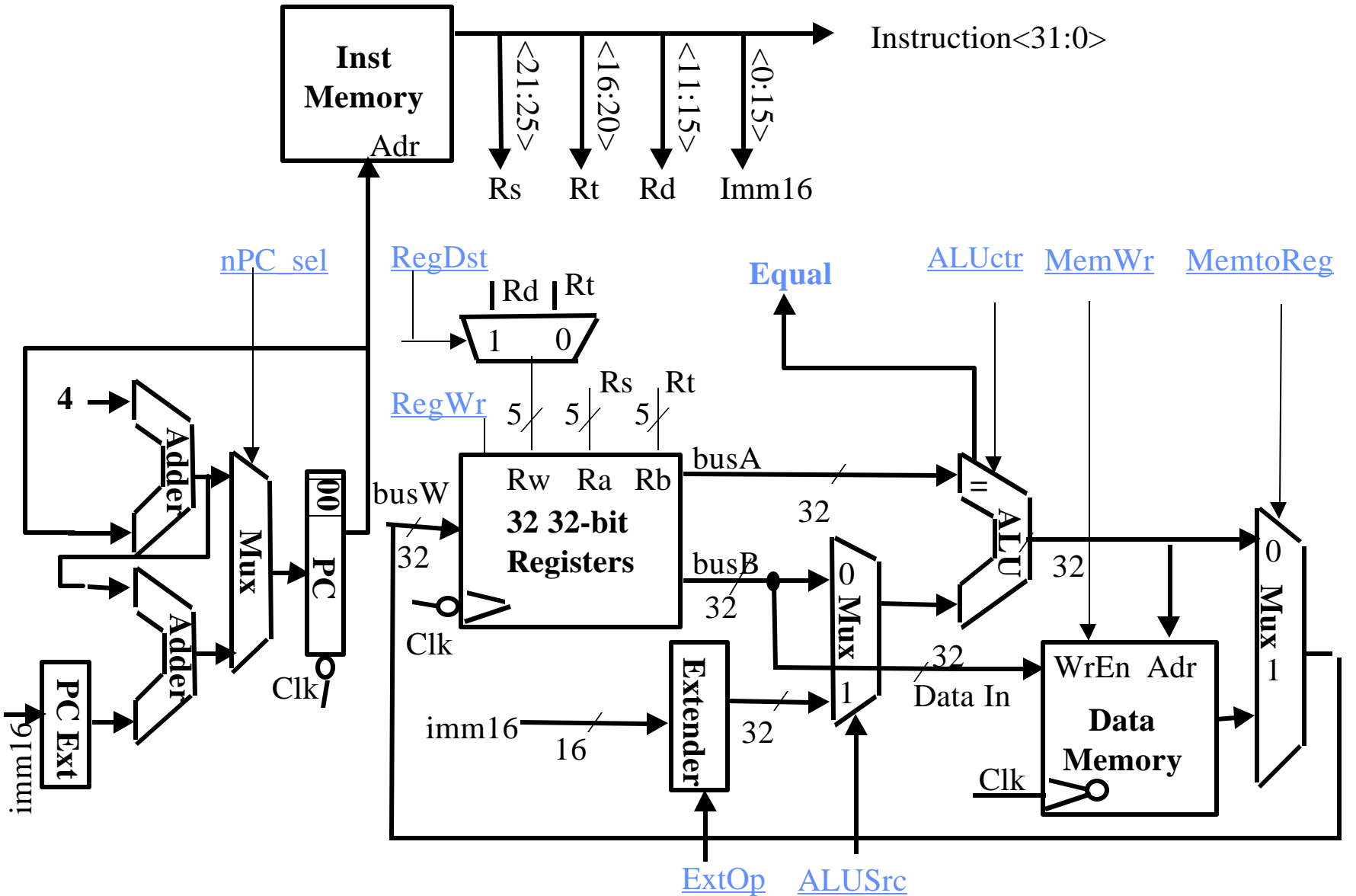


CPU Design Steps

1. Analyze instruction set operations using independent RTN => datapath requirements.
2. Select required datapath components & establish clock methodology.
3. Assemble datapath meeting the requirements.
4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.
5. Assemble the control logic.

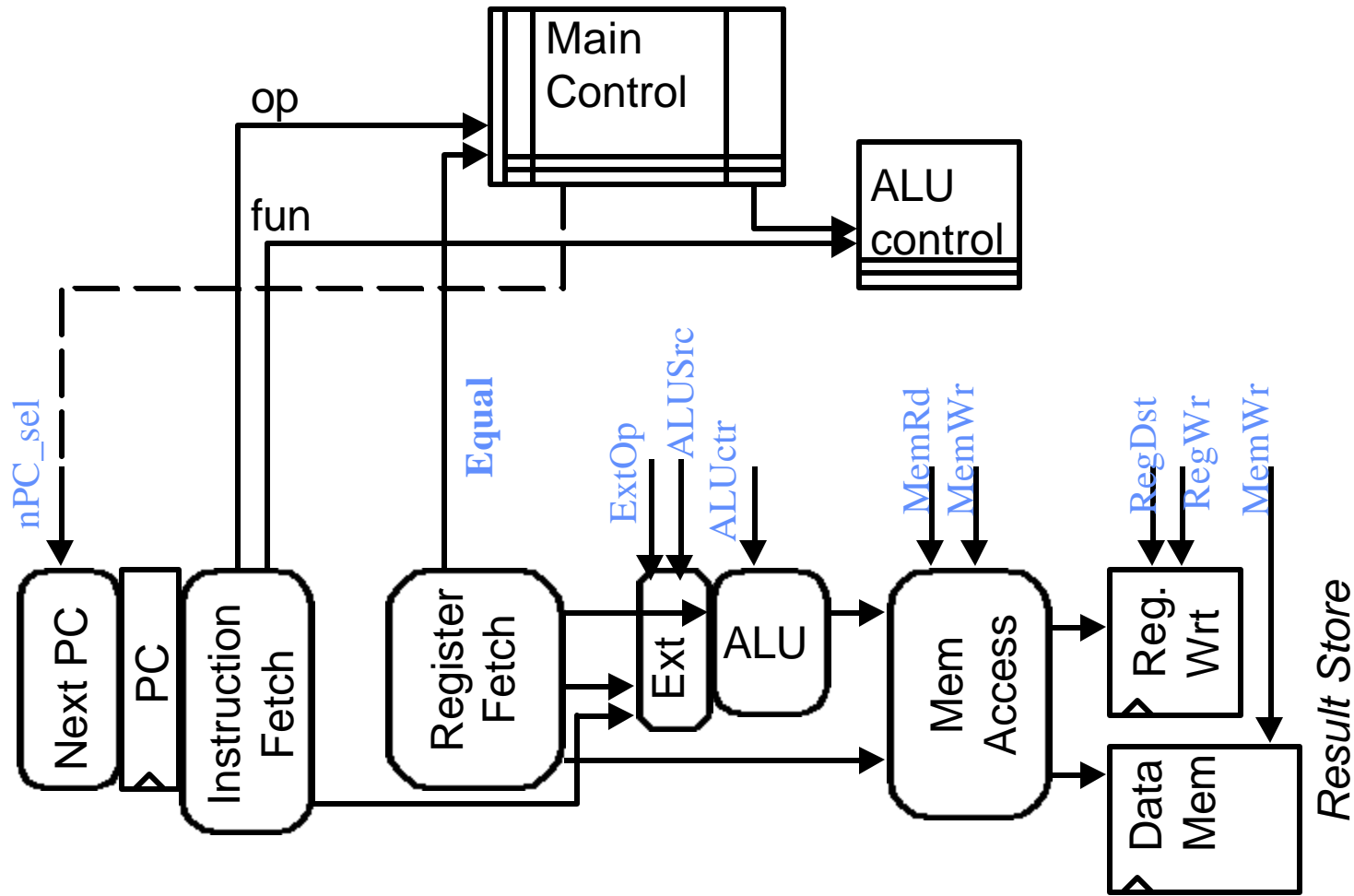
Single Cycle MIPS Datapath: CPI = 1, Long Clock Cycle



Drawbacks of Single-Cycle Processor

- **Long cycle time.**
- **All instructions must take as much time as the slowest:**
 - **Cycle time for load is longer than needed for all other instructions.**
- **Real memory is not as well-behaved as idealized memory**
 - **Cannot always complete data access in one (short) cycle.**
- **Cannot pipeline (overlap) the processing of one instruction with the previous instructions.**

