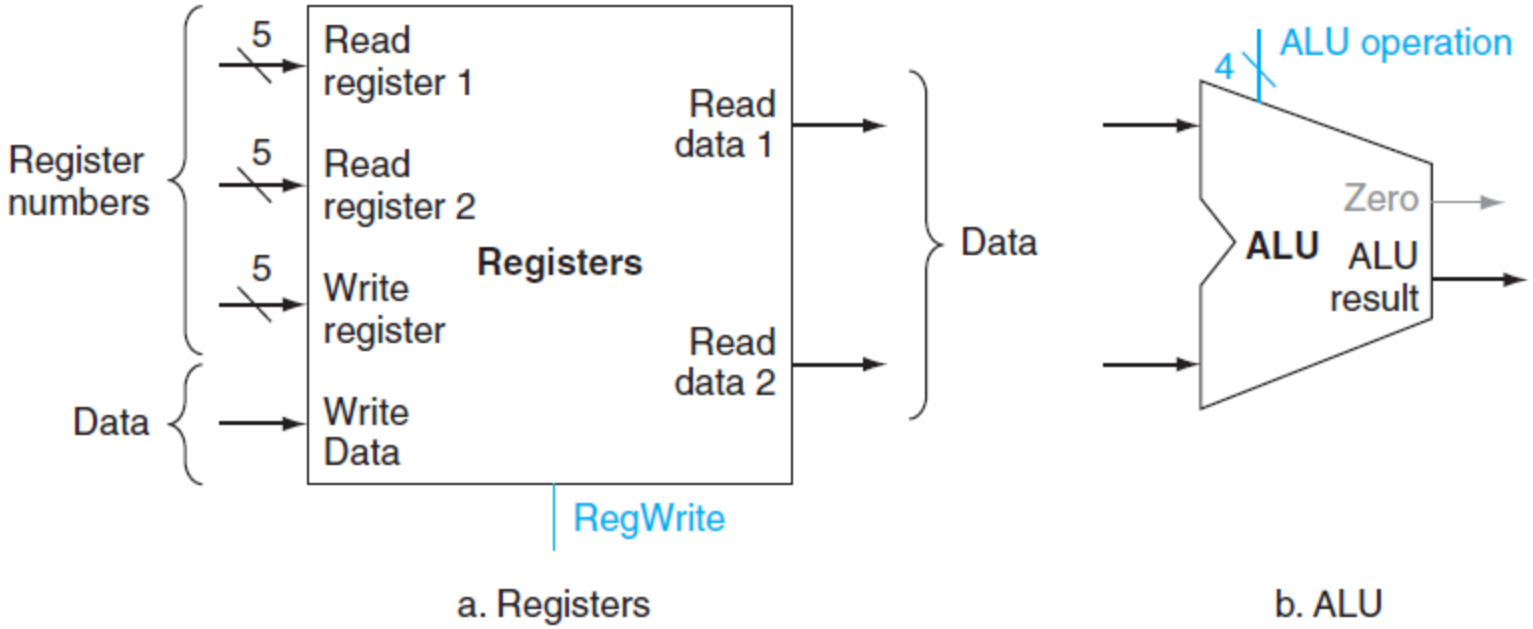


- perform the (op and funct) operation on values in rs and rt
- store the result back into the Register File (into location rd)

