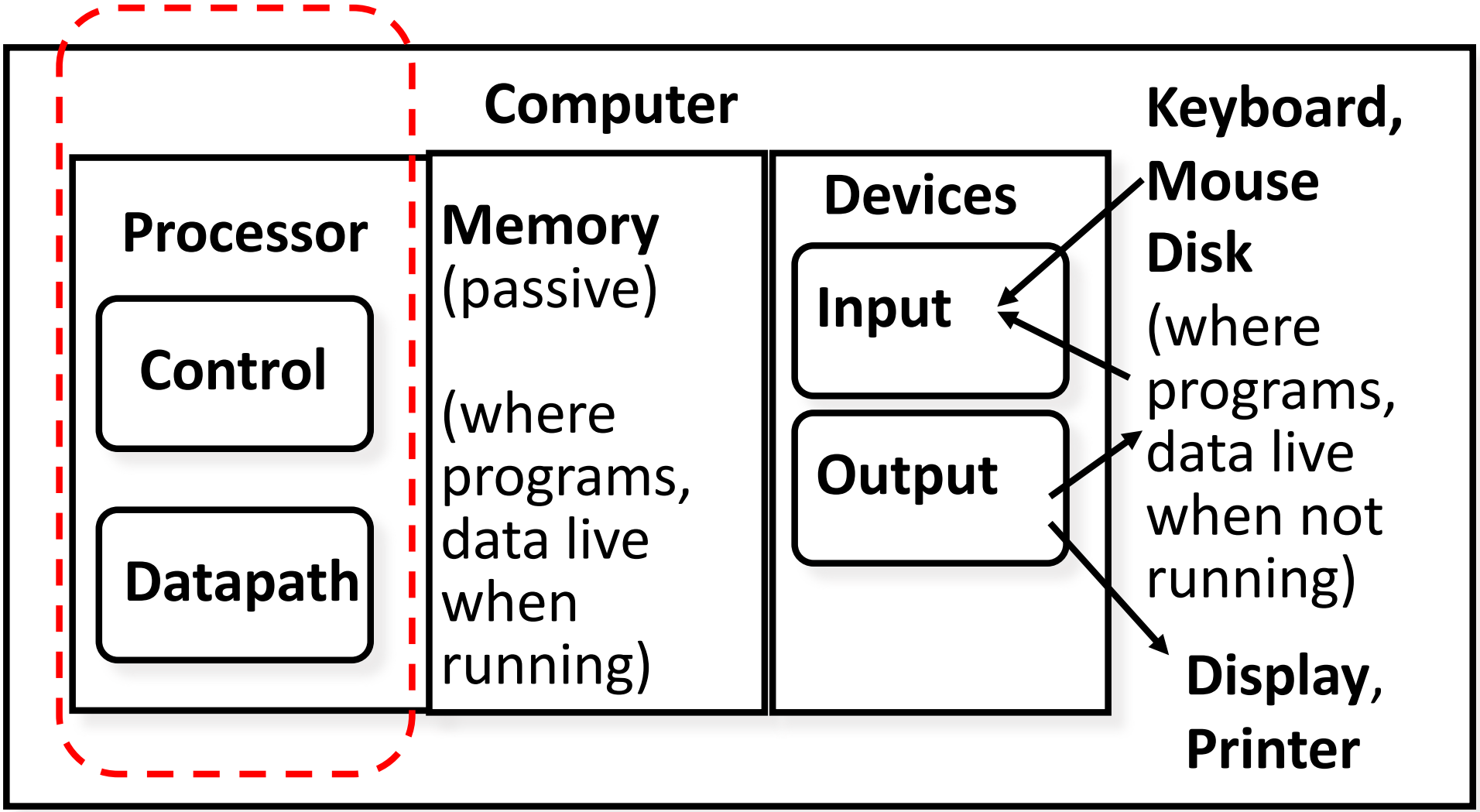
Lecture 11: Single Cycle CPU

(CPEG323: Intro. to Computer System Engineering)

Five Components of a Computer



The Processor

- ***Processor* (CPU):** the active part of the computer, which does all the work (data manipulation and decision-making)
- ***Datapath*:** portion of the processor which contains hardware necessary to perform operations required by the processor
- ***Control*:** portion of the processor which tells the datapath what needs to be done

Stages of the Datapath : Overview

- Break up the process of "executing an instruction" into *stages* or *phases*, and then connect the phases to create the whole datapath
  - Smaller phases are easier to design
  - Easy to optimize (change) one phase without touching the others

There is a wide variety of MIPS instructions: so what general steps do they have in common?

