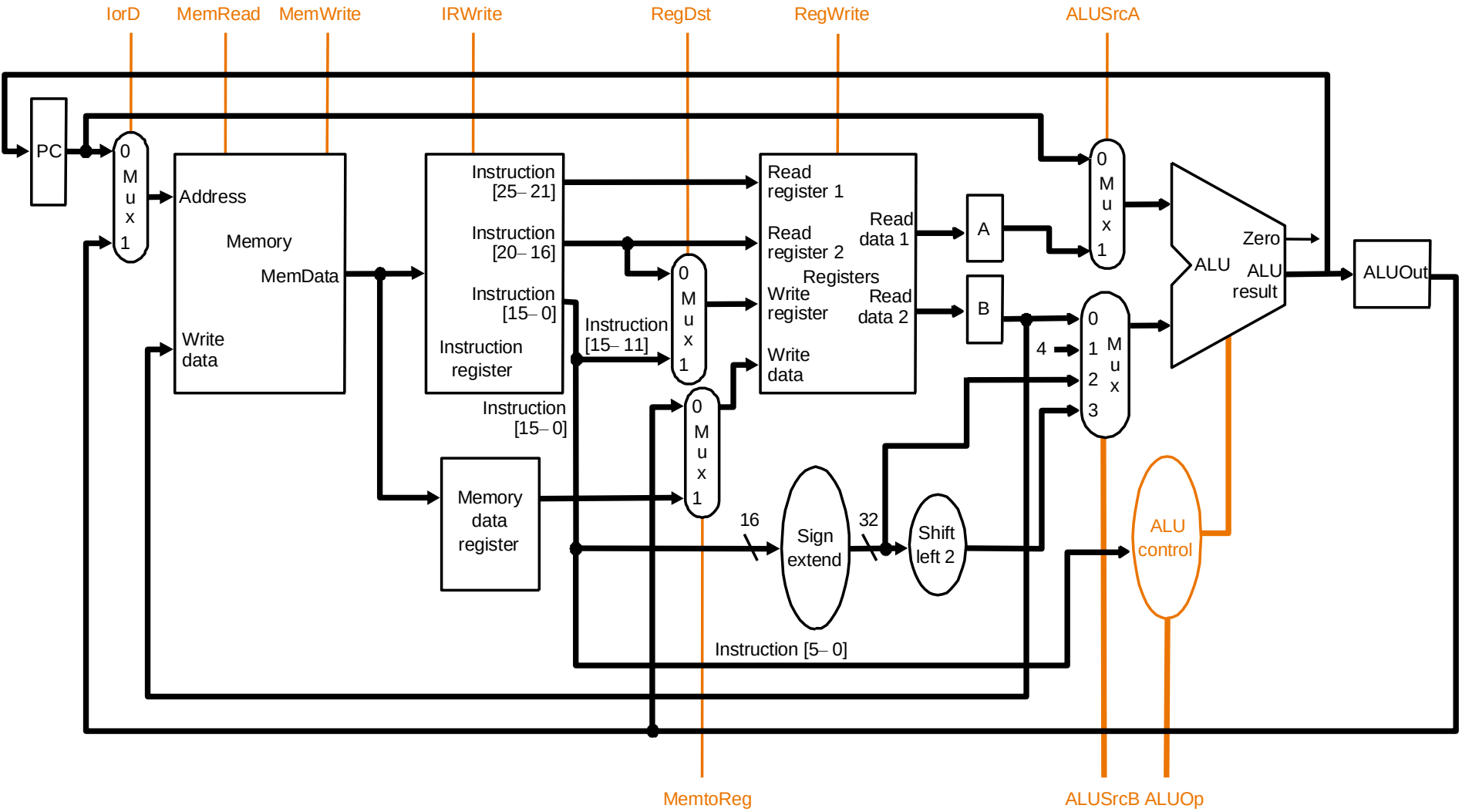
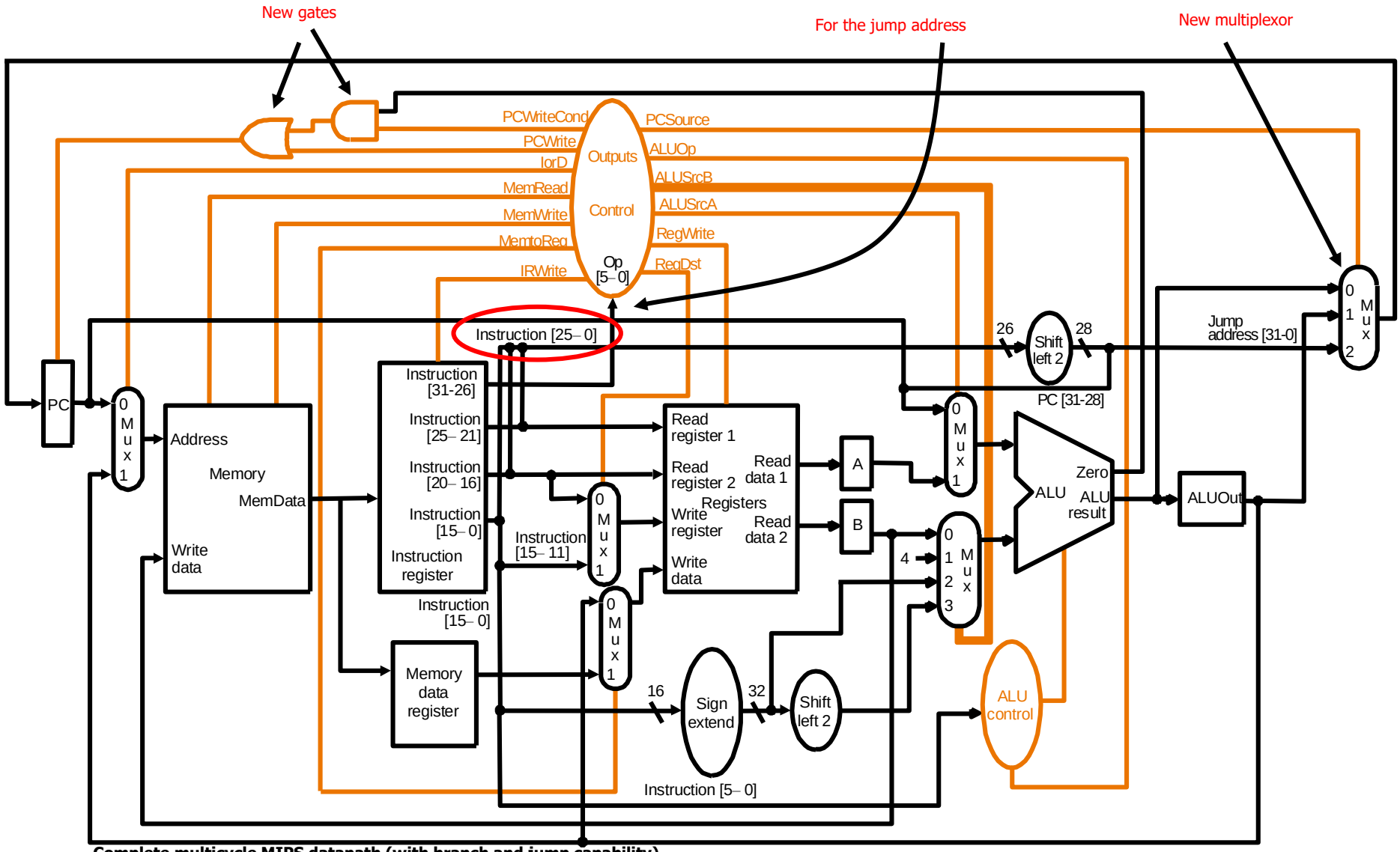
Multicycle Datapath with Control I



… with control lines and the ALU control block added – *not all* control lines are shown

