Processor: Datapath and Control

[Control Unit for Multicycle Processor]
Book of David A. Patterson
Appendix D

Simple Questions

- How many cycles will it take to execute this code?

```
    lw $t2, 0($t3)
    lw $t3, 4($t3)
    beq $t2, $t3, Label #assume not equal
    add $t5, $t2, $t3
    sw $t5, 8($t3)
Label:  ...
```

- What is going on during the 8th cycle of execution?

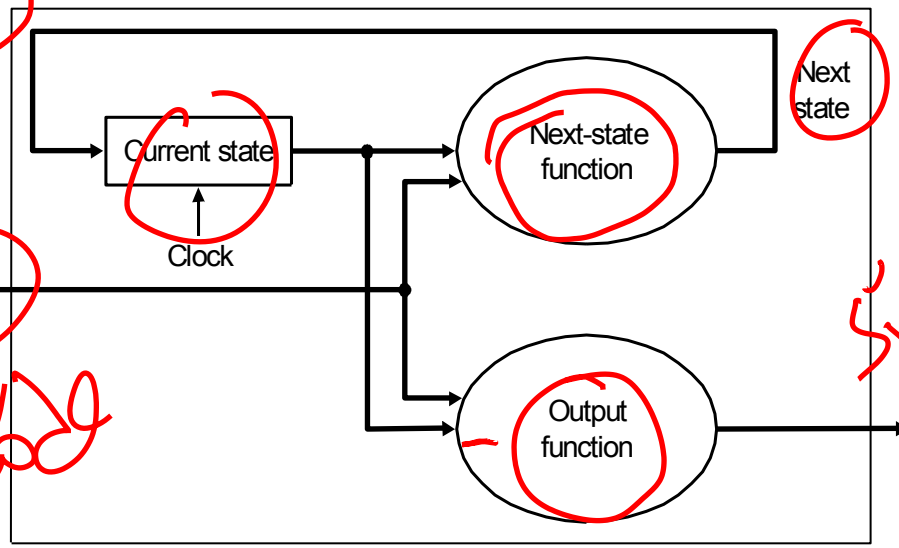
- In what cycle does the actual addition of $t2 and $t3 takes place?

Defining Control for Multicycle Processor

- ✔ The control is more complex because the instruction is executed in a series of steps.
- ✔ The control must specify the signal to be set in any step and the next step in the sequence.

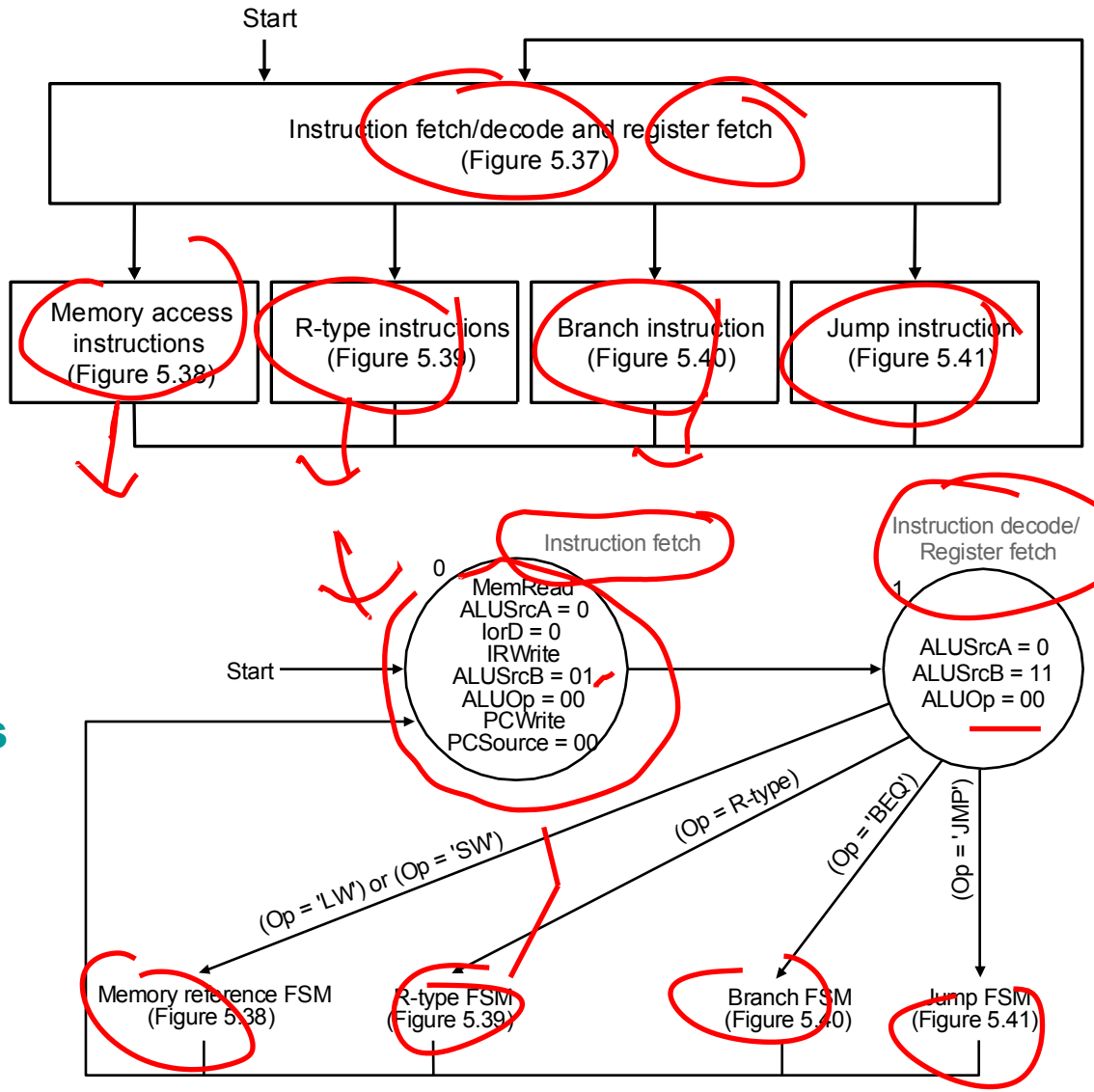
Finite State Machine (FSM) Control

- 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



  - It corresponds to the five steps of execution.
  - Each step of FSM will take 1 clk cycle.

