

## 10. Basic Processor Design: Single-Cycle and Multi-Cycle Datapaths

---

EECS 370 – Introduction to Computer Organization – Winter 2009

**Prof. Todd Austin & Prof. Marios Papaefthymiou**

EECS Department  
University of Michigan in Ann Arbor, USA

© T. Austin & M. Papaefthymiou, 2009

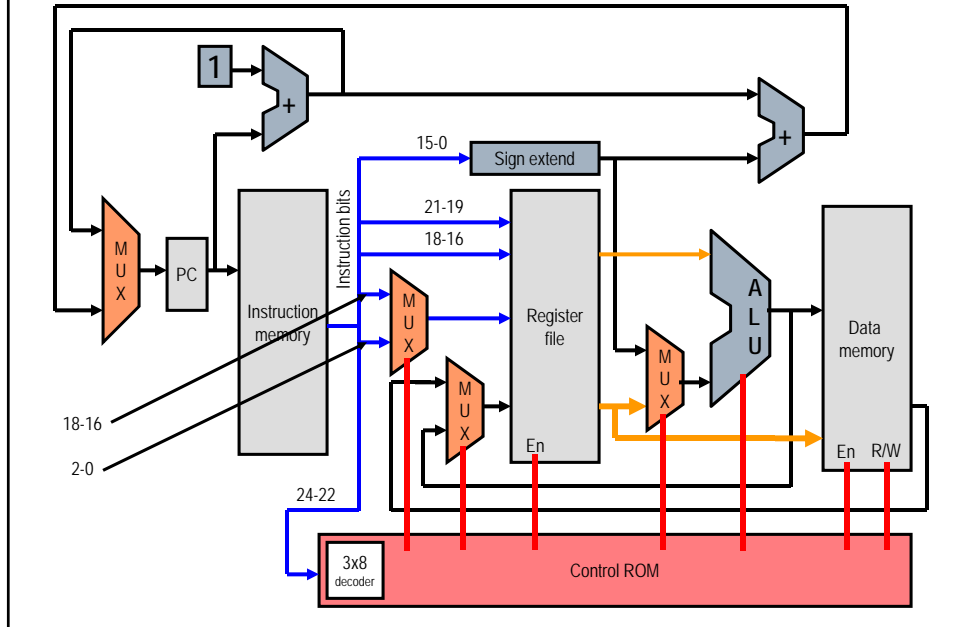
The material in this presentation cannot be  
copied in any form without our written permission

### Announcements

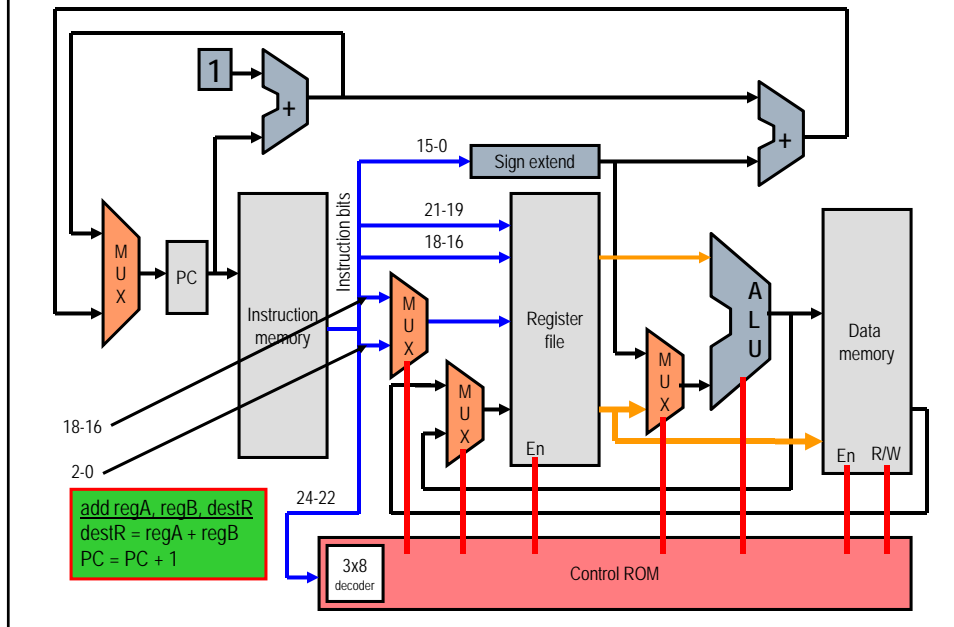
---

- ❑ Exam 1 – Thursday, February 19 in class
  - Open book, open notes
  - But, no laptops, only simple calculators
- ❑ Exam 1 review
  - Tuesday, February 17 in class
- ❑ Grades
  - HW1 is graded and available outside my office (CSE 4637)

