CprE 381: Computer Organization and Assembly Level Programming

Processor Design

Henry Duwe
Electrical and Computer Engineering
Iowa State University

Administrative

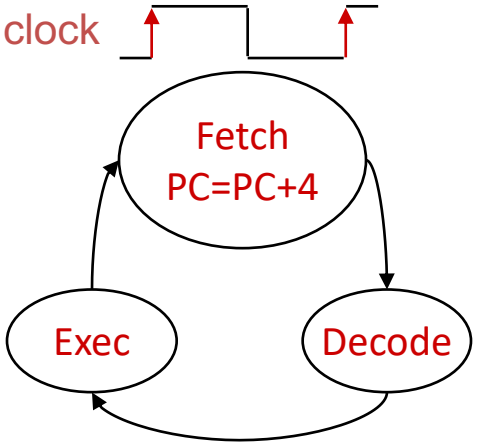
- Exam 1
  - Graded, returned on Canvas
  - Mean/Median/Mode = 60
    - Min:max: 8:99
  - Will curve, currently raw score on Canvas
    - If you scored <40%, you should be very concerned; come see me
  - Any grading/counting/scoring issues:
    - In writing as e-mail to me

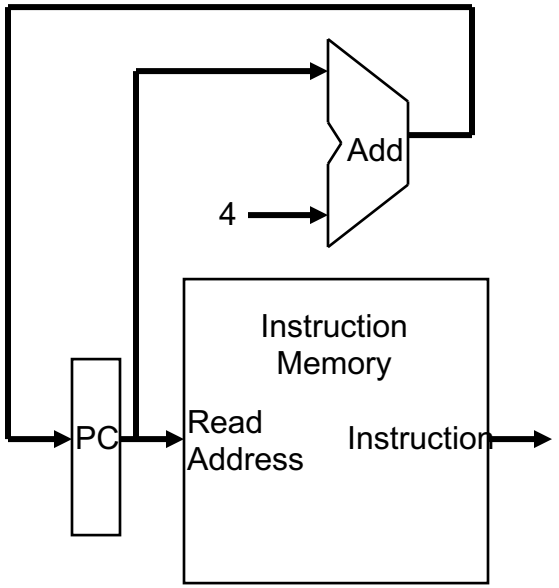
Administrative

- HW4
  - Redo problem you lost the most points on (total points).
  - Make it perfect
    - Ask questions
    - Talk with each other, but make sure your solution write-up is yours
  - Turn it in as pdf on Canvas

Review: Fetching Instructions

- Fetching instructions involves
  - reading the instruction from the Instruction Memory
  - updating the PC value to be the address of the next (sequential) instruction

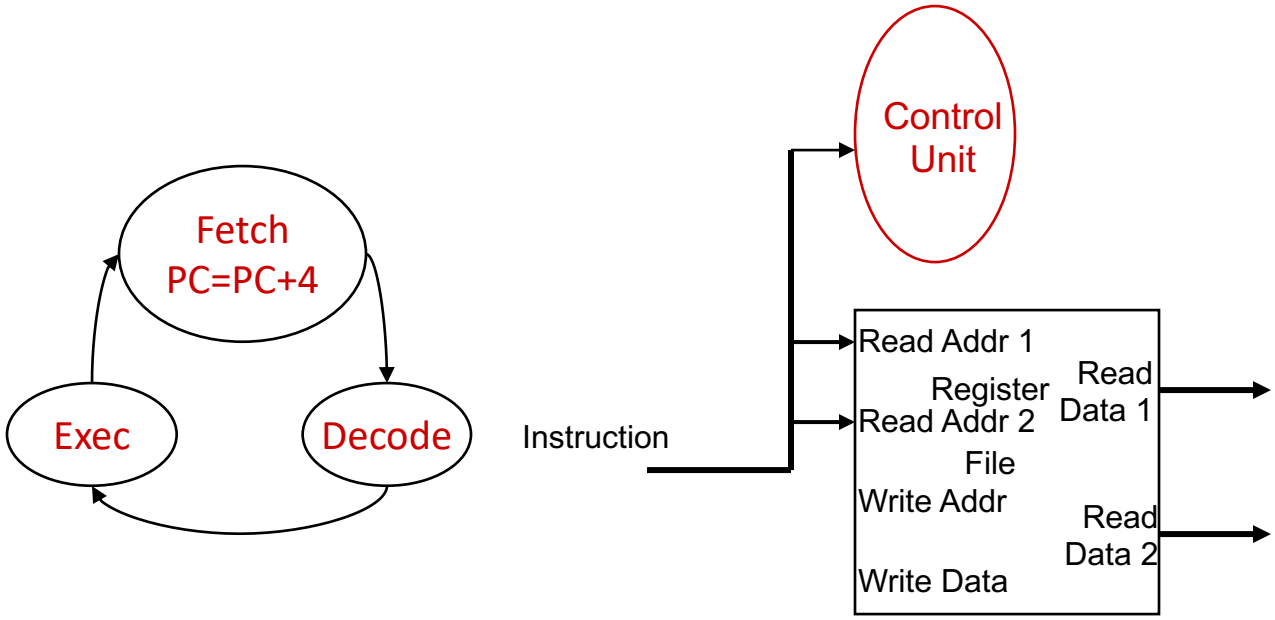




  - PC is updated every clock cycle, so it does not need an explicit write control signal – just a clock signal
  - Reading from the Instruction Memory is a combinational activity

