

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

EECS 370 – Introduction to Computer Organization – Winter 2007

Prof. Valeria Bertacco & Prof. Scott Mahlke

EECS Department
University of Michigan in Ann Arbor, USA

© V. Bertacco & S. Mahlke, 2007

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

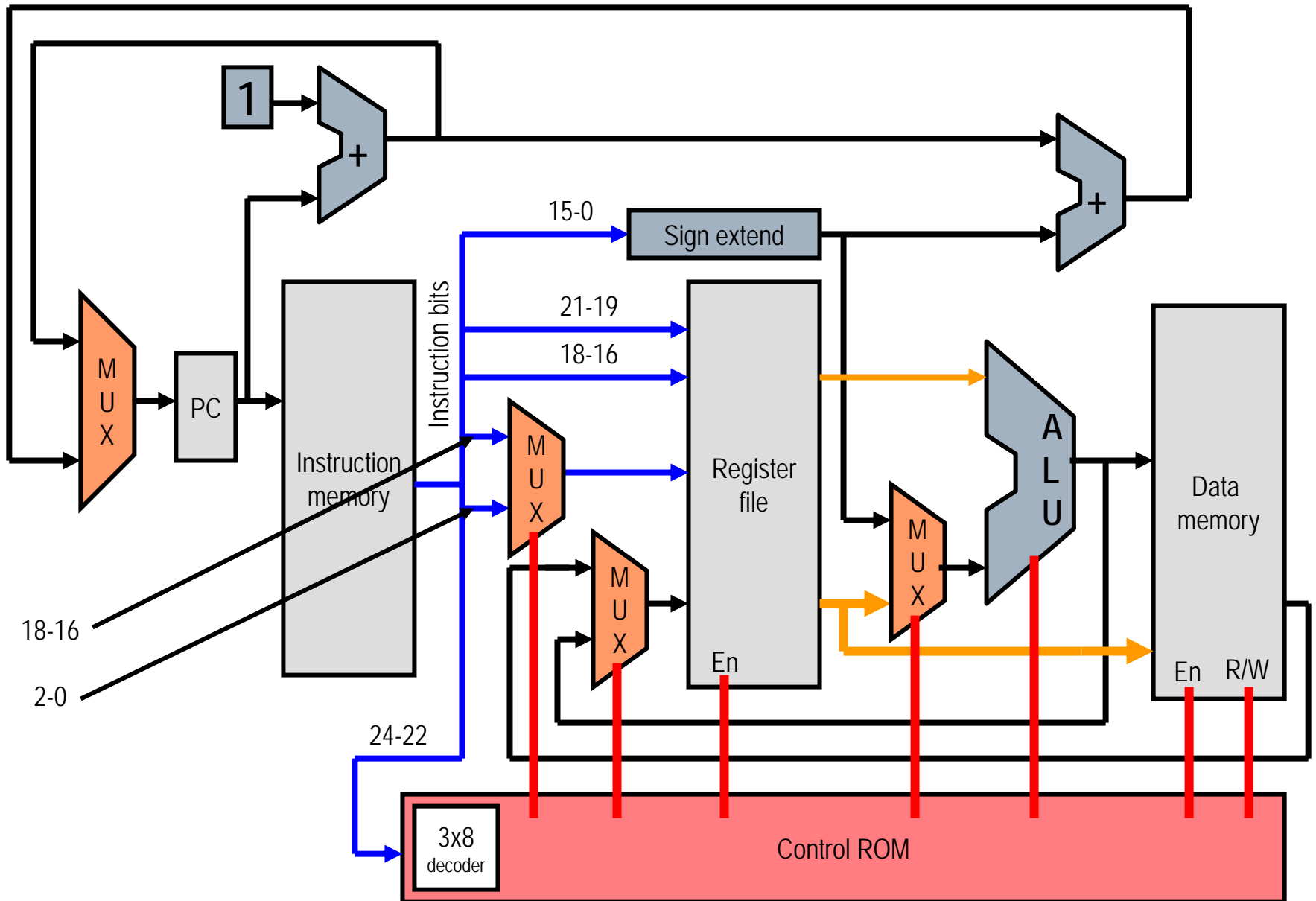
Announcements

- ❑ HW 4
 - Is available on the course website, due Feb 22 (Thurs)
- ❑ Exam 1
 - Not graded yet
 - Will be returned next class (Tues)
- ❑ Project 2 – Combinations program contest
 - Fewest number of cycles on a variety of inputs is winner
 - 1 winner per section – Fabulous prizes for winner
 - Think about how to create a fast implementation
 - Reducing assembly instructions, what about more exotic tricks?
 - Requirements
 - Recursive implementation must be maintained

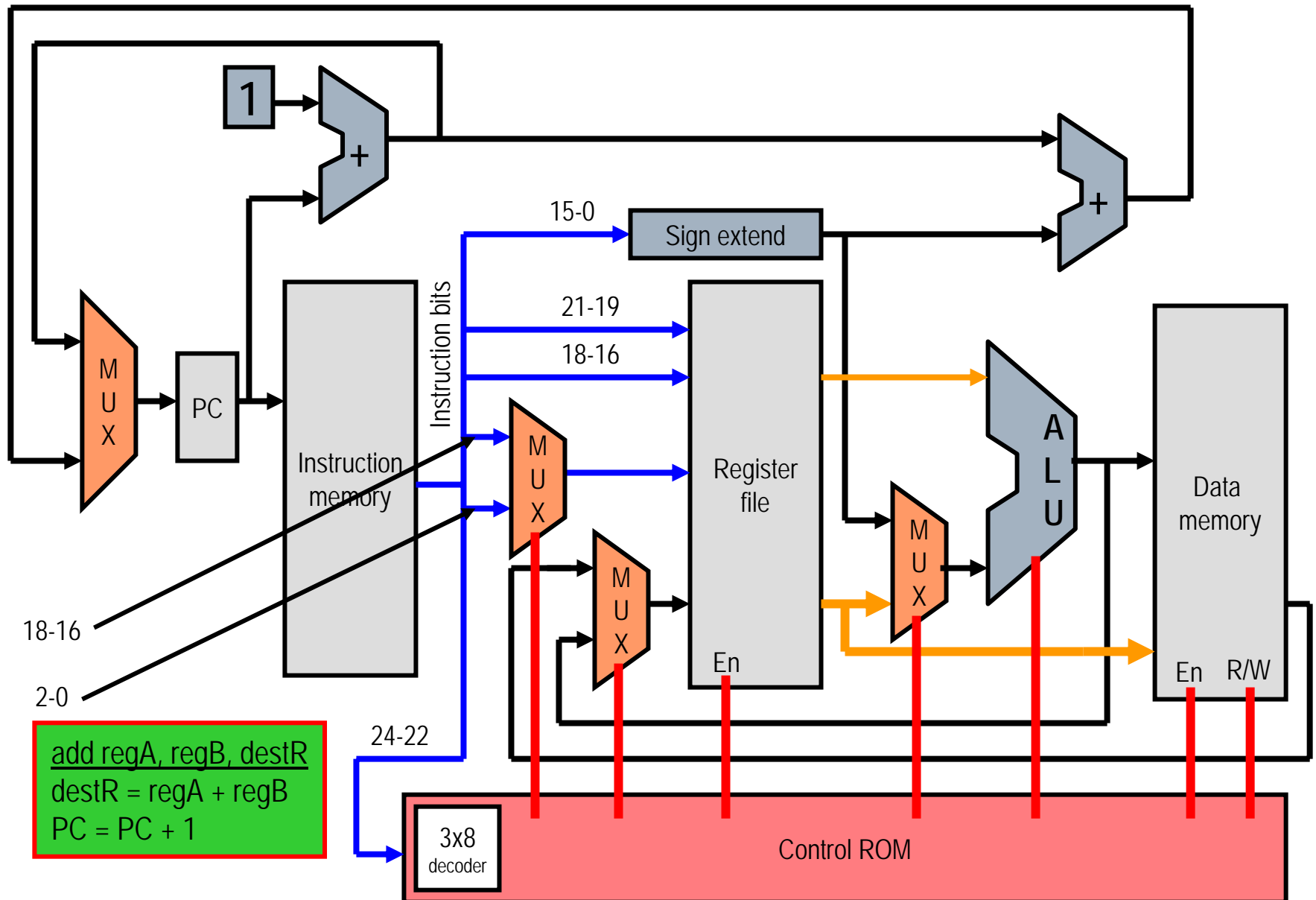
Previous Semester Evaluation Process

- ❑ Ran 20 C 10 and took the top 10 solutions from each section
- ❑ For the top 10
 - Ran 500 C 200
 - Winner is the best time on this input
 - Note, we ran a few other large inputs (300 C 100 and 800 C 300) and the winner was best for all runs
- ❑ This will be changed so you cannot hard code a solution!
 - But, it gives you a general idea