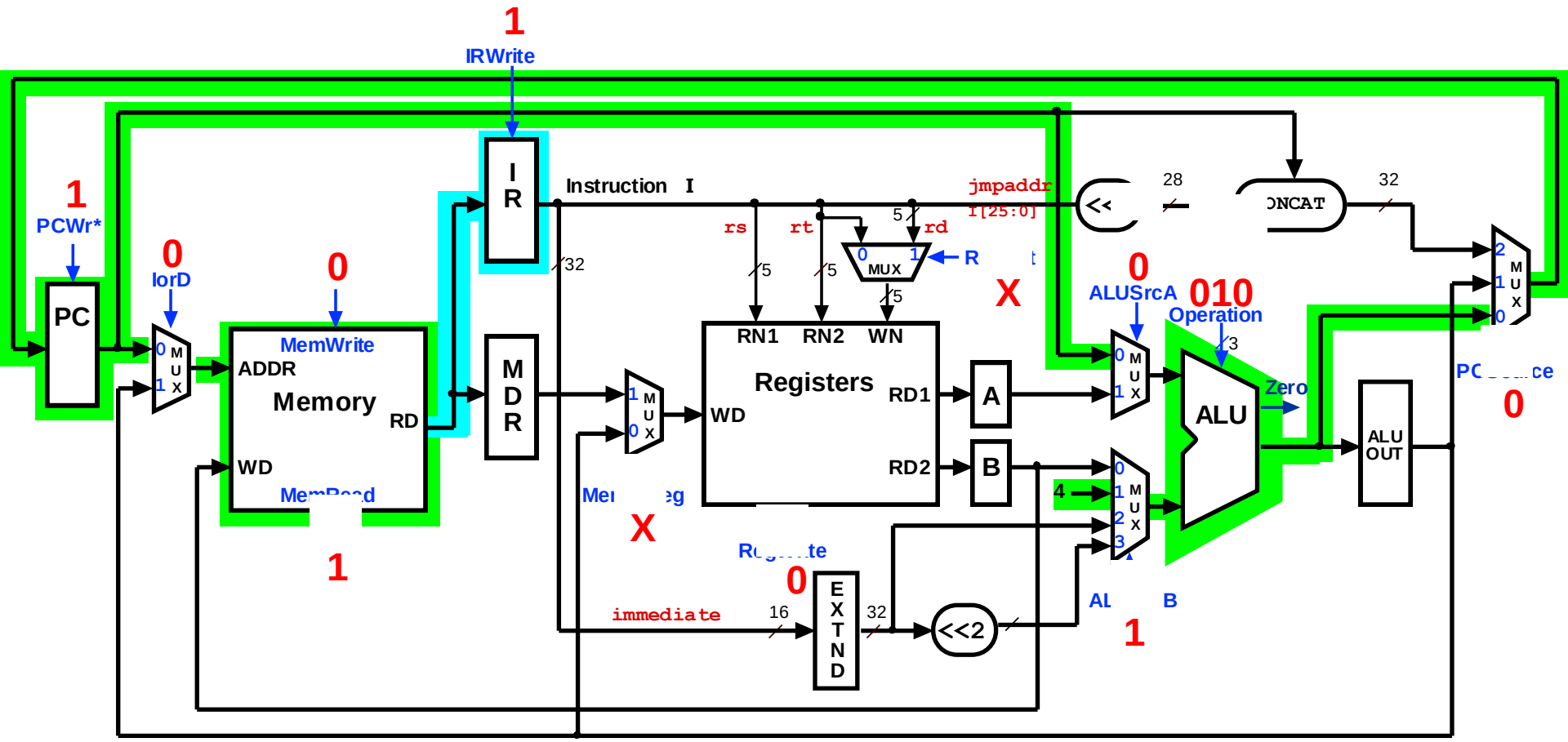
Multicycle Datapath with Control II



Complete multicycle MIPS datapath (with branch and jump capability) and showing the main control block and all control lines

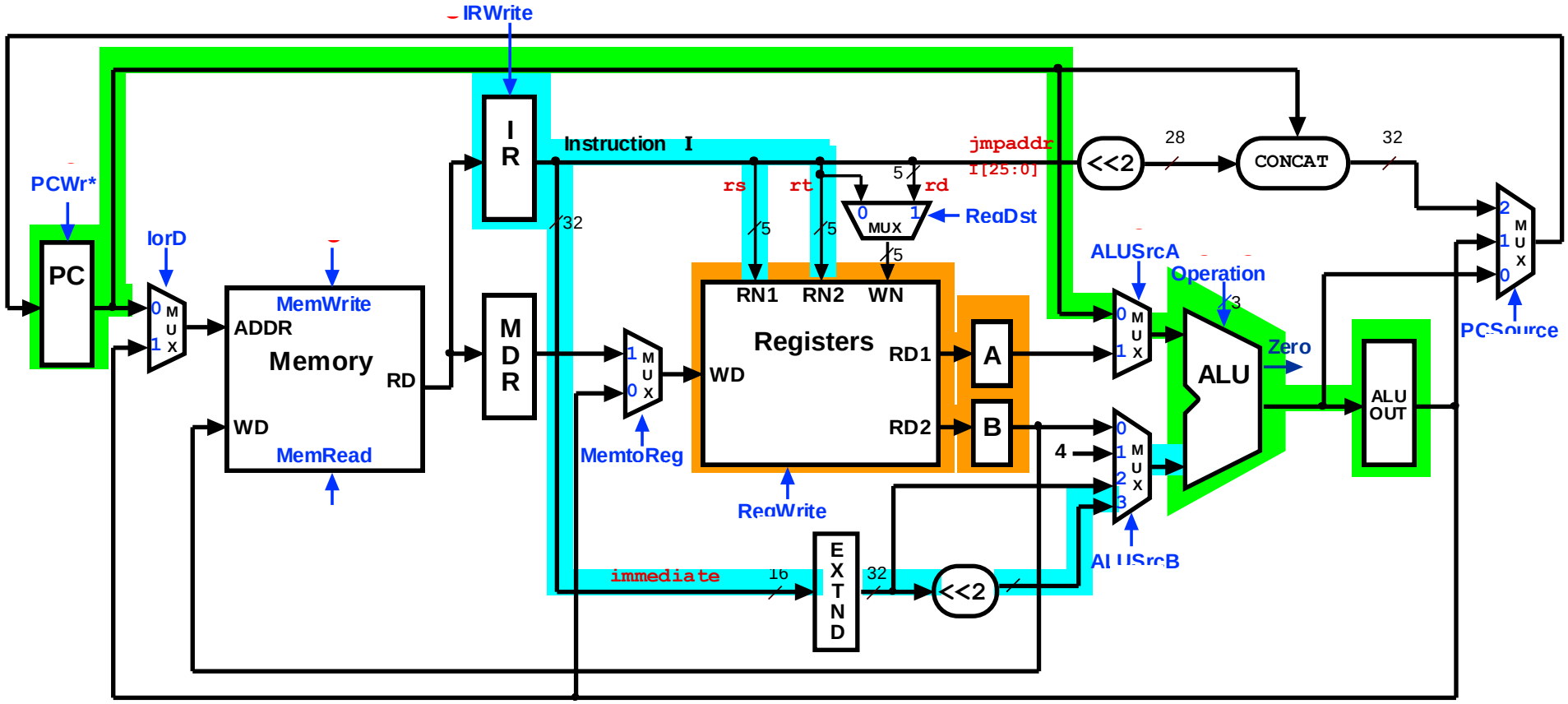
Multicycle Control Step (1): Fetch

IR = Memory[PC];
PC = PC + 4;

