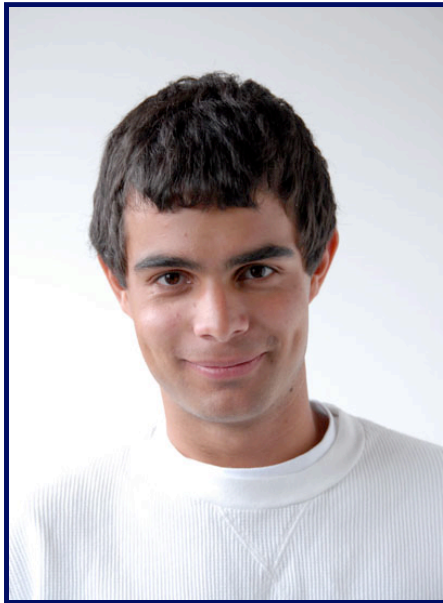


inst.eecs.berkeley.edu/~cs61c
CS61C : Machine Structures

Lecture #18 Introduction to CPU Design

2007-7-25



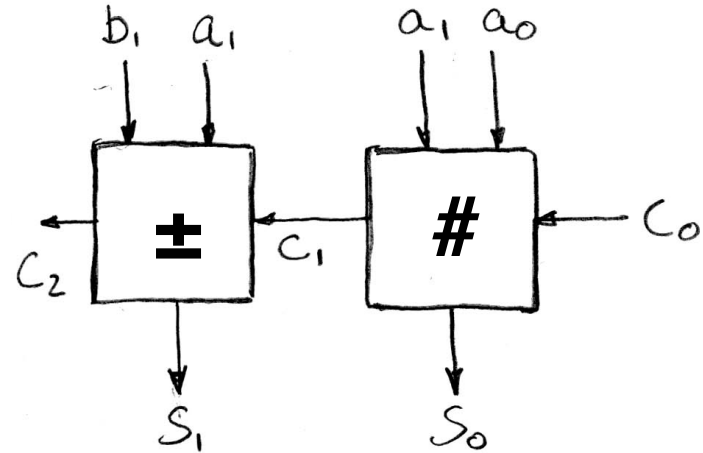
Scott Beamer, Instructor



What about overflow?

- Consider a 2-bit signed # & overflow:

10 = -2
11 = -1
00 = 0
01 = 1



- Overflows when...

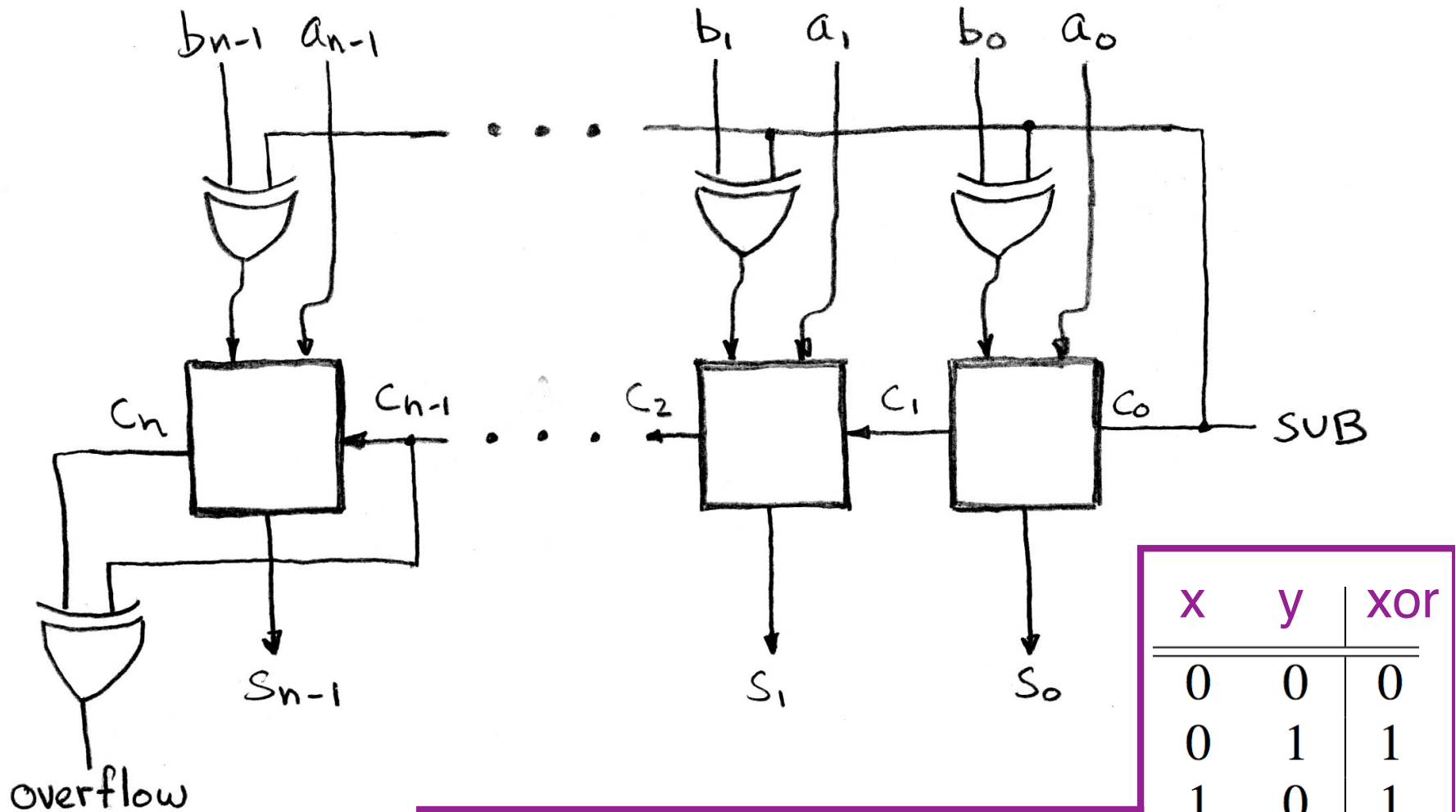
- C_{in} , but no $C_{out} \Rightarrow A, B$ both > 0 , overflow!
- C_{out} , but no $C_{in} \Rightarrow A, B$ both < 0 , overflow!

