10. Basic processor design – Single Cycle Datapaths

EECS 370 – Introduction to Computer Organization - Winter 2016

Profs. Valeria Bertacco & Reetu Das

EECS Department
University of Michigan in Ann Arbor, USA

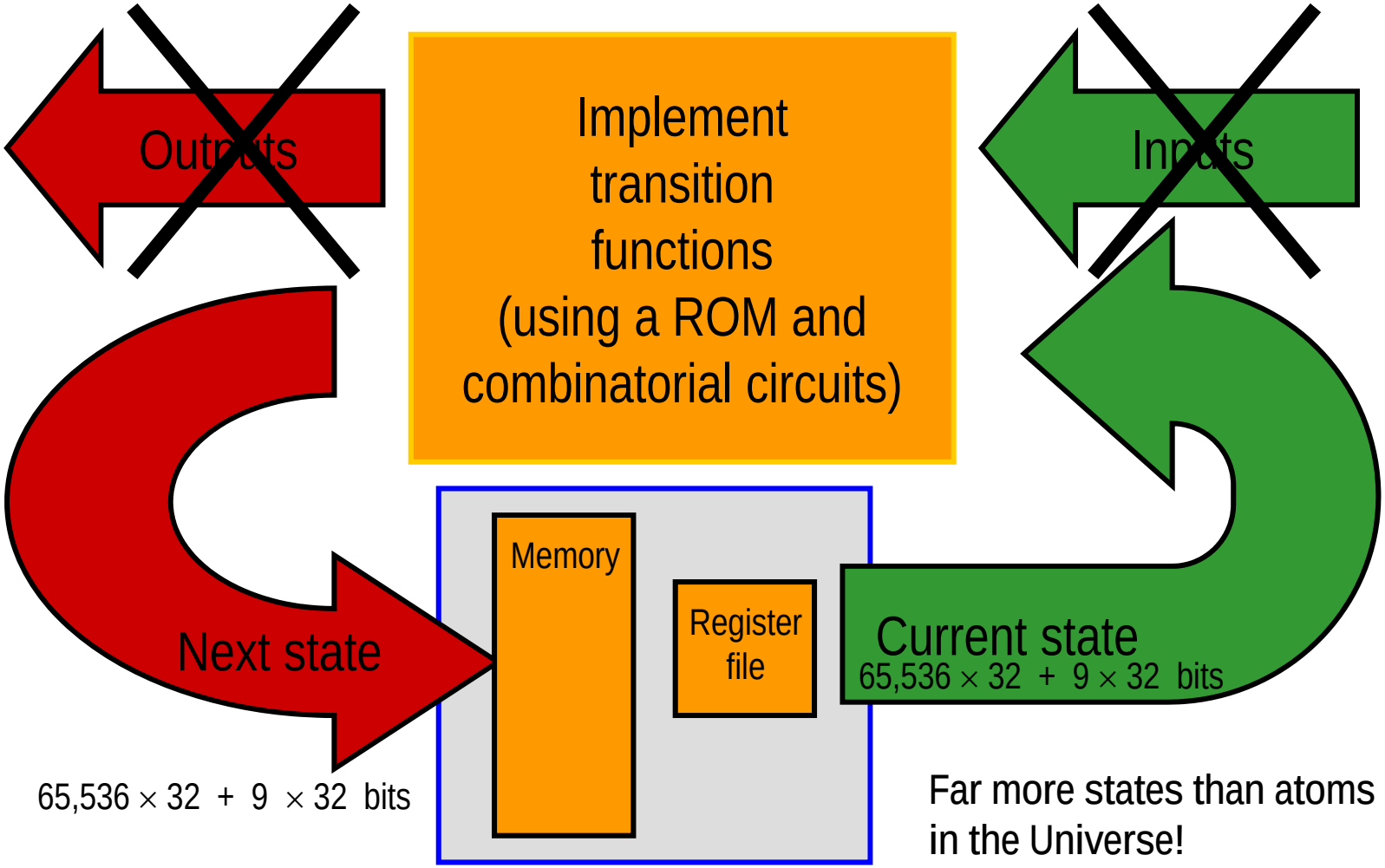
Announcements

- Get busy on Project 2!!!
  - Due 02/26
- Competition due March 7
- Homework 3 due today

Recap

- Combinational logic
  - Silicon, CMOS
  - Basic logic gates: AND, OR, NOT, NOR, XOR
  - Mux, Decoder, Half-adder, Full-Adder
  - Ripple-Carry and Carry Look-Ahead Adders
- Sequential logic
  - SR latch, D latch, edge-triggered flip-flop
  - Finite state machines: diagrams and math representation
- Timing diagrams

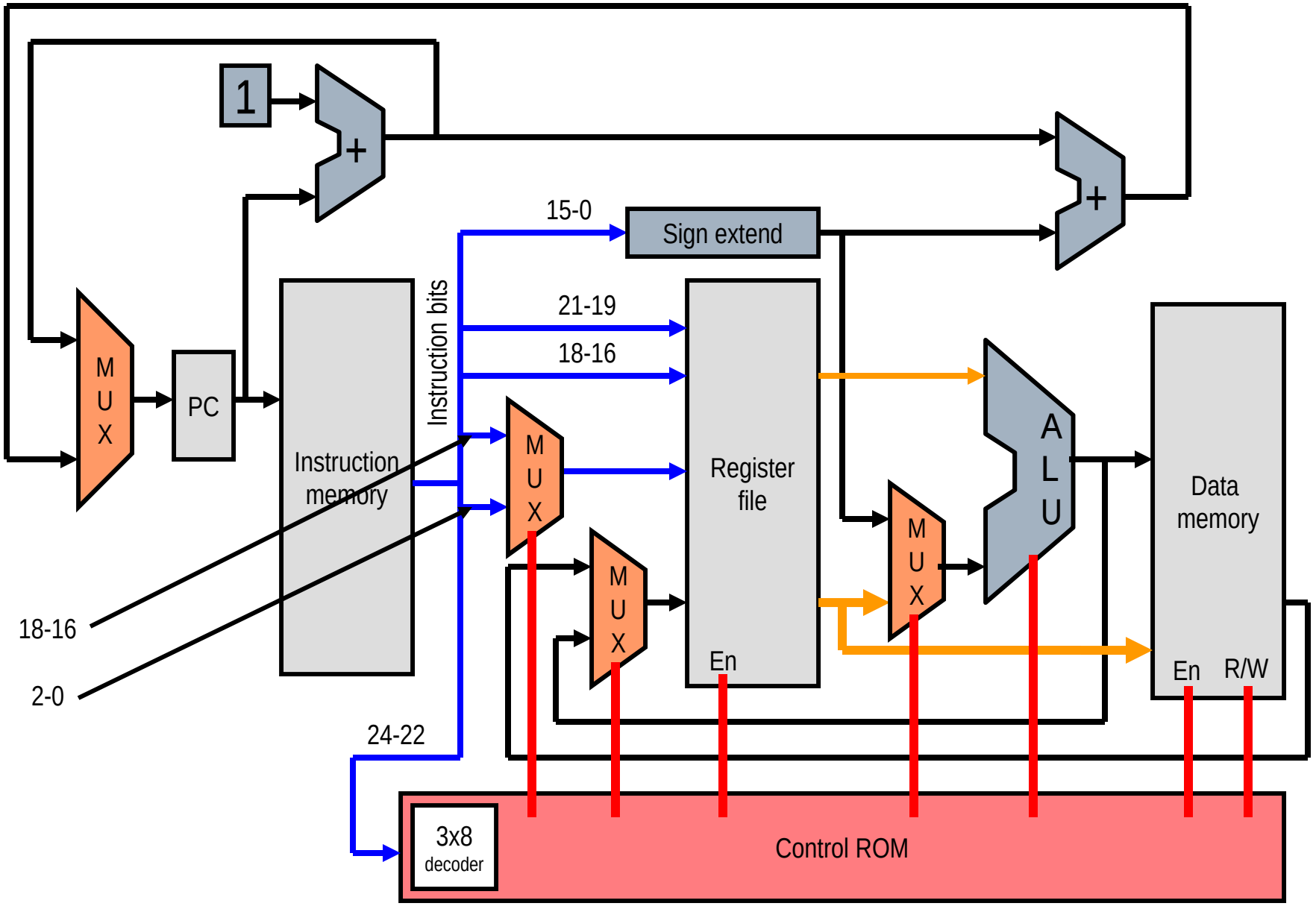
LC2K Processor as FSM



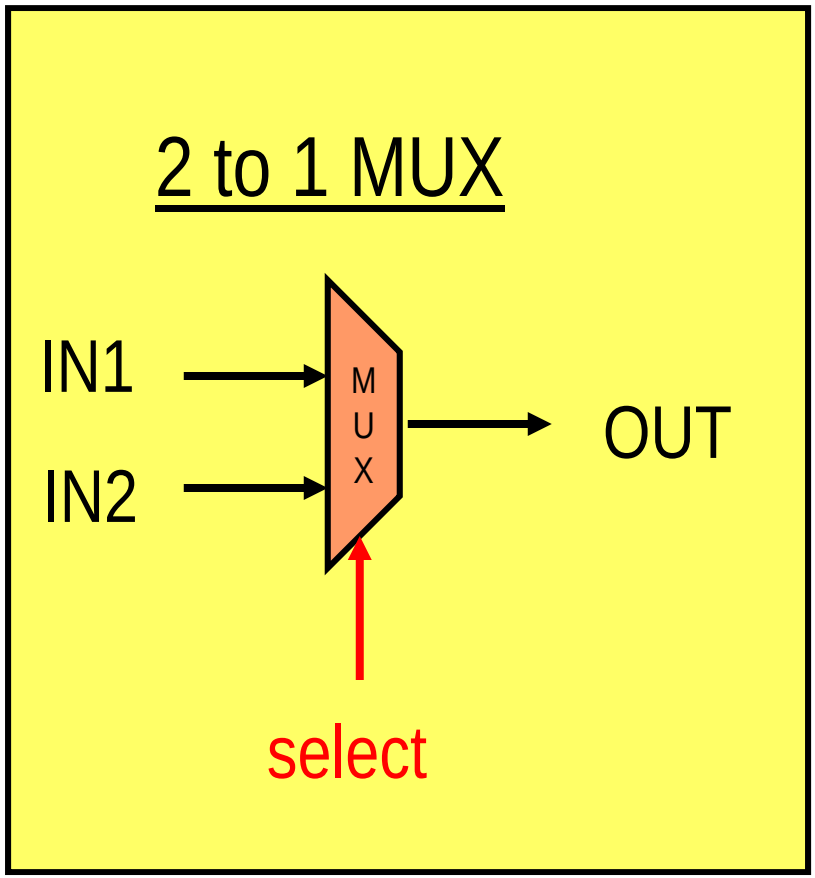
Single-Cycle Processor Design

- General-Purpose Processor Design
  - Fetch Instructions
  - Decode Instructions
    - Instructions are input to control ROM
  - ROM data controls movement of data
    - Incrementing PC, reading registers, ALU control
  - Clock drives it all
  - Single-cycle datapath: Each instruction completes in one clock cycle

