THE PROCESSOR MIPS MULTYCYCLE

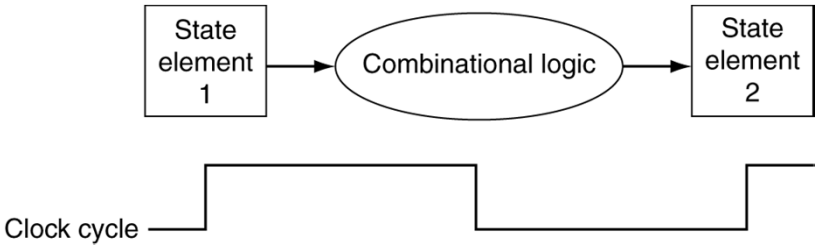
Prof. Sebastian Eslava

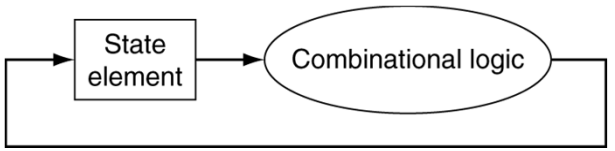
Multicycle MIPS Processor

- Single-cycle microarchitecture:
  + simple
  - cycle time limited by longest instruction (`lw`)
  - two adders/ALUs and two memories
- Multicycle microarchitecture:
  + higher clock speed
  + simpler instructions run faster
  + reuse expensive hardware on multiple cycles
  - sequencing overhead paid many times
- Same design steps: datapath & control

Clocking Methodology

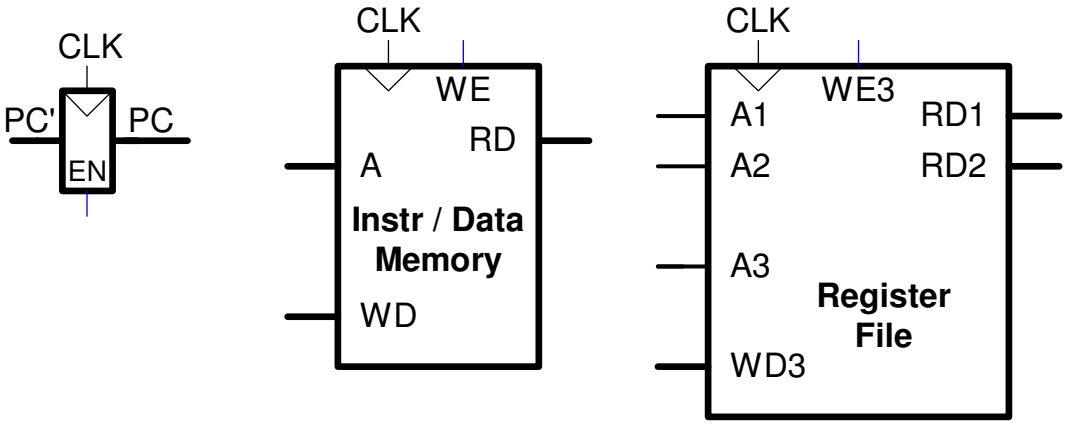
- Combinational logic transforms data during clock cycles
  - Between clock edges
  - Input from state elements, output to state element
  - Longest delay determines clock period





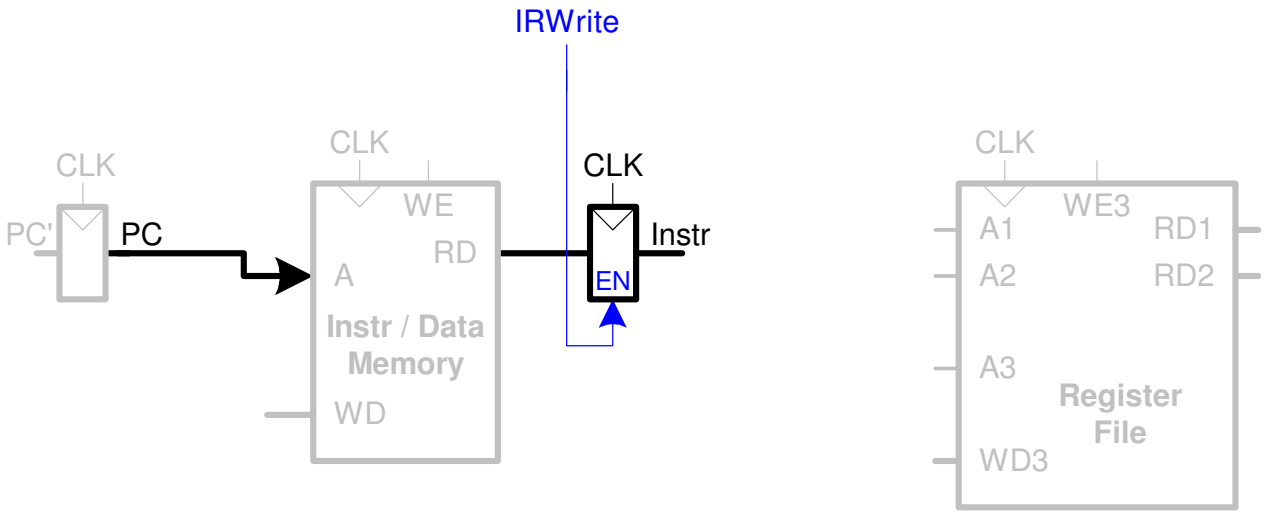
Multicycle State Elements

- Replace Instruction and Data memories with a single unified memory
  - More realistic

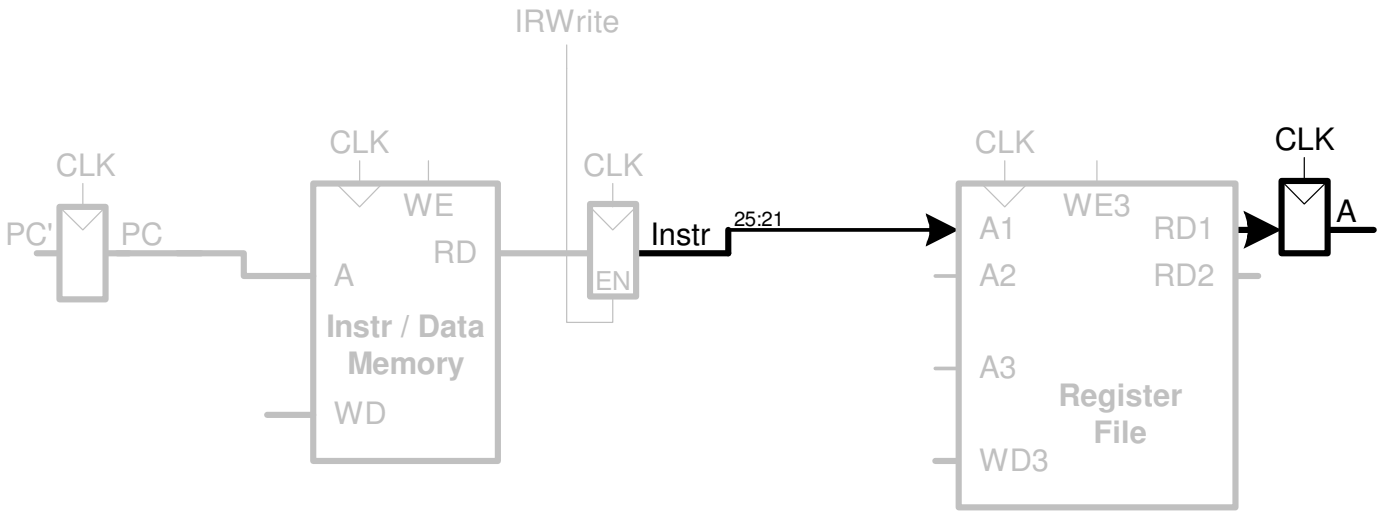


Multicycle Datapath: instruction fetch

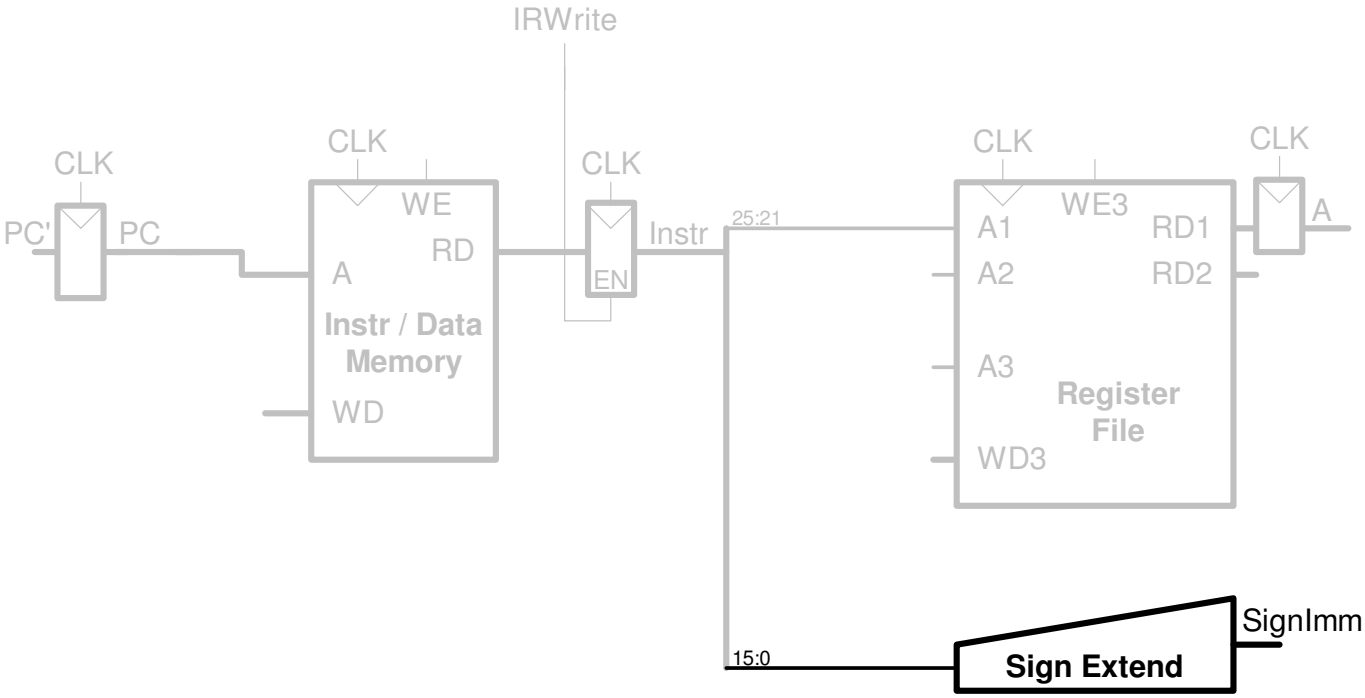
- First consider executing `lw`
- **STEP 1:** Fetch instruction



