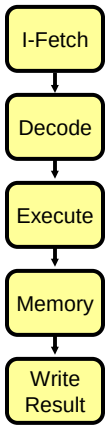
Lecture 8: Datapaths and Pipelining

- **Last Time**
  - **Last of the ISA lectures**
- **Today**
  - **Role of compiler**
  - **Datapath organization**
  - **Datapath control**

Instruction Execution

- **5 basic steps**
  - **fetch instruction (F)**
  - **decode instruction and read registers (R)**
  - **execute (X)**
  - **access memory (M)**
  - **store result (W)**

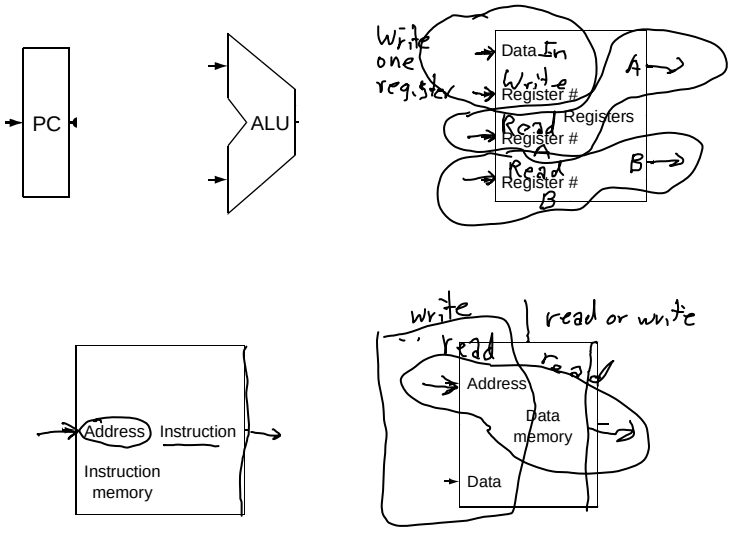


The Processor: Datapath & Control

- 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

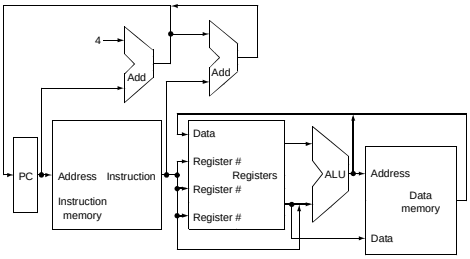
  Why? memory-reference? arithmetic? control flow?

Pieces we'll need



More Implementation Details

- **Abstract / Simplified View:**

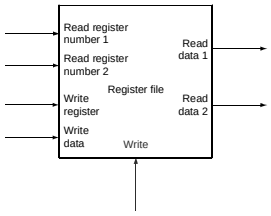


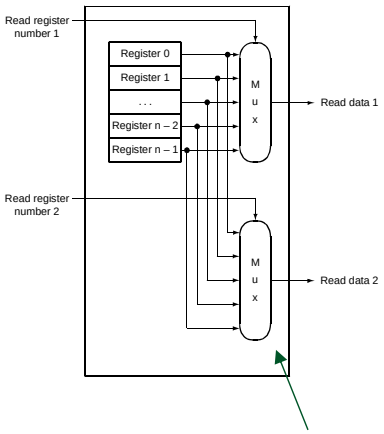
Two types of functional units:

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

Register File

- **Built using D flip-flops**

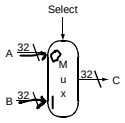


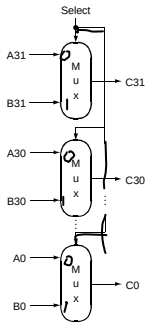


Do you understand? What is the "Mux" above?

Abstraction

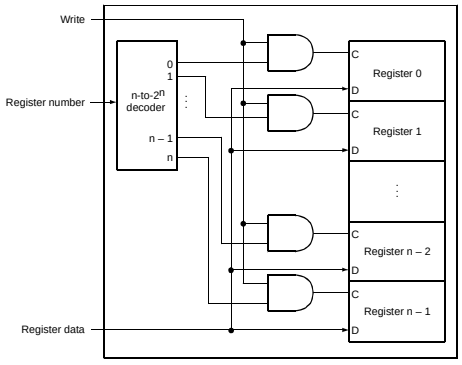
- **Make sure you understand the abstractions!**
- **Sometimes it is easy to think you do, when you don't**





