CSSE232
Computer Architecture I

Multicycle Control

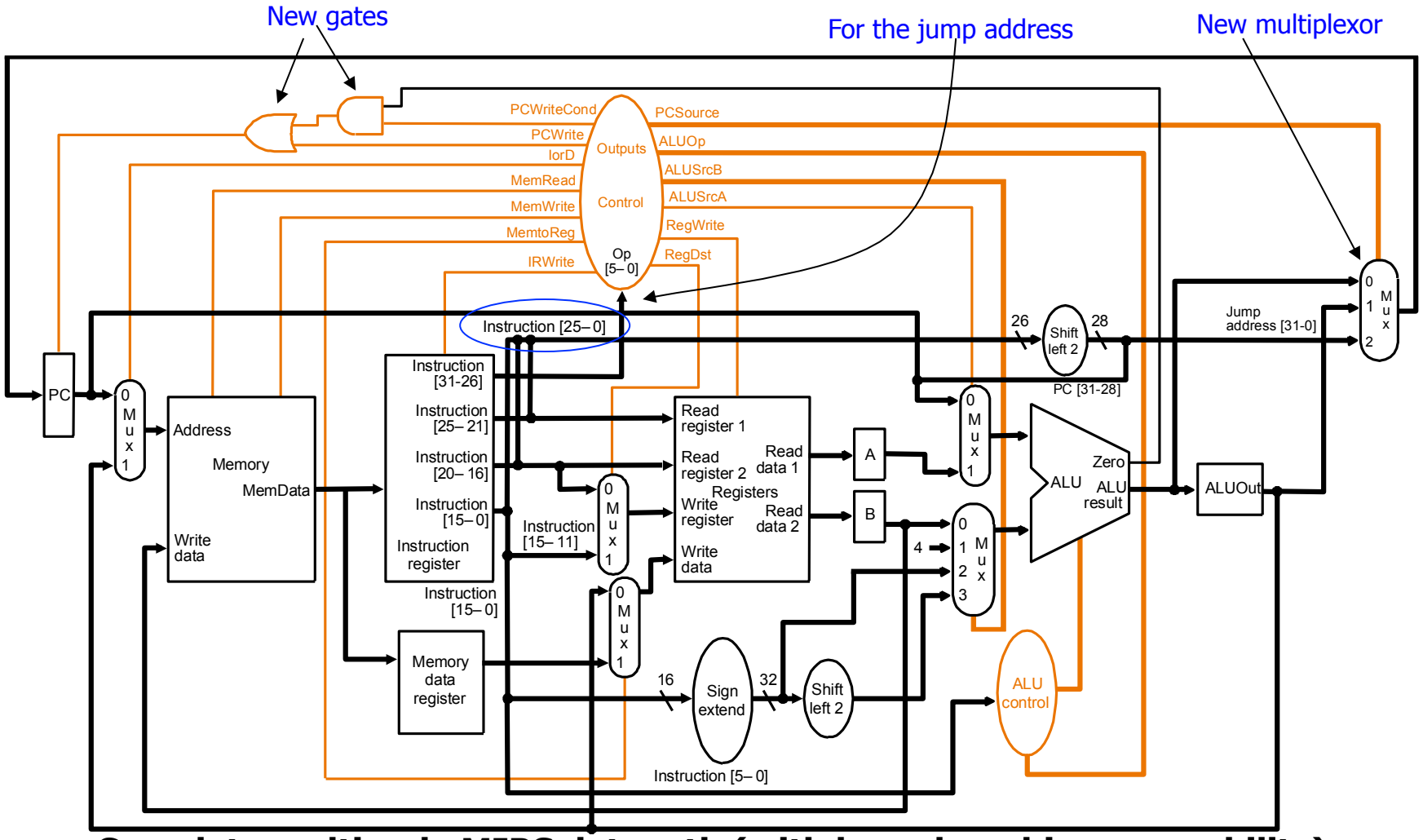
Groups

- GroupA: Megan, Chris, Zach, Michael, Chase
- GroupB: Gary, Heather, Brodie, Philip, Joe
- GroupC: Brian, Abby, Austin, Gordon
- GroupD: Francis, Aler, L.E., Alvin, Matthew

