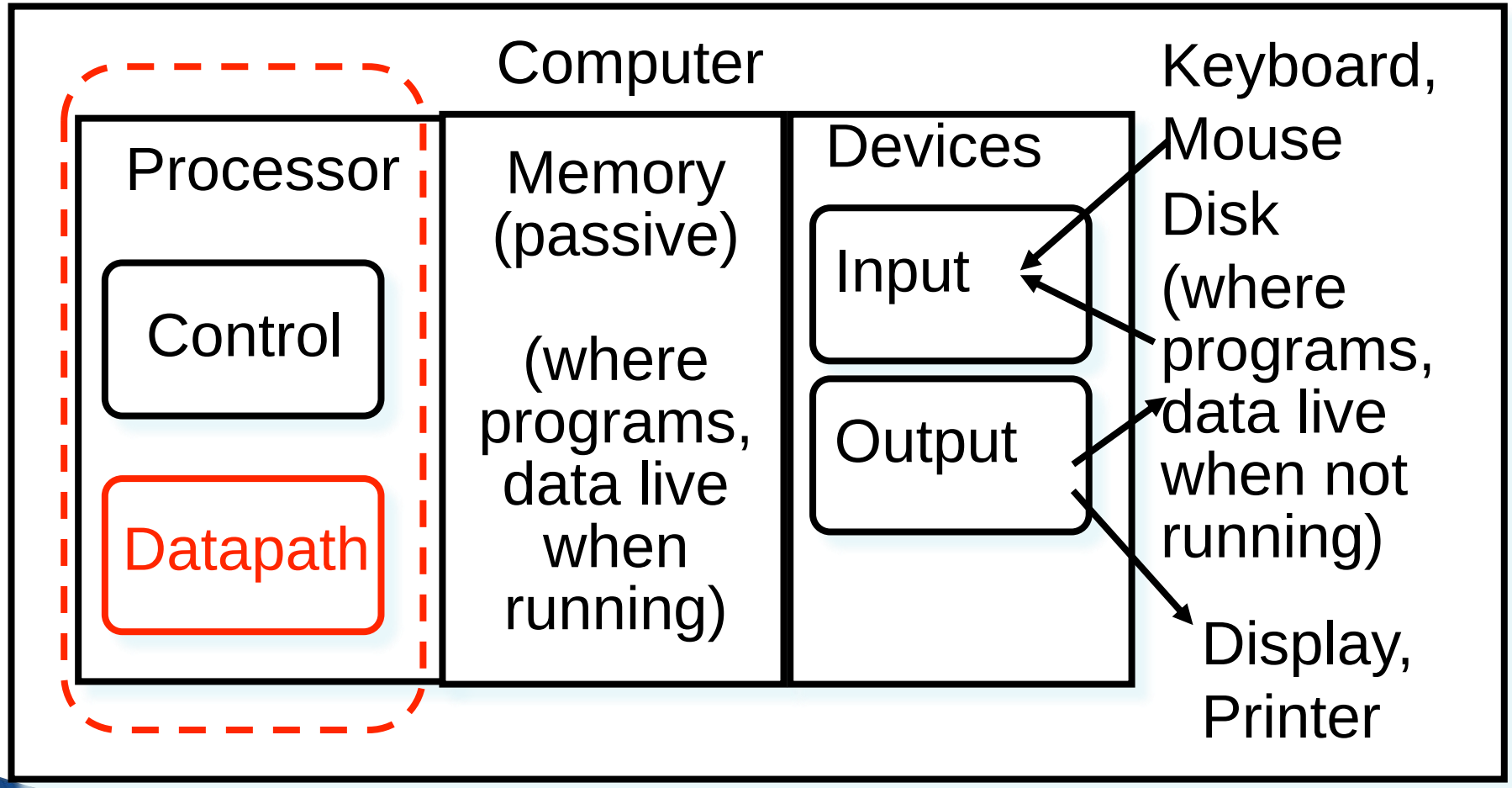
CSE 31
Computer Organization

Lecture 22 – CPU Design (2)

Announcement

- Lab #10
  - Due in 1 week
- Project #2
  - Start working on it during lab this week
  - Due Monday (4/29)
- HW #7 in CatCourses
  - Due Monday (5/6) at 11:59pm
- Reading assignment
  - Chapter 5.7-5.11 of zyBooks
    - Make sure to do the Participation Activities
    - Due Friday (5/3) at 11:59pm
- Course evaluation online by 5/9