LC2K Datapath Implementation



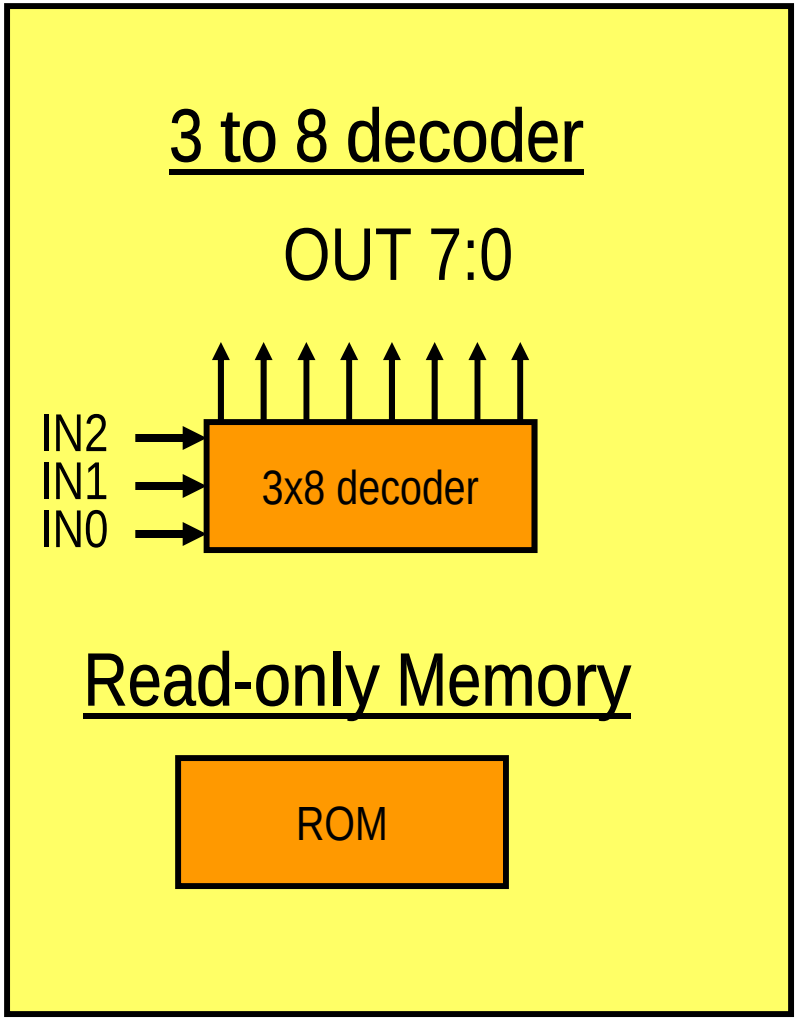
Control Building Blocks (1)



Connect one of the inputs to OUT based on the value of select

```
If (select == 0)
    OUT = IN1
Else
    OUT = IN2
```

Control Building Blocks (2)

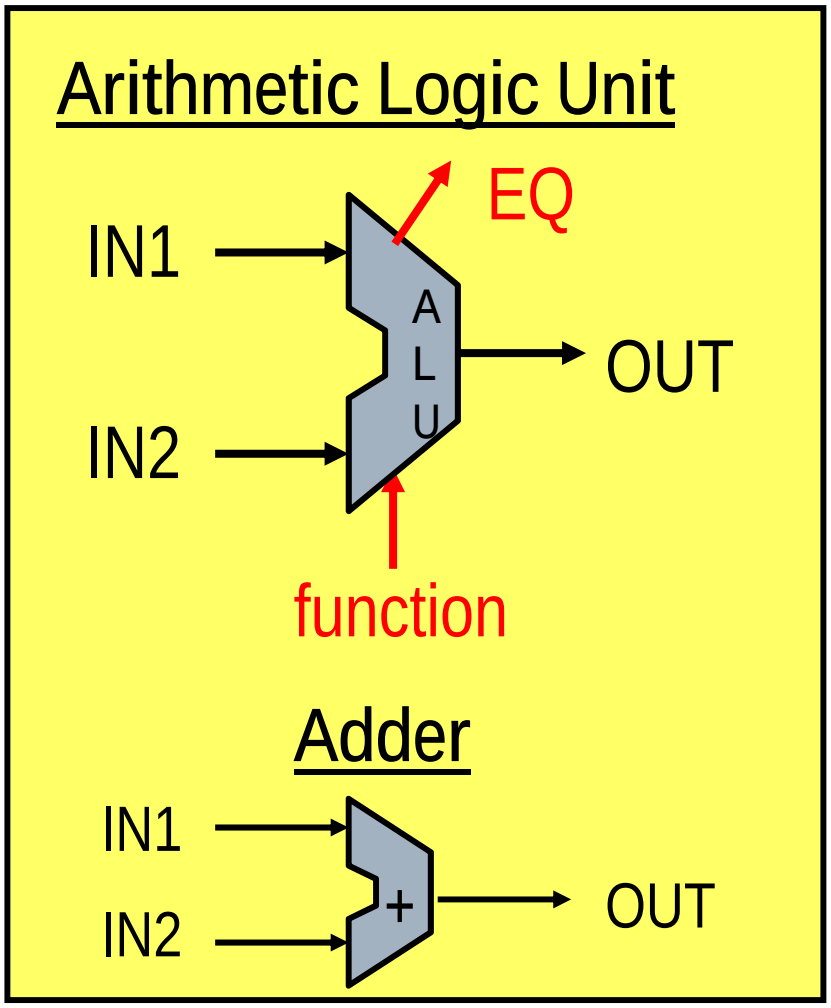


Decoder activates one of the output lines based on the input

IN	OUT
210	76543210
000	00000001
001	00000010
010	00000100
011	00001000
etc.	

ROM just stores preset data in each location.
Give address to access data.

Compute Building Blocks (1)



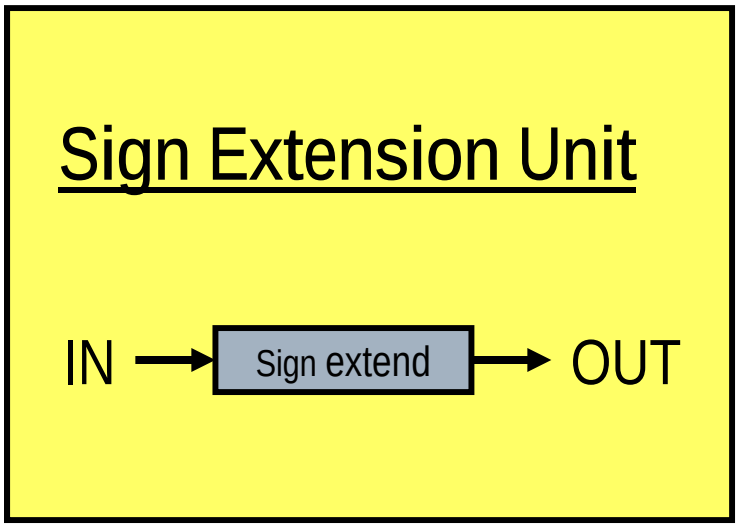
Perform basic arithmetic functions

OUT = f(IN1, IN2)
EQ = (IN1 == IN2)

For LC2k, f = add, nor.
For other processors, there are many more functions.

Simple ALU – Does only adds

Compute Building Blocks (2)



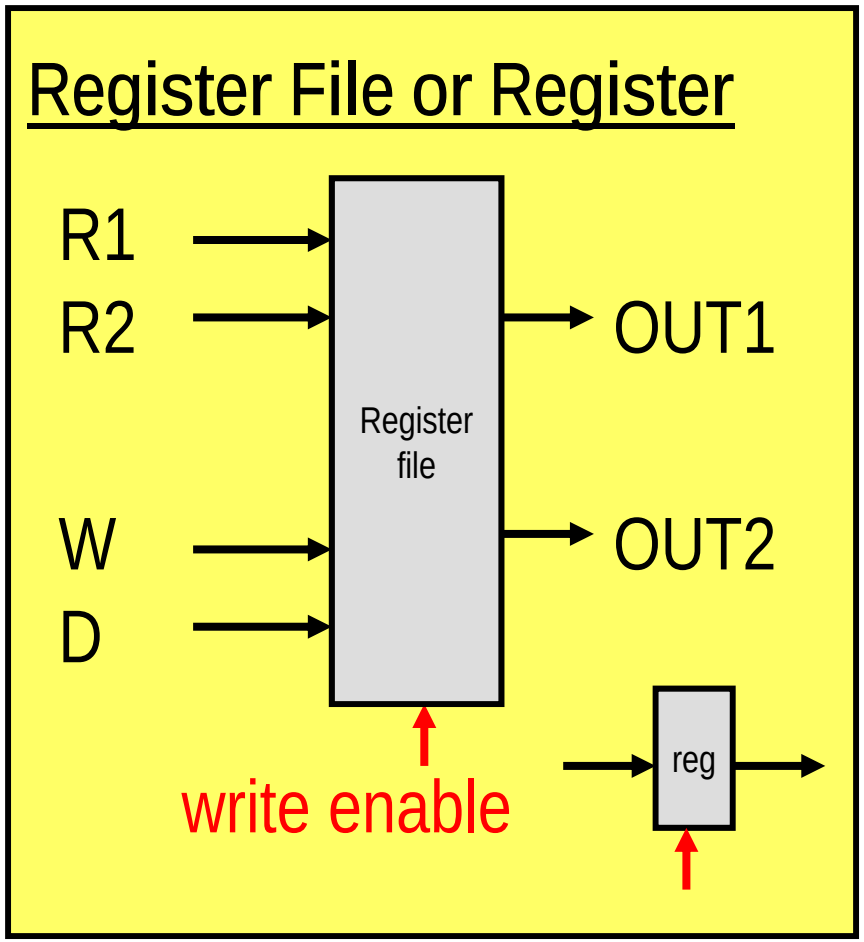
Sign extend input by replicating the MSB to width of output

OUT(31:0) = SE(IN(15:0))

OUT(31:16) = IN(15)
OUT(15:0) = IN(15:0)

Useful when compute unit is wider than data

State Building Blocks (1)



