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)?

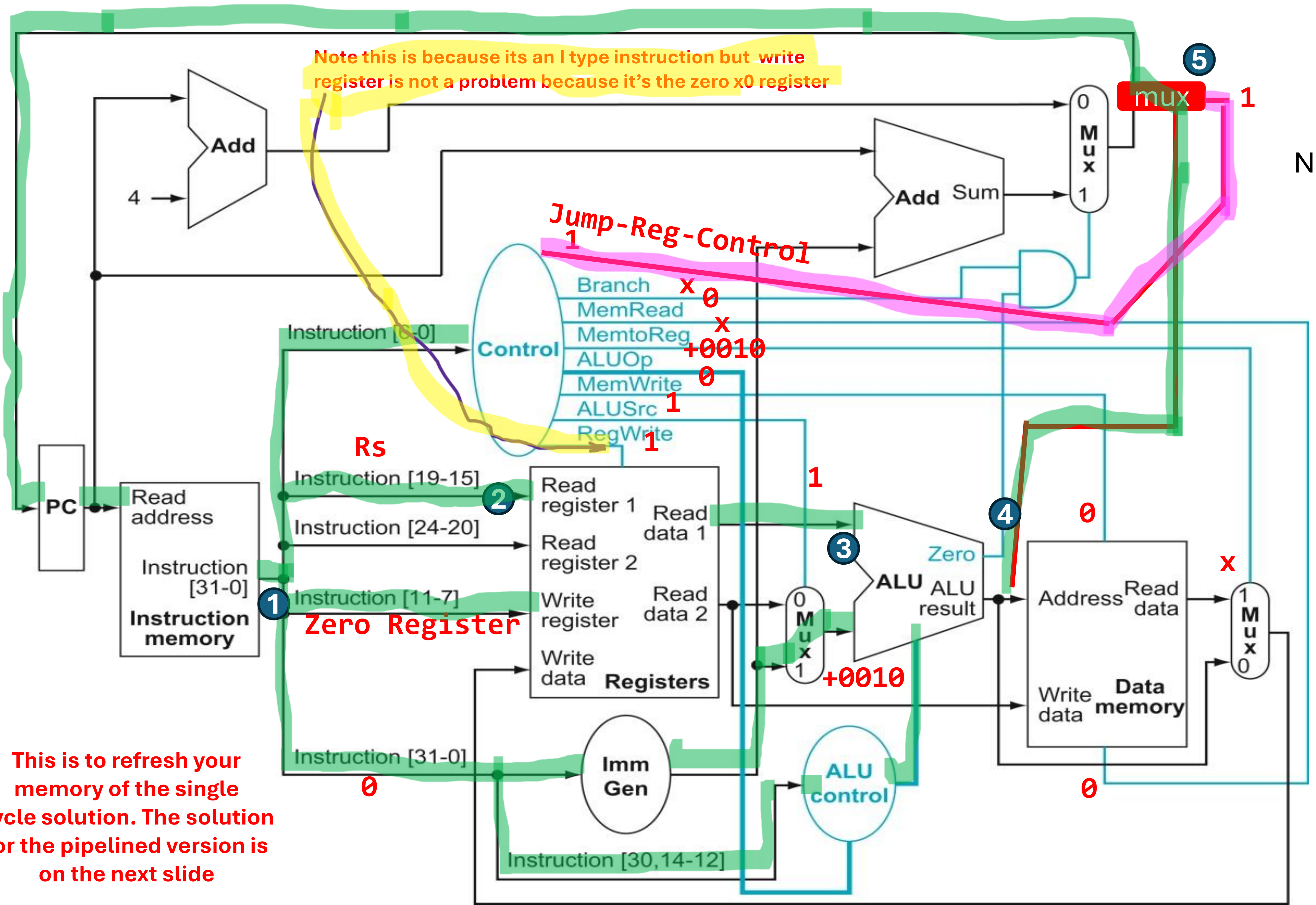
3. You can mark up and annotate each datapath any way that you want – on the computer, by hand, take pictures, etc. But your changes must be clear and easy to follow to receive full credit. On my sample I used different colors for each phase of the pipeline, and for my pipeline datapath and control signal changes:

⬜	For the Instruction Fetch (IF) Phase
🟩	For the Instruction Decode (ID) Phase
🟪	For the Execute (EX) Phase
🟦	For Memory (MEM) Phase
🟧	For the Register Writeback (WB) Phase
⬛	For new components (MUX, Gates, ETC)
🟥 #	For new data path lines
🟪	For new control signals