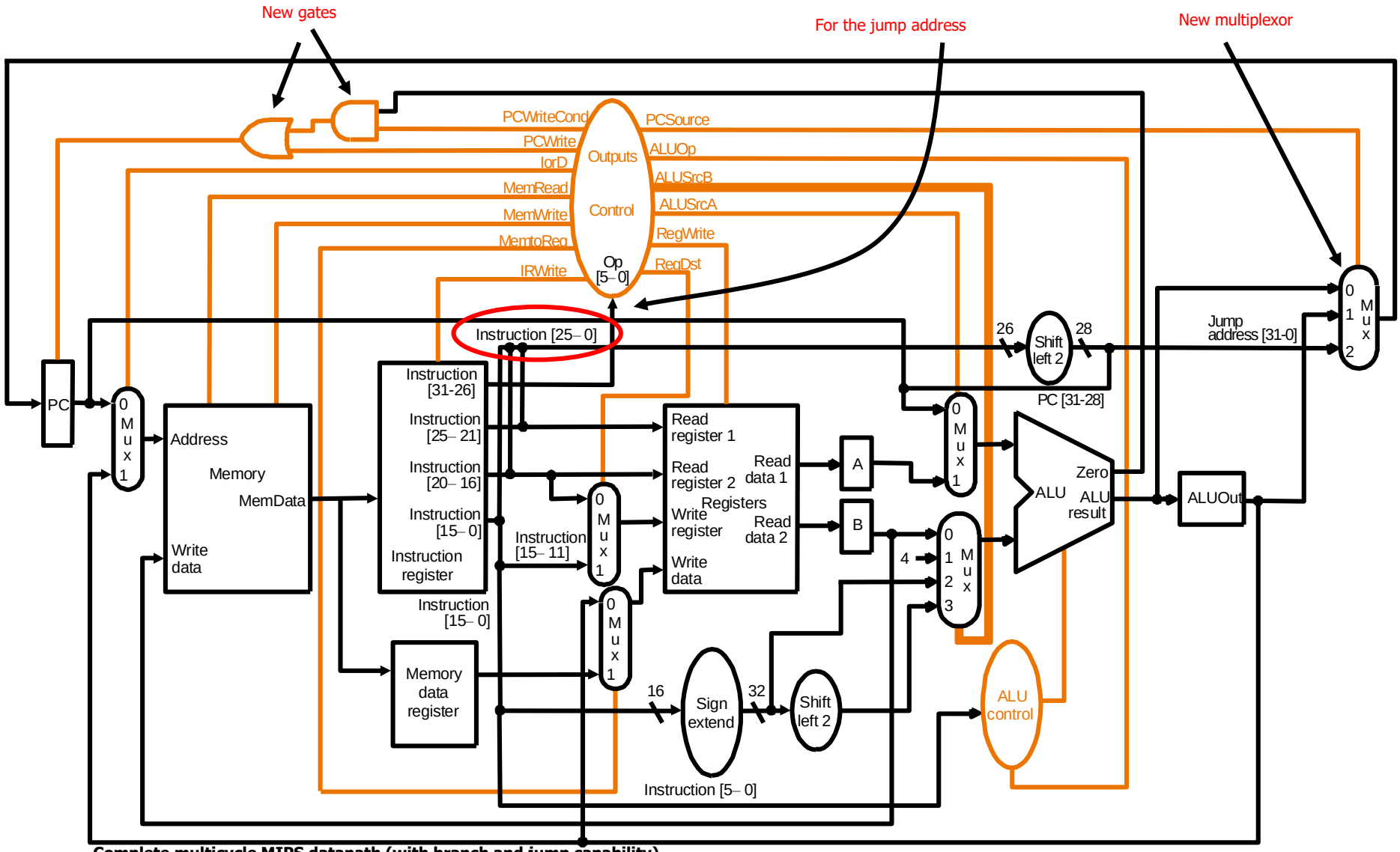


Multicycle Control Step (2): Instruction Decode & Register Fetch

A = Reg[IR[25-21]]; (A = Reg[rs])
B = Reg[IR[20-15]]; (B = Reg[rt])
ALUOut = (PC + sign-extend(IR[15-0]) << 2);



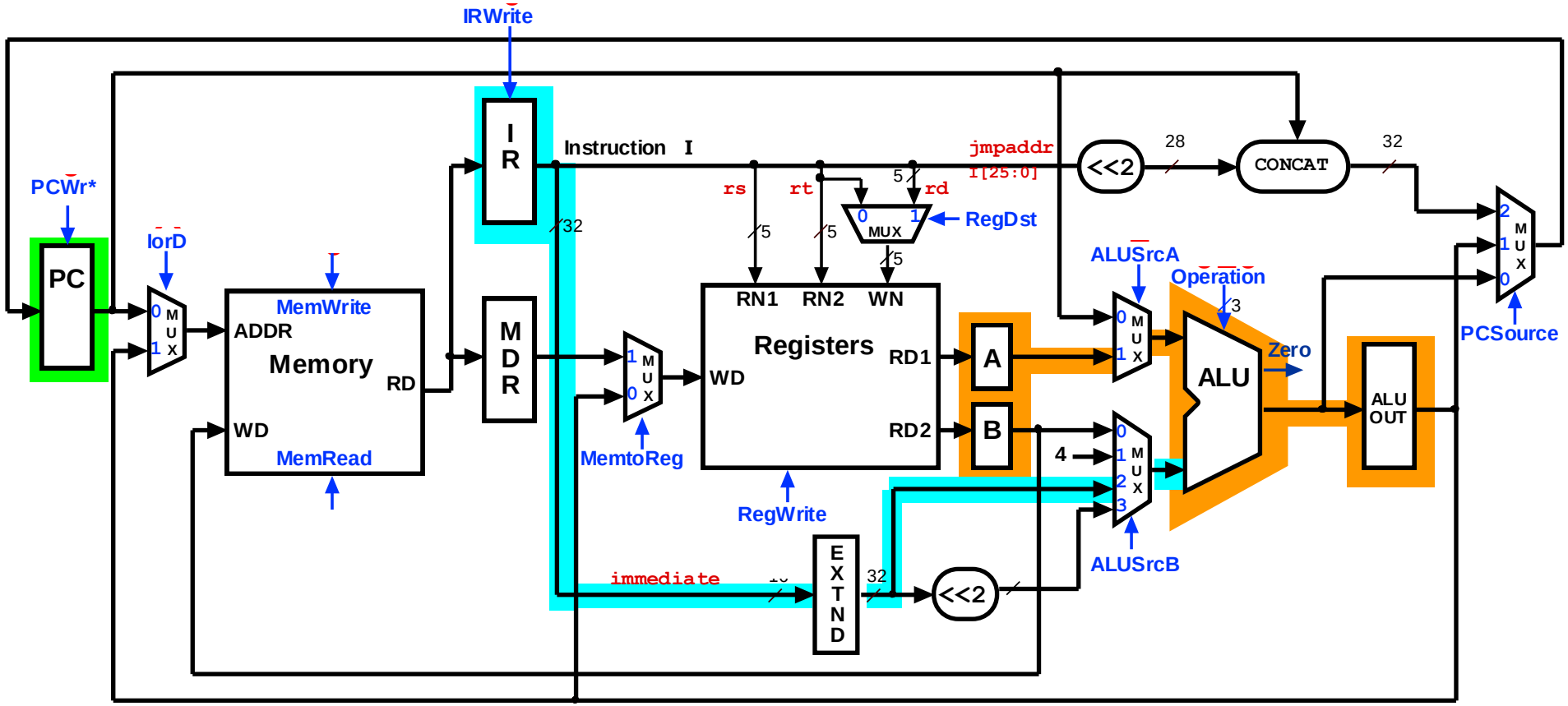
Multicycle Datapath with Control II



Complete multicycle MIPS datapath (with branch and jump capability) and showing the main control block and all control lines

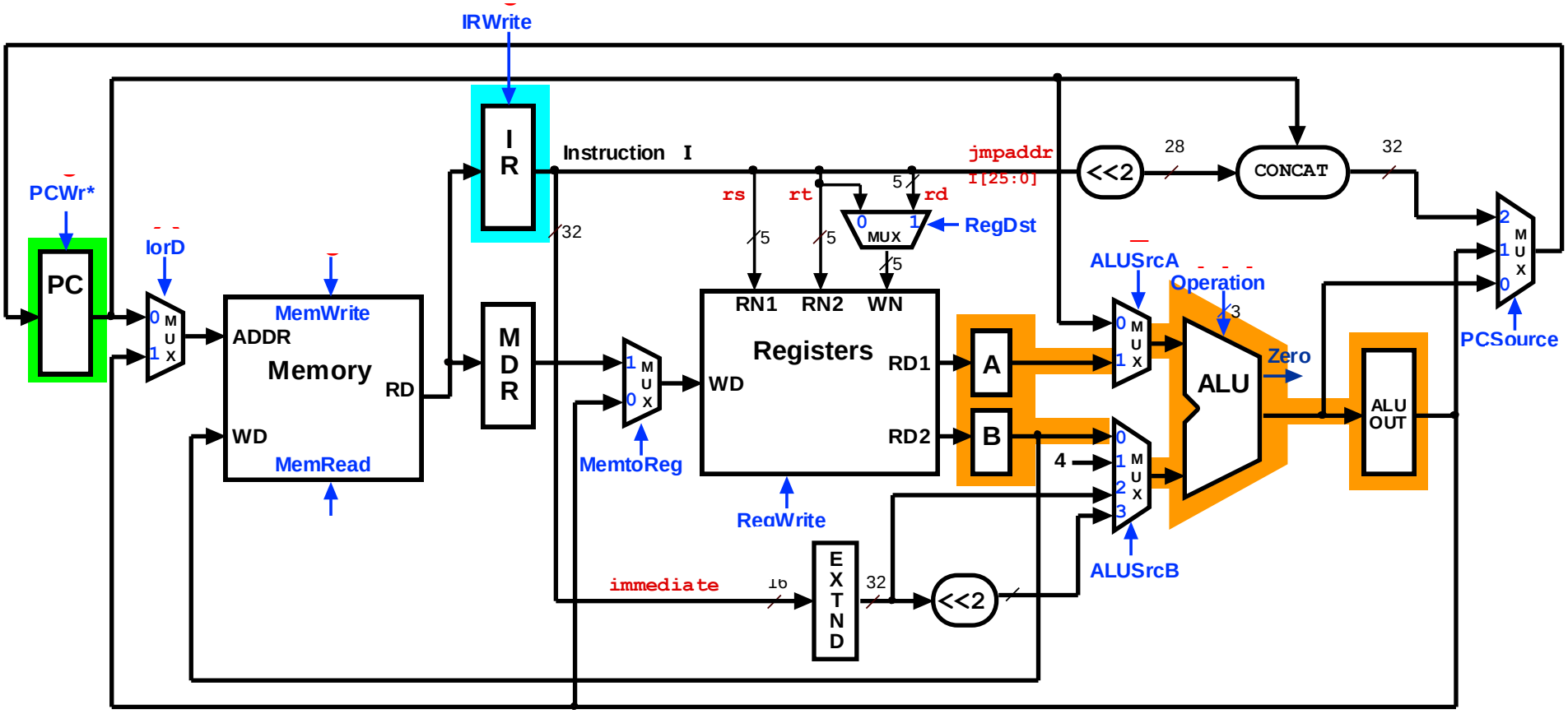
Multicycle Control Step (3): Memory Reference Instructions