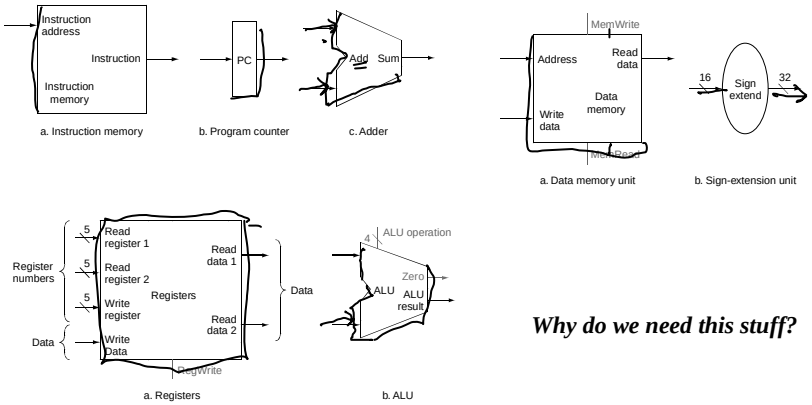
Register File

- **Note: we still use the real clock to determine when to write**



Simple Implementation

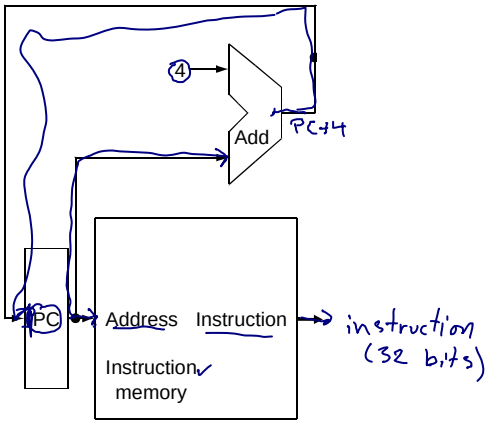
- **Include the functional units we need for each instruction**



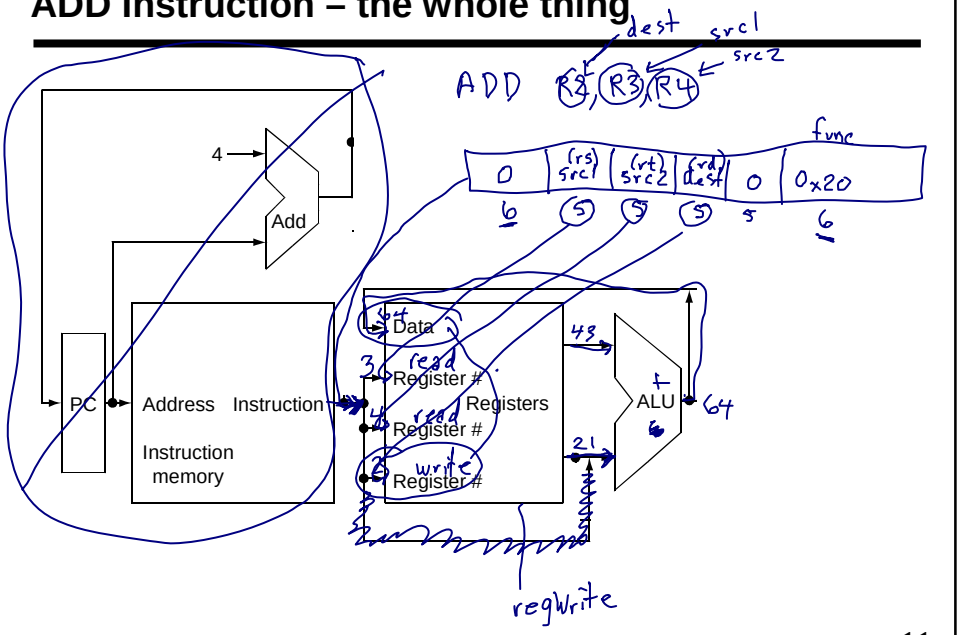
a. Instruction memory | b. Program counter | c. Adder | a. Data memory unit | b. Sign-extension unit | a. Registers | b. ALU

Why do we need this stuff?

