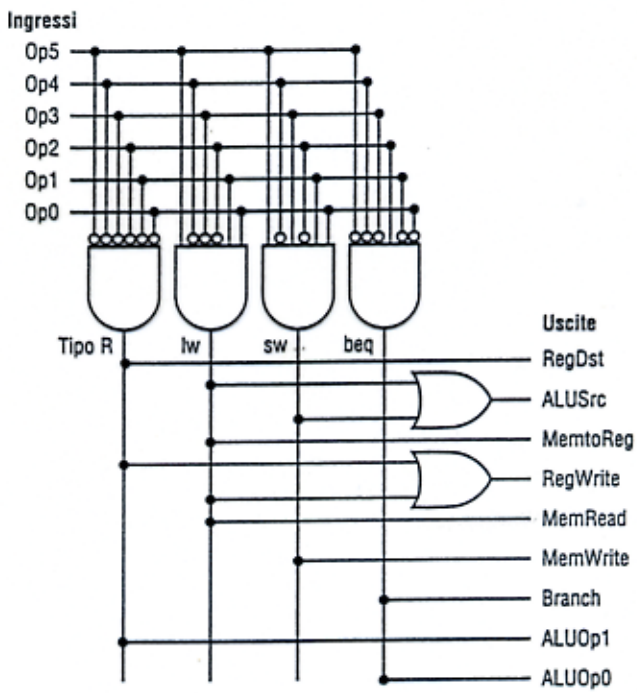
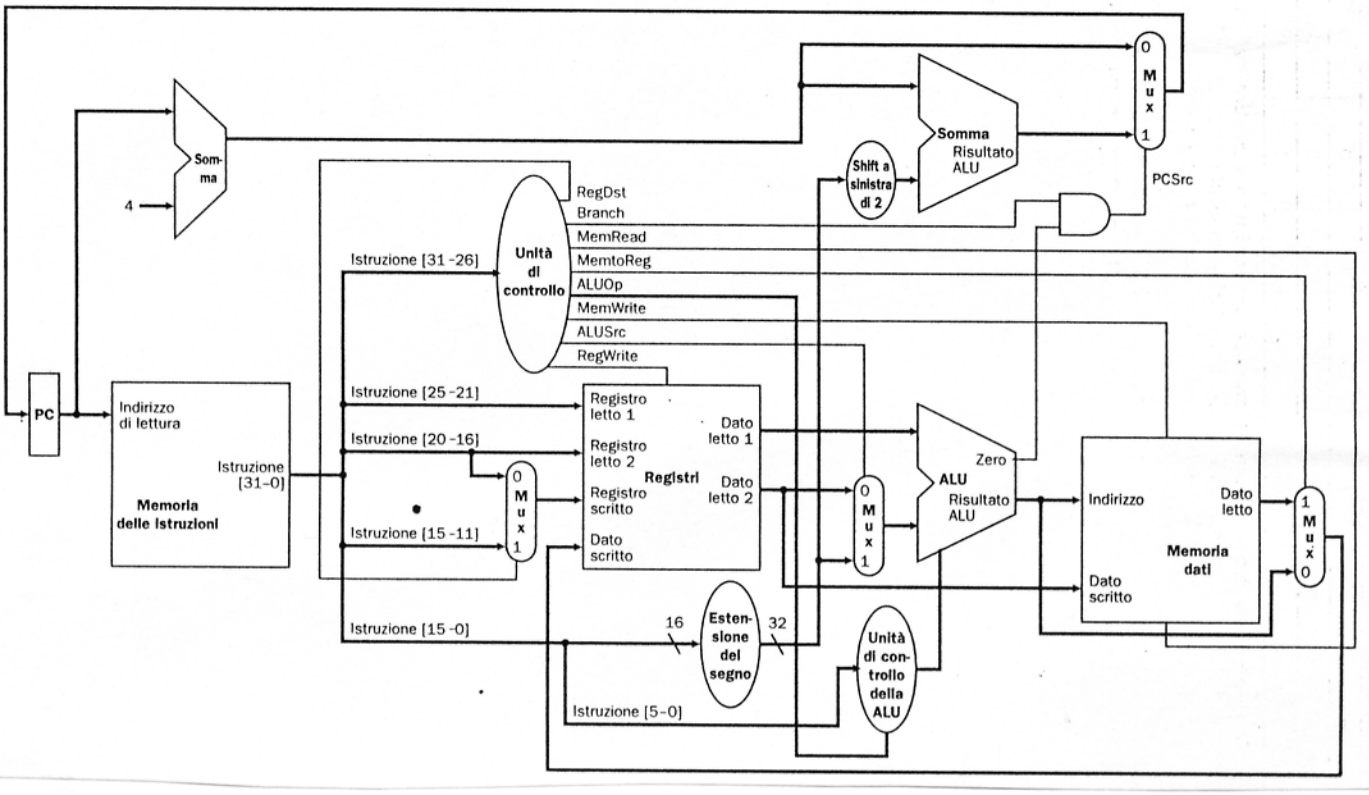
Sintesi SCO con PLA