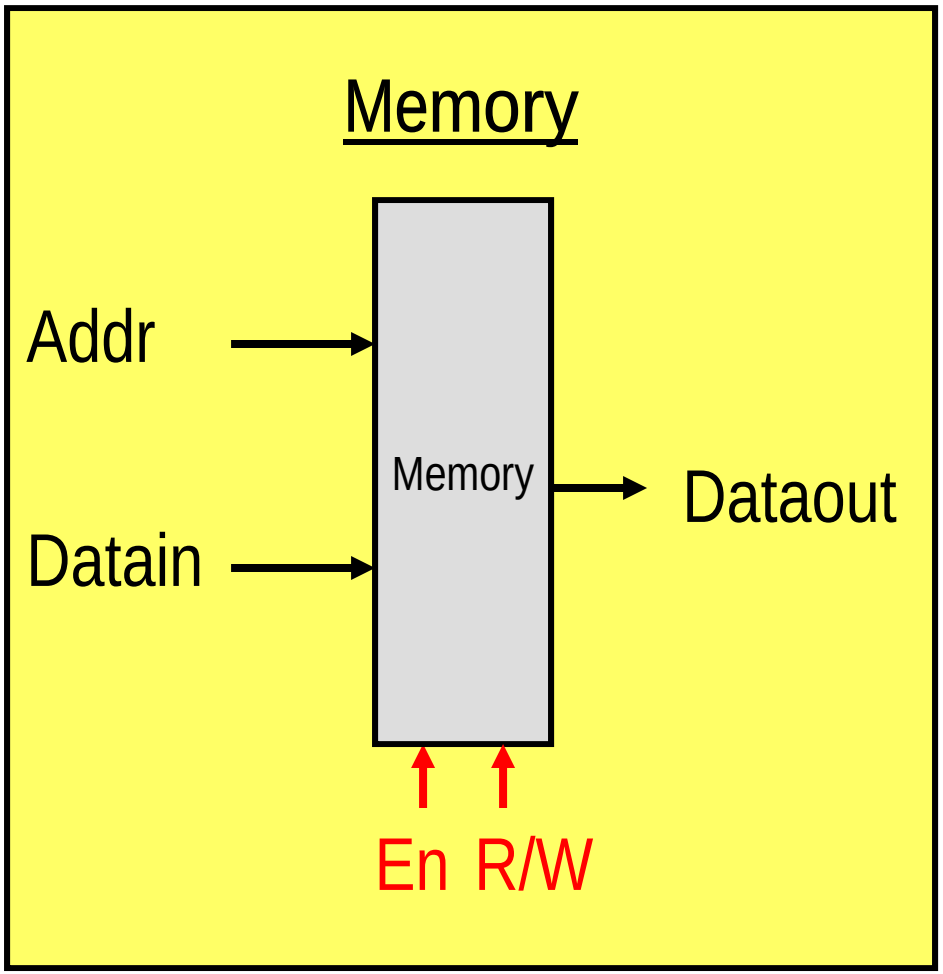
Small/fast memory to store temporary values

- n entries (LC2 = 8)
- r read ports (LC2 = 2)
- w write ports (LC2 = 1)

* Ri specifies register number to read
* W specifies register number to write
* D specifies data to write

How many bits are Ri and W in LC2?

State Building Blocks (2)

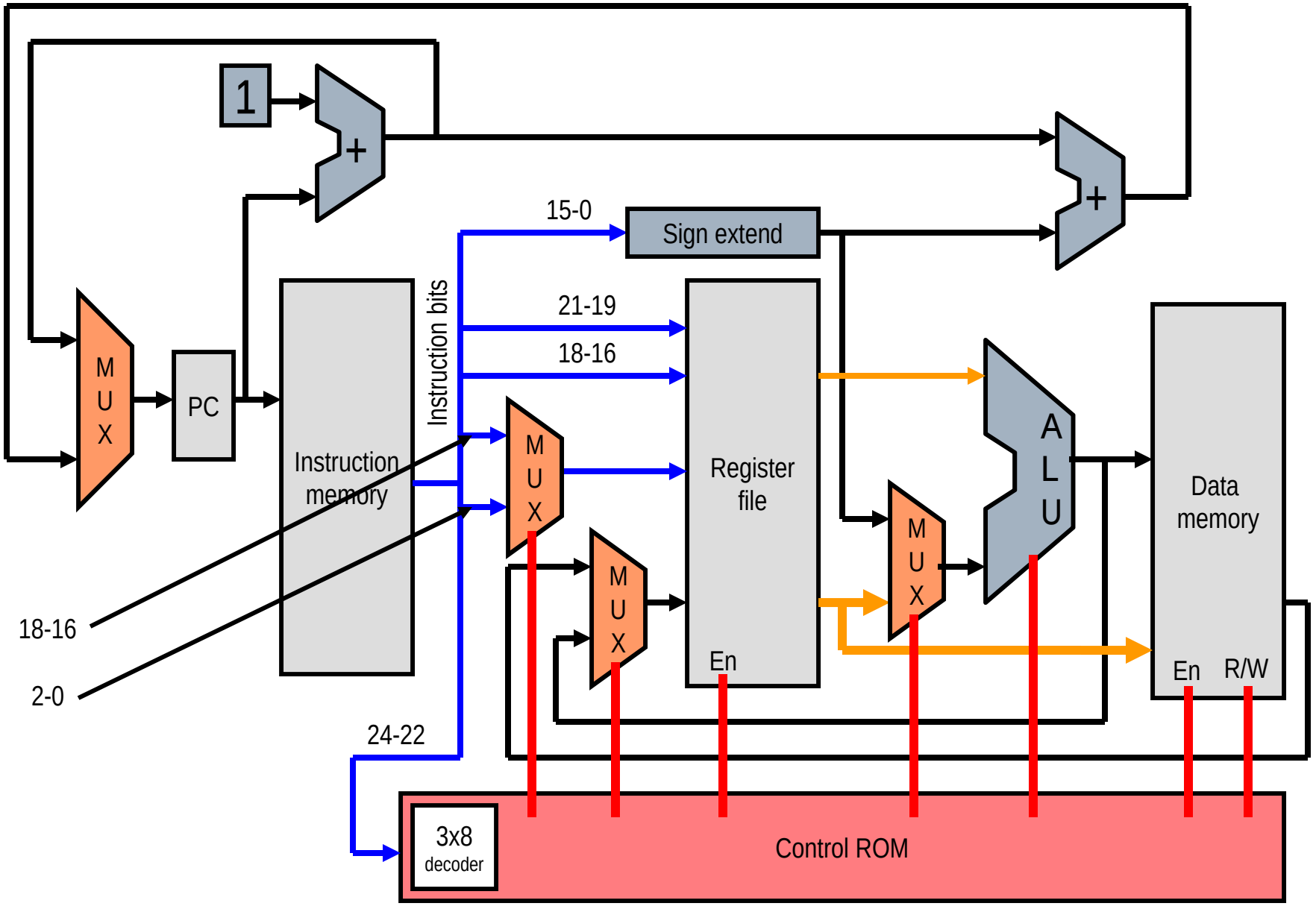


Slower storage structure to hold large amounts of stuff.