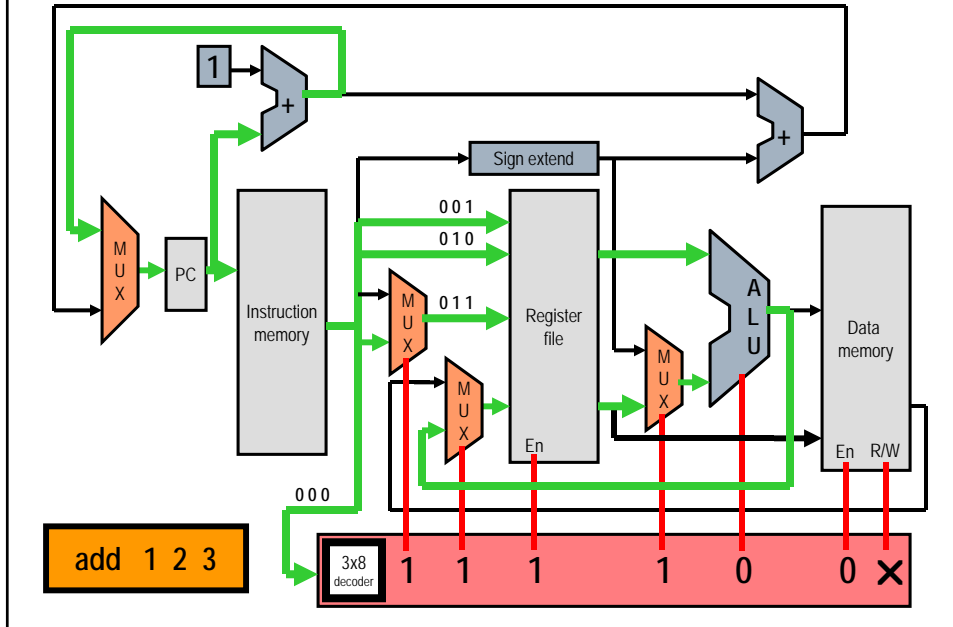
## LC2Kx Datapath Implementation



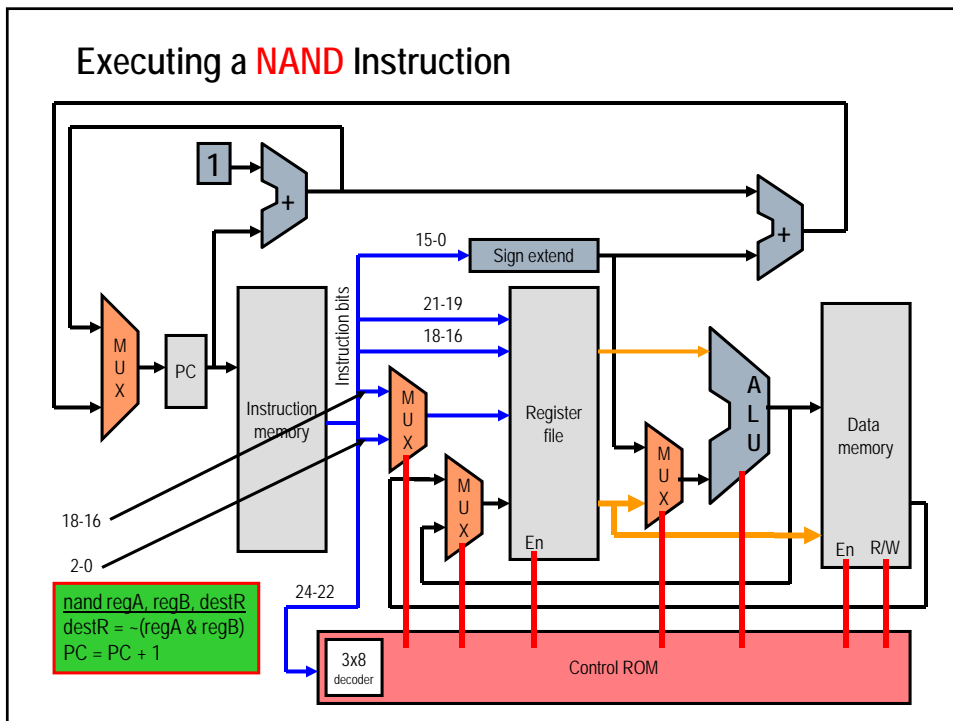
