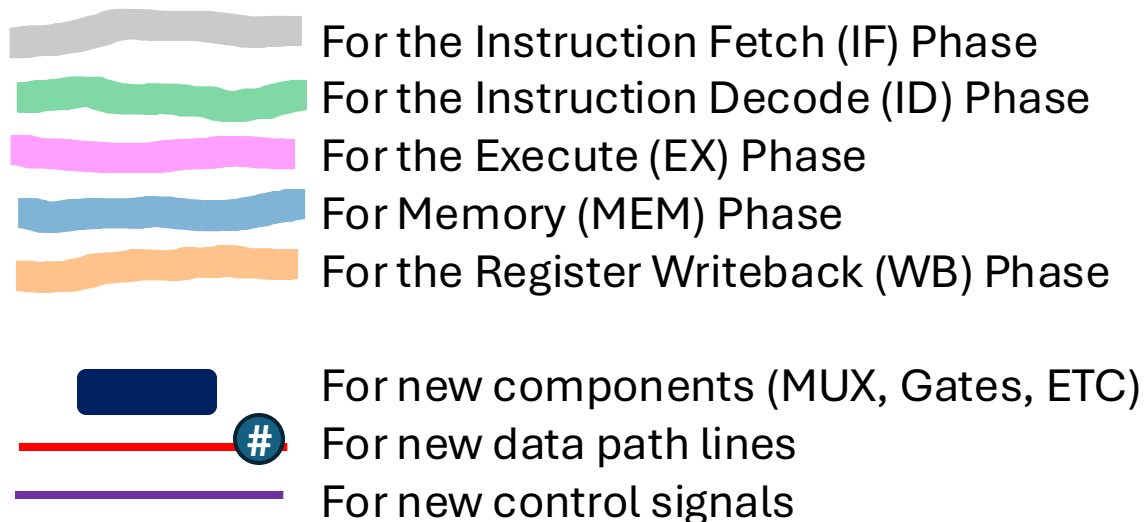


## Homework 7 - Directions

1. For this homework you will be highlighting the datapath through the **pipelined version of the RISC-V processor**. I have provided as a review a solution for the jr instruction on the single cycle datapath, as well as a solution for jr on the pipelined version of the RISC-V processor.
2. Like Homework 6 for each instruction you are allowed to cut wires, add any new wires, add new connections to existing wires, and to add additional components that we have used in Logisim – aka, multiplexors, adders, shifters, demux, etc. See the example I provided for jr as an example. Don't forget about the control signals, how should they be set? Does your instruction need new control signal(s)?
3. You can mark up and annotate each datapath any way that you want – on the computer, by hand, take pictures, etc. But your changes must be clear and easy to follow to receive full credit. On my sample I used different colors for each phase of the pipeline, and for my pipeline datapath and control signal changes:



**HINT: This is the pipelined version of the processor – MAKE sure that everything flows through all stages of the pipeline as you formulate your solutions**

Q1:  
Jump Register  
jr  
Datapath  
Note jr = jalr x0, Rs,0

1. PC points to jr instruction
2. Instruction is fetched and decoded – Rs1 and Rs2 placed on register file, Rs2 is zero register. Immediate value hard coded to 0 Control signals established
3. Immediate value added to Rs1
4. ALU result is new PC address
5. New wire routes PC address up to new mux that places PC value on output of upper right mux which will be the next PC address (on next clock cycle)