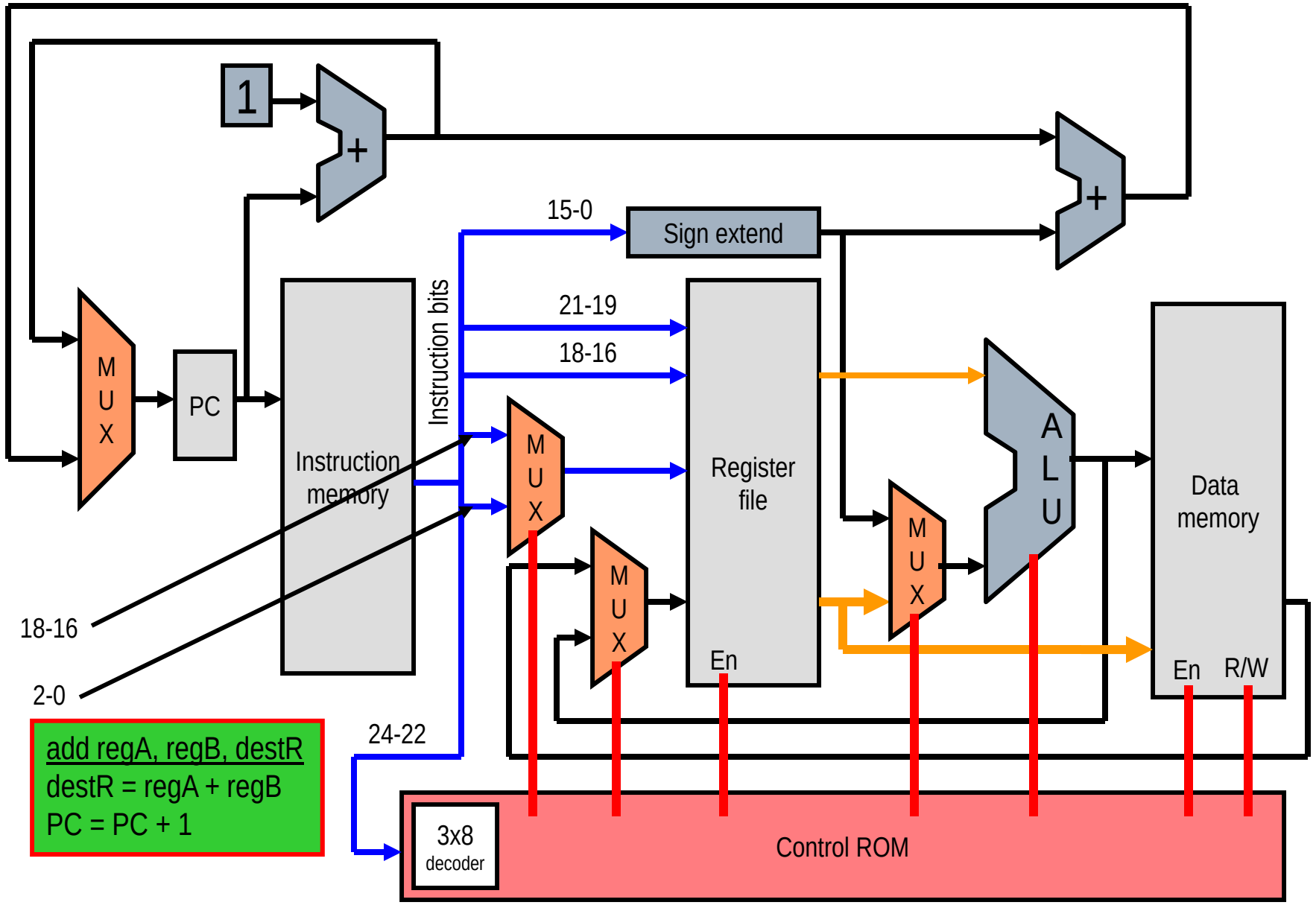
Use 2 memories for LC2

* Instructions
* Data
* 65,536 total words

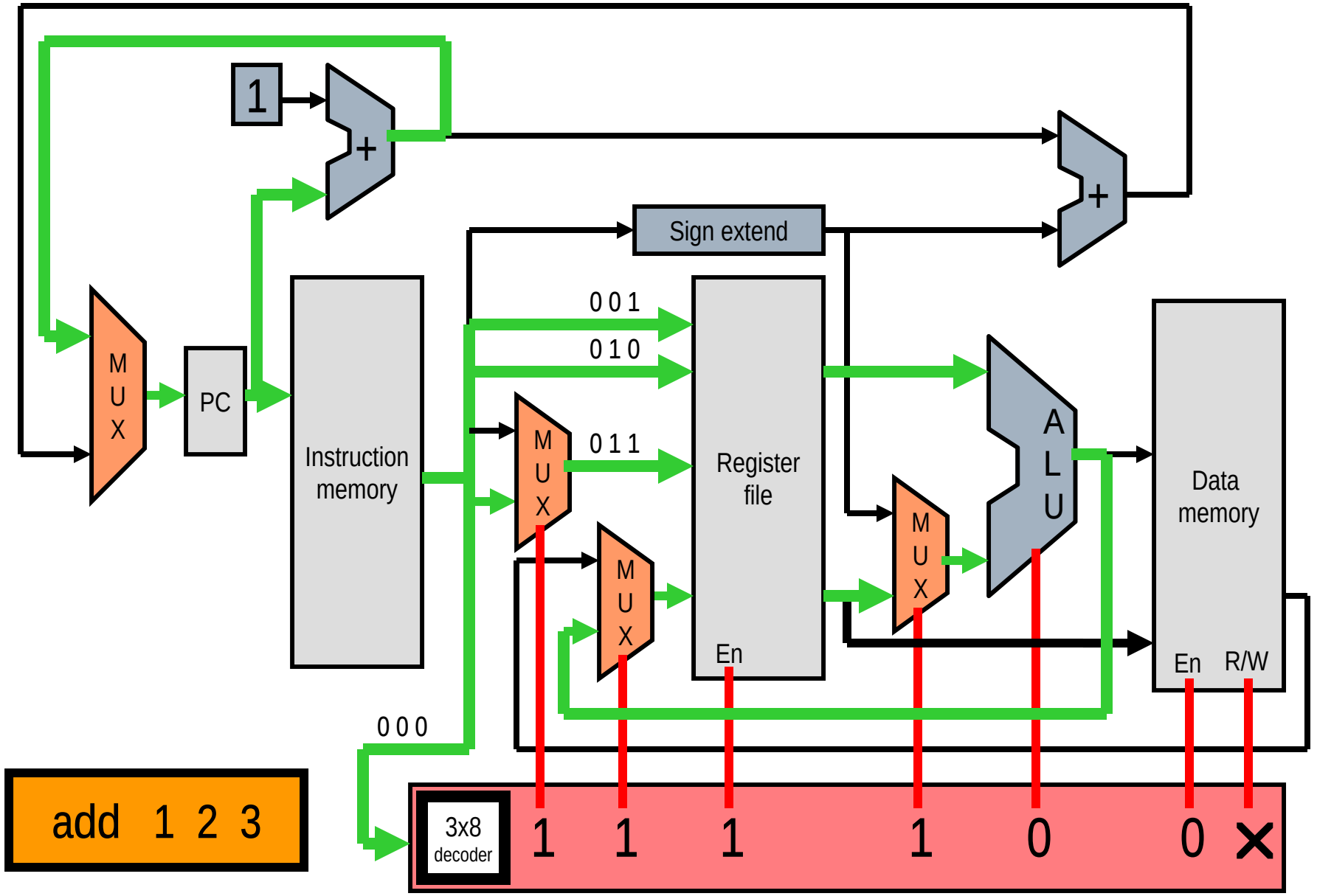
LC2K Datapath Implementation



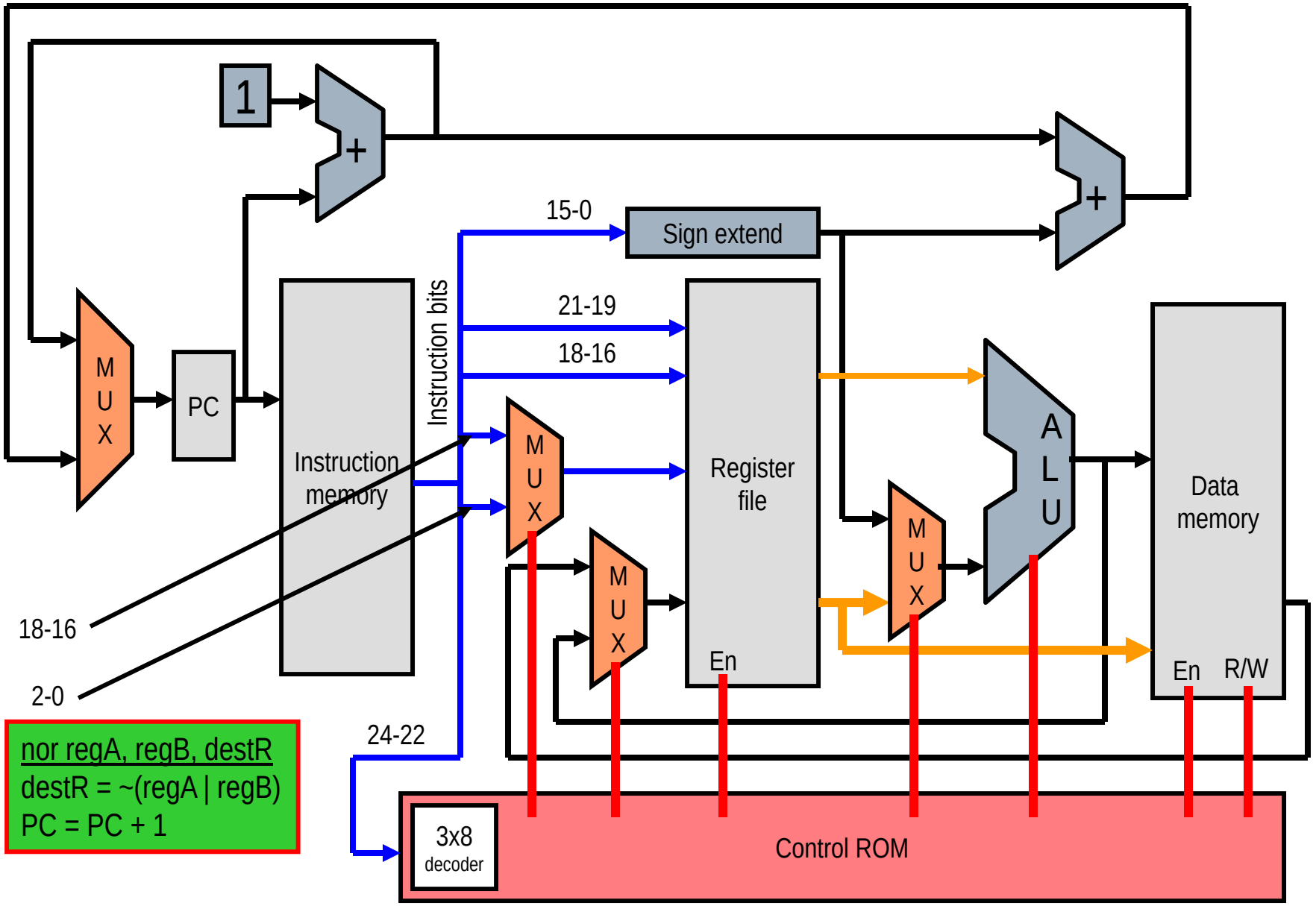
Executing an ADD Instruction