$$\text{overflow} = c_n \text{ XOR } c_{n-1}$$



Extremely Clever Signed Adder/Subtractor

$A - B = A + (-B)$; how do we make “-B”?

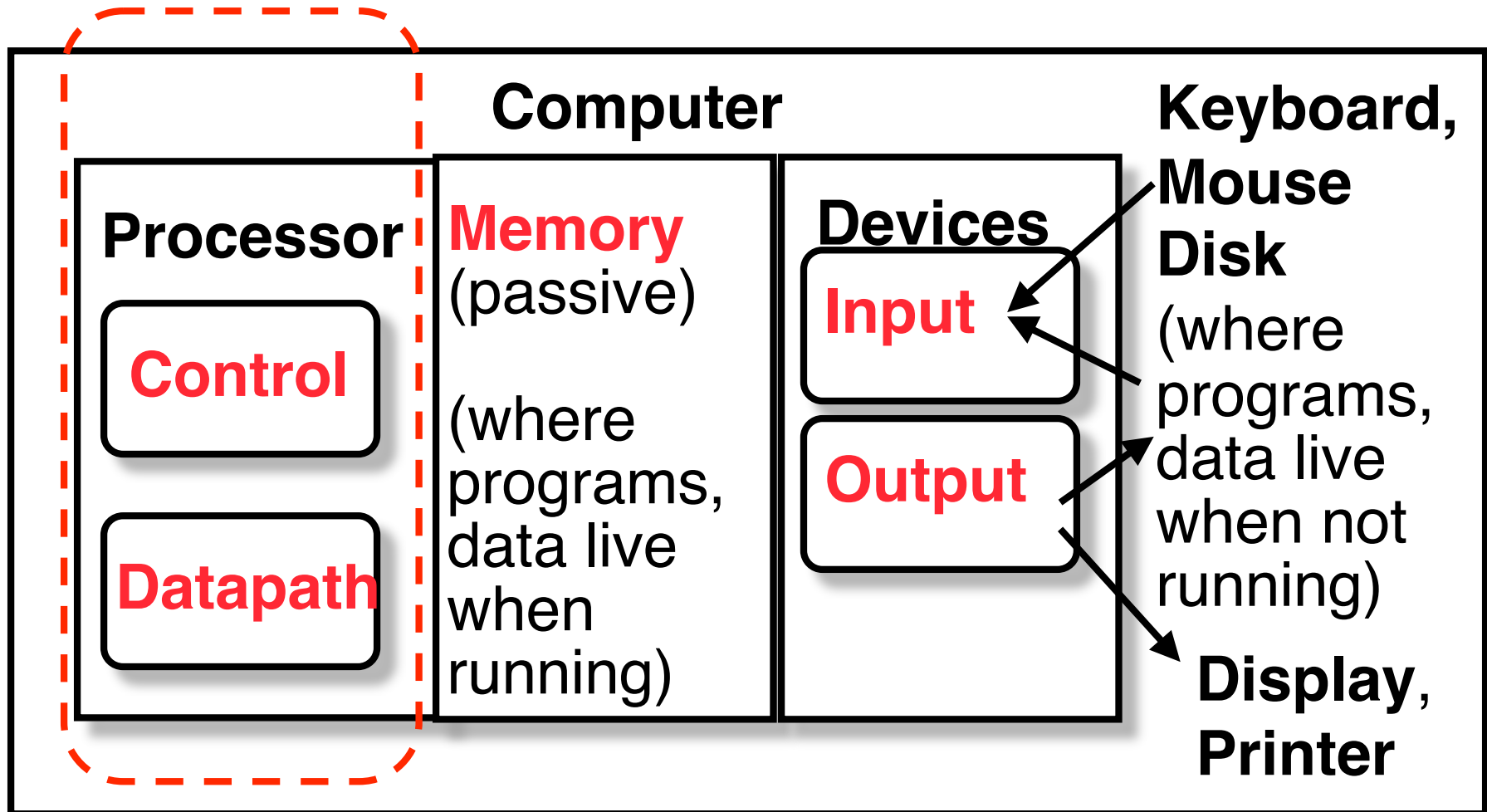


x	y	xor
0	0	0
0	1	1
1	0	1
1	1	0

Conditional Inverter



Five Components of a Computer



The CPU

- **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 (the brawn)
- **Control**: portion of the processor (also in hardware) which tells the datapath what needs to be done (the brain)