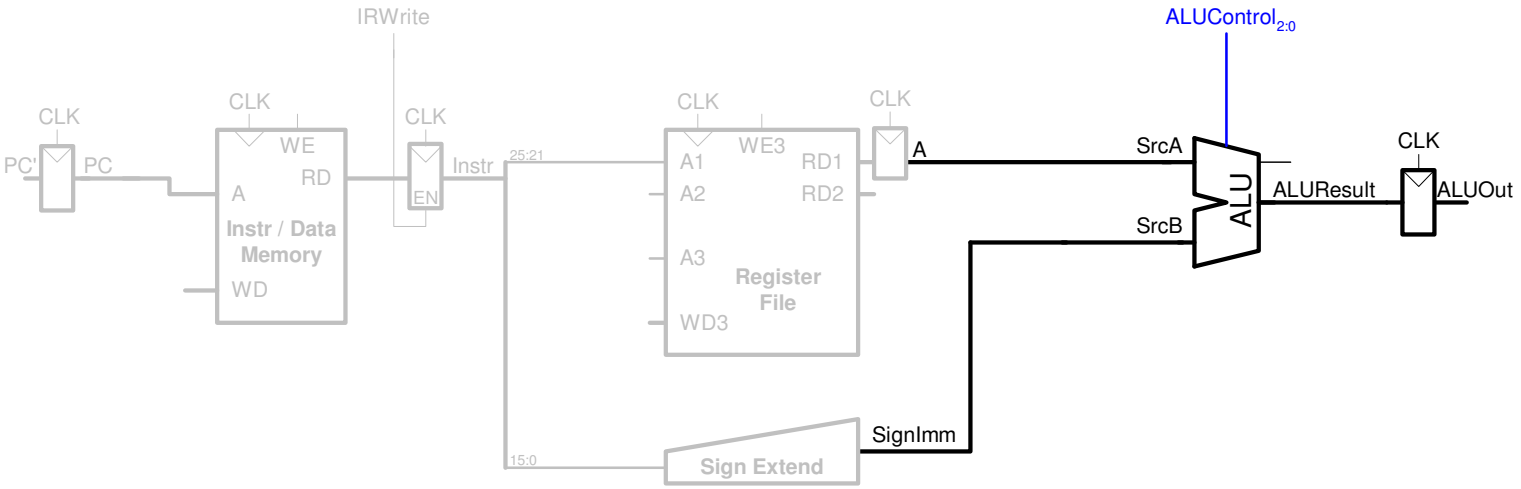
Multicycle Datapath: `lw` register read



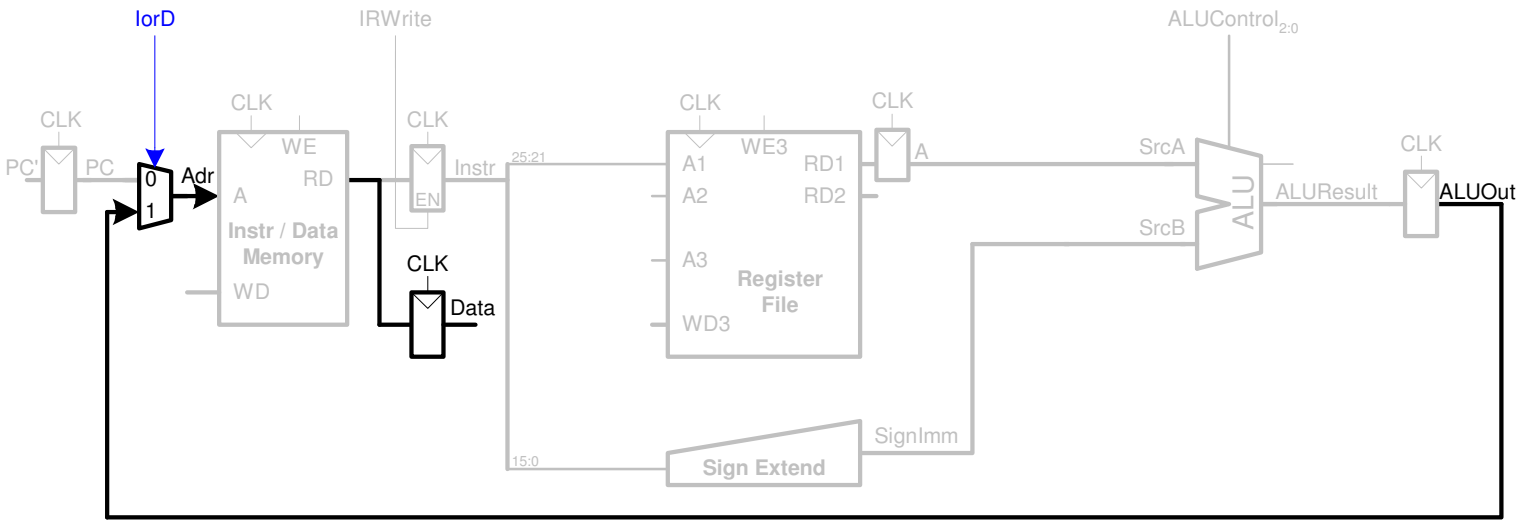
Multicycle Datapath: `lw` immediate



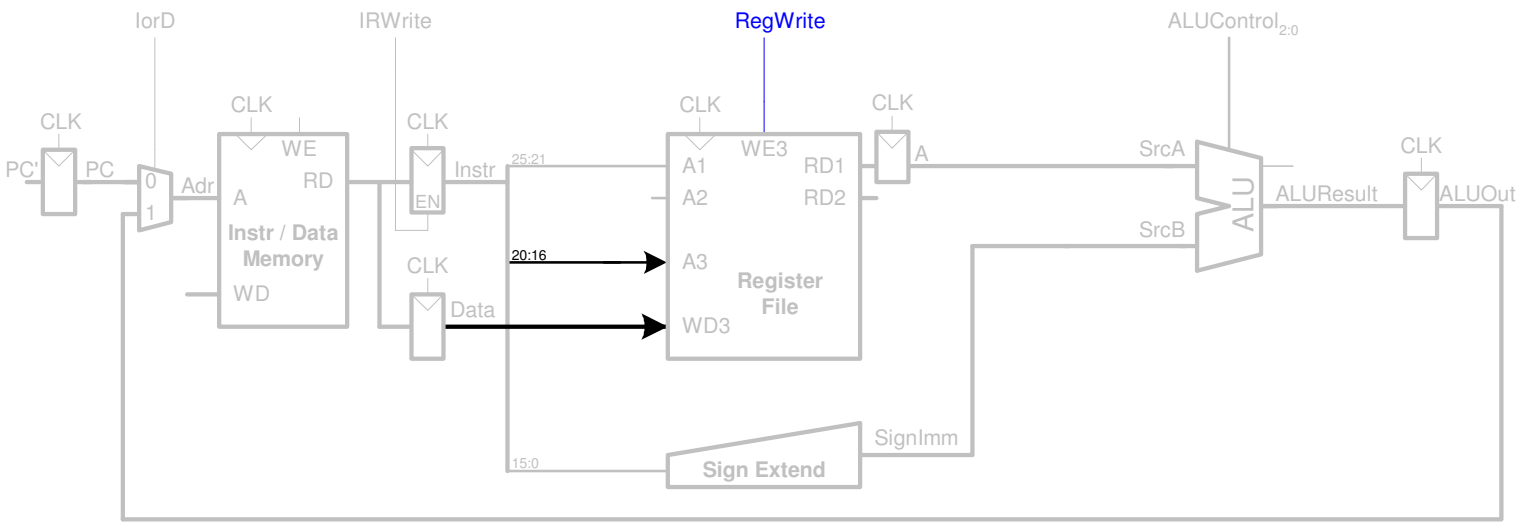
Multicycle Datapath: `lw` address



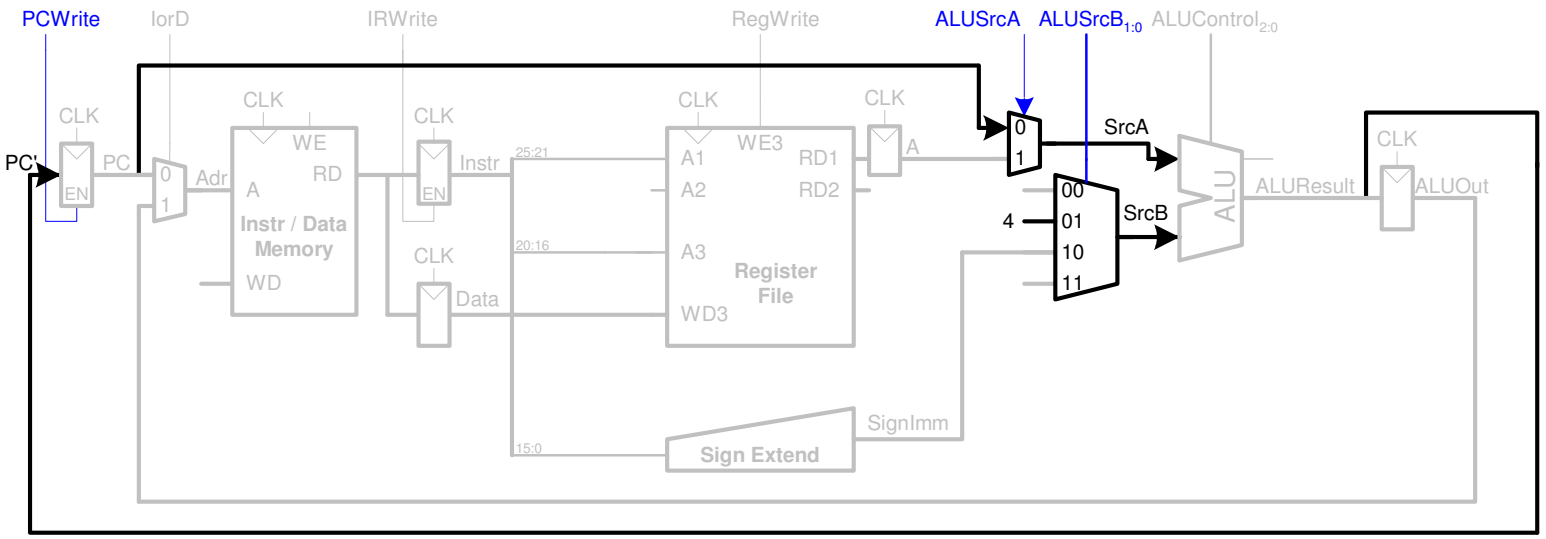
Multicycle Datapath: `lw` memory read



Multicycle Datapath: `lw` write register

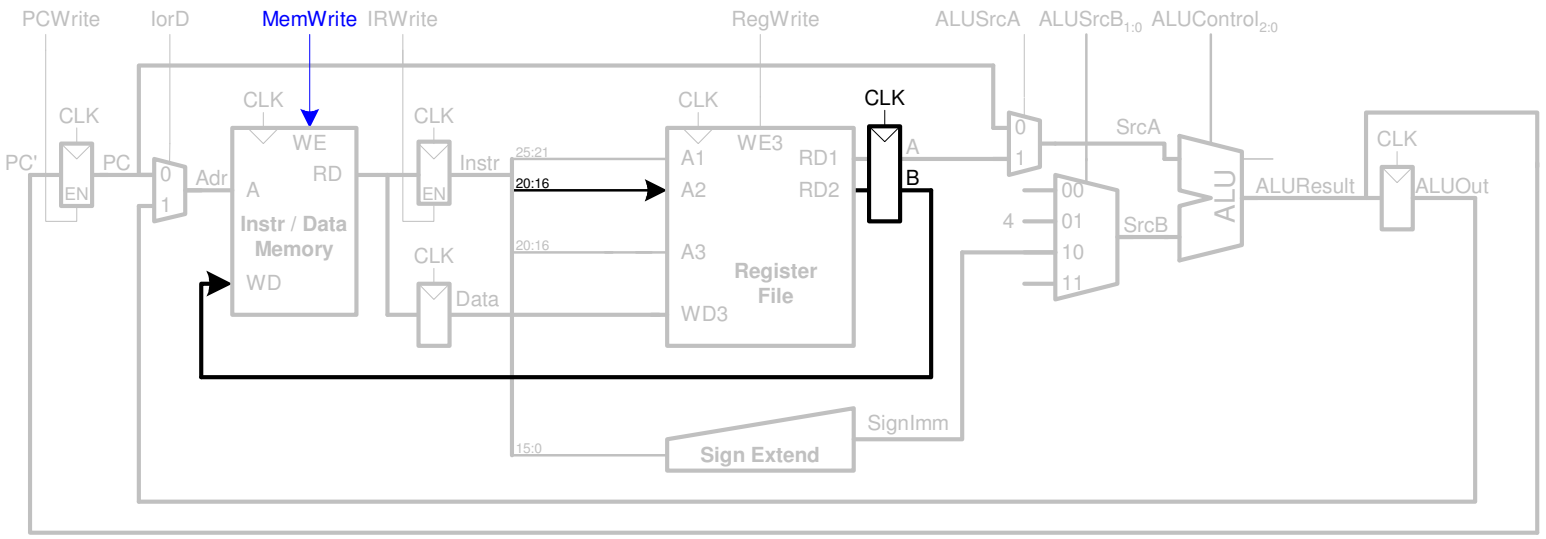


Multicycle Datapath: increment PC



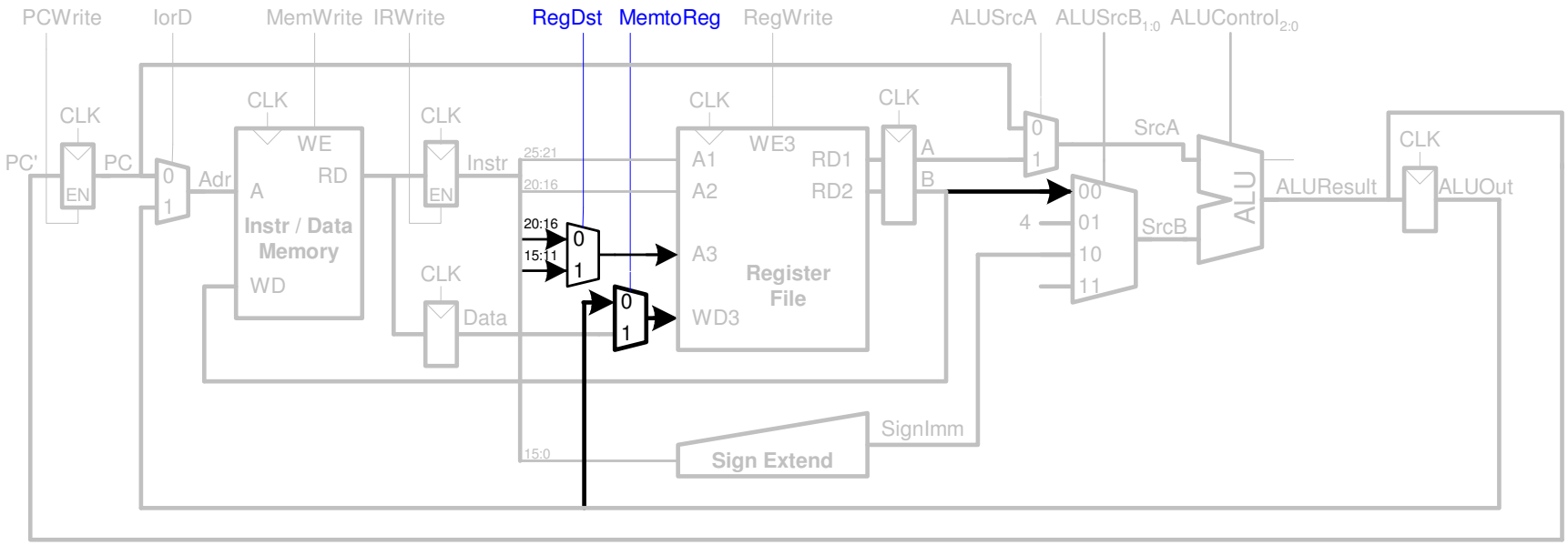