Review: Decoding Instructions

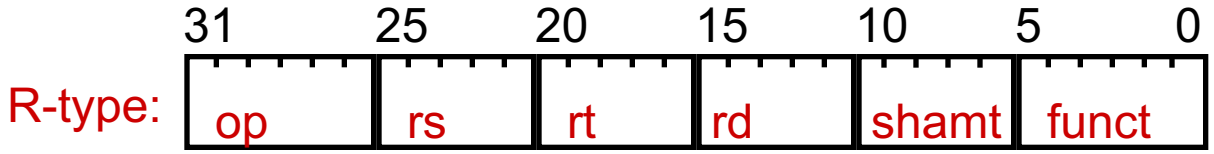
- Decoding instructions involves
  - Sending the fetched instruction's opcode and function field bits to the control unit



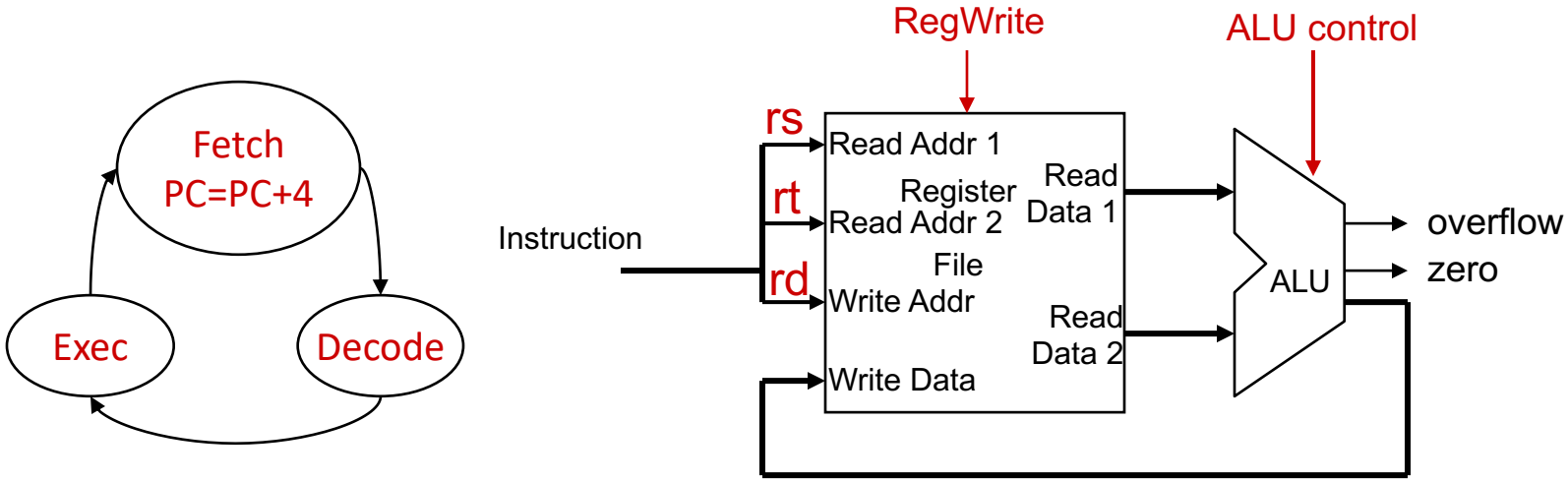
  - Reading two values from the Register File
    - Register File addresses are contained in the instruction

Review: Executing R Format Operations

- R format operations (**`add, sub, slt, and, or`**)



  - Perform operation (op and funct) on values in rs and rt
  - Store the result back into the Register File (into location rd)



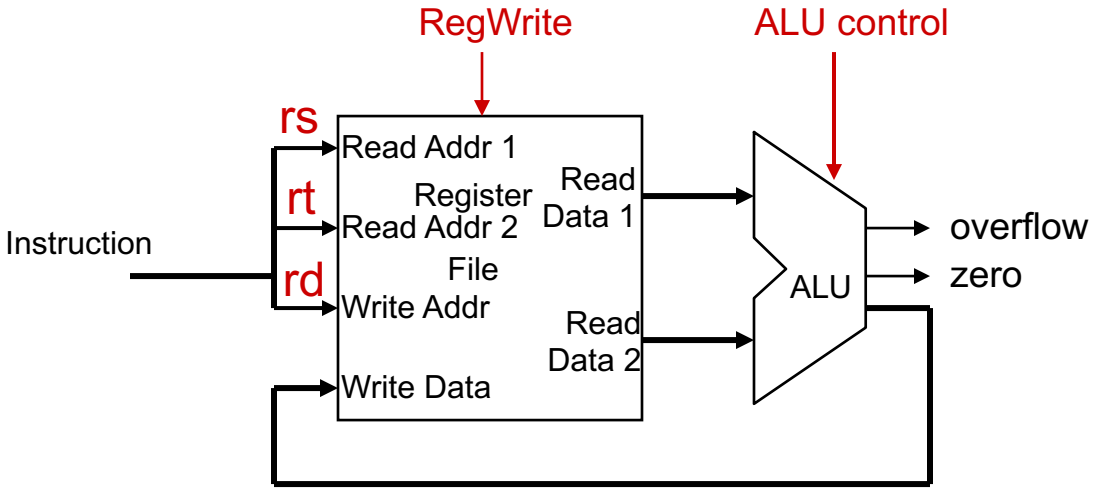
  - Note that Register File is not written every cycle (e.g. `sw` or `jr`), so we need an explicit write control signal for the Register File

Consider the `slt` Instruction

- Remember the R format instruction **`slt`**

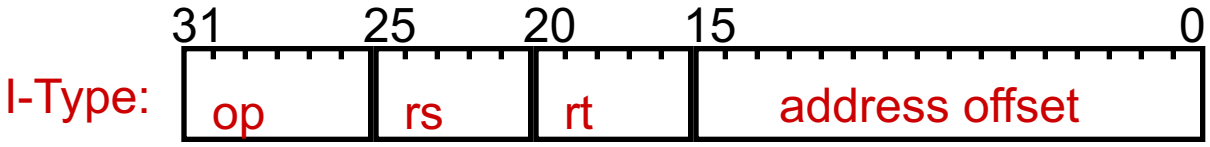
```
slt $t0, $s0, $s1   # if $s0 < $s1
                    #   then  $t0 = 1
                    # else  $t0 = 0
```

  - Where does the 1 (or 0) come from to store into $t0 in the Register File at the end of the execute cycle?



