Il Controllo della ALU (1)

- Direttamente tramite l'unità di controllo principale….

Ingresso di controllo della ALU	Funzione
000	AND
001	OR
010	somma
110	sottrazione
111	set on less than

Istruzioni	Codice operativo istruzione	Codice della funzione	Segnali controllo ALU
	$O_5 O_4 O_3 O_2 O_1 O_0$	$F_5 F_4 F_3 F_2 F_1 F_0$	O_{P2} O_{P1} O_{P0}
somma	0 0 0 0 0 0	1 0 0 0 0 0	0 1 0
sottraz.	0 0 0 0 0 0	1 0 0 0 1 0	1 1 0
AND	0 0 0 0 0 0	1 0 0 1 0 0	0 0 0
OR	0 0 0 0 0 0	1 0 0 1 0 1	0 0 1
set on less than	0 0 0 0 0 0	1 0 1 0 1 0	1 1 1
lw	1 0 0 0 1 1	- - - - - -	0 1 0
sw	1 0 1 0 0 1	- - - - - -	0 1 0
branch equal	0 0 0 1 0 0	- - - - - -	1 1 0

p.e. $O_{P2} = \bar{O}_5 \bar{O}_4 \bar{O}_3 \bar{O}_2 \bar{O}_1 \bar{O}_0 F_5 \bar{F}_4 (\bar{F}_3 \bar{F}_2 F_1 \bar{F}_0 + F_3 \bar{F}_2 F_1 \bar{F}_0) +$

$+ \bar{O}_5 \bar{O}_4 \bar{O}_3 O_2 \bar{O}_1 \bar{O}_0$

Il Controllo della ALU (2)

…oppure tramite un'apposita sotto-unità di controllo, detta:

ALU Control Unit

Idea di Base:

- ridurre dimensione e complessità dell'unità di controllo principale

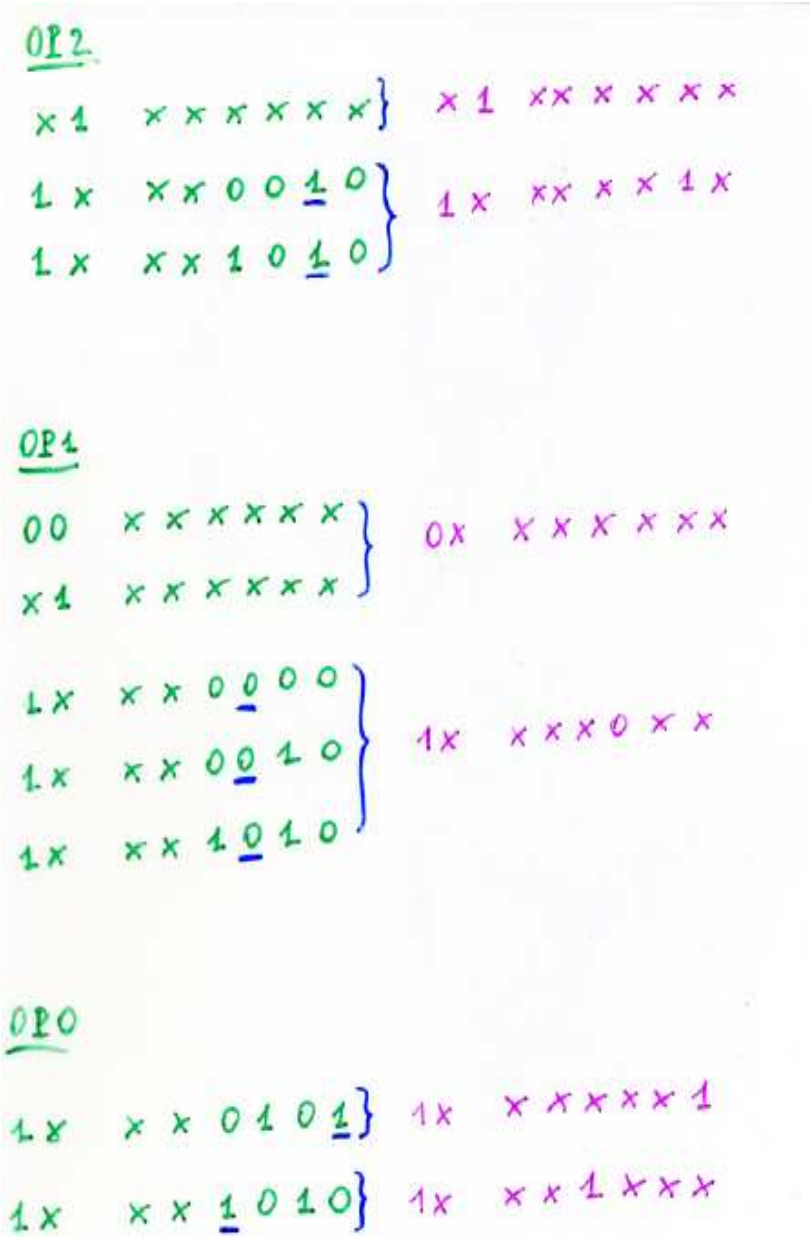
La ALU Control Unit genera in output i (3) segnali di controllo ALU, prendendo in input:

- il campo funct dell'istruzione
- due segnali di controllo (ALUOp) generati dalla Control Unit principale che codificano la tipologia di istruzione da eseguire

Codice operativo istruzione	ALUOp	Operazione dell'istruzione	Codice della funzione	Azione dell'ALU desiderata	Ingresso di controllo dell'ALU
LW	00	Caricamento parola	XXXXXX	add	010
SW	00	Memorizzazione parola	XXXXXX	add	010
Branch equal	01	Salto condizionato	XXXXXX	subtract	110
Tipo R	10	Somma	100000	add	010
Tipo R	10	Sottrazione	100010	~~add~~ sub	110
Tipo R	10	AND	100100	and	000
Tipo R	10	OR	100101	or	001
Tipo R	10	Confronto di minoranza	101010	set-on-less-than	111

ALUOp		Codice della funzione						Operazione ALU
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	010
X	1	X	X	X	X	X	X	110
1	X	X	X	0	0	0	0	010
1	X	X	X	0	0	1	0	110
1	X	X	X	0	1	0	0	000
1	X	X	X	0	1	0	1	001
1	X	X	X	1	0	1	0	111

Semplificazione sfruttando le d.c.c.