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



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

PROGETTO SCO



	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

1° PASSO LOGICO ISTRUZIONE CLASSE R

FETCH : istruzione = MEMORIA [PC]
PC ← PC+4 (che viene completato successivamente)

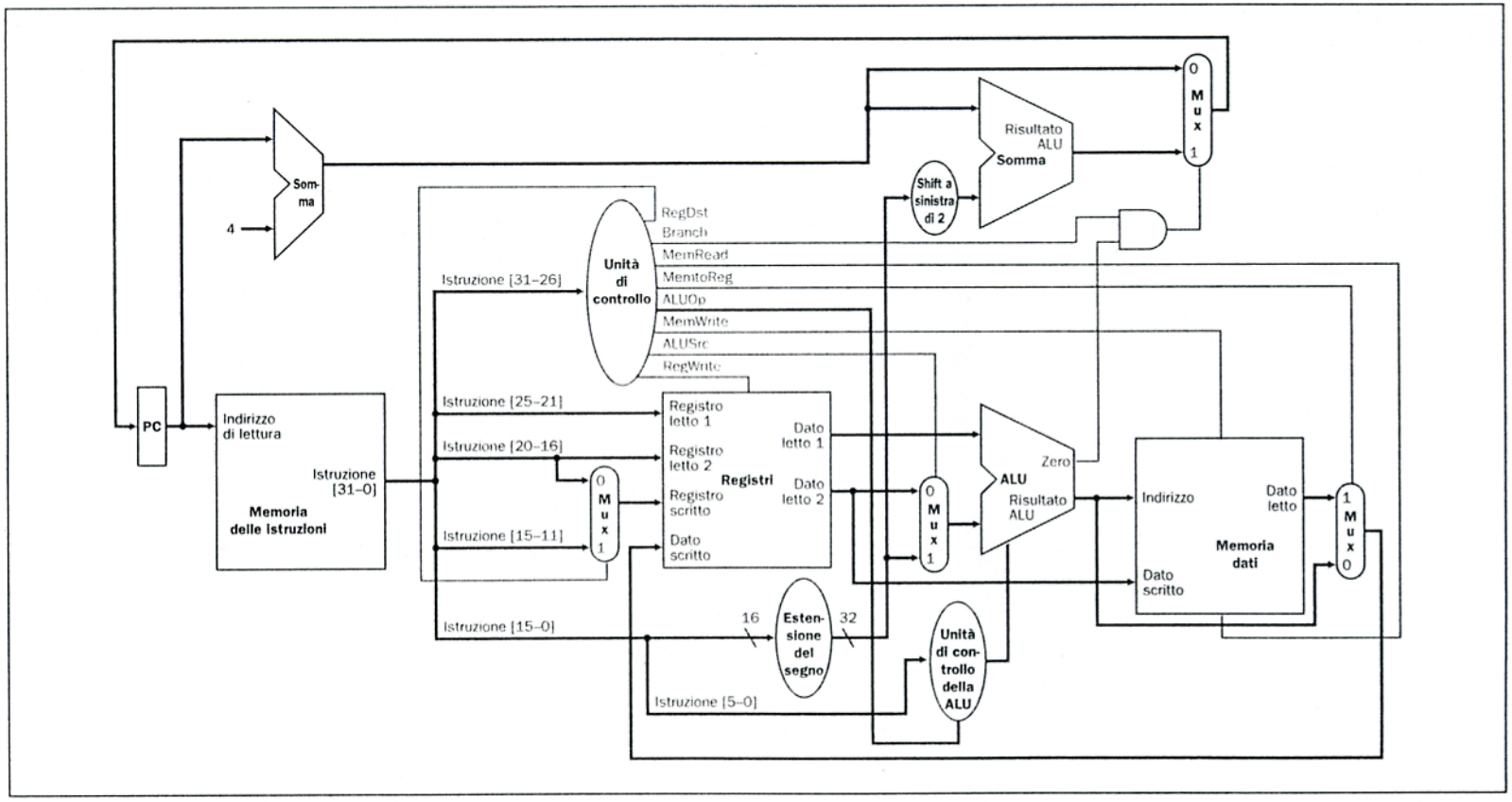


Figura 5.21 Il primo passo di un'istruzione di tipo-R preleva l'istruzione dalla memoria delle istruzioni ed incrementa PC. Le parti attive in questo passo sono evidenziate, mentre quelle disegnate in grigio chiaro non sono attive in questo passo ma lo diverranno in quelli successivi.