FSM Control: High Level View

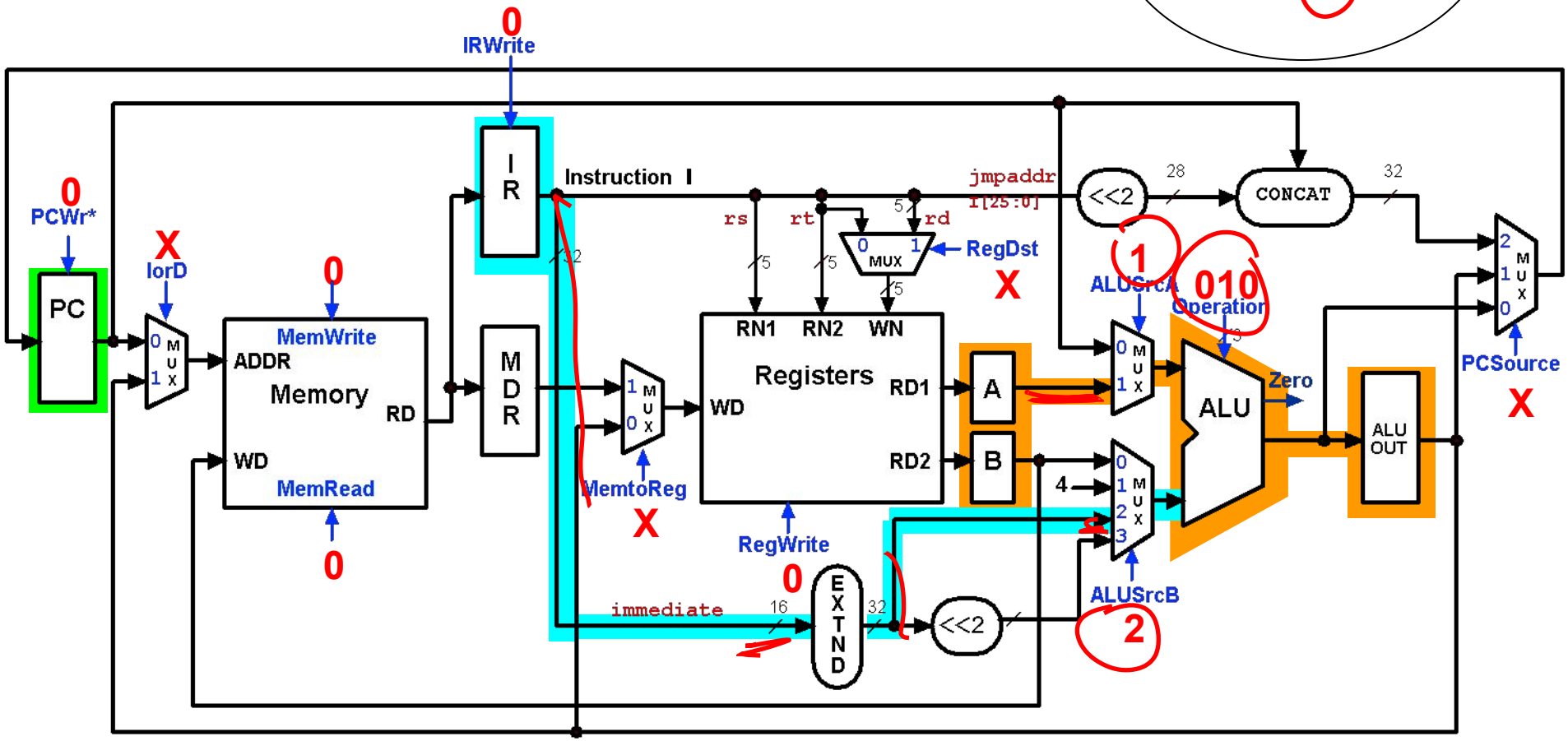


Asserted signals shown inside state circles

Step 3: Memory Reference Instructions

lw / sw