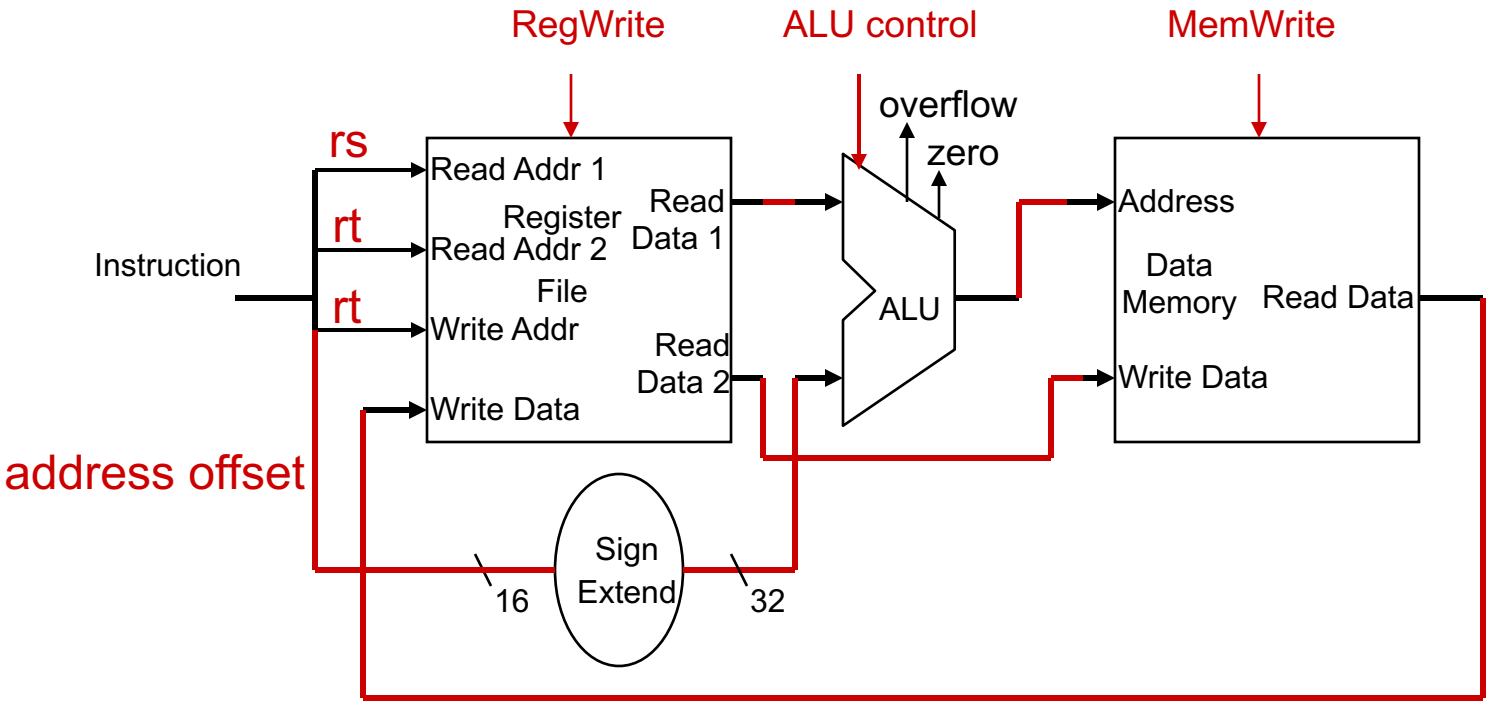
Review: Executing Load and Store Operations

- Load and store operations have to:



  - Compute a memory address by adding the base register (in rs) to the 16-bit signed offset field in the instruction
    - Base register was read from the Register File during decode
    - Offset value in the low order 16 bits of the instruction must be sign extended to create a 32-bit signed value
  - Store value, read from the Register File during decode, must be written to the Data Memory
  - Load value, read from the Data Memory, must be stored in the Register File

Executing Load / Store Operations (cont.)

