The Processor: Datapath & Control

- We're ready to look at an implementation of the MIPS
- Simplified to contain only:
  - memory-reference instructions: `lw, sw`
  - arithmetic-logical instructions: `add, sub, and, or, slt`
  - control flow instructions: `beq, j`

 Generic Implementation:

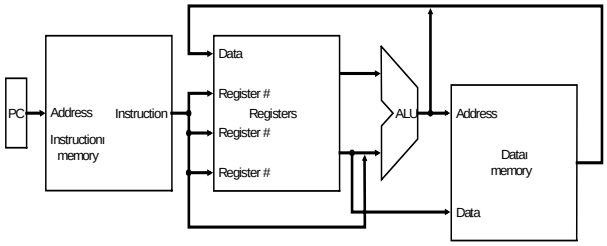
  - use the program counter (PC) to supply instruction address
  - get the instruction from memory
  - read registers
  - use the instruction to decide exactly what to do

 All instructions use the ALU after reading the registers

Why? memory-reference? arithmetic? control flow?

More Implementation Details

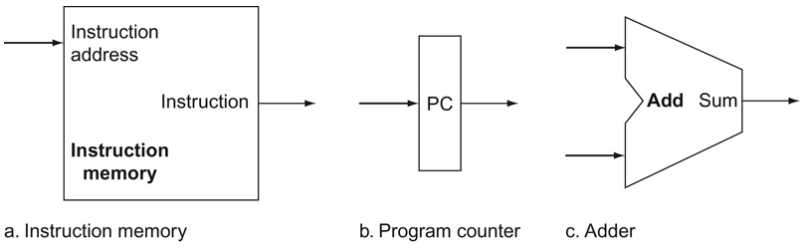
- Abstract / Simplified View:



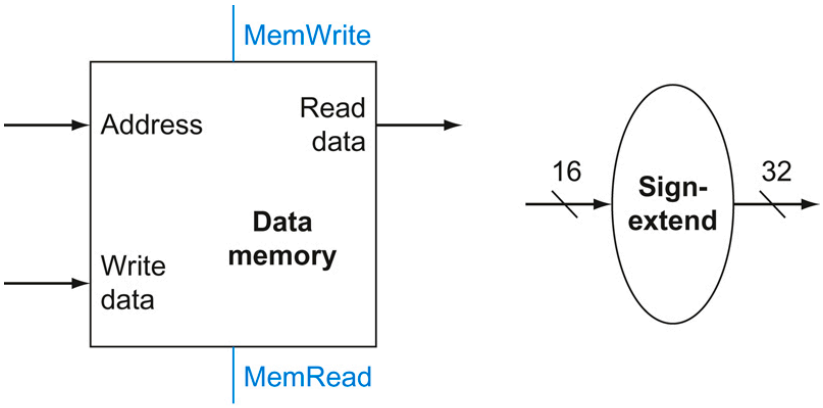
Two types of functional units:

- elements that operate on data values (combinational)
- elements that contain state (sequential)