ALUOut = A + sign-extend(IR[15-0]);



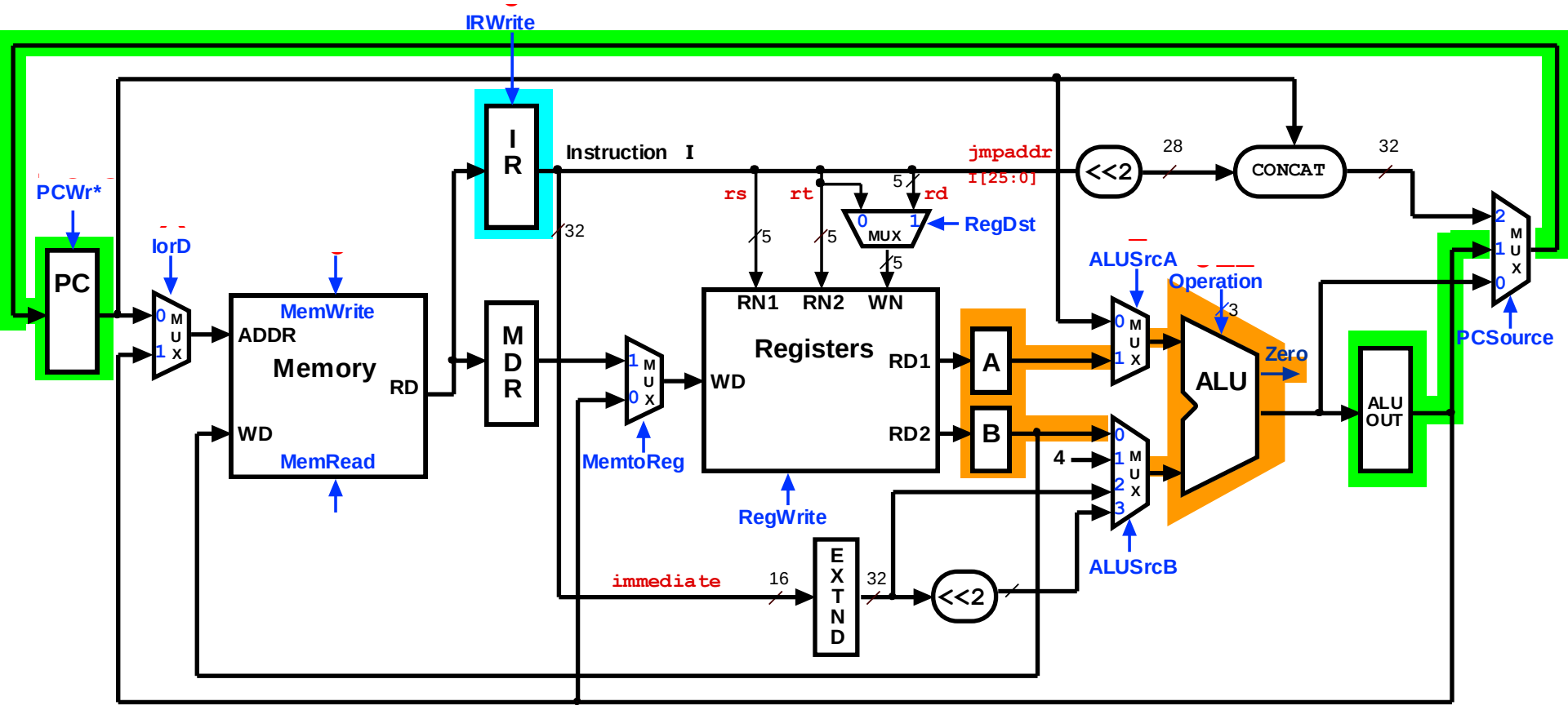
Multicycle Control Step (3): ALU Instruction (R-Type)

ALUOut = A op B;



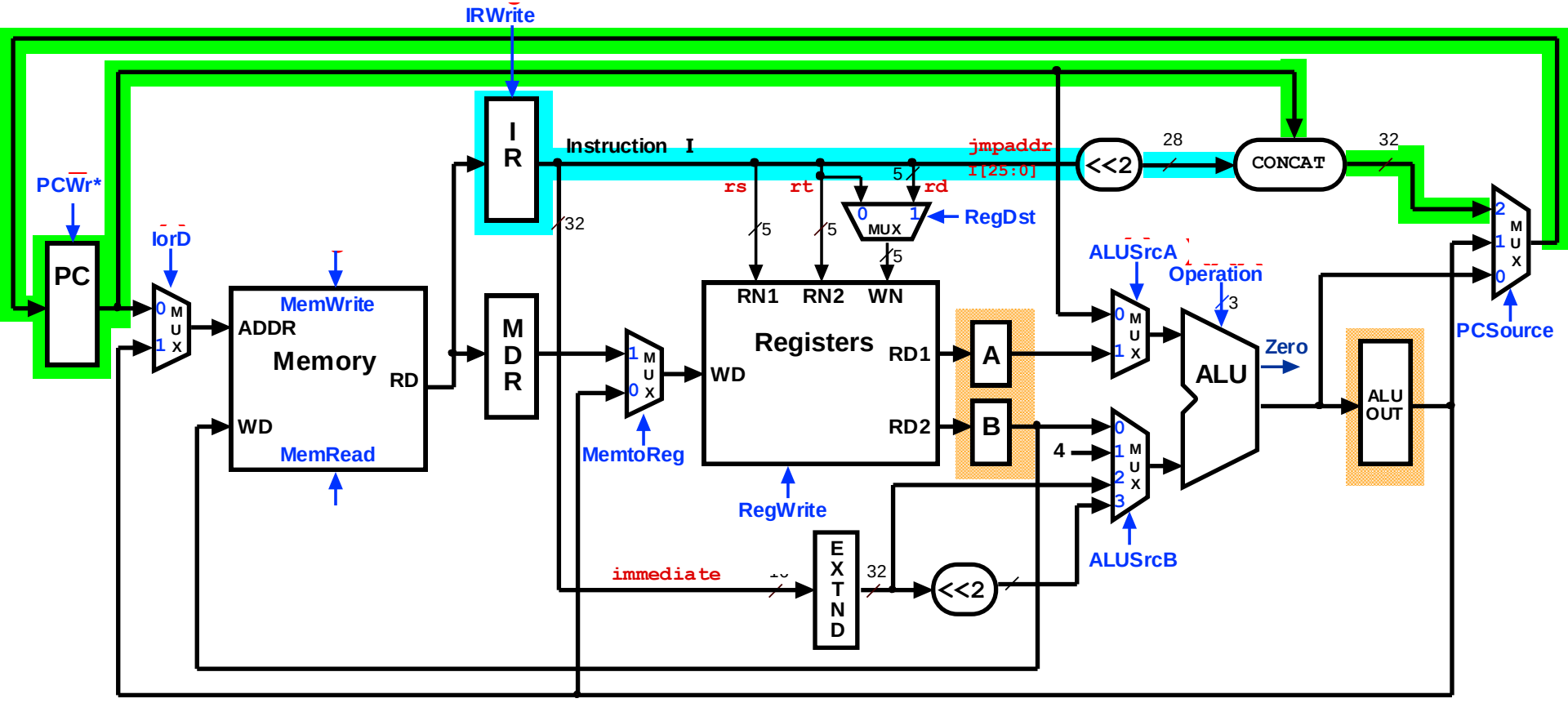
Multicycle Control Step (3): Branch Instructions

if (A == B) PC = ALUOut;