Abstract View of Single Cycle CPU



Single Cycle Instruction Timing

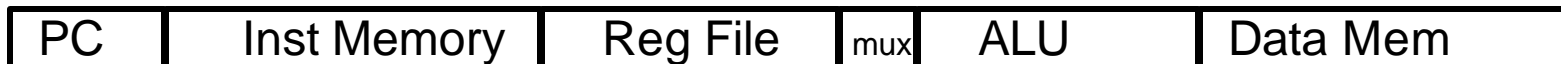
Arithmetic & Logical



Load



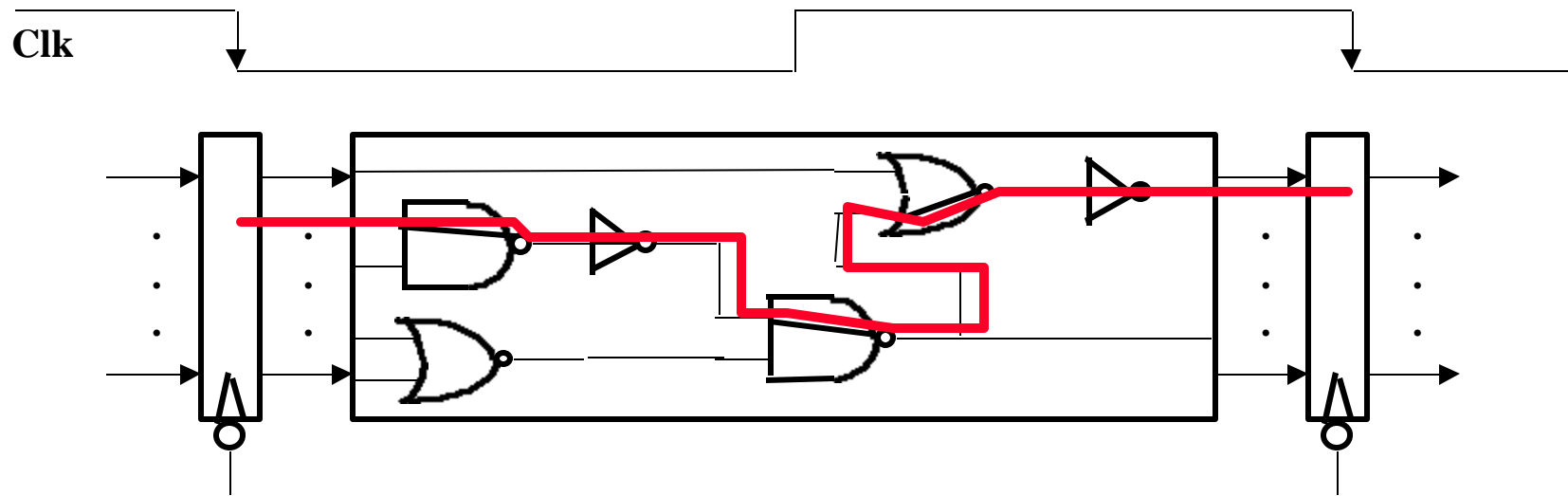
Store



Branch



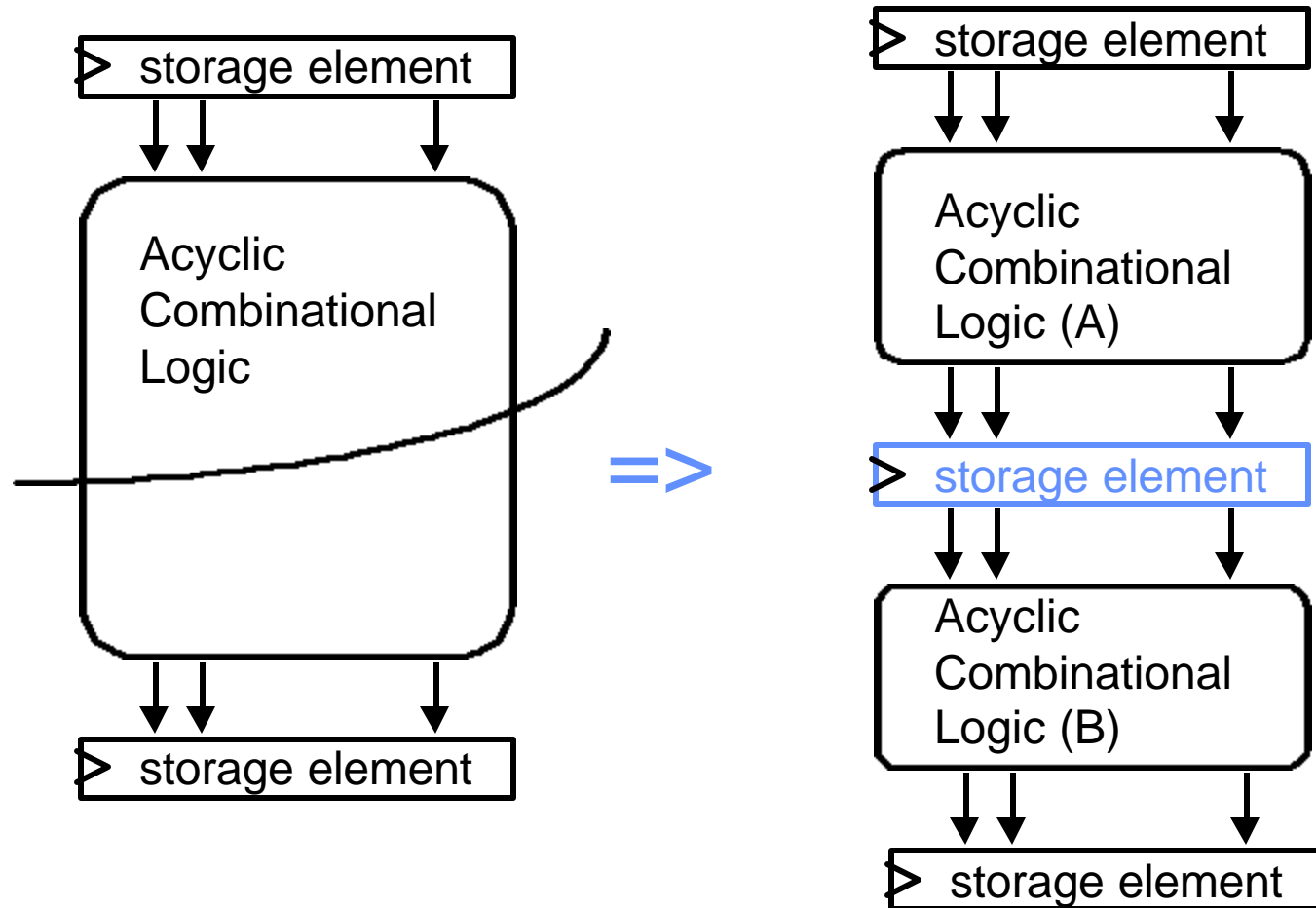
Clock Cycle Time & Critical Path



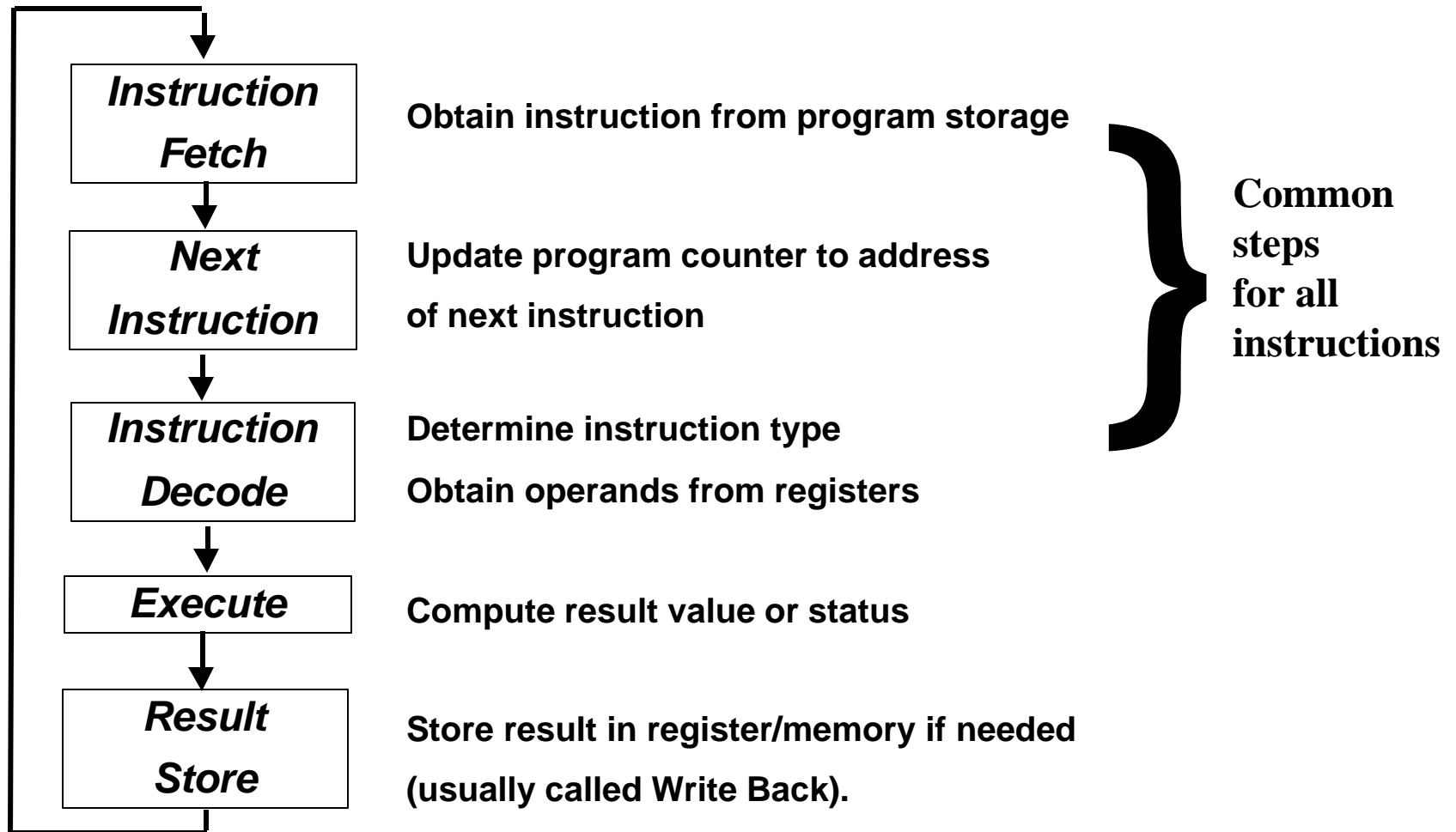
- **Critical path: the slowest path between any two storage devices**
- **Cycle time is a function of the critical path, must be greater than:**
 - **Clock-to-Q + Longest Path through the Combination Logic + Setup**

Reducing Cycle Time: Multi-Cycle Design

- Cut combinational dependency graph by inserting registers / latches.
- The same work is done in two or more fast cycles, rather than one slow cycle.