## Executing an **ADD** Instruction



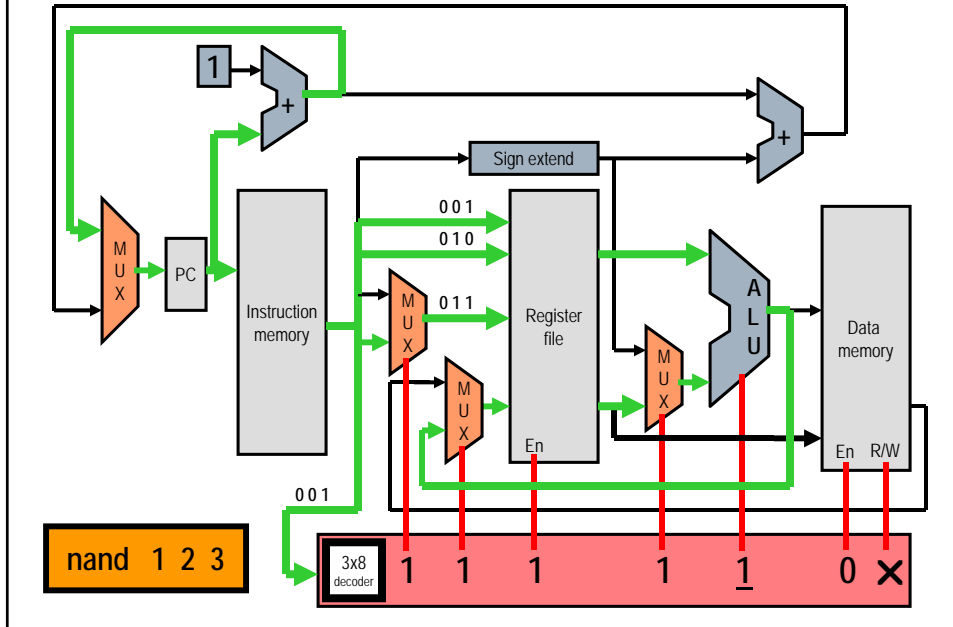
### Executing an **ADD** Instruction on LC2Kx Datapath