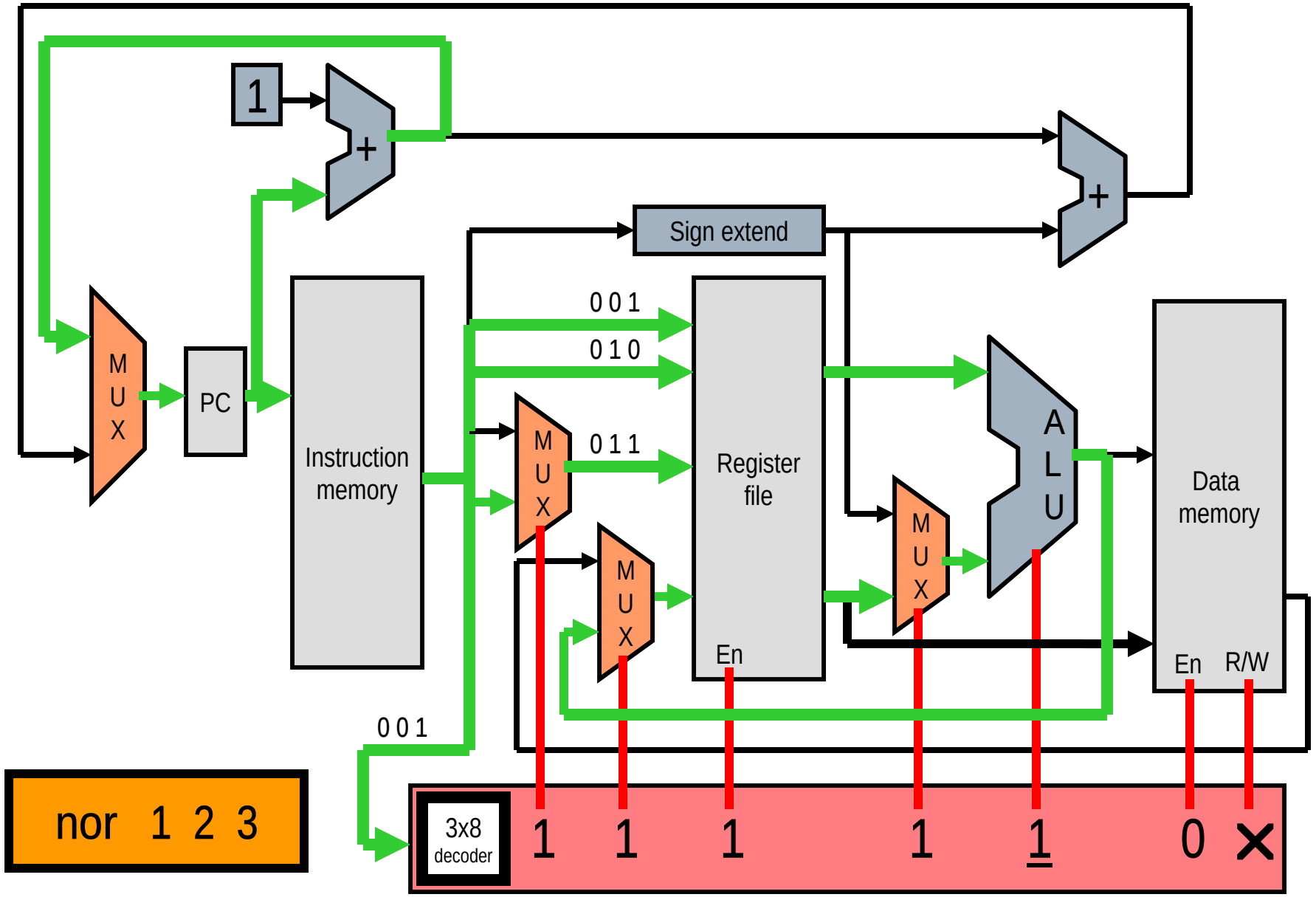
Executing an ADD Instruction on LC2K



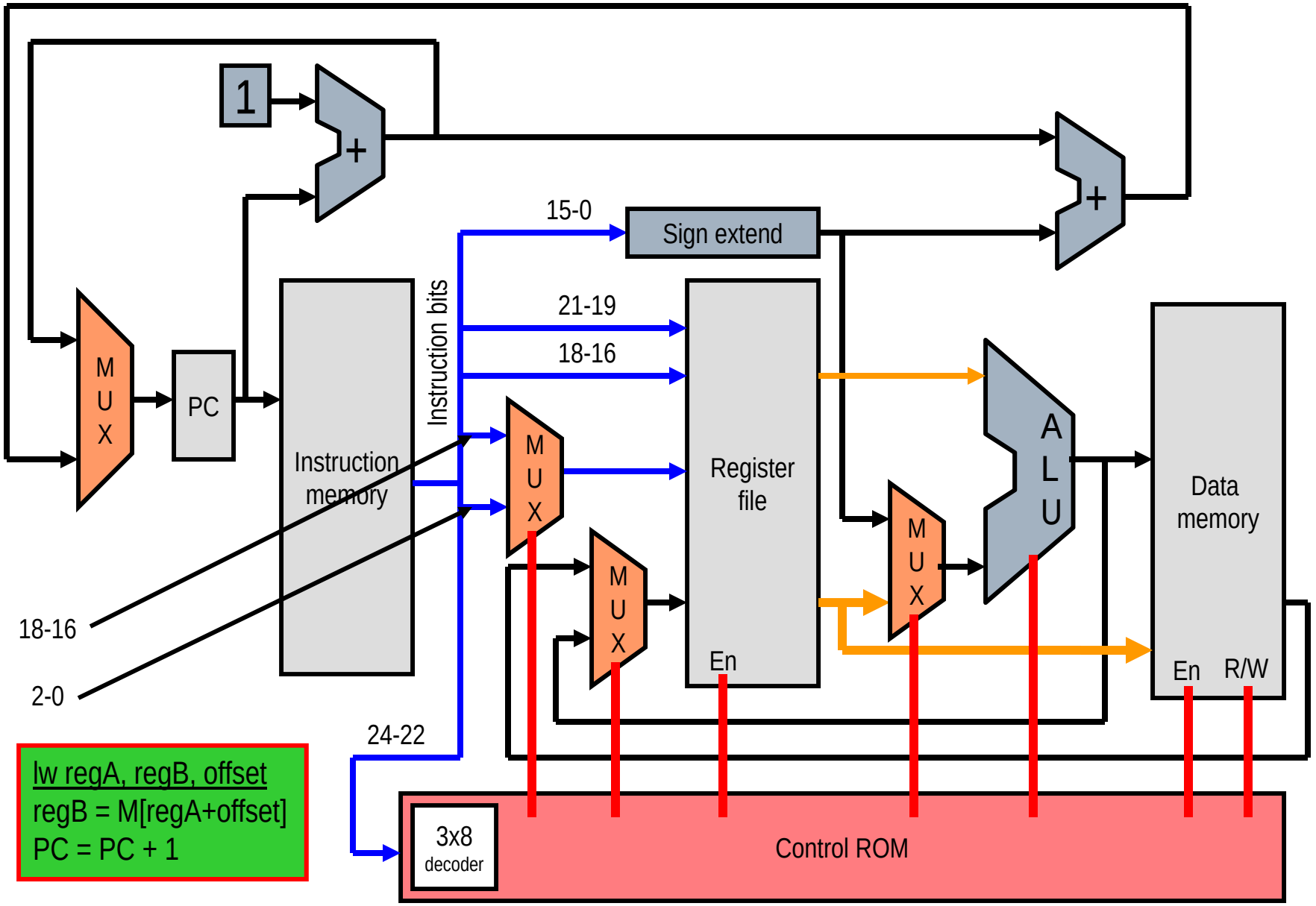
Executing a NOR Instruction



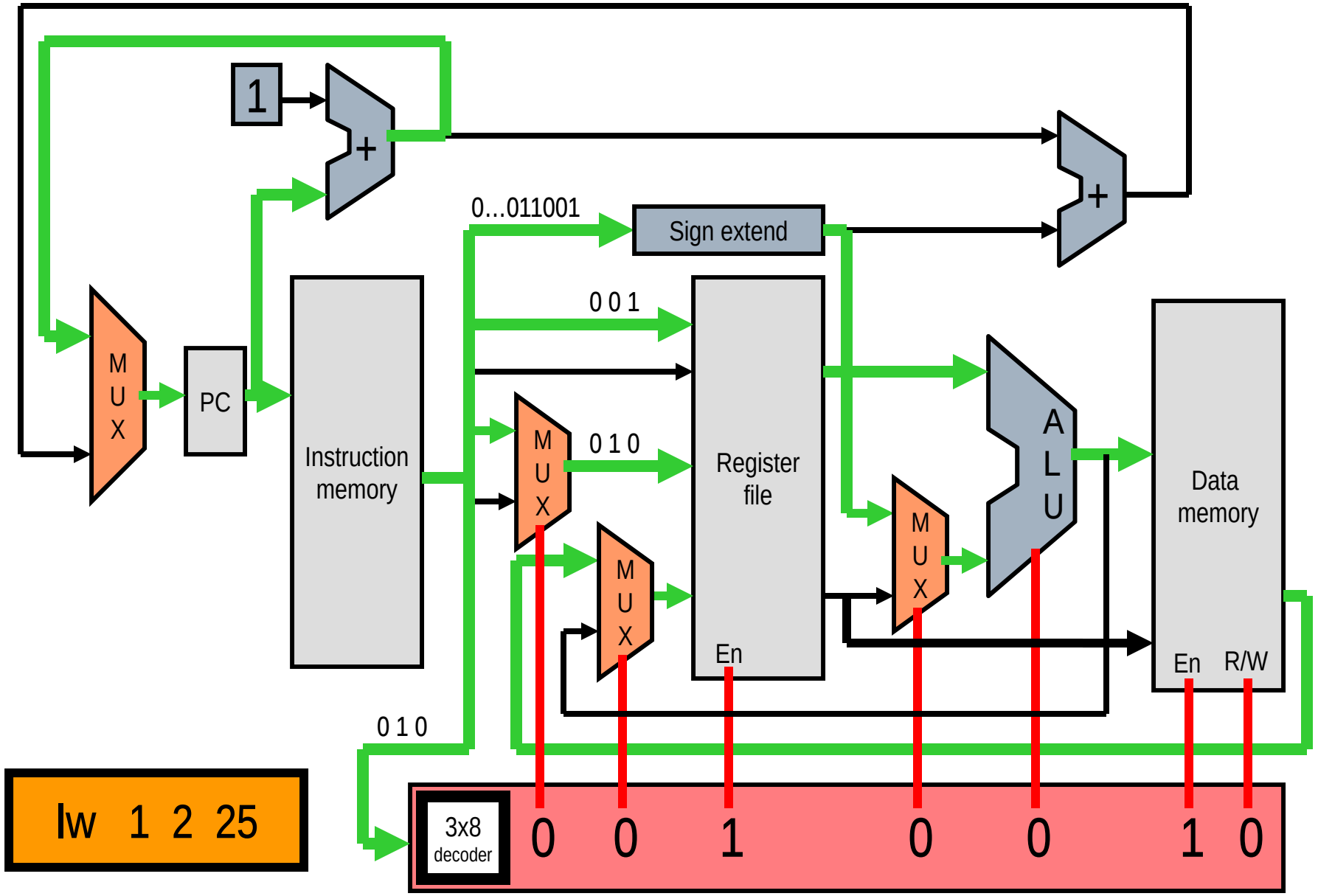
Executing NOR Instruction on LC2K



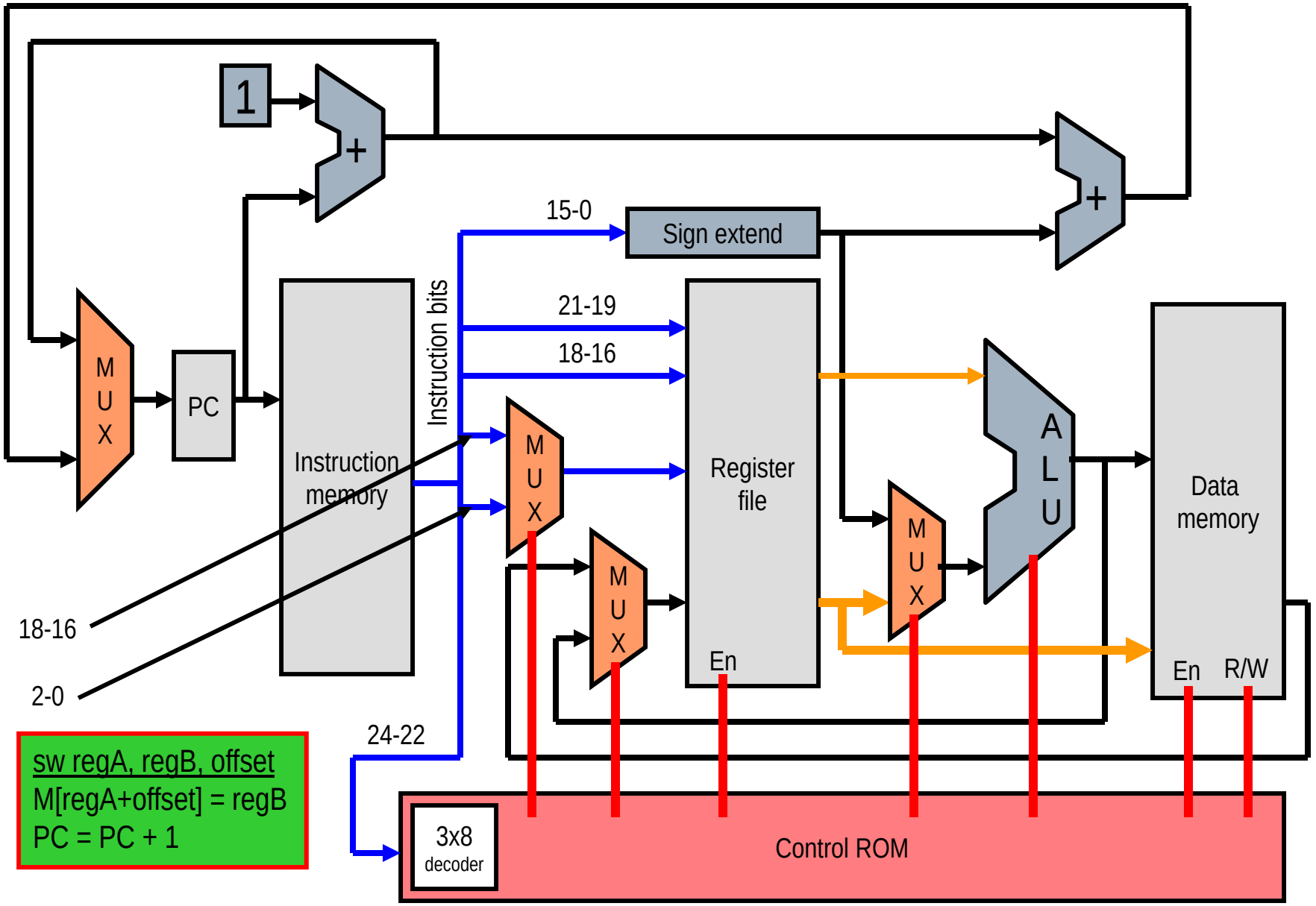
Executing a LW Instruction