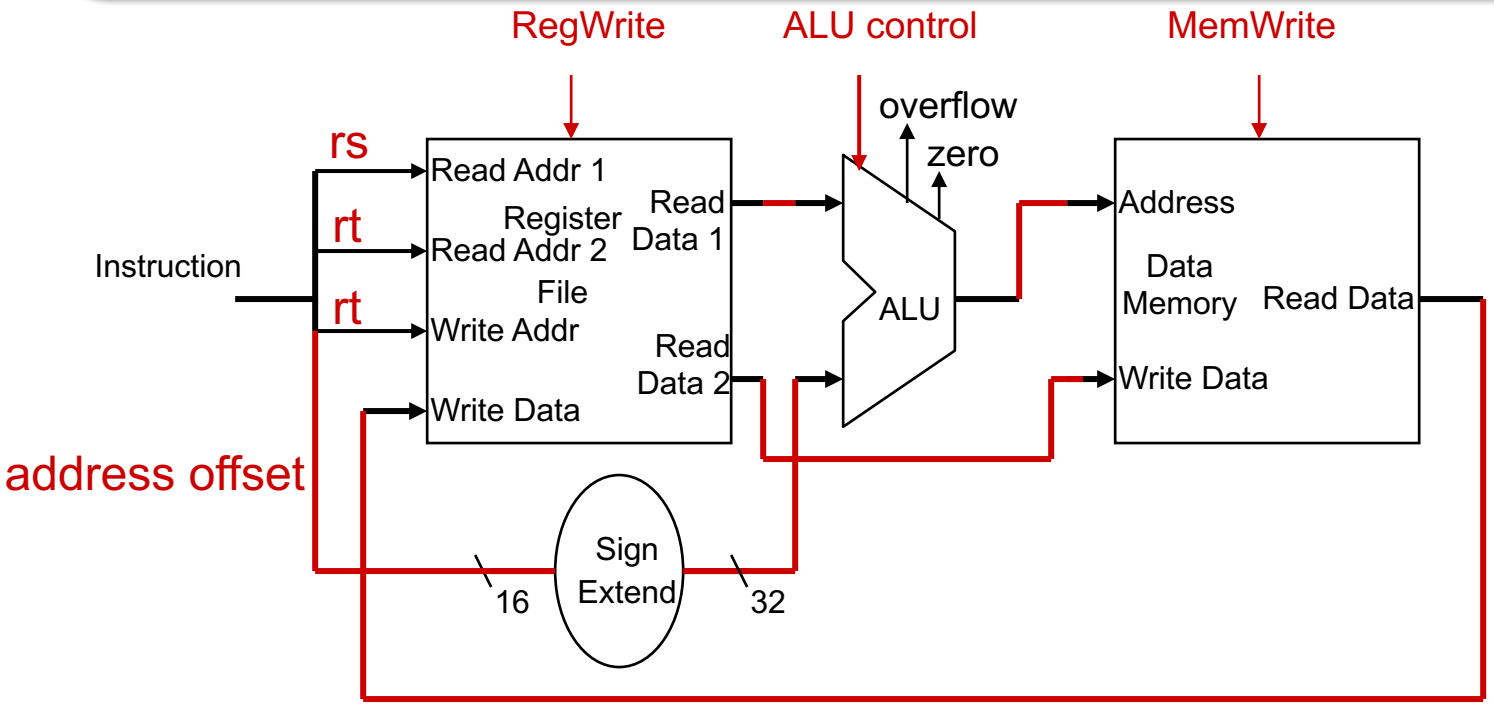


Executing Load / Store Operations (cont.)

In-class Assessment!

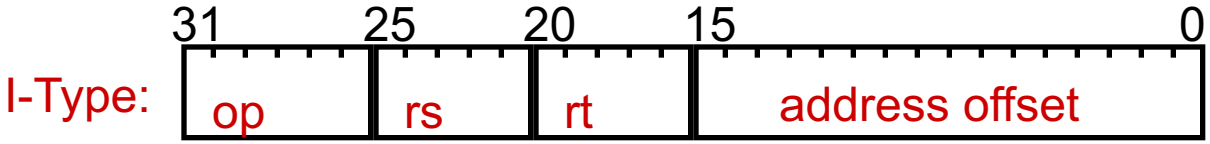
Access Code: RTGIF381

Note: sharing access code to those outside of classroom or using access while outside of classroom is considered cheating