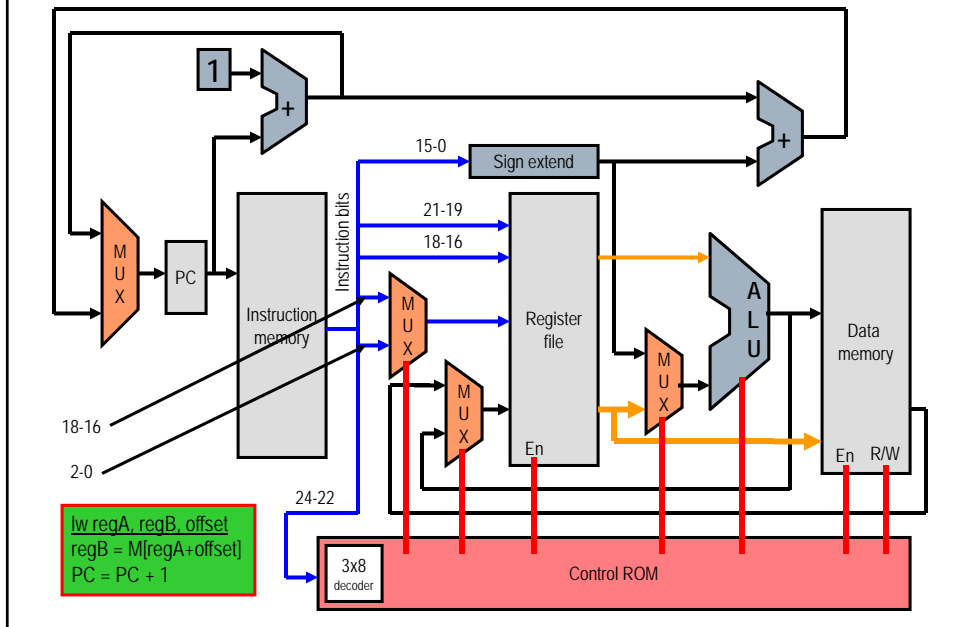
### Executing a **NAND** Instruction



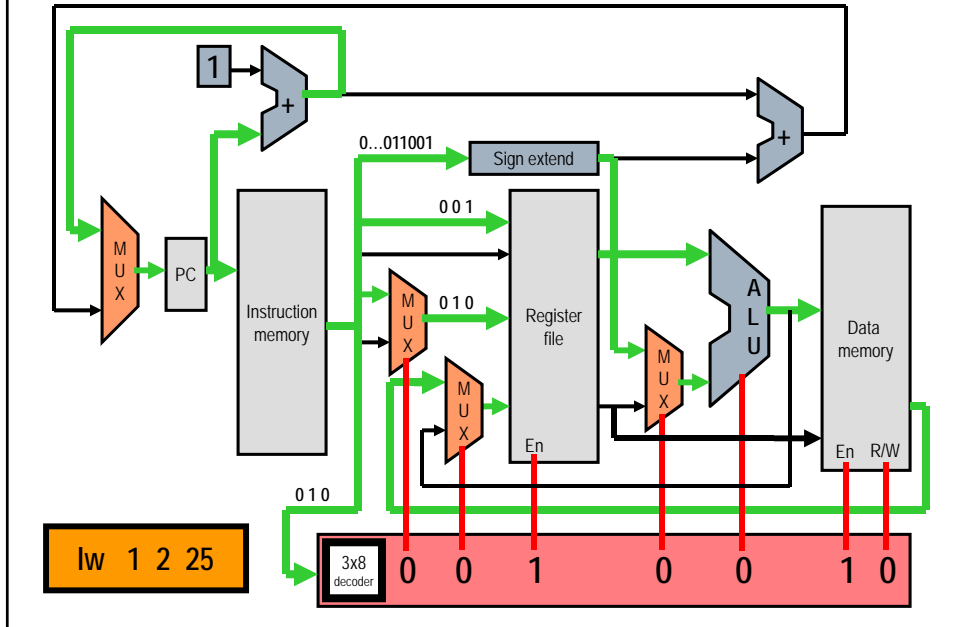
### Executing NAND Instruction on LC2Kx Datapath