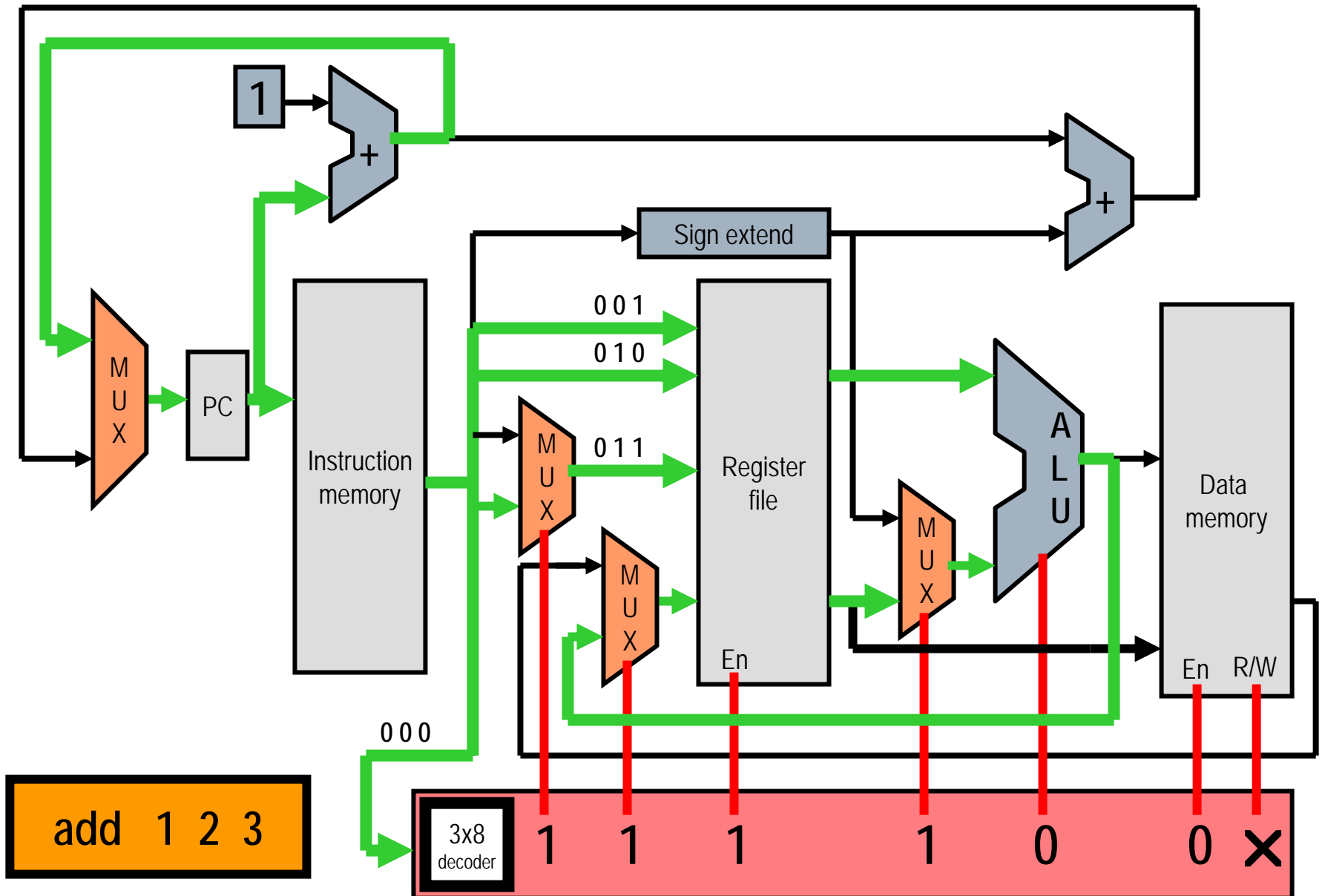
LC2Kx Datapath Implementation



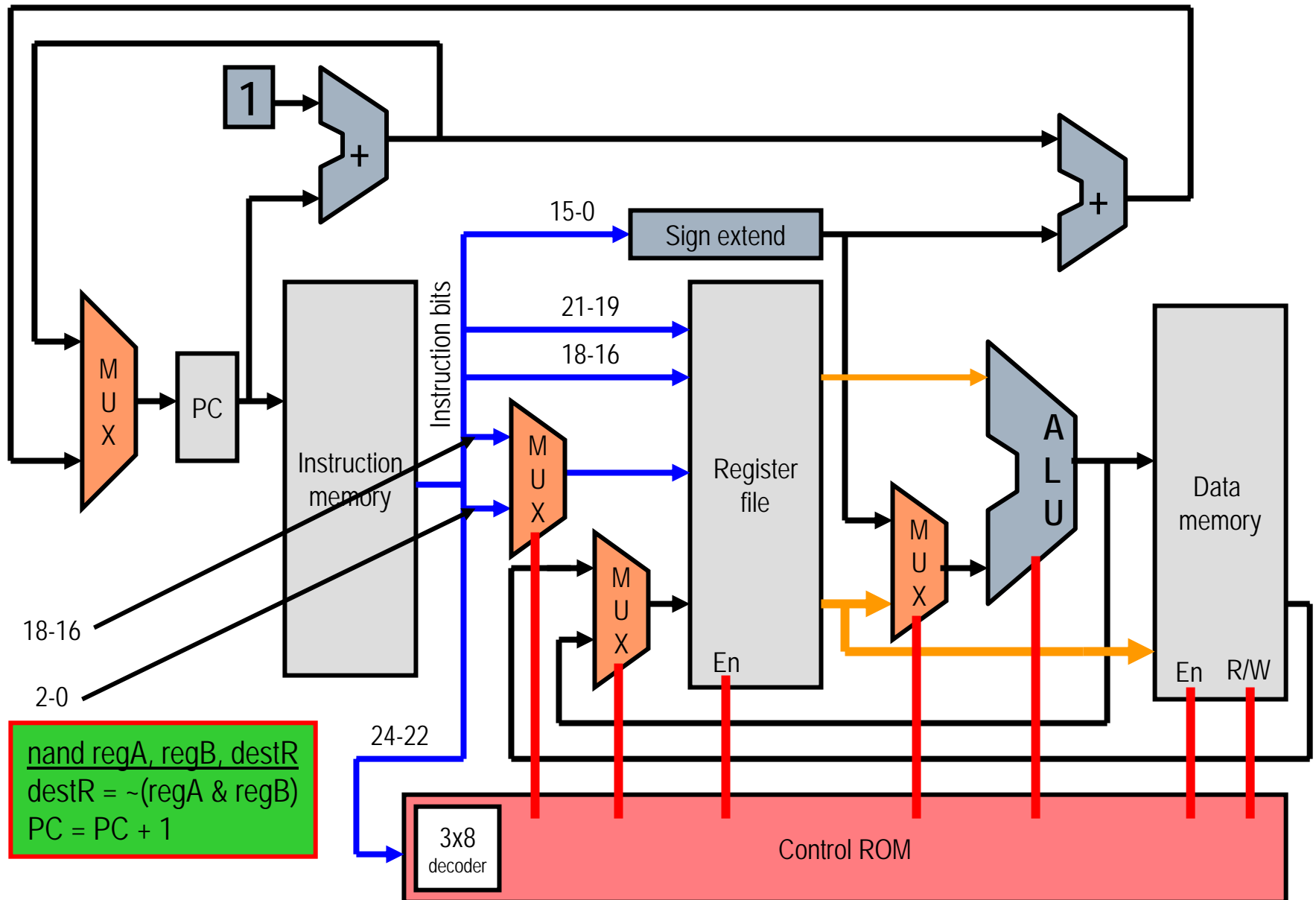
Executing an **ADD** Instruction



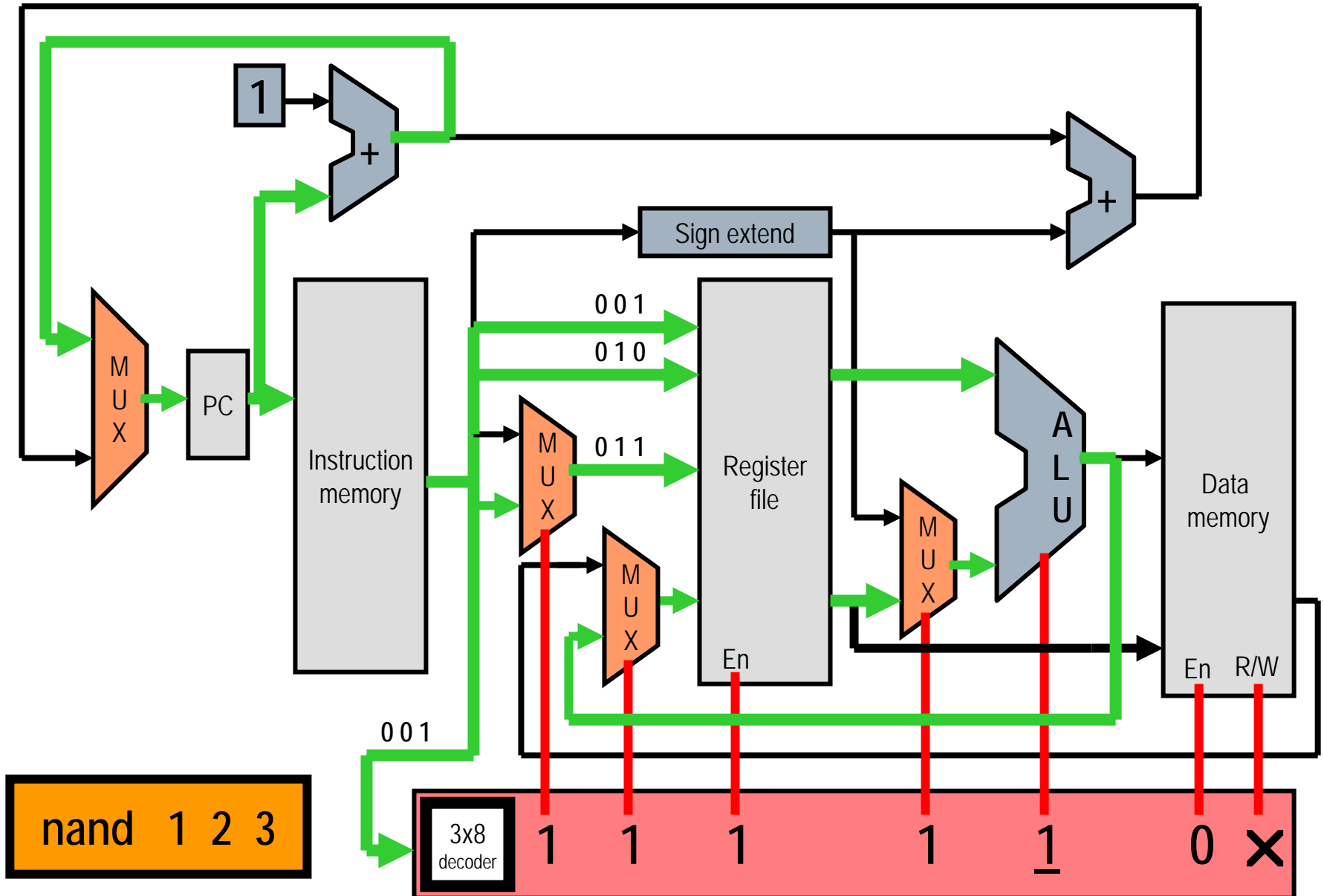