Outline

- Control lines
- Finite state machine
- Project review

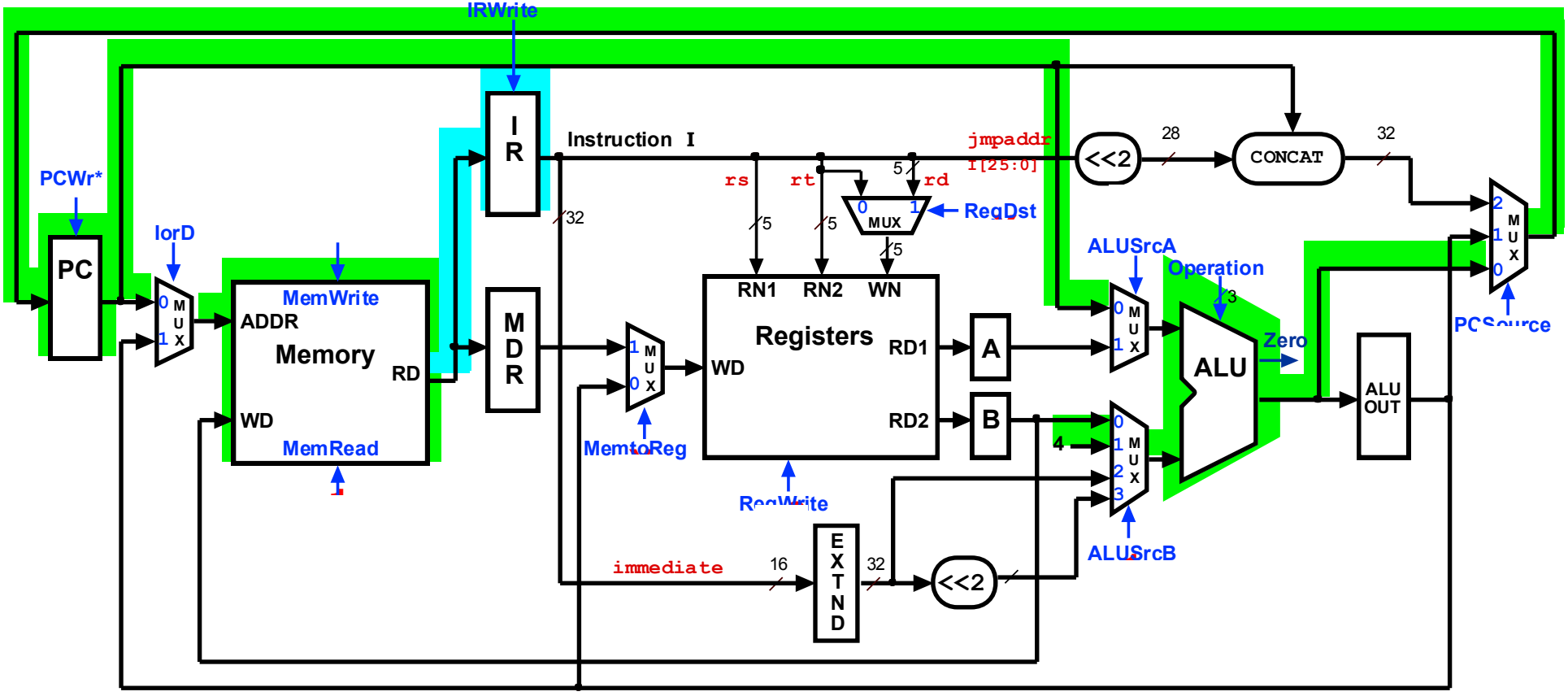
Multicycle Datapath with Control



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

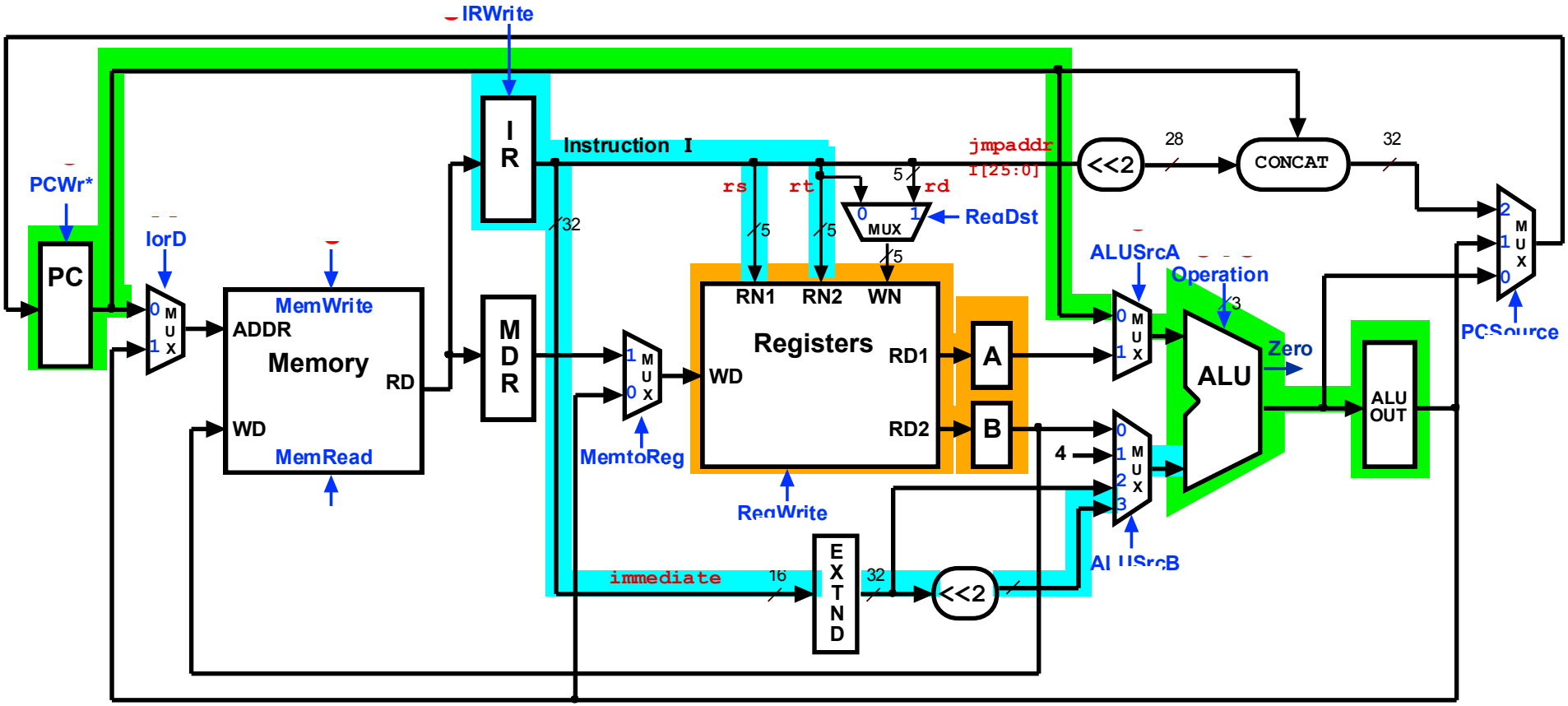
Multicycle Control Step (1): Fetch

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



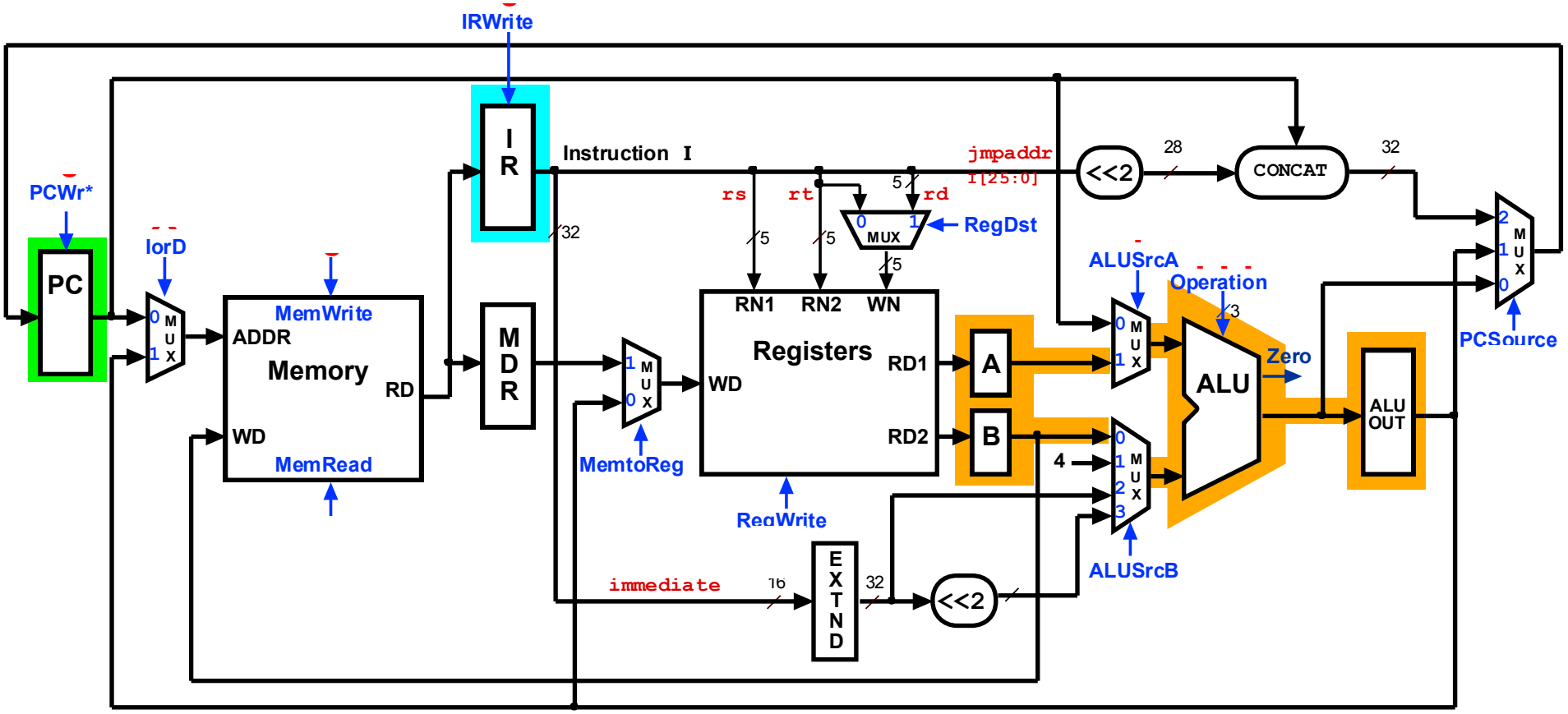
Multicycle Control Step (2): Instruction Decode & Register Fetch