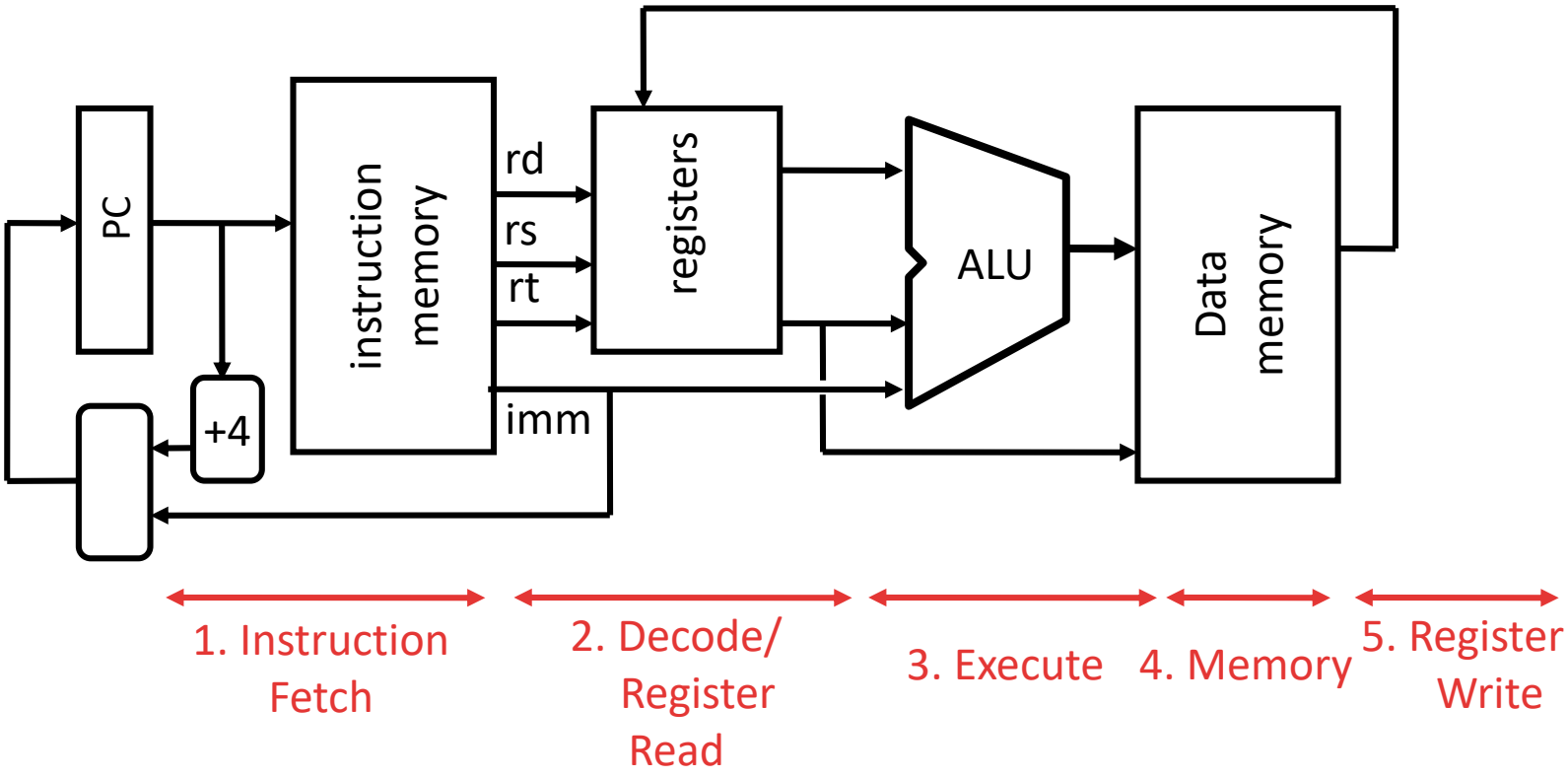
Five Components of a Computer



Review

- 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

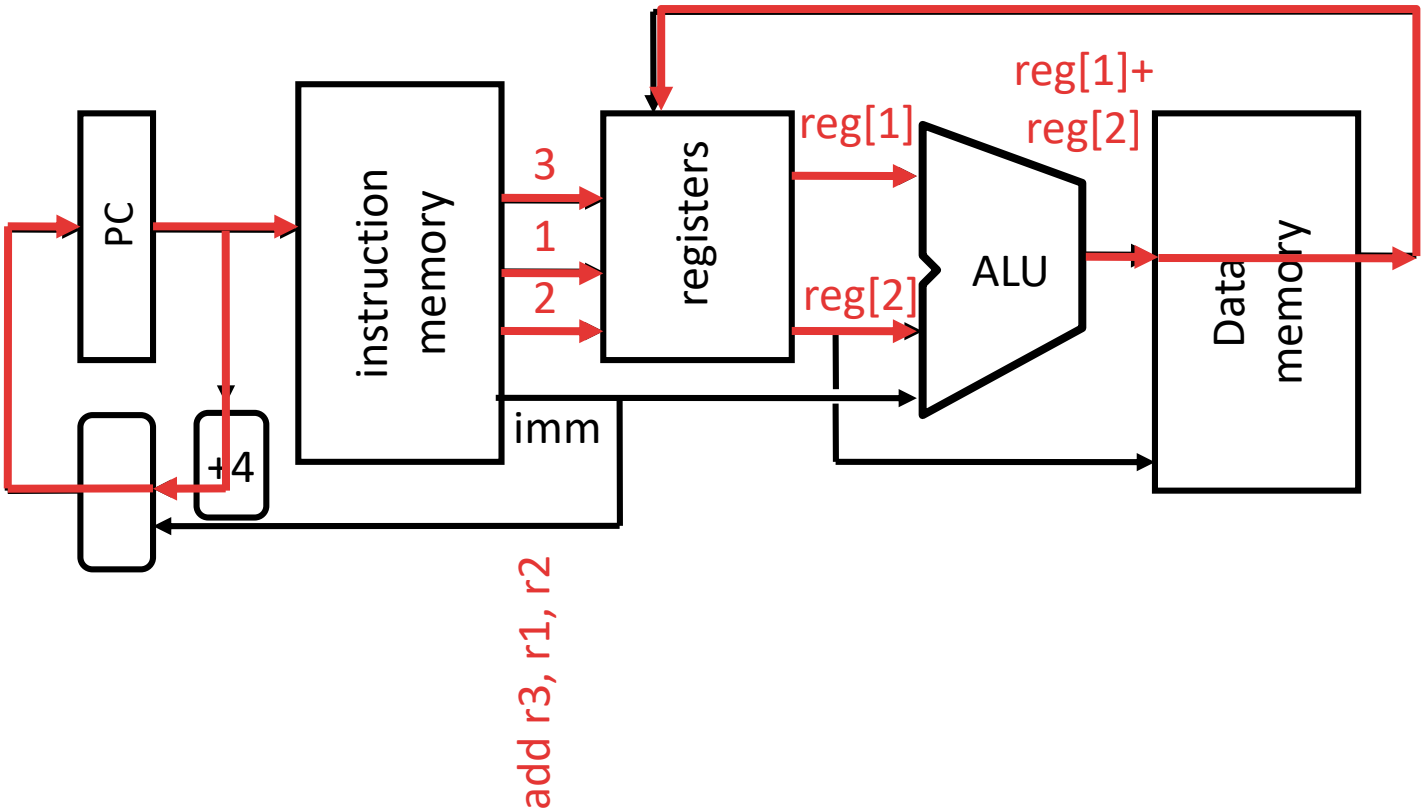
Generic Diagram of Datapath



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`

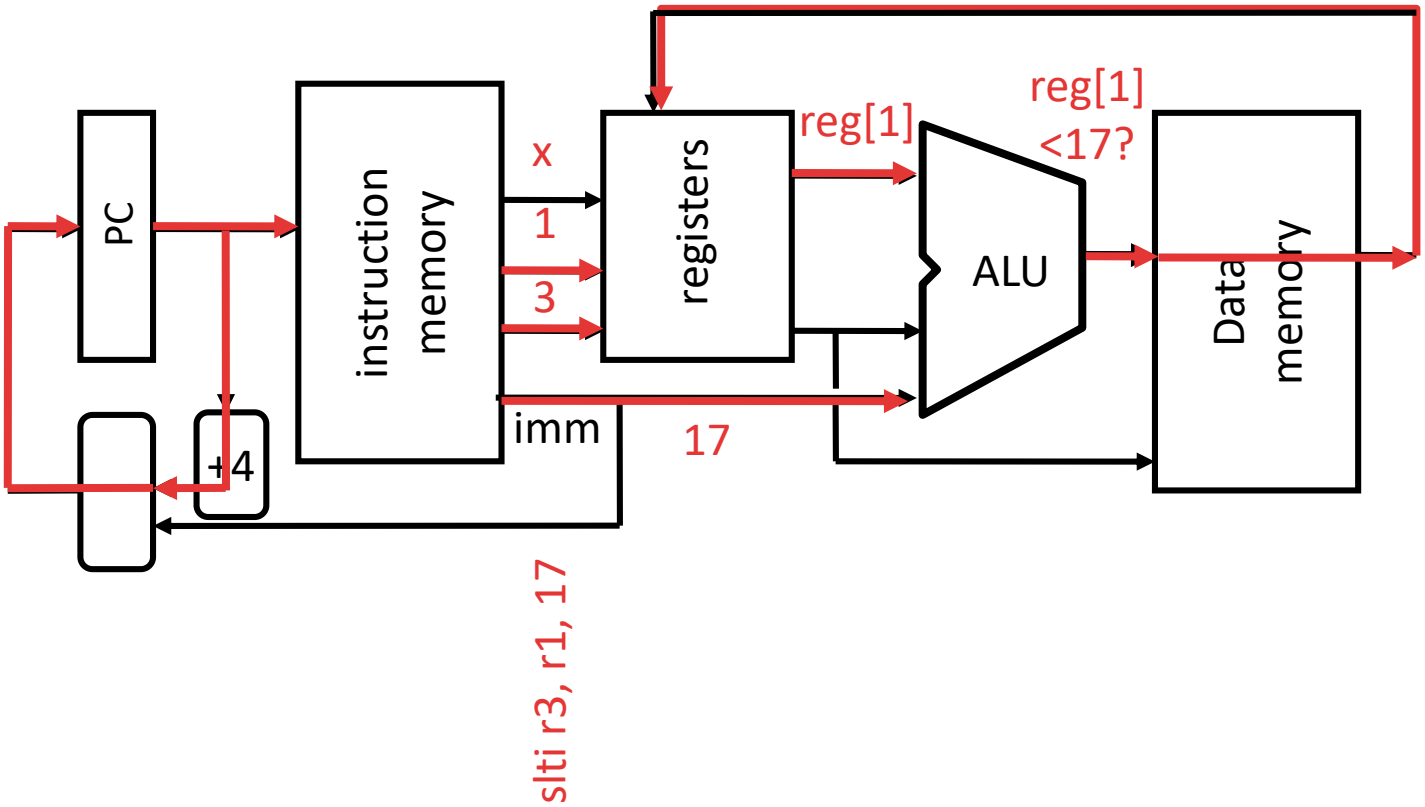
Example: add Instruction



Datapath Walkthroughs (2/3)