2° PASSO: Lettura dei due registri sorgenti logico

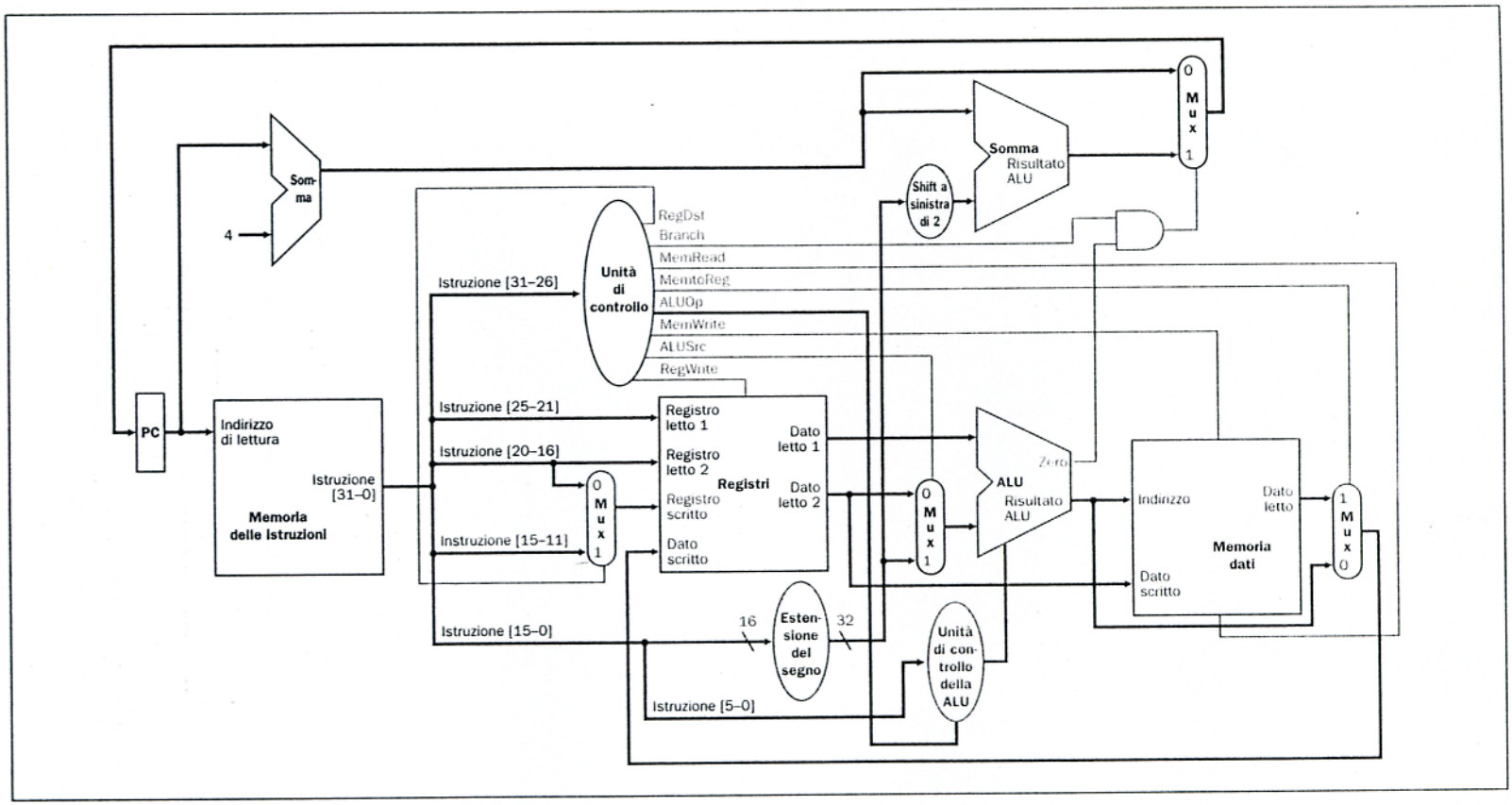


Figura 5.22 La seconda fase dell'esecuzione di un'istruzione di tipo-R legge i due registri sorgente dal register file. L'unità di controllo principale utilizza anche il campo opcode per determinare il valore dei segnali di controllo; queste unità diventano attive in aggiunta a quelle già attive durante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