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



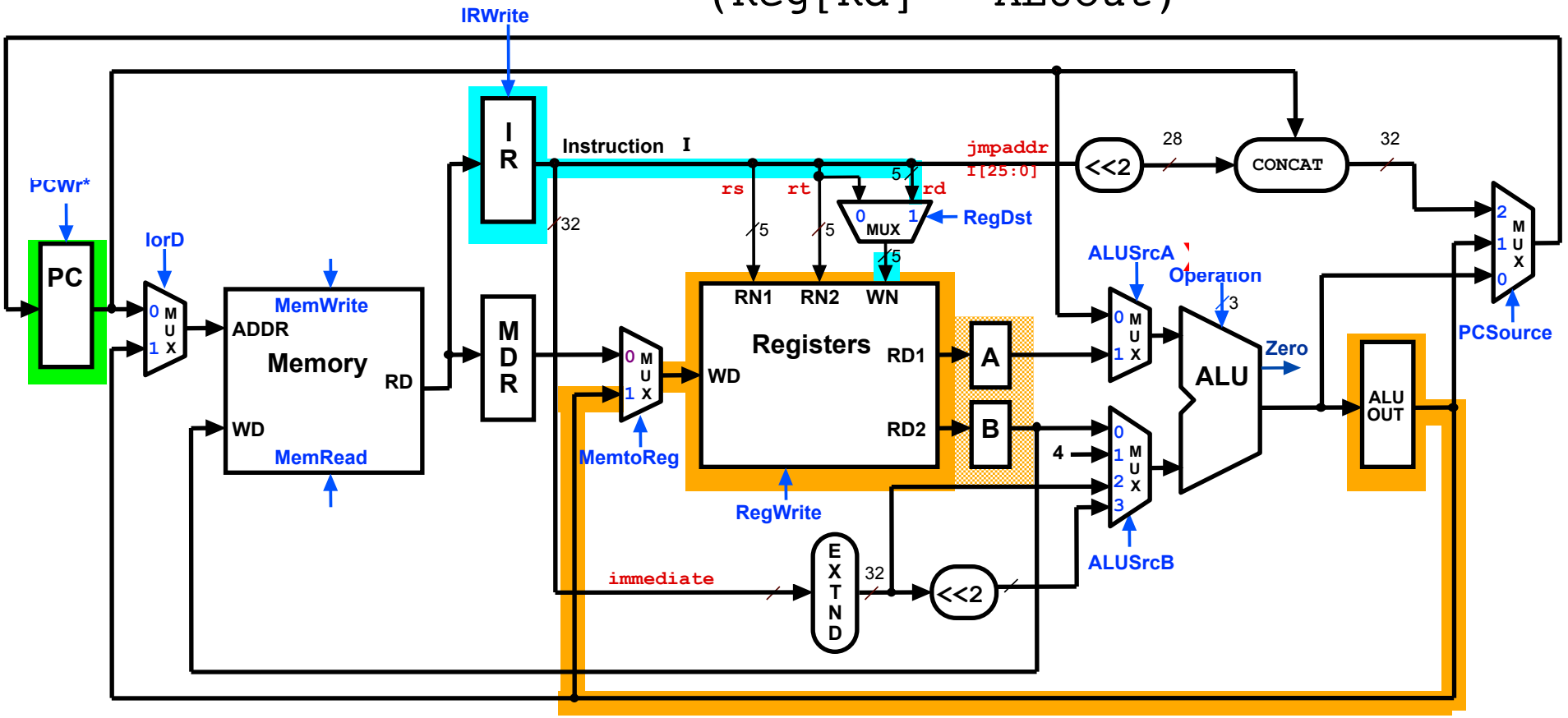
Multicycle Control Step (3): ALU Instruction (R-Type)

```
ALUOut = A op B;
```