ALUOp		Codice della funzione						Operazione ALU
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0	
0	0	X	X	X	X	X	X	010
X	1	X	X	X	X	X	X	110
1	X	X	X	0	0	0	0	010
1	X	X	X	0	0	1	0	110
1	X	X	X	0	1	0	0	000
1	X	X	X	0	1	0	1	001
1	X	X	X	1	0	1	0	111

- In blu sono evidenziati i valori delle variabili di ingresso che identificano tutti e soli i casi in cui le variabili di uscita sono vere (indipendentemente dalle altre variabili di ingresso)

Sintesi della rete combinatoria

ALUOp		Campi di codice della funzione					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
X	1	X	X	X	X	X	X
1	X	X	X	X	X	1	X

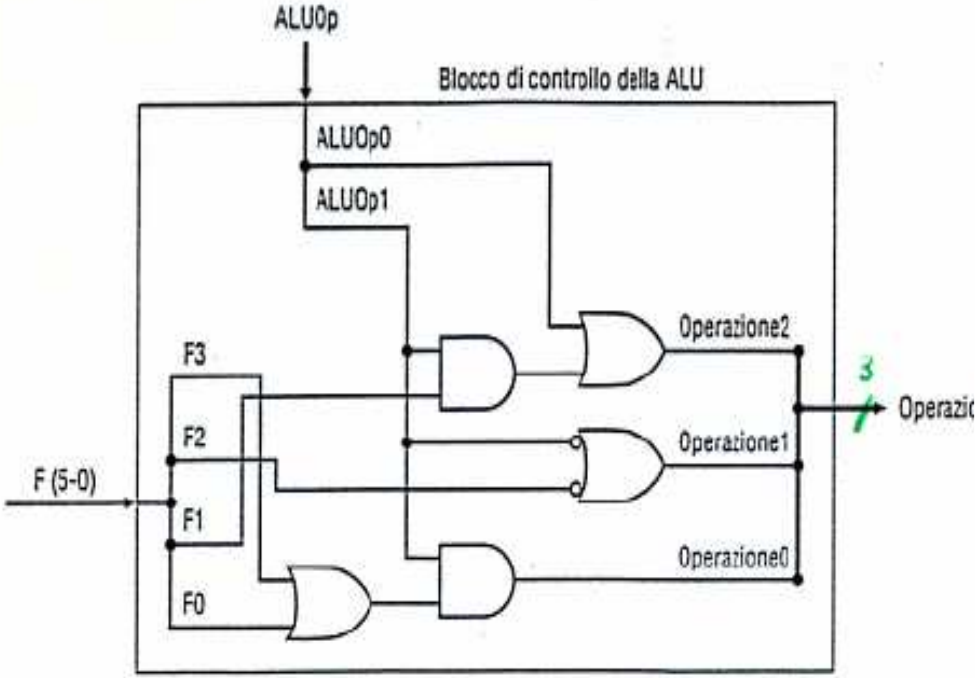
a. Tabella delle verità per Operazione2 = 1. (Questa tabella corrisponde al bit di sinistra del campo Operazione nella figura 5.16.)

ALUOp		Campi di codice della funzione					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
0	X	X	X	X	X	X	X
X	X	X	X	X	0	X	X

b. Tabella delle verità per Operazione1 = 1.

ALUOp		Campi di codice della funzione					
ALUOp1	ALUOp0	F5	F4	F3	F2	F1	F0
1	X	X	X	X	X	X	1
1	X	X	X	1	X	X	X

c. Tabella delle verità per Operazione0 = 1.