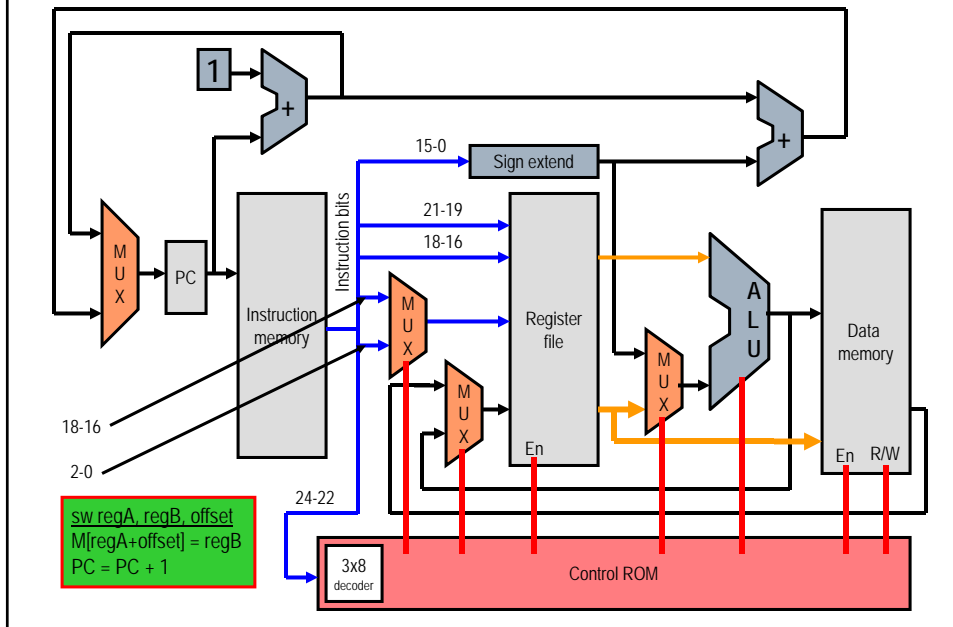
### Executing a LW Instruction



### Executing a **LW** Instruction on LC2Kx Datapath

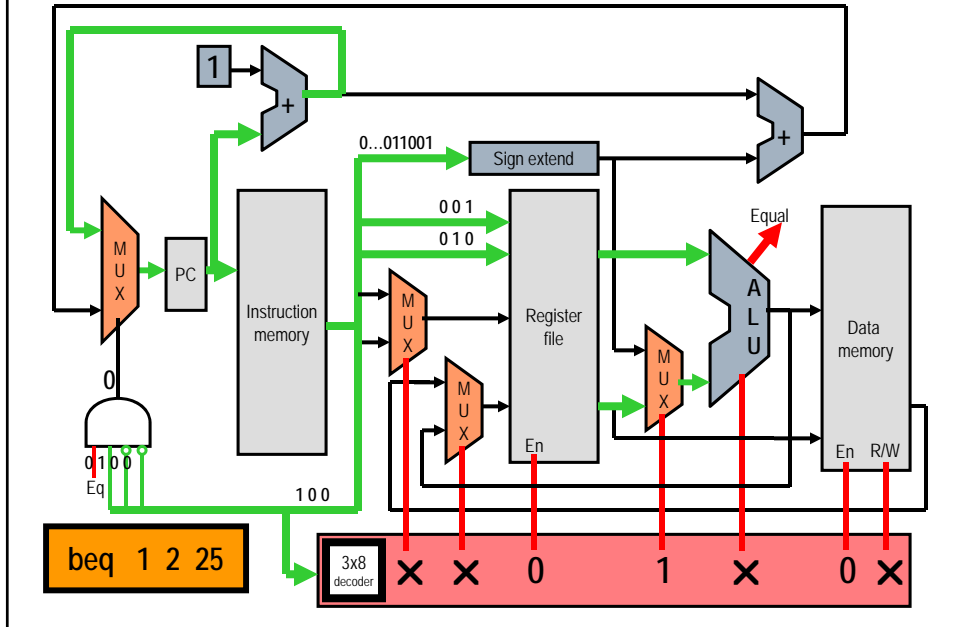


### Executing a **SW** Instruction

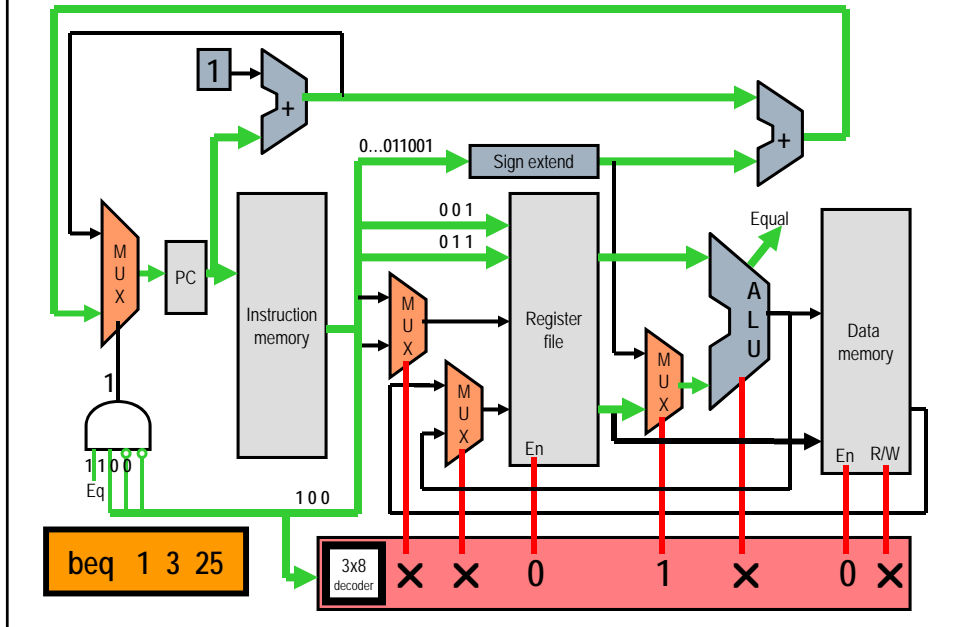




### Executing a "not taken" BEQ Instruction on LC2K Datapath



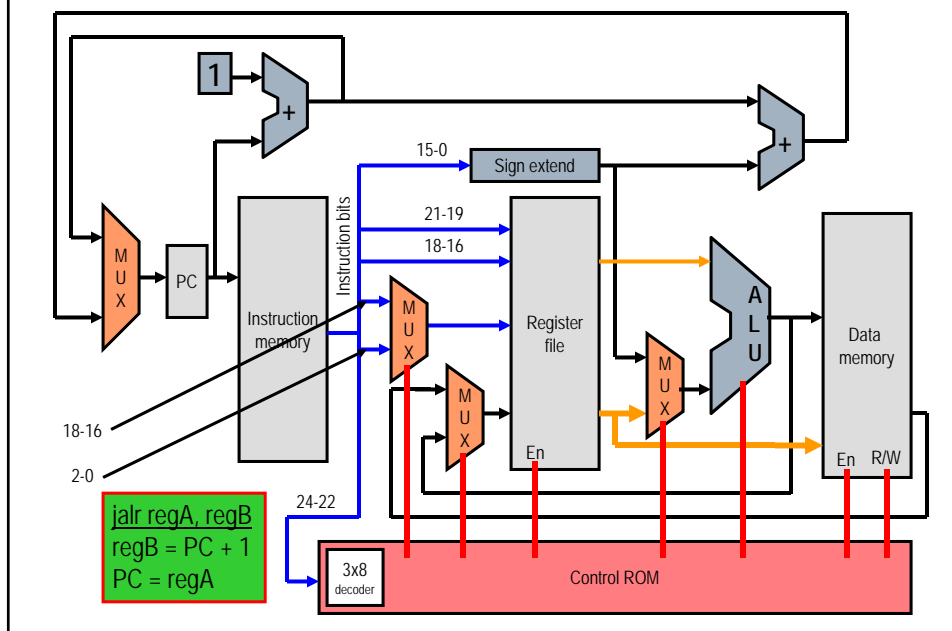