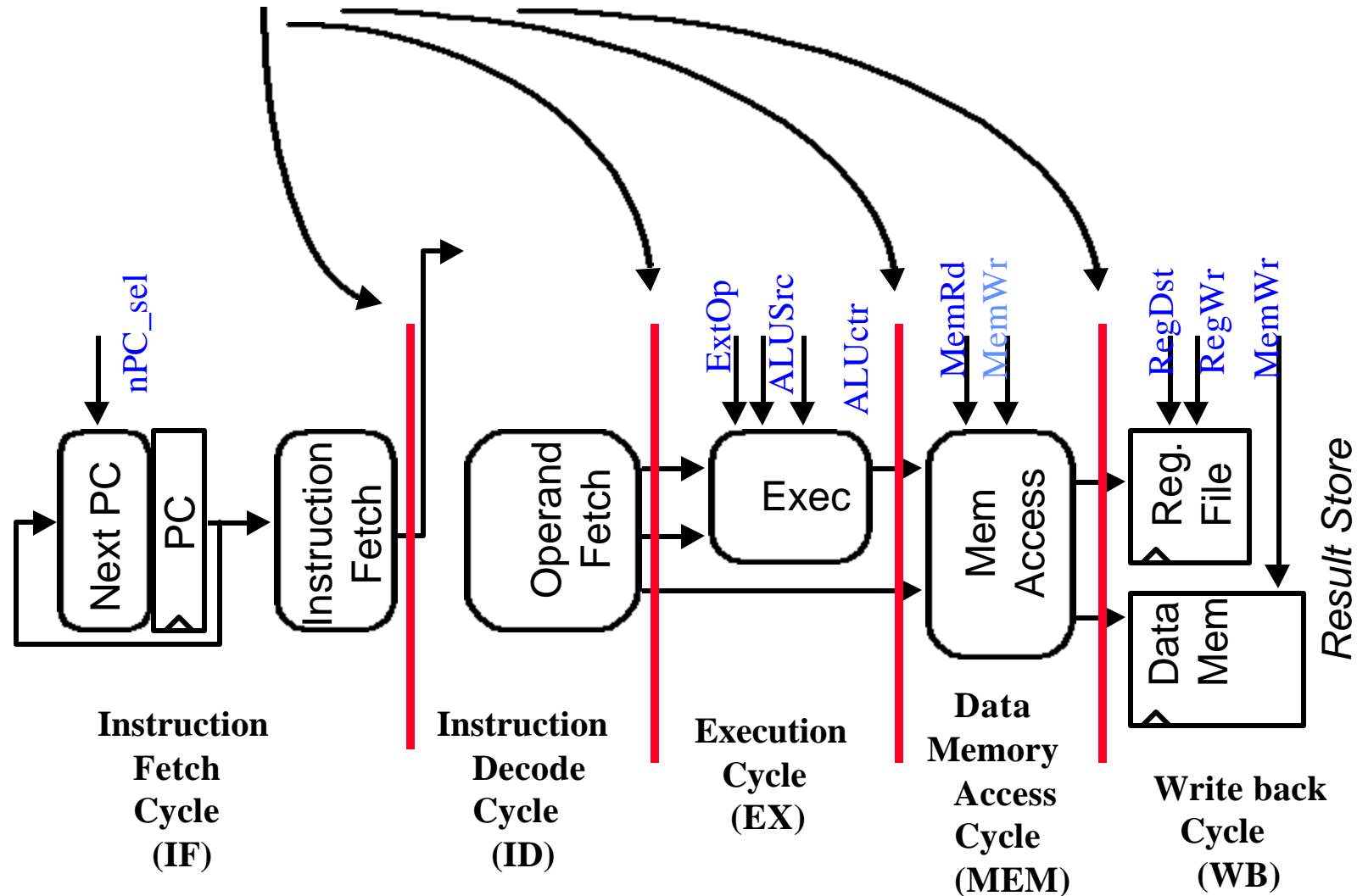


Instruction Processing Steps

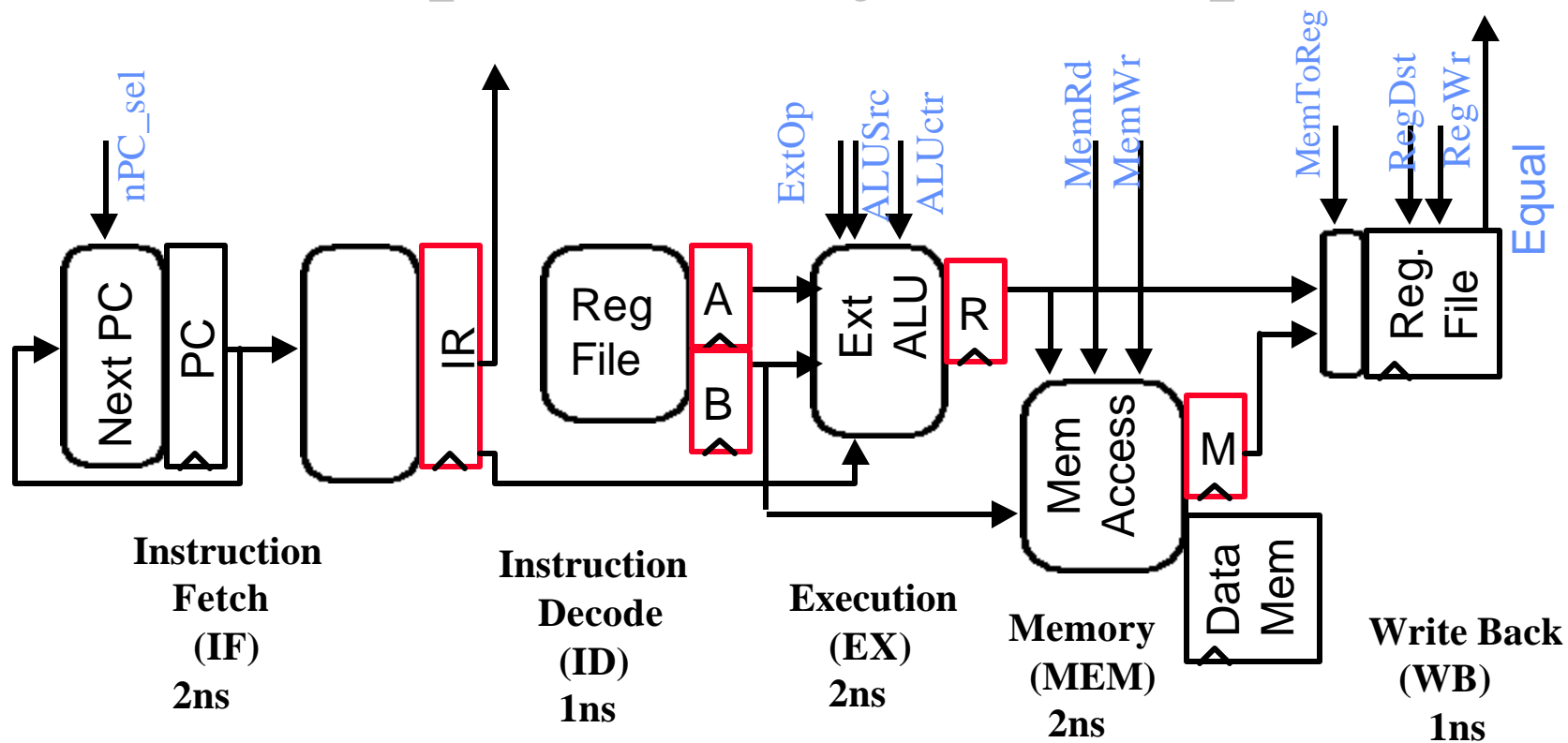


Partitioning The Single Cycle Datapath

Add registers between steps to break into cycles



Example Multi-cycle Datapath



Registers added: (not shown register write enable control lines)

IR: Instruction register

A, B: Two registers to hold operands read from register file.

R: or ALUOut, holds the output of the main ALU

M: or Memory data register (MDR) to hold data read from data memory

Cycle Time: Worst cycle delay = C = 2ns

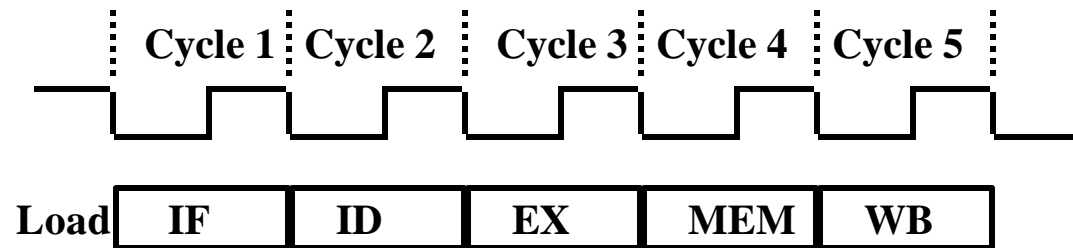
EECC550 - Shaaban

Operations In Each Cycle

	R-Type	Logic Immediate	Load	Store	Branch
IF	Instruction Fetch $IR \rightarrow Mem[PC]$	$IR \rightarrow Mem[PC]$	$IR \rightarrow Mem[PC]$	$IR \rightarrow Mem[PC]$	$IR \rightarrow Mem[PC]$
ID	Instruction Decode $A \rightarrow R[rs]$ $B \rightarrow R[rt]$	$A \rightarrow R[rs]$	$A \rightarrow R[rs]$	$A \rightarrow R[rs]$ $B \rightarrow R[rt]$	$A \rightarrow R[rs]$ $B \rightarrow R[rt]$
EX	Execution $R \rightarrow A + B$	$R \rightarrow A \text{ OR } ZeroExt[imm16]$	$R \rightarrow A + SignEx(Imm16)$	$R \rightarrow A + SignEx(Imm16)$	If Equal = 1 $PC \rightarrow PC + 4 +$ $(SignExt(imm16) \times 4)$ else $PC \rightarrow PC + 4$
MEM	Memory		$M \rightarrow Mem[R]$	$Mem[R] \rightarrow B$ $PC \rightarrow PC + 4$	
WB	Write Back $R[rd] \rightarrow R$ $PC \rightarrow PC + 4$	$R[rt] \rightarrow R$ $PC \rightarrow PC + 4$	$R[rt] \rightarrow M$ $PC \rightarrow PC + 4$		

MIPS Multi-Cycle Datapath:

Five Cycles of Load

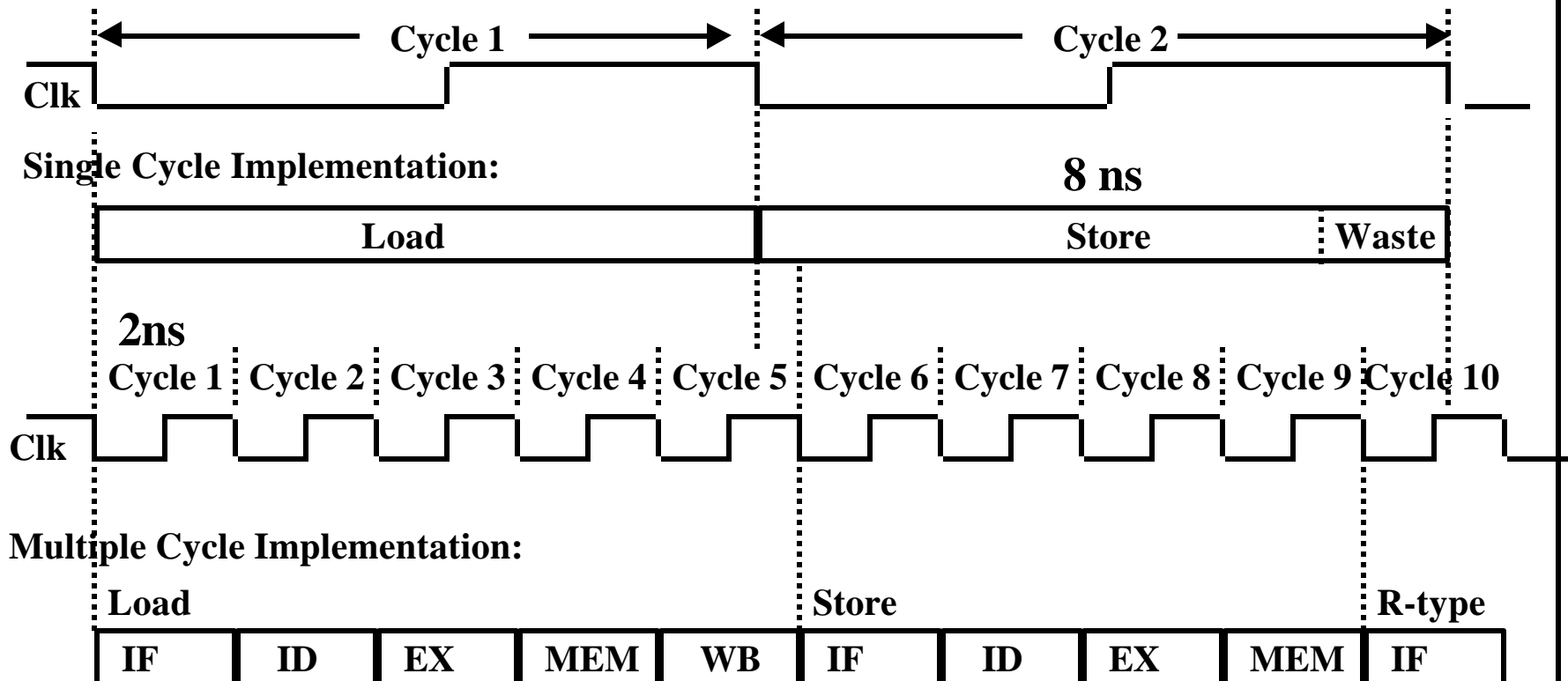


- 1- Instruction Fetch (IF) Instruction Fetch**
 - Fetch the instruction from the Instruction Memory.
- 2- Instruction Decode (ID): Registers Fetch and Instruction Decode.**
- 3- Execute (EX): Calculate the effective memory address.**
- 4- Memory (MEM): Read the data from the Data Memory.**
- 5- Write Back (WB): Write the data back to the register file. Update PC.**