MODIFICA SCO - SCA

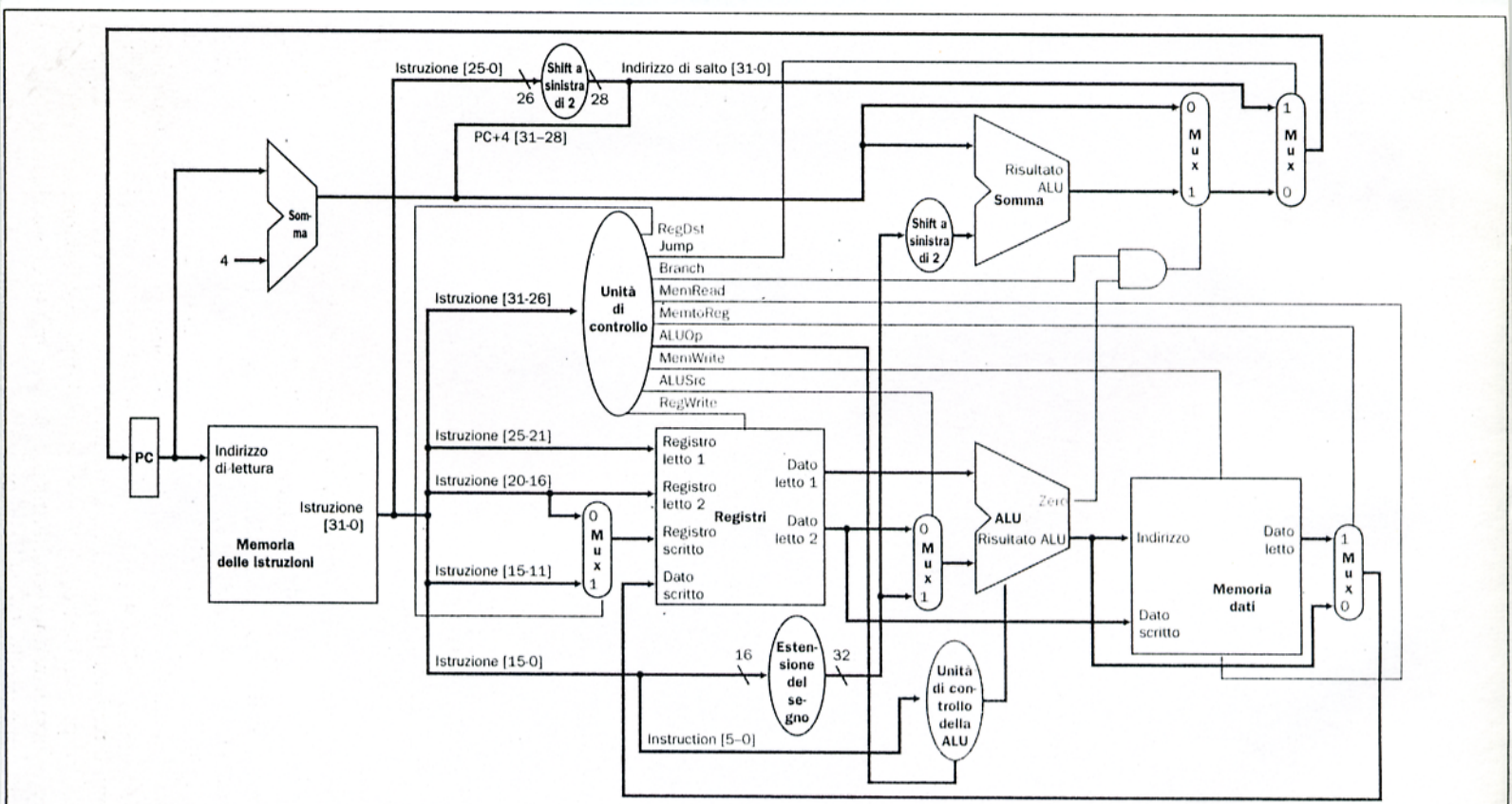
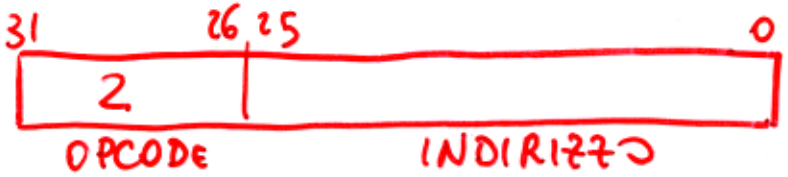


Figura 5.29 Estensione dell'unità di elaborazione elementare e della relativa unità di controllo per implementare l'istruzione jump. Viene utilizzato un multiplexer aggiuntivo (in alto a destra) per selezionare la destinazione del jump o, in alternativa, la destinazione del branch o l'istruzione seguente; 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.