Phases of the Datapath (1/5)

- **Phase 1: *Instruction Fetch* (IF)**
  - No matter what the instruction, the 32-bit instruction word must first be fetched from memory
  - Also, this is where we increment PC (that is, PC = PC + 4, to point to the next instruction)

Phases of the Datapath (2/5)

- **Phase 2: *Instruction Decode* (ID)**
  - Upon fetching the instruction, we next gather data from the fields
  - Read the opcode and each of the possible fields from the 32 bits of the instruction
  - Read data from all necessary registers
    - For add, read two registers
    - For addi, read one register
    - For jal, no reads necessary

Phases of the Datapath (3/5)

- **Phase 3: *Execution* (EXE)**
  - Real work of most instructions is done here: arithmetic (+, -, *, /), shifting, logic (&, |), comparisons (slt)
  - What about loads and stores?
    - lw $t0, 40($t1)
    - Memory address = Offset (i.e., 40) + Value in $t1
    - We perform this addition in this stage.

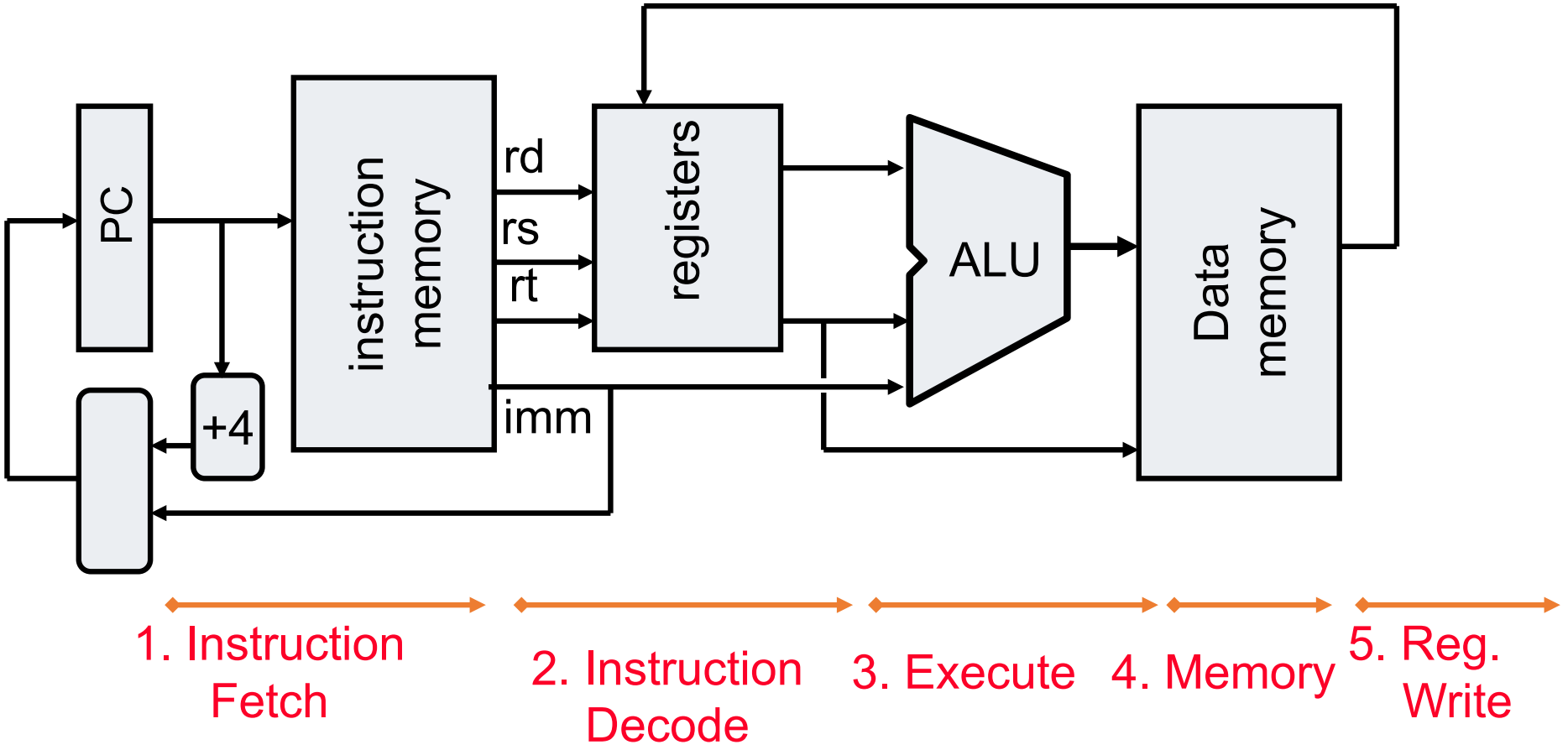
Phases of the Datapath (4/5)

