






## Homework 6 - Directions

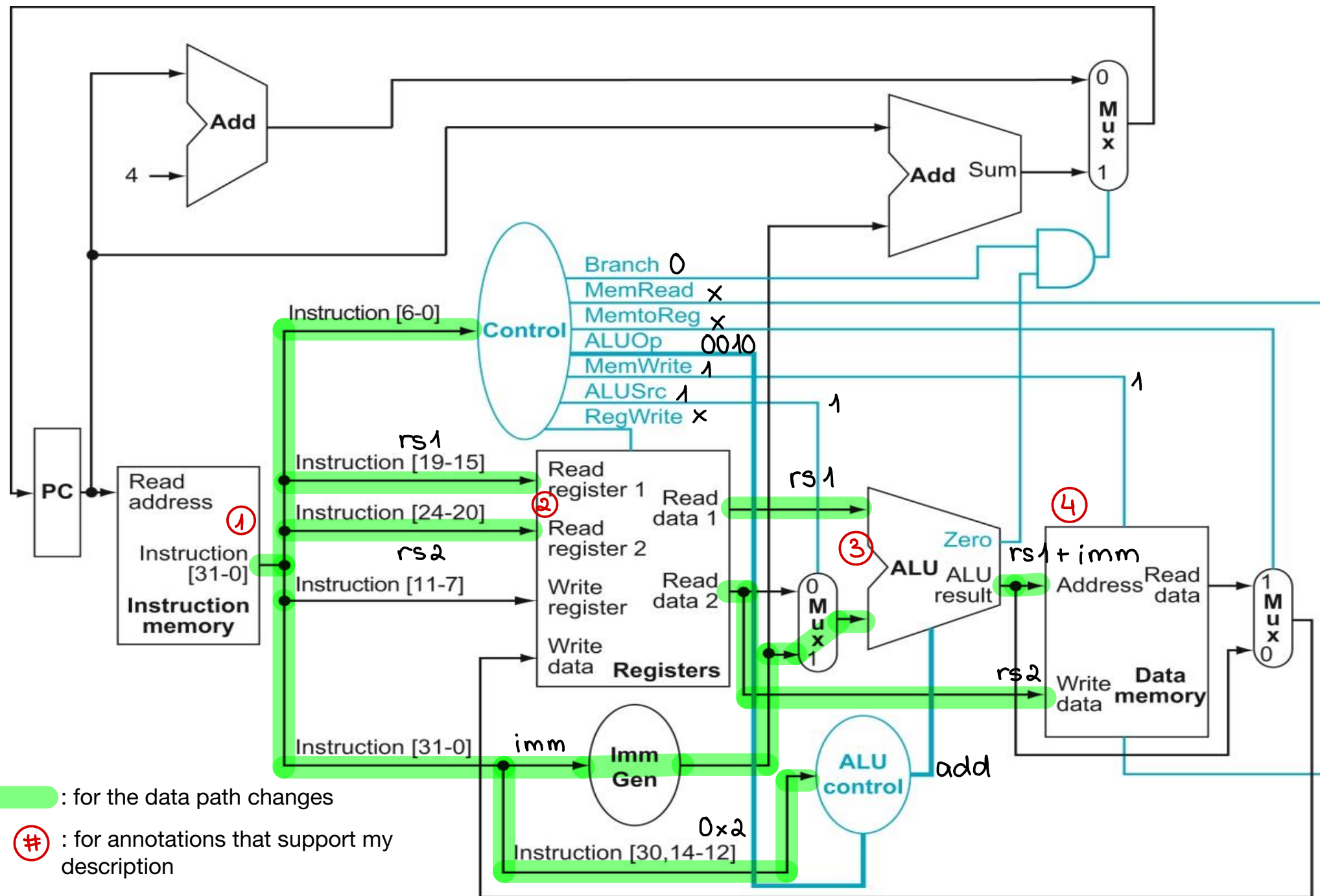
1. For this homework you will be highlighting the datapath through the single cycle RISC-V design for 3 instructions. Note that I did Question 0 to show you what I want for the `jr` datapath. Notice how the data path itself is highlighted, the control signals are specified, and a brief writeup is provided showing the key steps that happen from fetching the `jr` instruction, all the way through the data path, until ultimately the PC is updated
2. You will be doing the same for the `sw` instruction, the `jal` instruction, and the `lui` instruction. Please consult the datasheet it will help you to setup the register file.
3. 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)?
4. 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:

-  For the data path changes
-  For new control signals
-  For new components
-  For new wires
-  For annotations that support my description



# Q1: Store Word (sw) Datapath

1. PC points to sw instruction
2. Instruction is fetched and decoded - rs1 and rs2 placed on register file. Immediate value is converted to a 32bit value. funct3 is loaded to the ALU control
3. Immediate value is added to rs1. The value at rs2 is loaded to the write data pin of the data memory component
4. The value at rs2 is written to the memory address calculated by adding the offset to the memory address stored at rs1