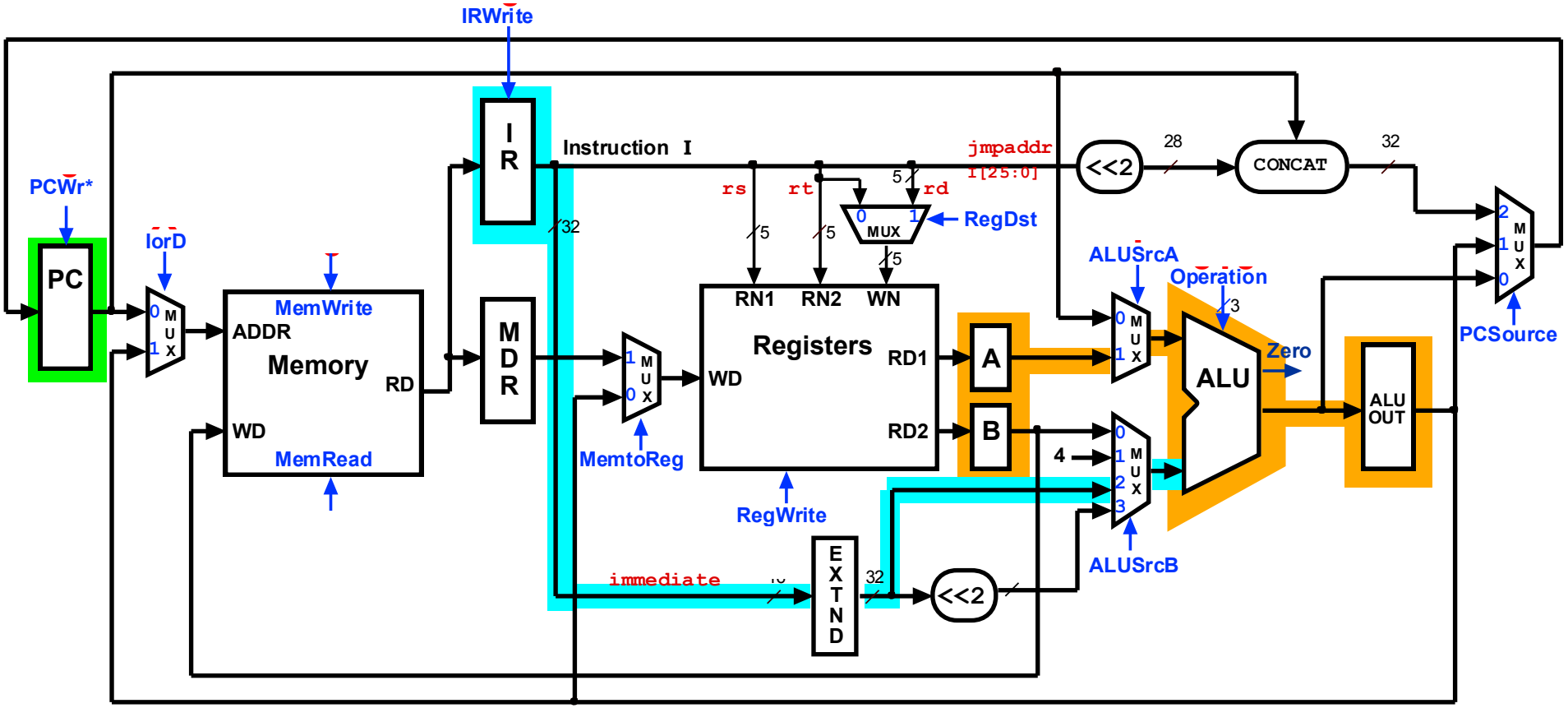
Multicycle Control Step (4): ALU Instruction (R-Type)

```
Reg[IR[15:11]] = ALUOut;
(Reg[Rd] = ALUOut)
```



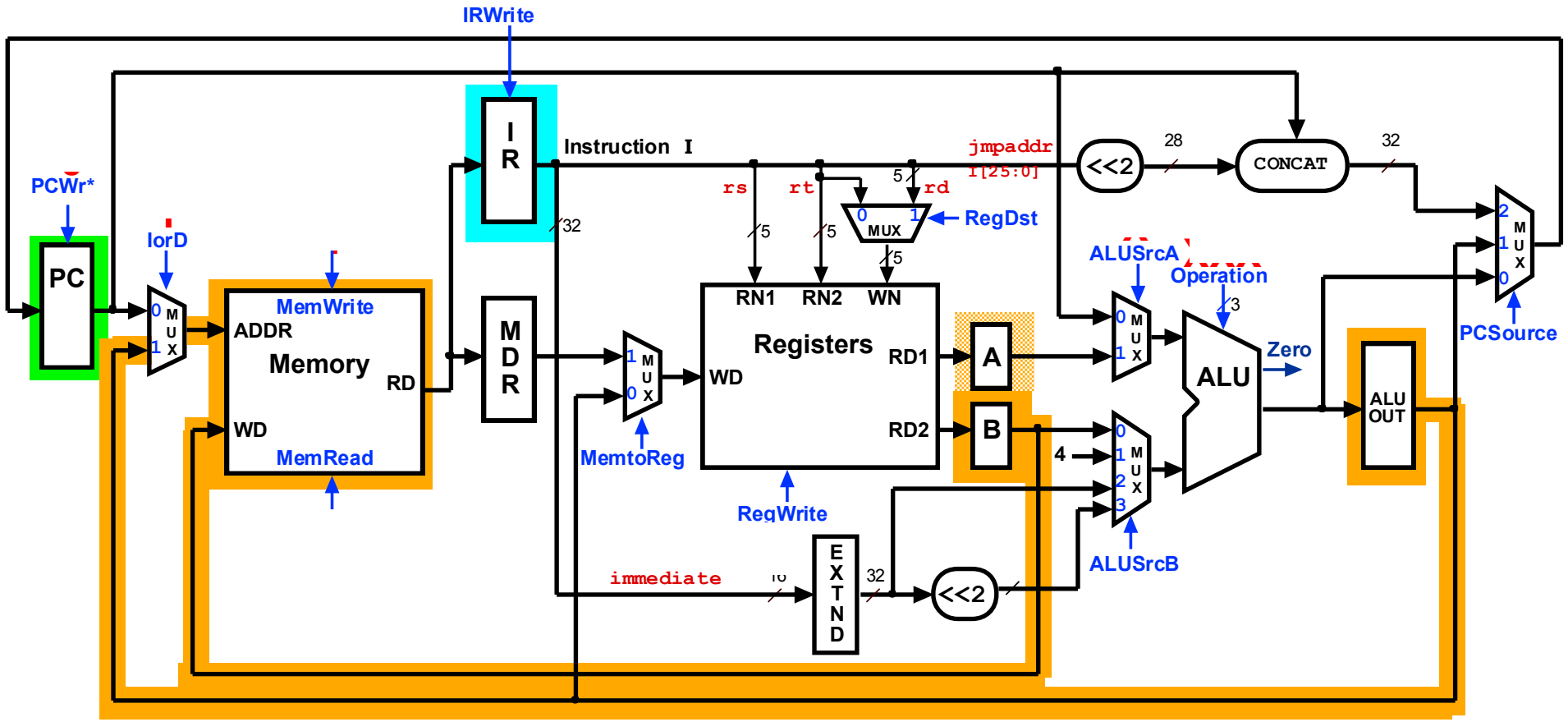
Multicycle Control Step (3): Memory Reference Instructions