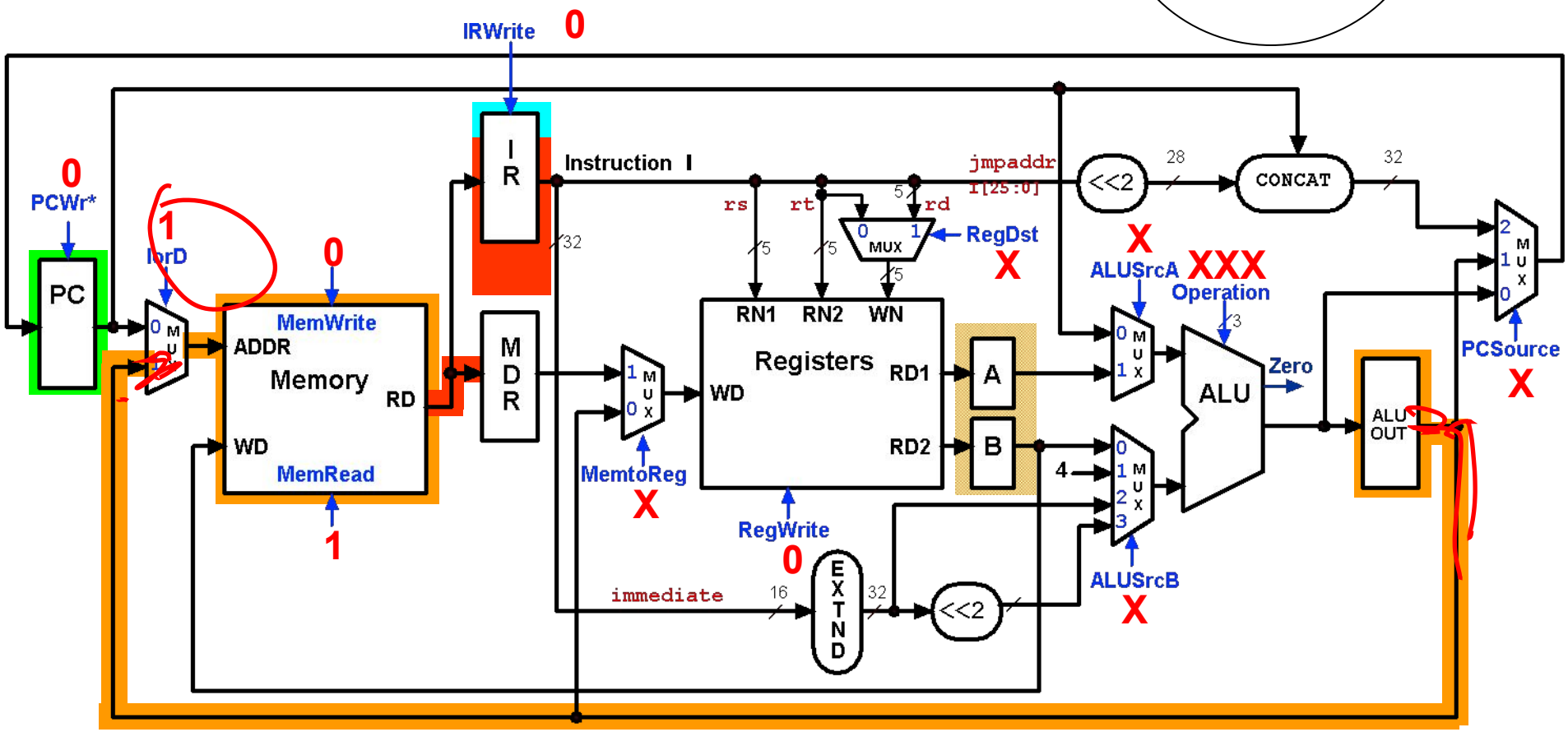
ALUSrcA=1
ALUSrcB=10
ALUOp=00



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

Step 4: Memory Access - Read (lw)