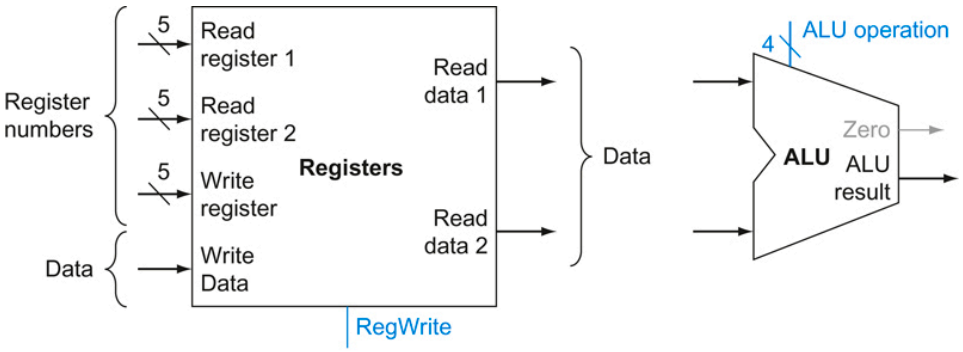
Major components



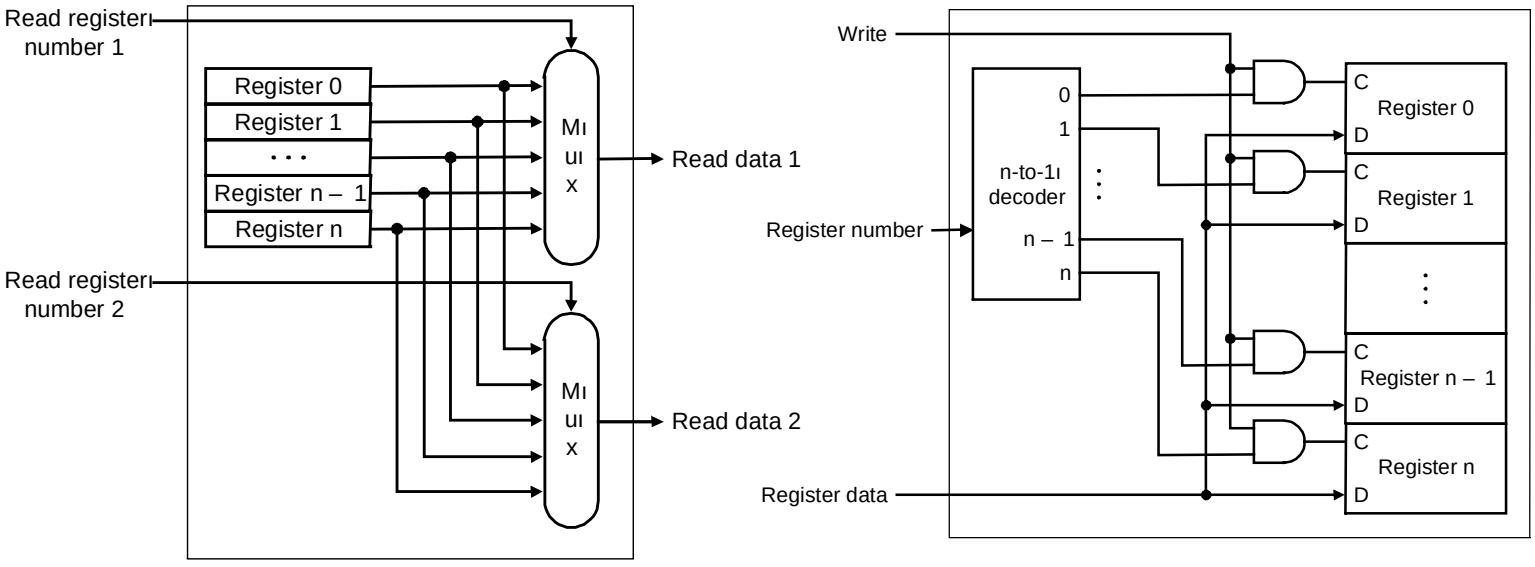
a. Instruction memory b. Program counter c. Adder



a. Data memory unit b. Sign extension unit

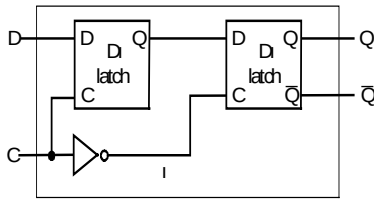


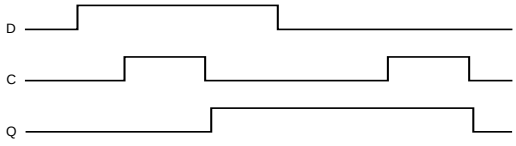
a. Registers b. ALU



D flip-flop

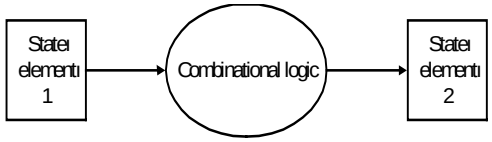
- **Output changes only on the clock edge**