### Executing a "taken" BEQ Instruction on LC2K Datapath



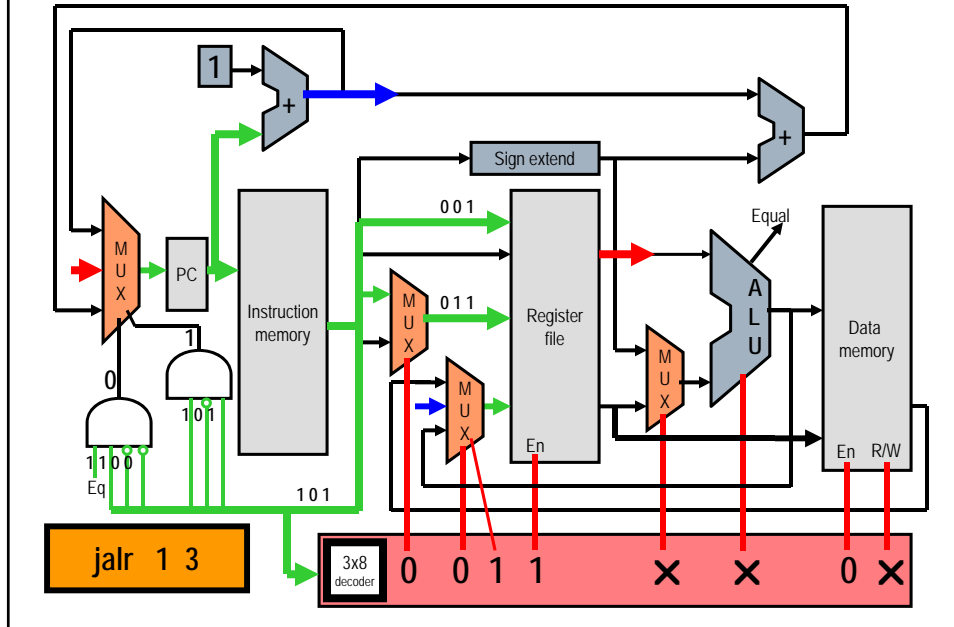
## So Far, So Good

- ❑ Every architecture seems to have at least one ugly instruction.
  - JALR doesn't fit into our nice clean datapath
  - To implement JALR we need to
    - Write PC+1 into regB
    - Move regA into PC
  - Right now there is:
    - No path to write PC+1 into a register
    - No path to write a register to the PC