SCA/SCO: istruzioni R/lw-sw/beq

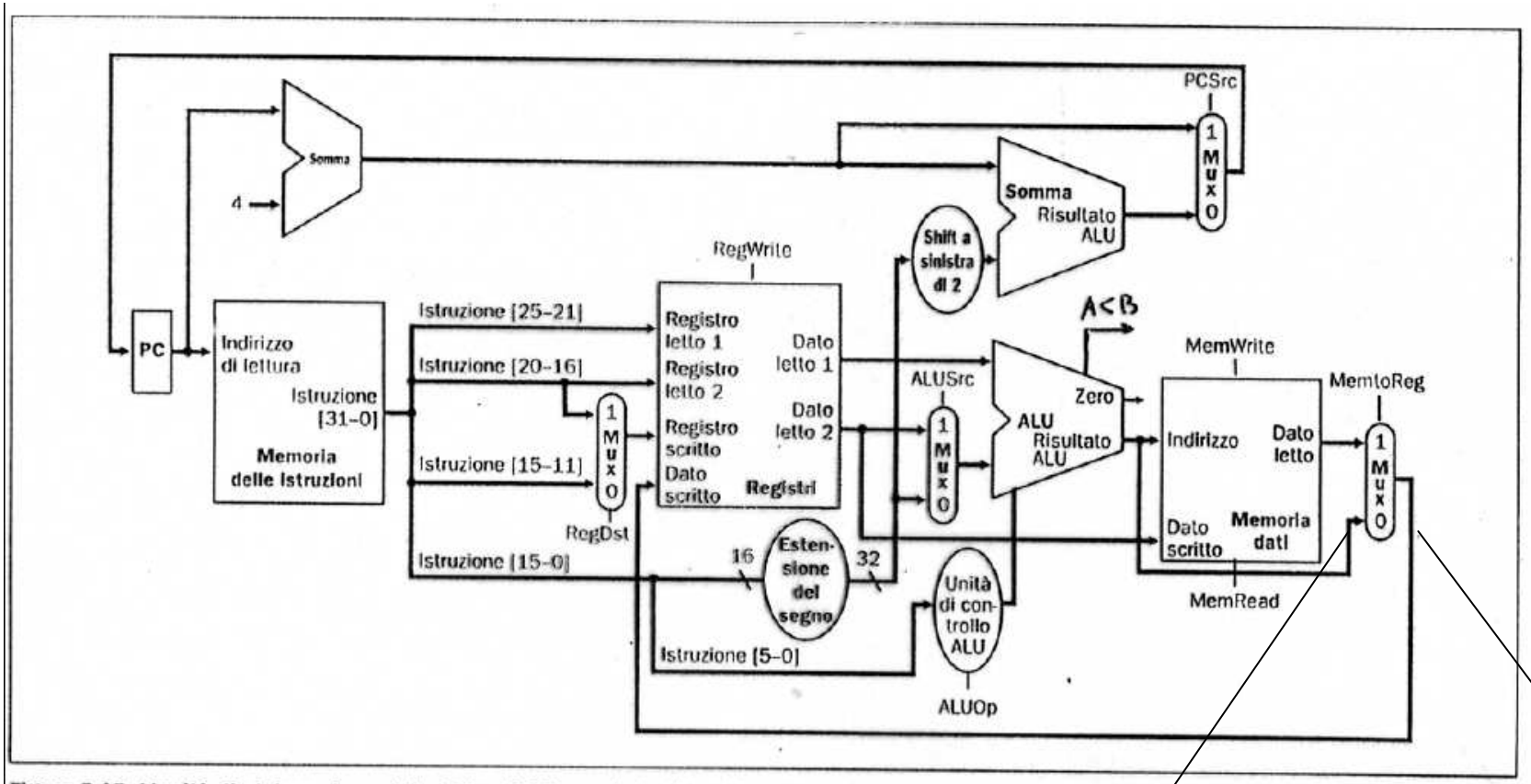
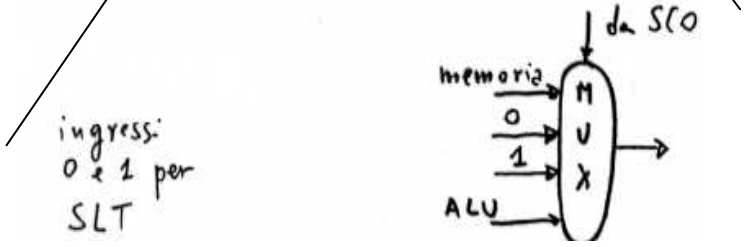
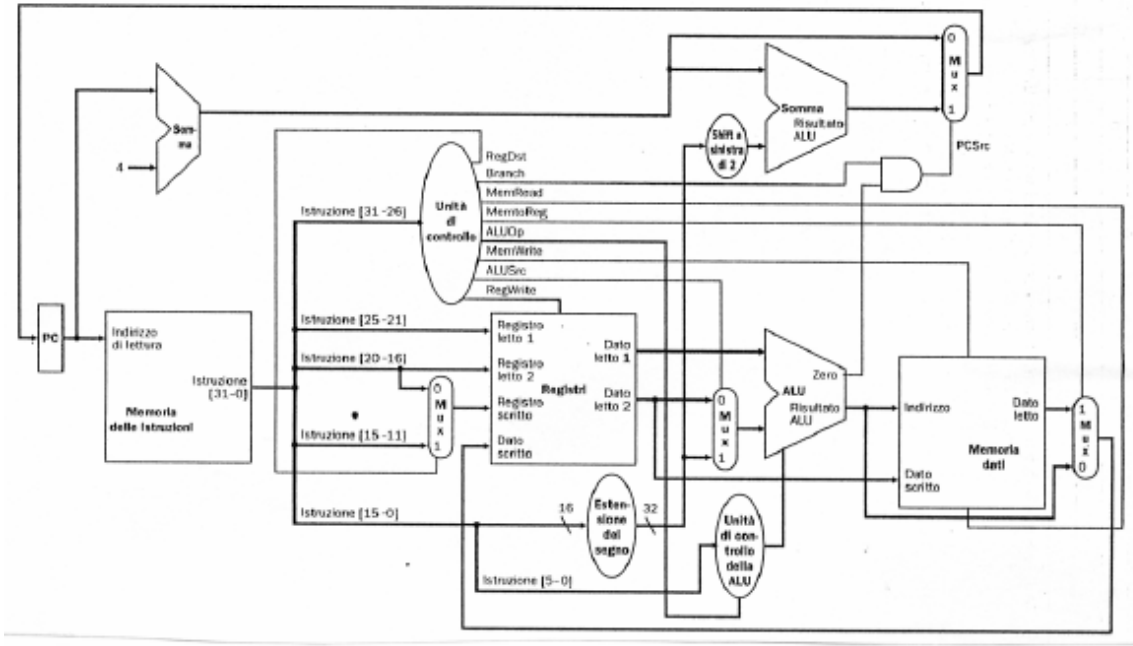


Figura 5.17 L'unità di elaborazione della figura 5.13 completa di tutti i multiplexer necessari e con l'identificazione di tutti i segnali di controllo. I segnali di controllo sono indicati con il colore ed è stata aggiunta anche l'unità di controllo della ALU. PC non necessita di segnali di controllo in quanto viene scritto al termine di ogni ciclo di clock; la logica di controllo dei salti determina se viene scritto PC incrementato o l'indirizzo di destinazione del salto.



