Corso di Laurea in Informatica
Architettura degli elaboratori
a a 2019-2020

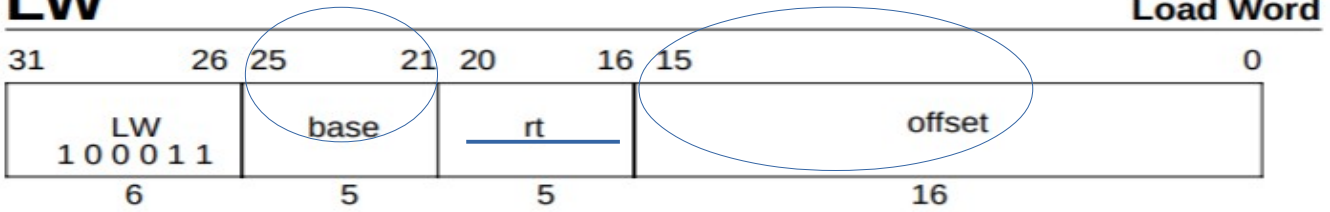
Architettura degli Elaboratori
2019-2020

Datapath: Automa per il controllo

Slide: Claudia Raibulet (commento in video Moodle: Claudio Ferretti)

LW

Load Word



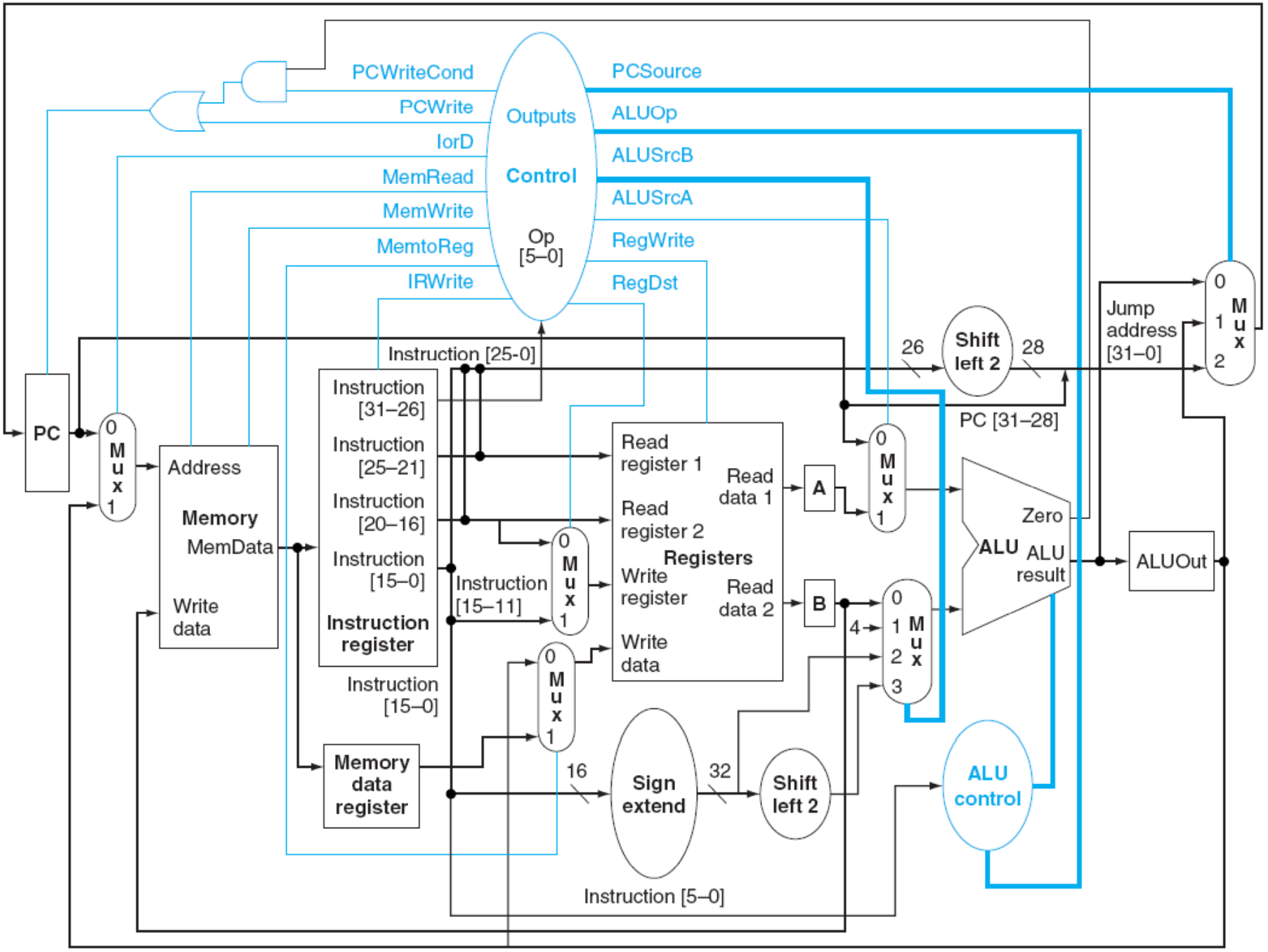
Format: LW rt, offset(base) MIPS I

Purpose: To load a word from memory as a signed value.

Description: rt ← memory[base+offset]

The contents of the 32-bit word at the memory location specified by the aligned effective address are fetched, sign-extended to the GPR register length if necessary, and placed in GPR *rt*. The 16-bit signed *offset* is added to the contents of GPR *base* to form the effective address.

Step name	Action for R-type instructions	Action for memory-reference instructions	Action branch
Instruction fetch		IR <= Memory[PC] PC <= PC + 4	
Instruction decode/register fetch		A <= Reg [IR[25:21]] B <= Reg [IR[20:16]] ALUOut <= PC + (sign-extend (IR[15:0]) << 2)	