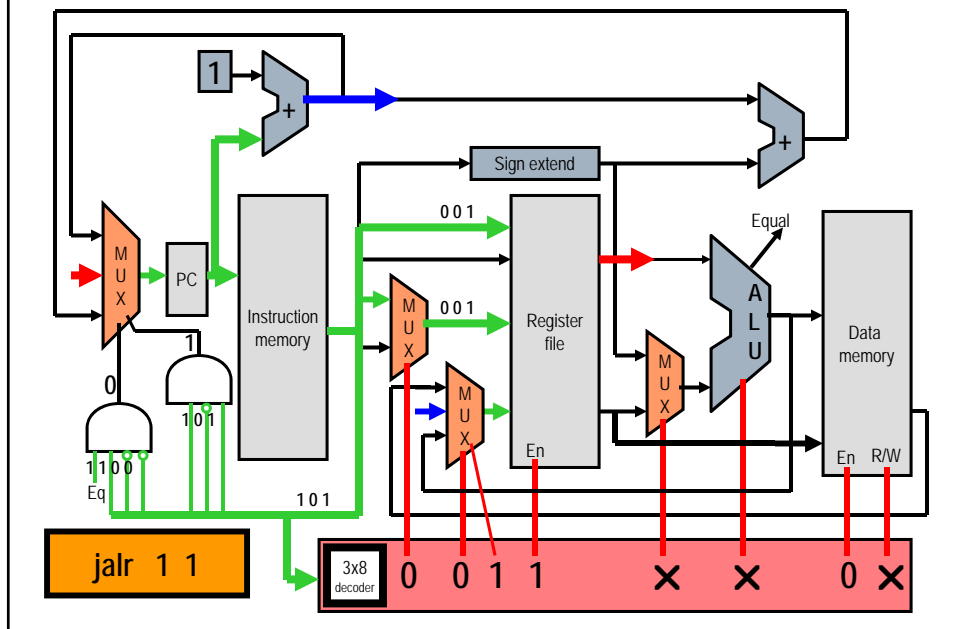
## Executing a **JALR** Instruction



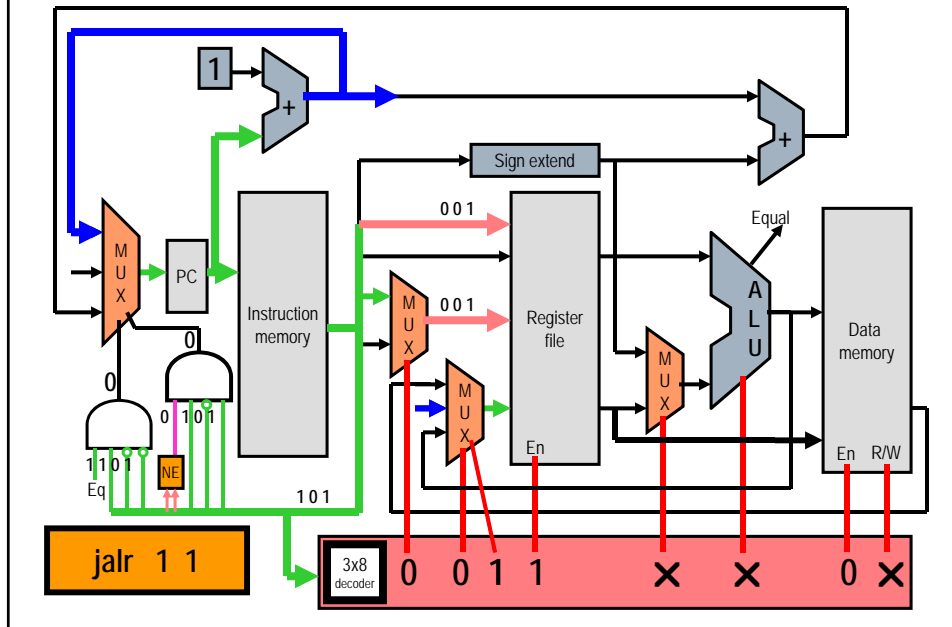
### Executing a **JALR** Instruction on LC2Kx Datapath



### What If `regA = regB` for a **JALR** ?



## Changes for a JALR 1 1 Instruction



## Class Problem

- Extend the single cycle datapath to perform the following operation
  - `cmov regA, regB, destR`
    - `destR = regA` (if `regB != 0`)
    - `PC = PC + 1`



## What's Wrong with Single Cycle?

- 1 ns – Register read/write time
- 2 ns – ALU/adder
- 2 ns – memory access
- 0 ns – MUX, PC access, sign extend, ROM

	Get Instr	read reg	ALU oper.	mem	write reg	
• add:	2ns	+ 1ns	+ 2ns		+ 1ns	= 6 ns
• beq:	2ns	+ 1ns	+ 2ns			= 5 ns
• sw:	2ns	+ 1ns	+ 2ns	+ 2ns		= 7 ns
• lw:	2ns	+ 1ns	+ 2ns	+ 2ns	+ 1ns	= 8 ns