Executing an **ADD** Instruction on LC2Kx Datapath



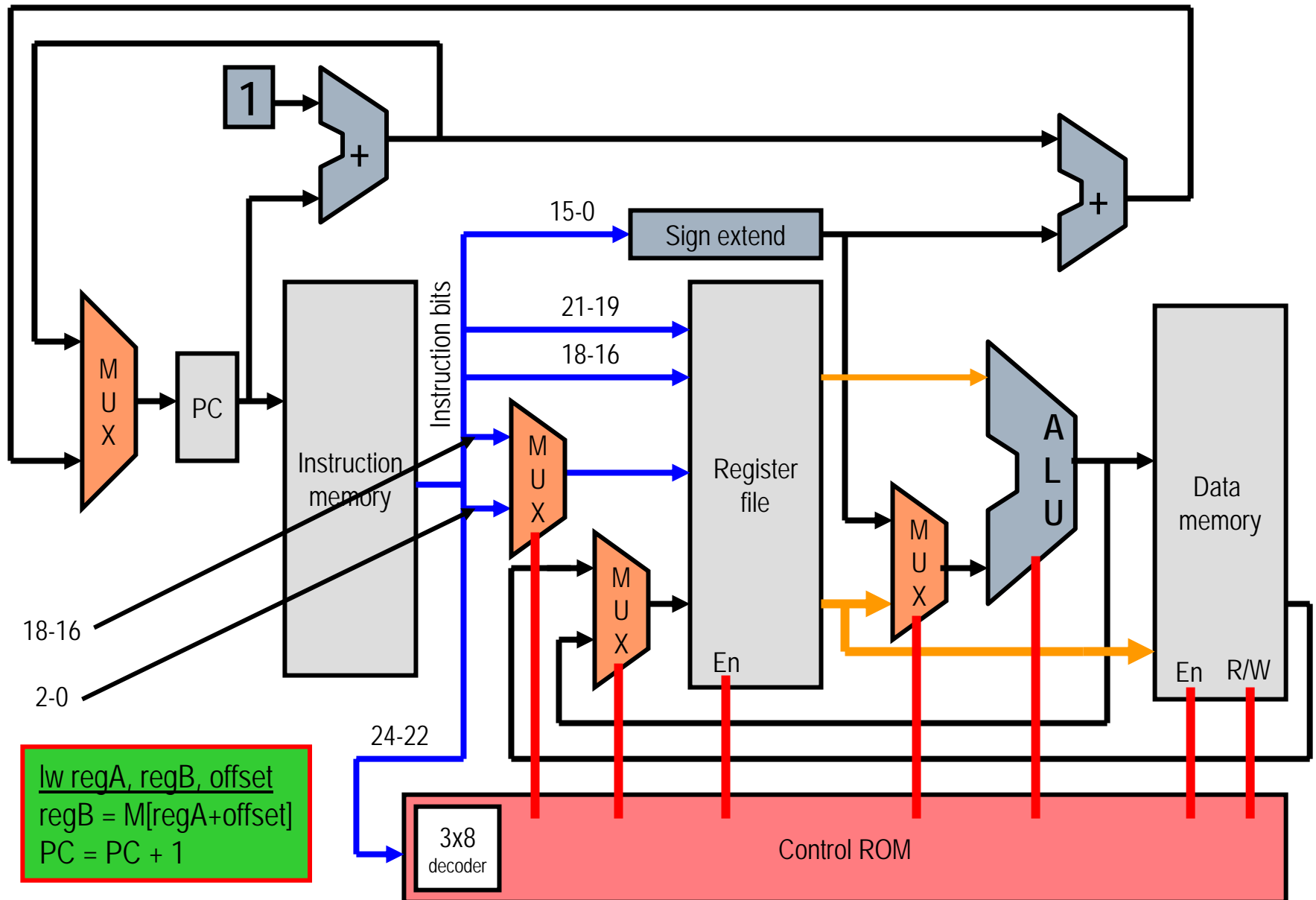
Executing a **NAND** Instruction



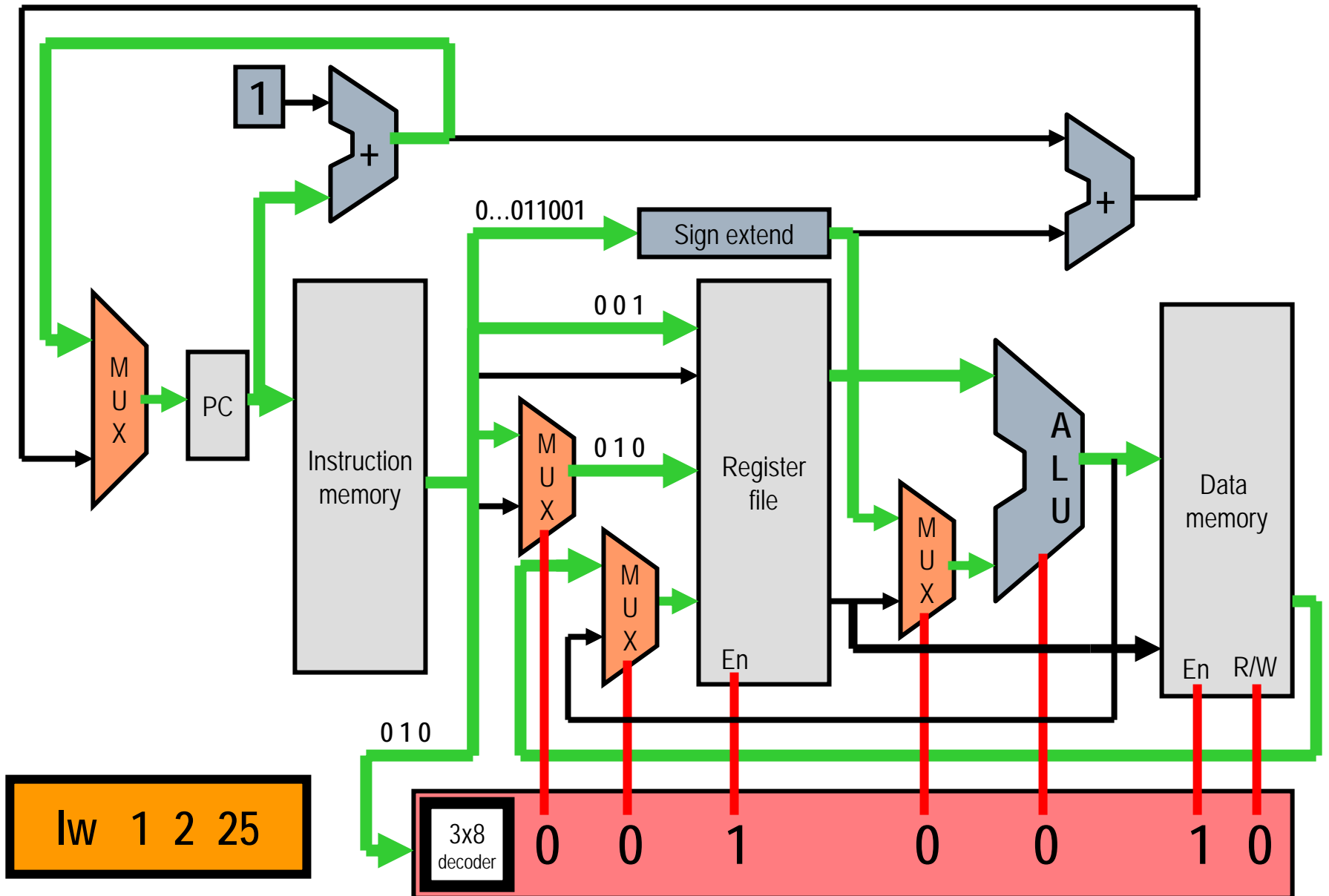
Executing **NAND** Instruction on LC2Kx Datapath