Multicycle Datapath: `sw`

- Write data in `rt` to memory



Multicycle Datapath: R-type Instructions

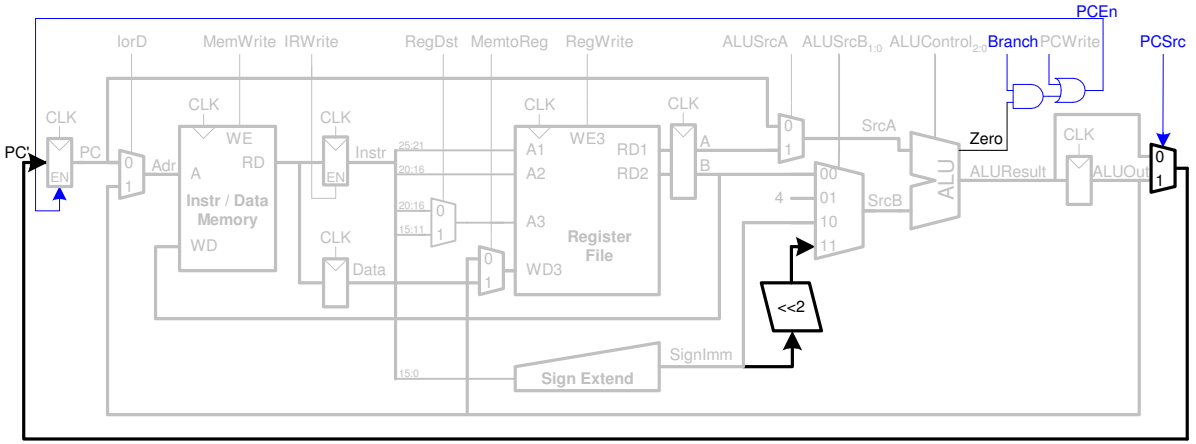
- Read from `rs` and `rt`
- Write *ALUResult* to register file
- Write to `rd` (instead of `rt`)



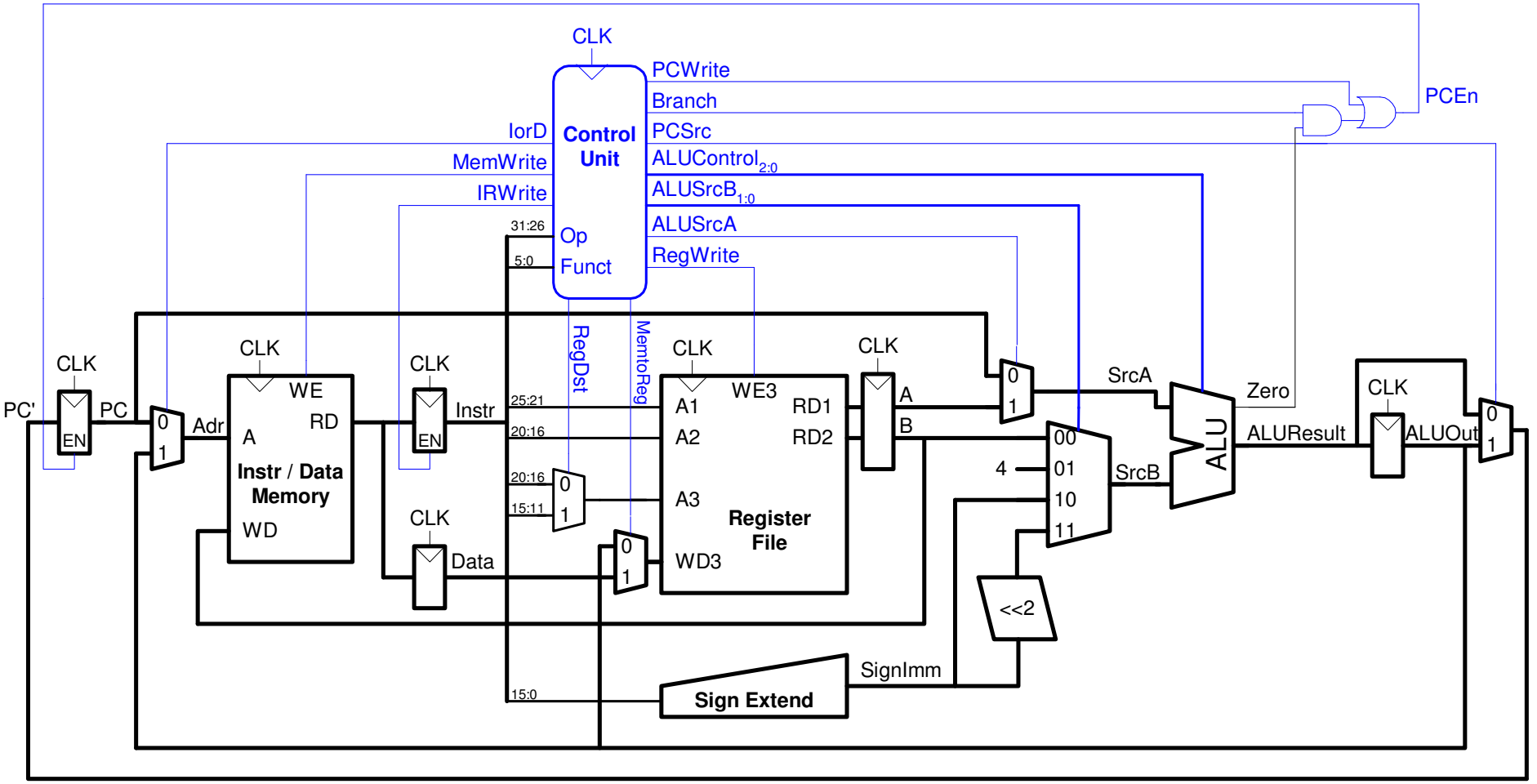
Multicycle Datapath: `beq`

- Determine whether values in `rs` and `rt` are equal
- Calculate branch target address:

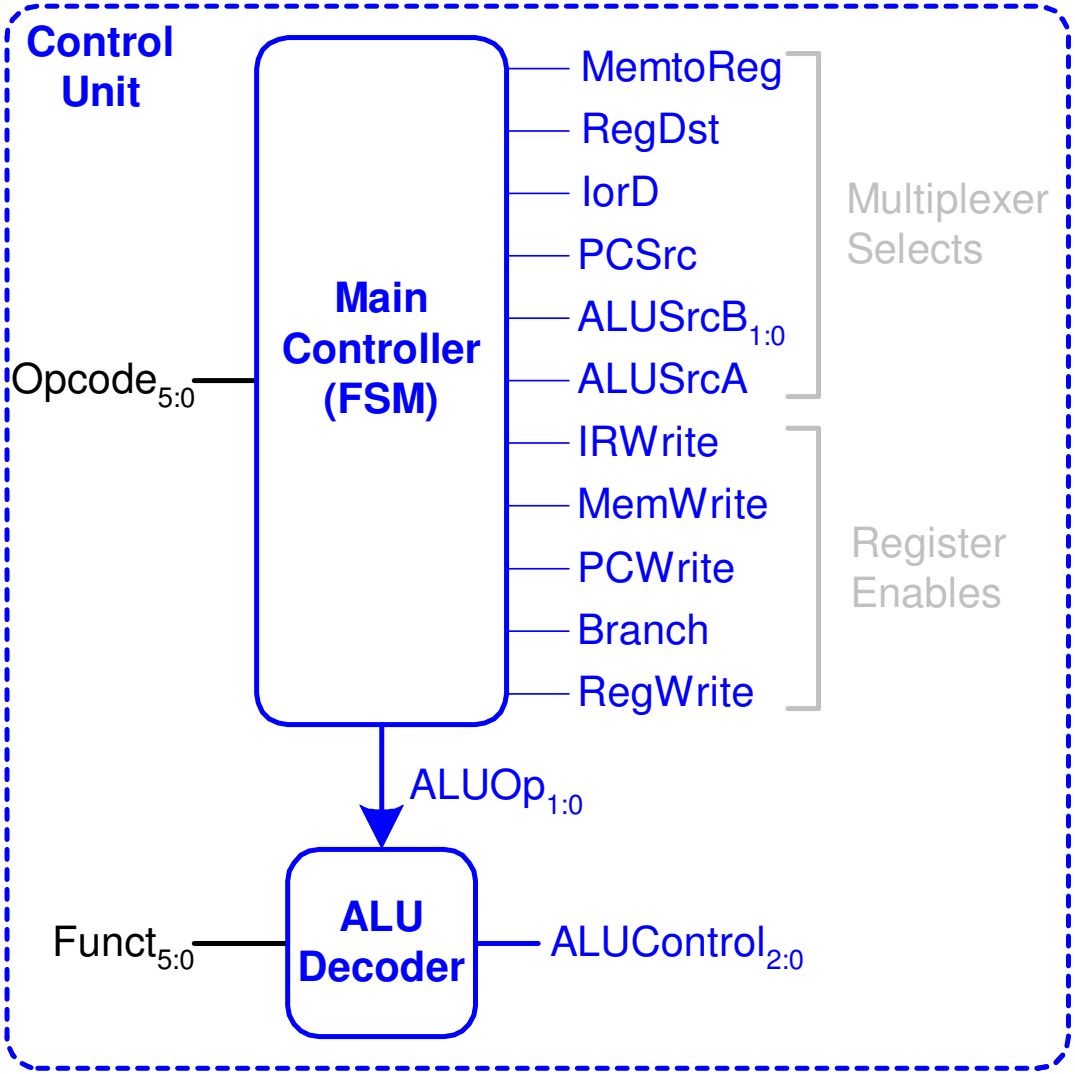
  BTA = (sign-extended immediate << 2) + (PC+4)