Multi-cycle Datapath Instruction CPI

- **R-Type/Immediate: Require four cycles, CPI =4**
 - IF, ID, EX, WB
- **Loads: Require five cycles, CPI = 5**
 - IF, ID, EX, MEM, WB
- **Stores: Require four cycles, CPI = 4**
 - IF, ID, EX, MEM
- **Branches: Require three cycles, CPI = 3**
 - IF, ID, EX
- **Average program $3 \leq \text{CPI} \leq 5$ depending on program profile (instruction mix).**

Single Cycle Vs. Multi-Cycle CPU



Single-Cycle CPU:

$$CPI = 1 \quad C = 8ns$$

One million instructions take =

$$I \times CPI \times C = 10^6 \times 1 \times 8 \times 10^{-9} = 8 \text{ msec}$$

Multi-Cycle CPU:

$$CPI = 3 \text{ to } 5 \quad C = 2ns$$

One million instructions take from

$$10^6 \times 3 \times 2 \times 10^{-9} = 6 \text{ msec}$$

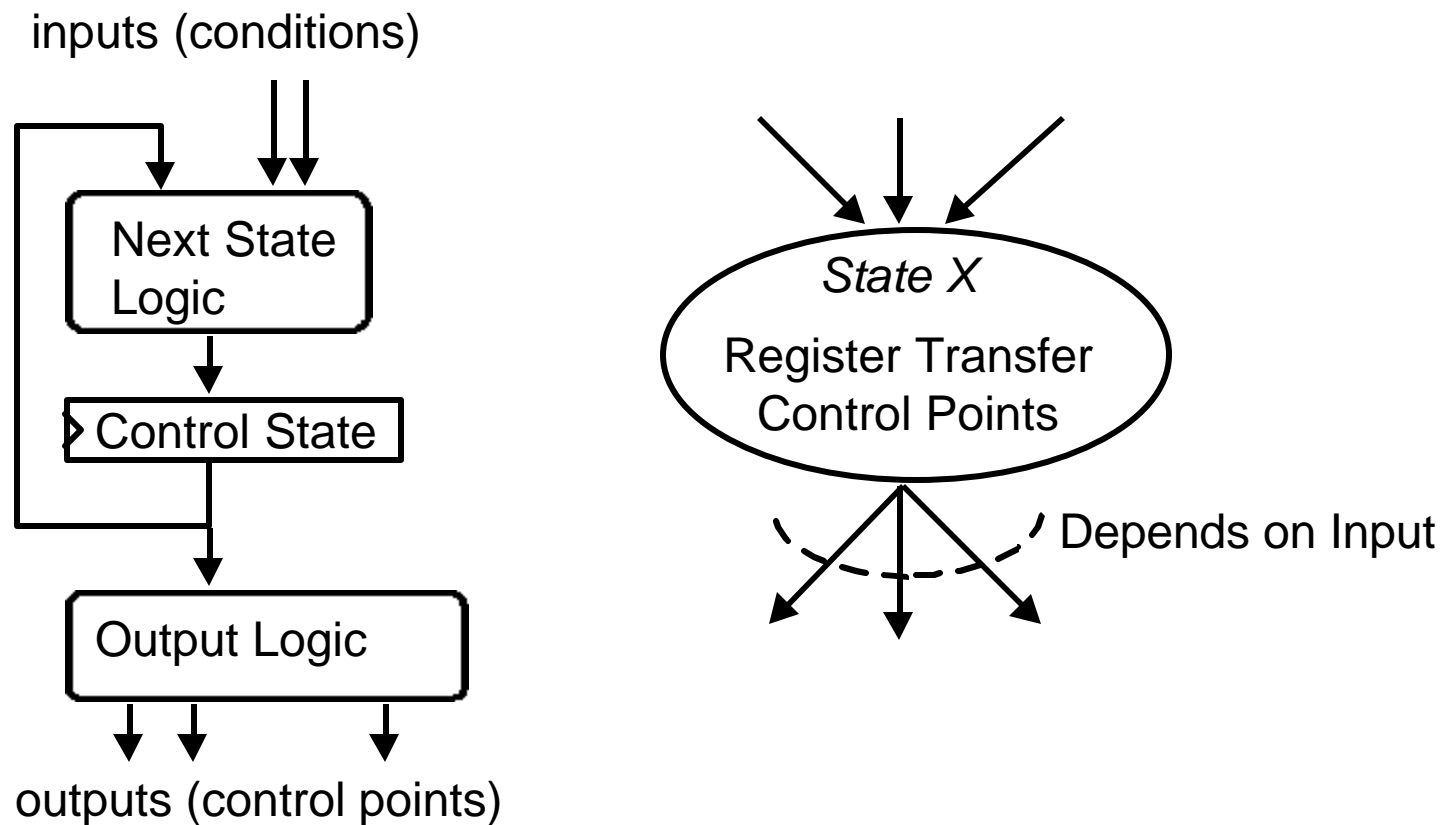
to $10^6 \times 5 \times 2 \times 10^{-9} = 10 \text{ msec}$

depending on instruction mix used.

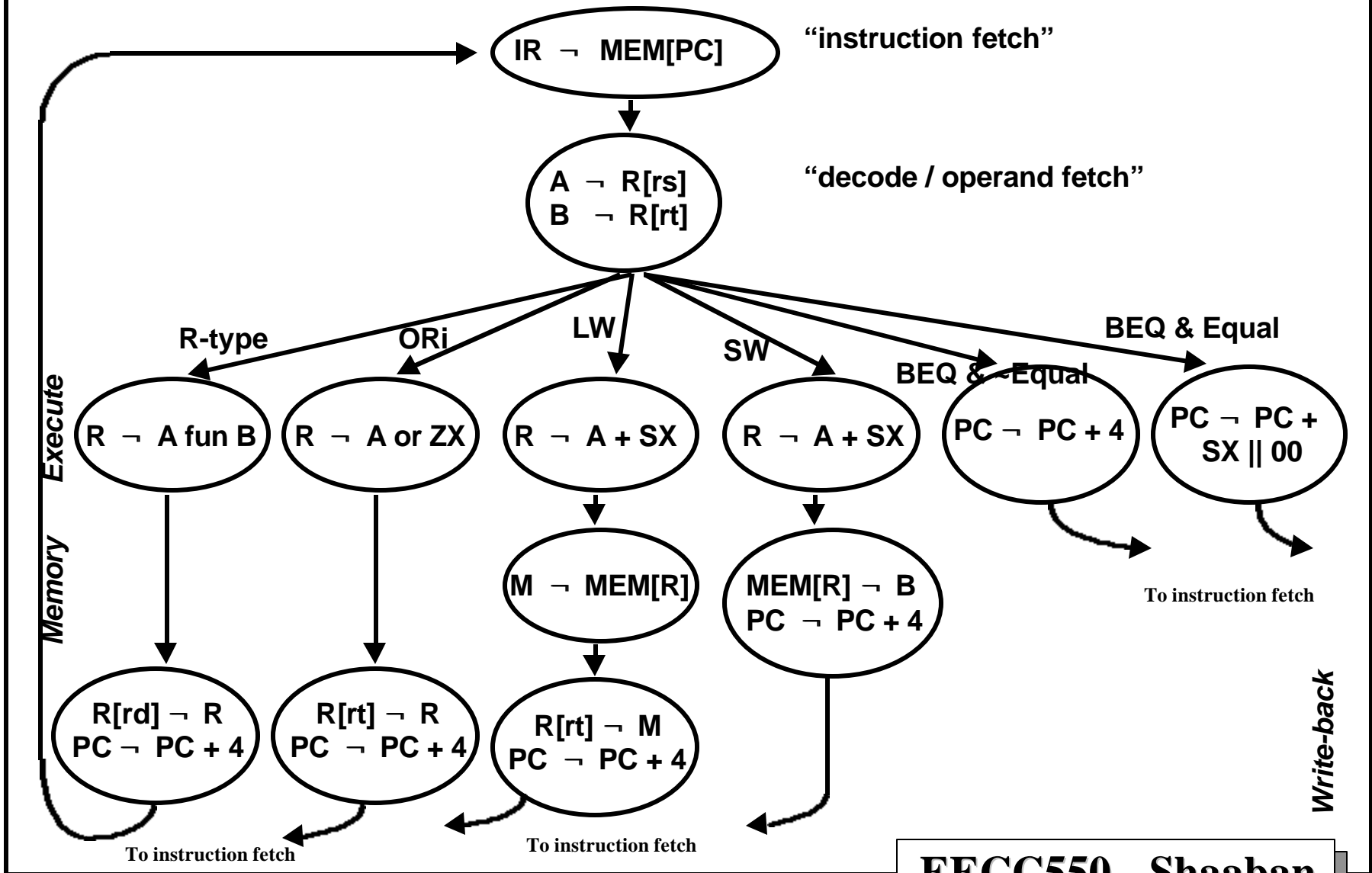
EECC550 - Shaaban

Finite State Machine (FSM) Control Model

- State specifies control points for Register Transfer.
- Transfer occurs upon exiting state (falling edge).