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



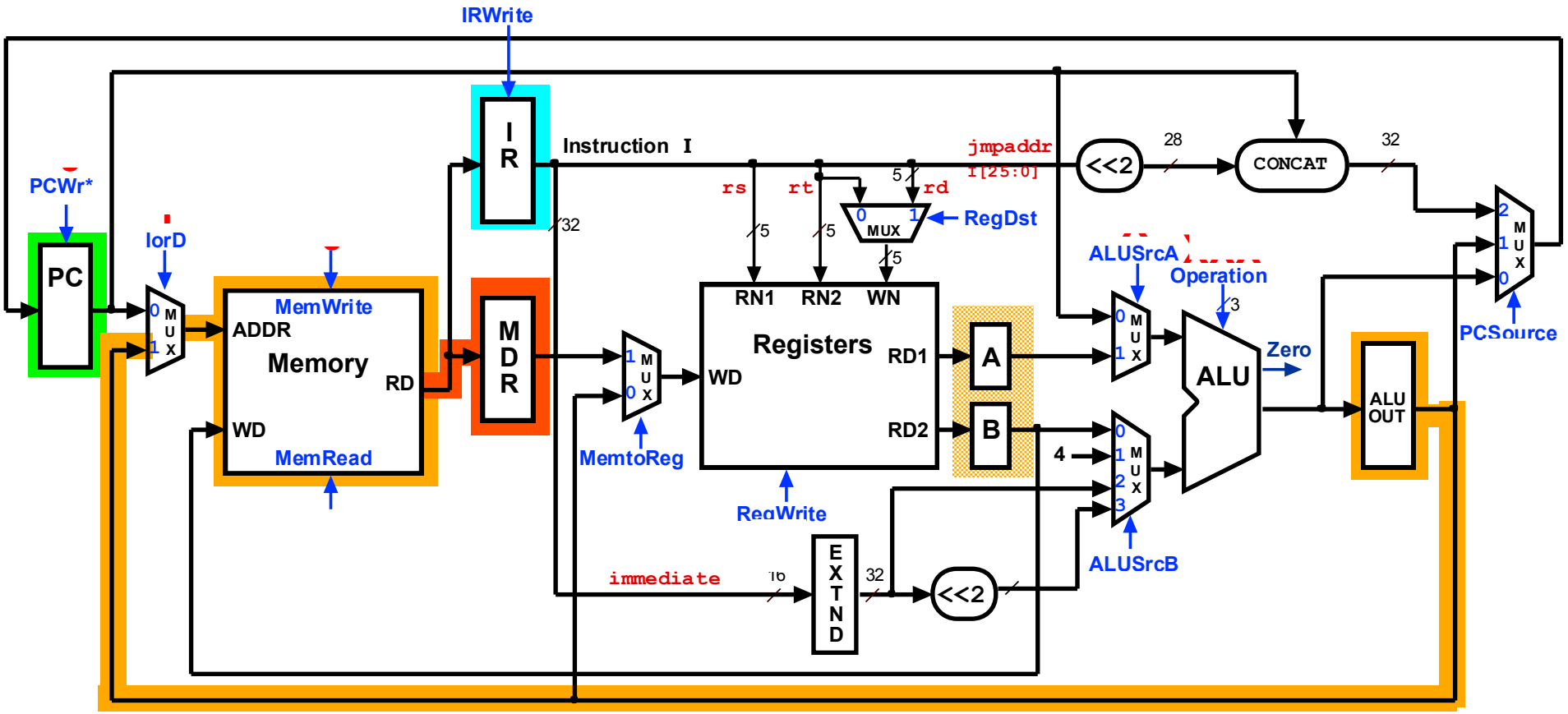
Multicycle Execution Steps (4)
Memory Access - Write (sw)

```
Memory[ALUOut] = B;
```



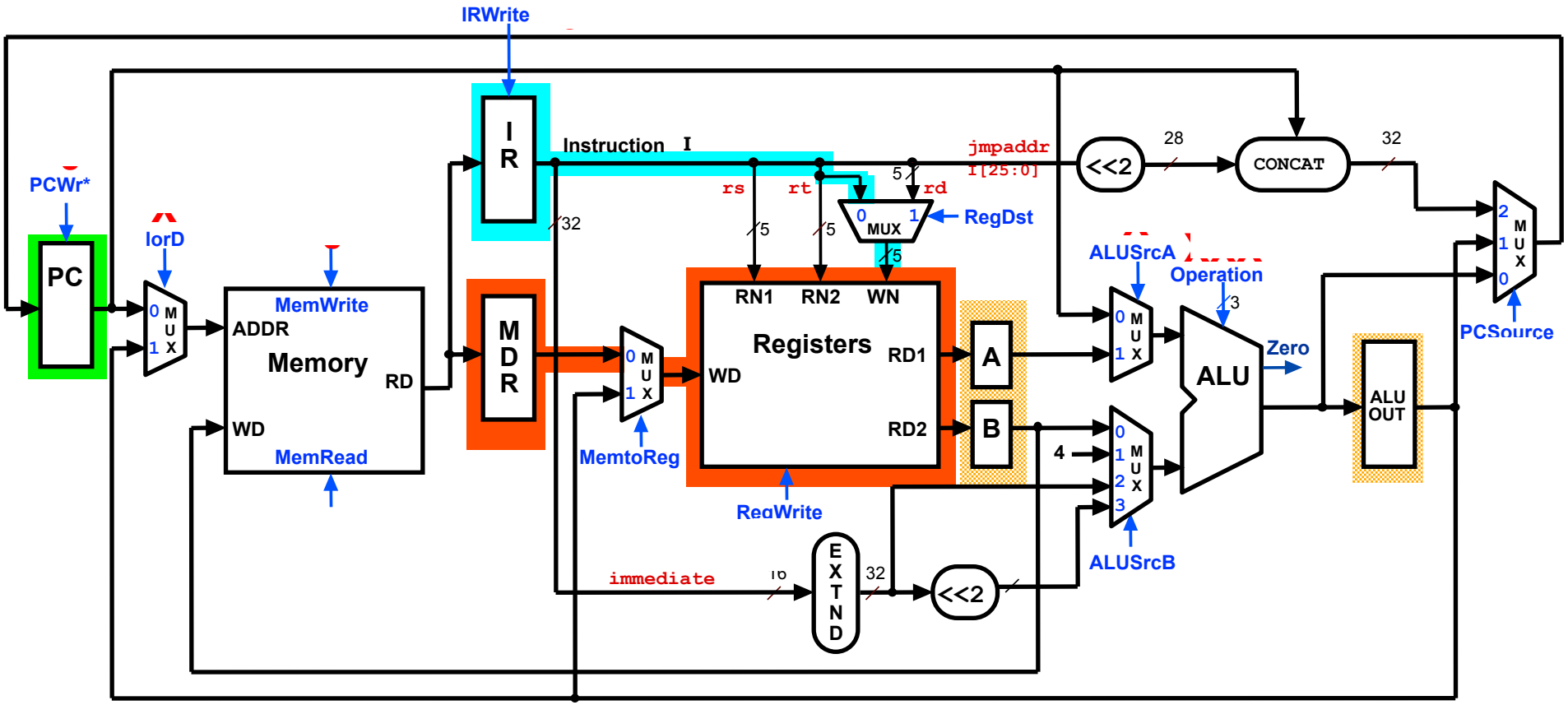
Multicycle Control Step (4): Memory Access - Read (`lw`)

```
MDR = Memory[ALUOut];
```