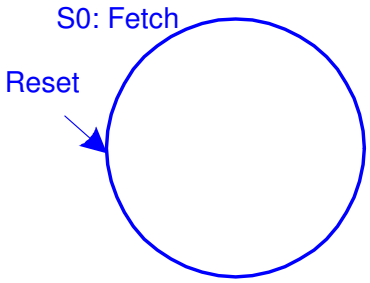
Complete Multicycle Processor

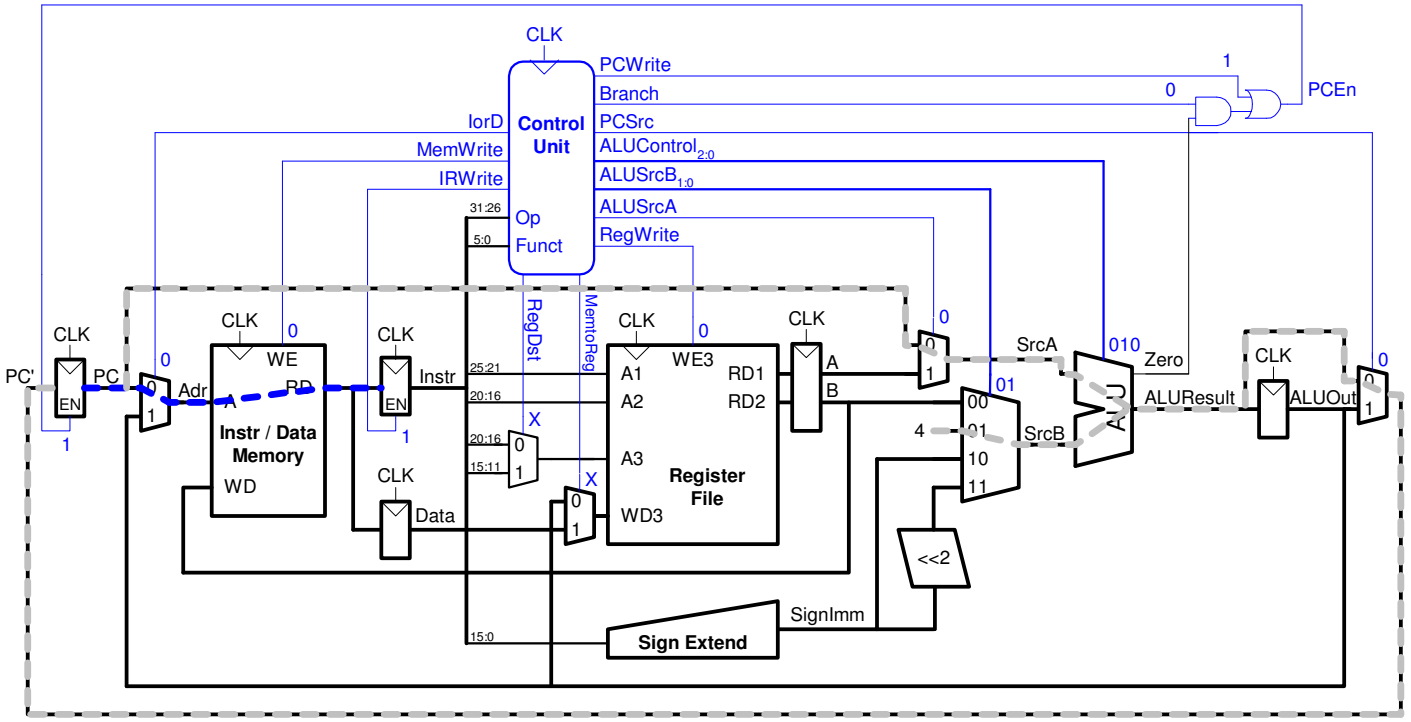


Control Unit

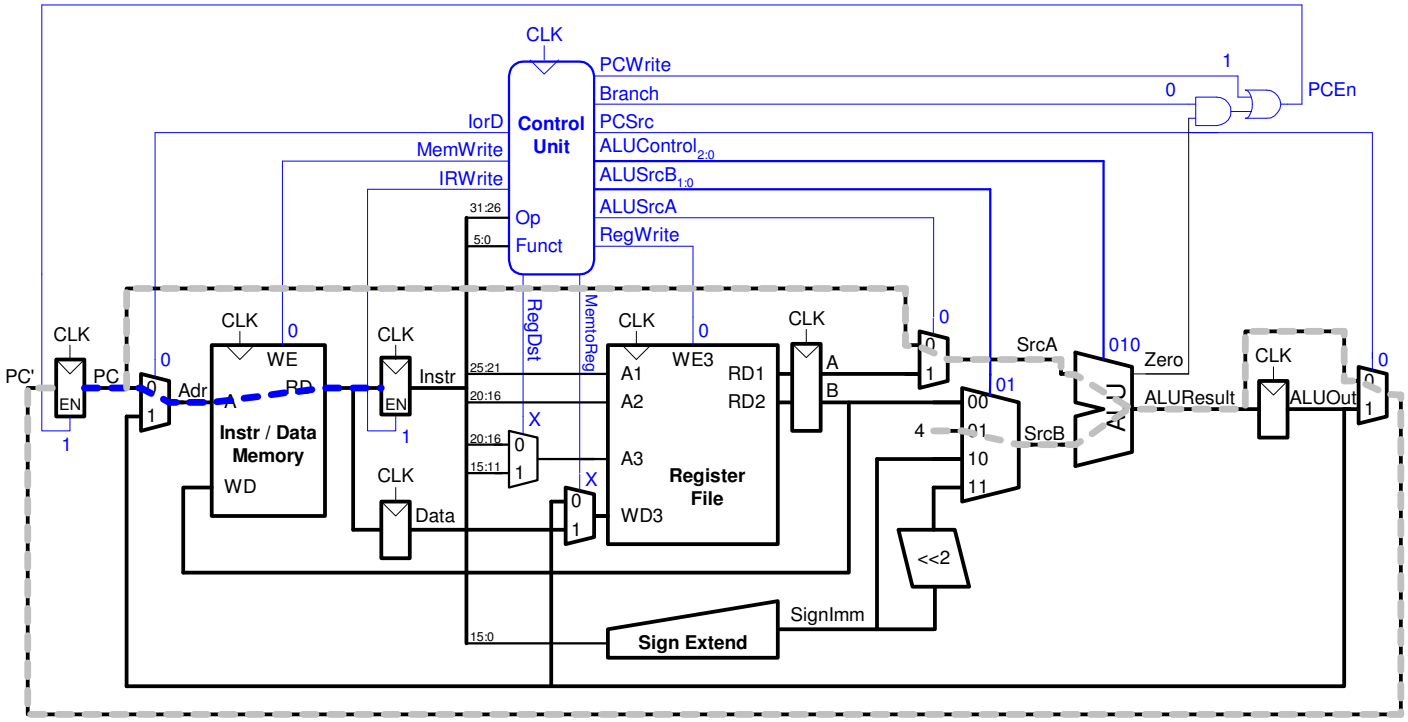


Main Controller FSM: Fetch

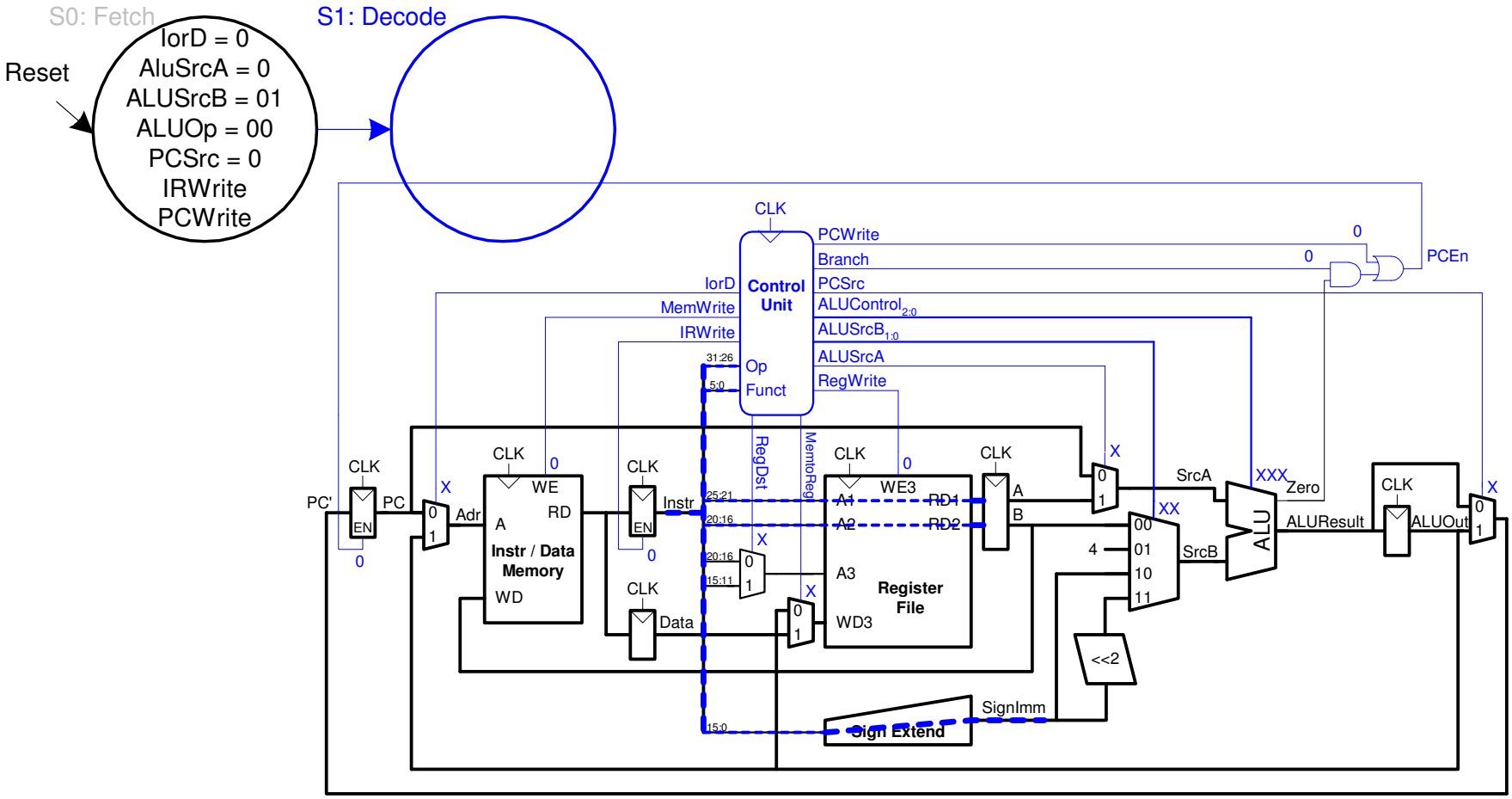




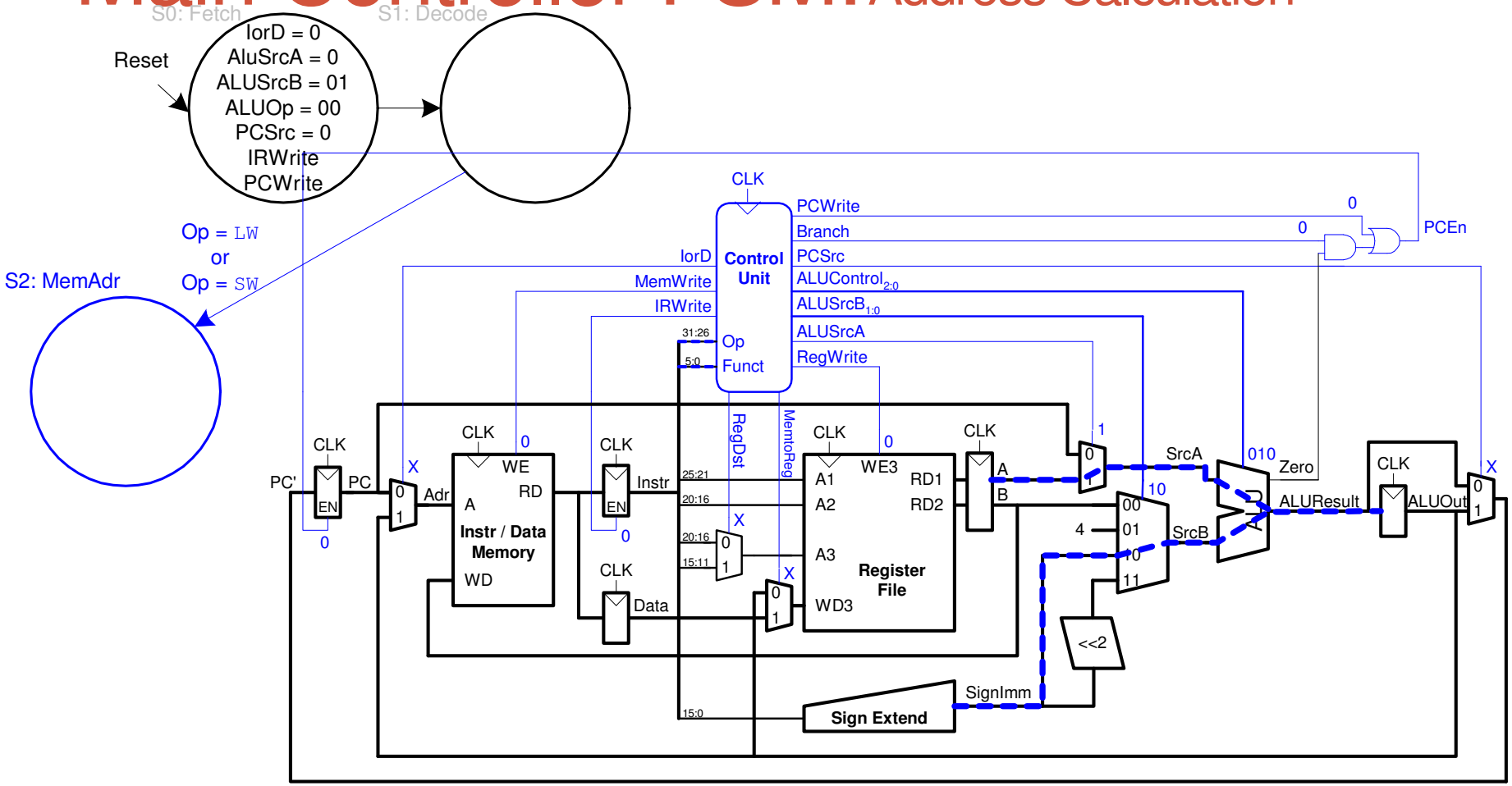