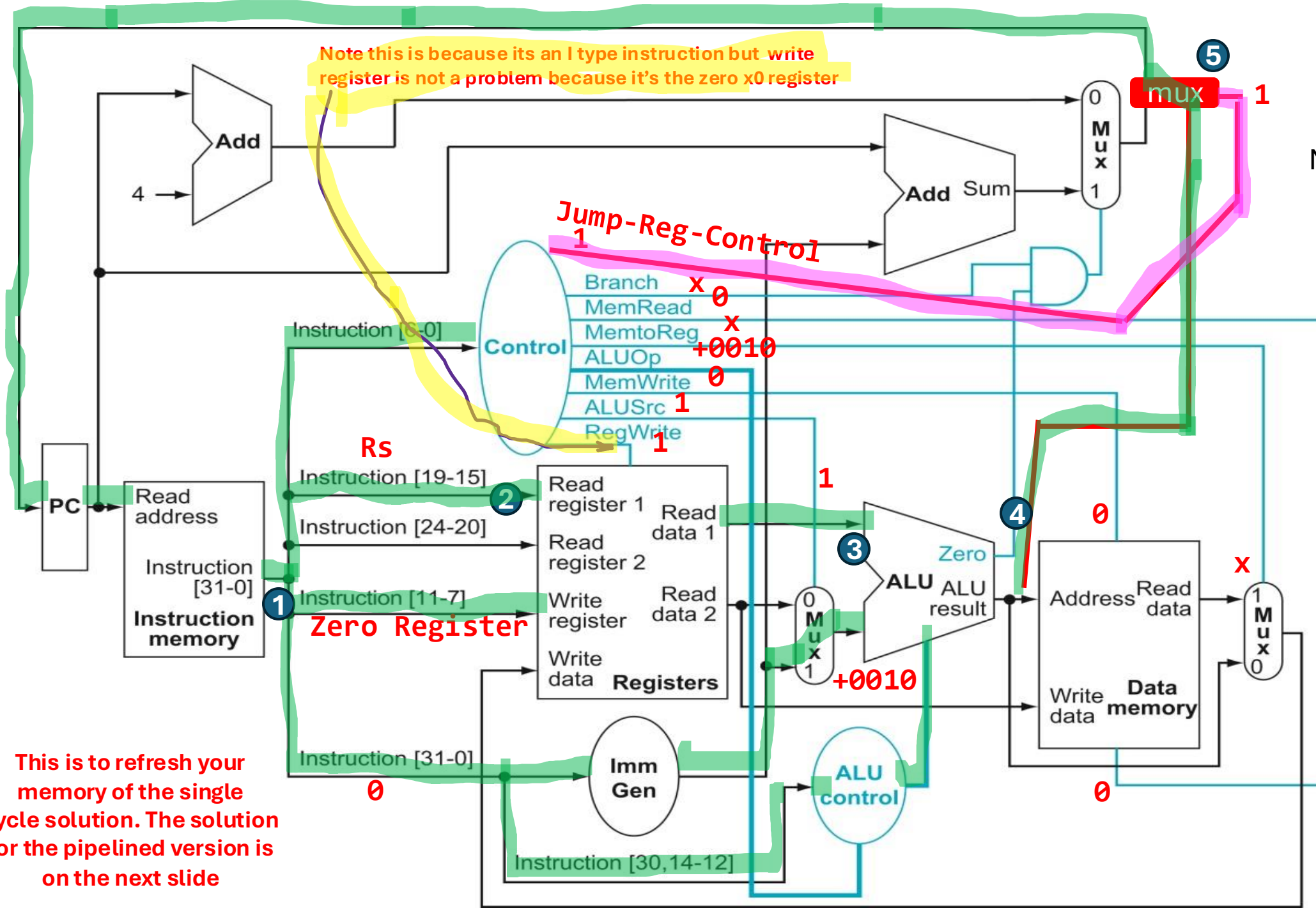
Note this is because its an I type instruction but write register is not a problem because it's the zero x0 register

Jump-Reg-Control

Rs

Zero Register

This is to refresh your memory of the single cycle solution. The solution for the pipelined version is on the next slide