Stages of the Datapath : Overview

- **Problem: a single, atomic block which “executes an instruction” (performs all necessary operations beginning with fetching the instruction) would be too bulky and inefficient**
- **Solution: break up the process of “executing an instruction” into stages, and then connect the stages to create the whole datapath**
 - **smaller stages are easier to design**
 - **easy to optimize (change) one stage without touching the others**



Stages of the Datapath (1/5)

- There is a *wide* variety of MIPS instructions: so what general steps do they have in common?
- **Stage 1: Instruction Fetch**
 - no matter what the instruction, the 32-bit instruction word must first be fetched from memory (the cache-memory hierarchy)
 - also, this is where we **Increment PC** (that is, $PC = PC + 4$, to point to the next instruction: byte addressing so + 4)



Stages of the Datapath (2/5)

- **Stage 2: Instruction Decode**

- upon fetching the instruction, we next gather data from the fields (*decode* all necessary instruction data)
- first, read the Opcode to determine instruction type and field lengths
- second, read in data from all necessary registers
 - for `add`, read two registers
 - for `addi`, read one register
 - for `jal`, no reads necessary



Stages of the Datapath (3/5)

- **Stage 3: ALU (Arithmetic-Logic Unit)**

- the real work of most instructions is done here: arithmetic (+, -, *, /), shifting, logic (&, |), comparisons (slt)

- what about loads and stores?

- `lw $t0, 40($t1)`

- the address we are accessing in memory = the value in `$t1` PLUS the value 40

- so we do this addition in this stage



Stages of the Datapath (4/5)

- **Stage 4: Memory Access**
 - actually only the load and store instructions do anything during this stage; the others remain idle during this stage or skip it all together
 - since these instructions have a unique step, we need this extra stage to account for them
 - as a result of the cache system, this stage is expected to be fast



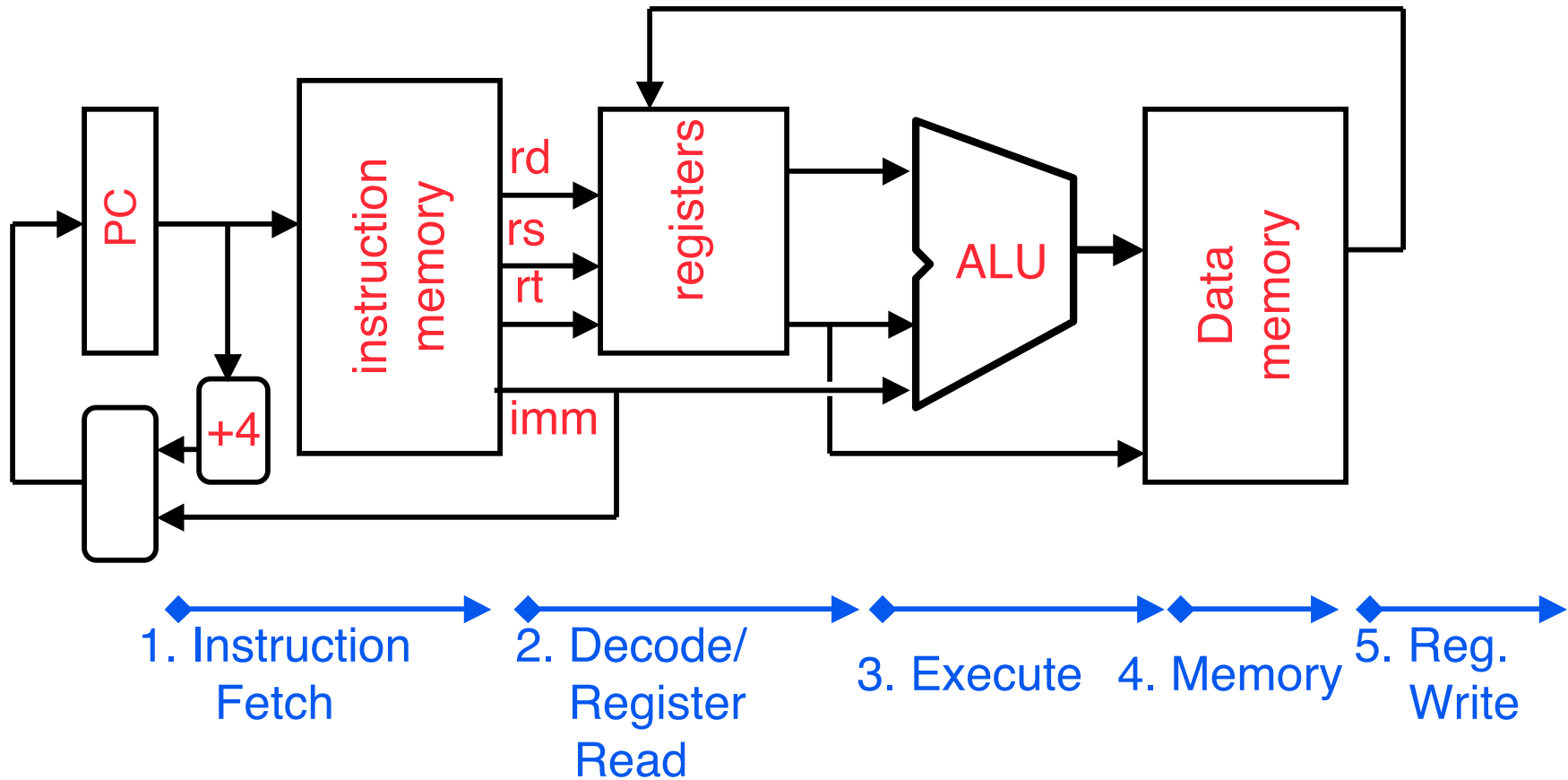
Stages of the Datapath (5/5)

- **Stage 5: Register Write**

- most instructions write the result of some computation into a register
- examples: 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 stage or skip it all together



Generic Steps of Datapath



Peer Instruction

- A. Our **ALU** is a synchronous device
- B. We should use the main **ALU** to compute $PC=PC+4$
- C. The **ALU** is inactive for memory reads or writes.