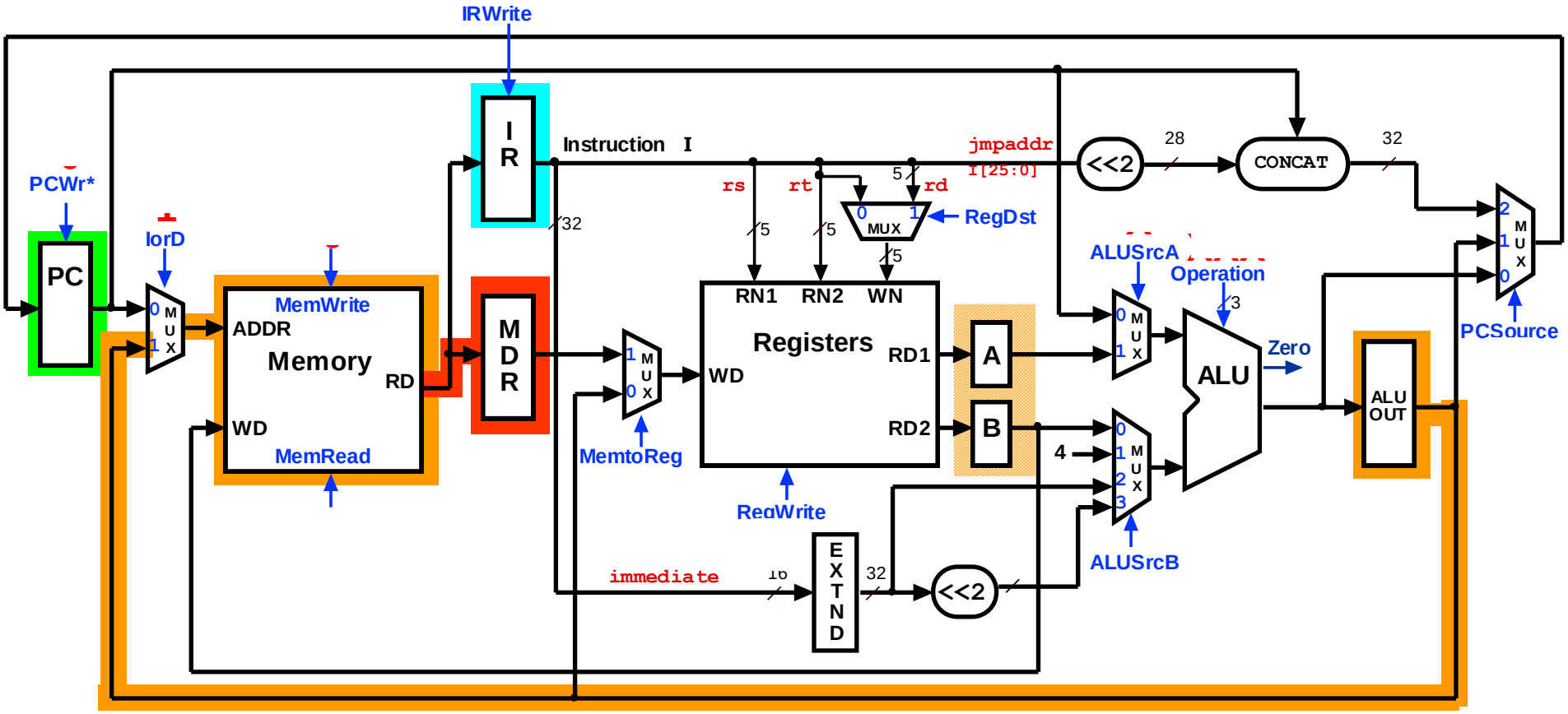
Multicycle Execution Step (3): Jump Instruction

PC = PC[21-28] concat (IR[25-0] << 2);



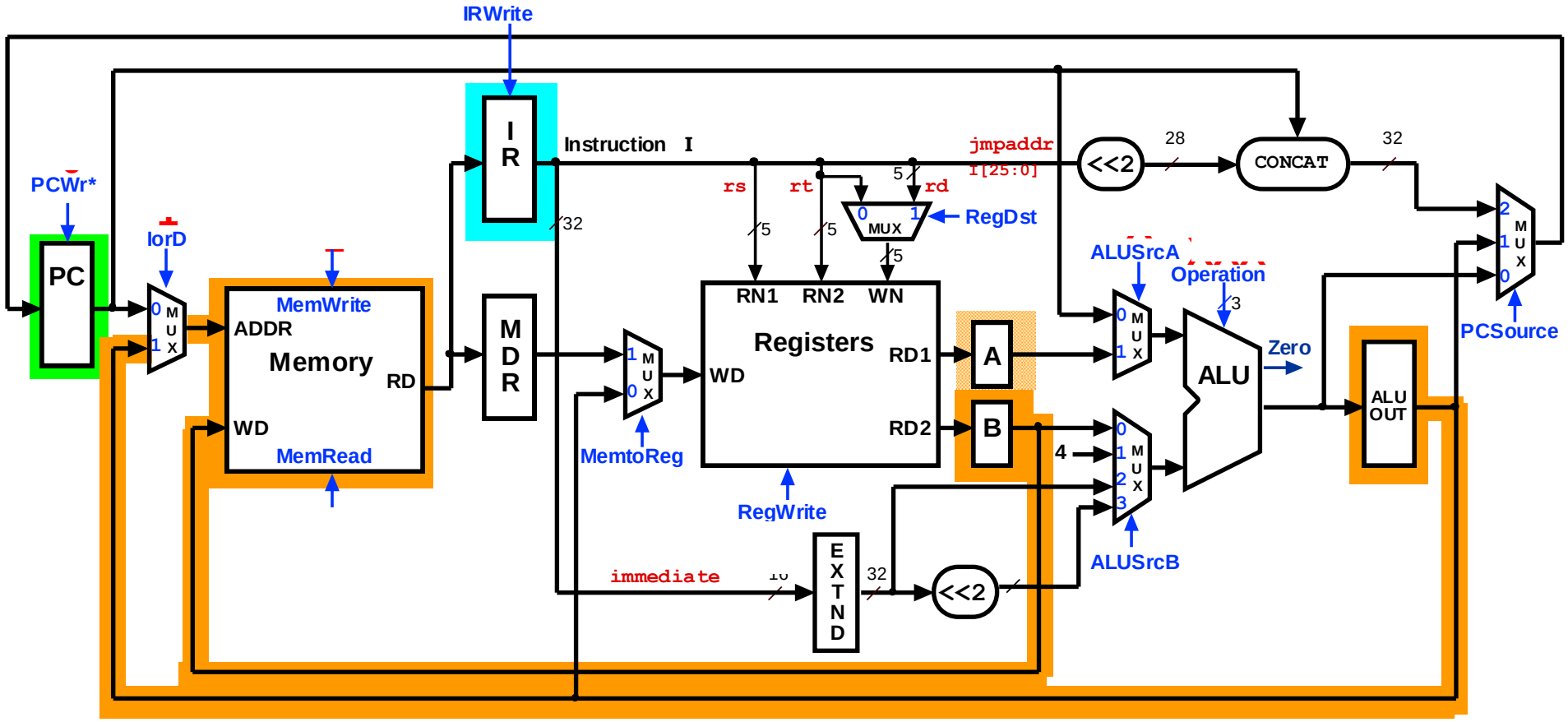
Multicycle Control Step (4): Memory Access - Read (`lw`)

MDR = Memory[ALUOut];