Execution, address computation, branch/jump completion	ALUOut <= A op B	ALUOut <= A + sign-extend (IR[15:0])	if (A == PC <= ALU
Memory access or R-type completion	Reg [IR[15:11]] <= ALUOut	Load: MDR <= Memory[ALUOut] or Store: Memory [ALUOut] <= B	
Memory read completion		Load: Reg[IR[20:16]] <= MDR	


PCWriteCond
PCWrite
IorD
MemRead
MemWrite
MemtoReg
IRWrite
Outputs
Control
Op
[5–0]
PCSource
ALUOp
ALUSrcB
ALUSrcA
RegWrite
RegDst
PC
0
M
u
x
1
Address
Memory
MemData
Write
data
Instruction [25-0]
Instruction
[31–26]
Instruction
[25–21]
Instruction
[20–16]
Instruction
[15–0]
Instruction
register
Instruction
[15–0]
Instruction
[15–11]
Memory
data
register
Read
register 1
Read
register 2
Registers
Write
register
Write
data
Read
data 1
Read
data 2
A
B
4
26
Shift
left 2
28
PC [31–28]
Jump
address
[31–0]
0
1
2
3
ALU
Zero
ALU
result
ALUOut
16
Sign
extend
32
Shift
left 2
ALU
control
Instruction [5–0]