Main Controller FSM: Fetch



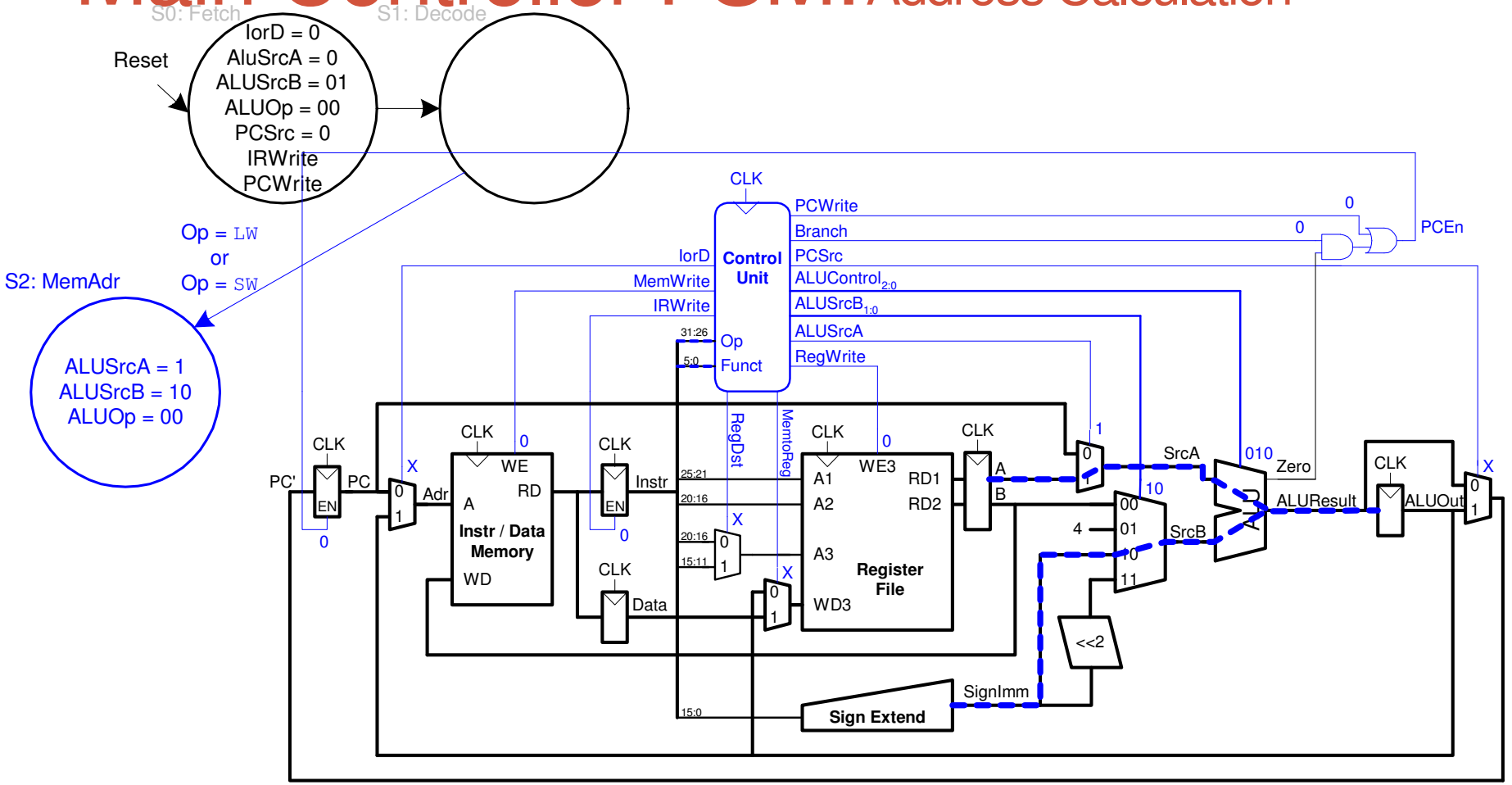
Main Controller FSM: Decode



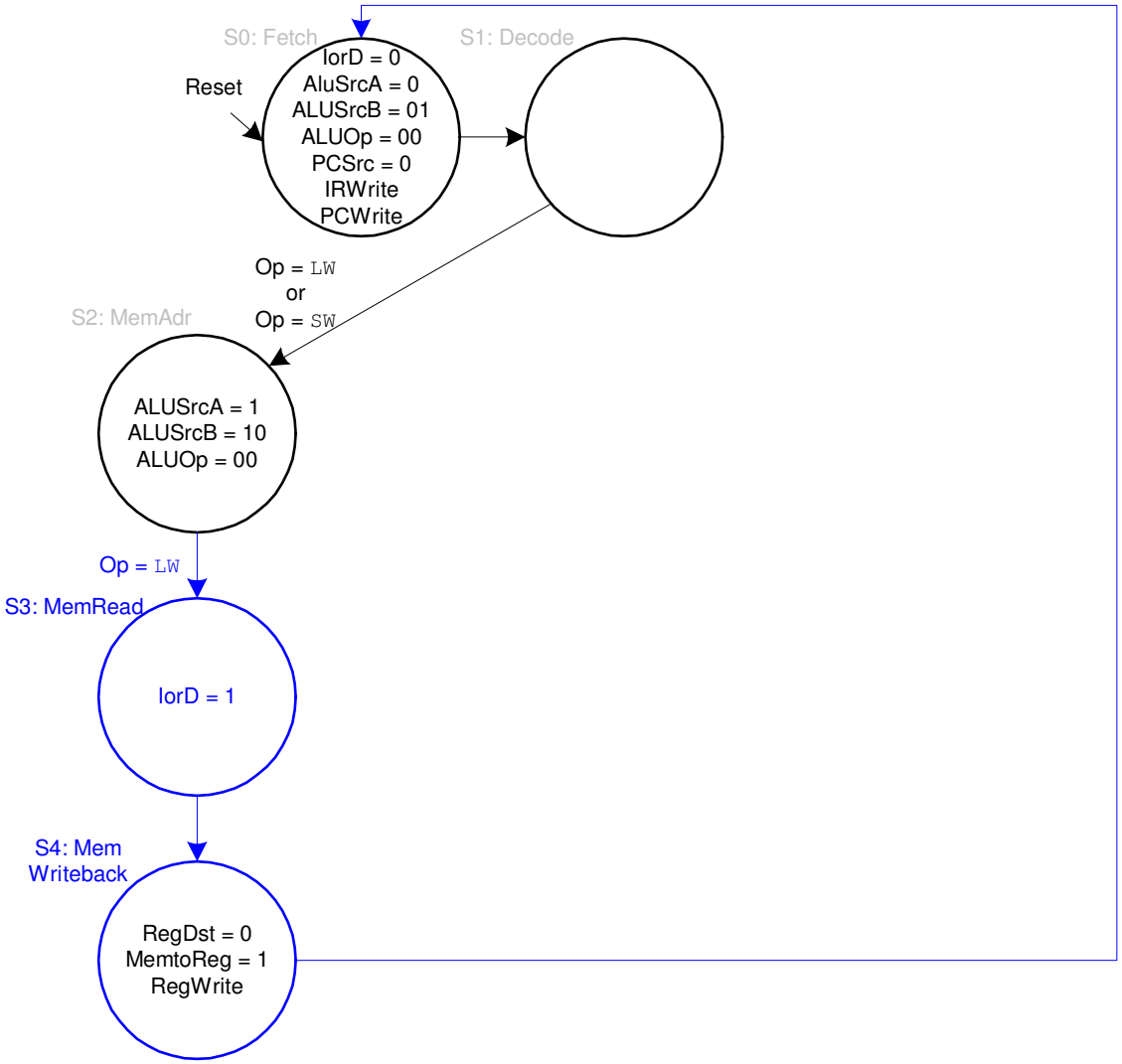
Main Controller FSM: Address Calculation



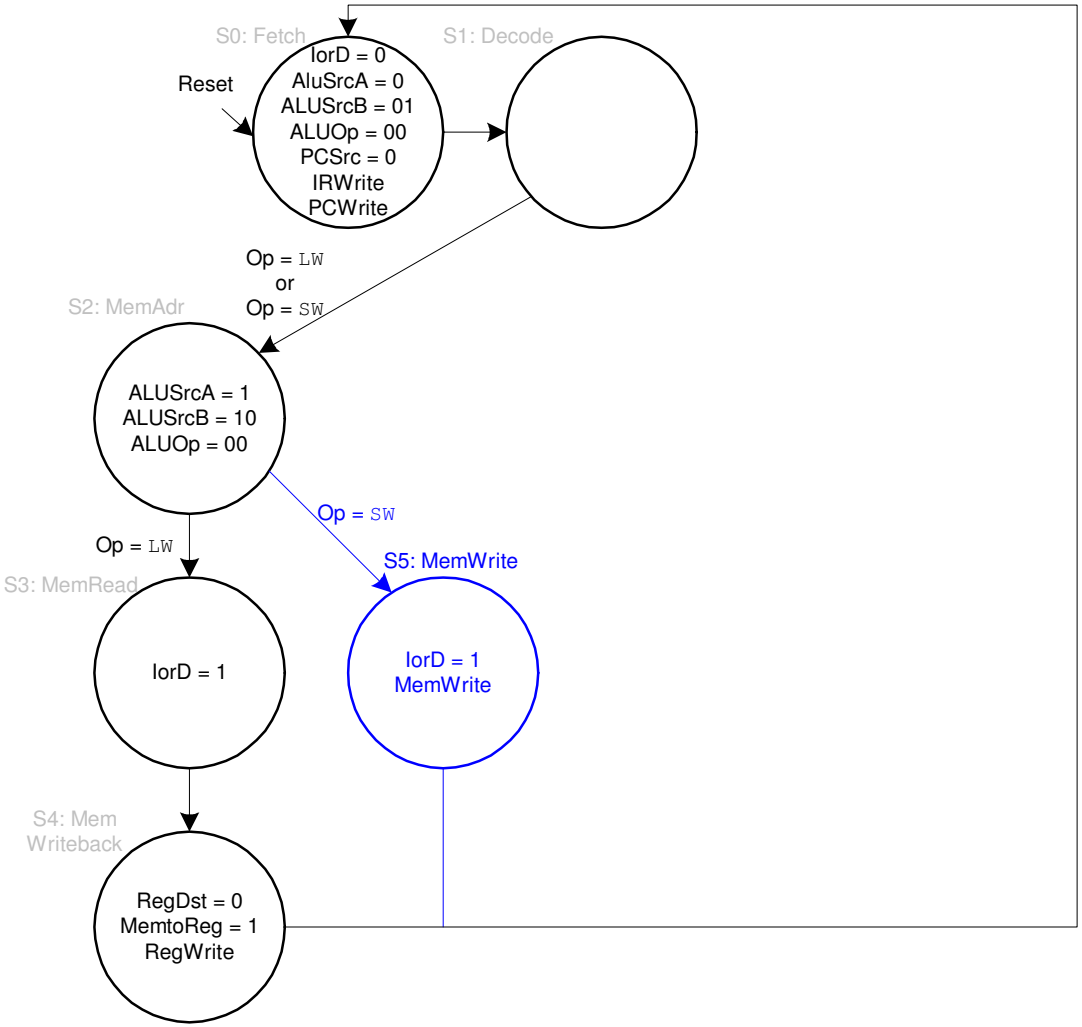
Main Controller FSM: Address Calculation