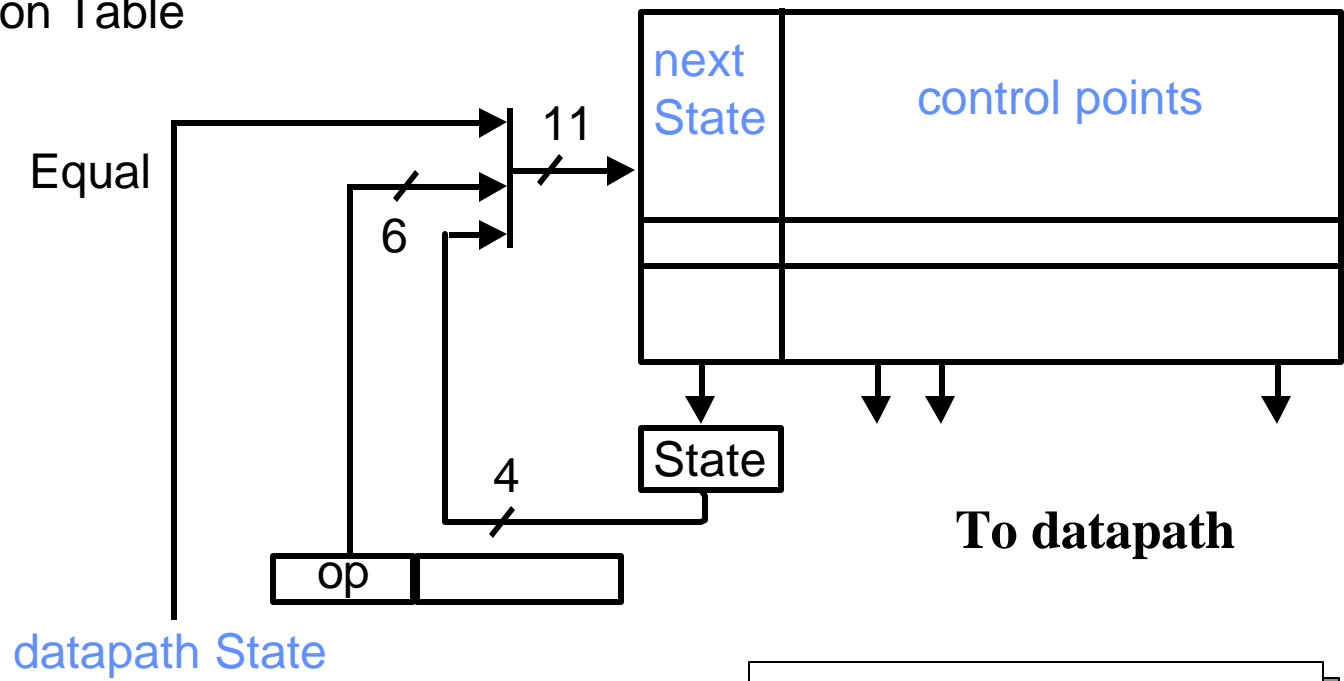
Control Specification For Multi-cycle CPU Finite State Machine (FSM)



Traditional FSM Controller

state	op	cond	next state	control points

Truth or Transition Table

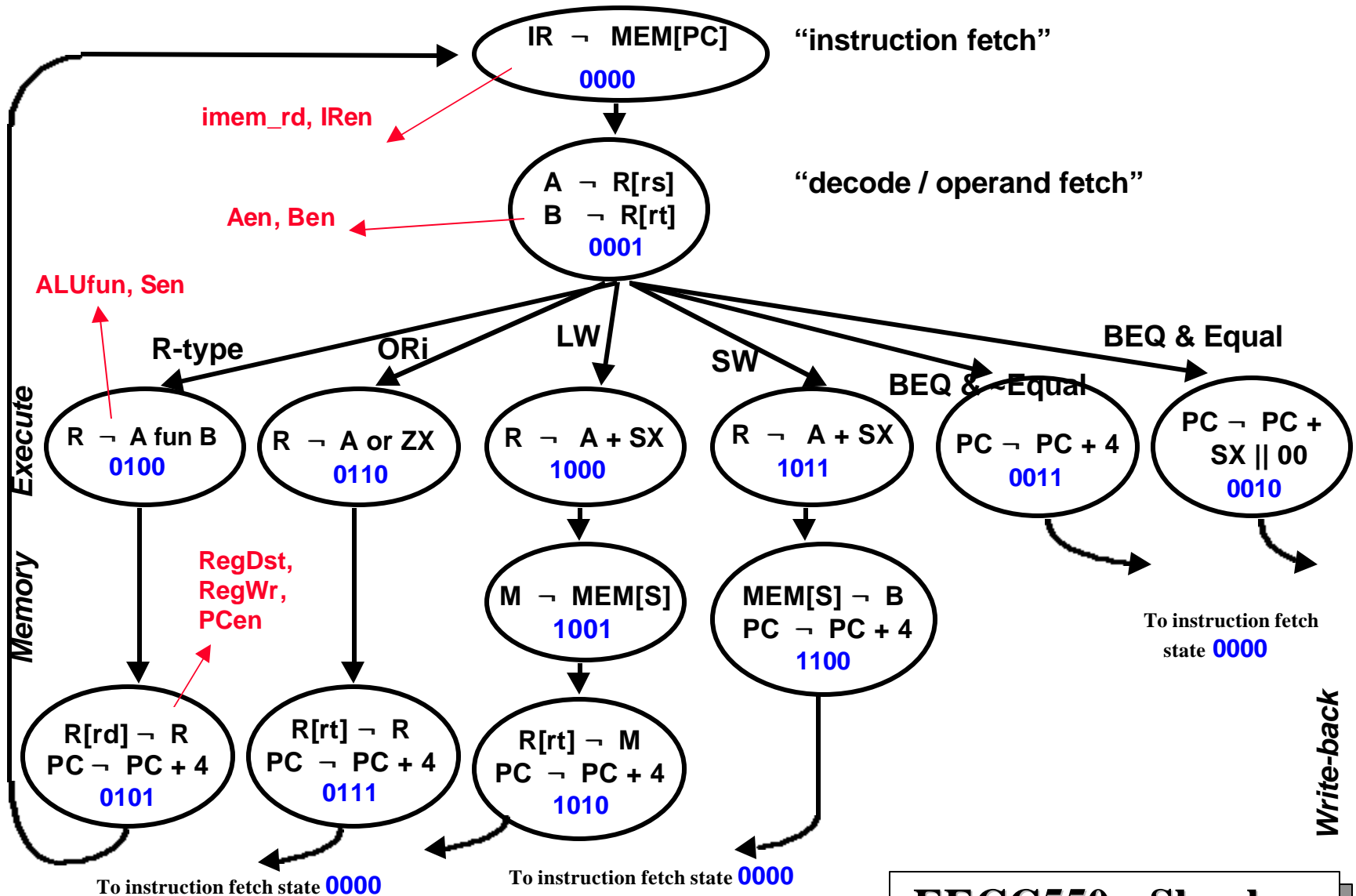


Traditional FSM Controller

datapath + state diagram \Rightarrow control

- **Translate RTN statements into control points.**
- **Assign states.**
- **Implement the controller.**