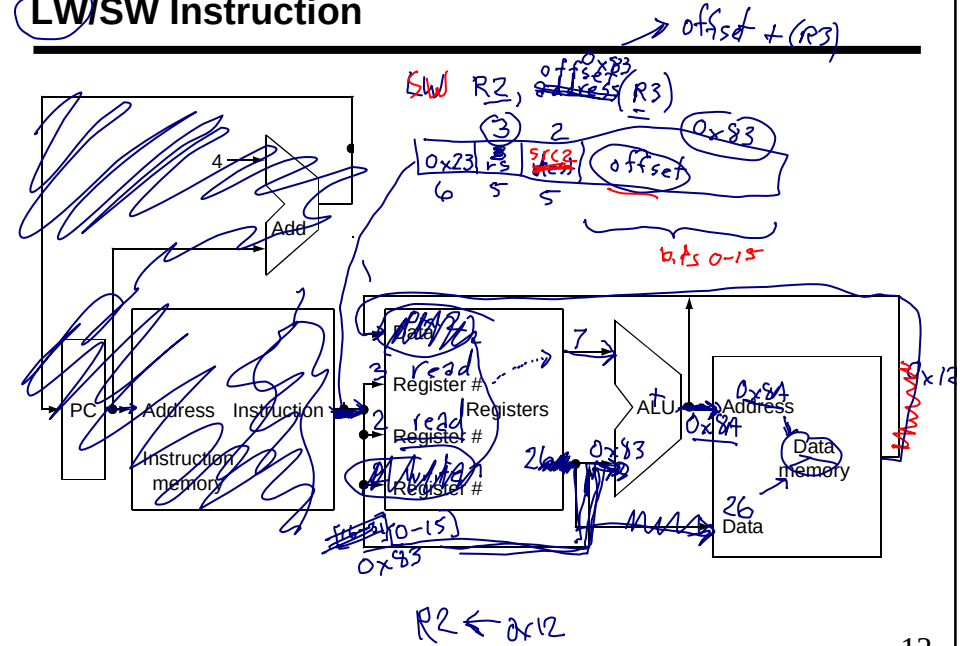
ADD Instruction – instruction fetch



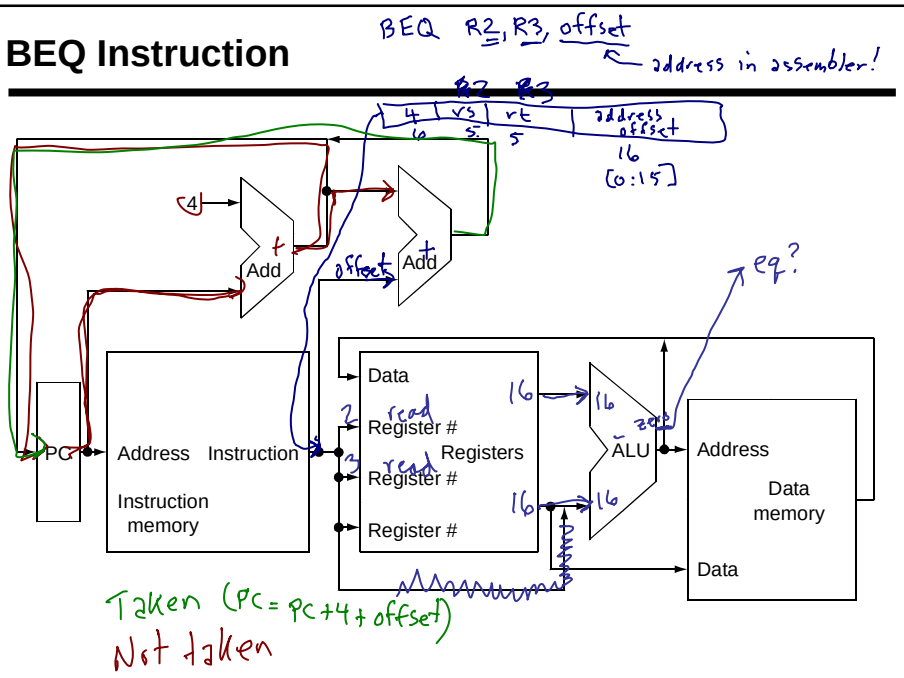
ADD Instruction – the whole thing



LW/SW Instruction

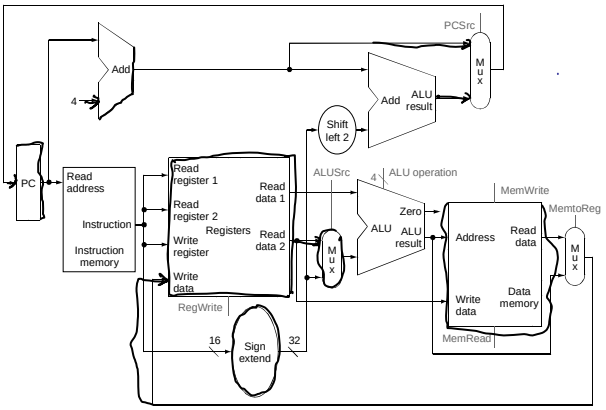


BEQ Instruction



Building the Datapath

- **Use multiplexors to stitch them together**



Control

- Selecting the operations to perform (ALU, read/write, etc.)
- Controlling the flow of data (multiplexor inputs)
- Information comes from the 32 bits of the instruction
- Example:

add $8, $17, $18 **Instruction Format:**

000000	10001	10010	01000	00000	100000
op	rs	rt	rd	shamt	funct

- ALU's operation based on instruction type and function code