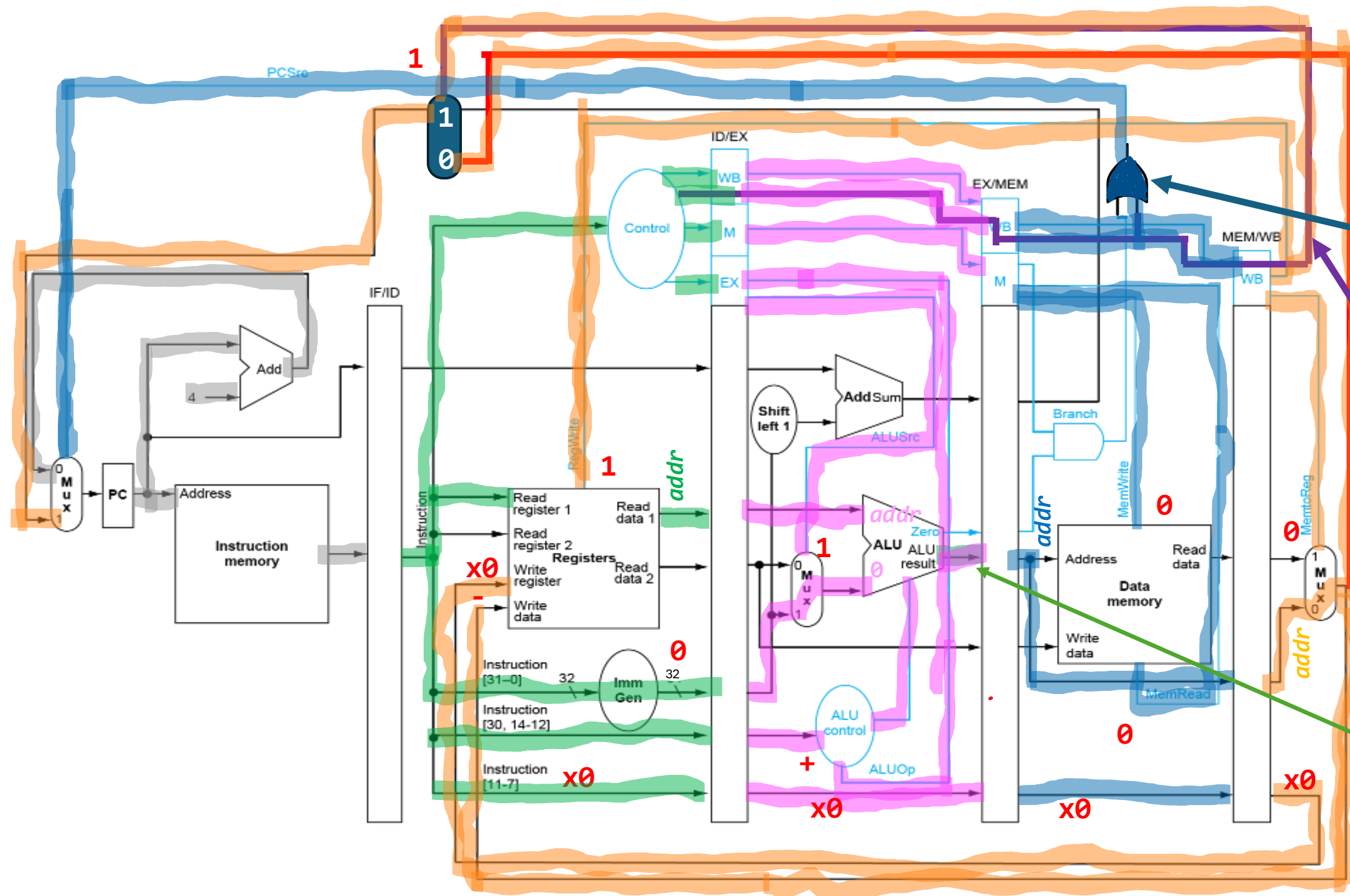
Q1:  
Jump Register  
**jr**  
Datapath  
 $jr = jalr\ x0, Rs, 0$