I Segnali di Controllo in > Dettaglio



	Nome del segnale	Effetto quando non affermato	Effetto quando affermato
1	RegDst	Il numero del registro destinazione per Registro scritto proviene dal campo rt (bit 20-16).	Il numero del registro destinazione per Registro scritto proviene dal campo rd (bit 15-11).
2	RegWrite	Nessuno.	Nel registro specificato dall'ingresso Registro scritto è scritto il valore presente sull'ingresso Dato scritto.
3	ALUSrc	Il secondo operando della ALU proviene dalla seconda uscita del register file (Dato letto 2).	Il secondo operando della ALU è la versione estesa dei 16 bit inferiori dell'istruzione.
7	PCSrc	Il valore di PC viene sostituito dall'uscita del sommatore che calcola il valore di PC+4.	Il valore di PC viene sostituito dall'uscita del sommatore che calcola la destinazione del salto.
4	MemRead	Nessuno.	Il contenuto della cella di memoria dati determinato dall'ingresso Indirizzo è posto sull'uscita Dato letto.
5	MemWrite	Nessuno.	Il contenuto della cella di memoria dati determinato dall'ingresso Indirizzo è sostituito dal valore presente sull'ingresso Dato scritto.
6	MemtoReg	Il valore inviato all'ingresso Dato scritto dei registri proviene dalla ALU.	Il valore inviato all'ingresso Dato scritto dei registri proviene dalla memoria dati.

7 Branch — Il PC viene aggiornato con PC+4

Se zero = 0 ⇒ PC ← PC+4

Se zero = 1 ⇒ PC ← PC+4 + (offset)*4

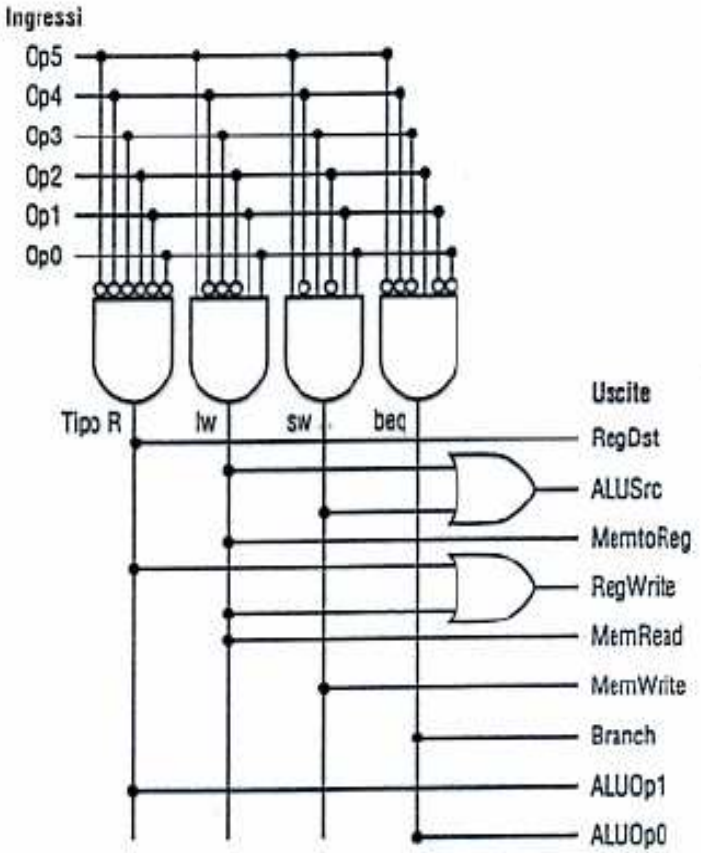
Sintesi SCO tramite PLA

Istruzione	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
Tipo R	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	x	1	x	0	0	1	0	0	0
beq	x	0	x	0	0	0	1	0	1

Associazione Istruzione – Segnali di Controllo
(inclusi segnali di controllo per ALU control unit)

Nome	Codice operativo decimale	Codice operativo binario					
		Op5	Op4	Op3	Op2	Op1	Op0
Tipo R	0_{10}	0	0	0	0	0	0
lw	35_{10}	1	0	0	0	1	1
sw	43_{10}	1	0	1	0	1	1
beq	4_{10}	0	0	0	1	0	0

Codifica istruzione tramite codice operativo



Istuzione Classe R (Passo LOGICO 1)

Istruzione	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
Tipo R	1	0	0	1	0	0	0	1	0

FETCH: **- istruzione=mem[PC]**

- PC = PC+4 (completato al colpo di clock successivo)