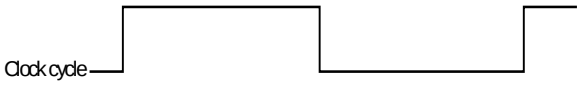




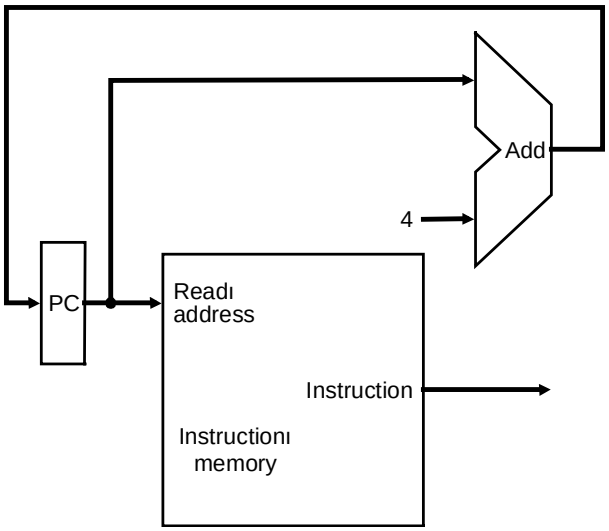
Our Implementation

- **An 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**

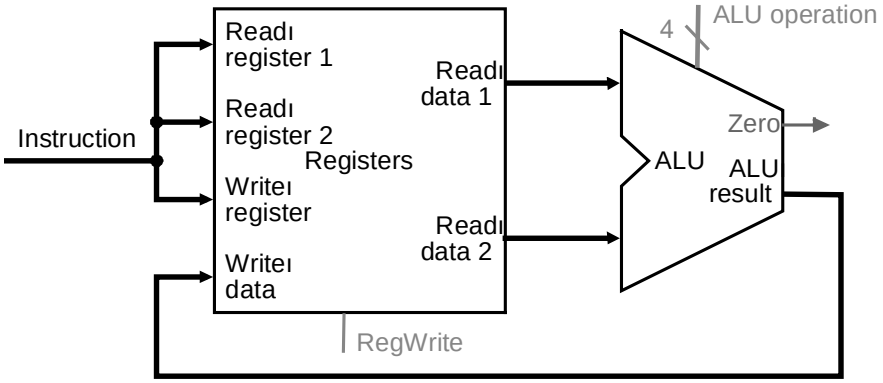




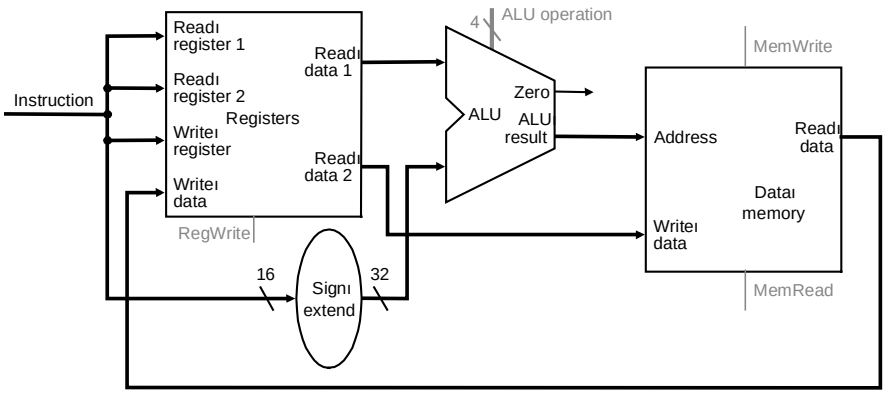
Fetching instructions and incrementing PC



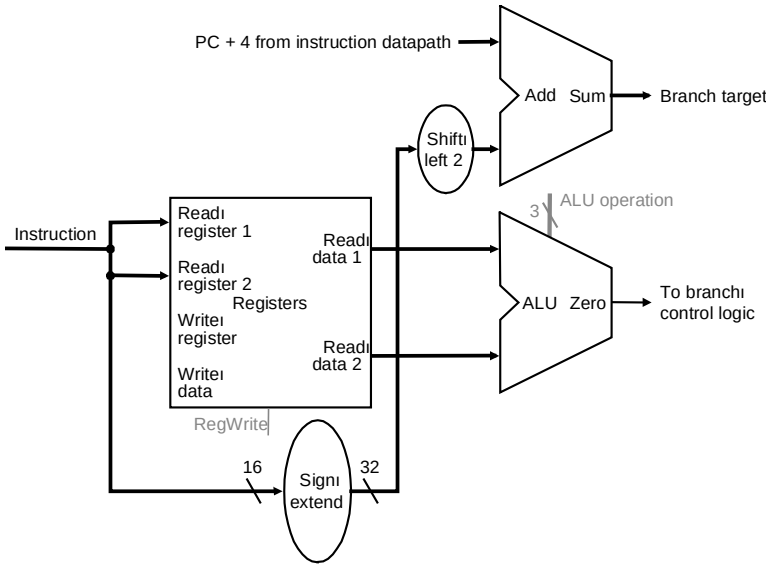
Datapath for R-type instructions



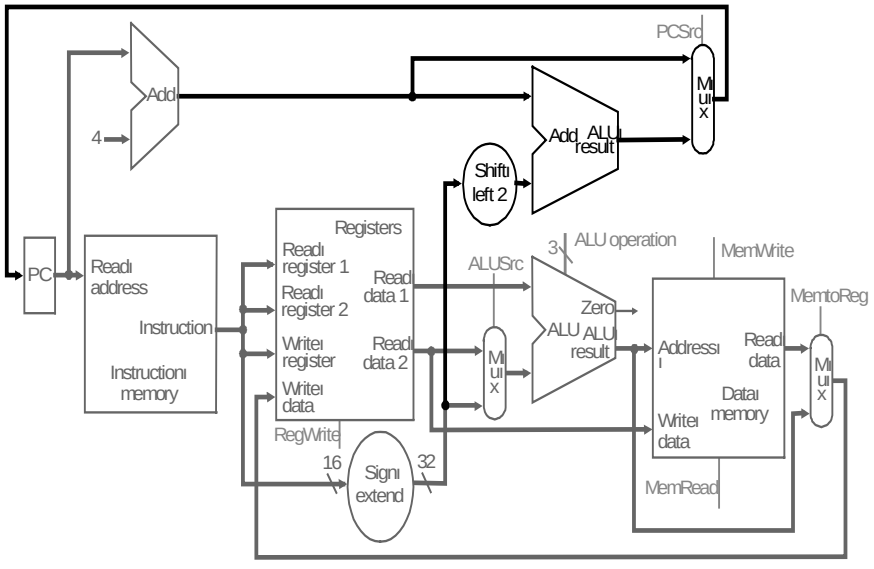
Datapath for load and store



Datapath for branching instructions



Building the Datapath



Complete Datapath