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

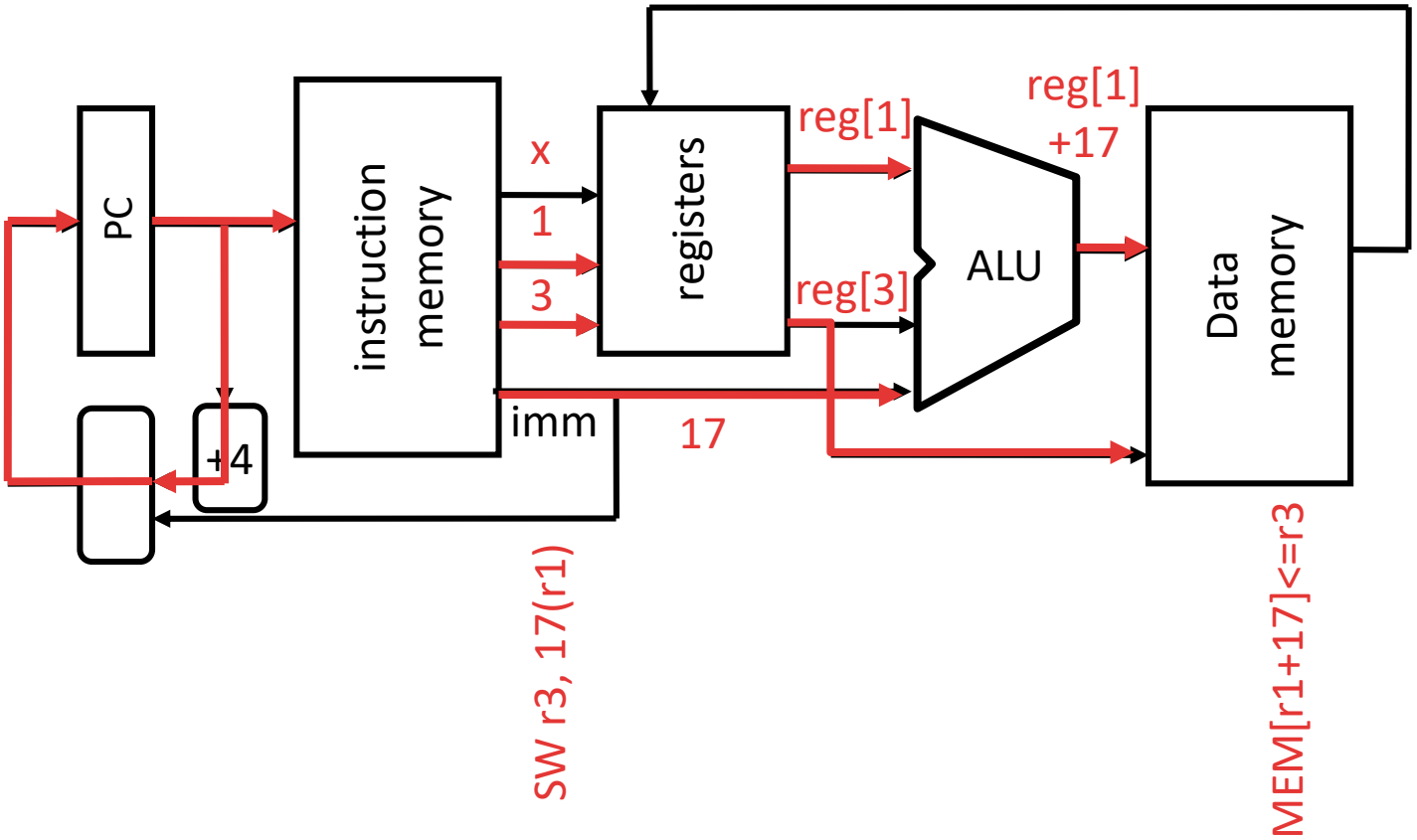
Example: `slti` Instruction



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



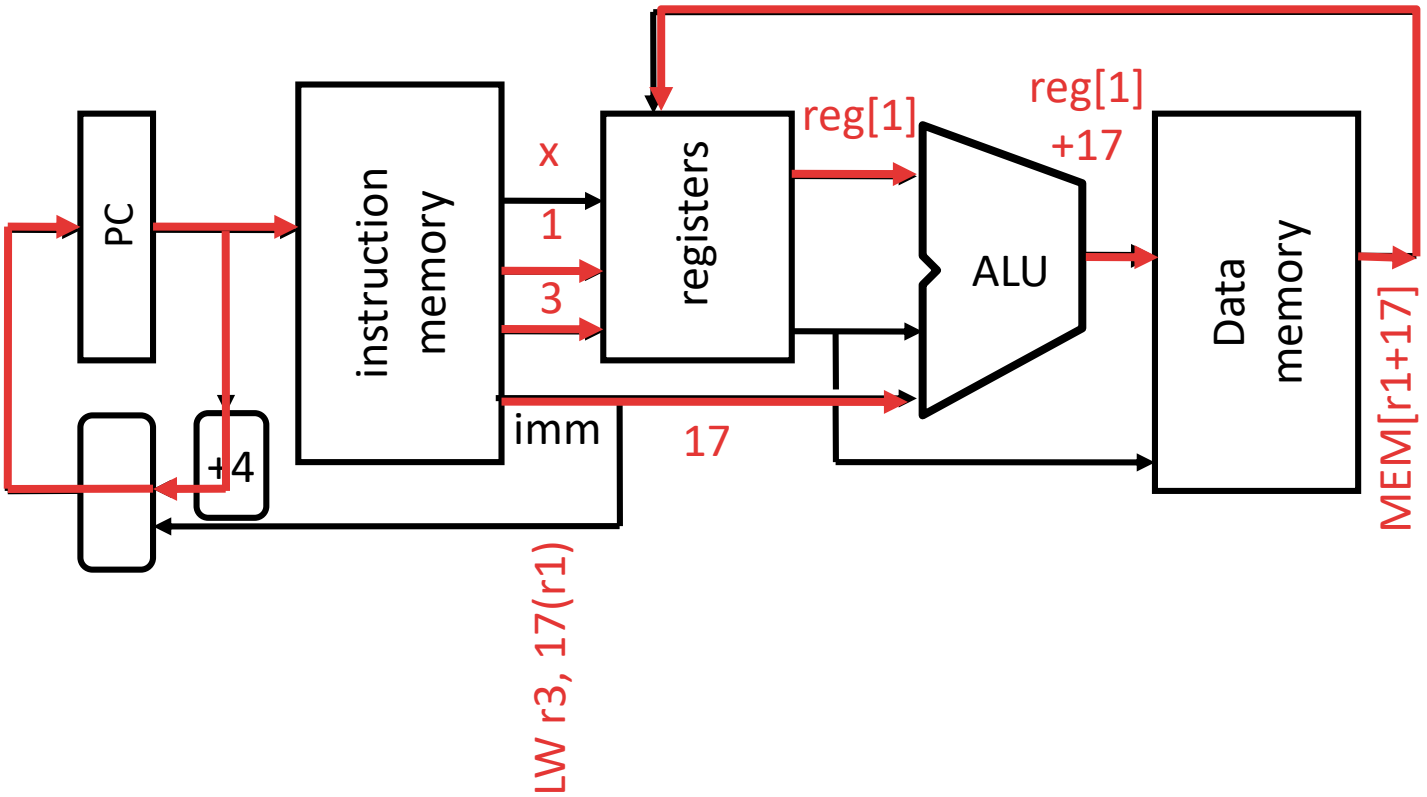
Why Five Stages? (1/2)

- Could we have a different number of stages?
  - Yes, and other architectures do
- 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)

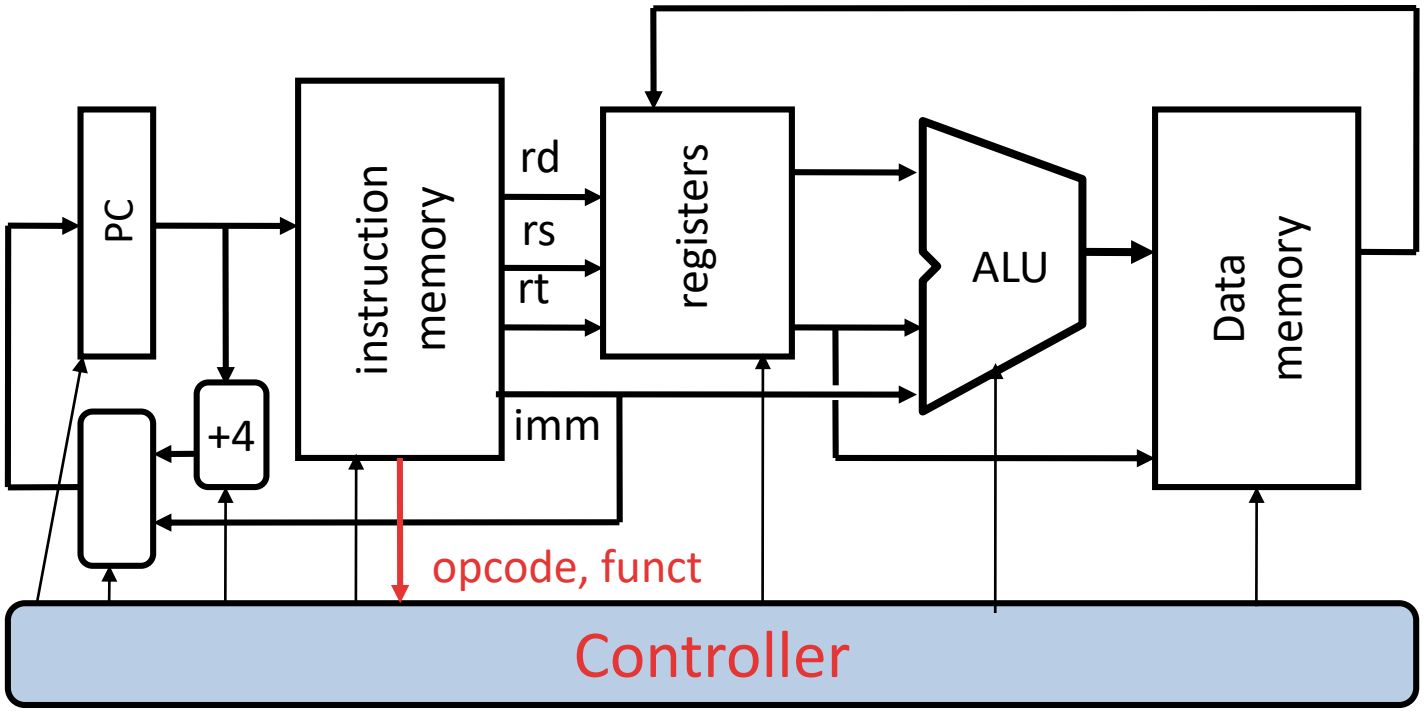
- `lw  $r3, 17($r1)`
  - Stage 1: fetch this instruction, inc. PC
  - Stage 2: decode to find it's a `lw`, then read register `$r1`
  - Stage 3: add `17` to value in register `$r1` (retrieved in Stage 2)
  - Stage 4: read value from memory address compute in Stage 3
  - Stage 5: write value found in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