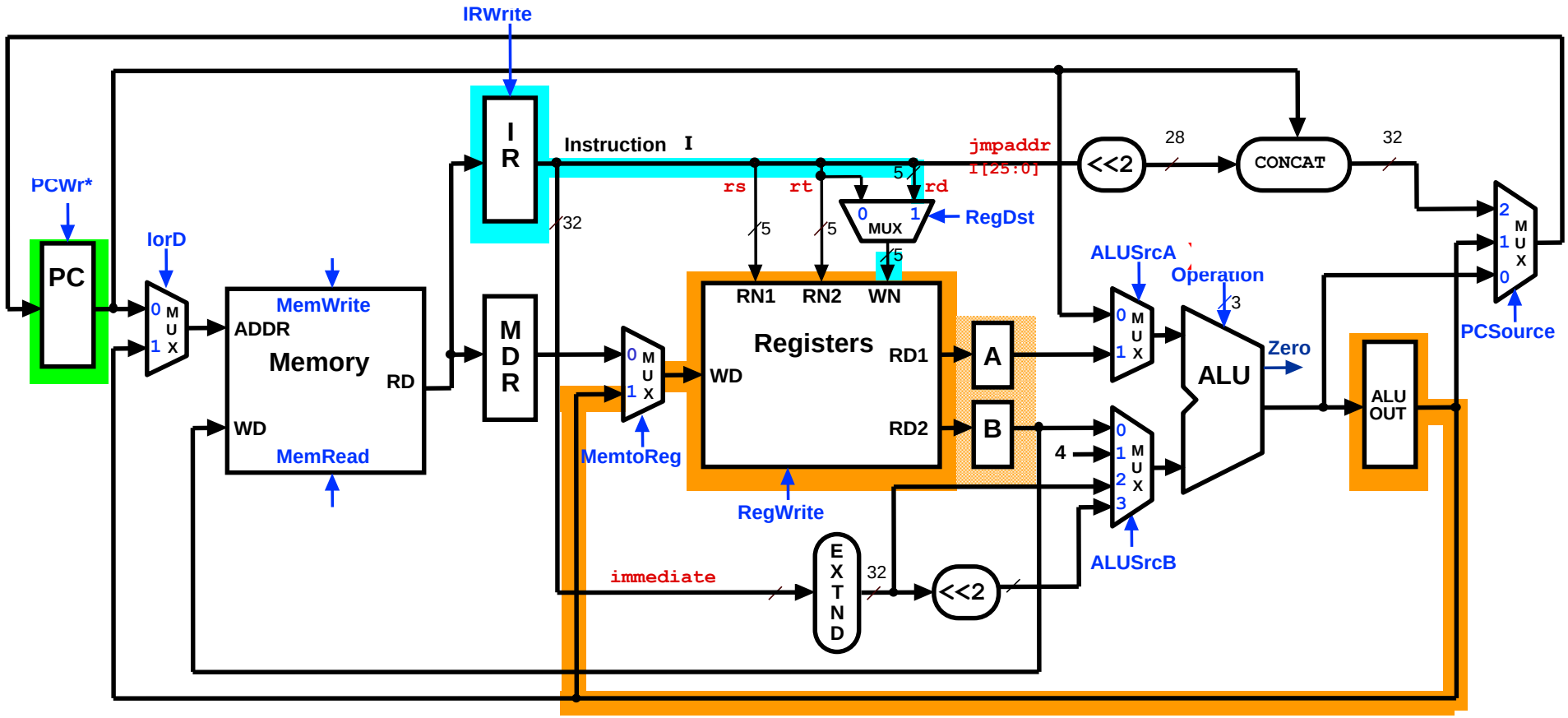
Multicycle Execution Steps (4) Memory Access - Write (sw)

Memory[ALUOut] = B;



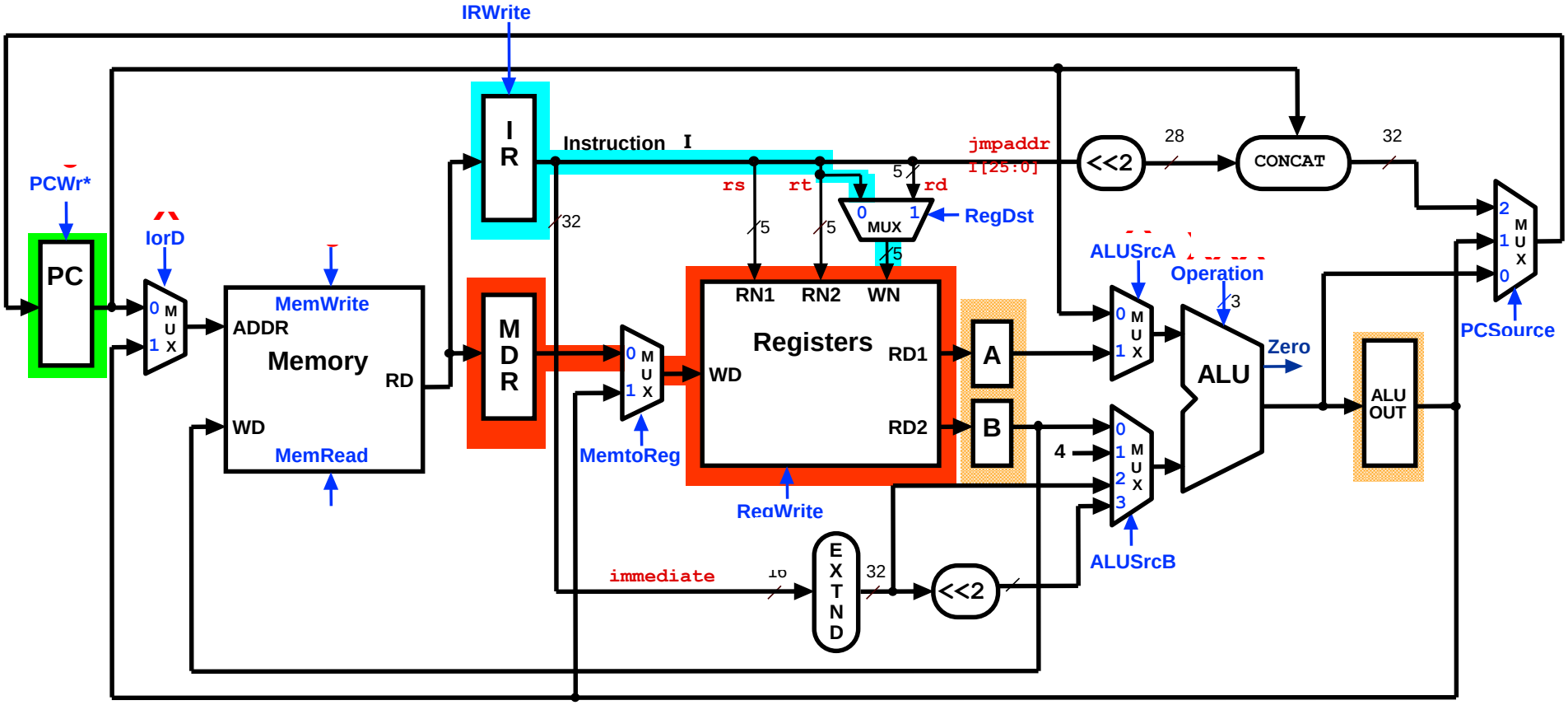
Multicycle Control Step (4): ALU Instruction (R-Type)

(Reg[Rd] = ALUOut)



Multicycle Execution Steps (5)
Memory Read Completion (lw)

Reg[IR[20-16]] = MDR;

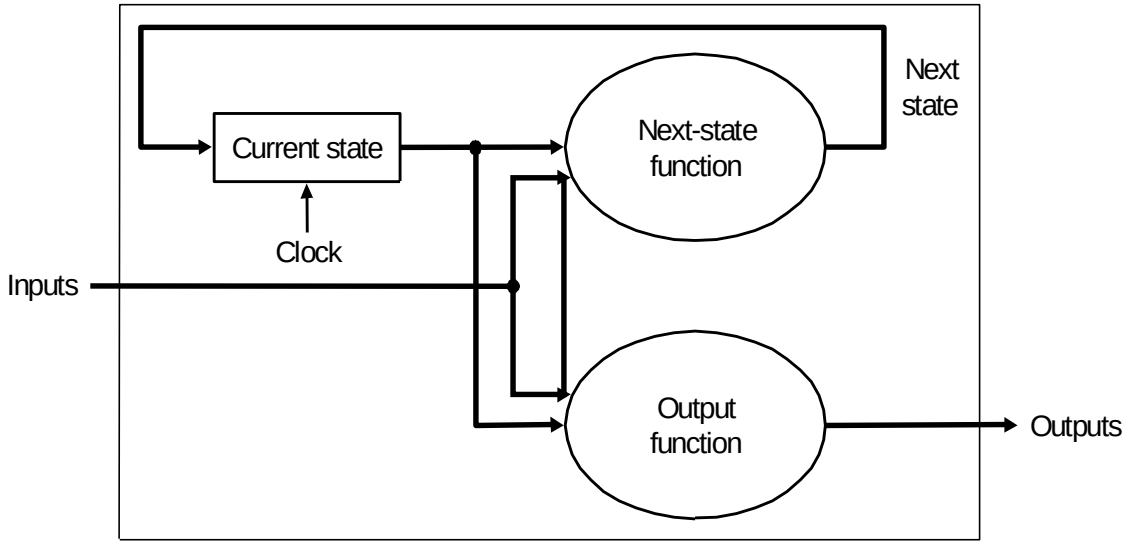


Implementing Control

- Value of control signals is dependent upon:
  - what instruction is being executed
  - which step is being performed
- Use the information we have accumulated to specify a finite state machine
  - specify the finite state machine graphically, or
  - use microprogramming
- Implementation is then derived from the specification