estensione SCO-SCA

SALTO INCONDIZIONATO

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



26 bit

mi mancano 6

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

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

concatenato

DATA PATH (SCA) INTERESSATO AD UNA ISTRUZIONE DI SALTO CONDIZIONATO

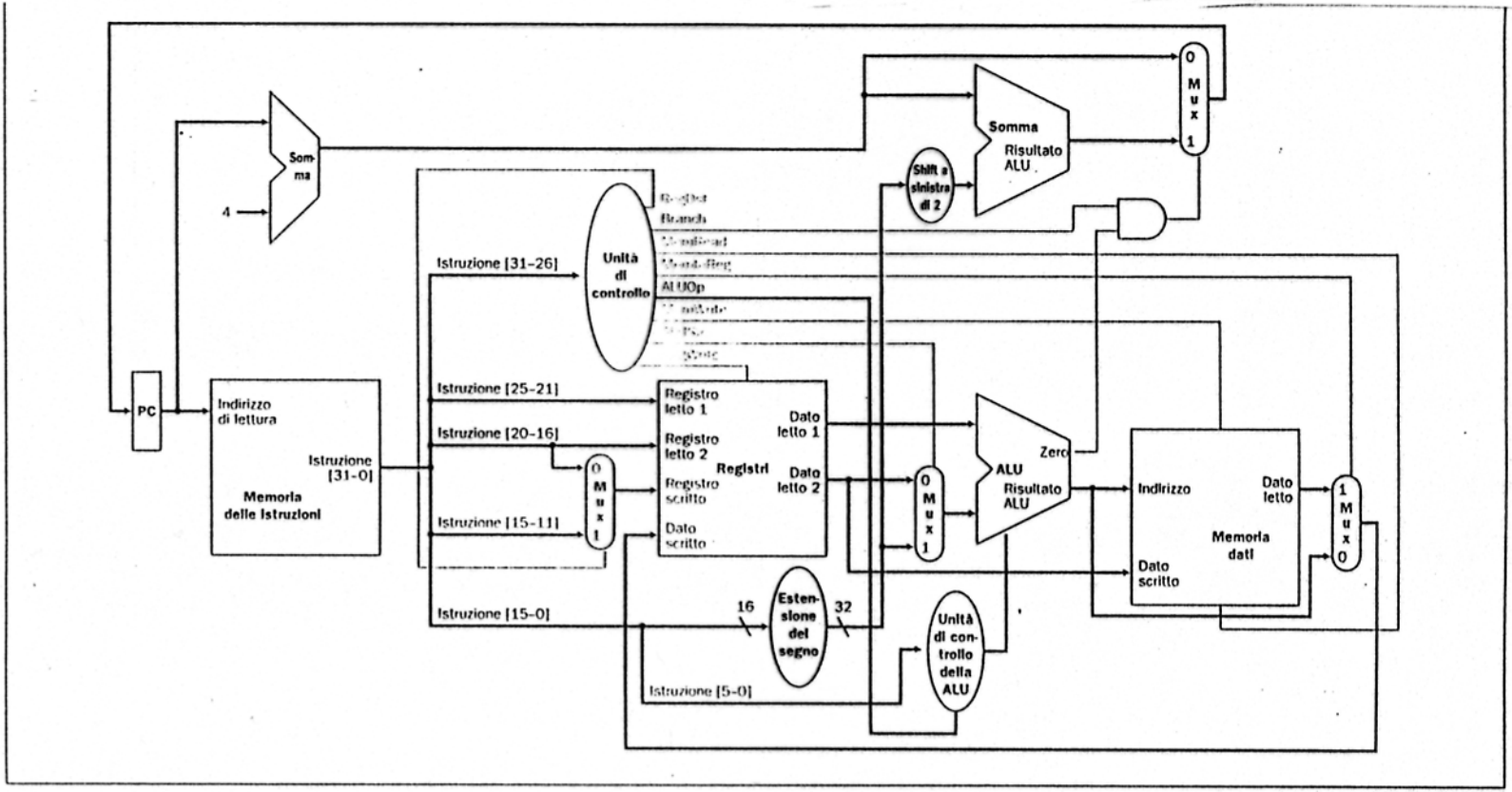


Figura 5.26 L'unità di elaborazione per un'istruzione branch equal. Dopo aver utilizzato il register file e la ALU per il confronto, l'uscita Zero è usata per scegliere il prossimo valore del program counter tra i due possibili.