Example: `lw` Instruction



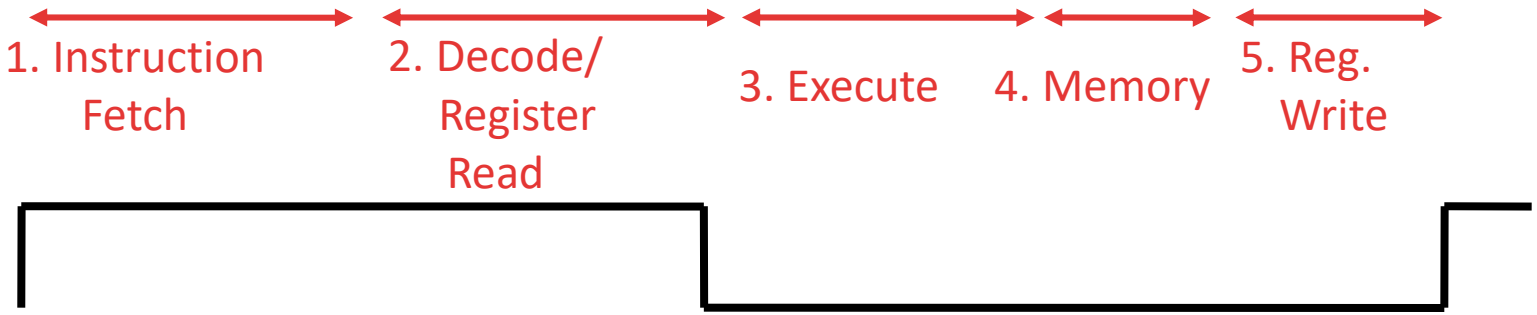
Datapath Summary

- Datapath based on data transfers required to perform instructions
- Controller causes the right transfers to happen



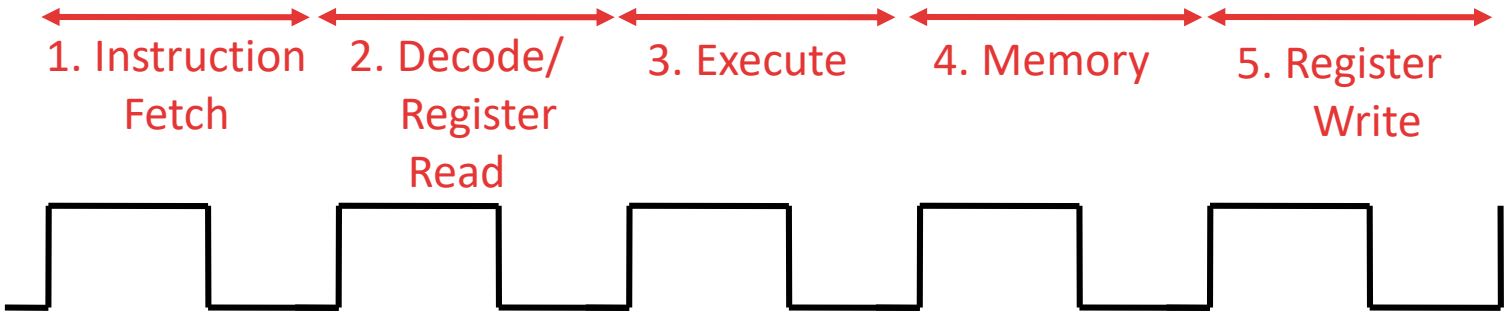
CPU clocking (1/2)

- For each instruction, how do we control the flow of information though the datapath?
- **Single Cycle CPU**: All stages of an instruction completed within one long clock cycle
  - Clock cycle sufficiently long to allow each instruction to complete all stages without interruption within one cycle



CPU clocking (2/2)

- **Multiple-cycle CPU**: only one stage of instruction per clock cycle
  - 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)
  - Will talk about this in CSE 140!

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 travels from stage to stage.
  - Note: Register is a general purpose term meaning something that stores bits. Not all registers are in the "register file".