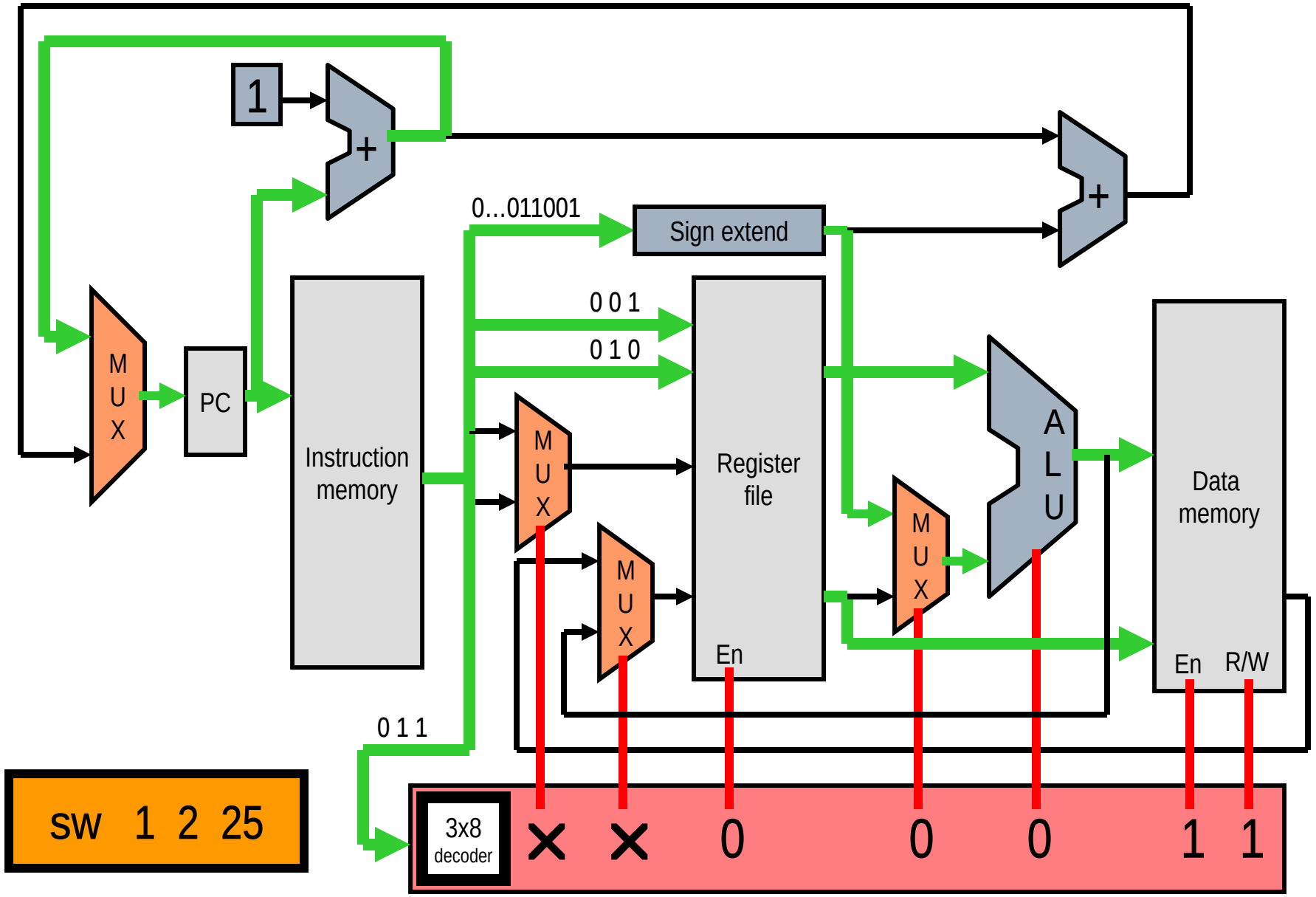
Executing a LW Instruction on LC2K



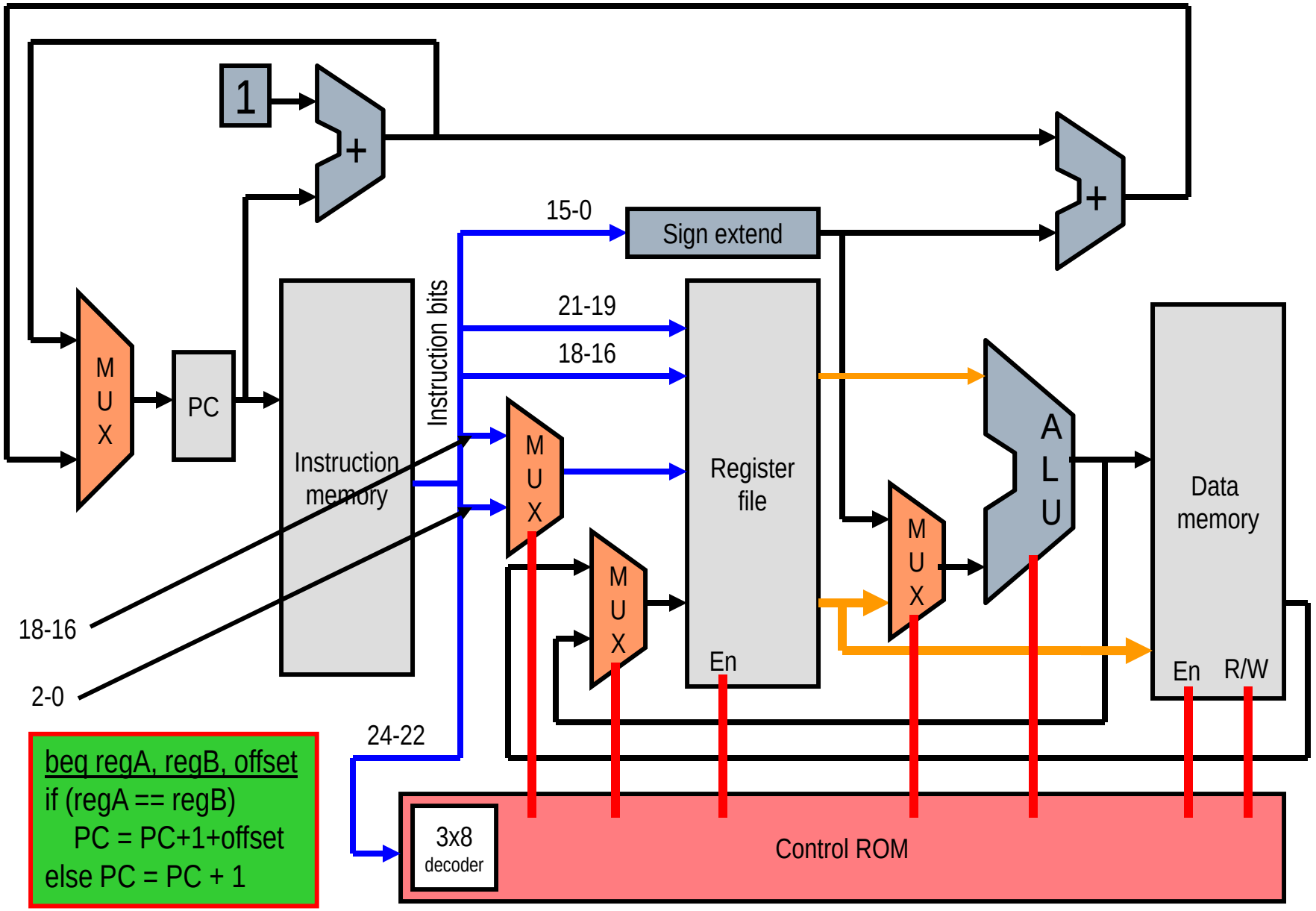
Executing a SW Instruction



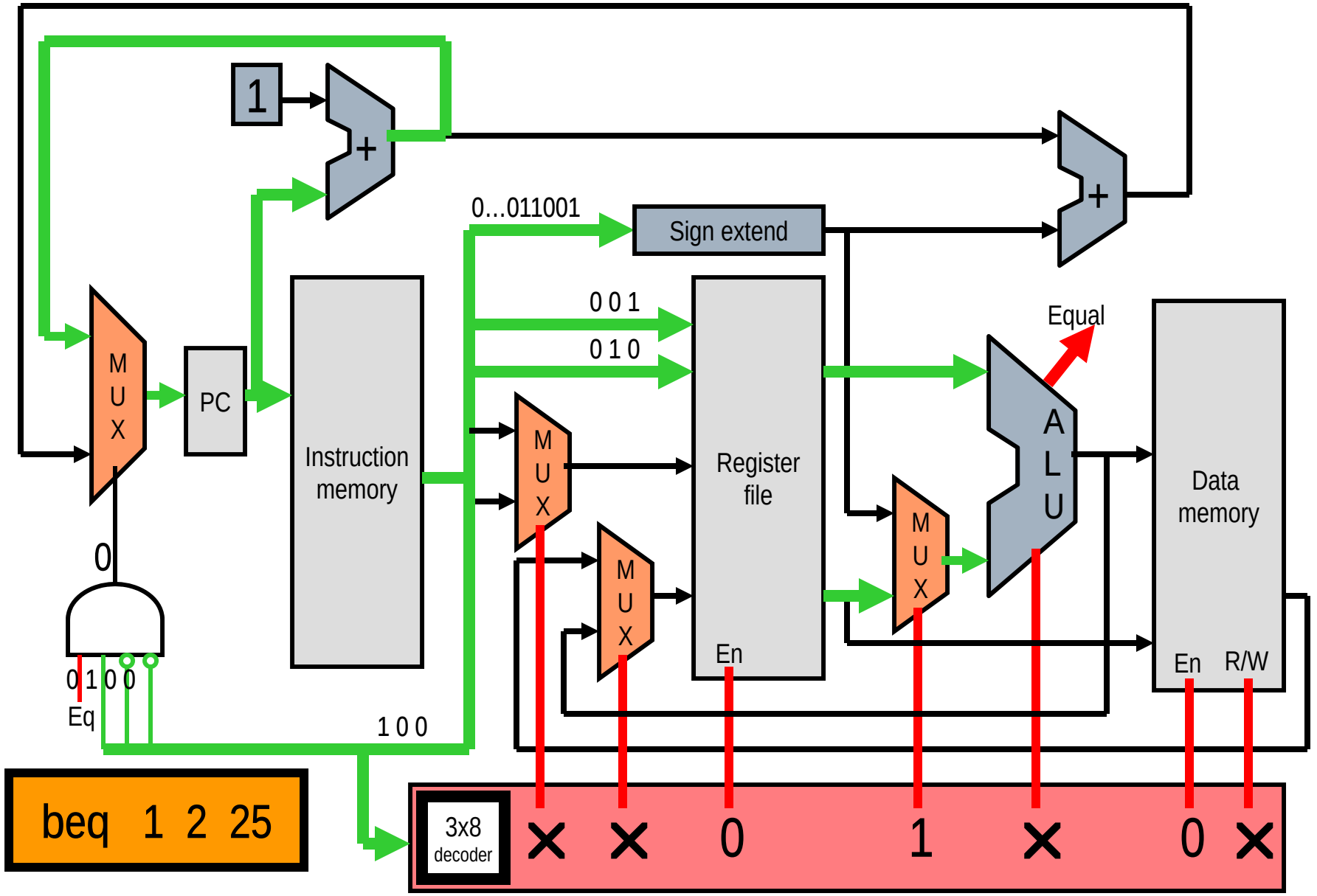
Executing a SW Instruction on LC2K



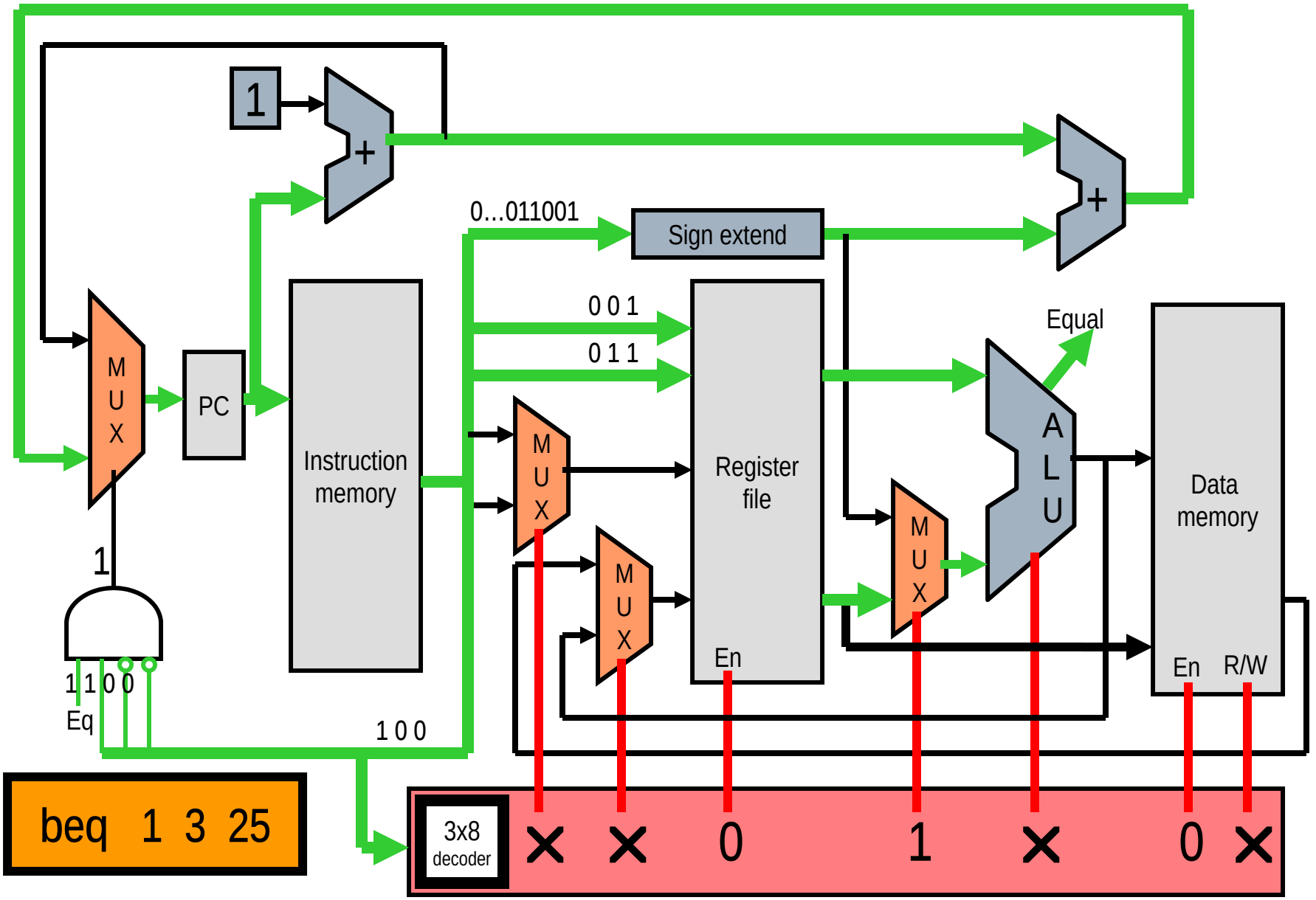