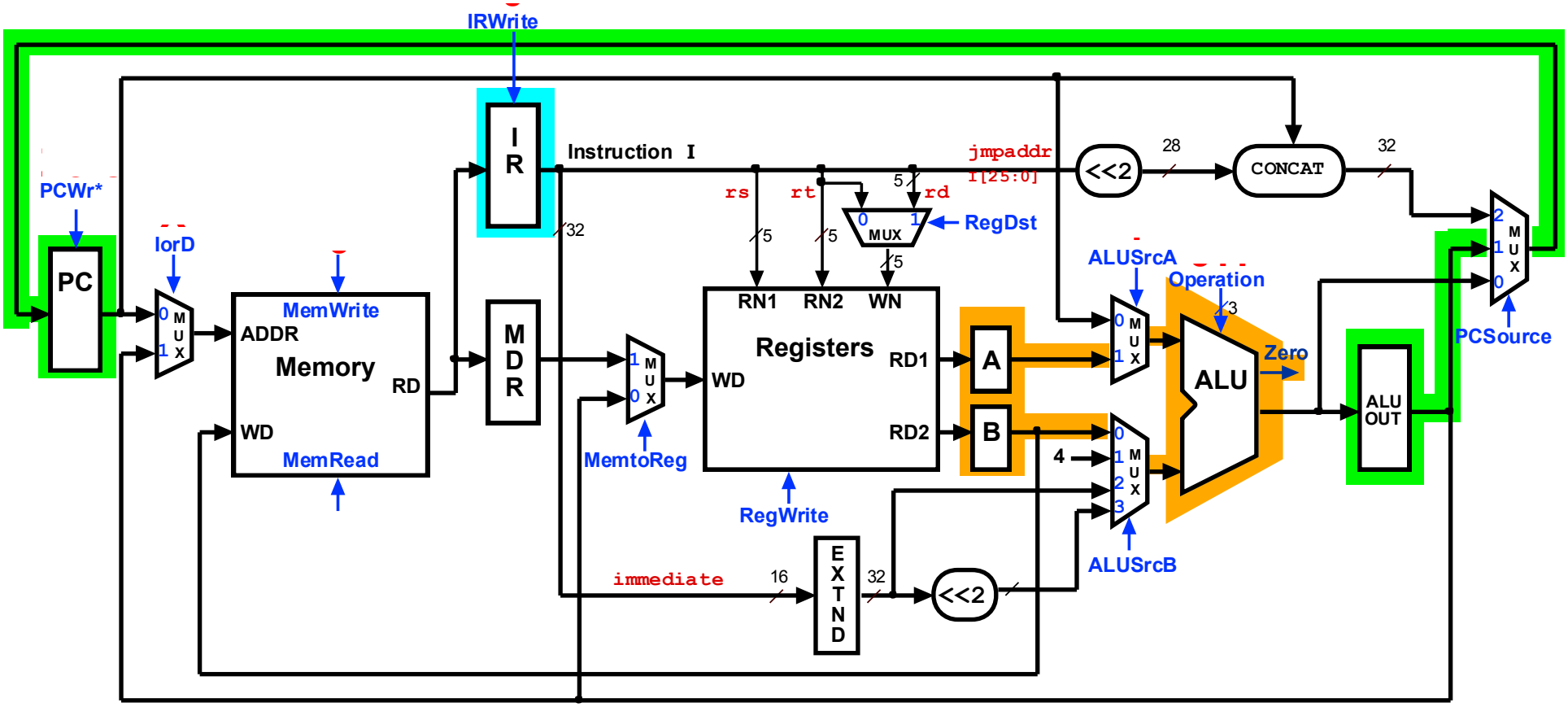
Multicycle Execution Steps (5)
Memory Read Completion (lw)

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



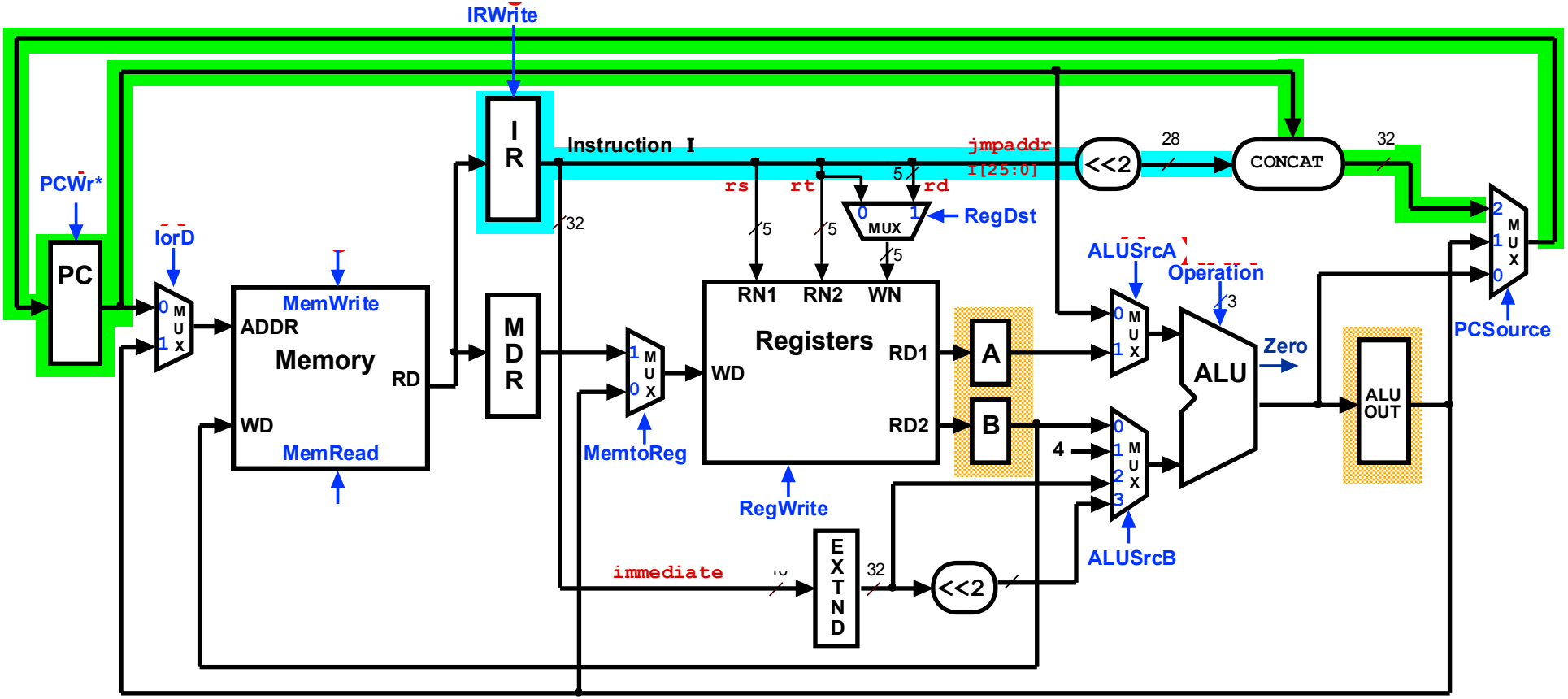
Multicycle Control Step (3): Branch Instructions

```
if (A == B) PC = ALUOut;
```