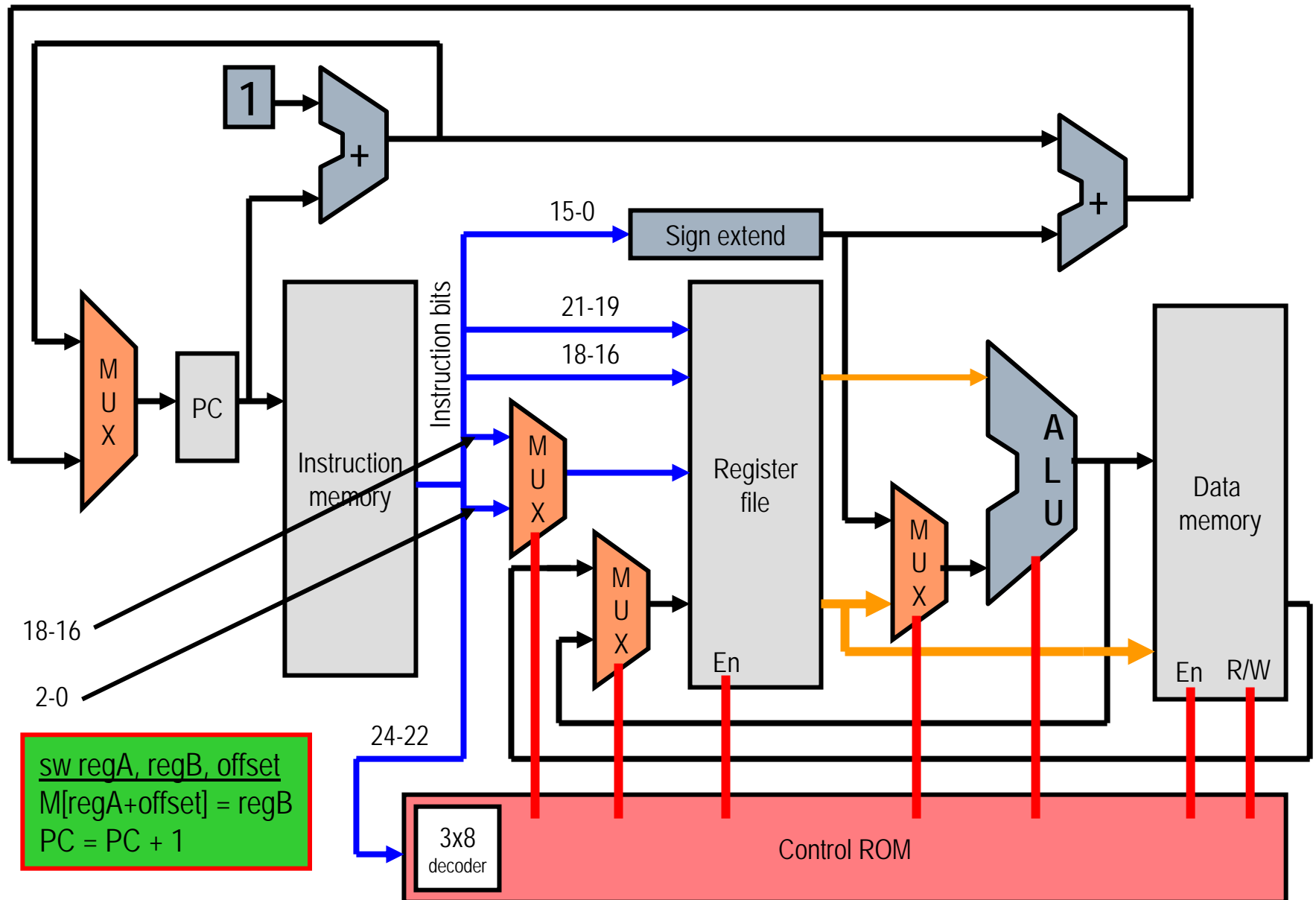
Executing a LW Instruction



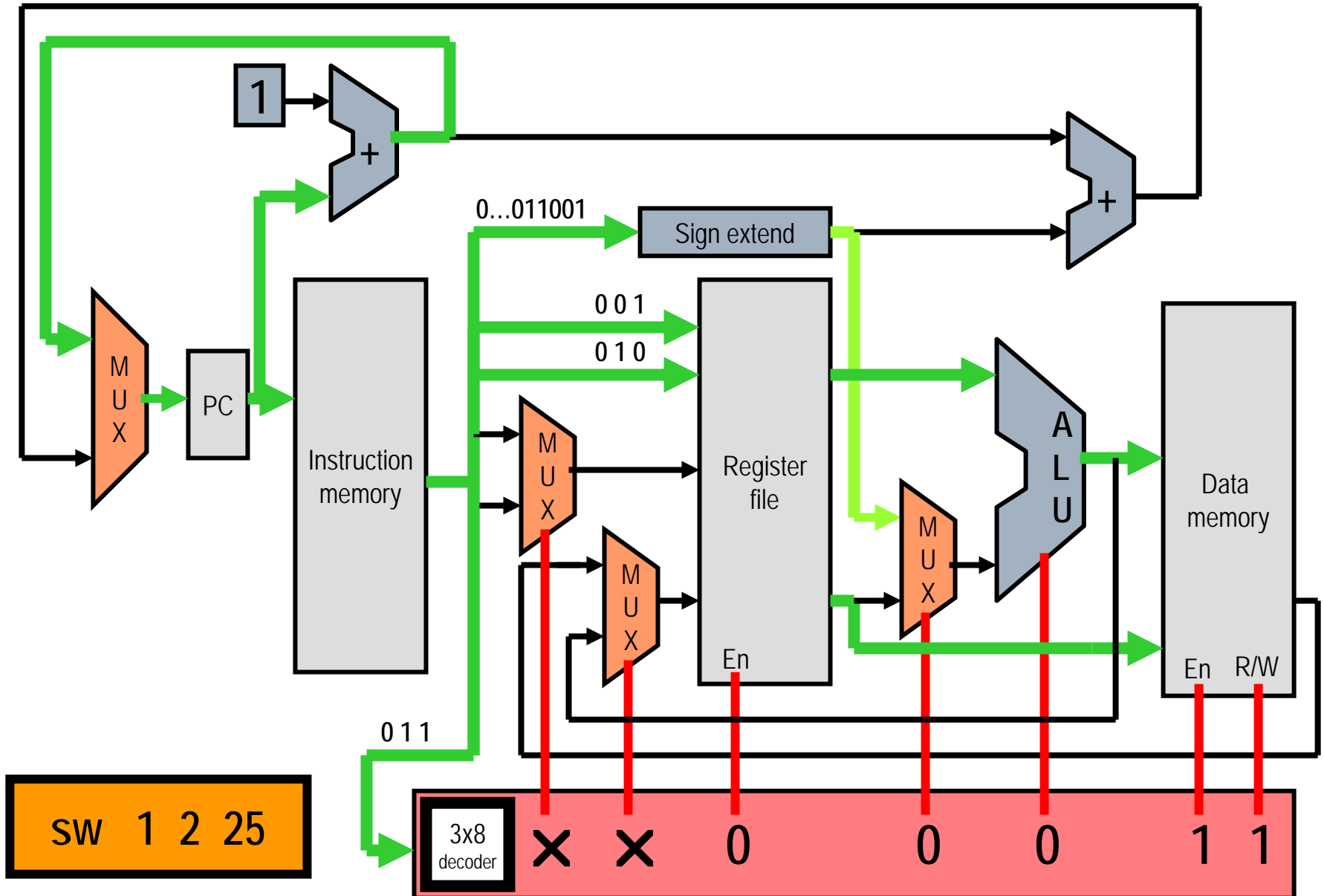
Executing a **LW** Instruction on LC2Kx Datapath