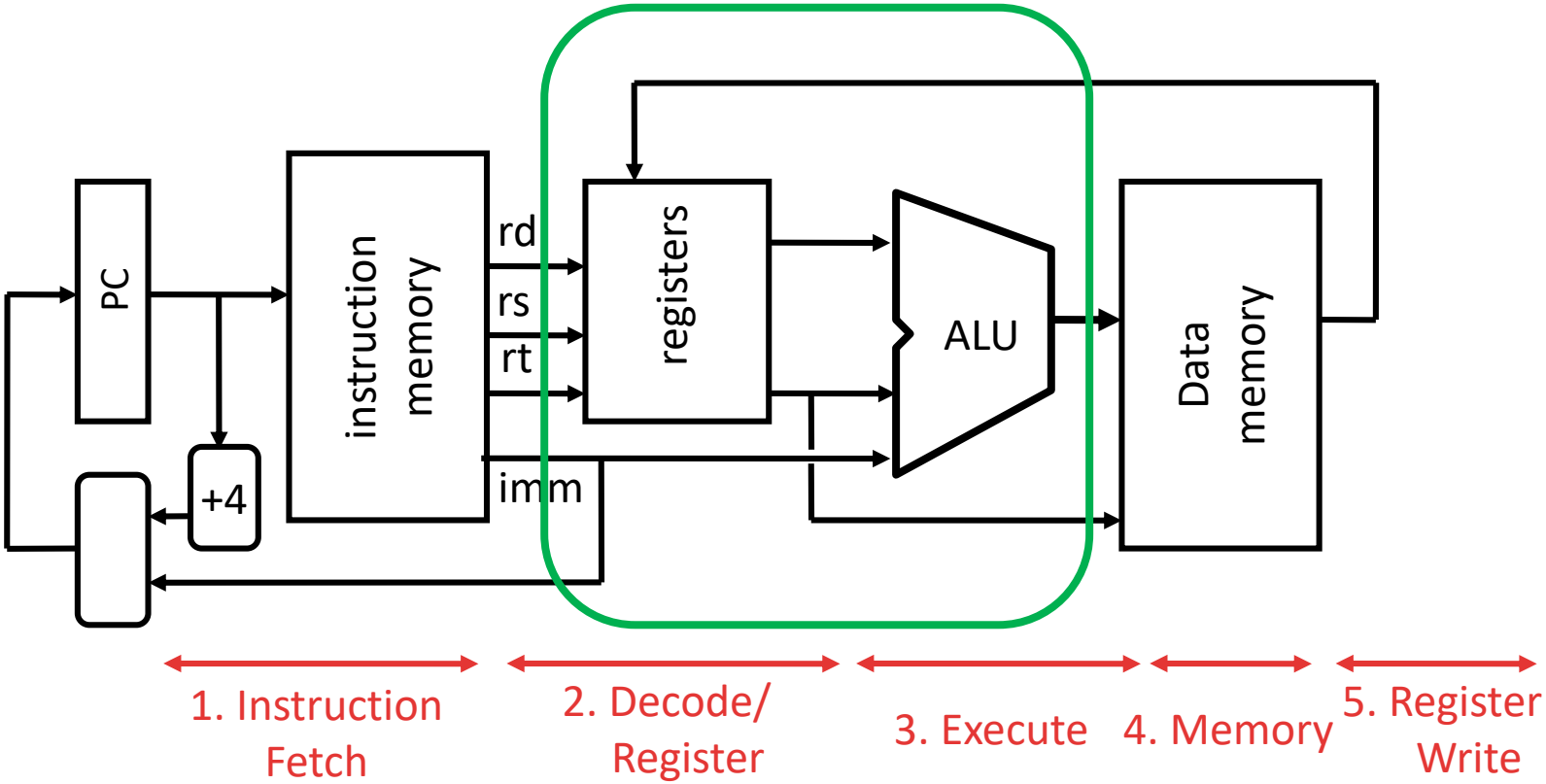
Quiz

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

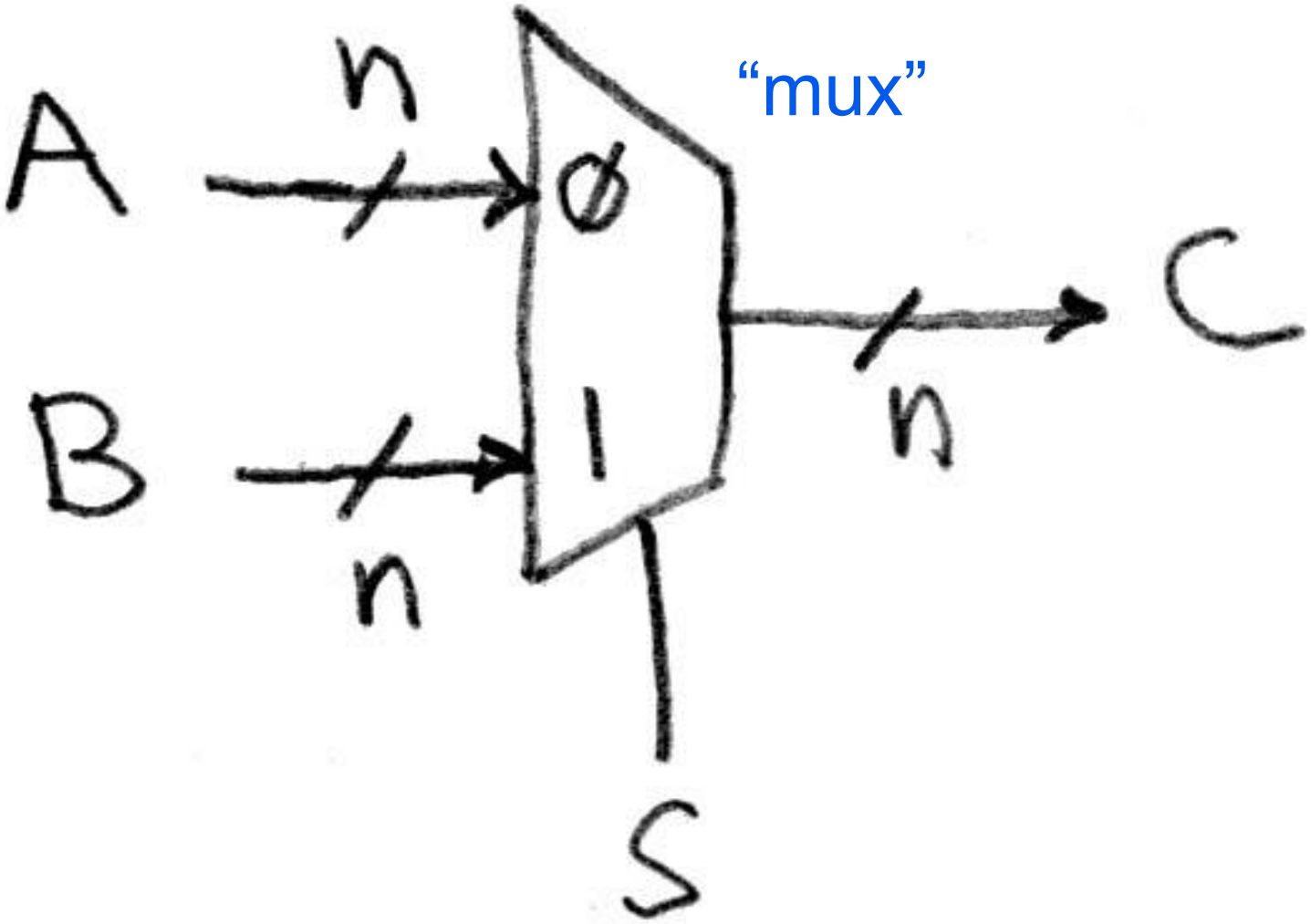
C. Datapath is hard, Control is easy

Generic Steps of Datapath

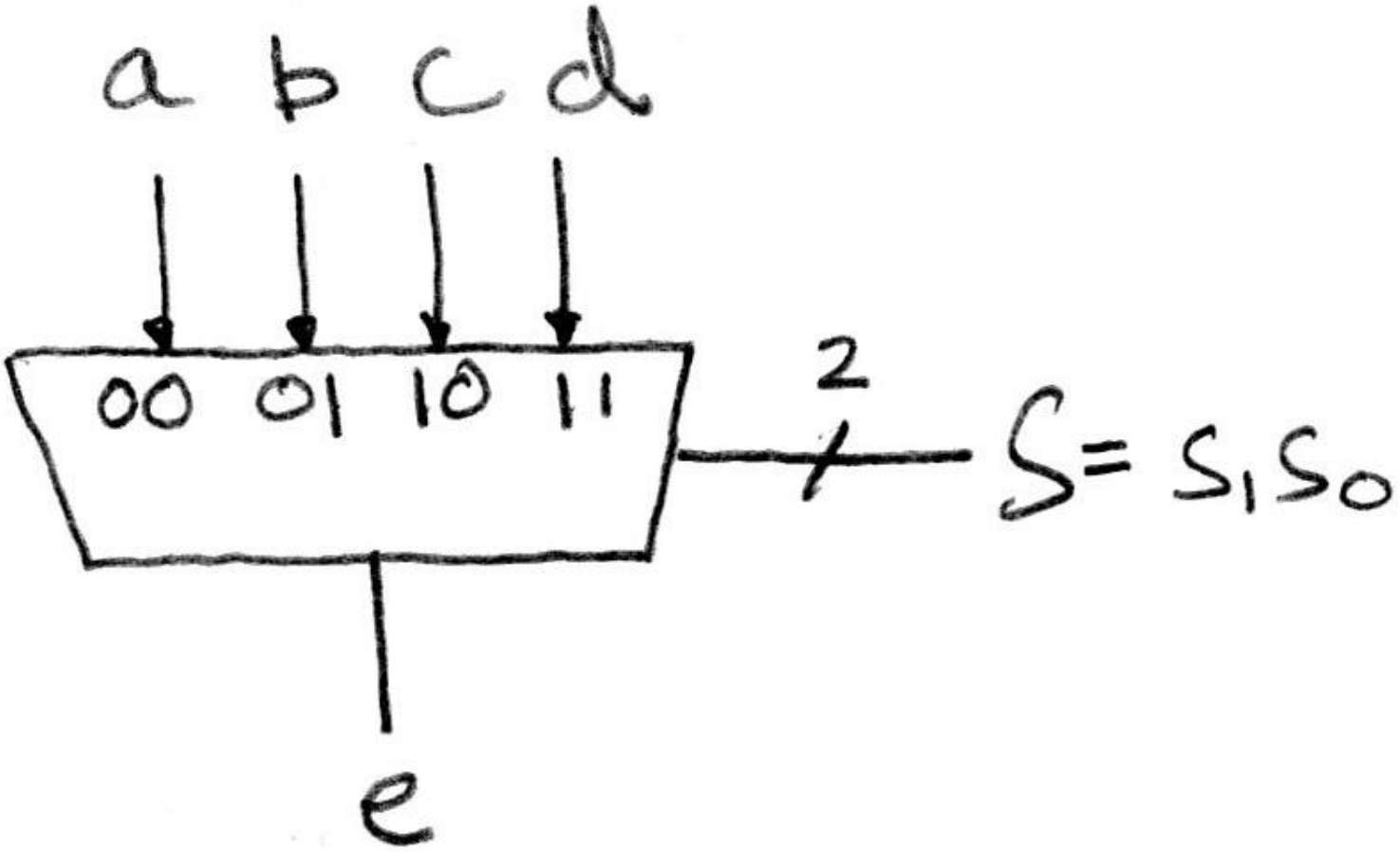


How do we handle the different register usage between r-type and i-type instructions?