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

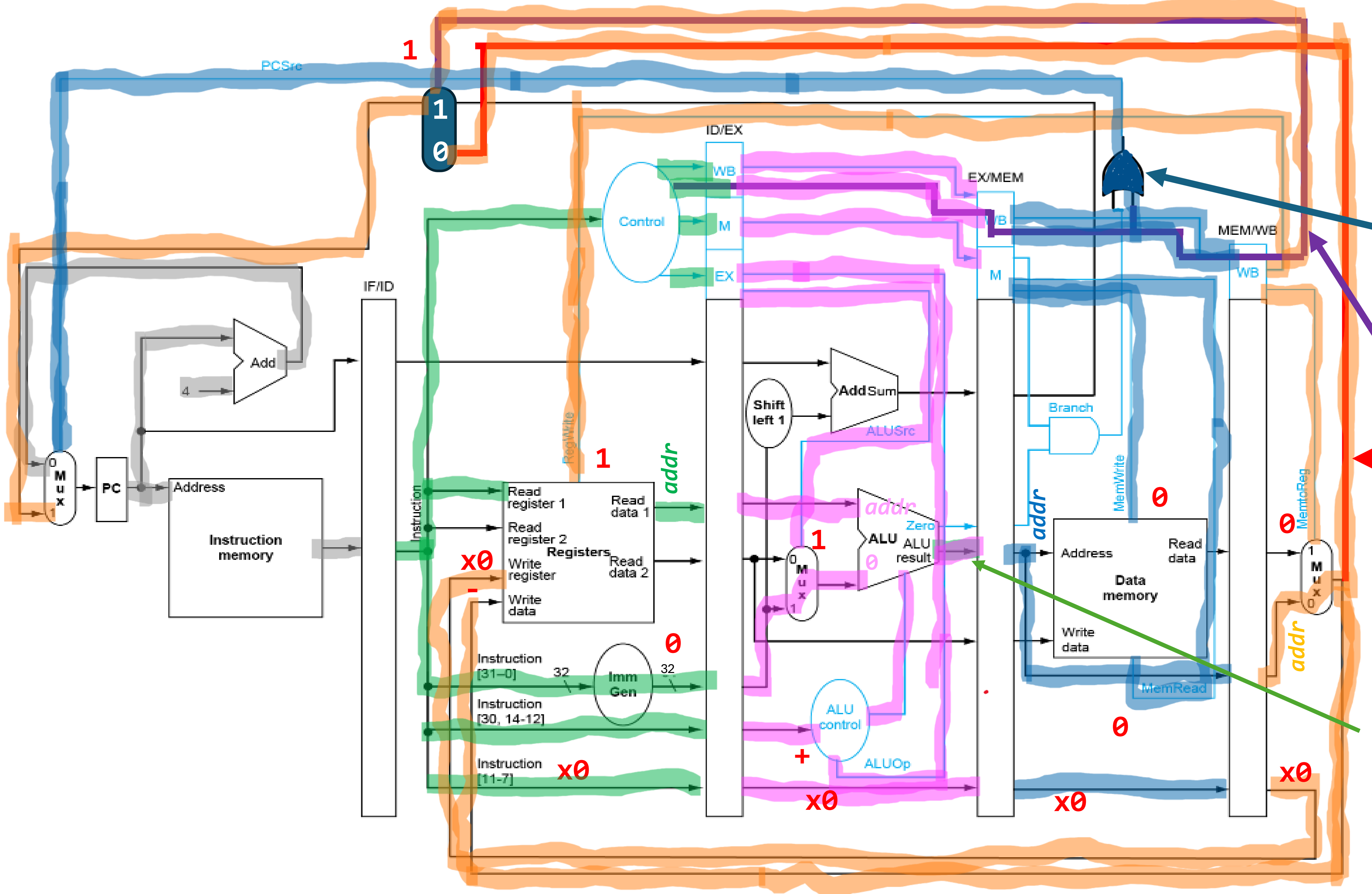
Q1: Jump Register **jr** Datapath

Note jr = jalr x0, Rs,0



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

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 addresss
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)



Q1:

Jump Register

jr

Datapath

jr = jalr x0, Rs,0

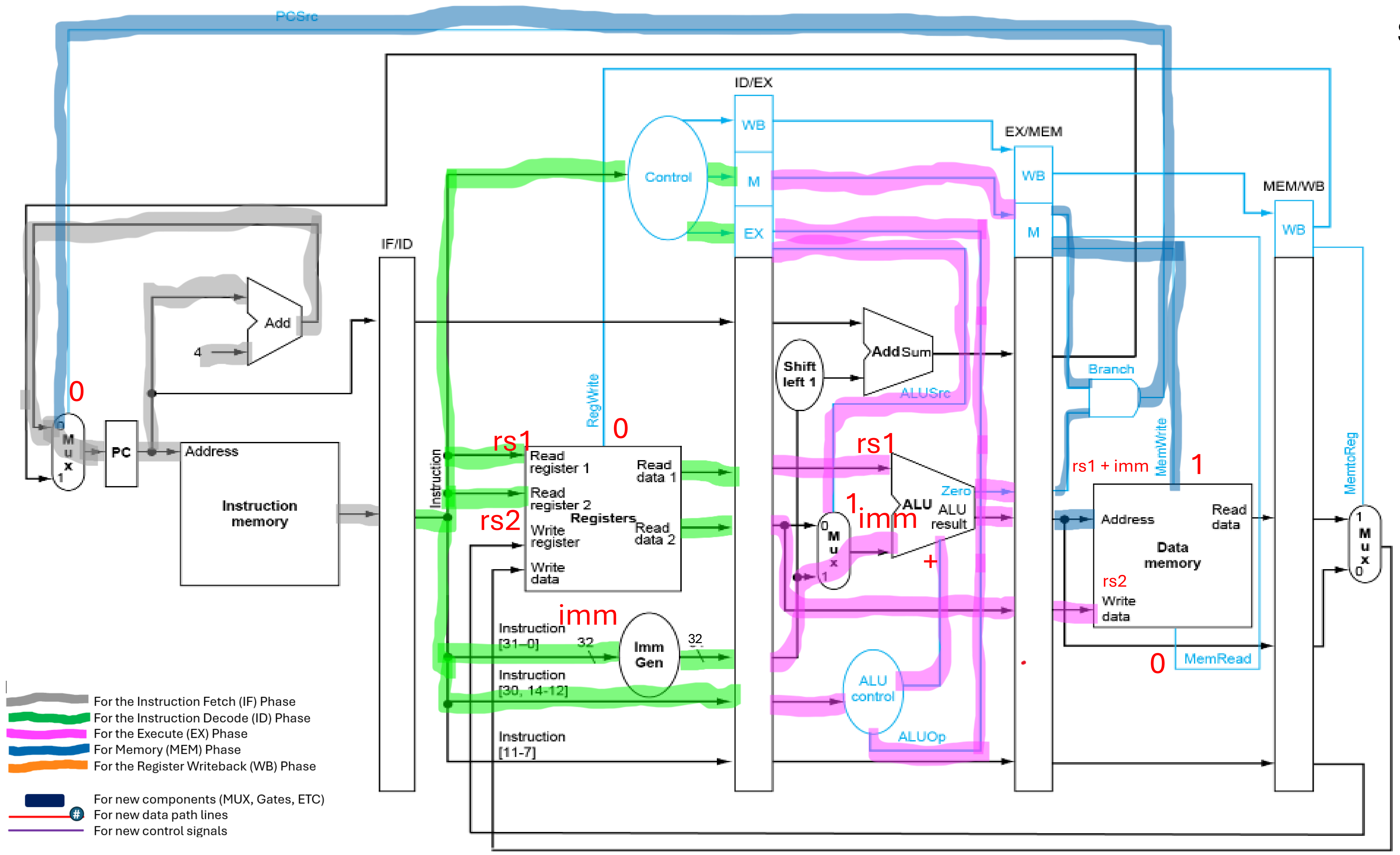
Updated logic to control PC address

New JR Control Signal

New wire to route address back to the PC

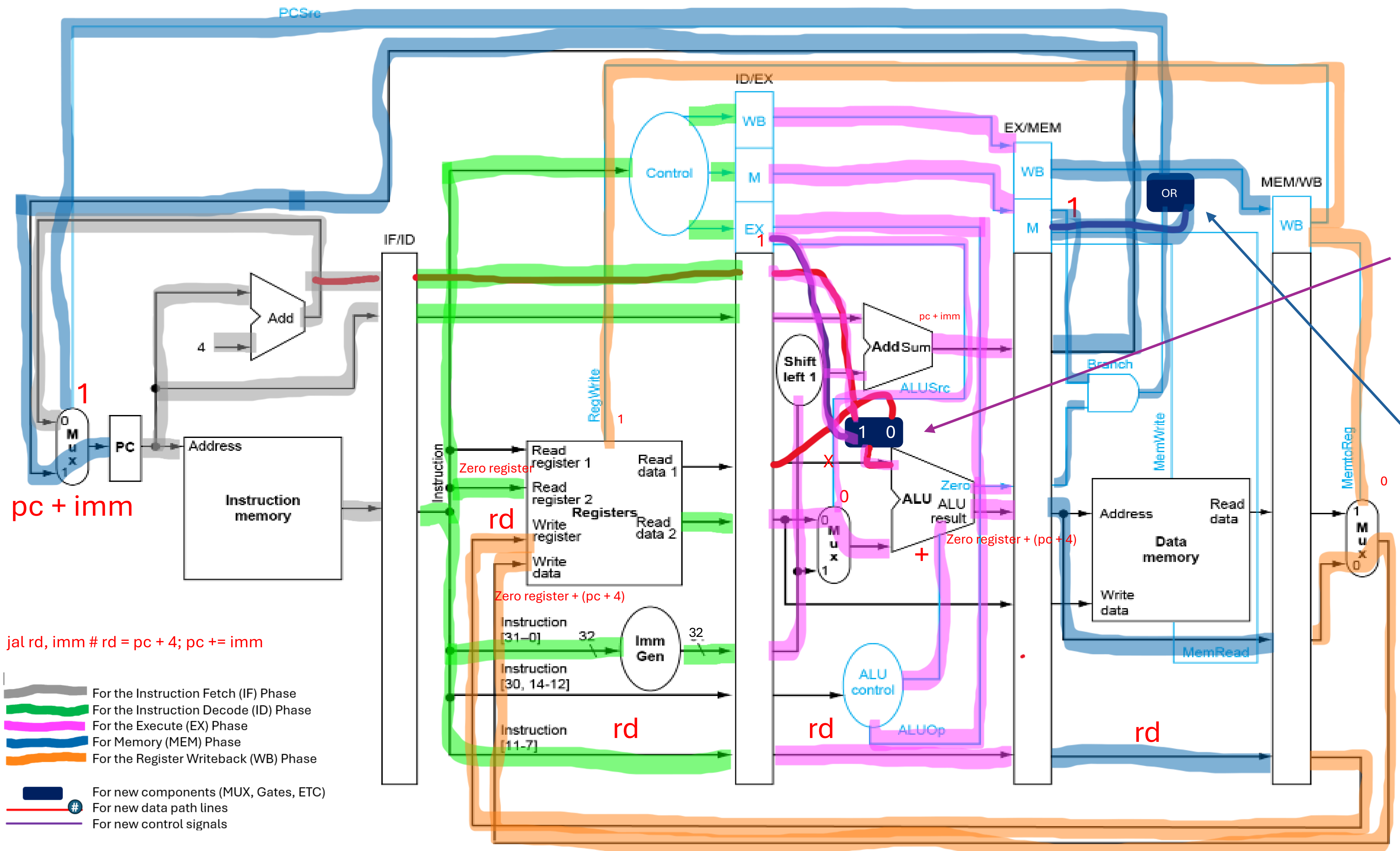
Correct address for PC established here

Q2: Store Word (sw) Datapath



- 🟫 For the Instruction Fetch (IF) Phase
- 🟩 For the Instruction Decode (ID) Phase
- 🟪 For the Execute (EX) Phase
- 🟦 For Memory (MEM) Phase
- 🟧 For the Register Writeback (WB) Phase
- ■ For new components (MUX, Gates, ETC)
- ● For new data path lines
- ― For new control signals