Mapping RTNs To Control Points Examples & State Assignments

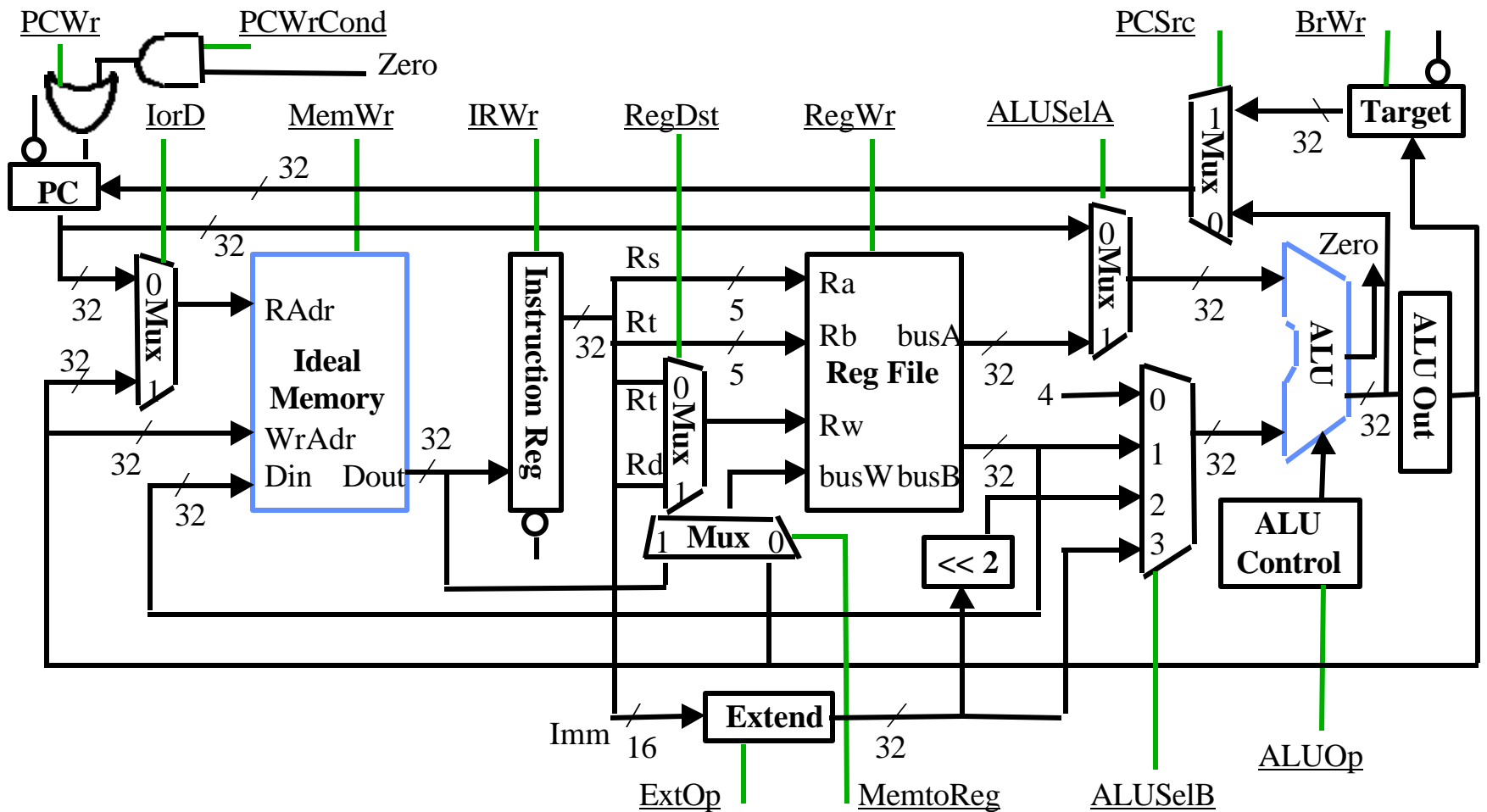


Detailed Control Specification

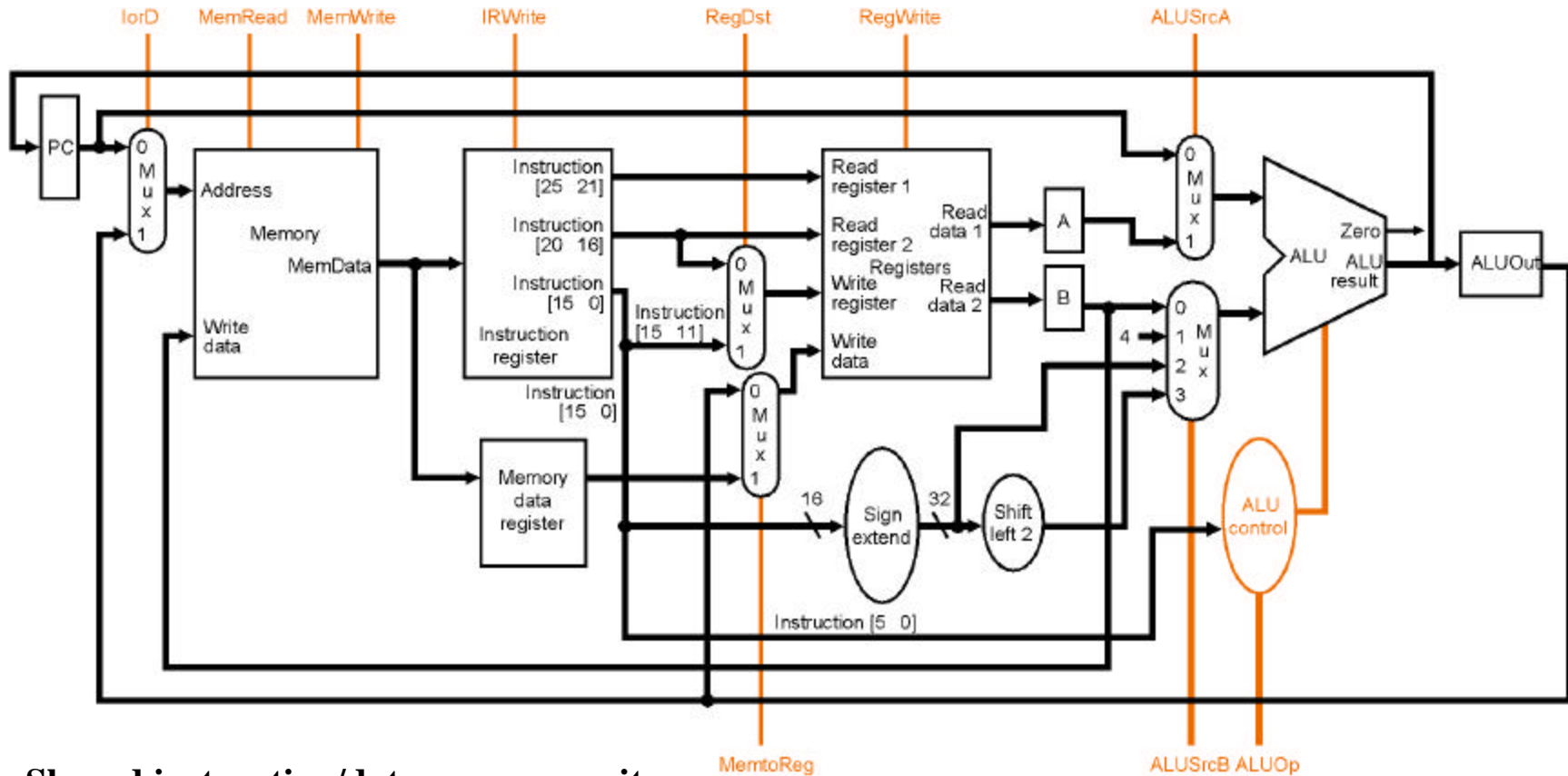
	Current State	Op field	Eq	Next	IR	PC en sel	Ops A B	Exec Ex Sr ALU S	Mem R W M	Write-Back M-R Wr Dst
IF	0000	??????	?	0001	1					
ID	0001	BEQ	0	0011			1 1			
	0001	BEQ	1	0010			1 1			
	0001	R-type	x	0100			1 1			
	0001	orI	x	0110			1 1			
	0001	LW	x	1000			1 1			
	0001	SW	x	1011			1 1			
	BEQ	0010	xxxxxx	x	0000		1 1			
0011		xxxxxx	x	0000		1 0				
R	0100	xxxxxx	x	0101				0 1 fun 1		
	0101	xxxxxx	x	0000		1 0				0 1 1
ORI	0110	xxxxxx	x	0111				0 0 or 1		
	0111	xxxxxx	x	0000		1 0				0 1 0
LW	1000	xxxxxx	x	1001				1 0 add 1		
	1001	xxxxxx	x	1010					1 0 1	
	1010	xxxxxx	x	0000		1 0				1 1 0
SW	1011	xxxxxx	x	1100				1 0 add 1		
	1100	xxxxxx	x	0000		1 0			0 1	

Alternative Multiple Cycle Datapath (In Textbook)

- Miminizes Hardware: 1 memory, 1 ALU

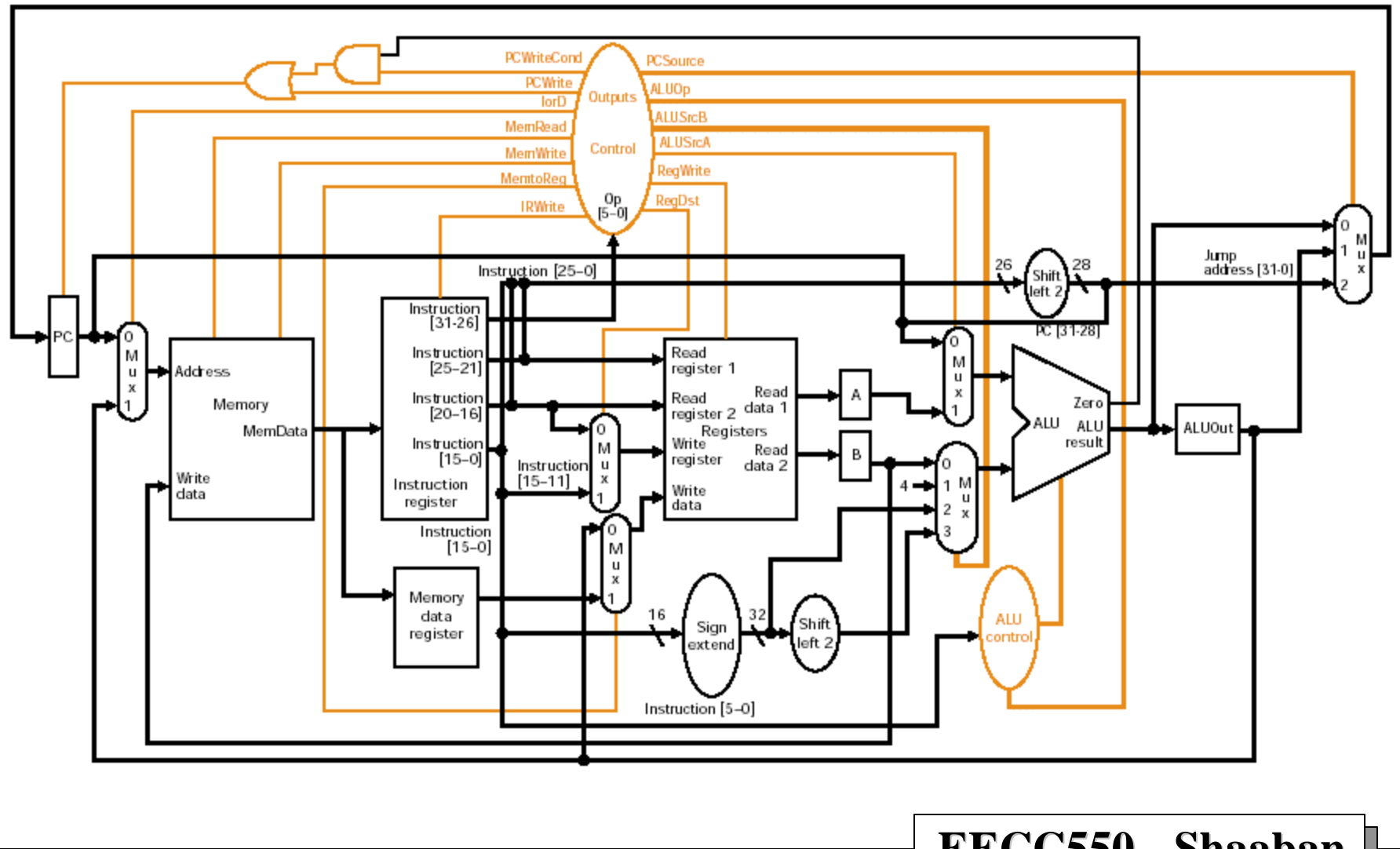


Alternative Multiple Cycle Datapath (In Textbook)



- Shared instruction/data memory unit
- A single ALU shared among instructions
- Shared units require additional or widened multiplexors
- Temporary registers to hold data between clock cycles of the instruction:
 - Additional registers: Instruction Register (IR), Memory Data Register (MDR), A, B, ALUOut

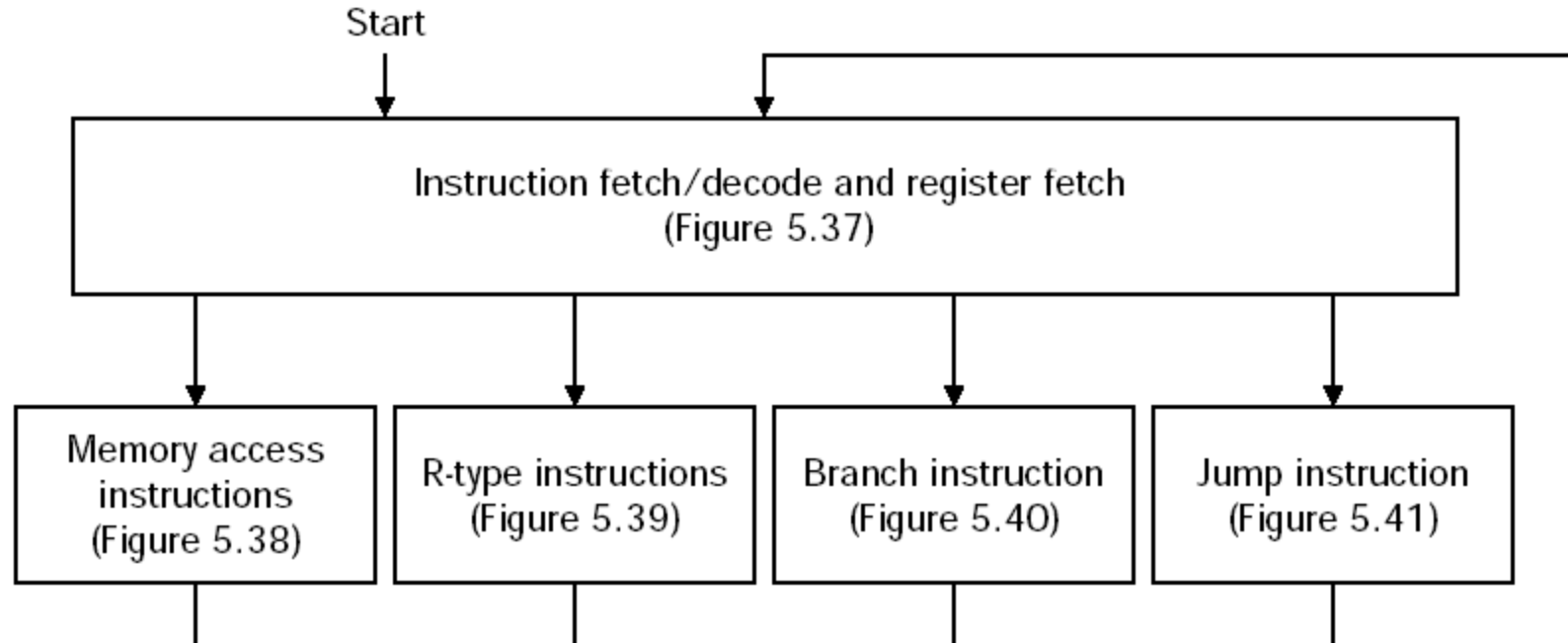
Alternative Multiple Cycle Datapath With Control Lines (Fig 5.33 In Textbook)