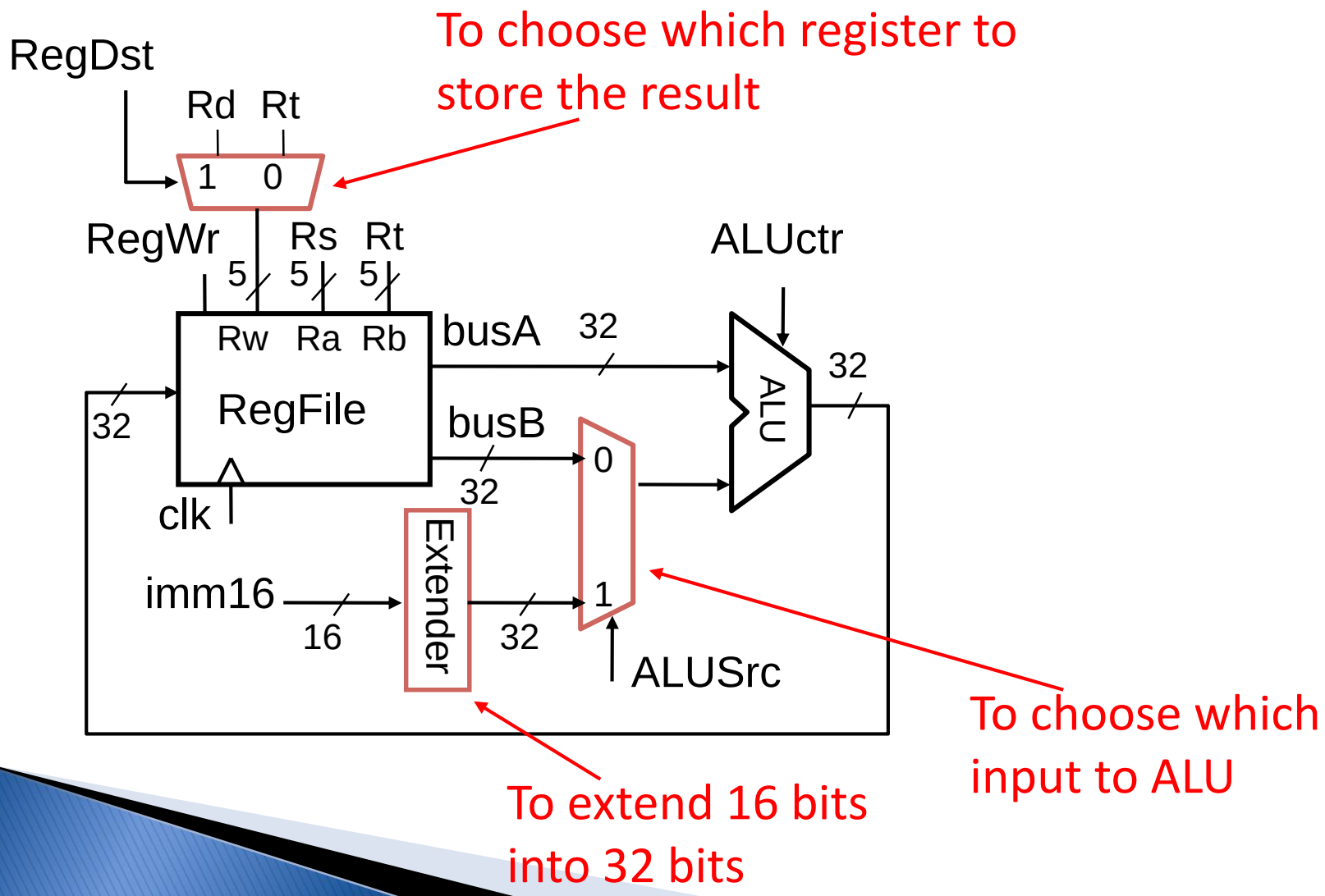
Data Multiplexor (mux)
(2-to-1, n-bits)



4-to-1 Multiplexor?