MemRead
IorD = 1

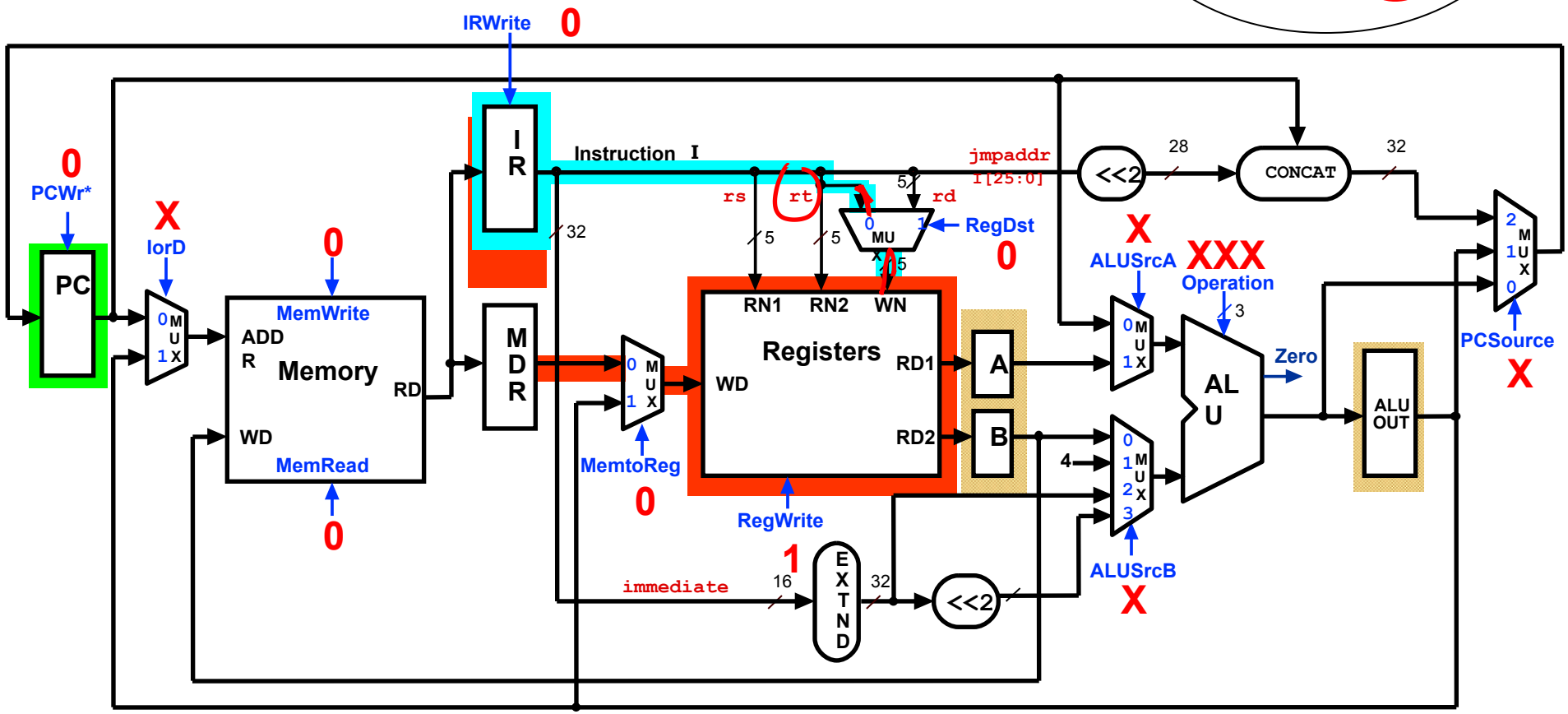


MDR = Memory[ALUOut];

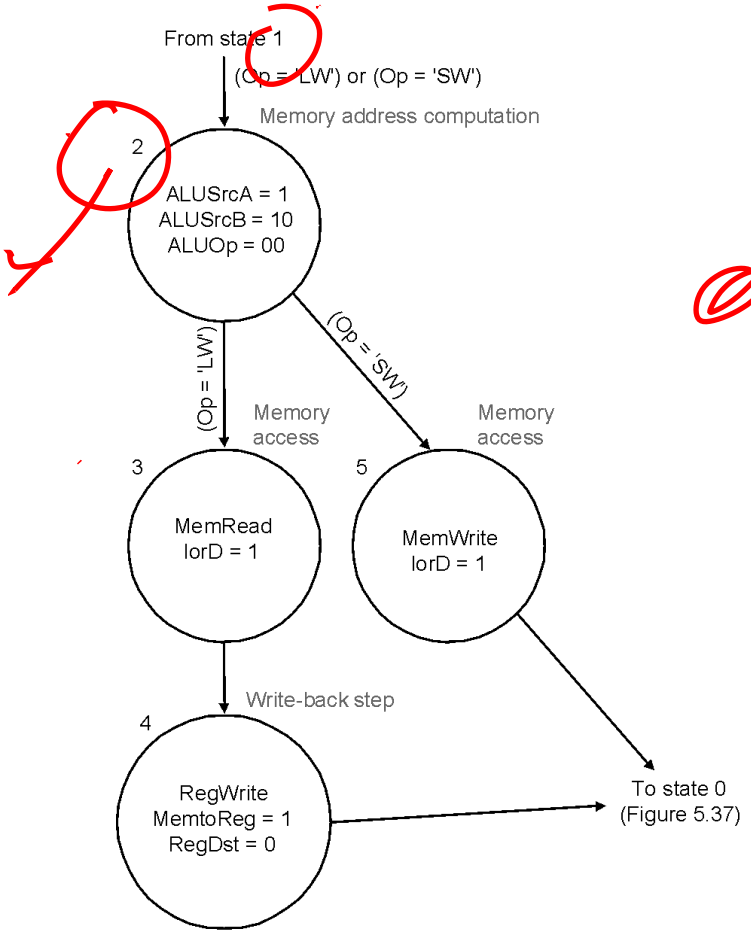
Step 5: Memory Read Completion (lw)