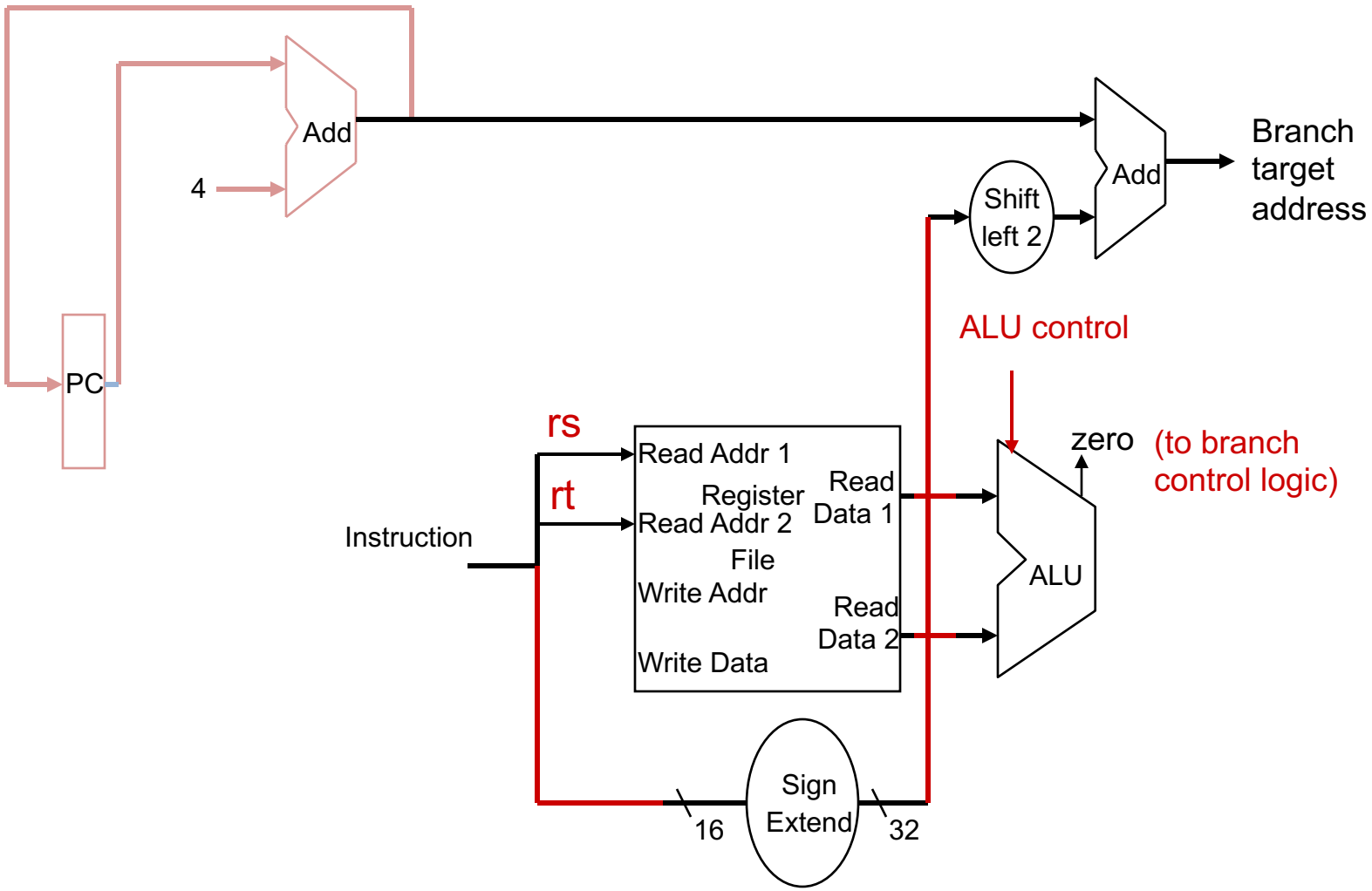
Executing Branch Operations

- Branch operations have to



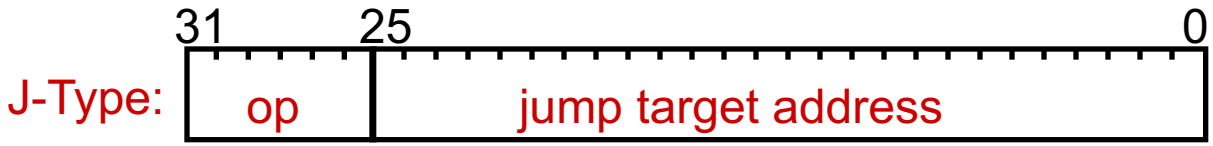
  - Compare the operands read from the Register File during decode (rs and rt values) for equality (**`zero`** ALU output)
  - Compute the branch target address by adding the updated PC to the sign extended16-bit signed offset field in the instruction
    - The "base register" is the updated PC
    - Offset value in the low order 16 bits of the instruction must be sign extended to create a 32-bit signed value and then shifted left 2 bits to turn it into a word address

Executing Branch Operations (cont.)