Operations In Each Cycle

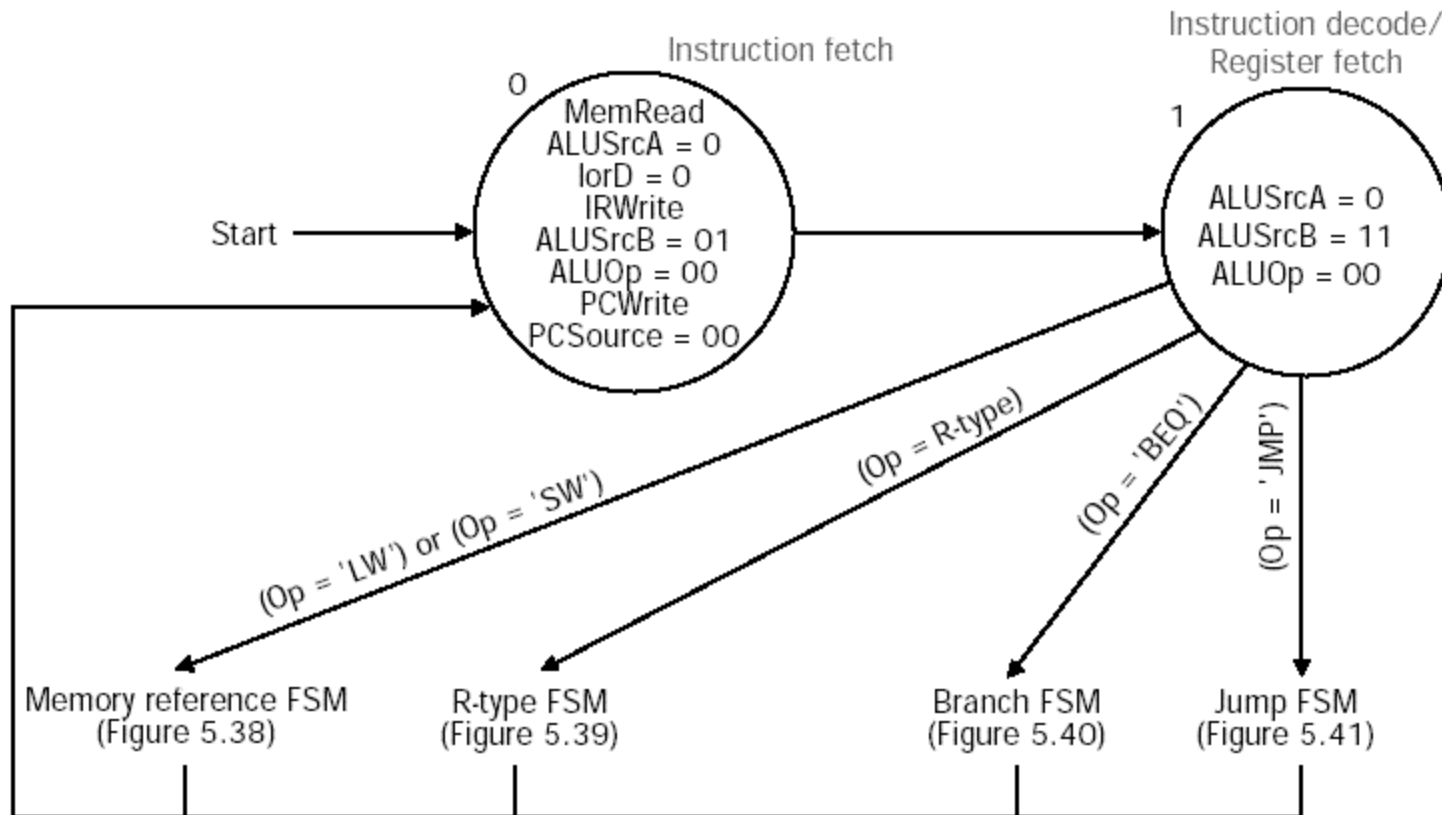
	R-Type	Logic Immediate	Load	Store	Branch
Instruction Fetch	IR \rightarrow Mem[PC] PC \rightarrow PC + 4	IR \rightarrow Mem[PC] PC \rightarrow PC + 4	IR \rightarrow Mem[PC] PC \rightarrow PC + 4	IR \rightarrow Mem[PC] PC \rightarrow PC + 4	IR \rightarrow Mem[PC] PC \rightarrow PC + 4
Instruction Decode	A \rightarrow R[rs] B \rightarrow R[rt] ALUout \rightarrow PC + (SignExt(imm16) x4)	A \rightarrow R[rs] B \rightarrow R[rt] ALUout \rightarrow PC + (SignExt(imm16) x4)	A \rightarrow R[rs] B \rightarrow R[rt] ALUout \rightarrow PC + (SignExt(imm16) x4)	A \rightarrow R[rs] B \rightarrow R[rt] ALUout \rightarrow PC + (SignExt(imm16) x4)	A \rightarrow R[rs] B \rightarrow R[rt] ALUout \rightarrow PC + (SignExt(imm16) x4)
Execution	ALUout \rightarrow A + B	ALUout \rightarrow A OR ZeroExt[imm16]	ALUout \rightarrow A + SignEx(Im16)	ALUout \rightarrow A + SignEx(Im16)	If Equal = 1 PC \rightarrow ALUout
Memory			M \rightarrow Mem[ALUout]	Mem[ALUout] \rightarrow B	
Write Back	R[rd] \rightarrow ALUout	R[rt] \rightarrow ALUout	R[rt] \rightarrow Mem		

High-Level View of Finite State Machine Control

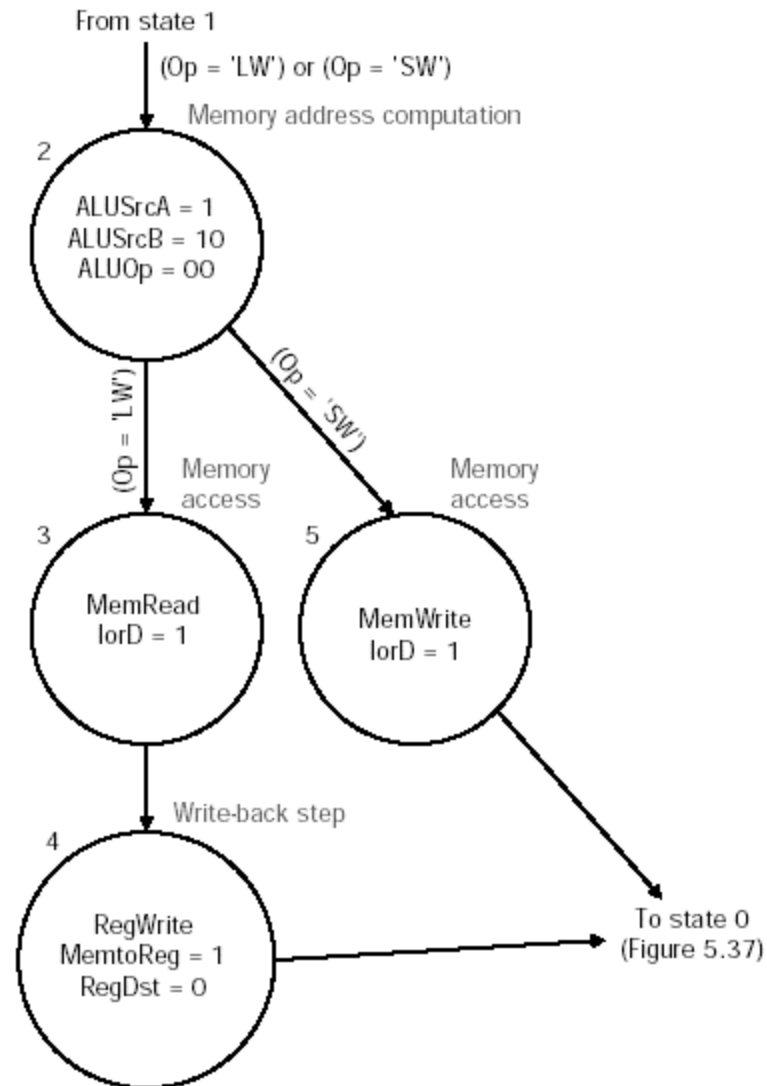


- **First steps are independent of the instruction class**
- **Then a series of sequences that depend on the instruction opcode**
- **Then the control returns to fetch a new instruction.**
- **Each box above represents one or several state.**