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

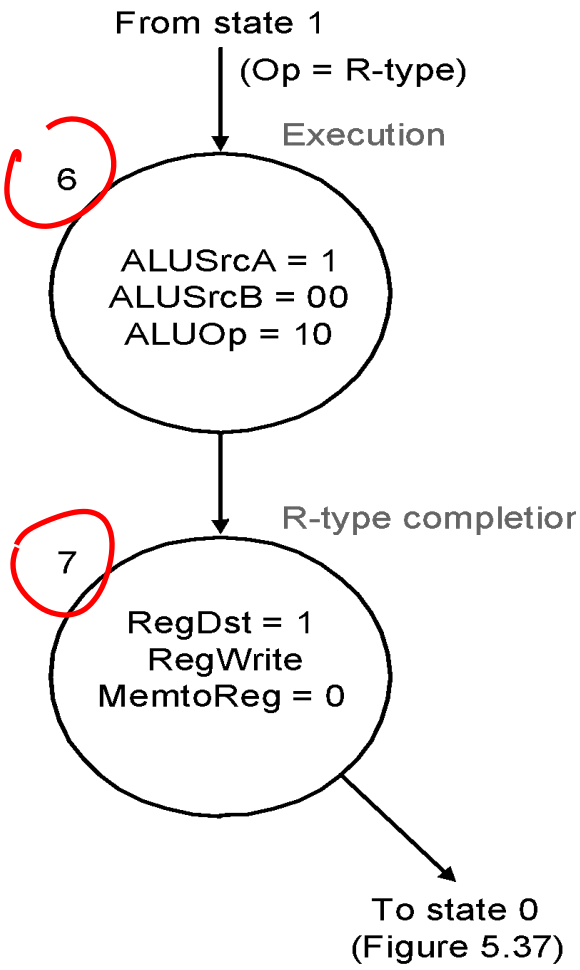
RegWrite
MemtoReg =0
RegDst =0



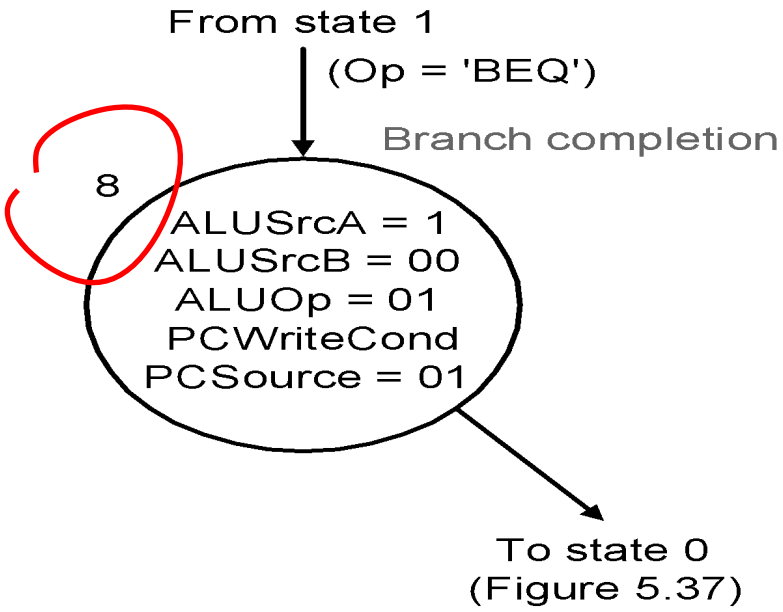
FSM Control : Memory Reference



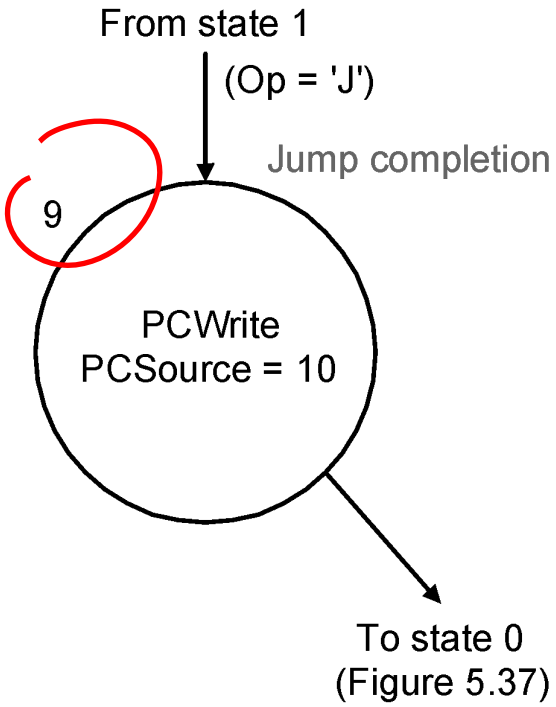
FSM Control: R-type Instruction



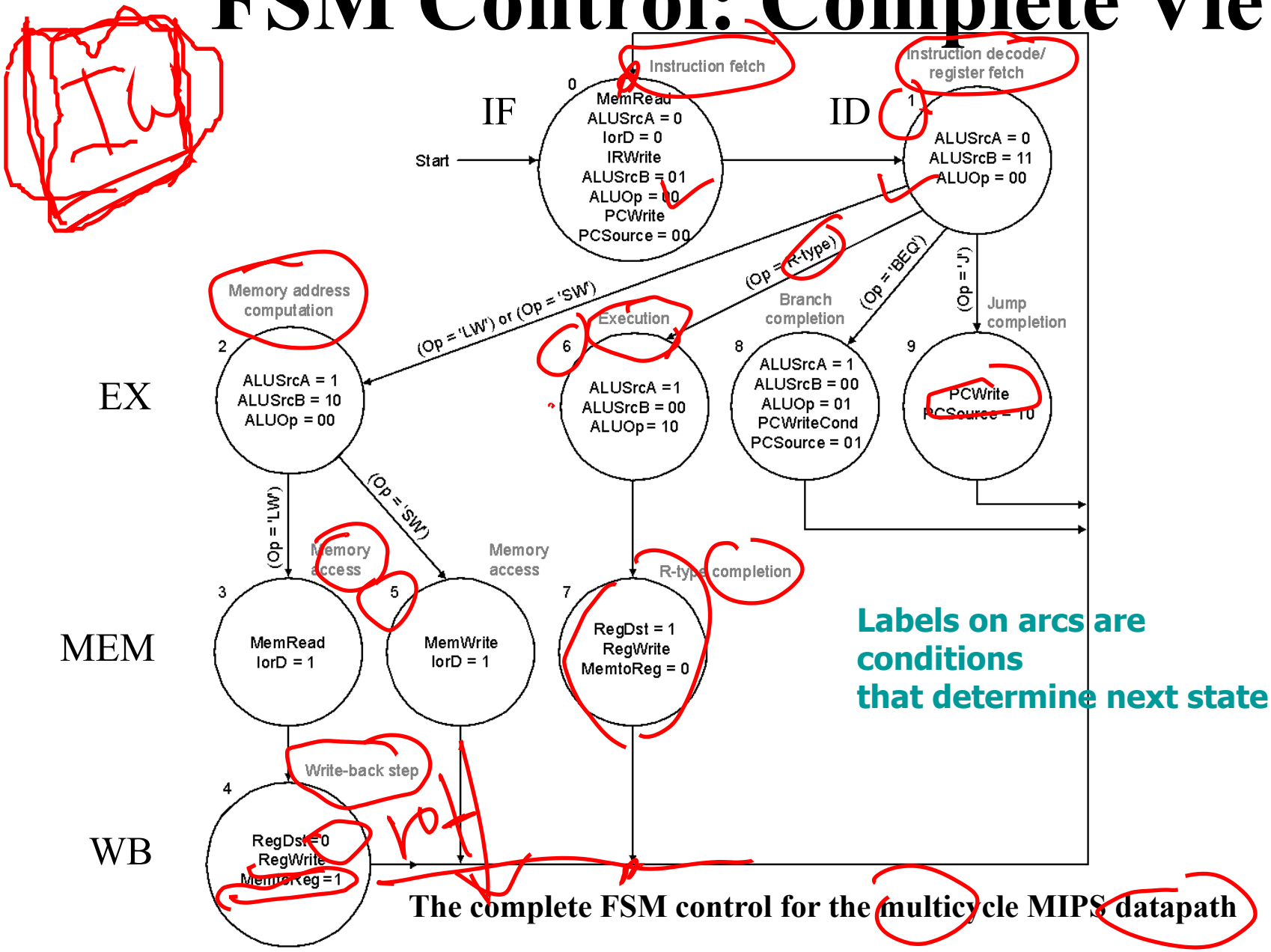
FSM Control: Branch Instruction



FSM Control: Jump Instruction