Executing a BEQ Instruction



Executing a “not taken” BEQ on LC2K



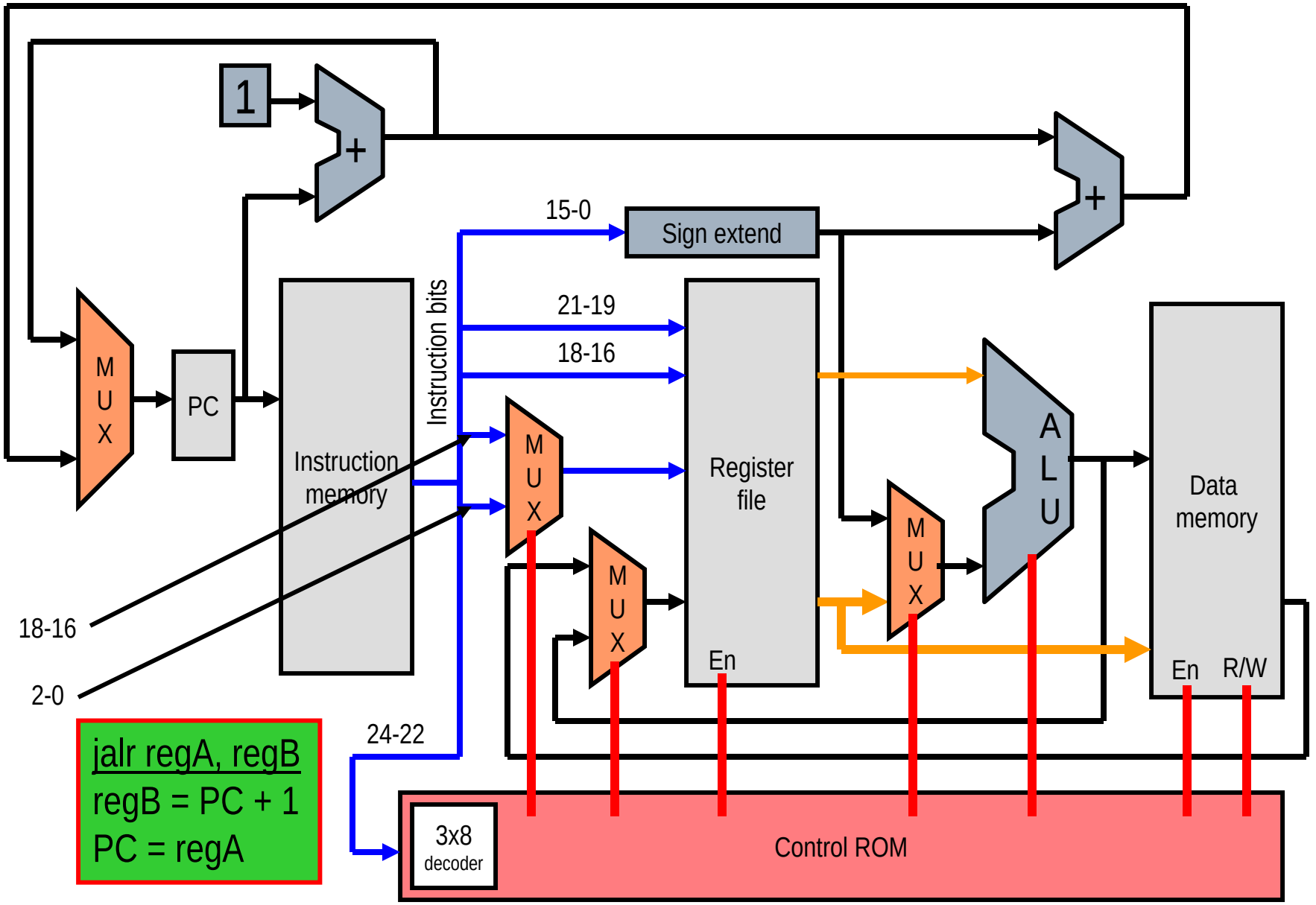
Executing a “taken” BEQ on LC2K



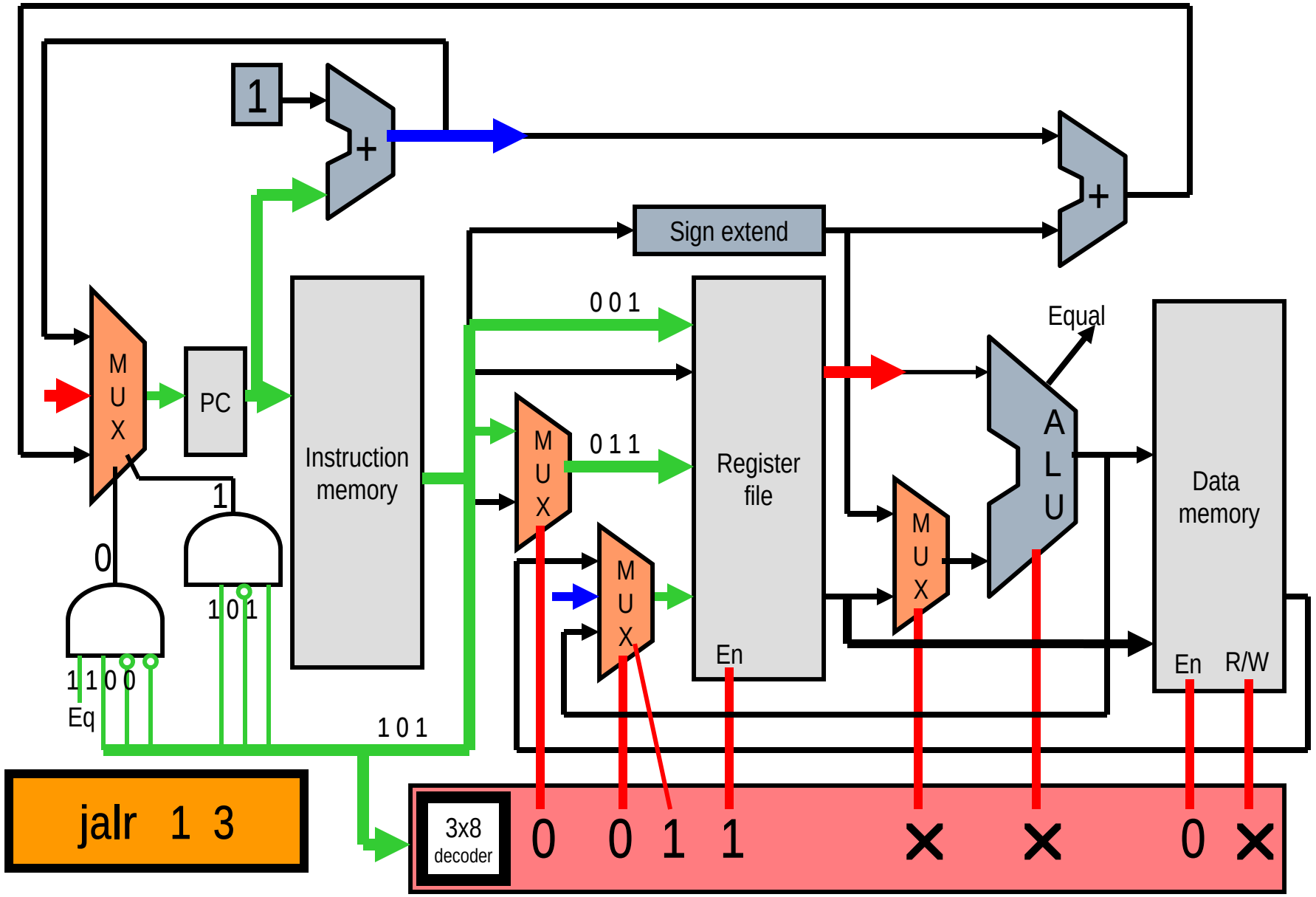
So Far, So Good

- 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