	ABC
1:	FFF
2:	FFT
3:	FTF
4:	FTT
5:	TFF
6:	TFT
7:	TF
8:	TTT



Administrivia

- **Assignments**
 - HW5 due 7/26
 - HW6 due 7/29
- **Midterm**
 - Grading standards up soon
 - If you wish to have a problem regraded
 - Staple your reasons to the front of the exam
 - Return your exam to your TA
- **Scott is now holding regular OH on Fridays 11-12 in 329 Soda**



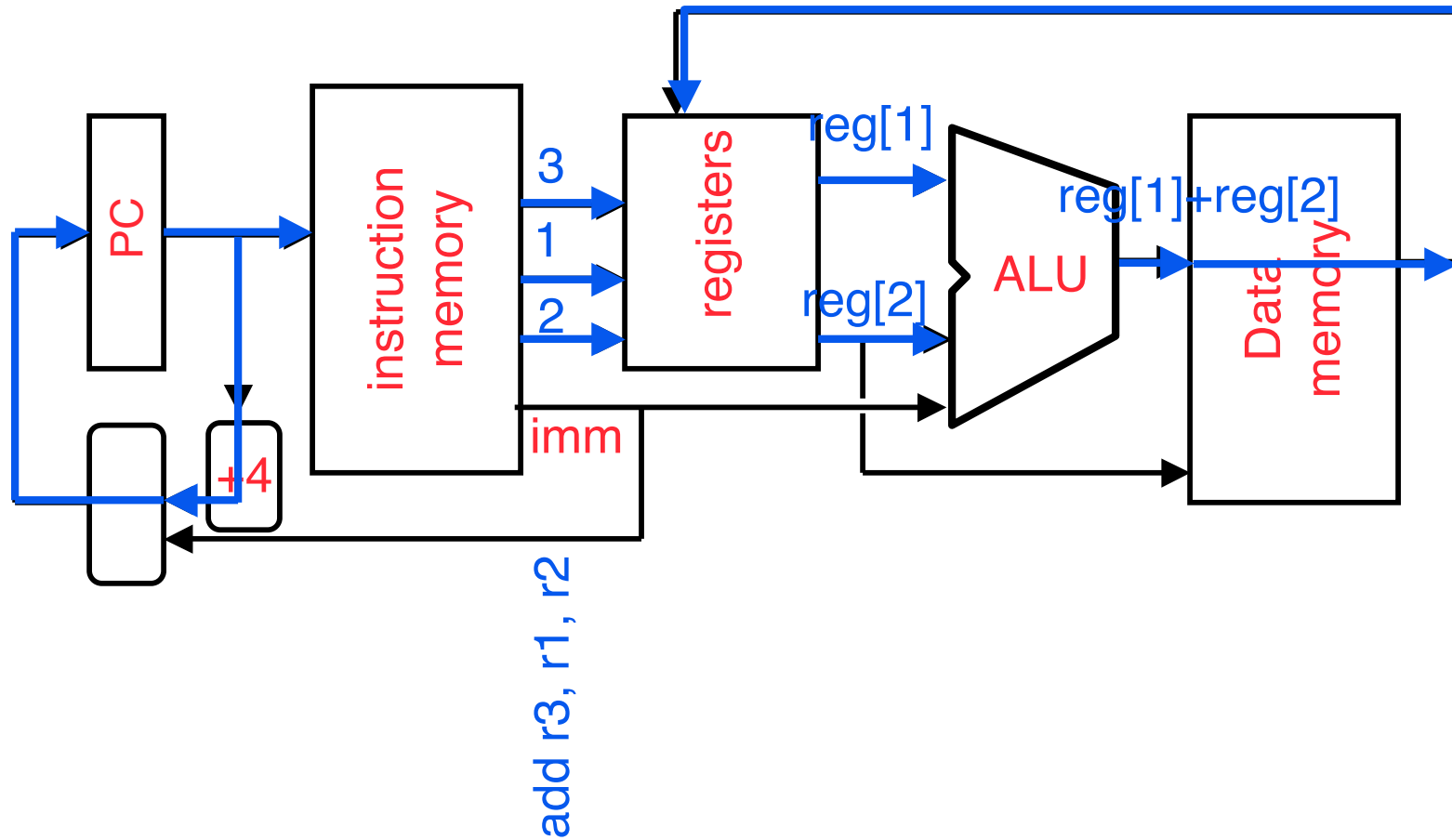
Don't unplug stuff in the labs!