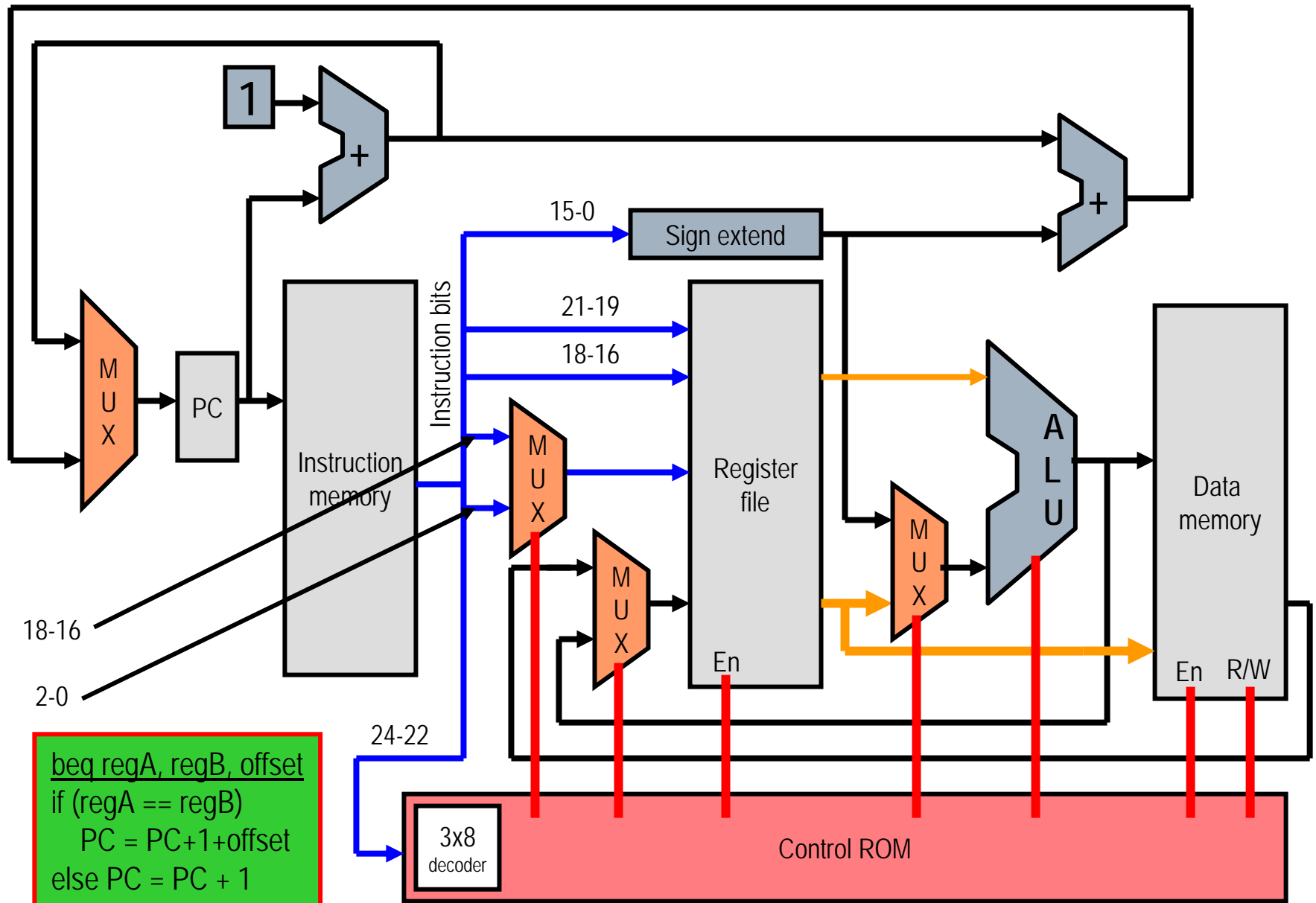
Executing a **SW** Instruction



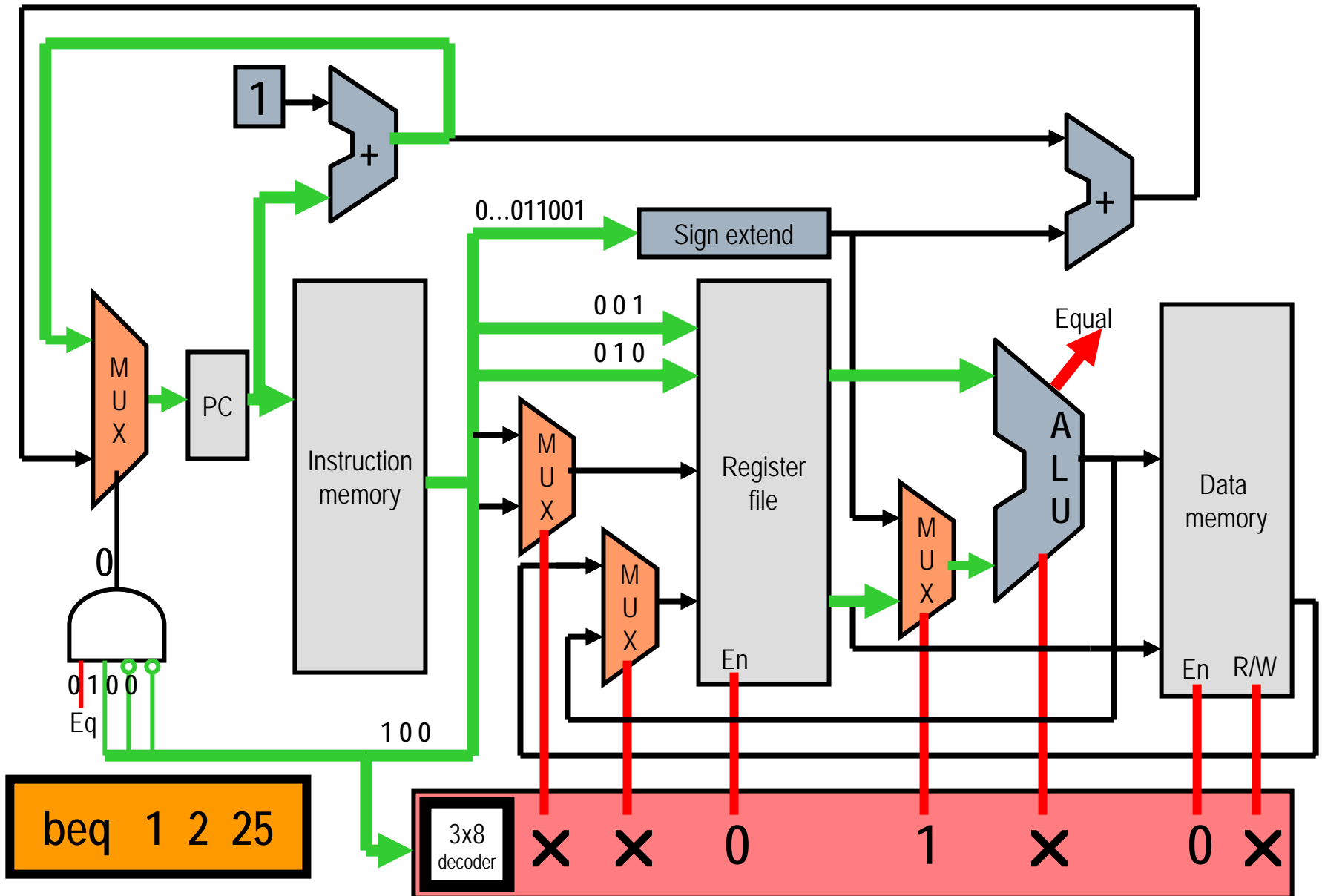
Executing a **SW** Instruction on LC2Kx Datapath