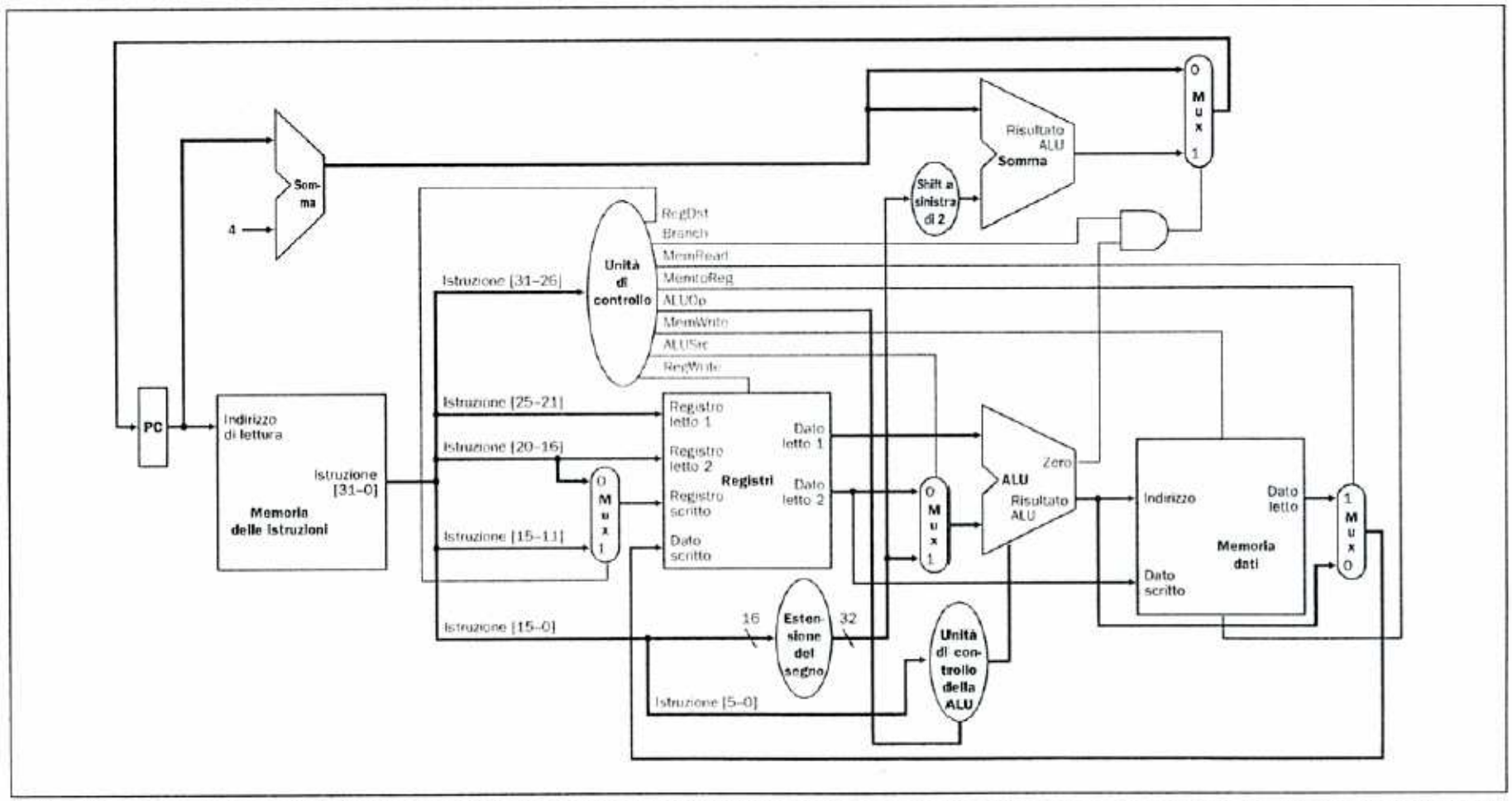


Figura 5.21 Il primo passo di un'istruzione di tipo-R preleva l'istruzione dalla memoria delle istruzioni ed incrementa PC. Le parti attive in questo passo sono evidenziate, mentre quelle disegnate in grigio chiaro non sono attive in questo passo ma lo diverranno in quelli successivi.

Istuzione Classe R (Passo LOGICO 2)

Istruzione	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
Tipo R	1	0	0	1	0	0	0	1	0