Q3: Jump Link **jal** Datapath



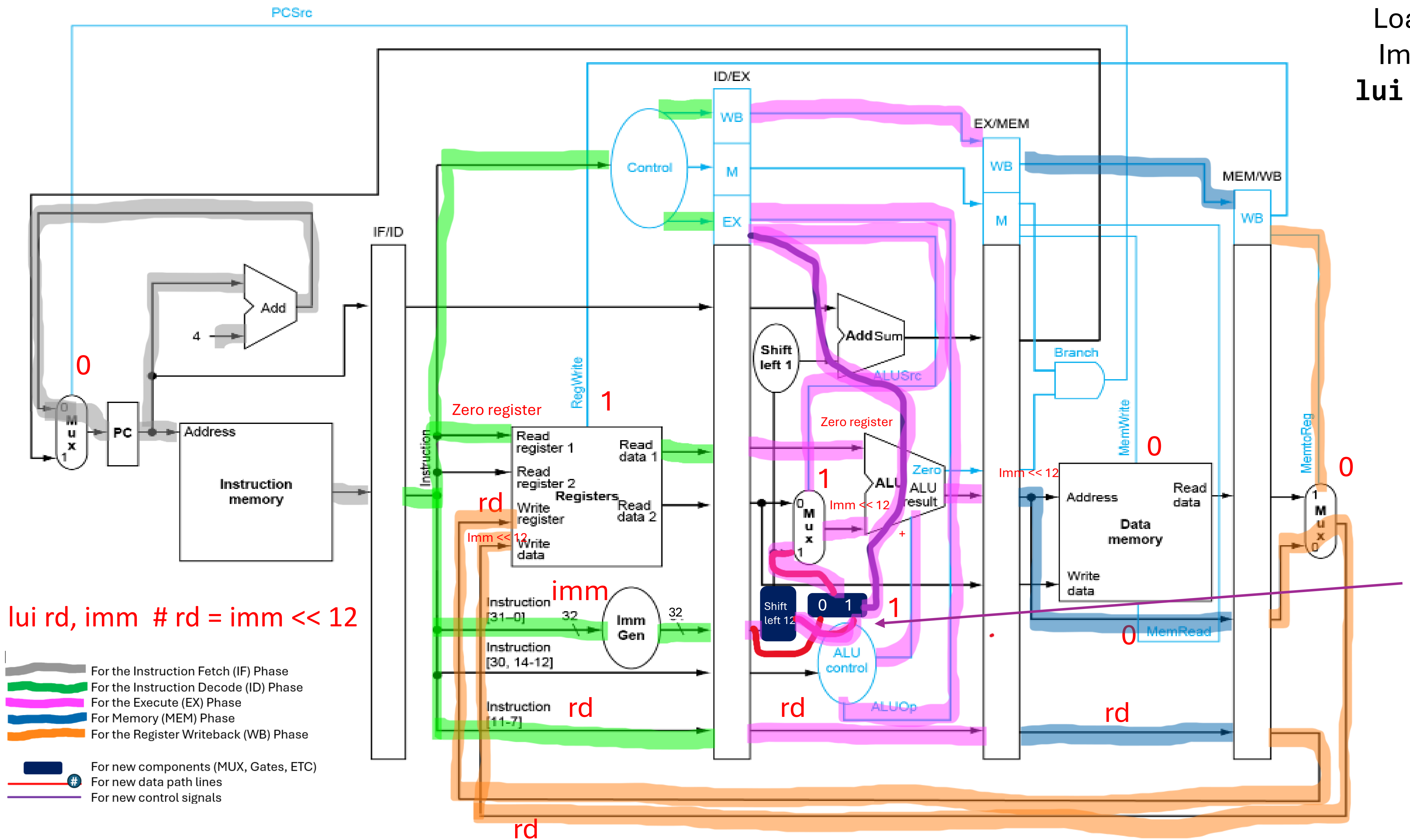
jal rd, imm # rd = pc + 4; pc += imm

Added new control signal and mux to choose between pc+4 and read data 1 in execute phase.

Added another control signal to choose between branch or jump in memory phase.

▬	For the Instruction Fetch (IF) Phase
🟩	For the Instruction Decode (ID) Phase
▬	For the Execute (EX) Phase
🟦	For Memory (MEM) Phase
🟧	For the Register Writeback (WB) Phase
■	For new components (MUX, Gates, ETC)
—	For new data path lines
—	For new control signals

Q4: Load Upper Immediate `lui` Datapath



lui rd, imm # rd = imm << 12

▬	For the Instruction Fetch (IF) Phase
🟩	For the Instruction Decode (ID) Phase
▬	For the Execute (EX) Phase
🟦	For Memory (MEM) Phase
🟧	For the Register Writeback (WB) Phase
■	For new components (MUX, Gates, ETC)
―	For new data path lines
―	For new control signals

Added new control signal, shifter and mux to choose between shifting the immediate by 12 bits in the execute phase of the pipeline.