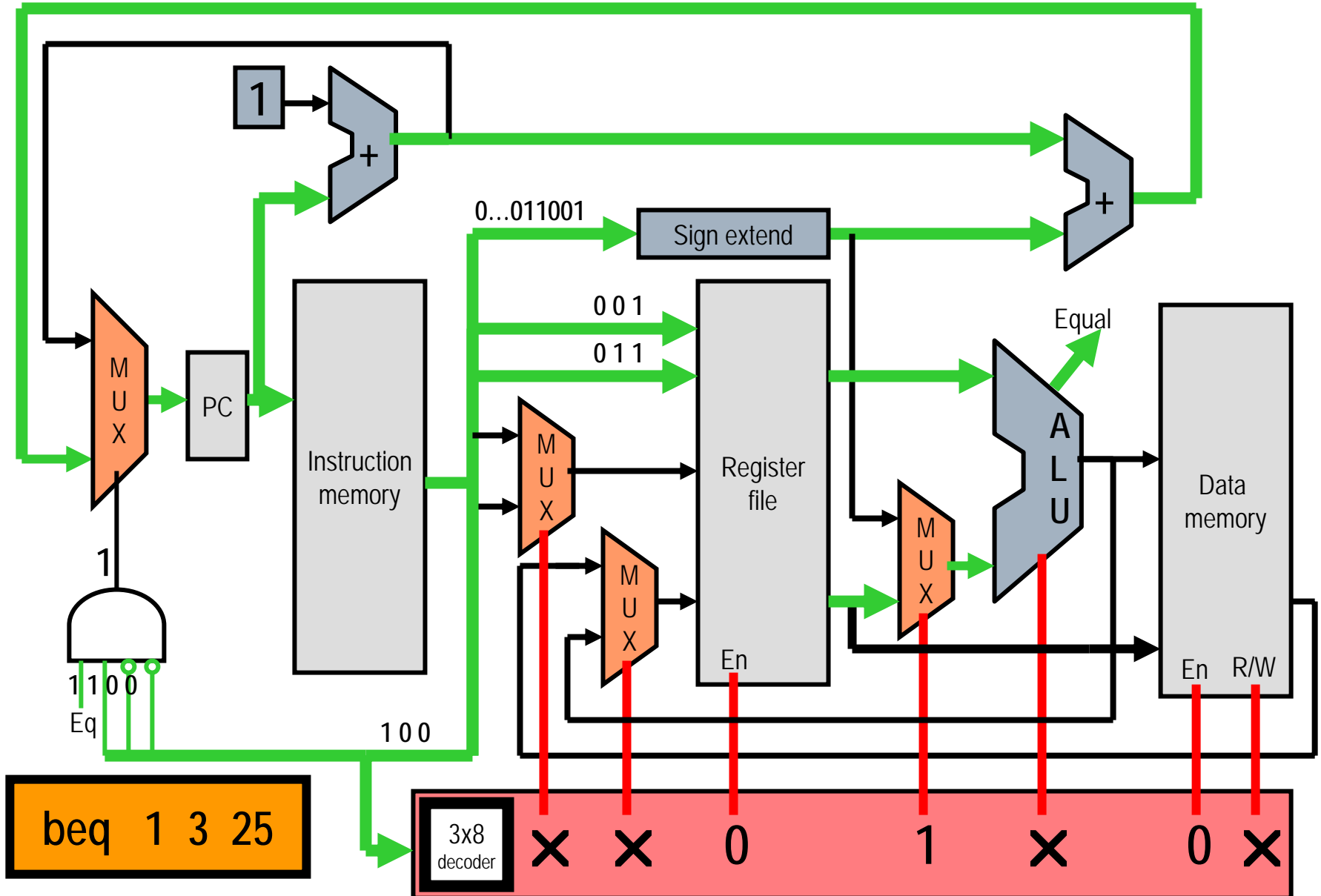
Executing a **BEQ** Instruction



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



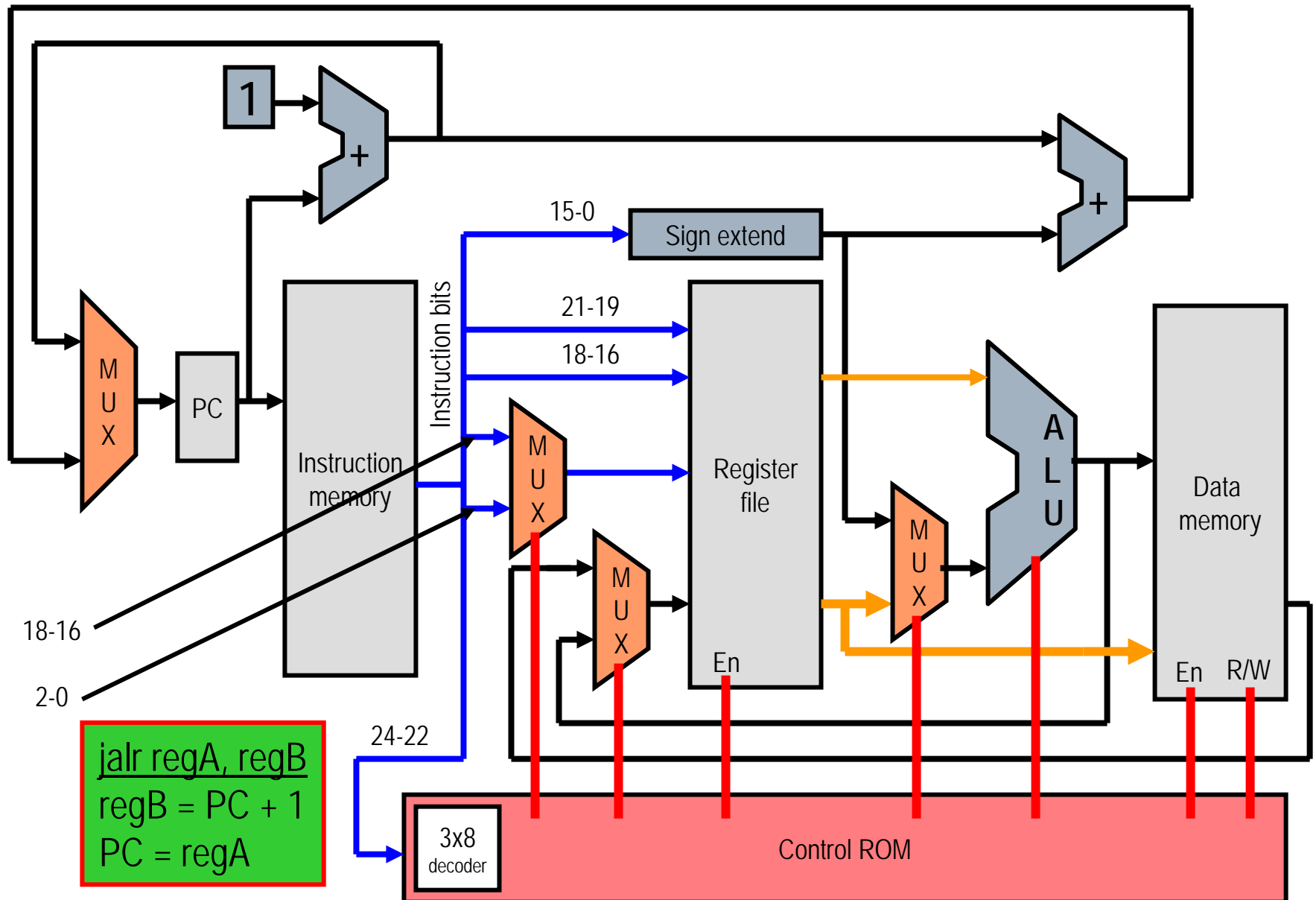
Executing a "taken" **BEQ** Instruction on LC2K Datapath