Multicycle Execution Step (3): Jump Instruction

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



Implementing Control

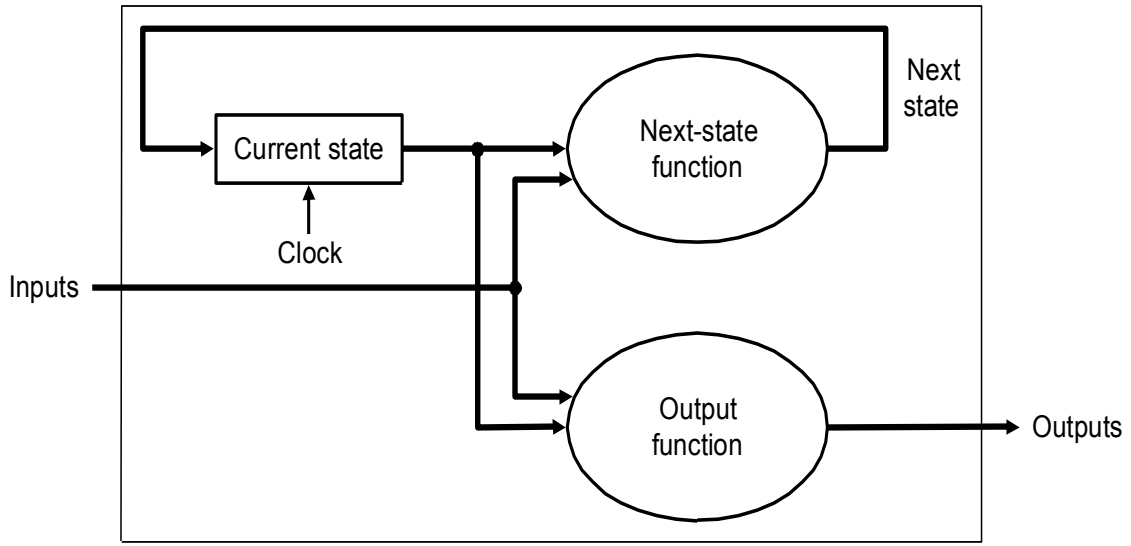
- Value of control signals is dependent upon:
  - what instruction is being executed
  - which step is being performed

- 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

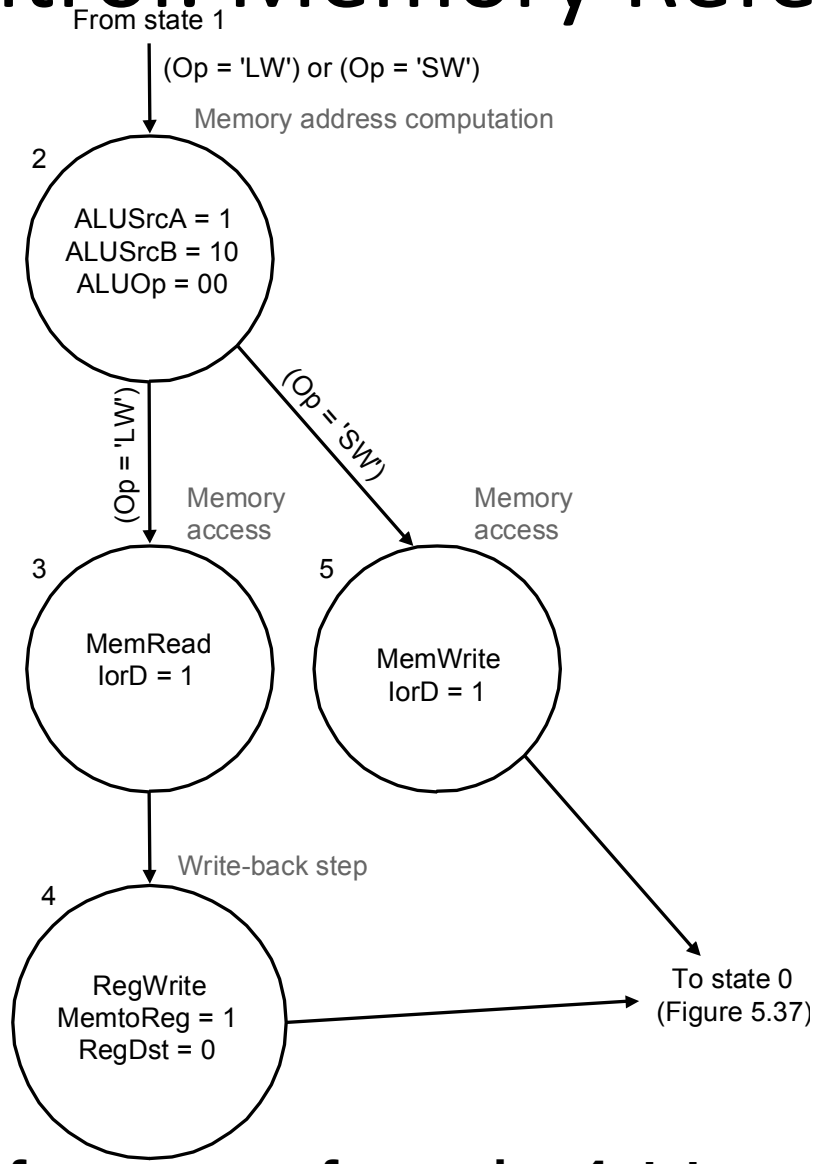
Review: Finite State Machines

- 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



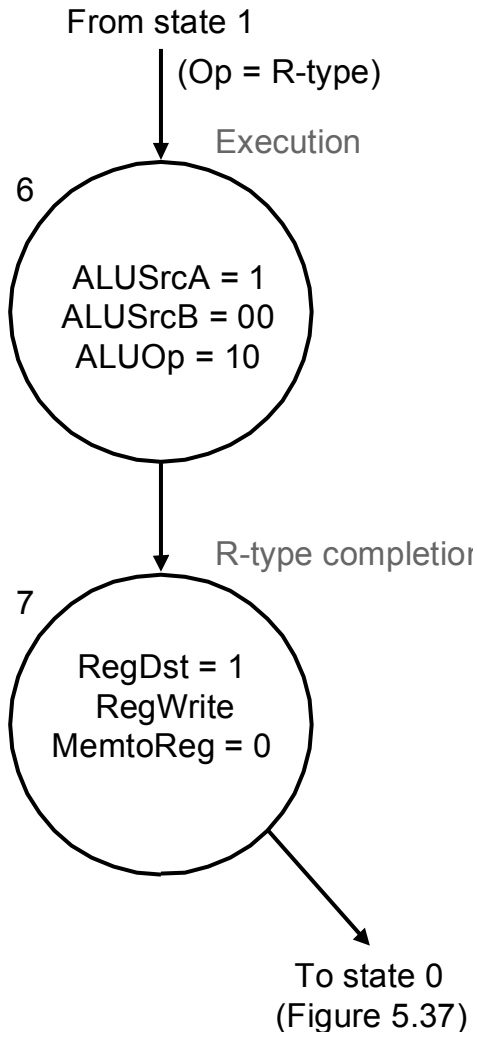
  - We'll use a *Moore machine* – output based *only* on current state