DATA PATH INTERESSATO AD UNA ISTRUZIONE DI CARICAMENTO

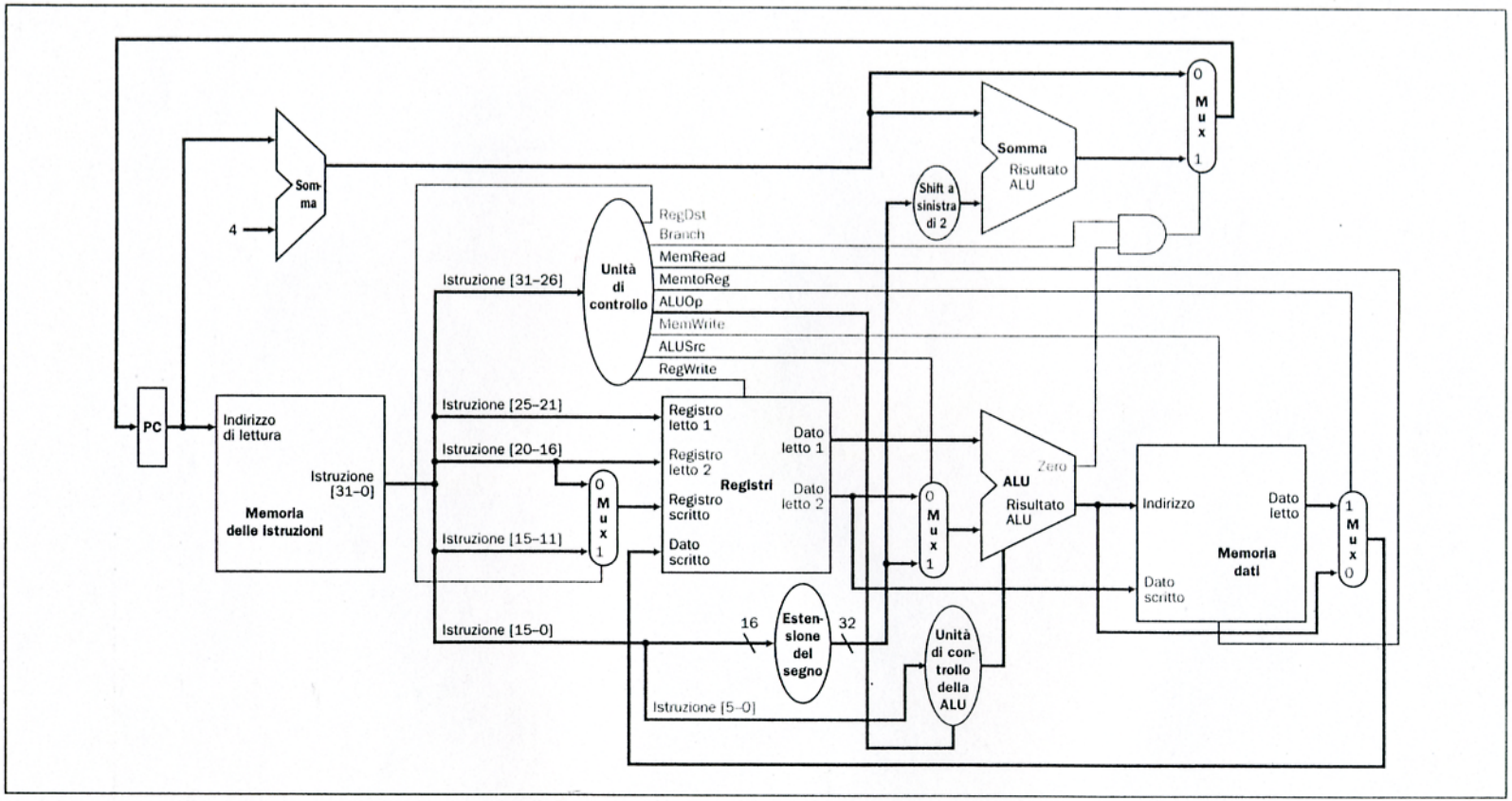


Figura 5.25 L'esecuzione di un'istruzione load nella unità di elaborazione elementare. Un'istruzione store si comporterebbe in modo molto simile: le differenze principali consterebbero dei segnali di controllo della memoria, che indicherebbero una scrittura anziché una lettura, del secondo registro letto, che conterrebbe il dato da memorizzare, e dell'operazione di scrittura del dato nel register file, che non avverrebbe.