Lettura registri sorgenti

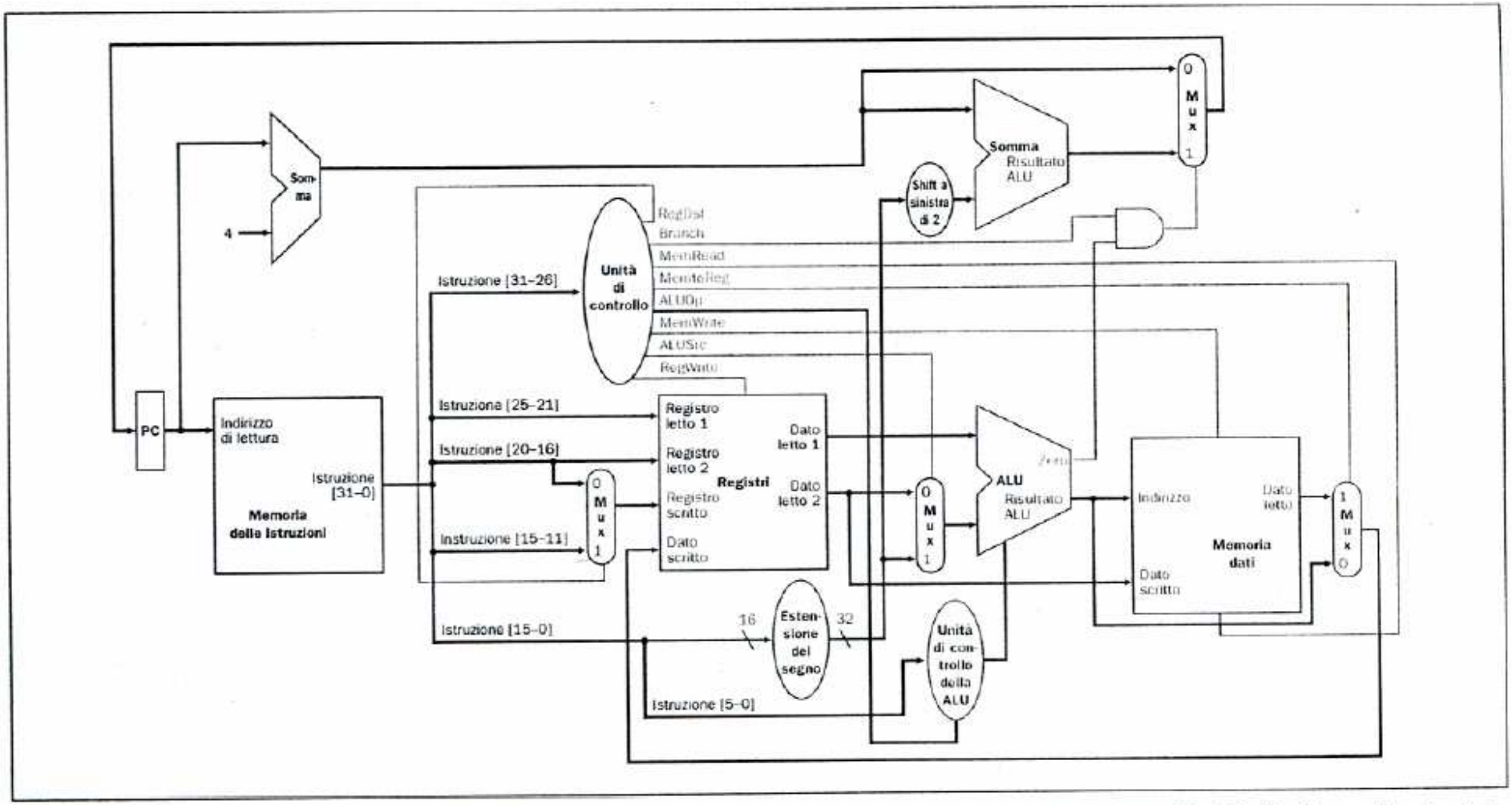
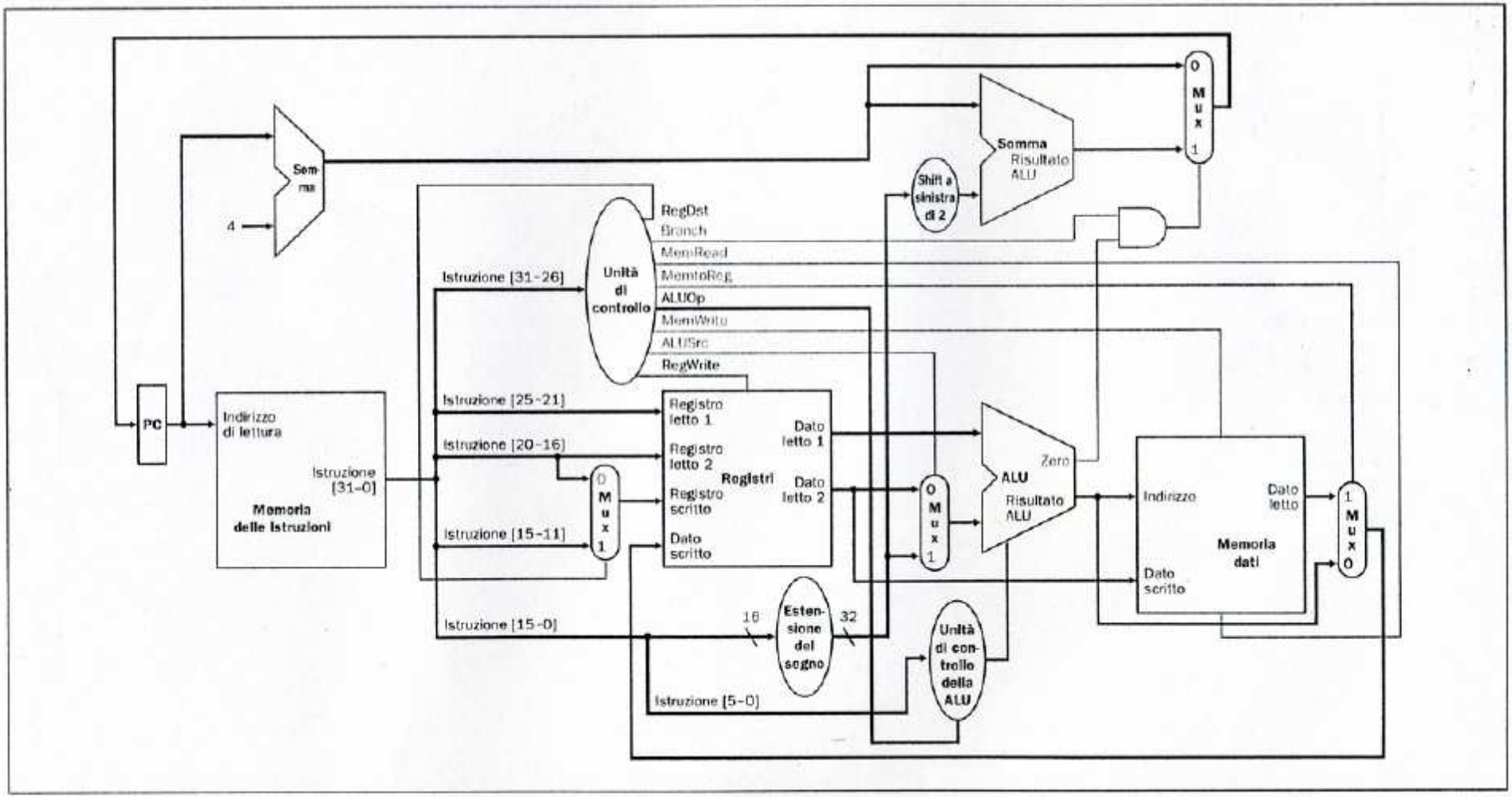


Figura 5.22 La seconda fase dell'esecuzione di un'istruzione di tipo-R legge i due registri sorgente dal register file. L'unità di controllo principale utilizza anche il campo opcode per determinare il valore dei segnali di controllo; queste unità diventano attive in aggiunta a quelle già attive durante il reperimento dell'informazione, riportate nella figura 5.21.