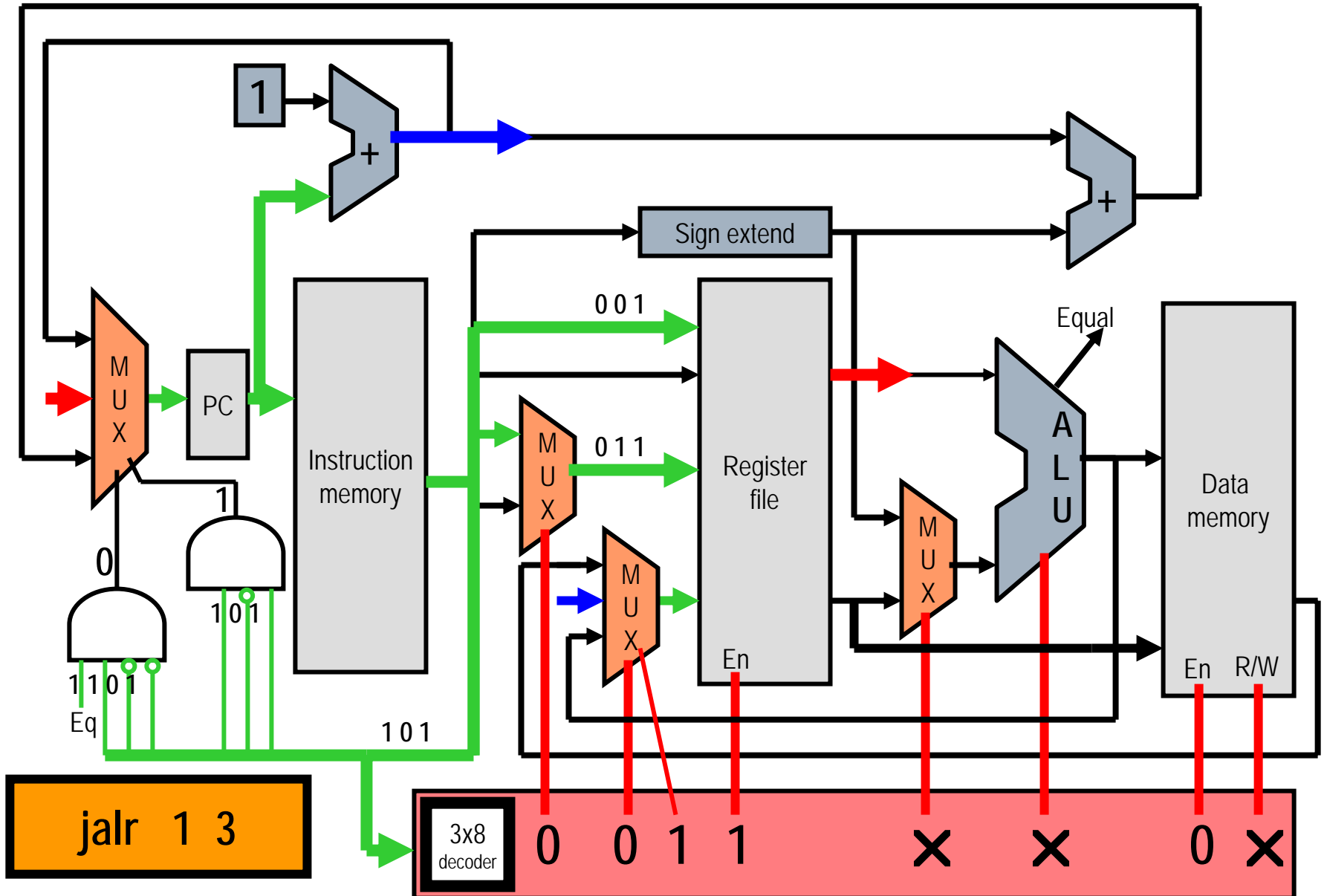
So Far, So Good

- ❑ 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

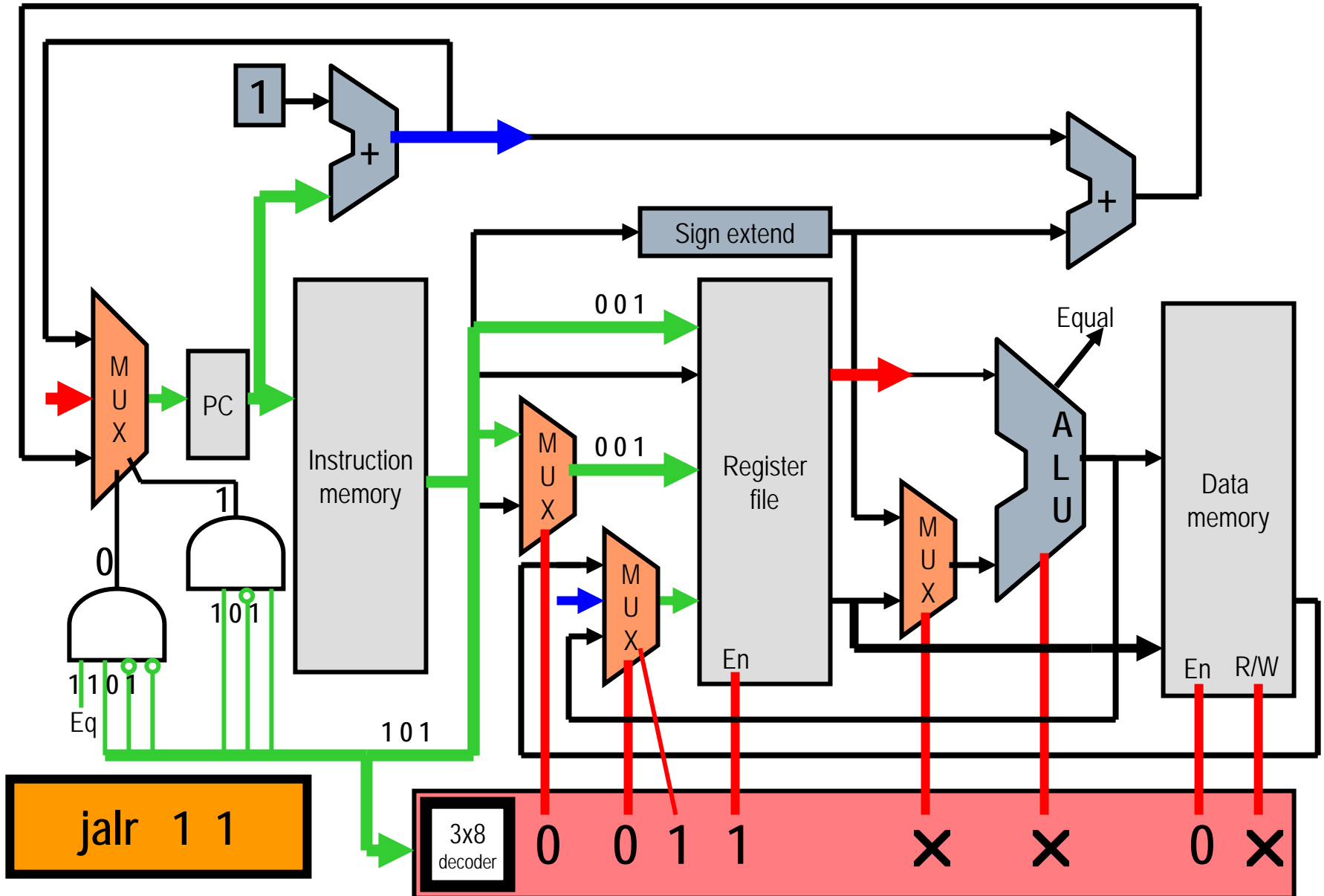
Executing a **JALR** Instruction



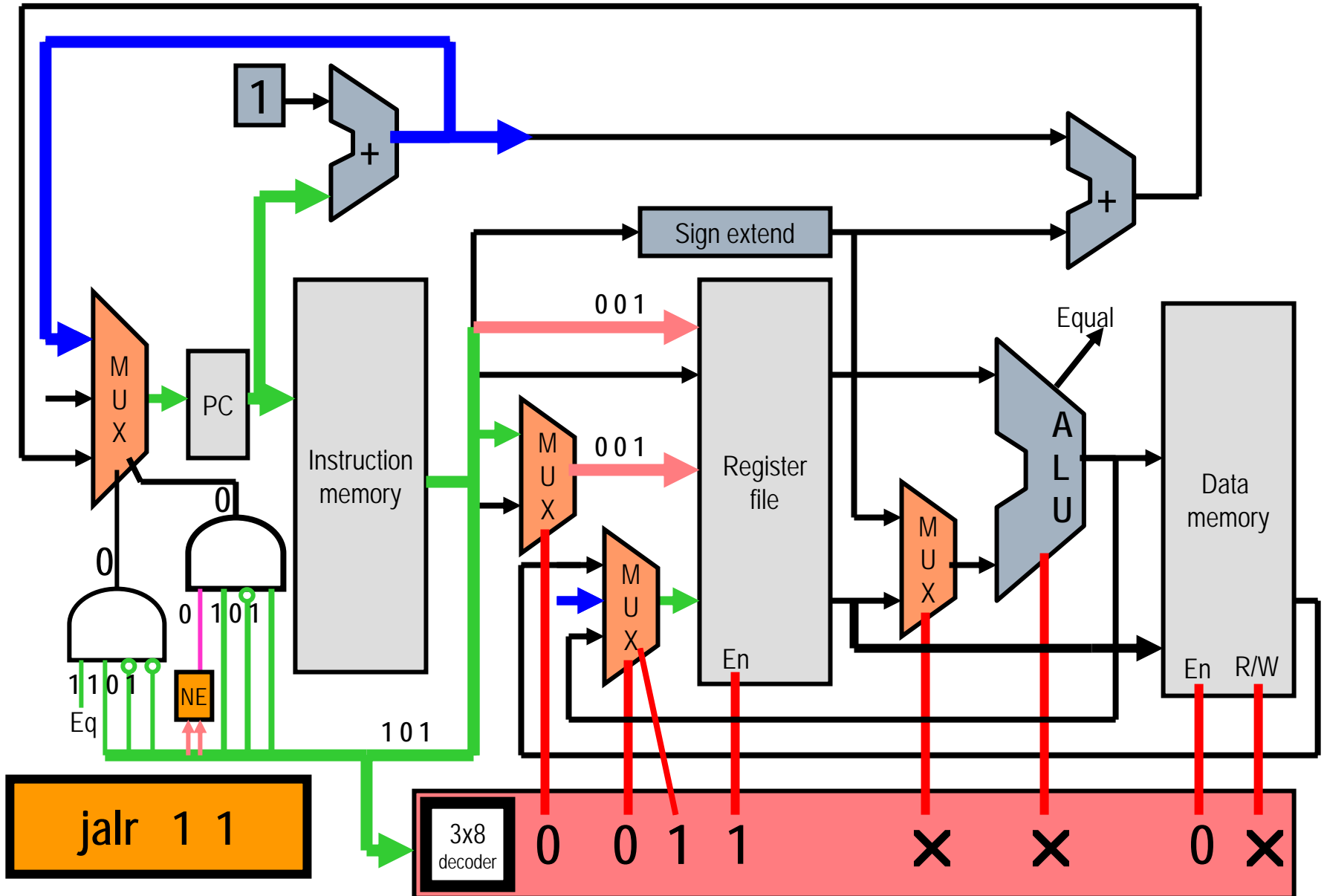
Executing a **JALR** Instruction on LC2Kx Datapath



What If $\text{regA} = \text{regB}$ for a **JALR** ?