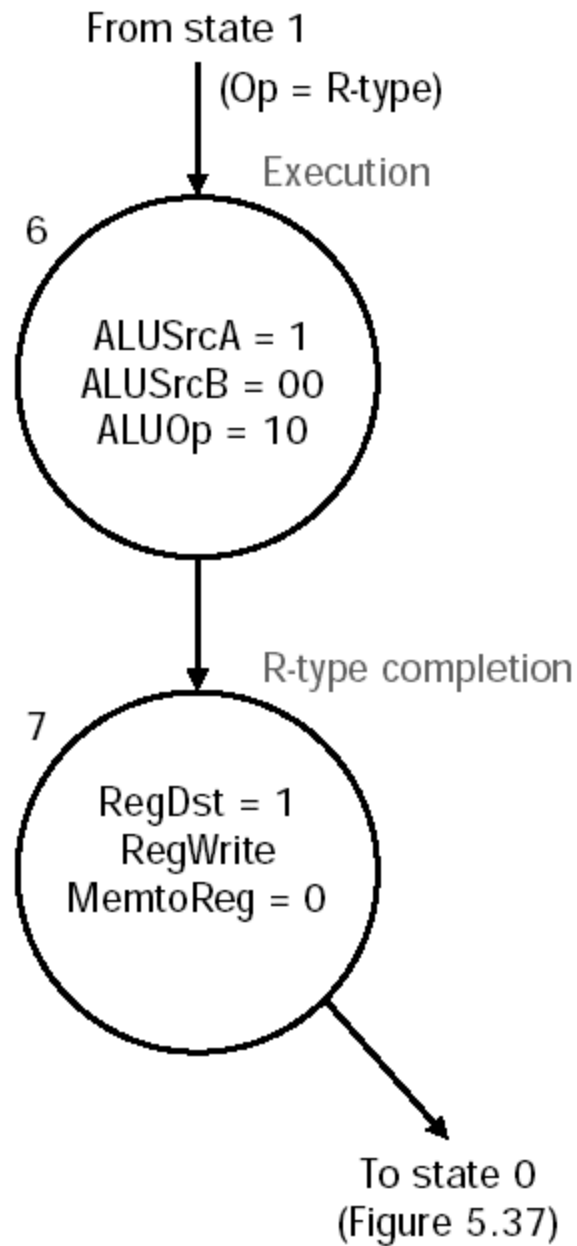
Instruction Fetch and Decode FSM States



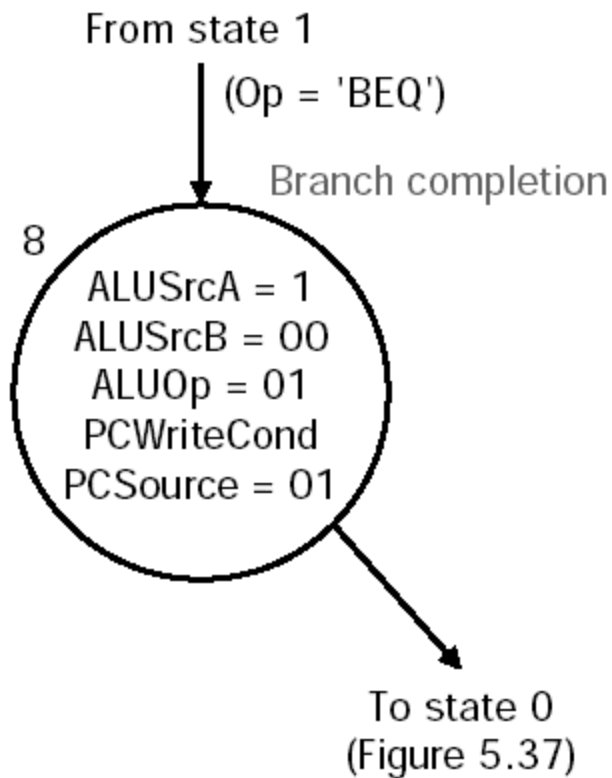
Load/Store Instructions FSM States



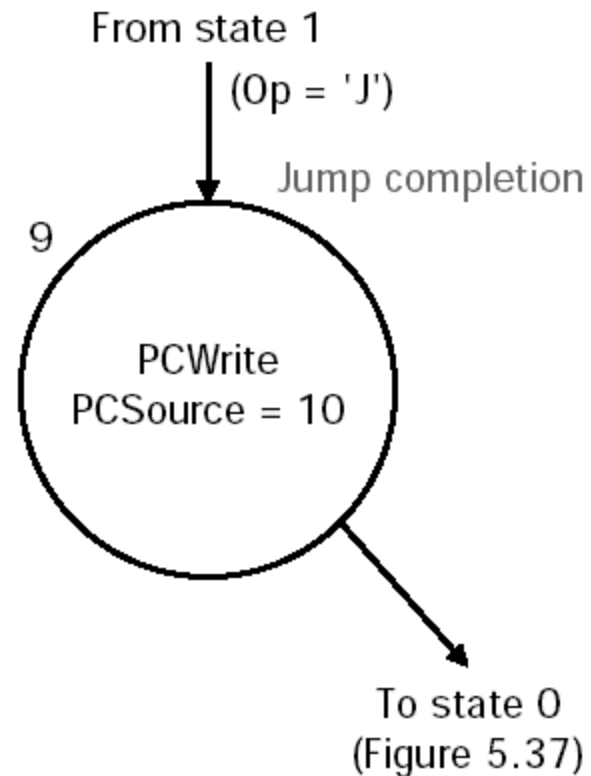
R-Type Instructions FSM States

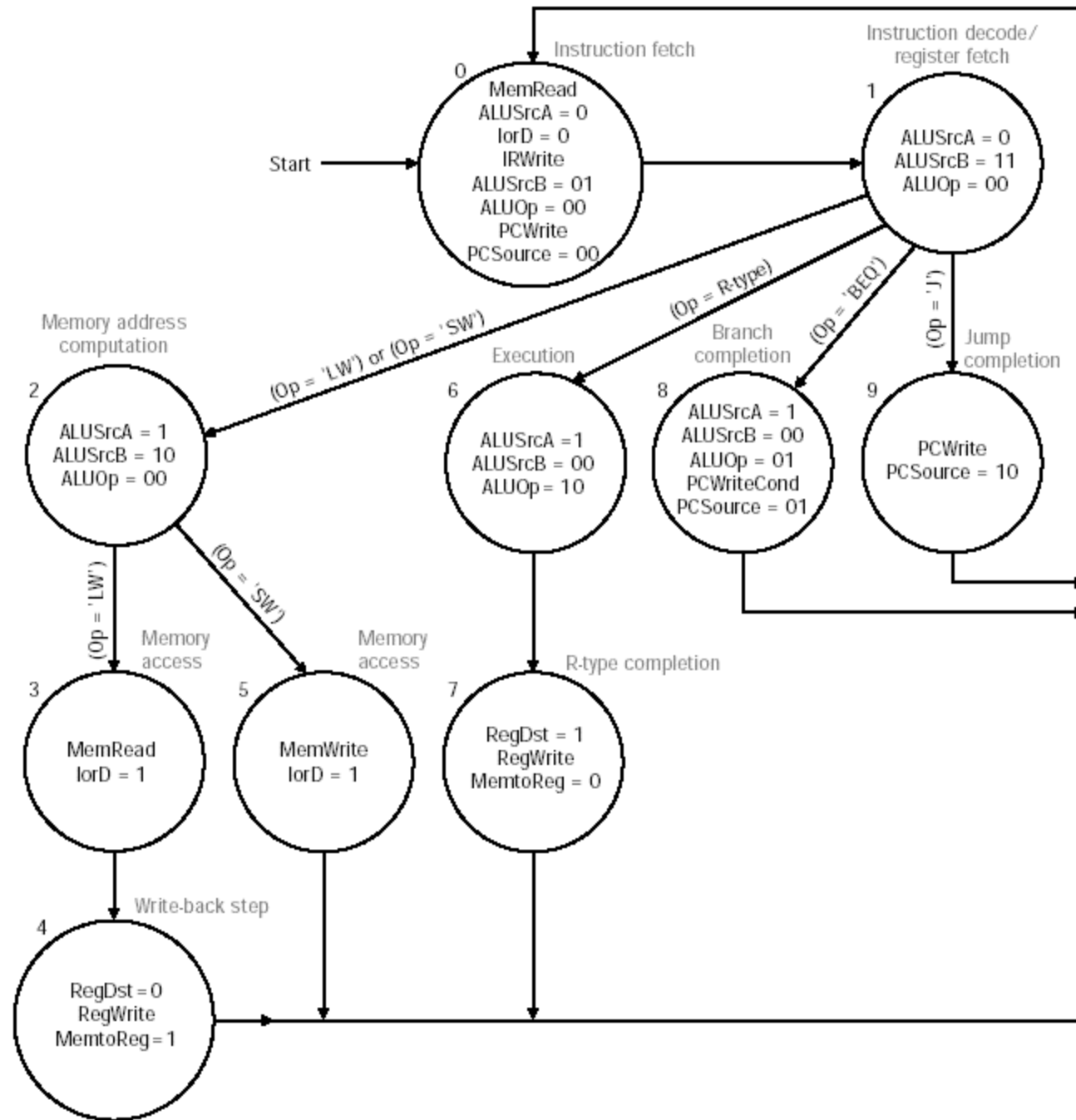


Branch Instruction Single State

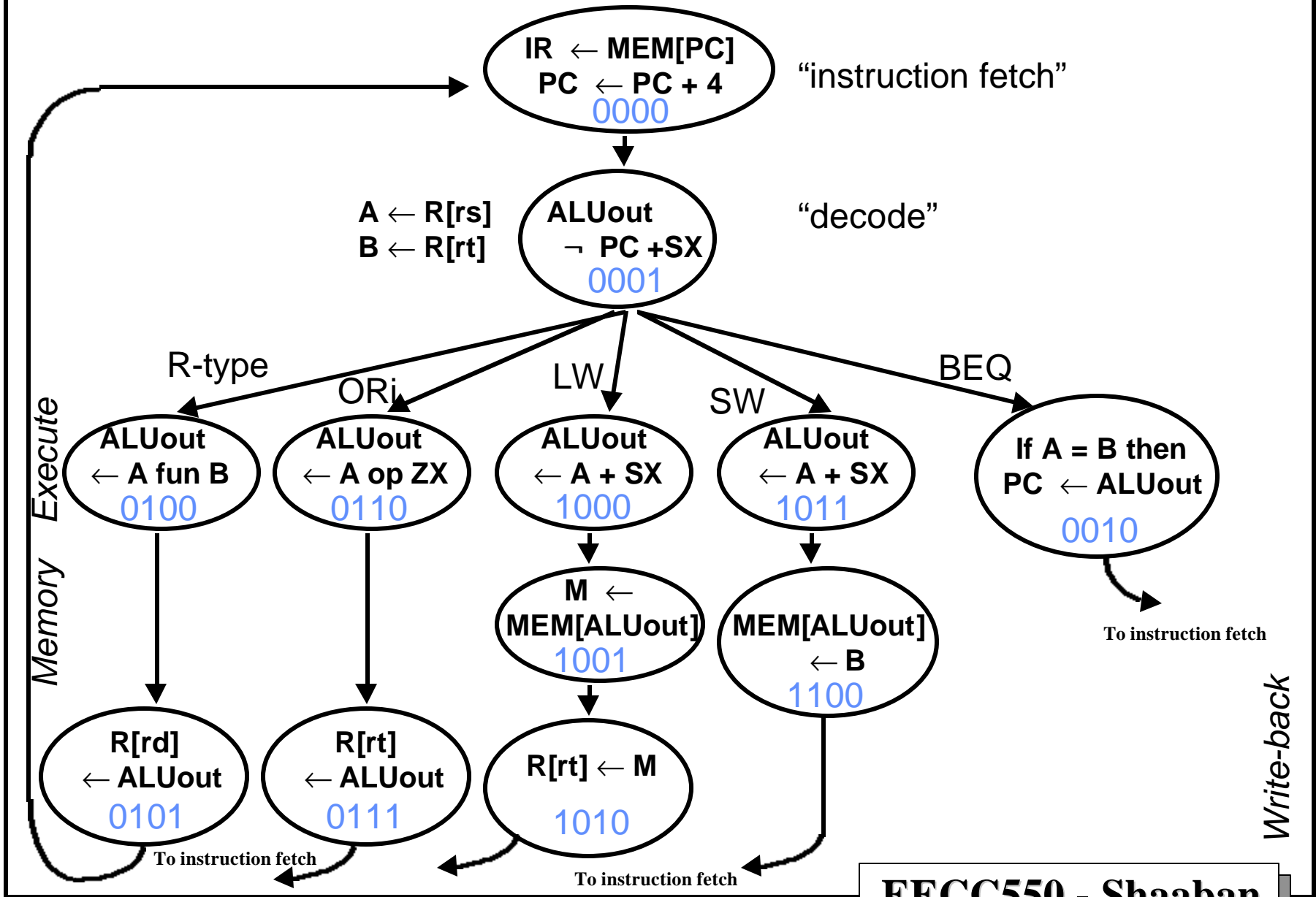


Jump Instruction Single State





Finite State Machine (FSM) Specification



MIPS Multi-cycle Datapath Performance Evaluation

- What is the average CPI?
 - State diagram gives CPI for each instruction type.
 - Workload (program) below gives frequency of each type.

Type	CPI _i for type	Frequency	CPI _i x freq _i
Arith/Logic	4	40%	1.6
Load	5	30%	1.5
Store	4	10%	0.4
branch	3	20%	0.6
Average CPI:			4.1

Better than CPI = 5 if all instructions took the same number of clock cycles (5).