Datapath Walkthroughs (1/3)

- **add** `$r3, $r1, $r2` # `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

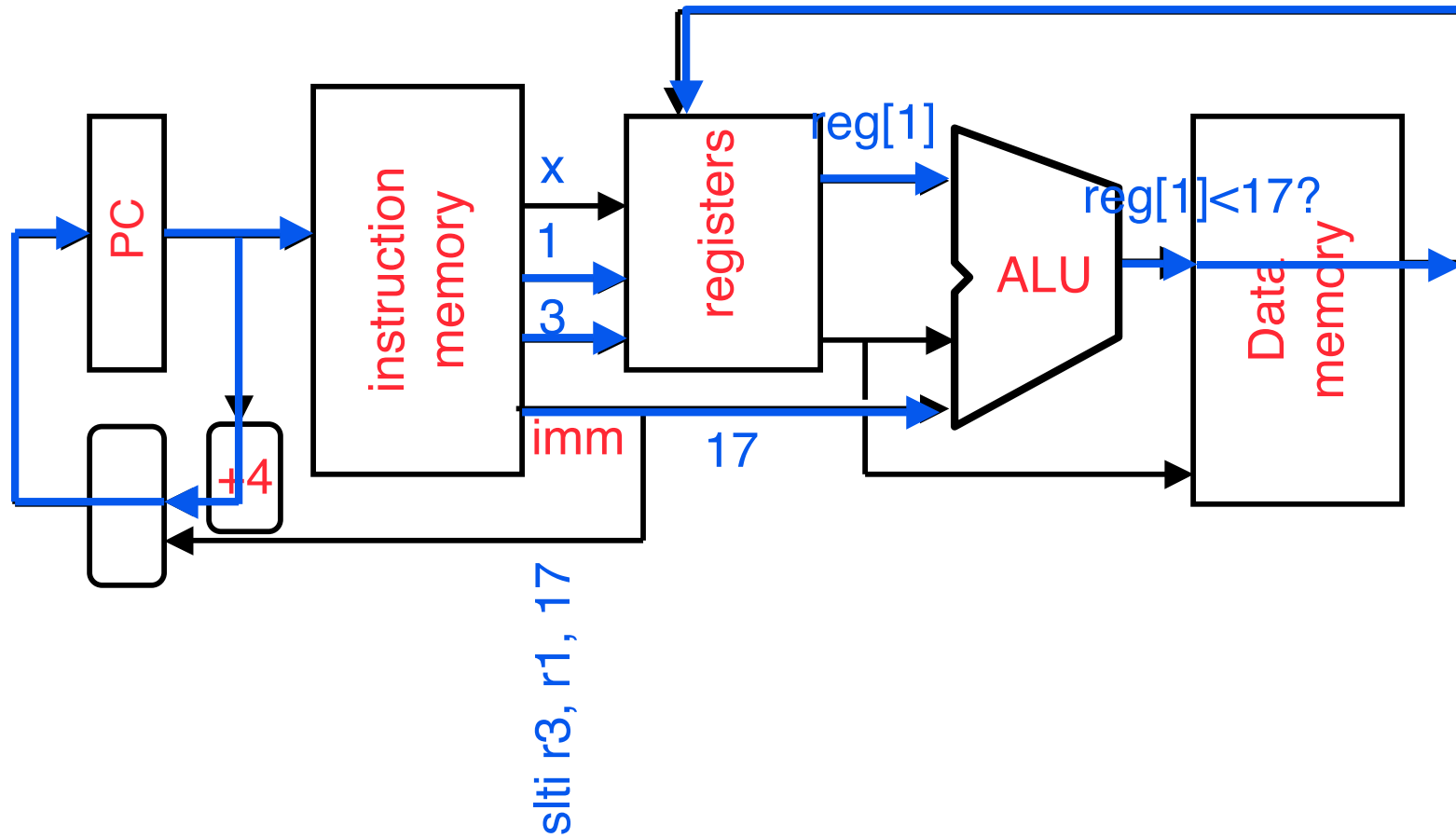


Datapath Walkthroughs (2/3)

- `slti $r3, $r1, 17`
 - Stage 1: fetch this instruction, inc. PC
 - Stage 2: decode to find it's an `slt`, then read register `$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