Segnali di controllo a 1 bit

Actions of the 1-bit control signals

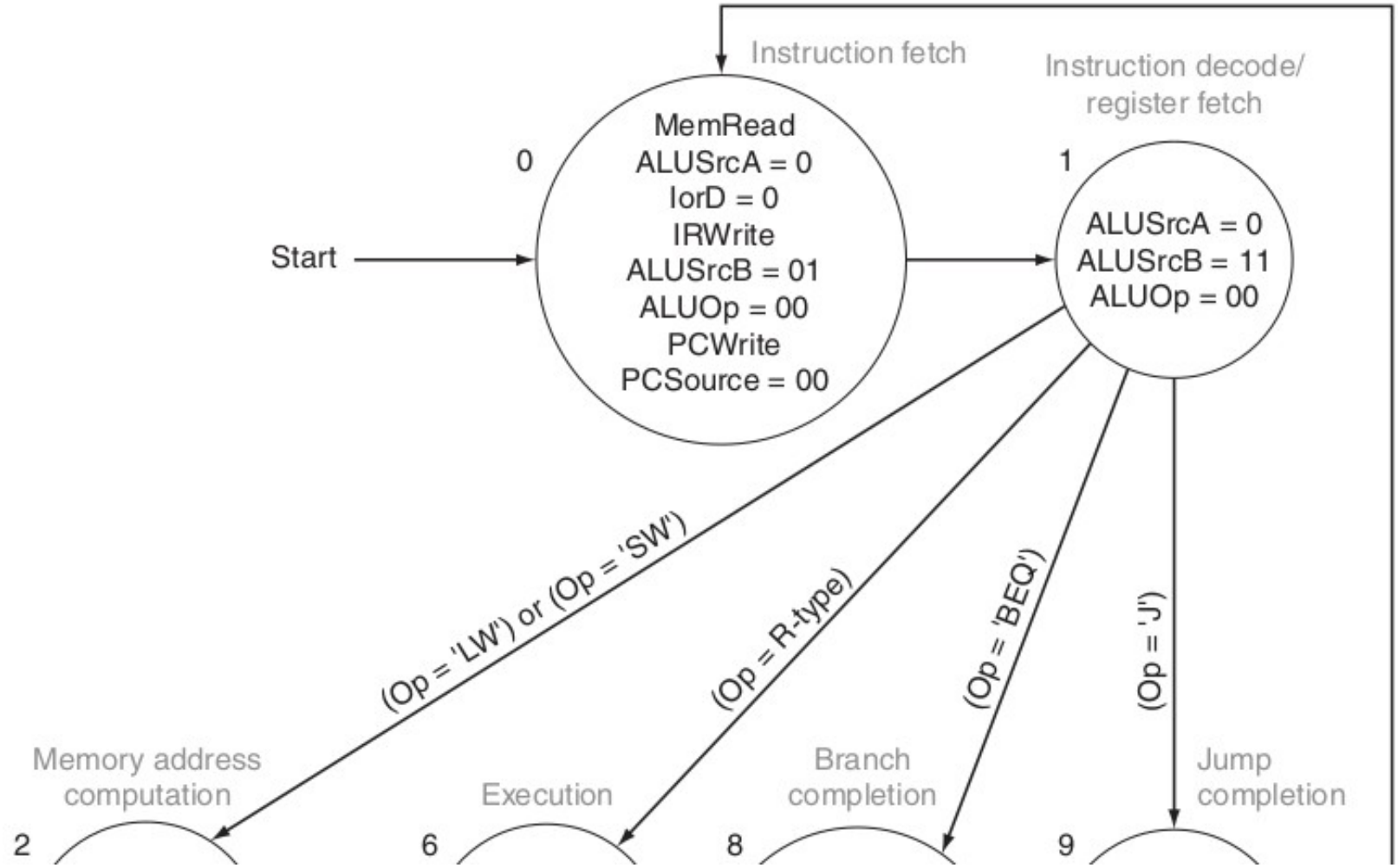
Signal name	Effect when deasserted	Effect when asserted
RegDst	The register file destination number for the Write register comes from the rt field.	The register file destination number for the Write register comes from the rd field.
RegWrite	None.	The general-purpose register selected by the Write register number is written with the value of the Write data input.
ALUSrcA	The first ALU operand is the PC.	The first ALU operand comes from the A register.
MemRead	None.	Content of memory at the location specified by the Address input is put on Memory data output.
MemWrite	None.	Memory contents at the location specified by the Address input is replaced by value on Write data input.
MemtoReg	The value fed to the register file Write data input comes from ALUOut.	The value fed to the register file Write data input comes from the MDR.
IorD	The PC is used to supply the address to the memory unit.	ALUOut is used to supply the address to the memory unit.
IRWrite	None.	The output of the memory is written into the IR.
PCWrite	None.	The PC is written; the source is controlled by PCSource.
PCWriteCond	None.	The PC is written if the Zero output from the ALU is also active.

Segnali di controllo a 2 bit

Actions of the 2-bit control signals

Signal name	Value (binary)	Effect
ALUOp	00	The ALU performs an add operation.
	01	The ALU performs a subtract operation.
	10	The funct field of the instruction determines the ALU operation.
ALUSrcB	00	The second input to the ALU comes from the B register.
	01	The second input to the ALU is the constant 4.
	10	The second input to the ALU is the sign-extended, lower 16 bits of the IR.
	11	The second input to the ALU is the sign-extended, lower 16 bits of the IR shifted left 2 bits.
PCSource	00	Output of the ALU (PC + 4) is sent to the PC for writing.
	01	The contents of ALUOut (the branch target address) are sent to the PC for writing.
	10	The jump target address (IR[25:0] shifted left 2 bits and concatenated with PC + 4[31:28]) is sent to the PC for writing.