FSM Control: Memory Reference



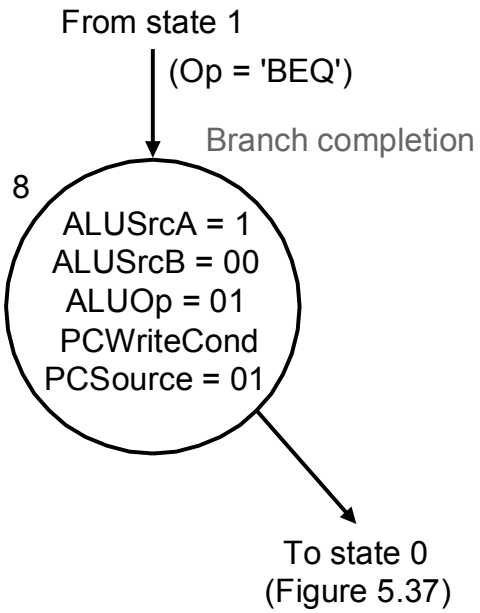
FSM control for memory-reference has 4 states

FSM Control: R-type Instruction



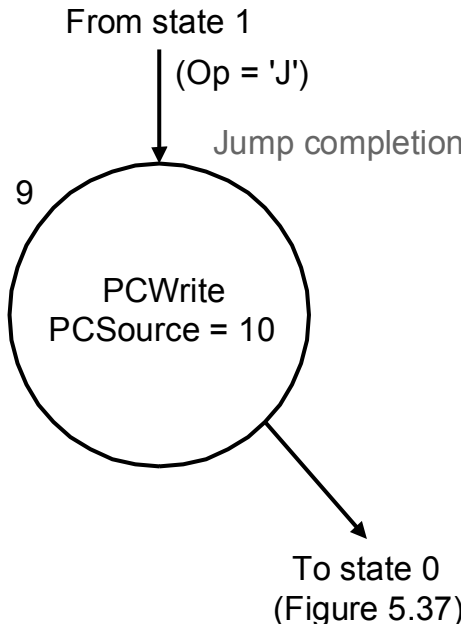
FSM control to implement R-type instructions has 2 states

FSM Control: Branch Instruction



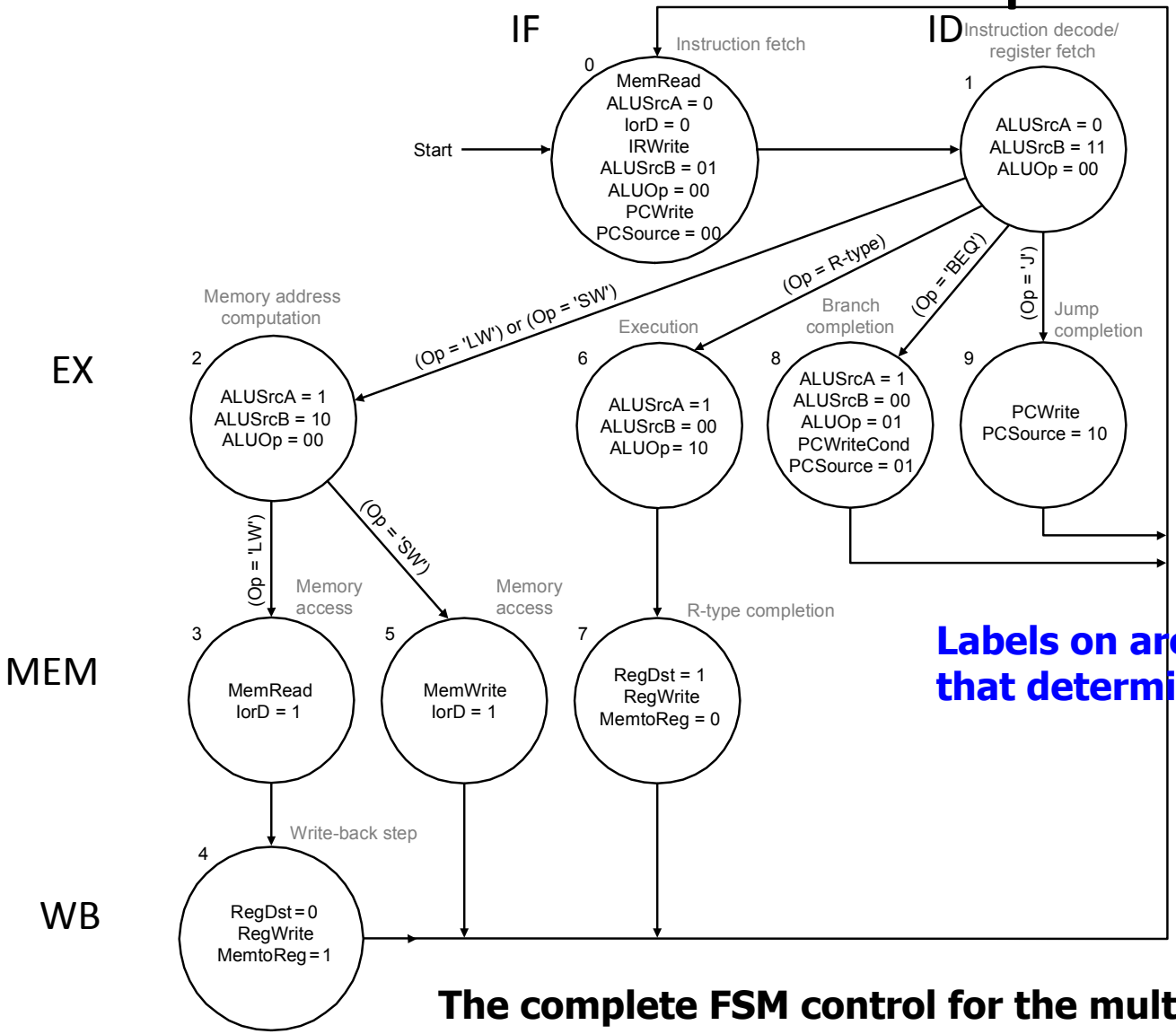
FSM control to implement branches has 1 state

FSM Control: Jump Instruction



FSM control to implement jumps has 1 state

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

Review and Questions

- Control lines
- Finite state machine