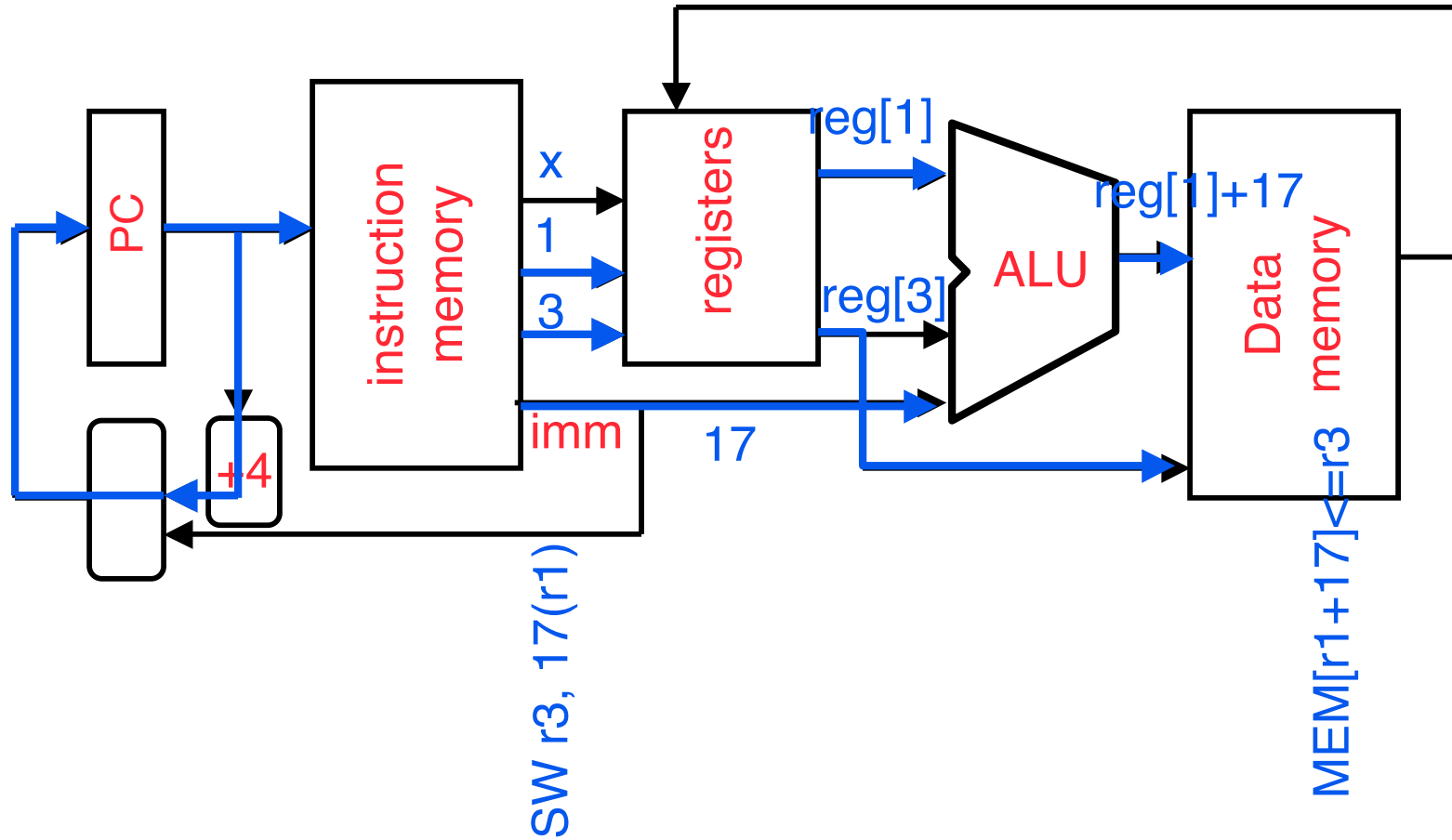


Datapath Walkthroughs (3/3)

- **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**
- **So 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.**
 - **There is one instruction that uses all five stages: the load**