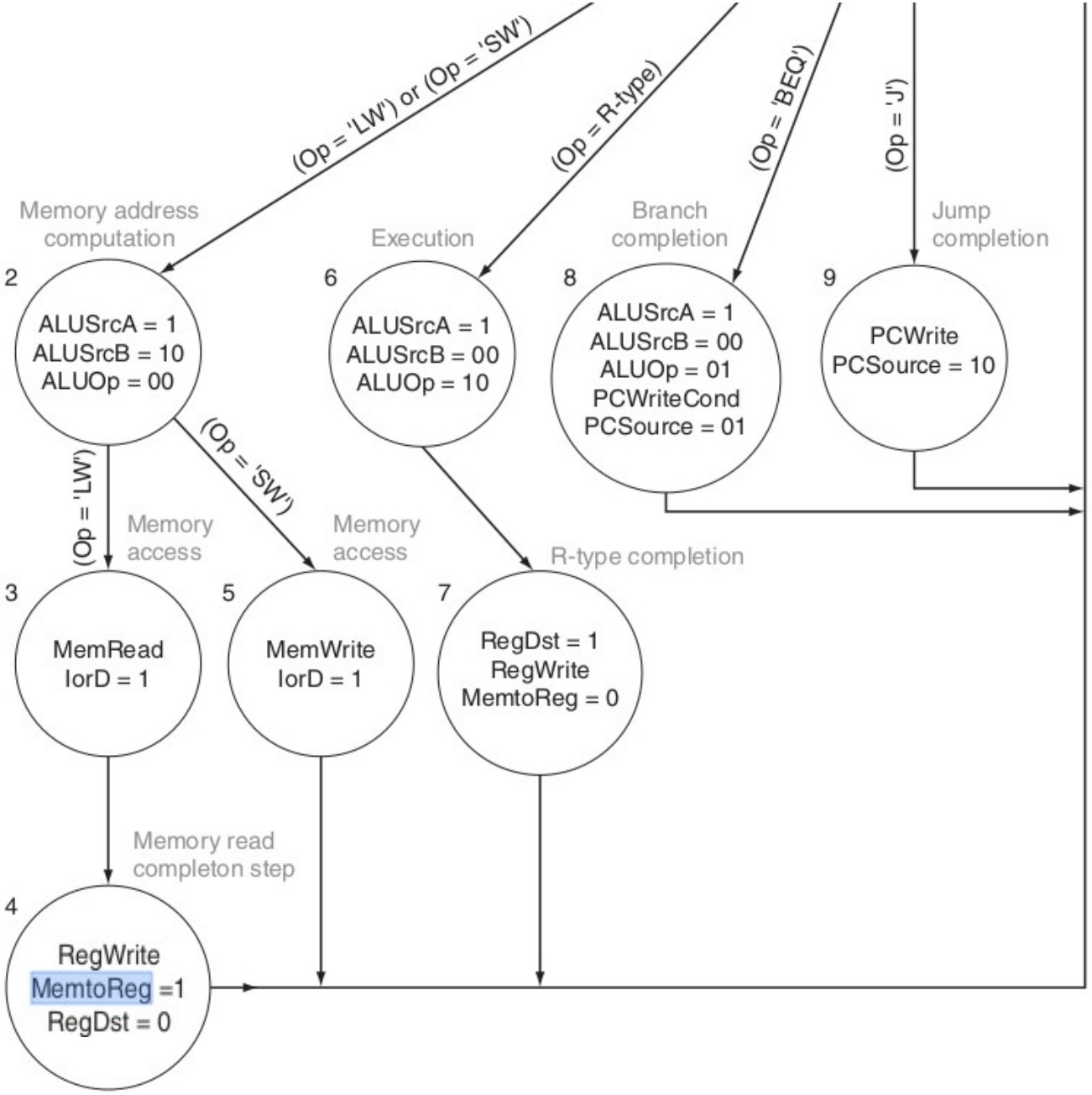
FIGURE 5.29 The action caused by the setting of each control signal in Figure 5.28 on page 323. The top table describes the 1-bit control signals, while the bottom table describes the 2-bit signals. Only those control lines that affect multiplexors have an action when they are deasserted. This information is similar to that in Figure 5.16 on page 306 for the single-cycle datapath, but adds several new control lines (IRWrite, PCWrite, PCWriteCond, ALUSrcB, and PCSource) and removes control lines that are no longer used or have been replaced (PCSrc, Branch, and Jump).