- **Phase 4: *Memory Access* (MEM)**
  - Only the load and store instructions do anything during this phase; the others remain idle or skip this this phase all together
  - Since these instructions have a unique step, we need this extra phase to account for them
  - As a result of the cache system, this phase is expected to be fast

Phases of the Datapath (5/5)

- **Phase 5: *Write Back* (WB)**
  - Most instructions write the result of some computation into a register
    - E.g.,: arithmetic, logical, shifts, loads, slt
  - What about stores, branches, jumps?
    - Don't write anything into a register at the end
    - These remain idle during this fifth phase or skip it all together

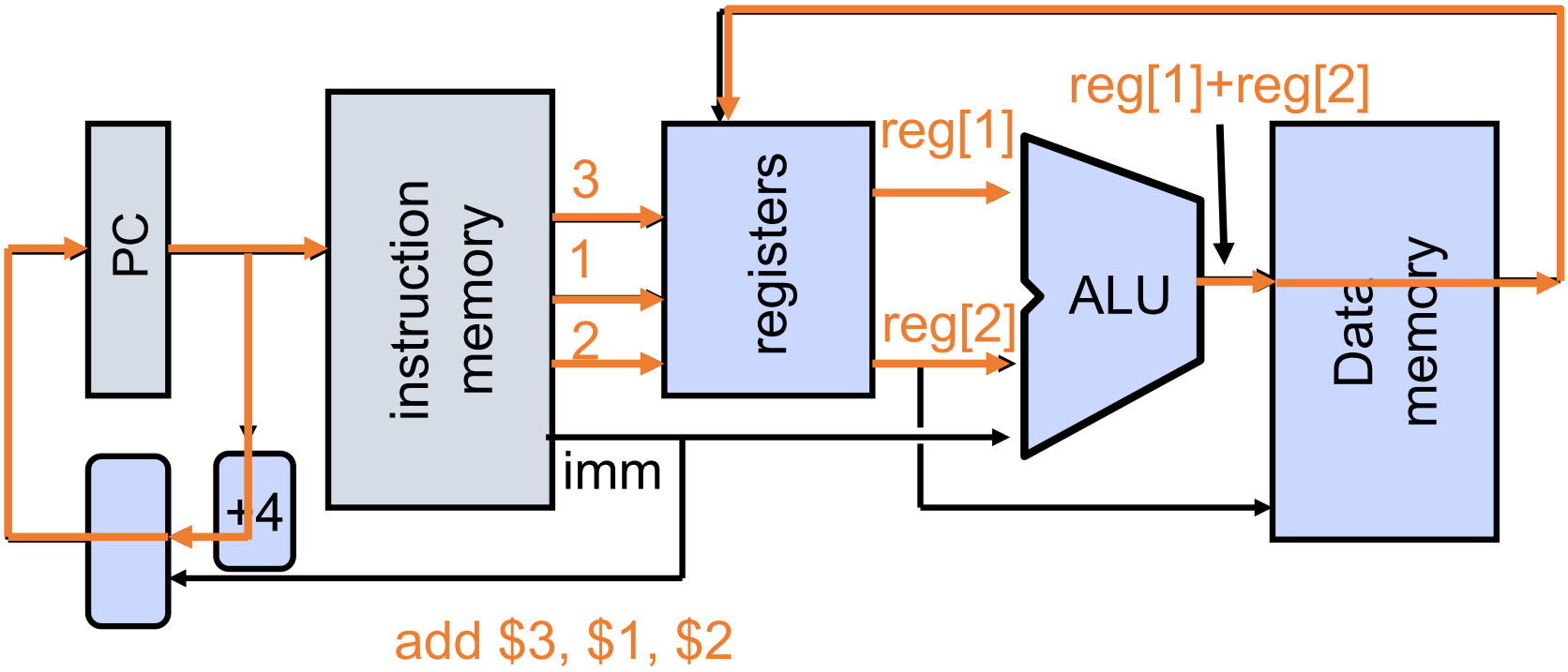
General Steps of Datapath



Datapath Walkthroughs (1/3)

- add $r3,$r1,$r2
  - Stage 1: fetch this instruction, inc. PC
  - Stage 2: decode to find it's an add, then read registers $r1 and $r2
  - Stage 3: add the two values retrieved in Stage 2
  - Stage 4: idle (nothing to write to memory)
  - Stage 5: write result of Stage 3 into register $r3

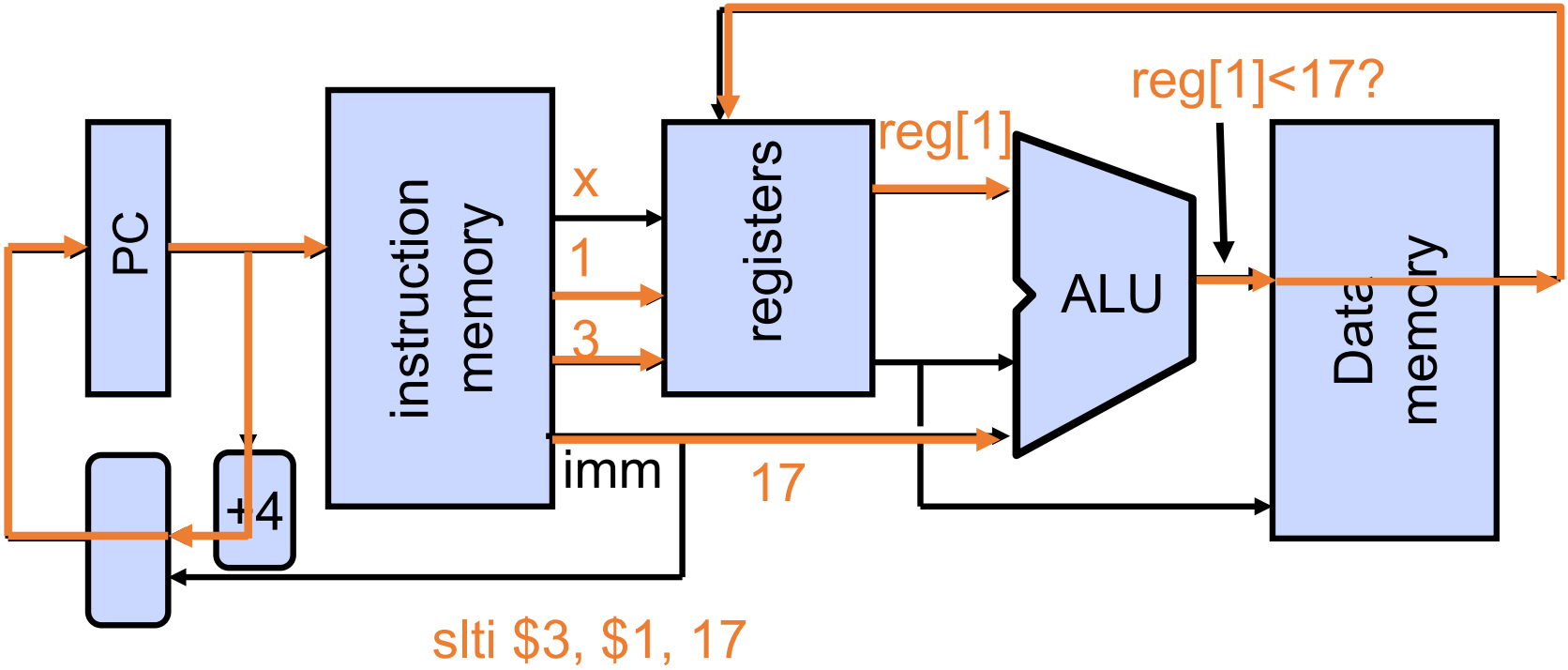
Example: **add** Instruction



Datapath Walkthroughs (2/3)