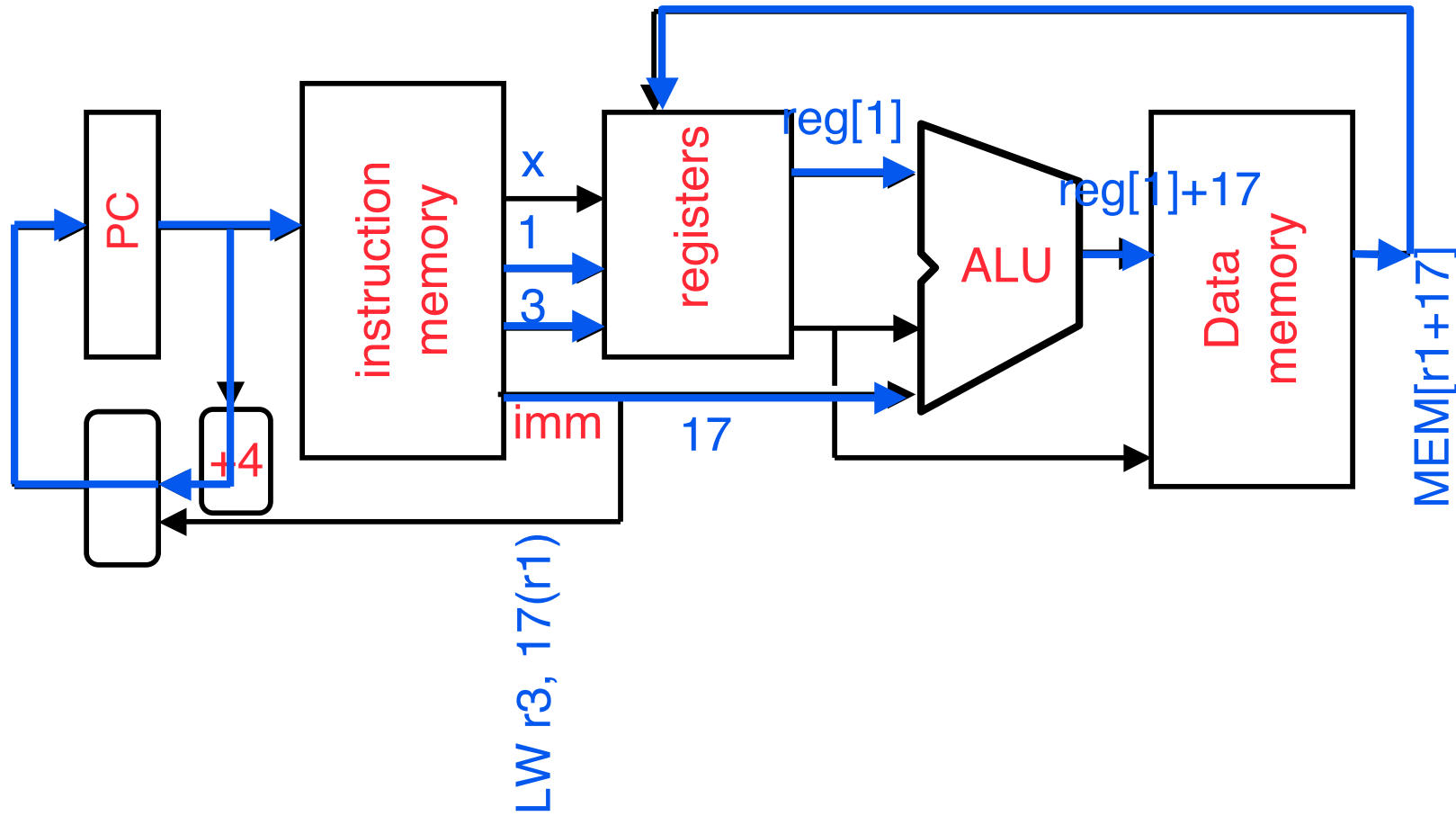


Why Five Stages? (2/2)

- **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**

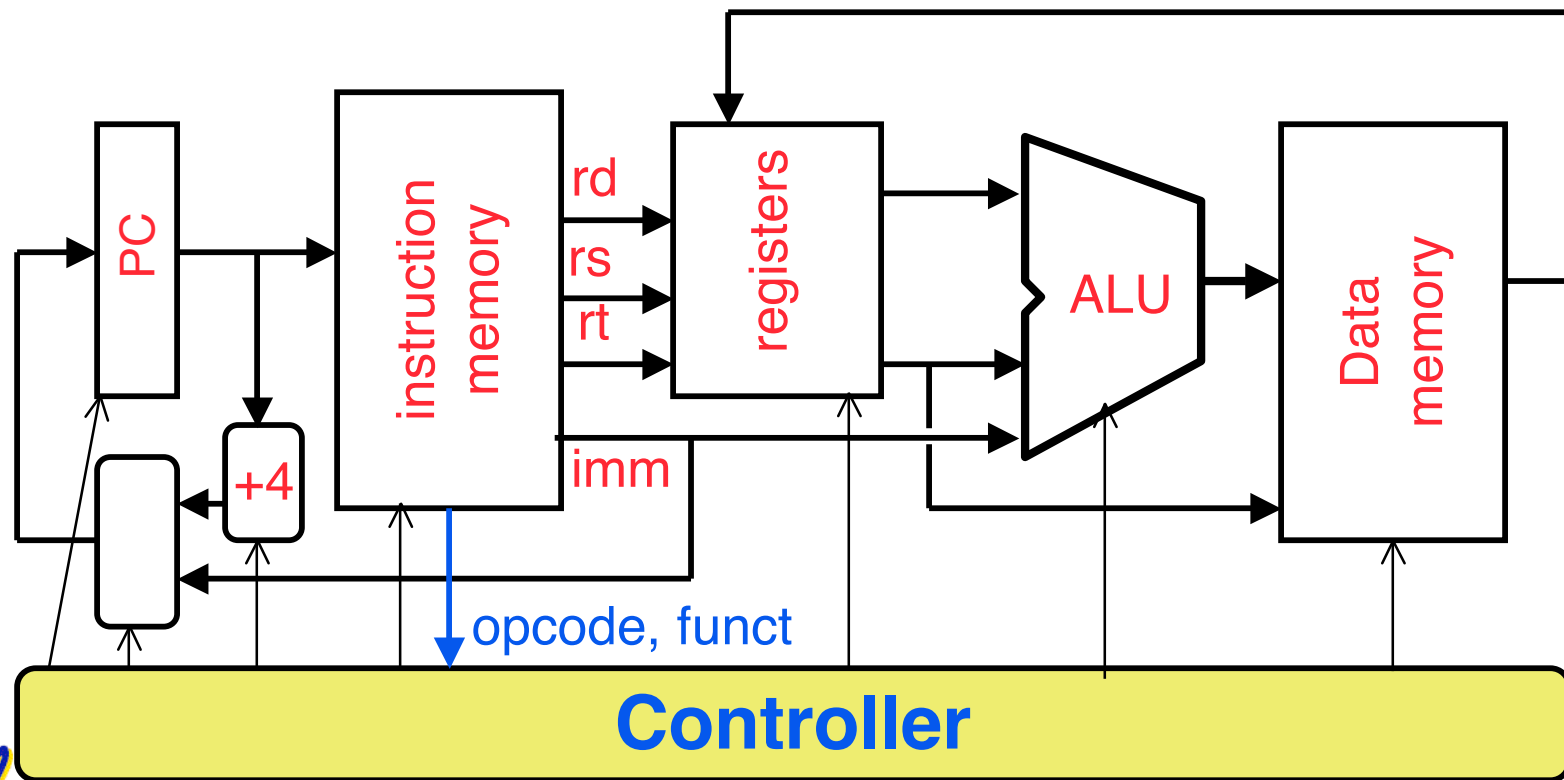


Example: lw Instruction



Datapath Summary

- The datapath based on data transfers required to perform instructions
- A *controller* causes the right transfers to happen



What Hardware Is Needed? (1/2)

- **PC: a register which keeps track of memory addr of the next instruction**
- **General Purpose Registers**
 - used in Stages 2 (Read) and 5 (Write)
 - MIPS has 32 of these
- **Memory**
 - used in Stages 1 (Fetch) and 4 (R/W)
 - cache system makes these two stages as fast as the others, on average