4° PASSO : SCRITTURA RISULTATO ALU LOGICO NEL REGISTRO DESTINAZIONE

AGGIORNAMENTO PC

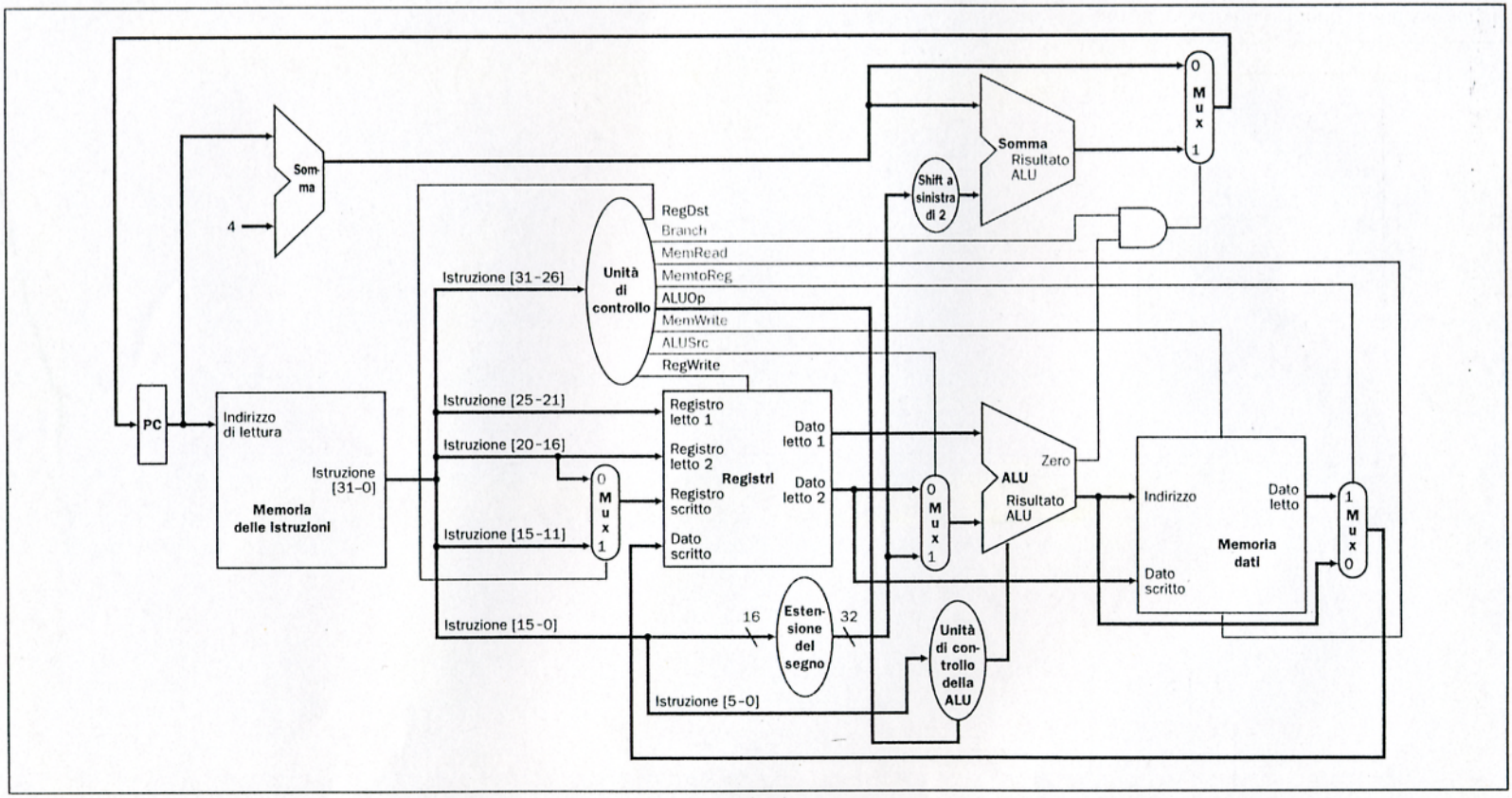


Figura 5.24 Il passo finale per le istruzioni di tipo-R prevede la scrittura del risultato; le relative unità attive sono state aggiunte a quelle dei tre passi precedenti, già indicate in figura 5.23. Al termine di questa fase avviene anche l'aggiornamento di PC. Essendo l'unità di elaborazione combinatoria, 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.