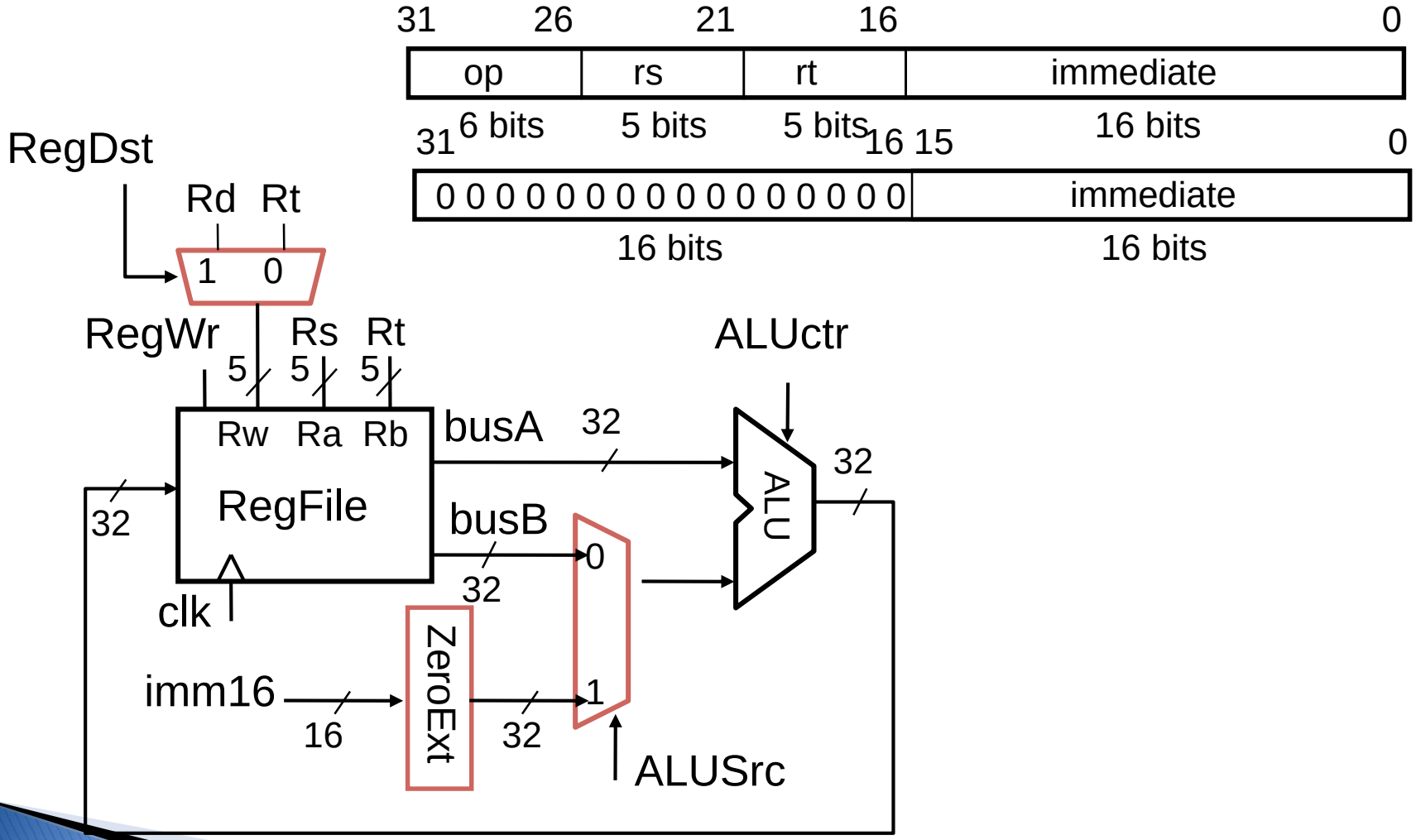


A zoomed in version of RegFile and ALU



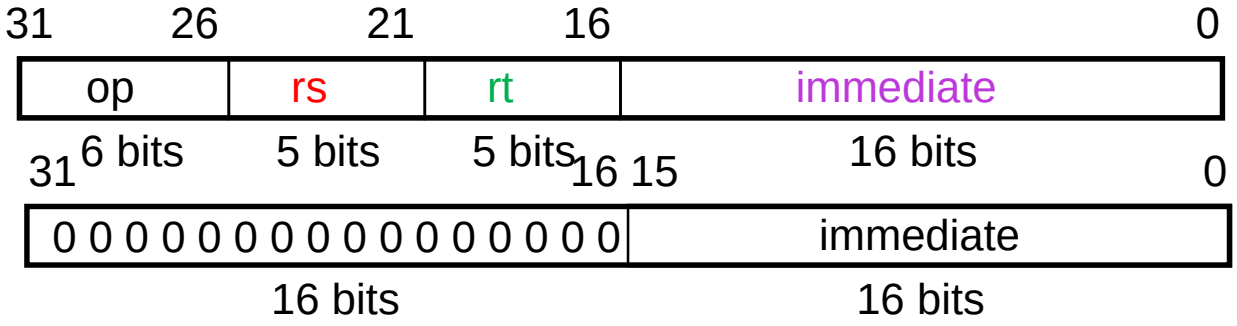
Operations with Immediate

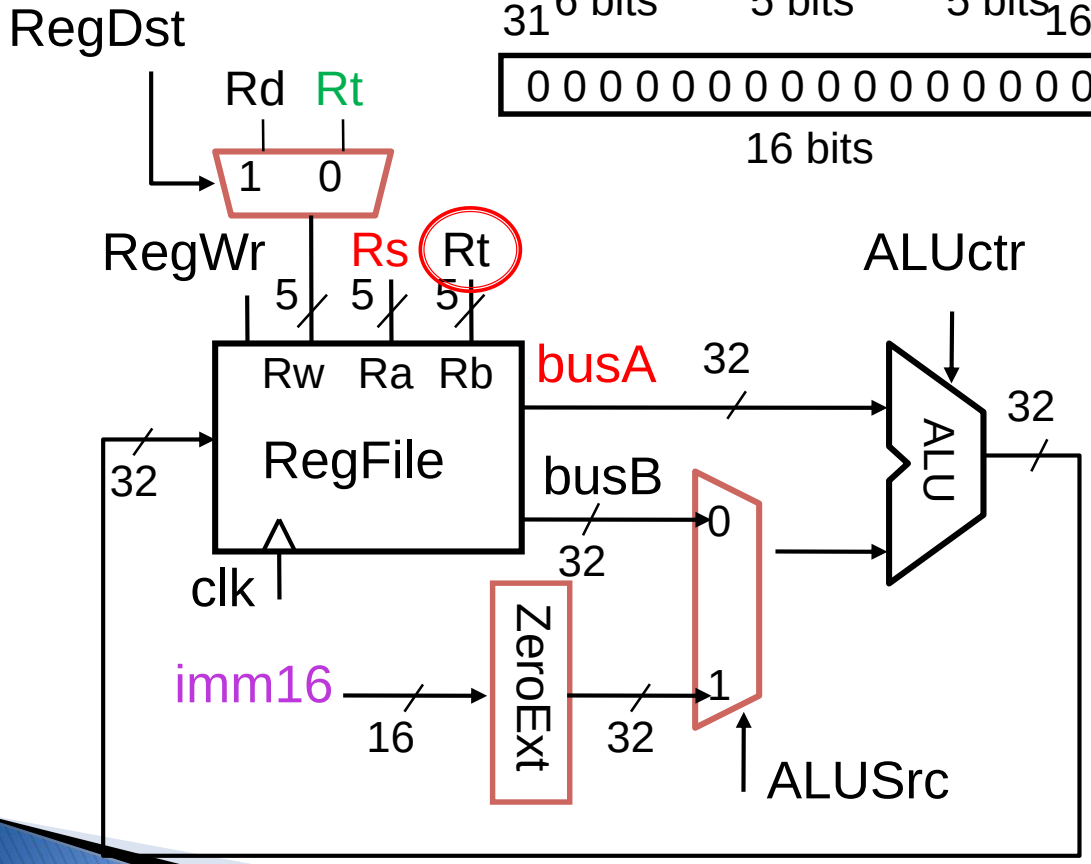
- R[rt] = R[rs] op ZeroExt[imm16]]



Operations with Immediate

- R[rt] = R[rs] op ZeroExt[imm16]]





What about Rt register read (bus B)??

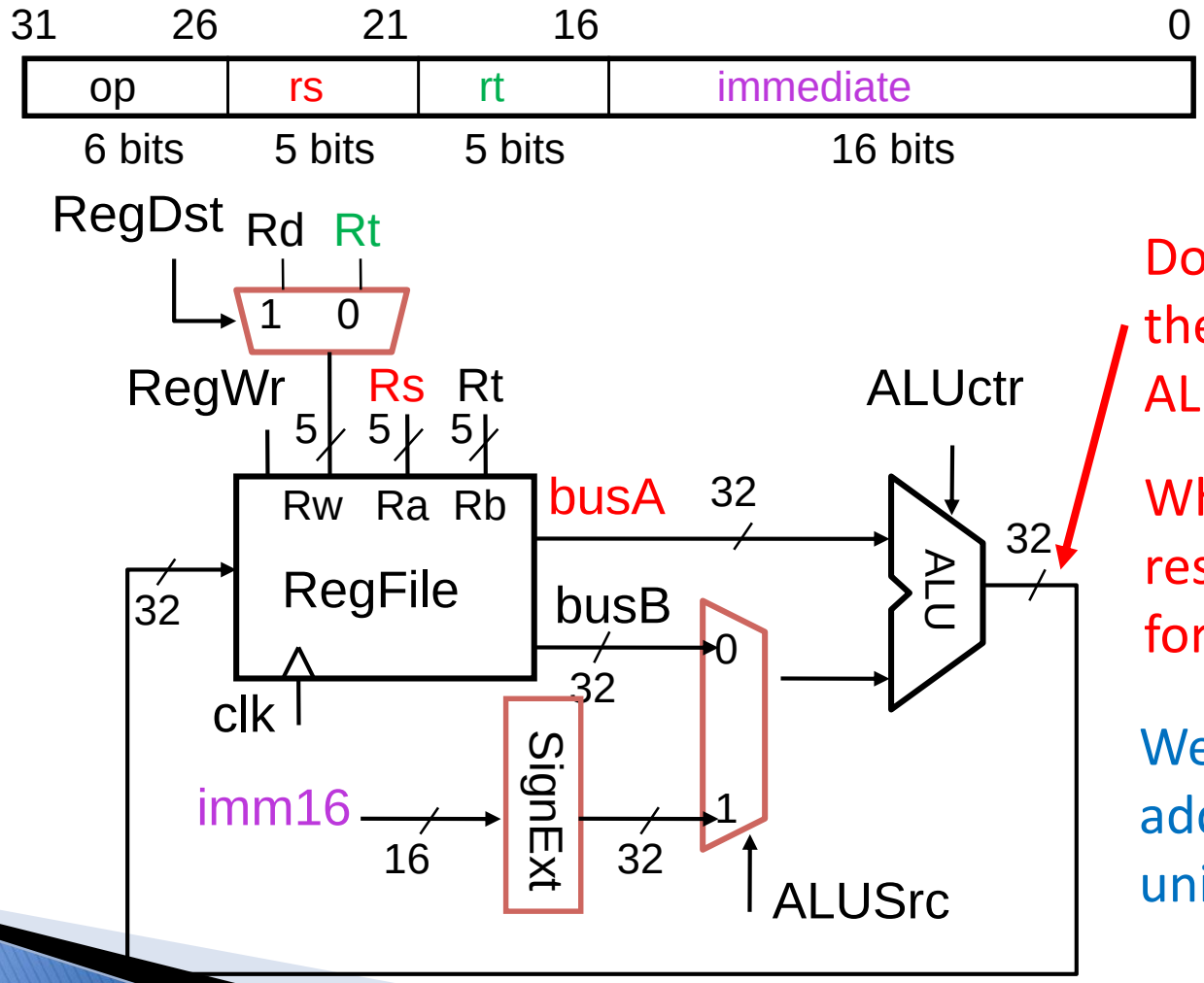
We don't care!