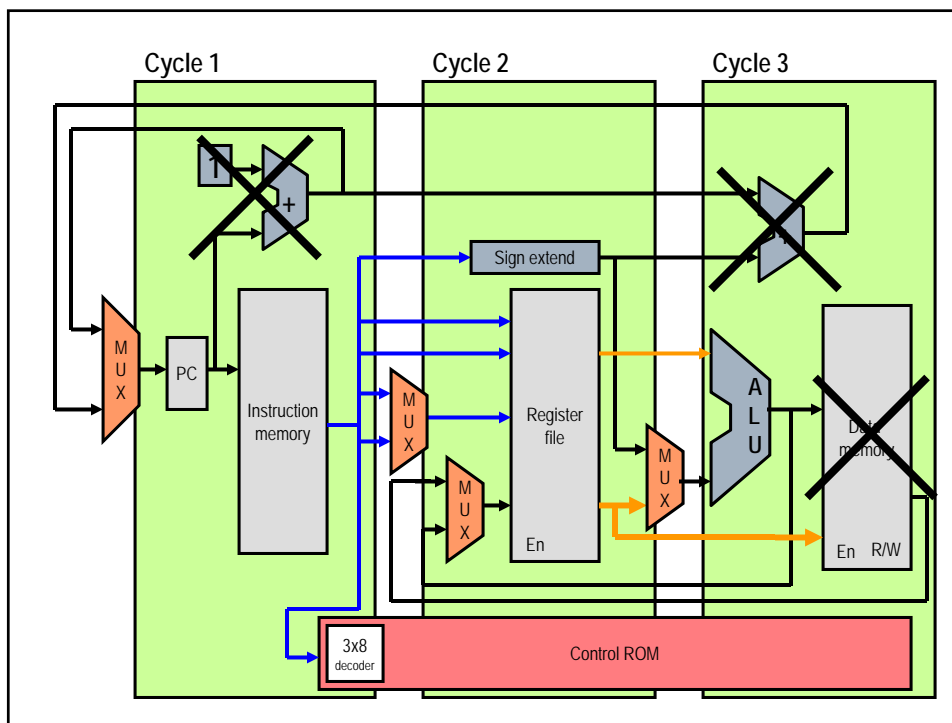
## Computing Execution Time

- ❑ Assume: 100 instructions executed
  - 25% of instructions are loads,
  - 10% of instructions are stores,
  - 45% of instructions are adds, and
  - 20% of instructions are branches.
- ❑ Single-cycle execution:  
= ??
- ❑ Optimal execution:  
= ??

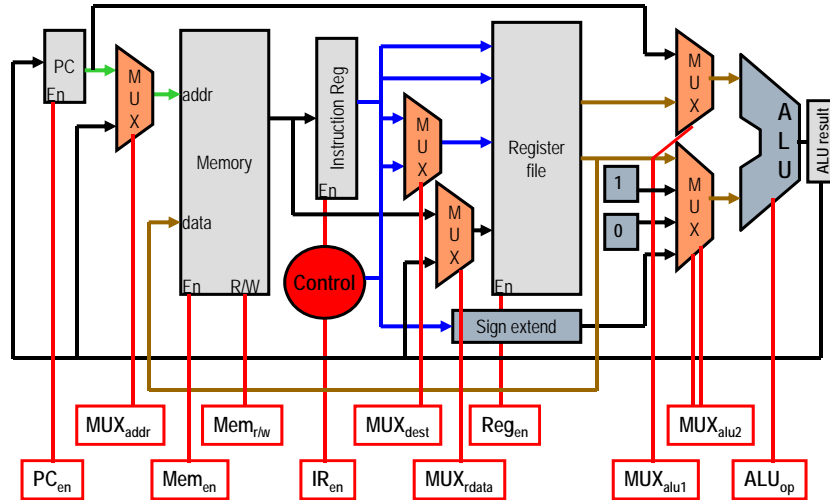
**END OF EXAM1 MATERIAL**

## Multiple-Cycle Execution

- ❑ Each instruction takes multiple cycles to execute
  - Cycle time is reduced
  - Slower instructions take more cycles
  - Can reuse datapath elements each cycle
- ❑ What is needed to make this work?
  - Since you are re-using elements for different purposes, you need more and/or wider MUXes.
  - You may need extra registers if you need to remember an output for 1 or more cycles.
  - Control is more complicated since you need to send new signals on each cycle.



## Multicycle LC2Kx Datapath – More on this Next Time



27/27