- slti $r3,$r1,17
  - Stage 1: fetch this instruction, inc. PC
  - Stage 2: decode to find it's an slti, then read register $r1
  - Stage 3: compare value retrieved in Stage 2 with the integer 17
  - Stage 4: idle
  - Stage 5: write the result of Stage 3 in register $r3

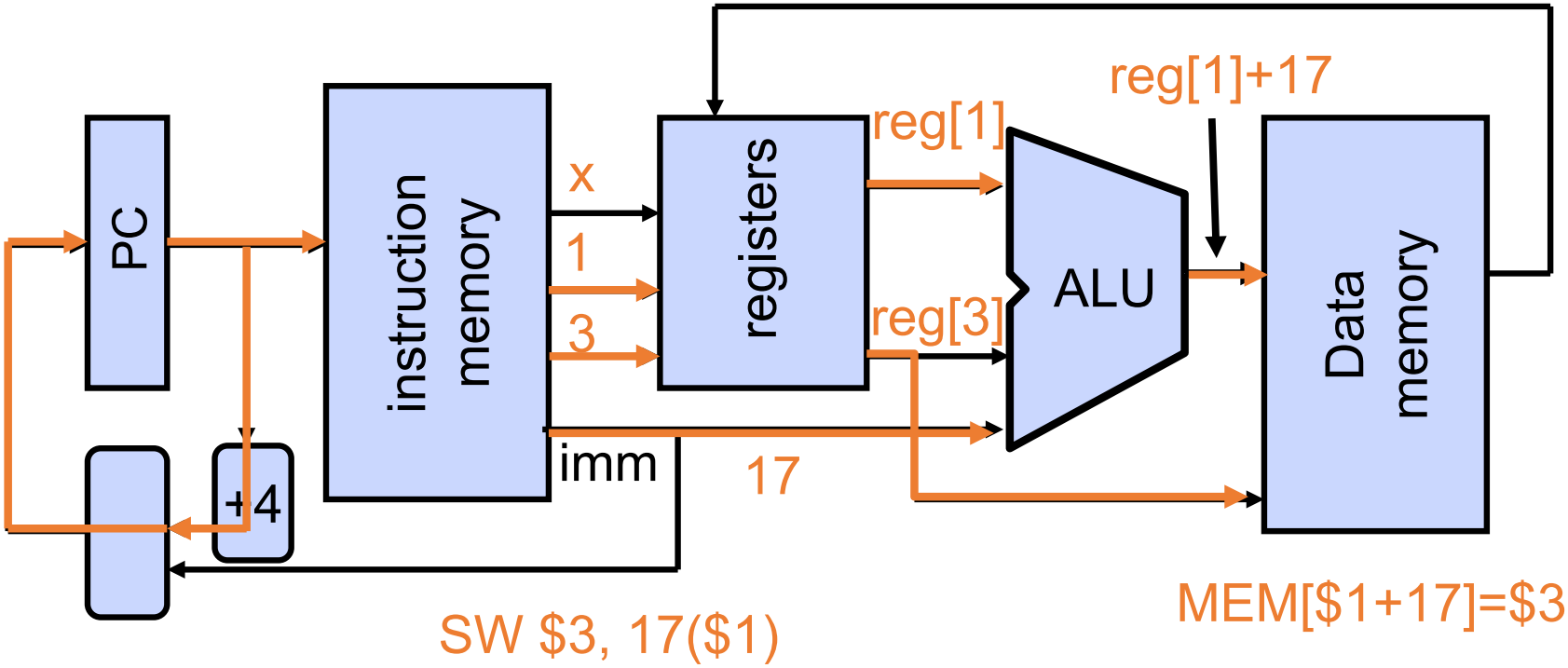
Example: `slti` Instruction



Datapath Walkthroughs (3/3)