Why?

Bus B is blocked

Load Memory

- R[rt] = Mem[R[rs] + SignExt[imm16]]
- Example: `lw rt,rs,imm16`



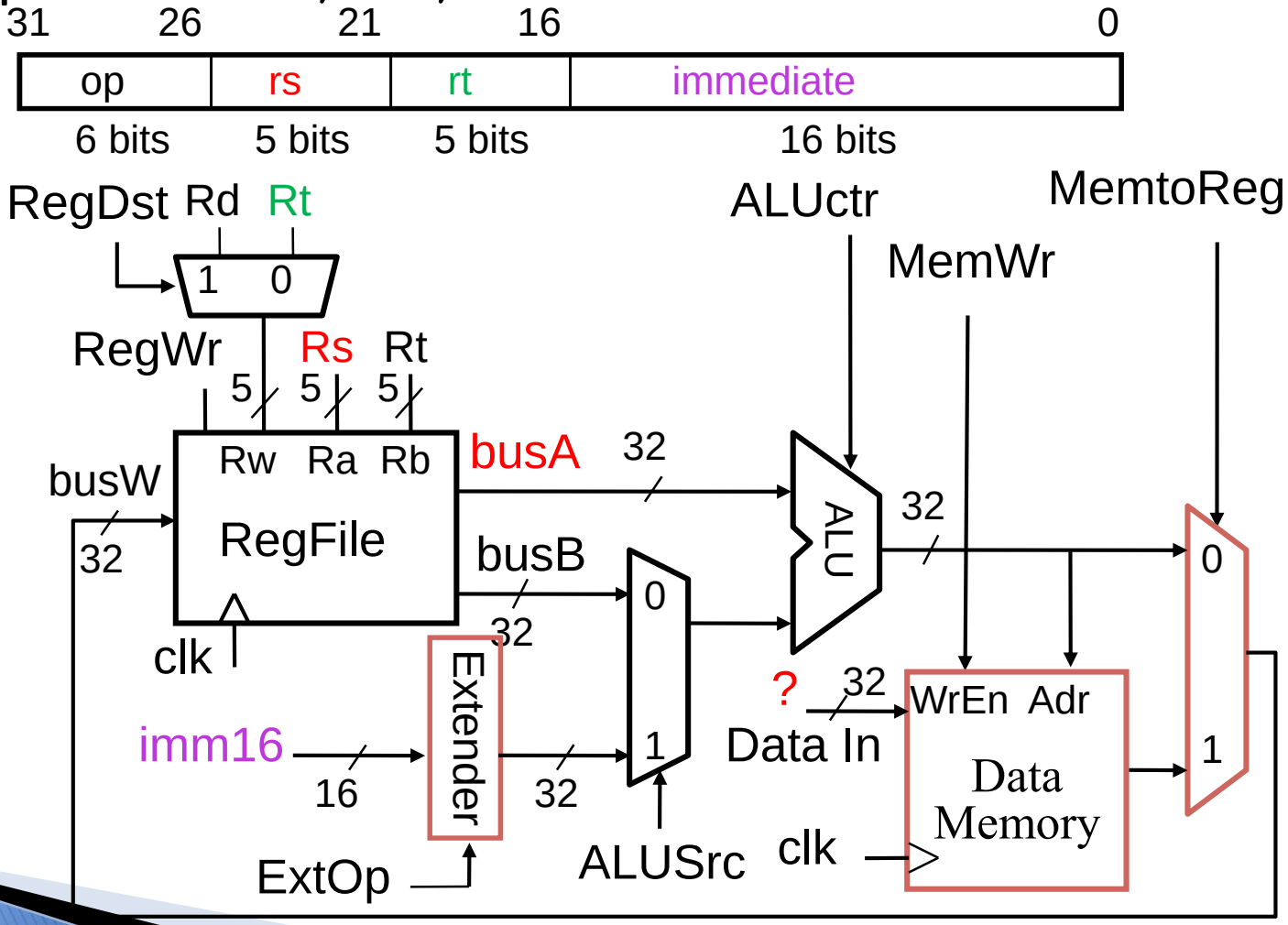
Do we store the result of ALU?

What is the result of ALU for?

We need to add a memory unit!

Load Memory

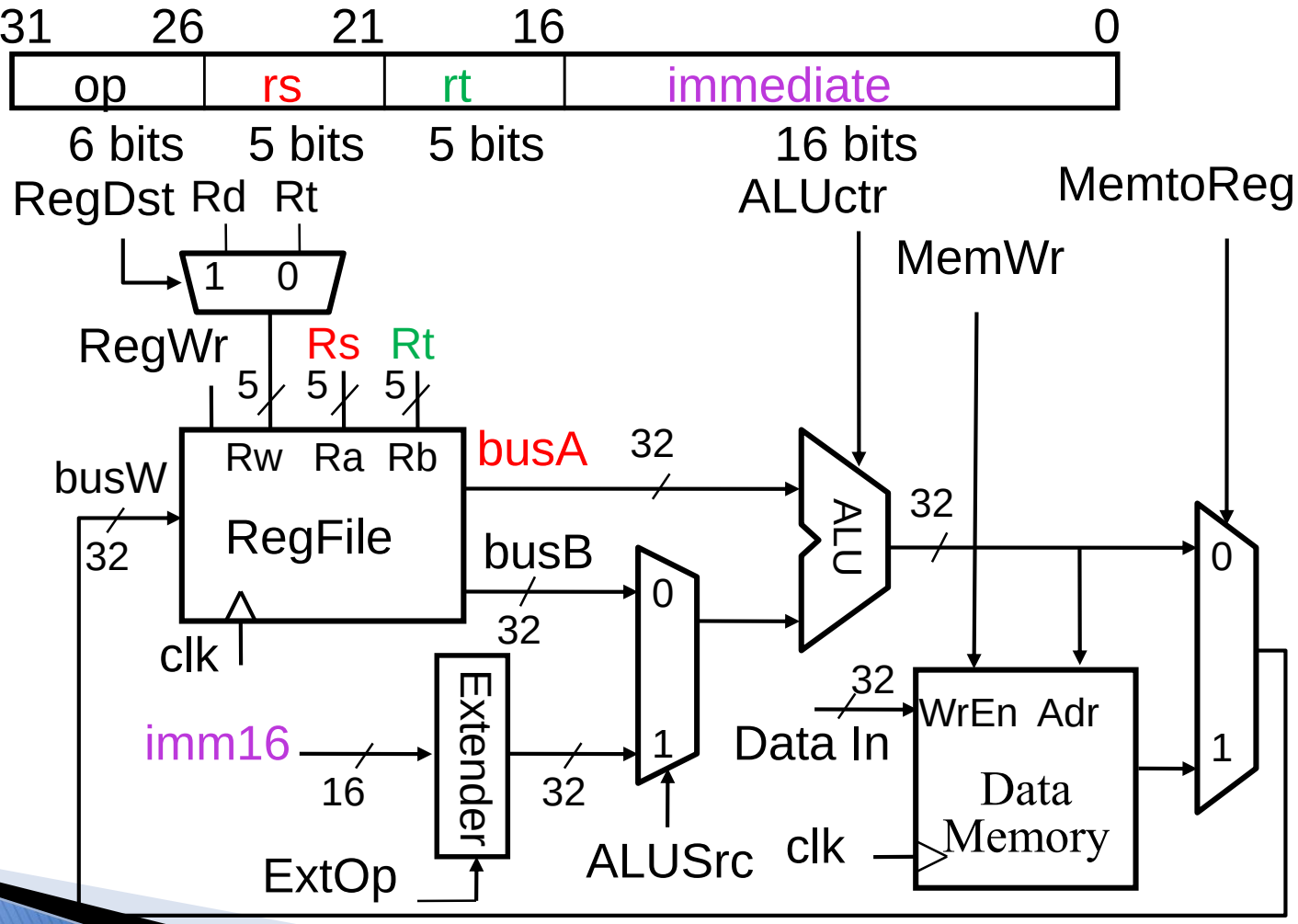
- R[rt] = Mem[R[rs] + SignExt[imm16]]
- Example: `lw rt,rs,imm16`



Store Memory

- Mem[R[rs] + SignExt[imm16]] = R[rt]

  Ex.: `sw rt, rs, imm16`



Store Memory

- Mem[R[rs] + SignExt[imm16]] = R[rt]

  Ex.: `sw rt, rs, imm16`