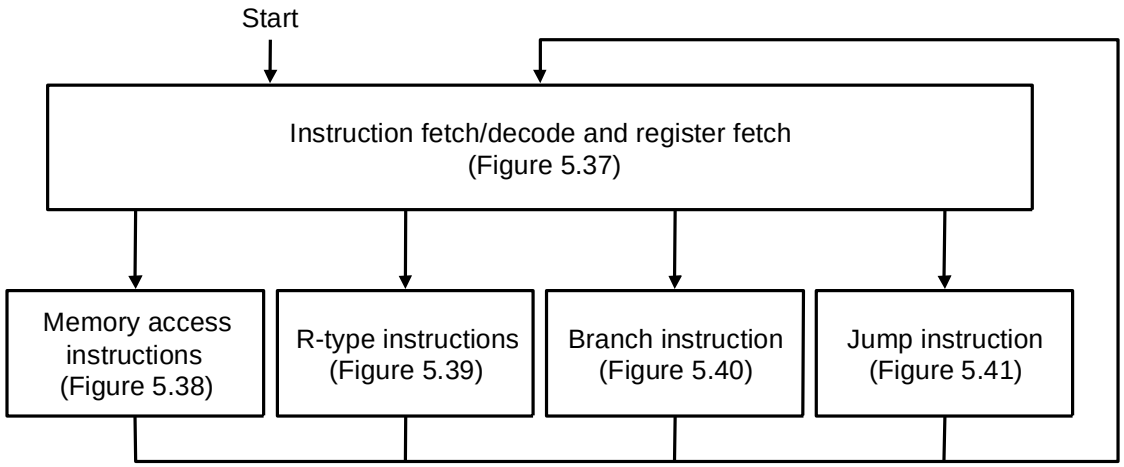
Review: Finite State Machines

- Finite state machines (FSMs):
  - a set of states and
  - next state function, determined by current state and the input
  - output function, determined by current state and possibly input



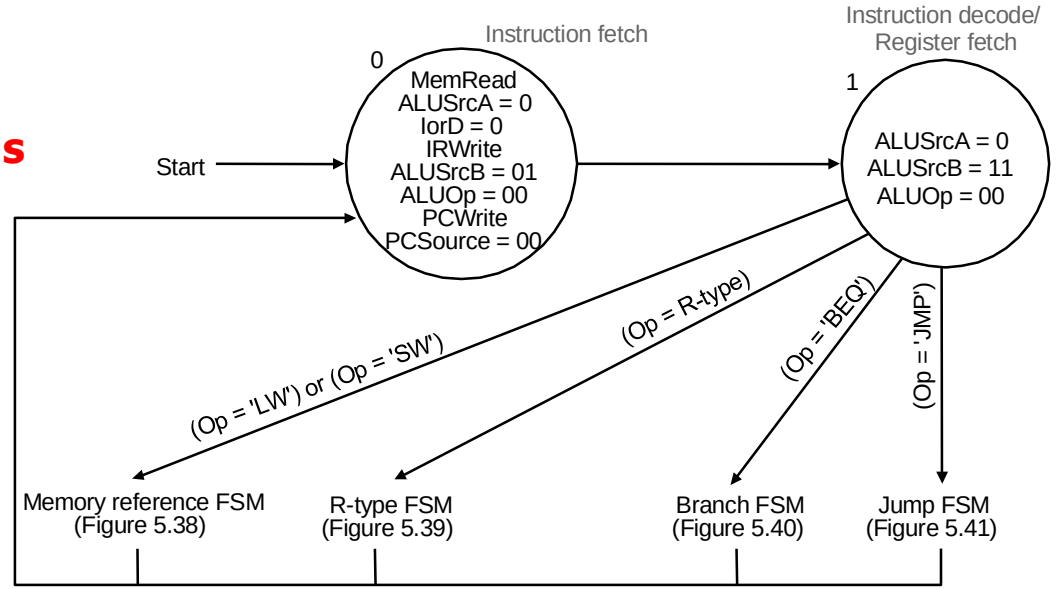
  - We'll use a *Moore machine* – output based *only* on current state