3° PASSO: ATTIVITÀ ALU

LOGICO

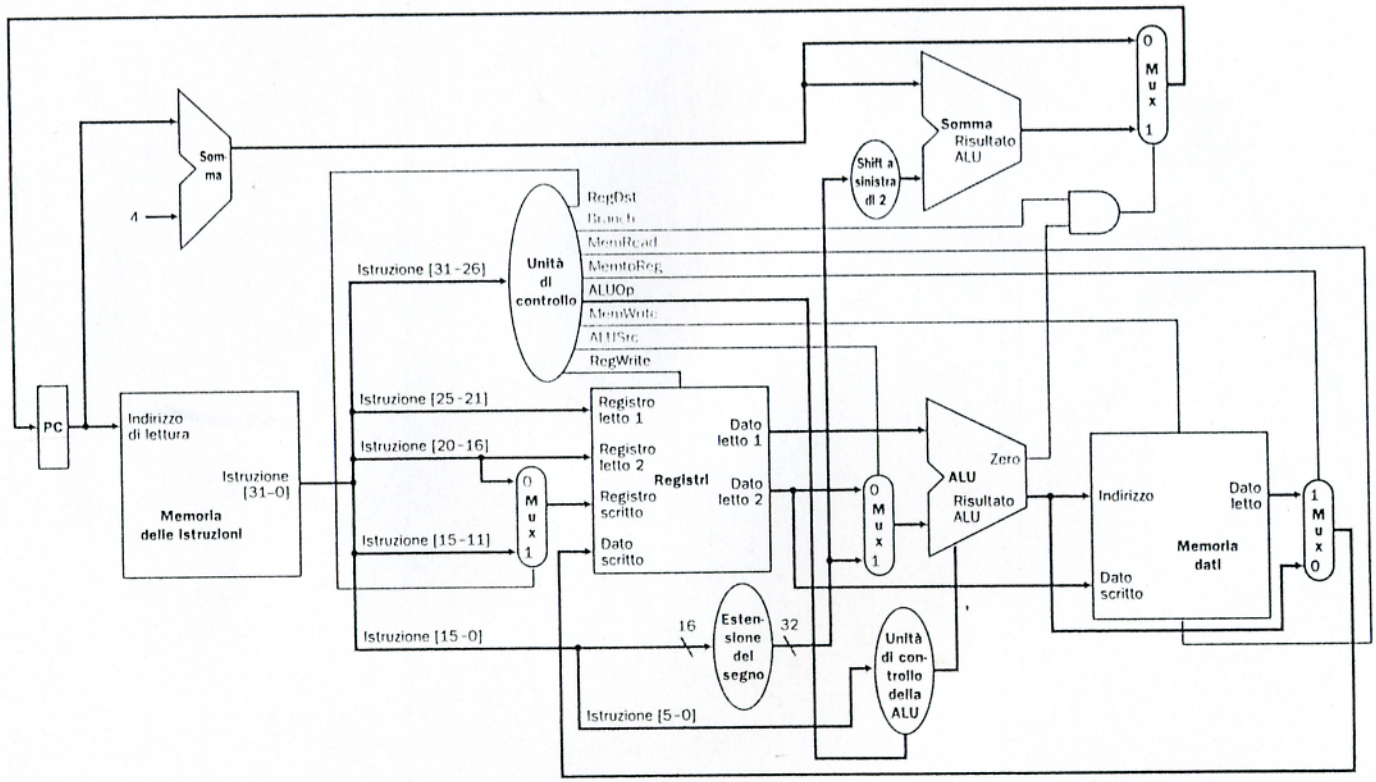


Figura 5.23 La terza fase dell'esecuzione delle istruzioni di tipo-R coinvolge la ALU, che elabora i valori dei registri utilizzati come operandi. Tutti i valori delle linee di controllo sono stati calcolati e l'unità di controllo della ALU ha calcolato la propria uscita; la ALU può quindi operare sui dati.

DETERMINAZIONE TEMPO DI CICLO

"IDENTIFICAZIONE PERCORSO PIU' LUNGO"

TIPO	UNITA' FUNZIONALI UTILIZZATE				
R	Prelievo istruz.	Accesso registri	ALU	Accesso registro	
lw	"	"	"	Accesso memoria	Accesso registro
sw	"	"	"	"	
beq	"	"	"		
J	"				

TIPO	MEMORIA ISTRUZ.	LETT. REG.	OPERAZ. ALU	MEMORIA DATI	SCRIT. REG.	TOT.
R	10 ns.	5 ns.	10 ns.	0 ns.	5 ns.	30
lw	10 ns.	5 ns.	10 ns.	10 ns.	5 ns.	40
sw	10 ns.	5 ns.	10 ns.	10 ns.		35
beq	10 ns.	5 ns.	10 ns.			25
J	10 ns.					10

VALORE MAX = 40 nsec.

LIMITI DEL PROGETTO A CICLO UNICO

- tutti i cicli istruzione eseguiti in 40 nsec indipendentemente dal loro tipo

↓ "ottimo"

avere un CK che cambi periodo in funzione del tipo di istruzione da eseguire

esempio di confronto: programma costituito da

22%	istr.	di	caricamento
11%	"	"	memorizzazione
49%	"	"	tipo R
16%	"	"	salto condizionato
2%	"	"	salto incondizionato

Tempo esecuzione $CPU_{CK\ singolo} = 40$

Tempo esecuzione $CPU_{CK\ variabile} = 31,6$

Conviene

↓

costoso da realizzare

↓

lo si virtualizza

tramite una realizzazione a

CK MULTIPLI

in cui ad ogni ciclo

di CK

viene eseguito un solo

passo dell'istruzione

↓

Tipo PD32