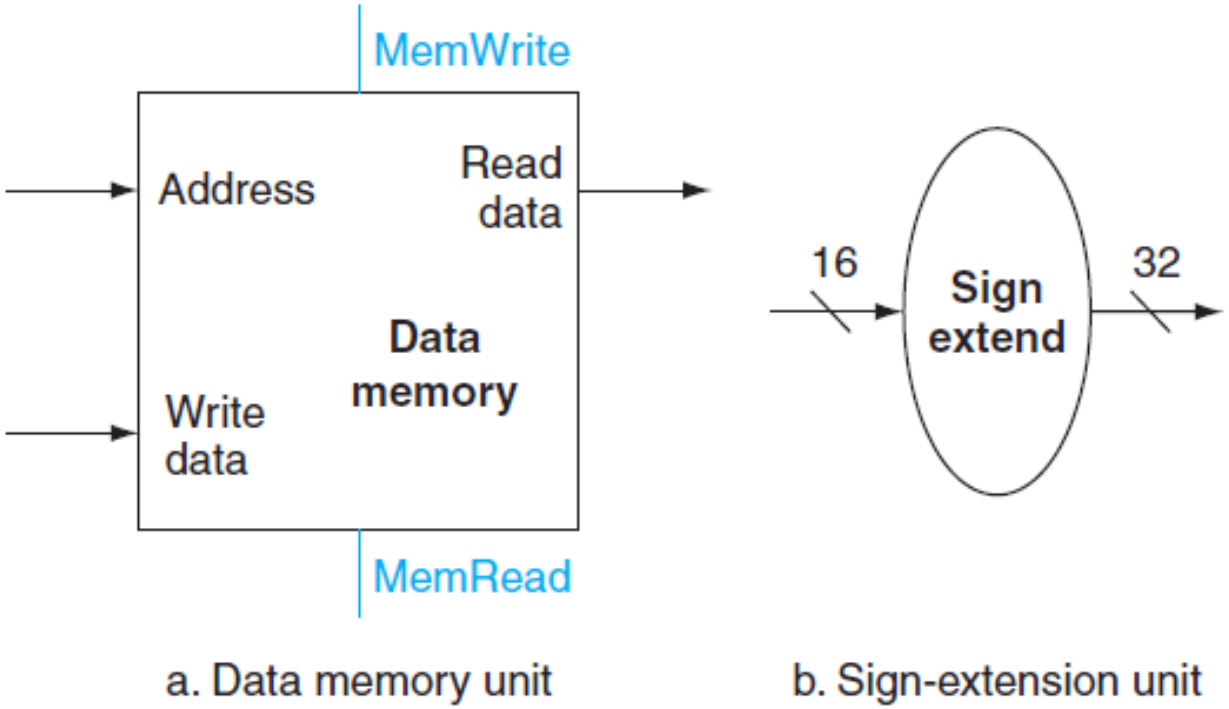


- The Register File is not written every cycle (e.g. **`sw`**), so we need an explicit write control signal for the Register File