Updated logic to  
control PC  
address

New JR  
Control Signal

New wire to  
route address  
back to the  
PC

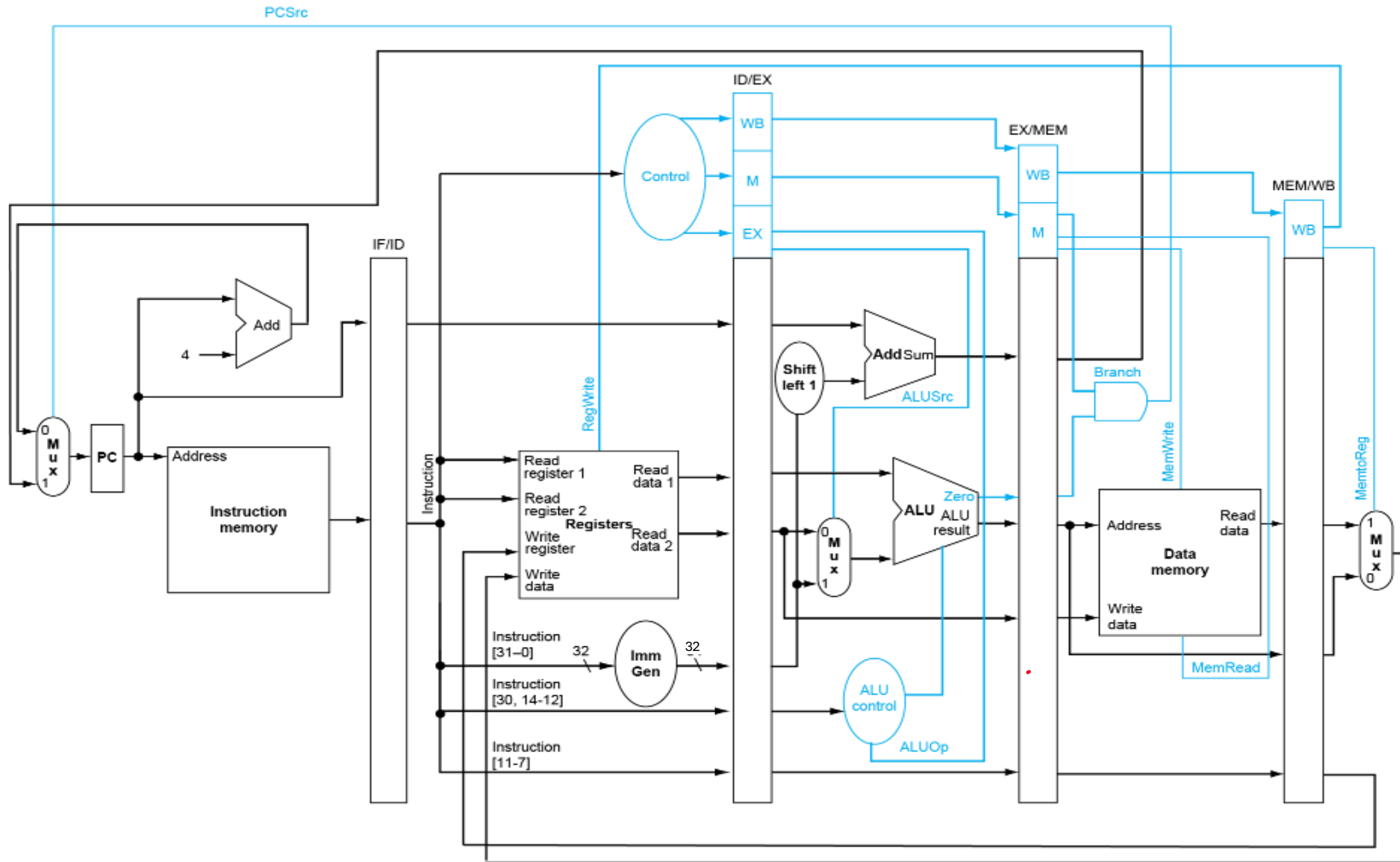
Correct address  
for PC established  
here





# Q3: Jump Link **jal** Datapath

Hint, modify  
jr solution jal =  
jalr, ra, Rs, imm



Q4:  
Load Upper  
Immediate  
**lui** Datapath