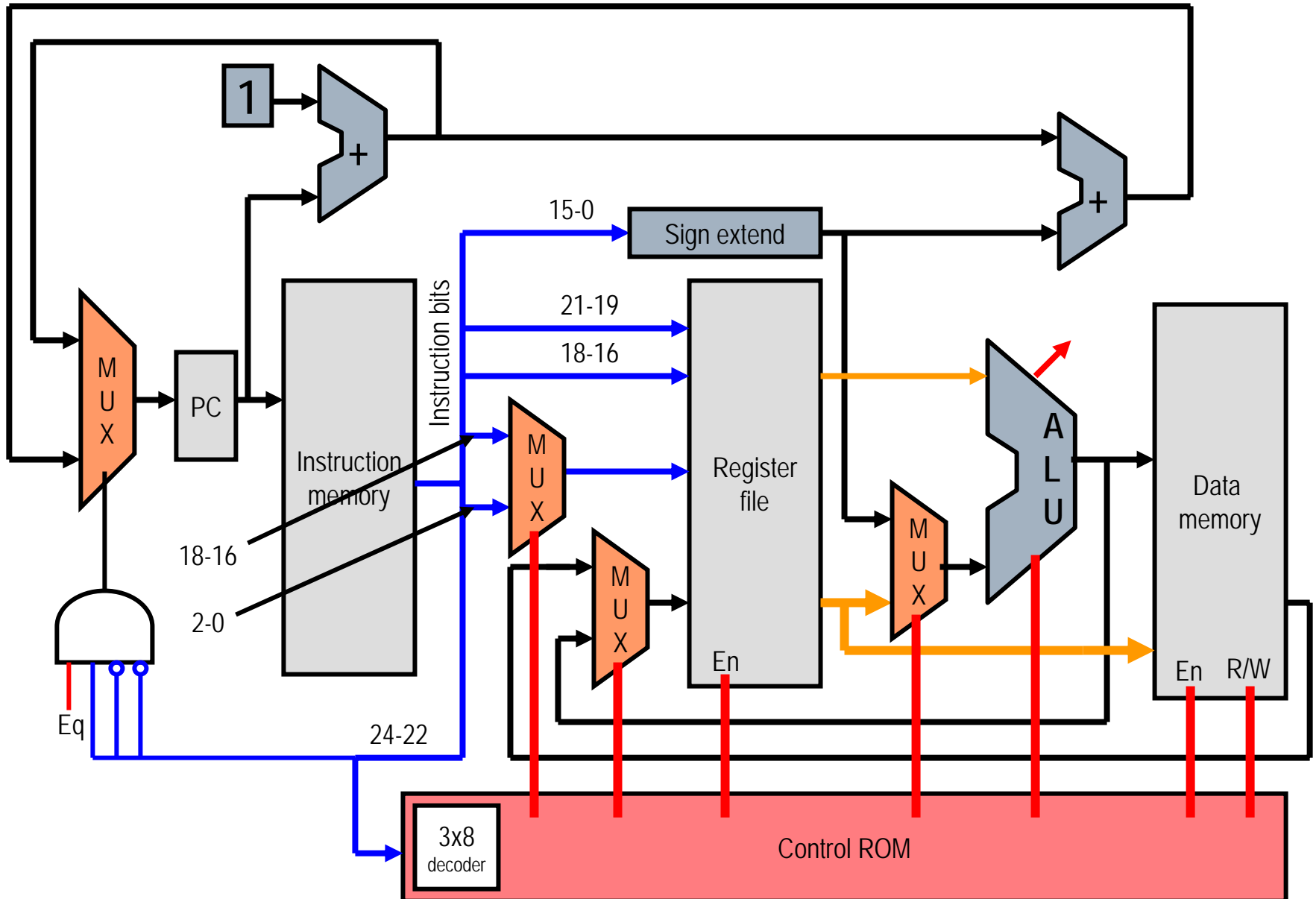
Changes for a **JALR 1 1** Instruction



Class Problem

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

Class Problem (continued)



What's Wrong with Single Cycle?

- ❑ **All instructions run at the speed of the slowest instruction.**
- ❑ Adding a long instruction can hurt performance
 - What if you wanted to include multiply?
- ❑ You cannot reuse any parts of the processor
 - We have 3 different adders to calculate $PC+1$, $PC+1+offset$ and the ALU
- ❑ No benefit in making the common case fast
 - Since every instruction runs at the slowest instruction speed
 - This is particularly important for loads as we will see later

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:

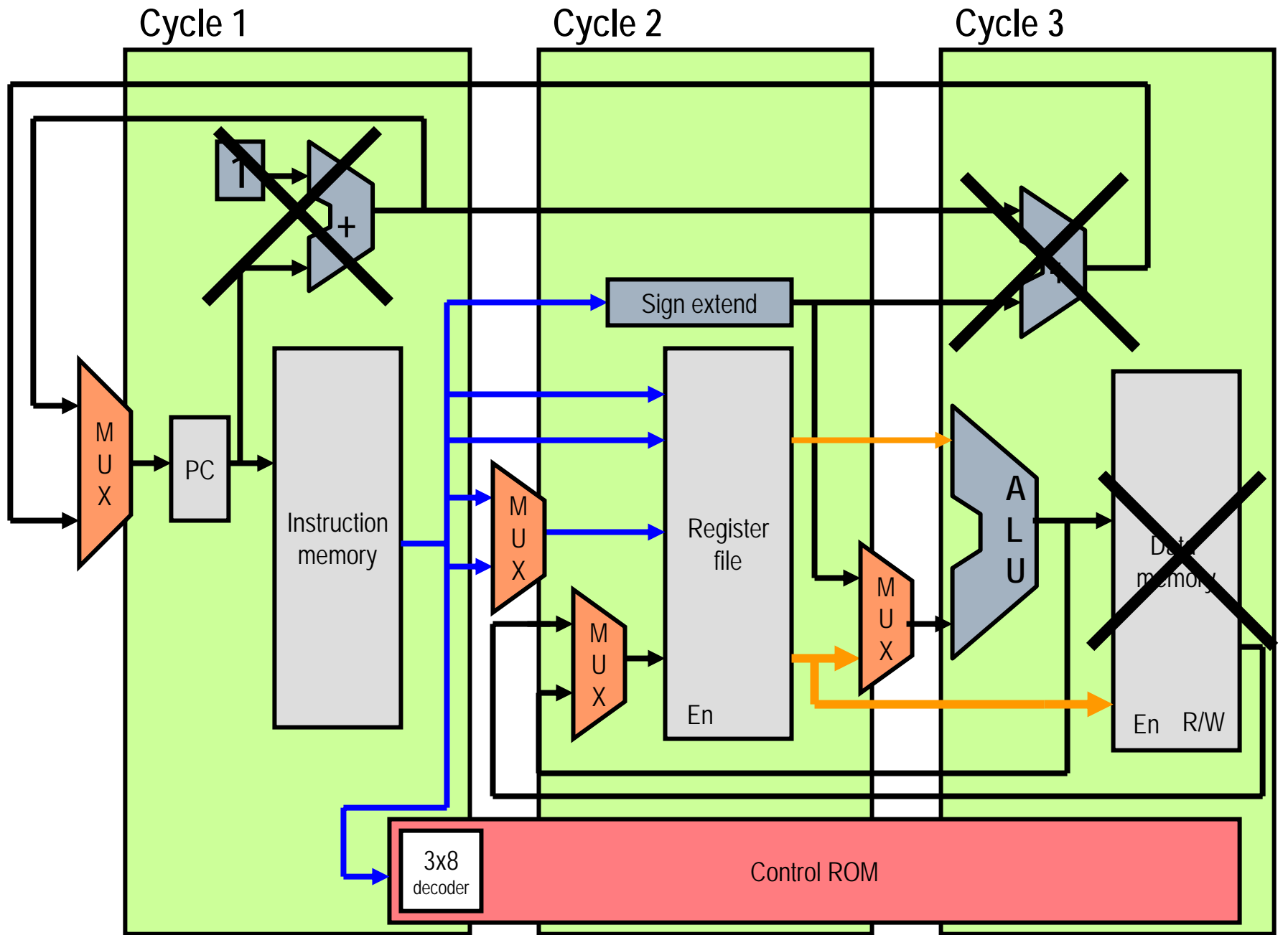
= ??

□ Optimal execution:

= ??

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