Data elements needed to implement R-type instructions



Data elements needed to implement Load and Store

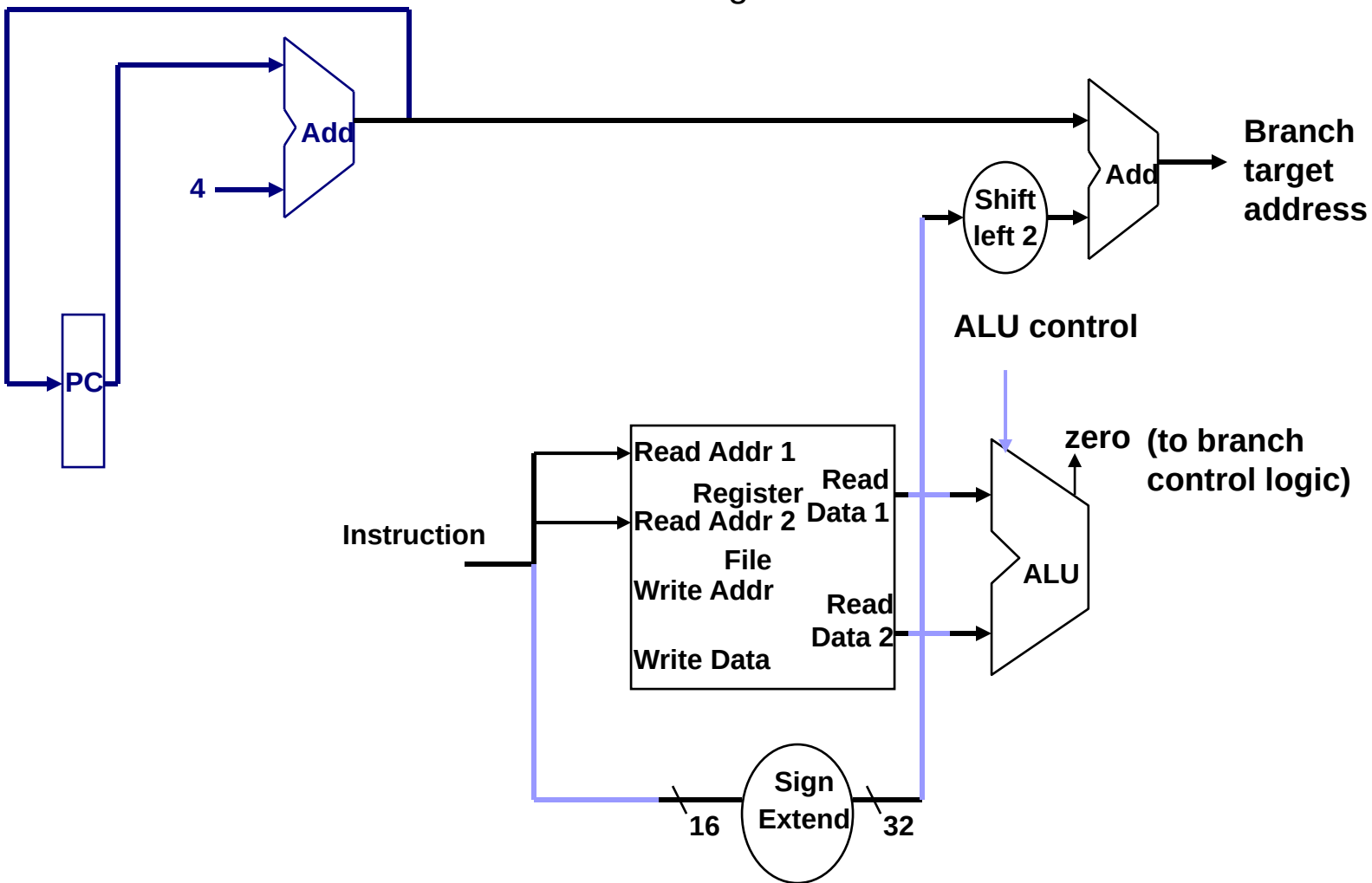


a. Data memory unit

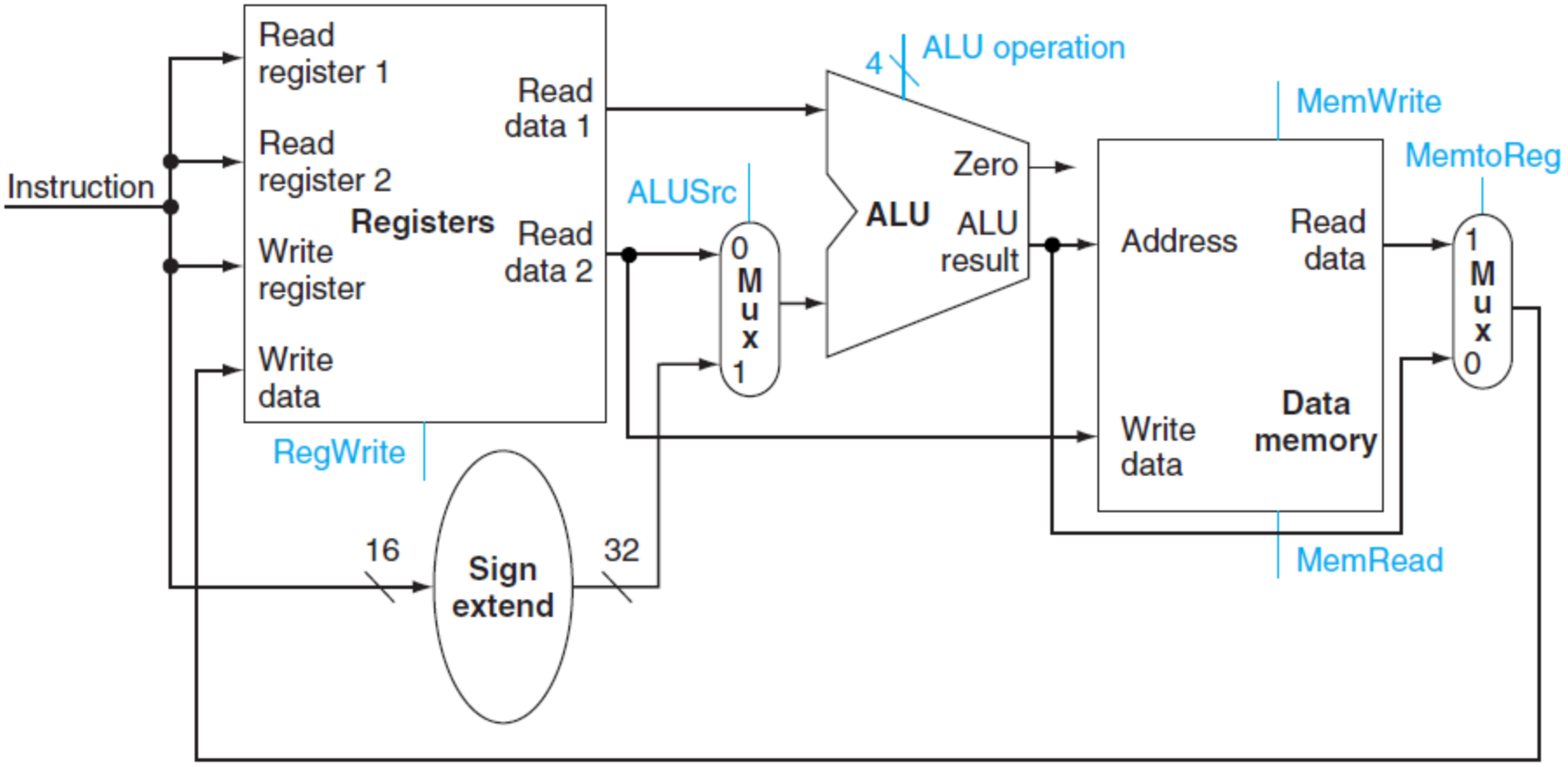
b. Sign-extension unit

Executing Branch Operations

- Branch operations involves
  - compare the operands read from the Register File during decode for equality (`zero` ALU output)
  - compute the branch target address by adding the updated PC to the 16-bit signed-extended offset field in the instr



Datapath for the Memory Instruction and R-type instruction



A Simple Datapath for the MIPS Architecture