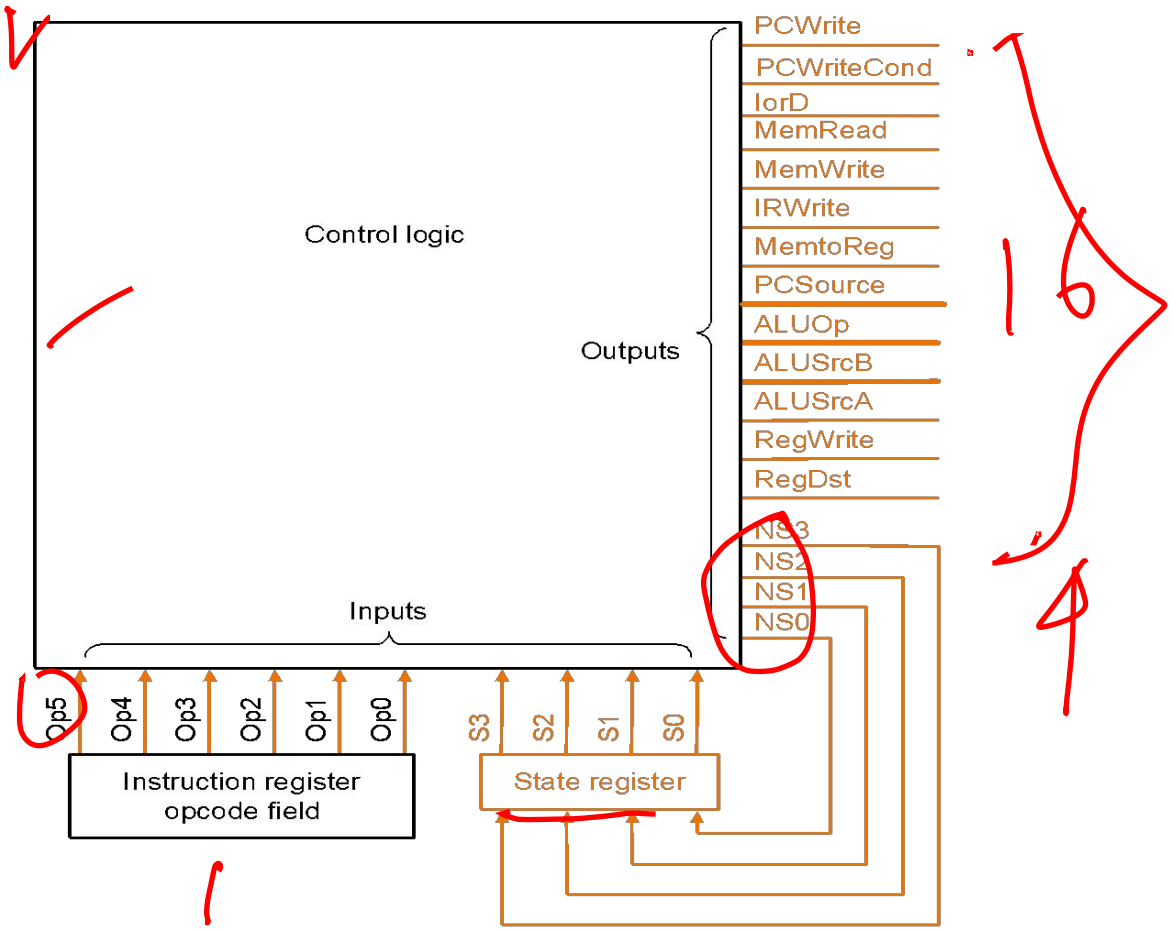
FSM Control: Complete View



Labels on arcs are conditions that determine next state

The complete FSM control for the multicycle MIPS datapath

FSM Controller



Moore Machine:

The output depends on the current state.

Mealy Machine:

The machine allows both the input and the current state to be used to determine the output.

Note:

Control logic designed using moore machine.