Istuzione Classe R (Passo LOGICO 3)

Istruzione	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
Tipo R	1	0	0	1	0	0	0	1	0

Elaborazione ALU (inputs stabili da registri e ALU Control Unit)



Istuzione Classe R (Passo LOGICO 4)

Istruzione	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp1	ALUOp0
Tipo R	1	0	0	1	0	0	0	1	0

Scrittura risultato ALU nel registro destinazione + aggiornamento PC

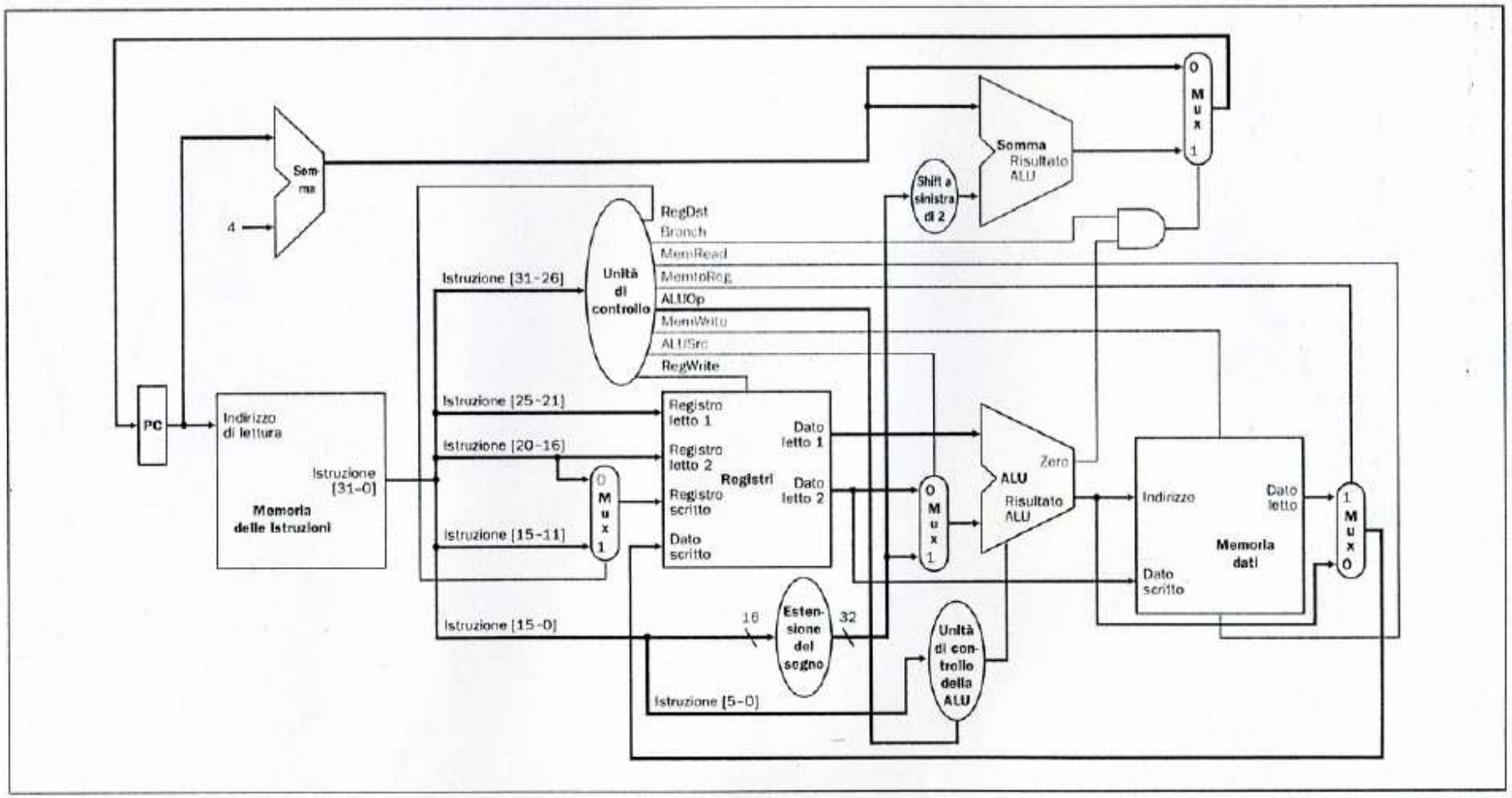
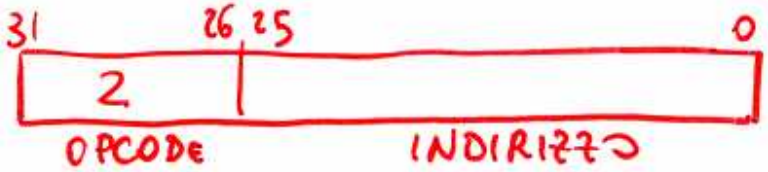


Figura 5.24 Il passo finale per le istruzioni di tipo-R prevede la scrittura del risultato; le relative unità attive sono state aggiunte a quelle dei tre passi precedenti, già indicate in figura 5.23. Al termine di questa fase avviene anche l'aggiornamento di PC. Essendo l'unità di elaborazione combinatoria, questo passo riporta tutte le unità attive e tutti i segnali di controllo affermati, una volta raggiunto lo stato stabile. Si osservi che l'istruzione verrà eseguita correttamente anche se utilizza lo stesso registro sia come ingresso che come uscita (come in `add R1, R1, R1`): il valore letto dai registri è il valore di R1 scritto al termine di un un ciclo di clock precedente, mentre il risultato non verrà scritto nel registro fino al prossimo fronte del clock.