- sw $r3, 17($r1)
  - Stage 1: fetch this instruction, inc. PC
  - Stage 2: decode to find it's a sw, then read registers $r1 and $r3
  - Stage 3: add 17 to value in register $r1 (retrieved in Stage 2)
  - Stage 4: write value in register $r3 (retrieved in Stage 2) into memory address computed in Stage 3
  - Stage 5: idle (nothing to write into a register)

Example: `sw` Instruction



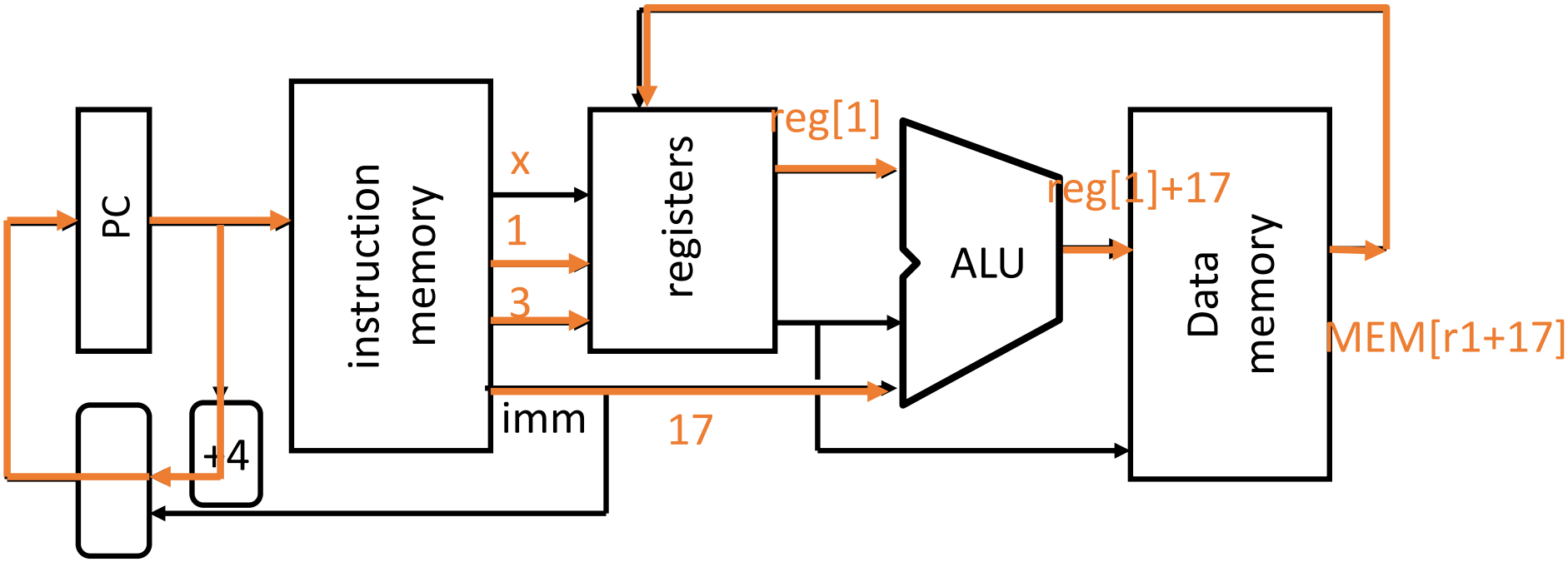
Why Five Stages? (1/2)

- Could we have a different number of stages?
  - Yes, and other architectures do
- Why does MIPS have five if instructions tend to idle for at least one stage?
  - The five stages are the union of all the operations needed by all the instructions.
- Which instruction uses all five stages?
  - **load**

Why Five Stages? (2/2)

- lw $r3, 17($r1)
  - Stage 1: fetch this instruction, inc. PC
  - Stage 2: decode to find it's a lw, then read register $r1
  - Stage 3: add 17 to value in register $r1 (retrieved in Stage 2)
  - Stage 4: read value from memory address compute in Stage 3
  - Stage 5: write value found in Stage 4 into register $r3

Example: lw Instruction



LW r3, 17(r1)