FSM Controller

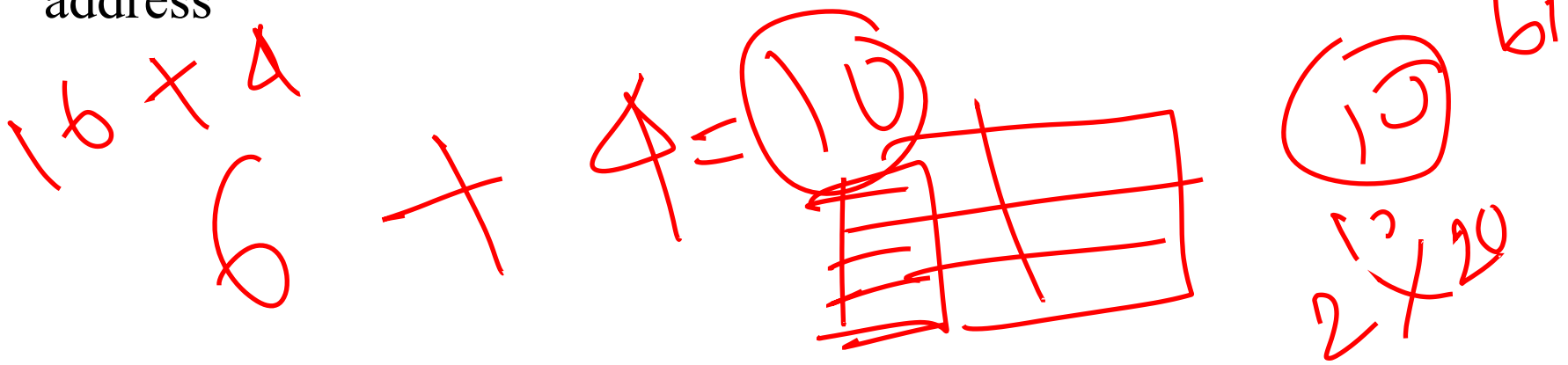
- 4 bits are needed to encode 10 states. For example is state is 6 then it is encoded as 0110_2 s3=0, s2=1, s1=1, s0=0. that is, $\overline{s3}$. s2. s1. $\overline{s0}$
- The control logic implements 2 parts of FSM.
  1. datapath control outputs, that depend only on the state bits.
  2. Next state bits based on the current state bits and the opcode.

Logic Equations for the Control Unit

Output	Current states	Op
PCWrite	state0 + state9	
PCWriteCond	state8	
IorD	state3 + state5	
MemRead	state0 + state3	
MemWrite	state5	
IRWrite	state0	
MemtoReg	state4	
PCSource1	state9	
PCSource0	state8	
ALUOp1	state6	
ALUOp0	state8	
ALUSrcB1	state1 +state2	
ALUSrcB0	state0 + state1	
ALUSrcA	state2 + state6 + state8	
RegWrite	state4 + state7	
RegDst	state7	
NextState0	state4 + state5 + state7 + state8 + state9	
NextState1	state0	
NextState2	state1	(Op = 'lw') + (Op = 'sw')
NextState3	state2	(Op = 'lw')
NextState4	state3	
NextState5	state2	(Op = 'sw')
NextState6	state1	(Op = 'R-type')
NextState7	state2	
NextState8	state1	(Op = 'beq')
NextState9	state1	(Op = 'jmp')

A ROM Implementation

- ✔ We can implement the control function by encoding the truth tables in the read only memory (ROM).
- ✔ The number of entries for the truth table is 2^{10} = 1024
- ✔ The input of the control unit is the address of the ROM and we get 20 bits control word for each input combination.
- ✔ The total size of the ROM is $2^{10} \times 20$ = 20 KB.
- ✔ The next state bits are the lower bits of the control word and the current state input bits are the lower order bits of the address



Truth table for the 16 Datapath Control Output

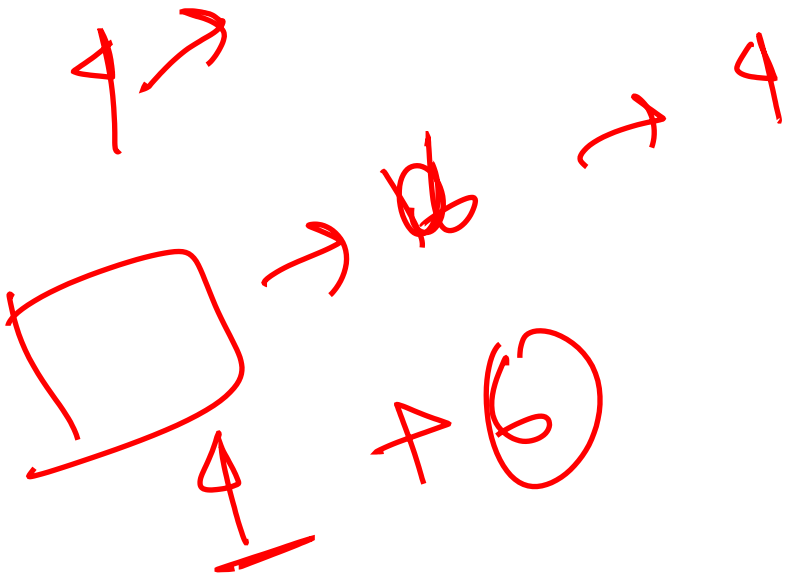
Outputs	Input values (S[3–0])									
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
PCWrite	1	0	0	0	0	0	0	0	0	1
PCWriteCond	0	0	0	0	0	0	0	0	1	0
IorD	0	0	0	1	0	1	0	0	0	0
MemRead	1	0	0	1	0	0	0	0	0	0
MemWrite	0	0	0	0	0	1	0	0	0	0
IRWrite	1	0	0	0	0	0	0	0	0	0
MemtoReg	0	0	0	0	1	0	0	0	0	0
PCSource1	0	0	0	0	0	0	0	0	0	1
PCSource0	0	0	0	0	0	0	0	0	1	0
ALUOp1	0	0	0	0	0	0	1	0	0	0
ALUOp0	0	0	0	0	0	0	0	0	1	0
ALUSrcB1	0	1	1	0	0	0	0	0	0	0
ALUSrcB0	1	1	0	0	0	0	0	0	0	0
ALUSrcA	0	0	1	0	0	0	1	0	1	0
RegWrite	0	0	0	0	1	0	0	1	0	0
RegDst	0	0	0	0	0	0	0	1	0	0