Modifica SCA/SCO per Jump

J 1000 ------ salta all'indirizzo "1000"



26 bit

mi mancano 6

- 2 meno signif. 00 x SPIAZZAMENTO PAROLA
- 4 bit più significativi del PC

$PC \leftarrow (PC+4)_{31-28} \cdot INDIRIZZO \cdot 00$

concatenato

Modifica SCA/SCO per Jump

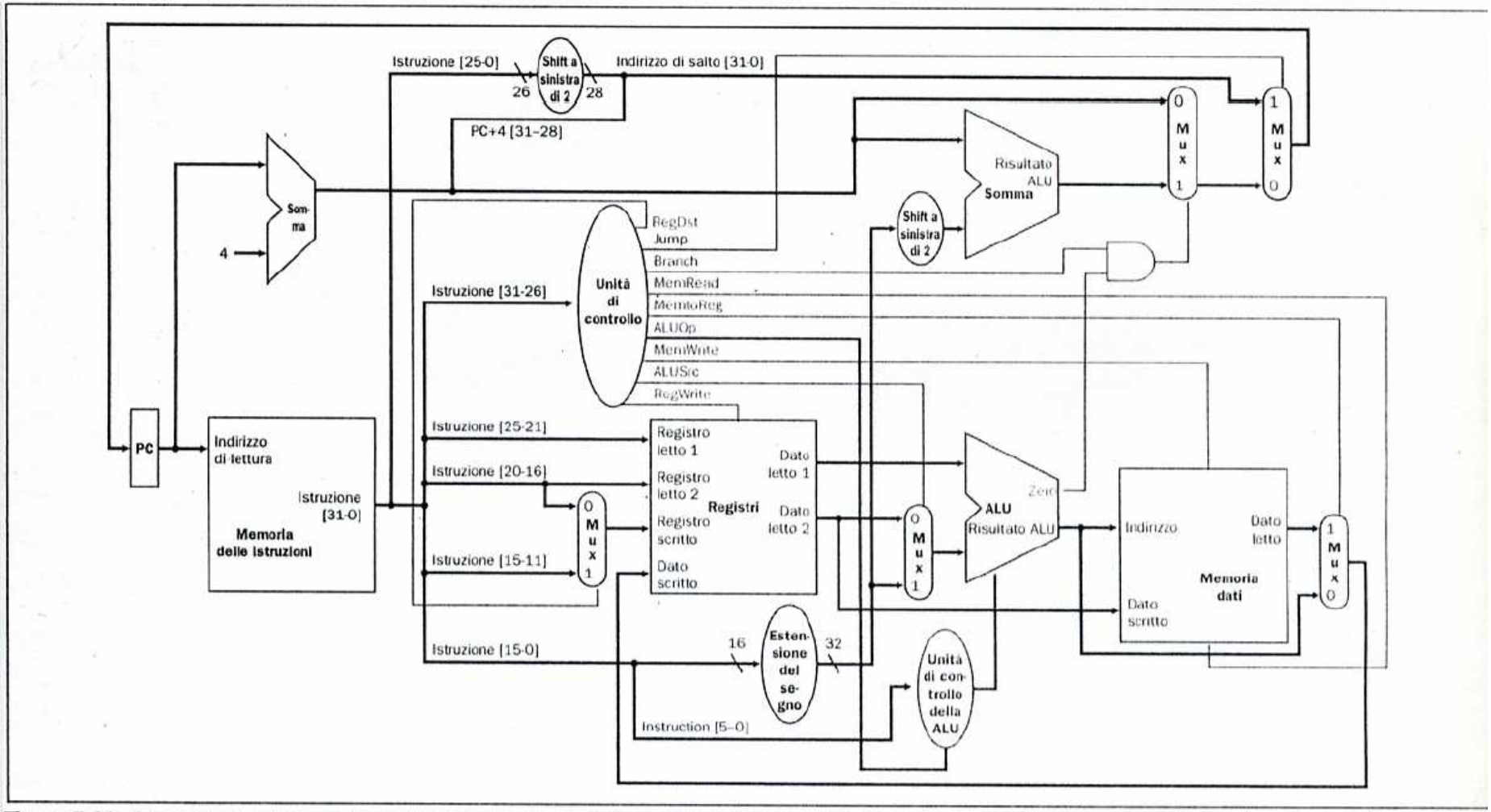


Figura 5.29 Estensione dell'unità di elaborazione elementare e della relativa unità di controllo per implementare l'istruzione jump. Viene utilizzato un multiplexer aggiuntivo (in alto a destra) per selezionare la destinazione del jump o, in alternativa, la destinazione del branch o l'istruzione seguente; il multiplexer è controllato dal segnale di controllo denominato Jump. Per ottenere l'indirizzo di destinazione del salto si procede nel seguente modo: si scalano a sinistra di 2 bit i 26 bit meno significativi dell'istruzione (aggiungendo in tal modo 00 come bit di ordine inferiore) e si concatenano al risultato ottenuto i 4 bit superiori di PC+4 in modo da ottenere un indirizzo su 32 bit.