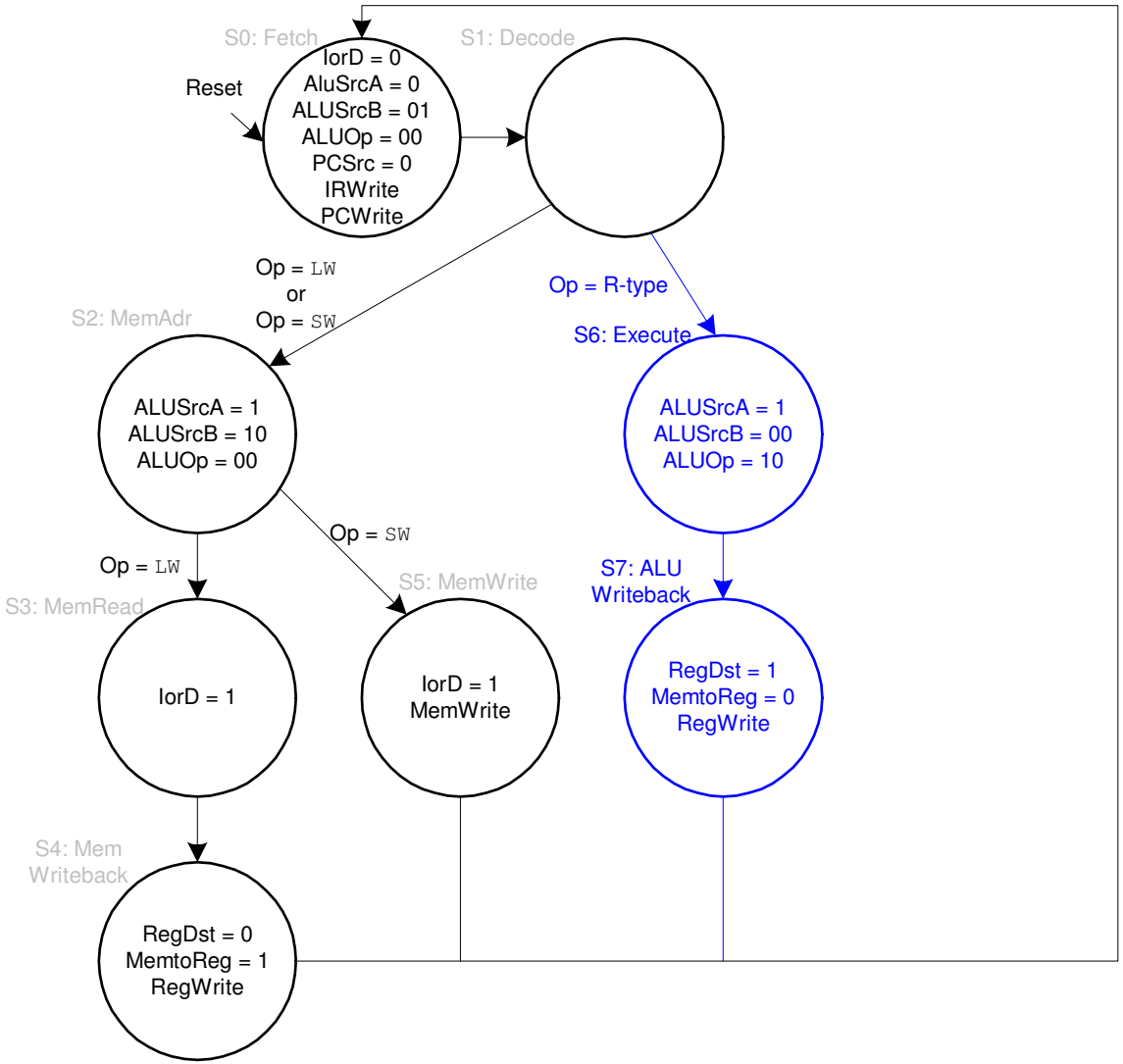
Main Controller FSM: `lw`



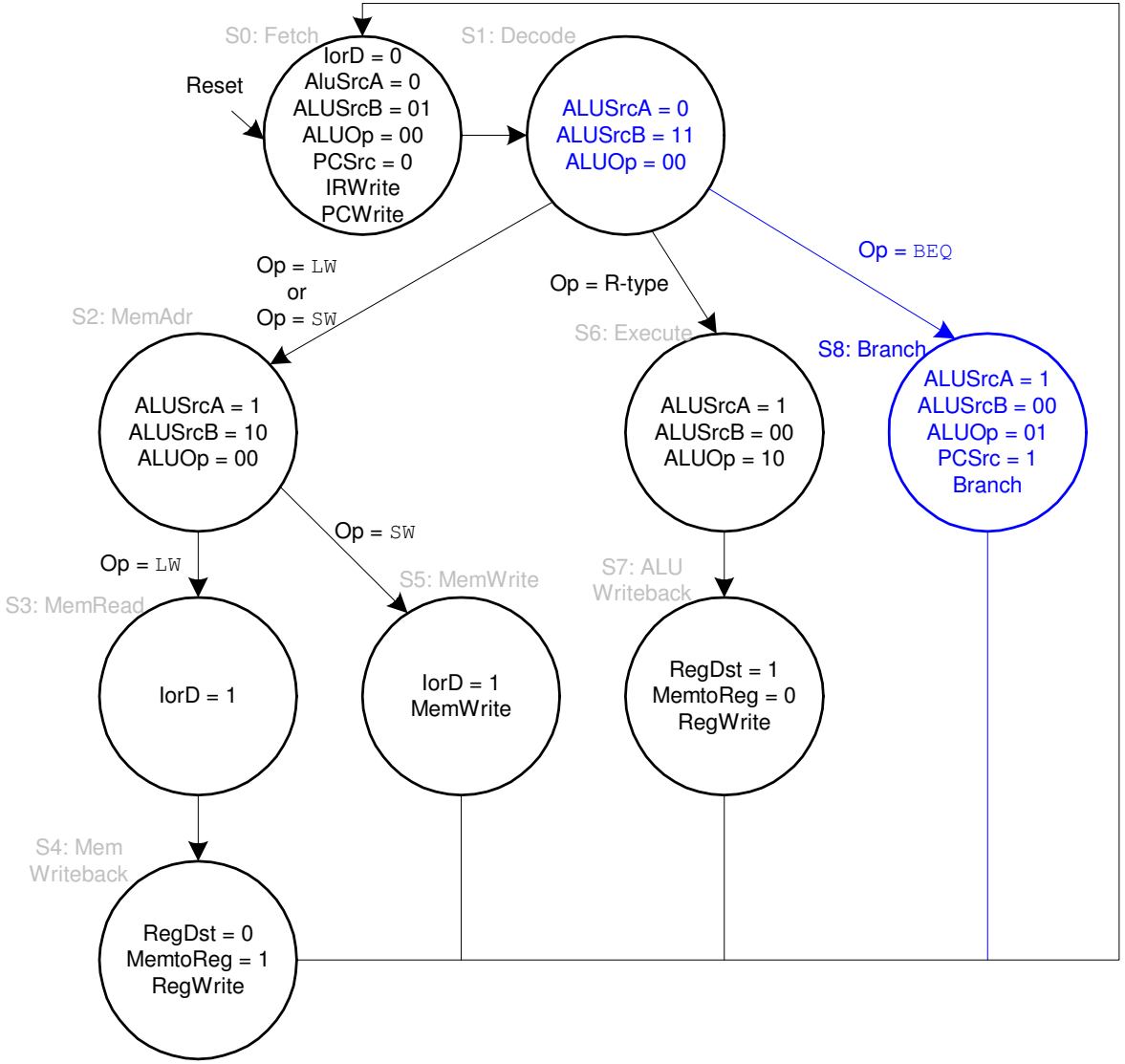
Main Controller FSM: SW



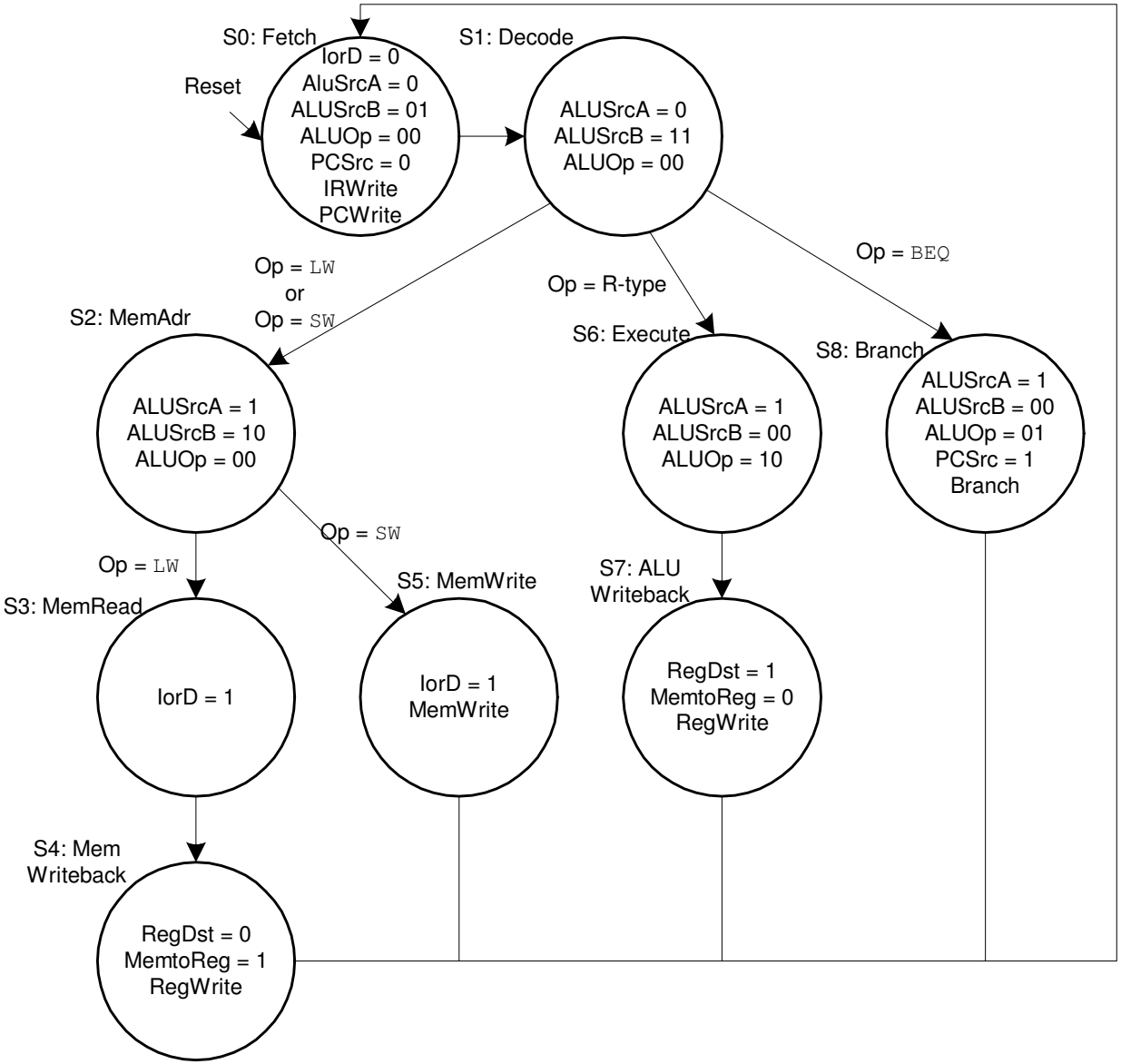
Main Controller FSM: R-Type



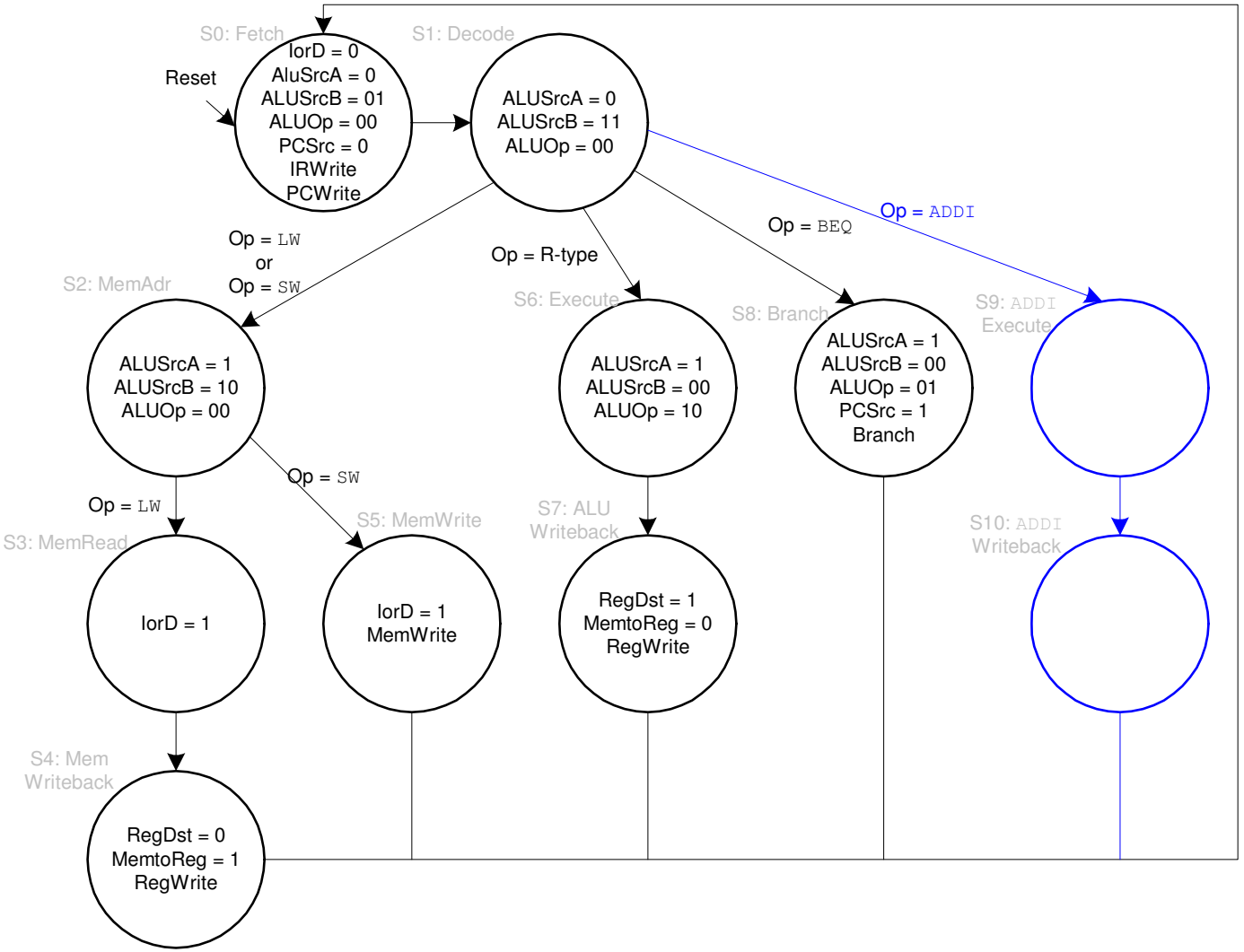
Main Controller FSM: beq



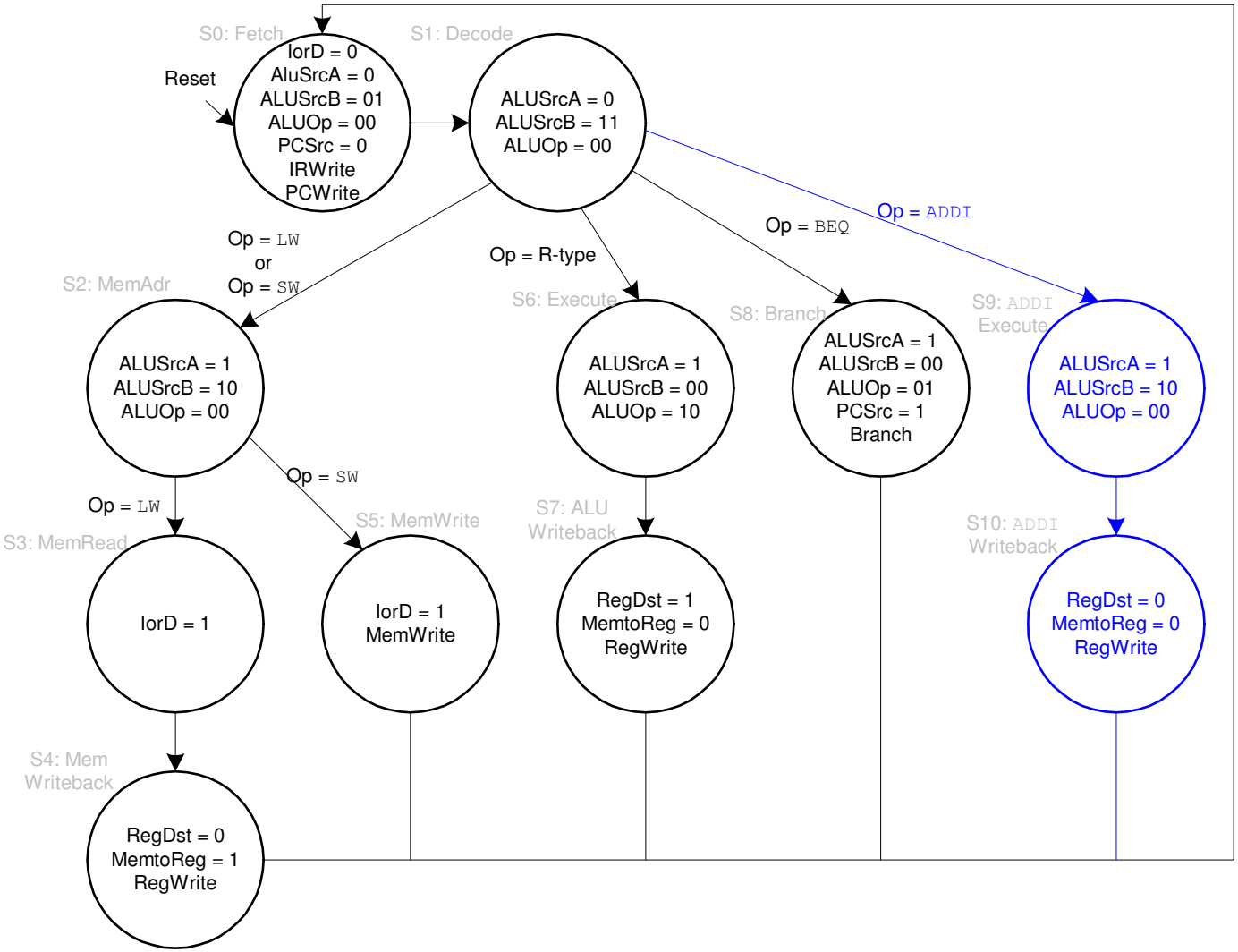
Complete Multicycle Controller FSM



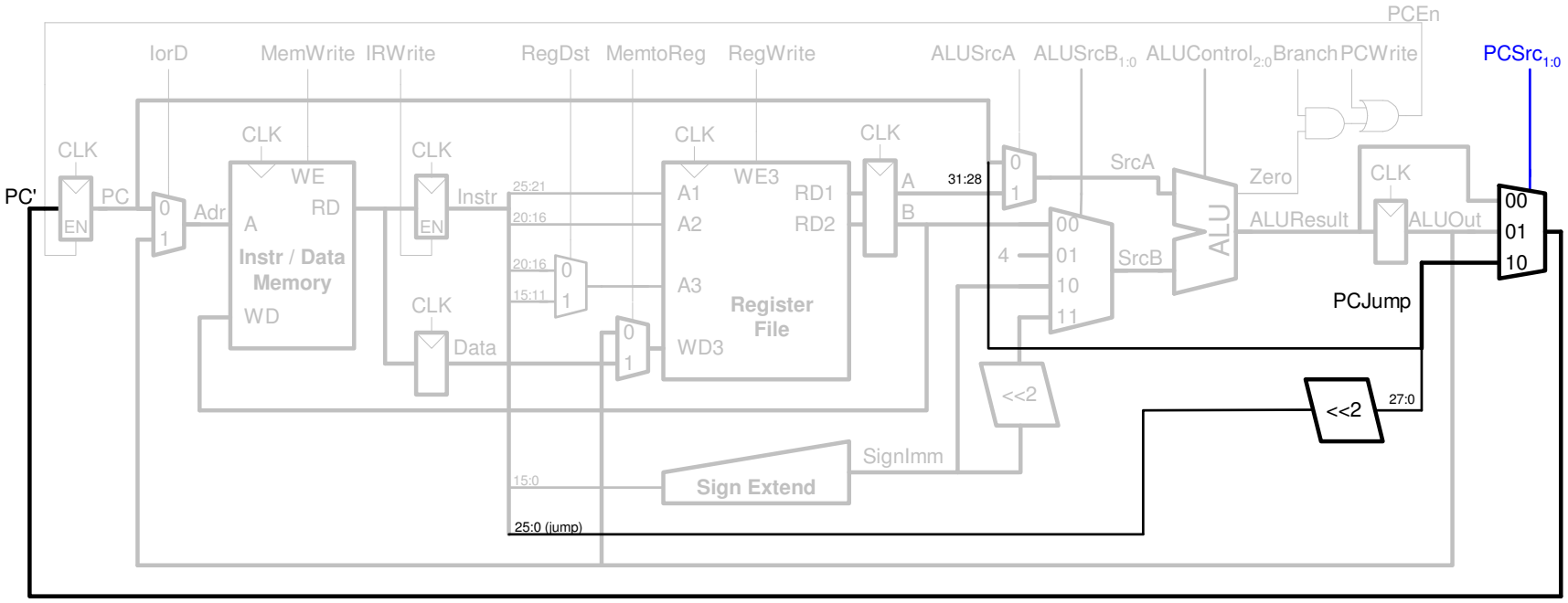
Main Controller FSM: addi



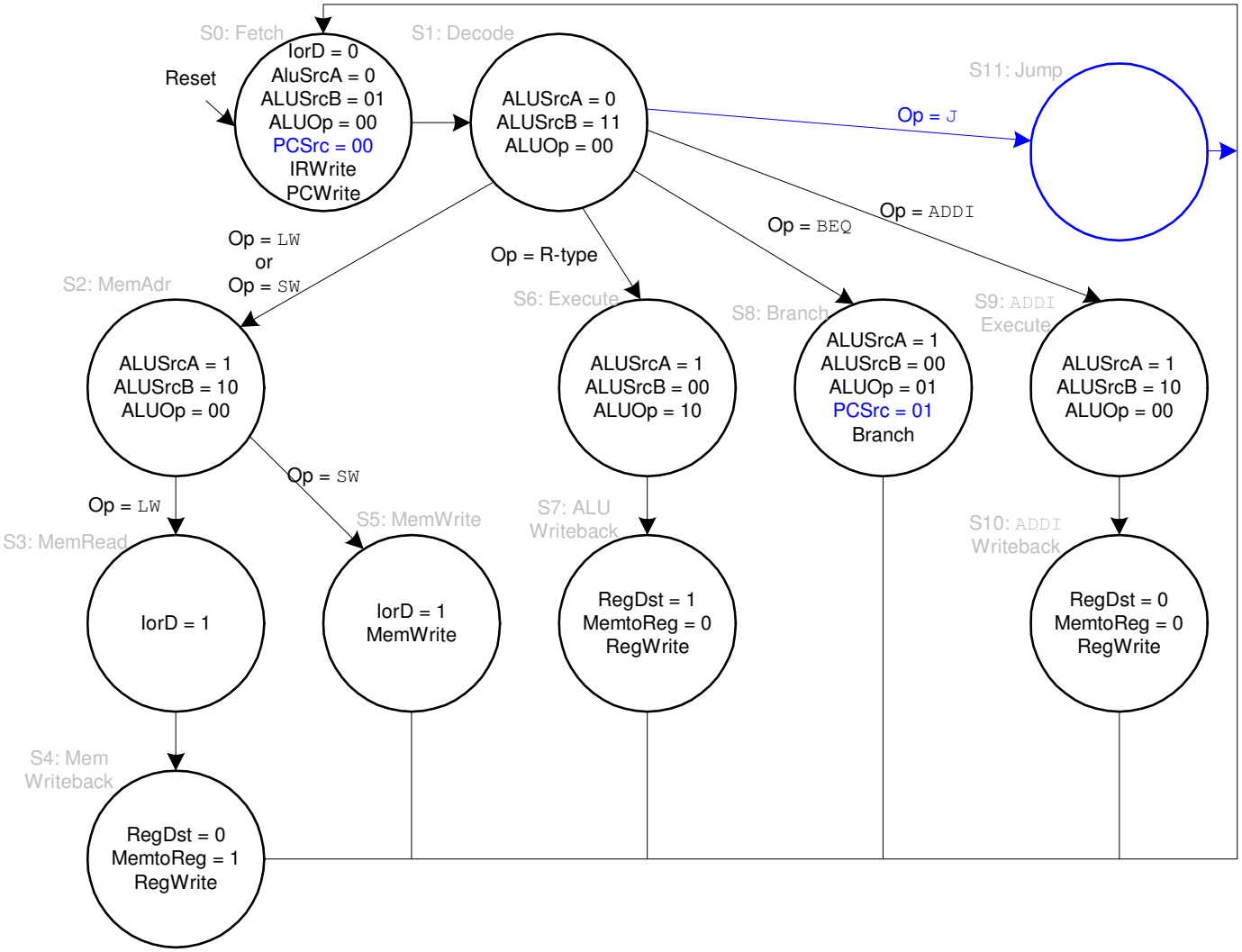
Main Controller FSM: addi



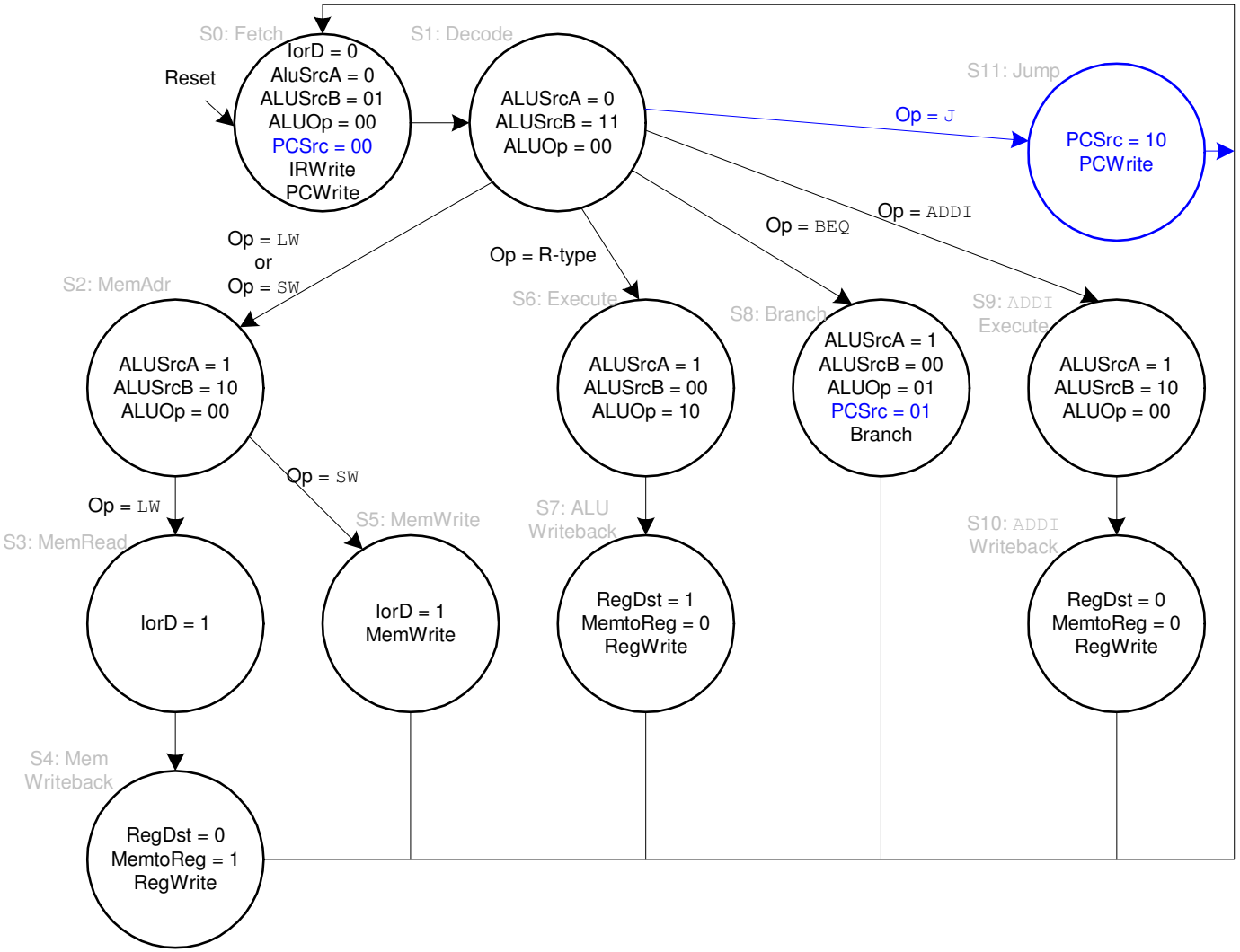
Extended Functionality: `j`



Control FSM: `j`



Control FSM: `j`



Multicycle Performance

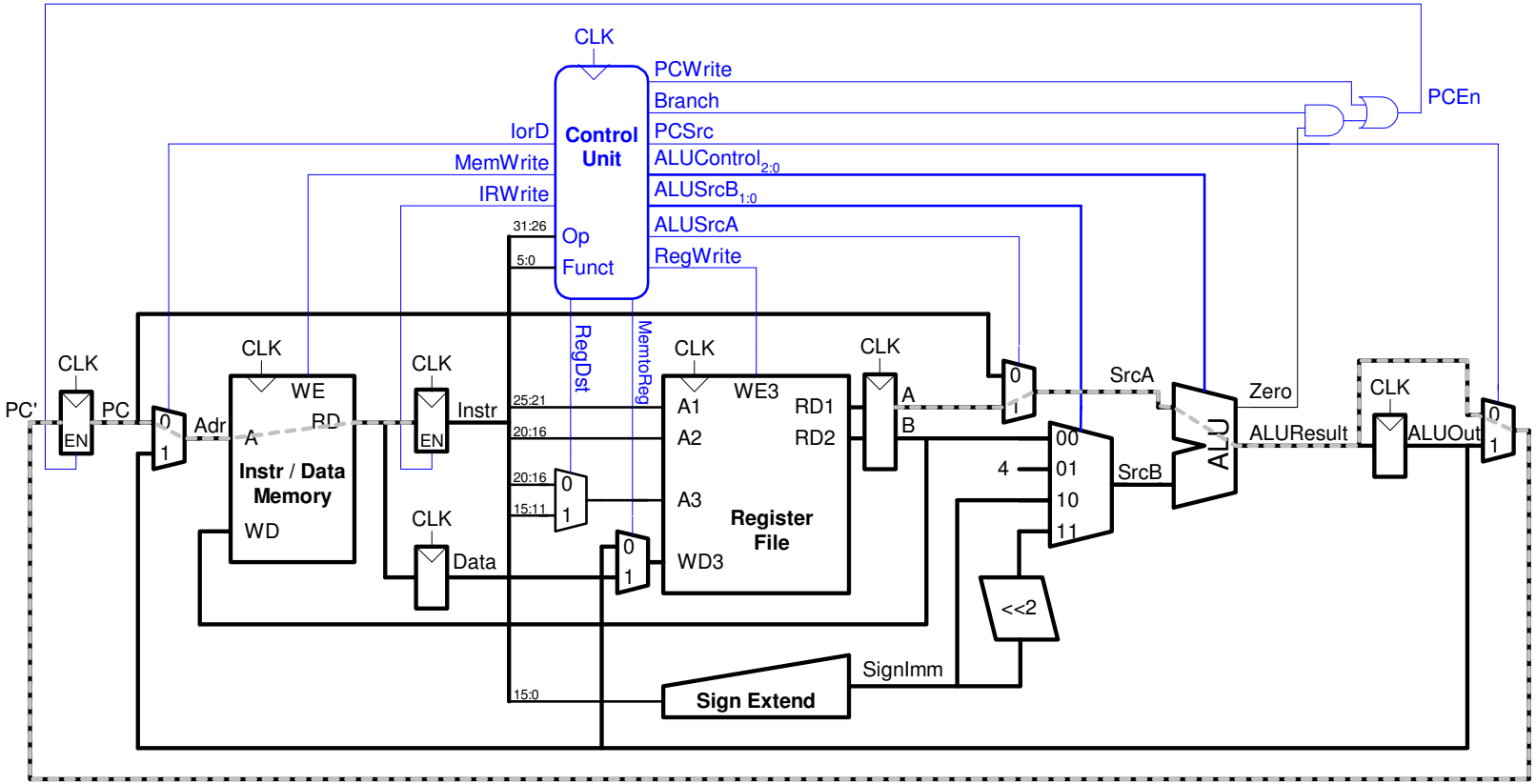
- Instructions take different number of cycles:
  - 3 cycles: `beq`, `j`
  - 4 cycles: R-Type, `sw`, `addi`
  - 5 cycles: `lw`
- CPI is weighted average
- SPECINT2000 benchmark:
  - 25% loads
  - 10% stores
  - 11% branches
  - 2% jumps
  - 52% R-type

Average CPI = (0.11 + 0.02)(3) + (0.52 + 0.10)(4) + (0.25)(5) = 4.12

Multicycle Performance

- Multicycle critical path:

$$T_c = t_{pcq} + t_{\text{mux}} + \max(t_{\text{ALU}} + t_{\text{mux}}, t_{\text{mem}}) + t_{\text{setup}}$$



Multicycle Performance Example

Element	Parameter	Delay (ps)
Register clock-to-Q	t_{pcq_PC}	30
Register setup	t_{setup}	20
Multiplexer	t_{mux}	25
ALU	t_{ALU}	200
Memory read	t_{mem}	250
Register file read	t_{RFread}	150
Register file setup	$t_{RFsetup}$	20

$$
\begin{aligned}
T_c &= t_{pcq\_PC} + t_{mux} + \max(t_{ALU} + t_{mux}, t_{mem}) + t_{setup} \\
&= t_{pcq\_PC} + t_{mux} + t_{mem} + t_{setup} \\
&= [30 + 25 + 250 + 20]\ \text{ps} \\
&= 325\ \text{ps}
\end{aligned}
$$

Multicycle Performance Example

- For a program with 100 billion instructions executing on a multicycle MIPS processor
  - CPI = 4.12
  - T_c = 325 ps

Execution Time = (# instructions) × CPI × T_c

= $(100 \times 10^9)(4.12)(325 \times 10^{-12})$

= 133.9 seconds

- This is slower than the single-cycle processor (92.5 seconds). Why?

Multicycle Performance Example

- For a program with 100 billion instructions executing on a multicycle MIPS processor
  - CPI = 4.12
  - T_c = 325 ps

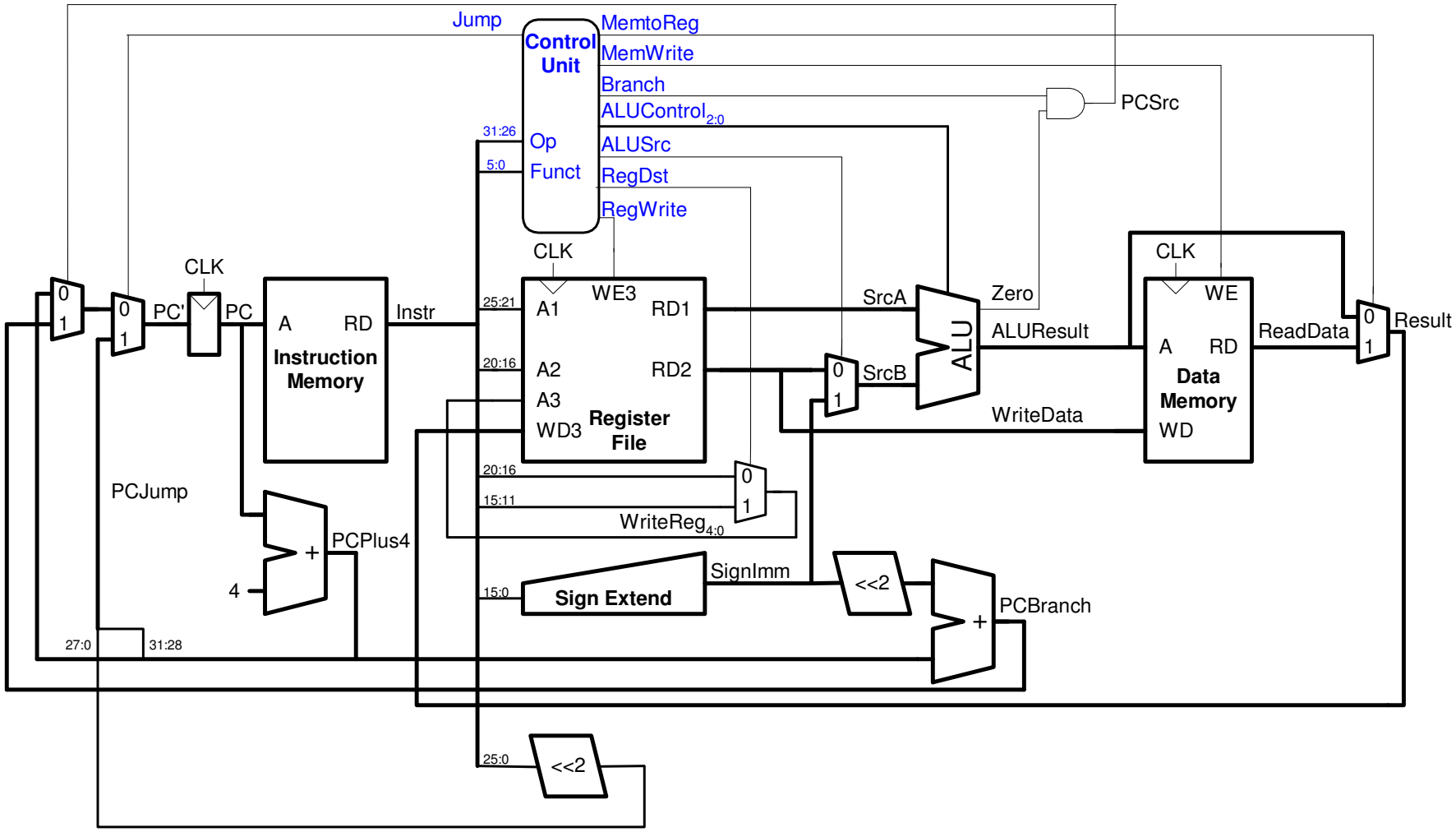
Execution Time = (# instructions) × CPI × T_c

$= (100 \times 10^9)(4.12)(325 \times 10^{-12})$

= 133.9 seconds

- This is slower than the single-cycle processor (92.5 seconds). Why?
  - Not all steps the same length
  - Sequencing overhead for each step ($t_{pcq} + t_{setup}$= 50 ps)

Review: Single-Cycle MIPS Processor



Review: Multicycle MIPS Processor