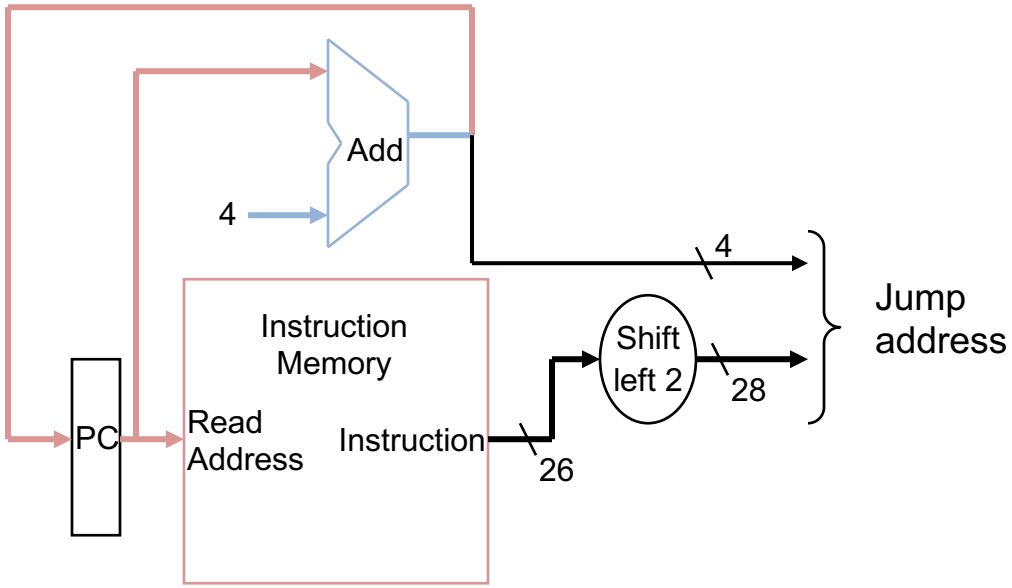


Executing Jump Operations

- Jump operations have to



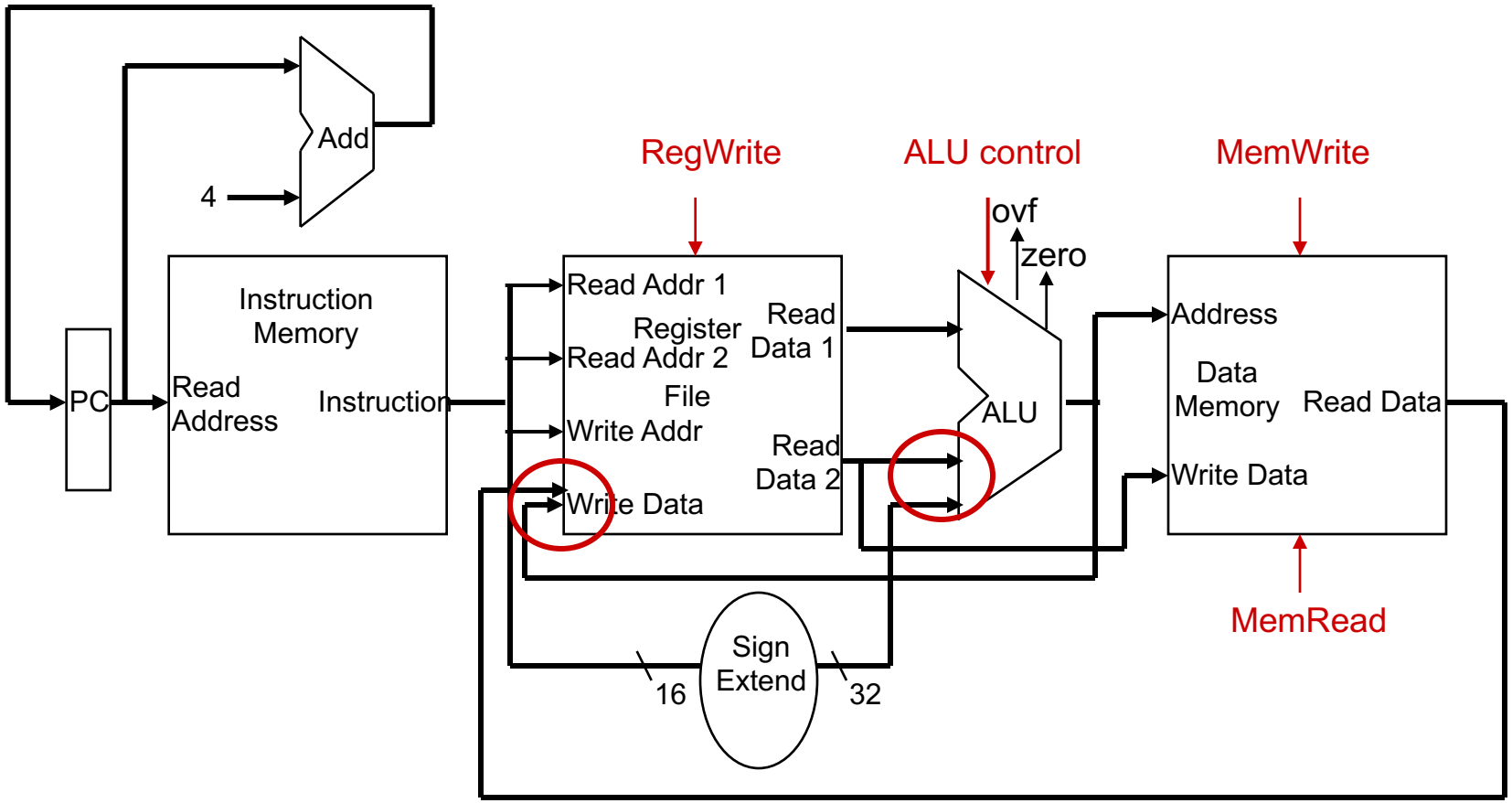
  - Replace the lower 28 bits of the PC with the lower 26 bits of the fetched instruction shifted left by 2 bits



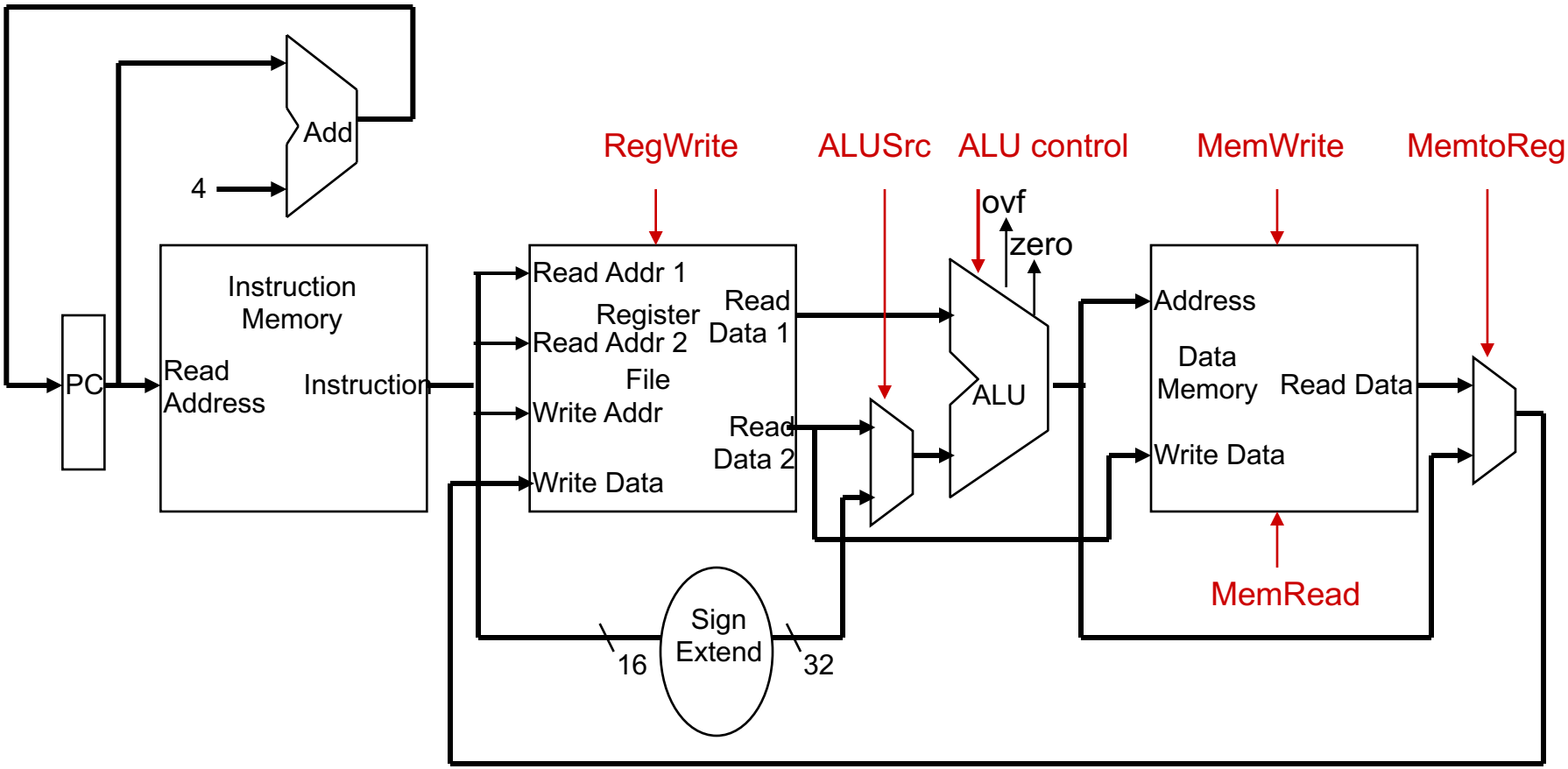
Creating a Single Datapath from the Parts