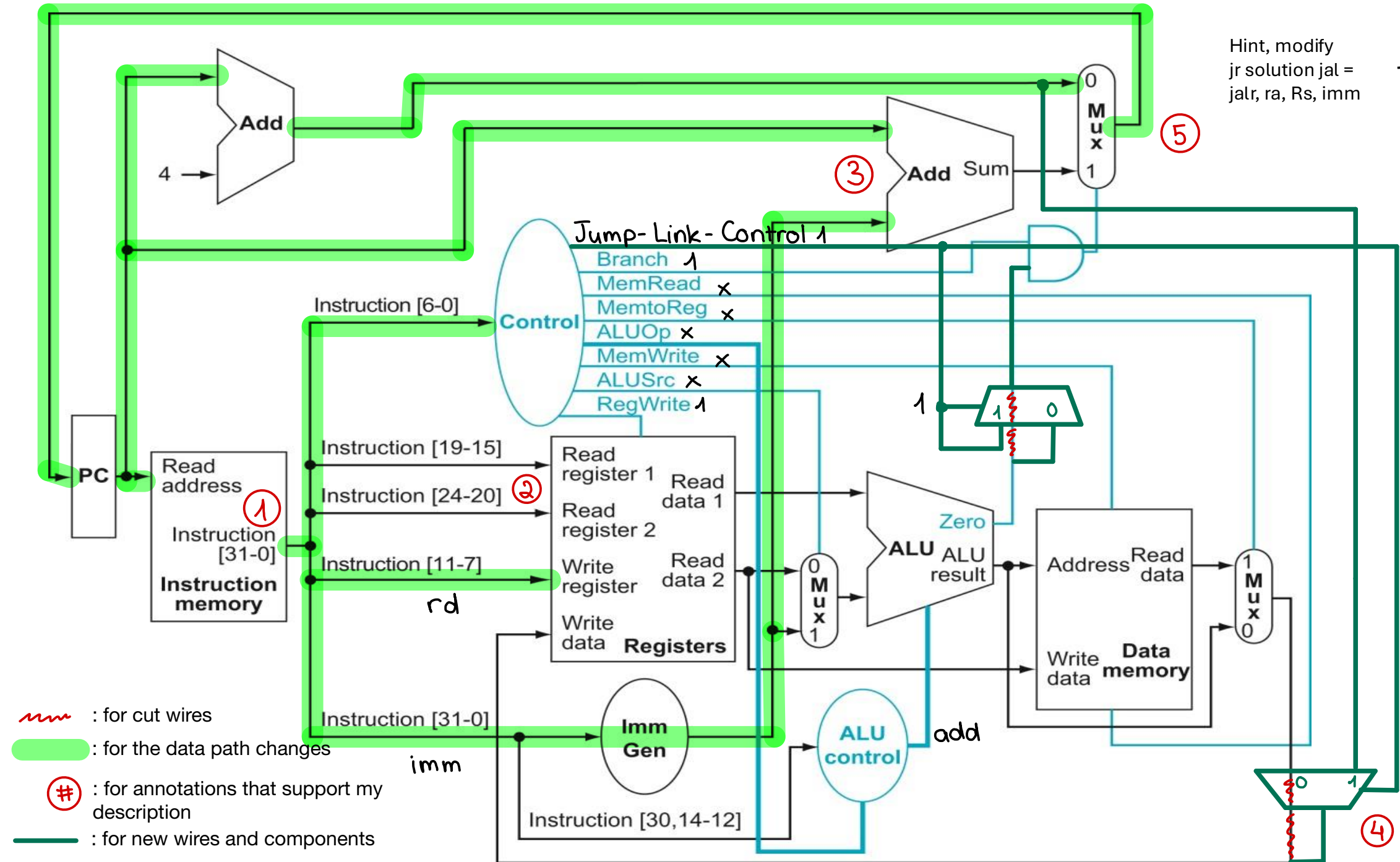
for the data path changes

# : for annotations that support my description

## Q2: Jump Link **jal** Datapath

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

1. PC points to jal instruction
2. Instruction is fetched and decoded - rd is placed on the write register. The immediate value is converted to a 32bit value
3. The immediate value is added to the PC counter. PC counter is incremented by 4
4. The incremented PC counter value is loaded to the Write data pin of the Registers control component. The incremented PC counter value is written on the rd register
5. The jumped PC counter value is loaded back to the PC counter





## Load Upper Immediate **lui** Datapath

- 
- The diagram illustrates the MIPS processor architecture with the following components and connections:
- PC (Program Counter):** Provides the address to Instruction memory.
  - Instruction memory:** Receives the address from the PC and outputs instruction fields [6-0], [19-15], [24-20], [11-7], and [31-0].
  - Control:** Receives instruction fields [6-0] and outputs control signals to the ALU control, Imm Gen, and Registers.
  - Registers:** Receives instruction fields [19-15], [24-20], [11-7], and [31-0]. It outputs read data 1 and read data 2 to the ALU.
  - ALU control:** Receives control signals from the Control and outputs control signals to the ALU.
  - Imm Gen (Immediate Generator):** Receives instruction field [31-0] and outputs the immediate value to the ALU.
  - ALU:** Receives read data 1, read data 2, and the immediate value. It outputs the ALU result to the Data memory.
  - Data memory:** Receives the ALU result and outputs the final result to the Mux.
  - Mux (Multiplexer):** Selects between the ALU result and the immediate value to output the final result.
  - Shift register:** Receives the ALU result and outputs the shifted result to the Mux.
- The diagram is annotated with green and blue lines and circles to highlight specific paths and control signals:
- Green lines and circles:** Highlight the data flow for the 'lui' instruction: PC to Instruction memory (1), Instruction [31-0] to Registers (2), Registers to ALU (3), and ALU result to Data memory (4).
  - Blue lines and circles:** Highlight control signals: Instruction [6-0] to Control, Control to ALU control, ALU control to ALU, and ALU result to Data memory.
  - Red circle:** Highlights the 'Shift-Immediate' control signal.