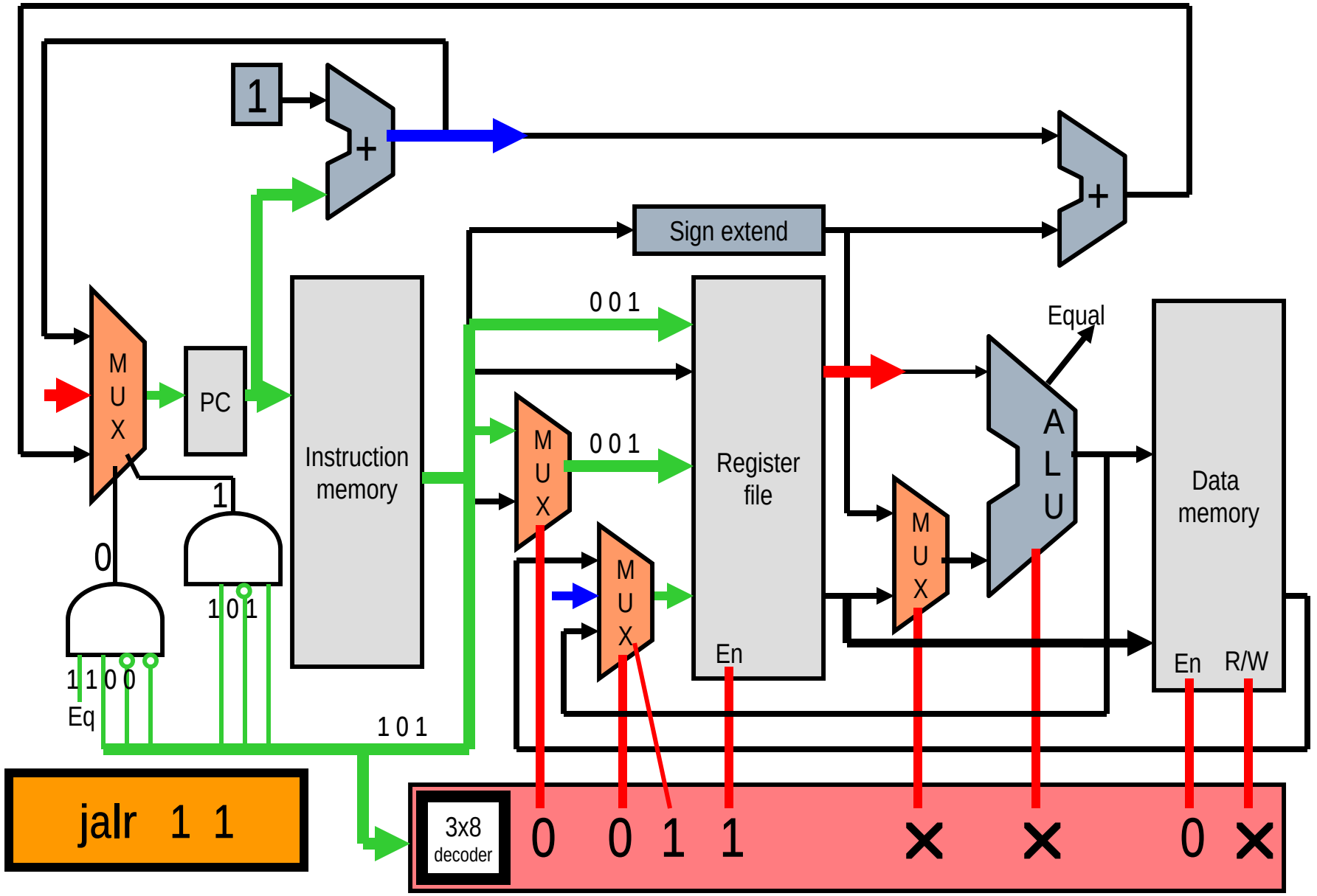
Executing a JALR Instruction



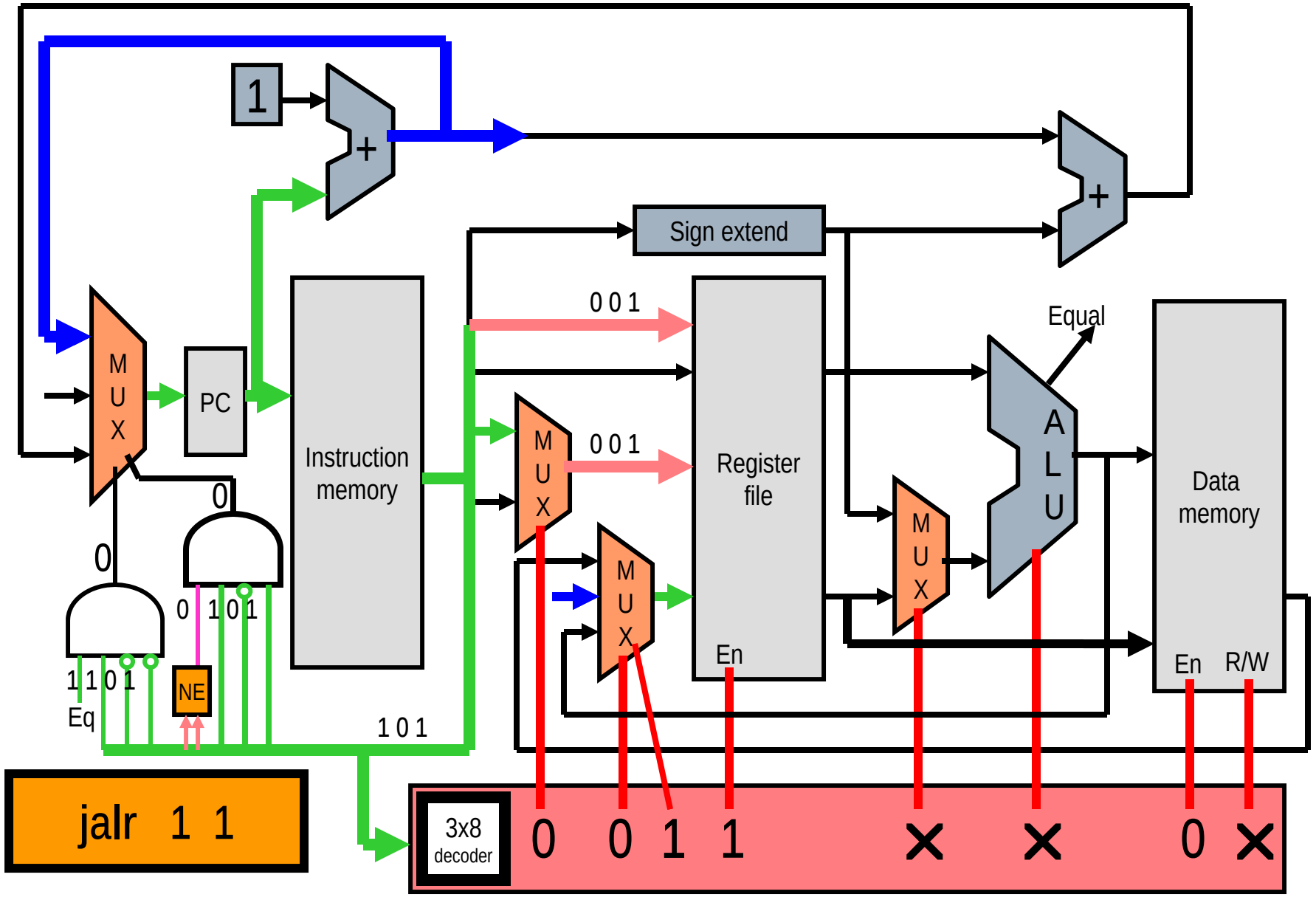
Executing a JALR Instruction on LC2K



What If regA = regB for a JALR ?



Changes for a JALR 1 1 Instruction



See you all in April!

- Professor DAS on deck!