- Assemble the datapath elements, add control lines as needed, and design the control path
- Fetch, decode and execute each instructions in one clock cycle – single cycle design
  - One instruction can't use same resource/structure twice (ergo Harvard split memory architecture)
  - Two different instructions need multiplexors at the input of the shared elements with control lines to do the selection
- Cycle time is determined by length of the longest path

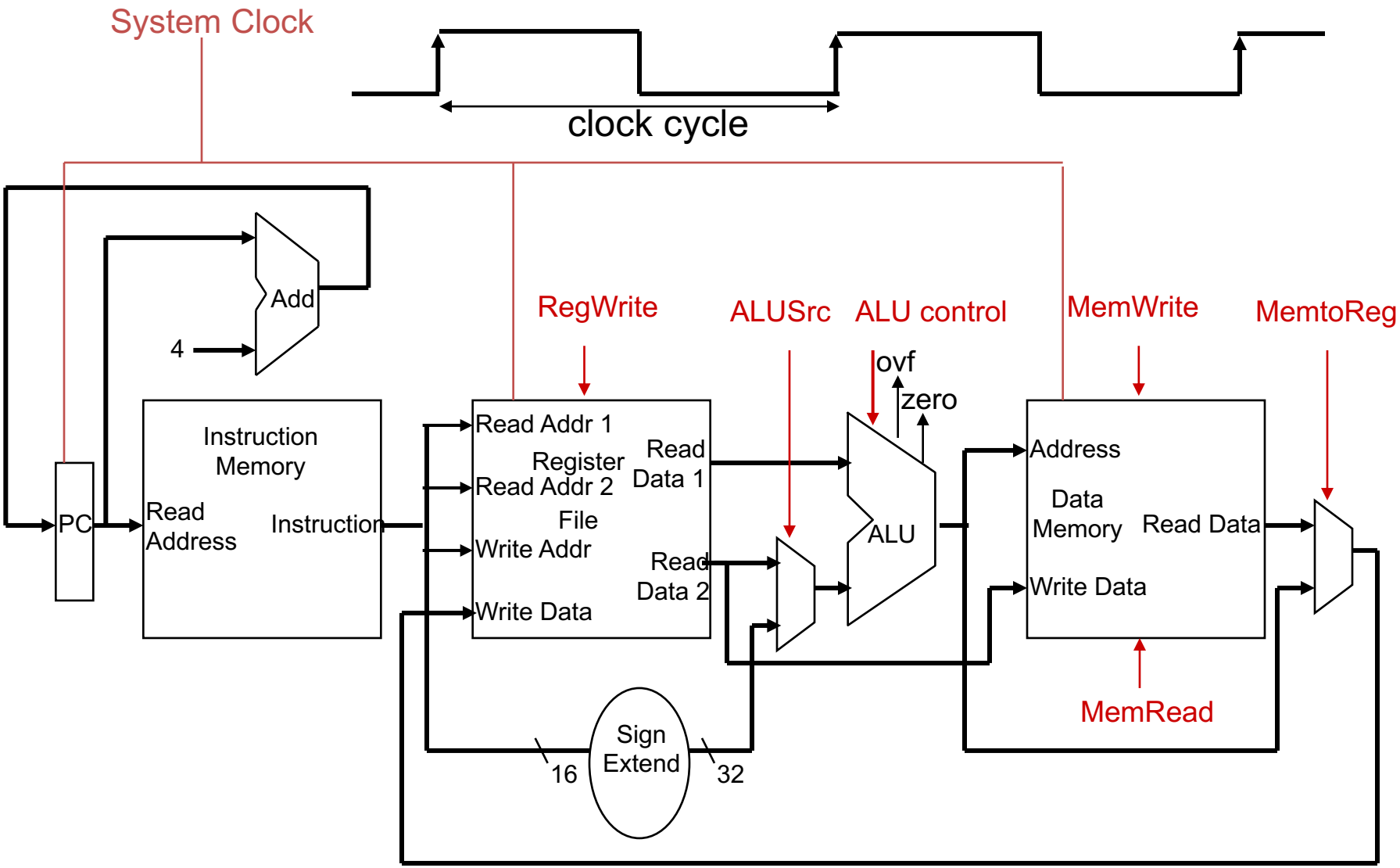
Fetch, R, and Memory Access Portions



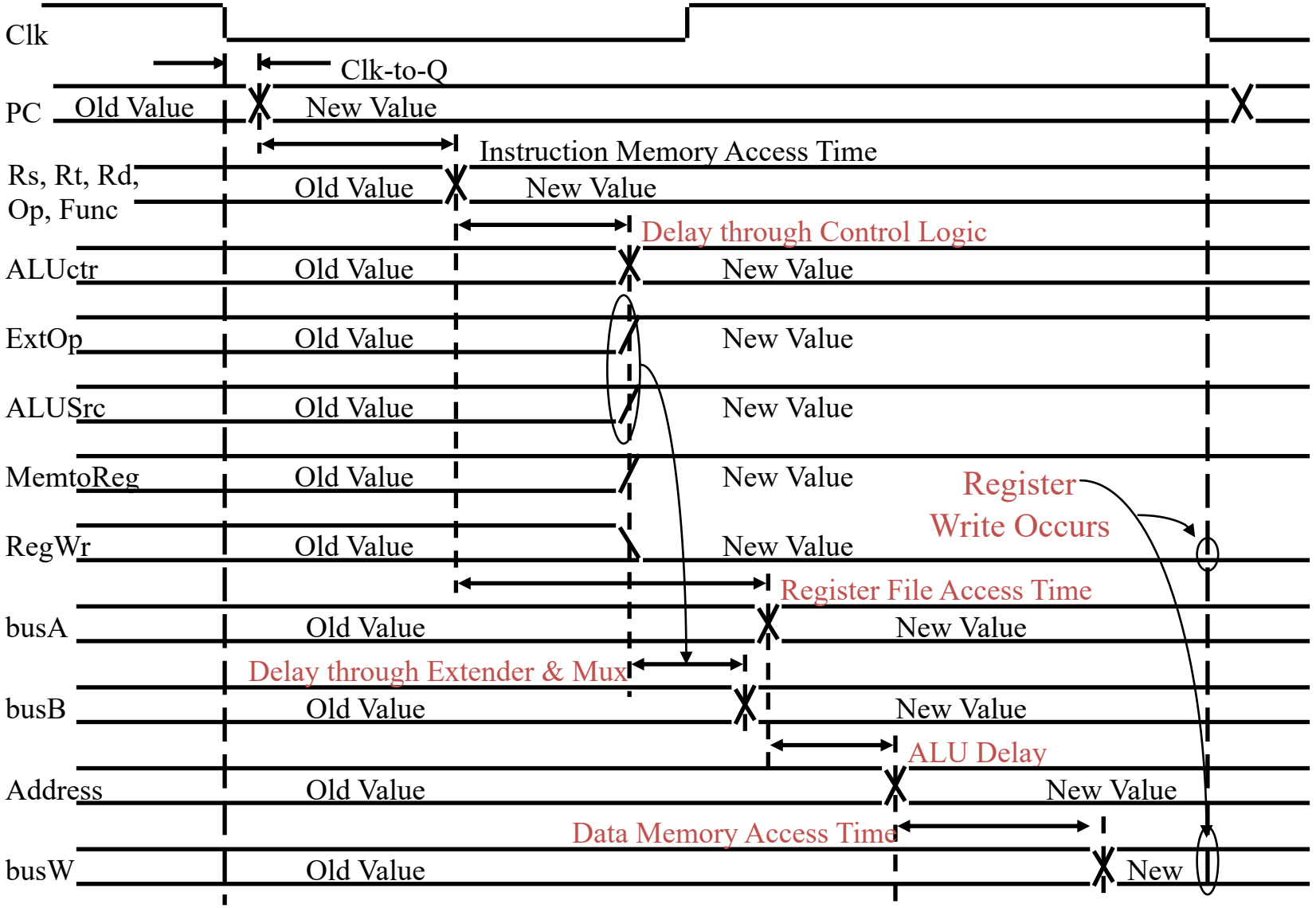
Multiplexor Insertion



Clock Distribution

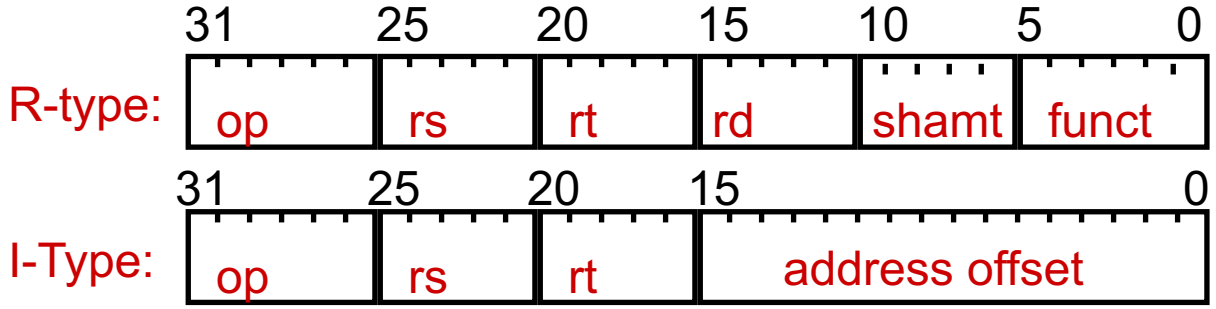


Preview: Worst Case Timing (Load Instruction)



Preview: Adding the Control

- Selecting the operations to perform (ALU, Register File and Memory read/write)
- Controlling the flow of data (multiplexor inputs)
- Information comes from the 32 bits of the instruction



- Observations
  - op field always in bits 31-26
  - addr of two registers to be read are always specified by the rs and rt fields (bits 25-21 and 20-16)
  - base register for `lw` and `sw` always in rs (bits 25-21)
  - addr. of register to be written is in one of two places – in rt (bits 20-16) for `lw`; in rd (bits 15-11) for R-type instructions
  - offset for `beq`, `lw`, and `sw` always in bits 15-0

Acknowledgments

- These slides contain material developed and copyright by:
  - Joe Zambreno (Iowa State)
  - David Patterson (UC Berkeley)
  - Mary Jane Irwin (Penn State)
  - Christos Kozyrakis (Stanford)
  - Onur Mutlu (Carnegie Mellon)
  - Krste Asanović (UC Berkeley)