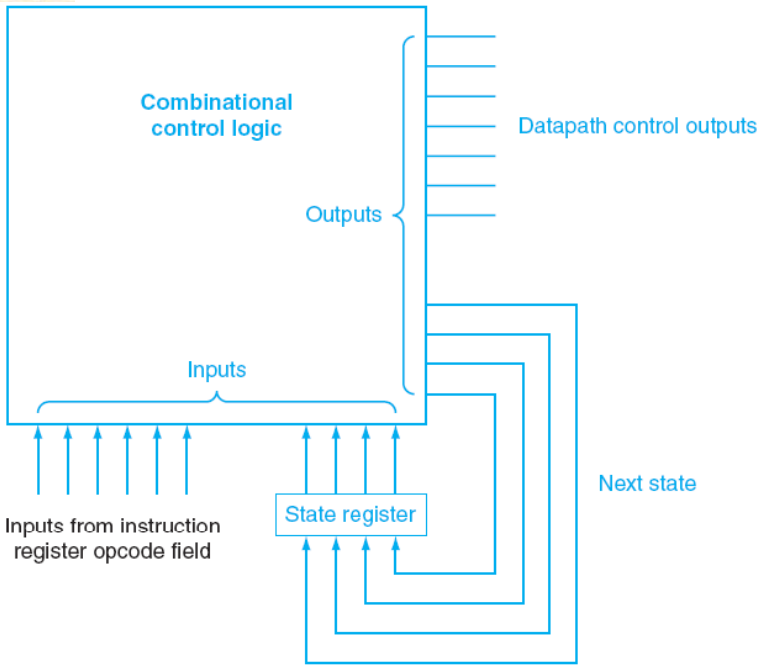
...altri stati, poi
torna a ‘0’

(poi torna a ‘0’) ^

...qui dai primi stati:



Circuito sequenziale per il controllo



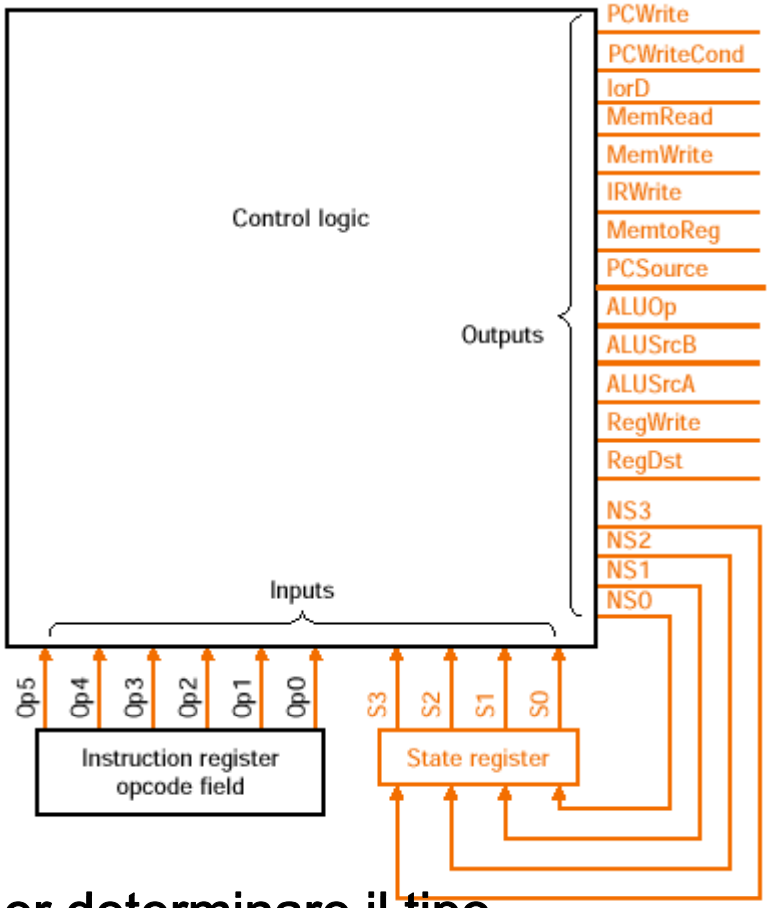


FIGURE 5.37 Finite state machine controllers are typically implemented using a block of combinational logic and a register to hold the current state. The outputs of the combinational logic are the next-state number and the control signals to be asserted for the current state. The inputs to the combinational logic are the current state and any inputs used to determine the next state. In this case, the inputs are the instruction register opcode bits. Notice that in the finite state machine used in this chapter, the outputs depend only on the current state, not on the inputs. The Elaboration above explains this in more detail.

- Controllo a due livelli: si usa OPCODE per determinare il tipo dell'istruzione e per R_type si usa anche il FUNC CODE
- State register memorizza lo stato corrente
- Blocco combinatorio (PLA) per il calcolo di NEXT_STATE e OUTPUT (memorizzate in ROM)