Truth Table for the Next State Function

Current state S[3–0]	Op [5–0] 000000 (R-format)	000010 (jmp)	000100 (beq)	100011 (lw)	101011 (sw)	Any other value
0000	0001	0001	0001	0001	0001	0001
0001	0110	1001	1000	0010	0010	illegal
0010	XXXX	XXXX	XXXX	0011	0101	illegal
0011	0100	0100	0100	0100	0100	illegal
0100	0000	0000	0000	0000	0000	illegal
0101	0000	0000	0000	0000	0000	illegal
0110	0111	0111	0111	0111	0111	illegal
0111	0000	0000	0000	0000	0000	illegal
1000	0000	0000	0000	0000	0000	illegal
1001	0000	0000	0000	0000	0000	illegal

Implementation of FSM Controller using ROM

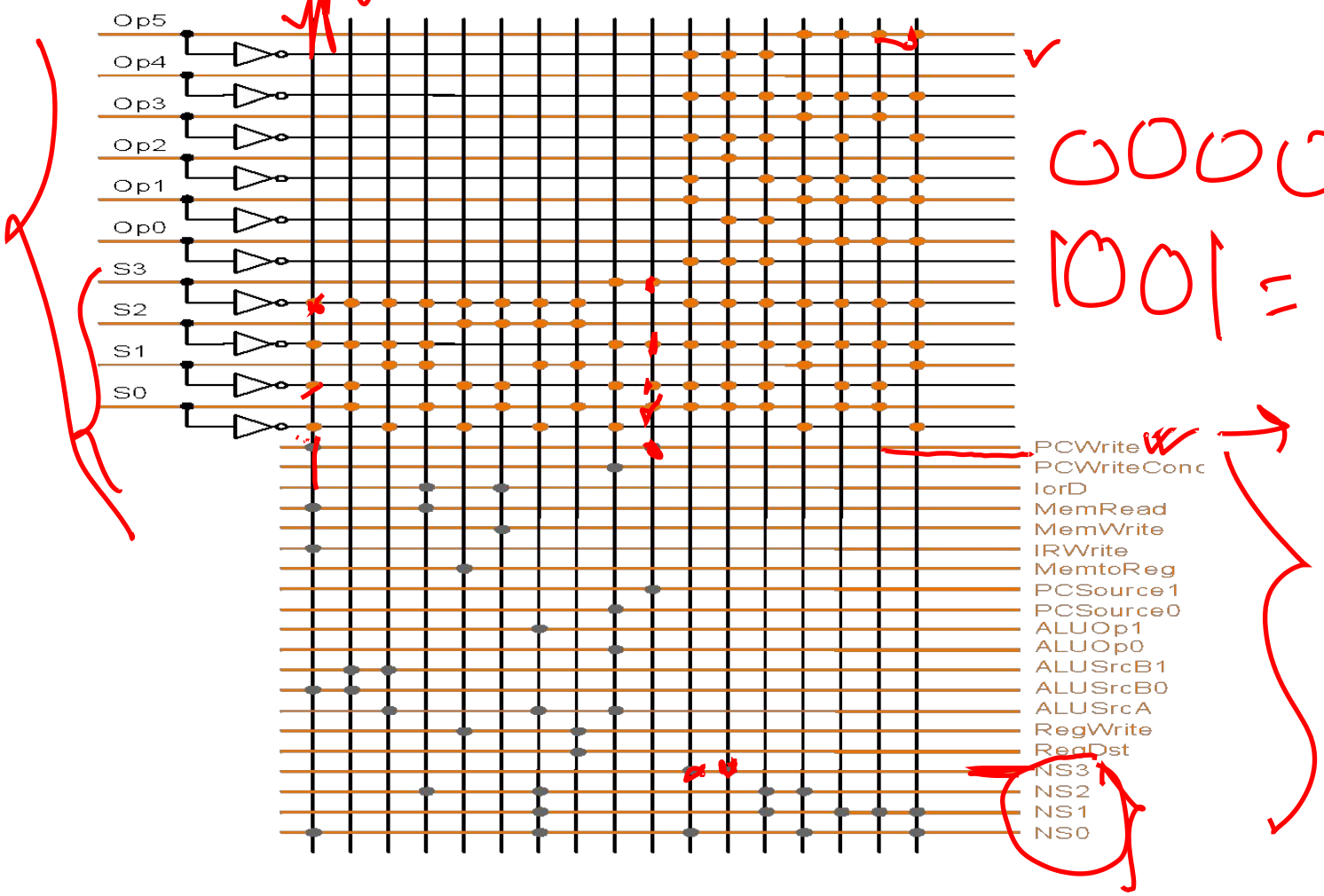
- ✔ We can implement the control unit by splitting into 2 ROMs-one for generating datapath control output and the other for generating the next state output.
- ✔ This requires $2^4 \times 16 + 2^{10} \times 4 = 4.3$ Kbits.



PLA Implementation of FSM Controller

- 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
- PLA is much smaller
  - can share product terms
  - only need entries that produce an active output

PLA Implementation of FSM Controller



PLA Implementation of FSM Controller

- PLA size = (#inputs * #product-terms) + (#outputs * #product-terms)
  - FSM control PLA = (10x17)+(20x17) = 460 PLA cells
- PLA cells usually about the size of a ROM cell (slightly bigger)