What Hardware Is Needed? (2/2)

- **ALU**

- used in Stage 3
- something that performs all necessary functions: arithmetic, logicals, etc.
- we'll design details later

- **Miscellaneous Registers**

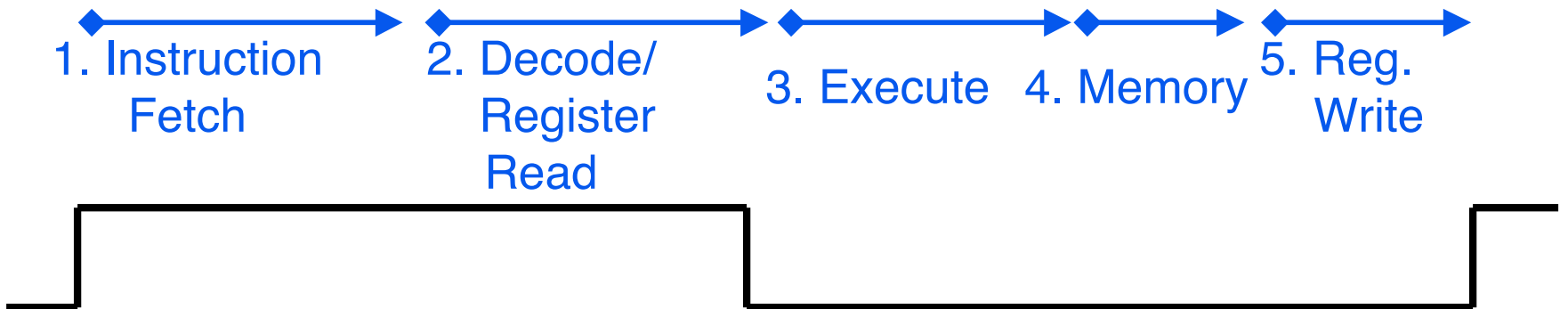
- In implementations with only one stage per clock cycle, registers are inserted between stages to hold intermediate data and control signals as they travel from stage to stage.
- Note: Register is a general purpose term meaning something that stores bits. Not all registers are in the “register file”.



CPU clocking (1/2)

For each instruction, how do we control the flow of information through the datapath?

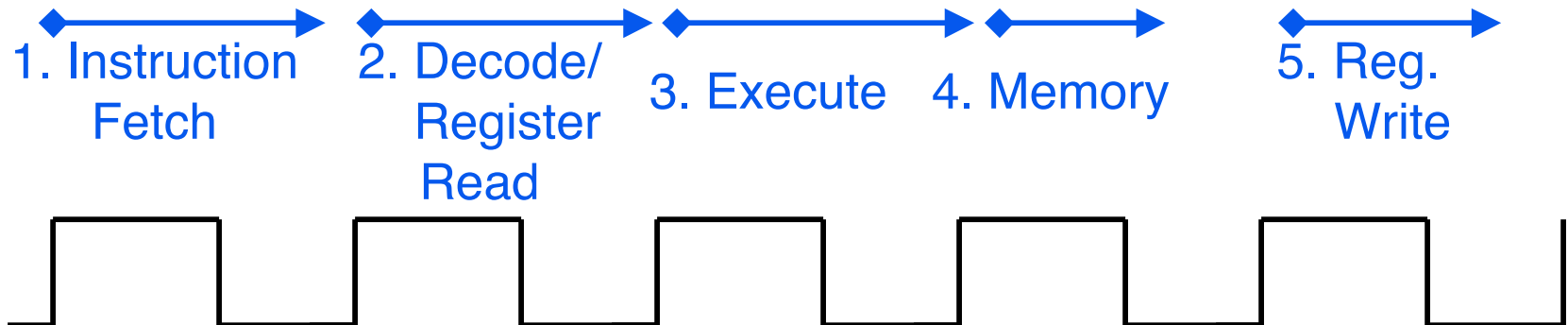
- **Single Cycle CPU:** All stages of an instruction are completed within one *long* clock cycle.
 - The clock cycle is made sufficient long to allow each instruction to complete all stages without interruption and within one cycle.



CPU clocking (2/2)

For each instruction, how do we control the flow of information through the datapath?

- **Multiple-cycle CPU:** Only one stage of instruction per clock cycle.
 - The clock is made as long as the slowest stage.



Several significant advantages over single cycle execution: Unused stages in a particular instruction can be skipped OR instructions can be pipelined (overlapped).



Peer Instruction

- A. If the destination reg is the same as the source reg, we **could compute the incorrect value!**
- B. We're going to be able to read 2 registers and write a 3rd in **1 cycle**
- C. Datapath is hard, **Control is easy**

	ABC
0:	FFF
1:	FFT
2:	FTF
3:	FTT
4:	TFF
5:	TFT
6:	TF
7:	TTT

“And In conclusion...”

- **N-bit adder-subtractor done using N 1-bit adders with XOR gates on input**
 - **XOR serves as conditional inverter**
- **CPU design involves Datapath, Control**
 - **Datapath in MIPS involves 5 CPU stages**
 - 1) **Instruction Fetch**
 - 2) **Instruction Decode & Register Read**
 - 3) **ALU (Execute)**
 - 4) **Memory**
 - 5) **Register Write**