FSM Control: High-level View



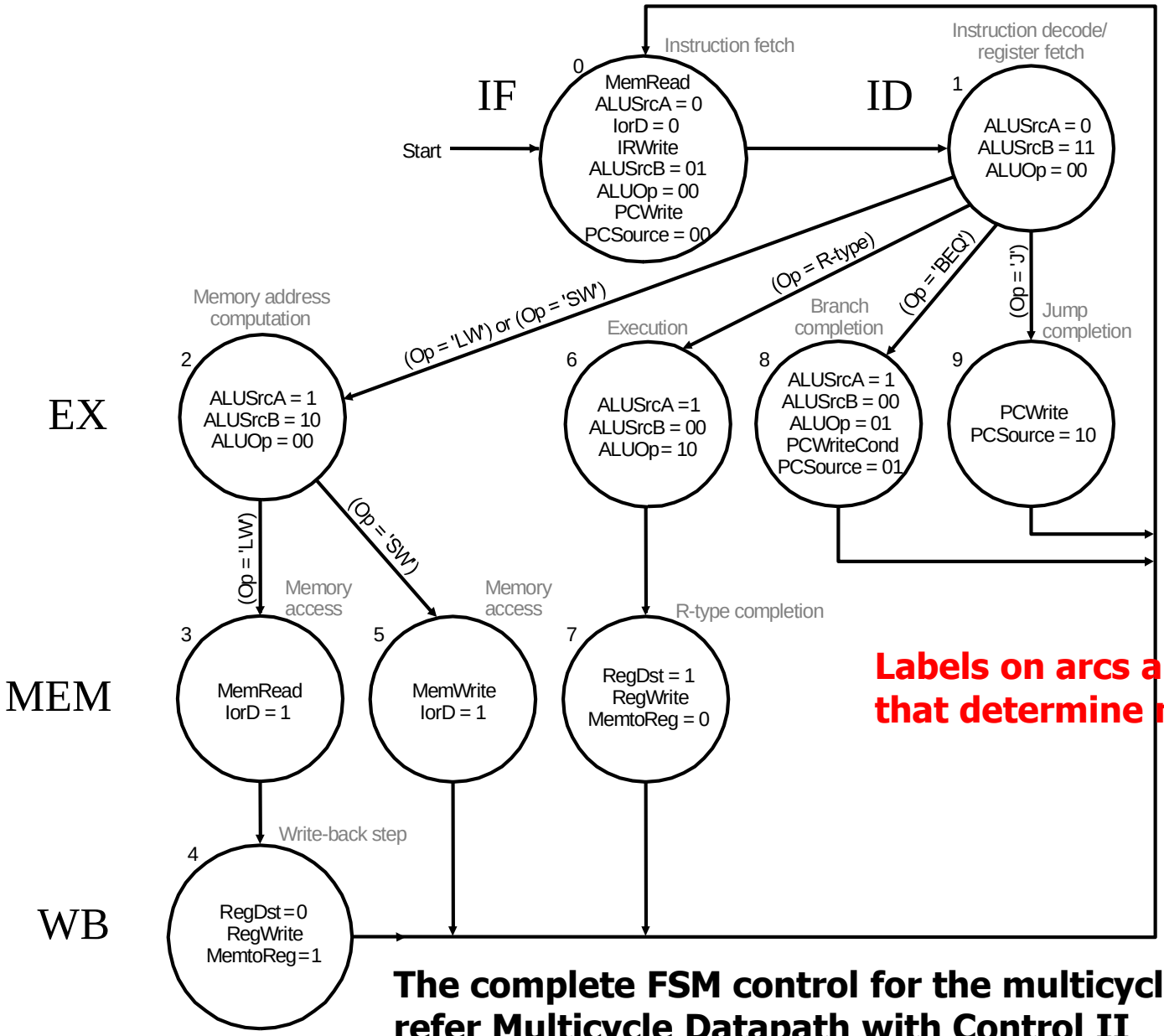
High-level view of FSM control

Asserted signals shown inside state circles



Instruction fetch and decode steps of every instruction is identical

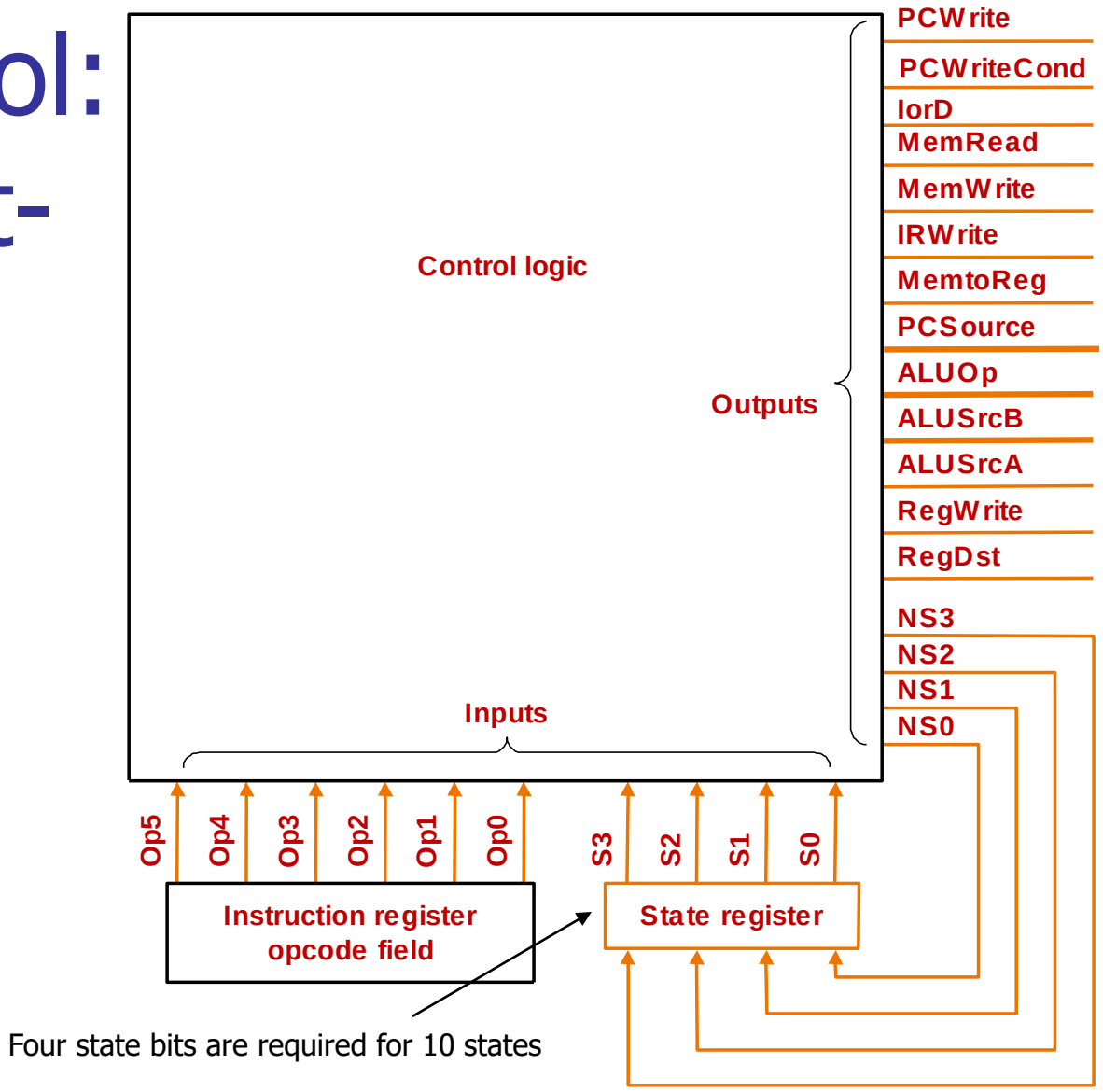
FSM Control: Complete View



Labels on arcs are conditions that determine next state

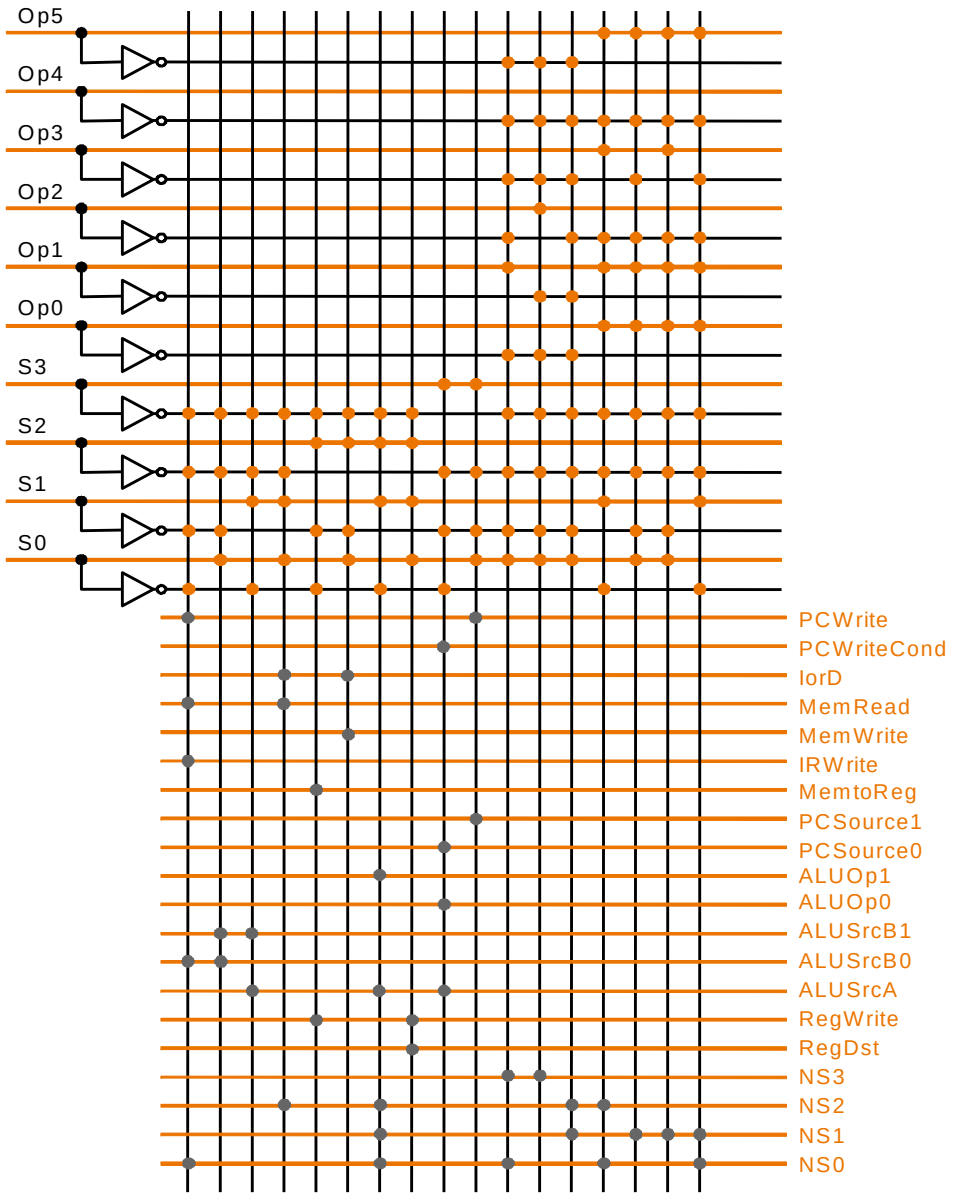
The complete FSM control for the multicycle MIPS datapath: refer Multicycle Datapath with Control II

FSM Control: Implement-ation



High-level view of FSM implementation: inputs to the combinational logic block are the current state number and instruction opcode bits; outputs are the next state number and control signals to be asserted for the current state

FSM Control: PLA Implem-entation



Upper half is the AND plane that computes all the products. The products are carried to the lower OR plane by the vertical lines. The sum terms for each output is given by the corresponding horizontal line

E.g., IorD = $\overline{S0}.\overline{S1}.S2.S3 + S0.\overline{S1}.S2.\overline{S3}$