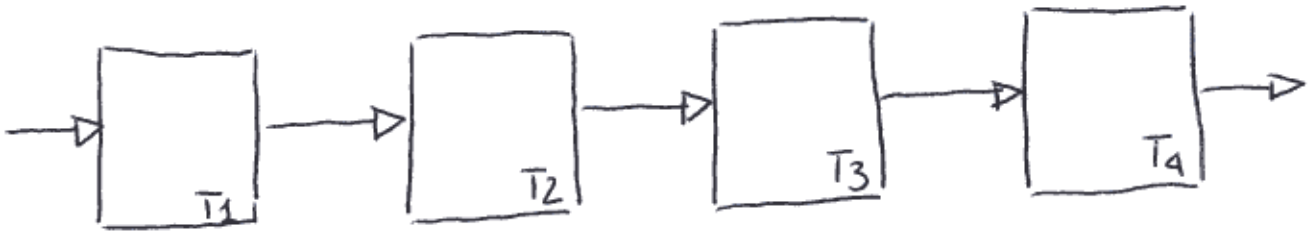
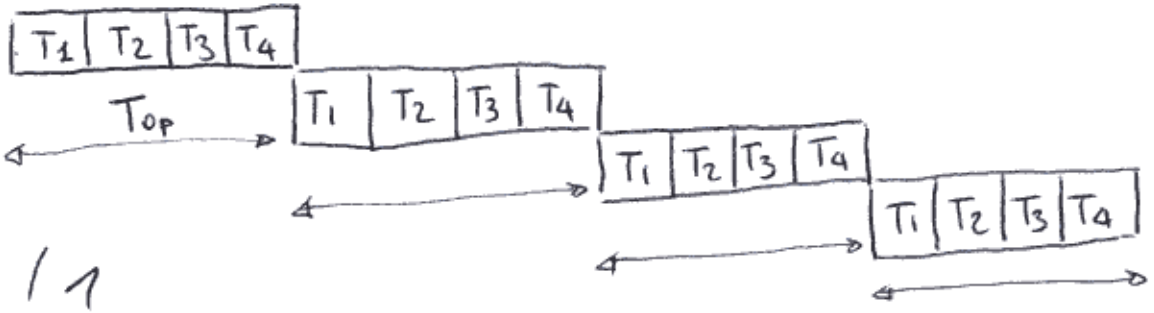
# COME MIGLIORARE LE PRESTAZIONI TRAMITE ORGANIZZAZIONE A PIPELINE

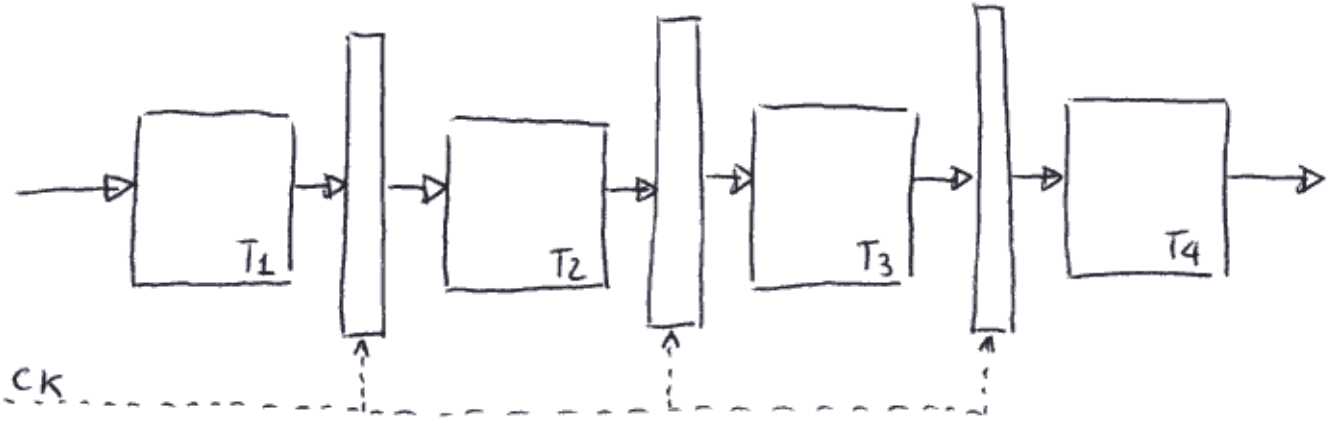


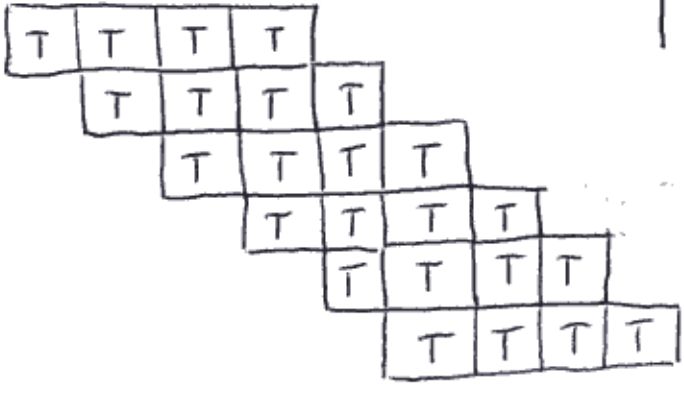




1/1

tempo fra due elaborazioni indipendenti $= T_{op} = T_1 + T_2 + T_3 + T_4$

$$\text{Throughput} = \frac{1}{T_{op}}$$




$$T = max(T_1, T_2, T_3, T_4)$$

$$T_{op} = 4 * T$$

$$Throughput = \frac{1}{T}$$

| Instruction class | Instruction fetch | Register read | ALU operation | Data access | Register write | Total time |
|---|---|---|---|---|---|---|
| Load word (`lw`) | 2 ns | 1 ns | 2 ns | 2 ns | 1 ns | 8 ns |
| Store word (`sw`) | 2 ns | 1 ns | 2 ns | 2 ns | | 7 ns |
| R-format (`add, sub, and, or, slt`) | 2 ns | 1 ns | 2 ns | | 1 ns | 6 ns |
| Branch (`beq`) | 2 ns | 1 ns | 2 ns | | | 5 ns |

**FIGURE 6.2 Total time for eight instructions calculated from the time for each component.** This calculation assumes that the multiplexors, control unit, PC accesses, and sign extension unit have no delay.

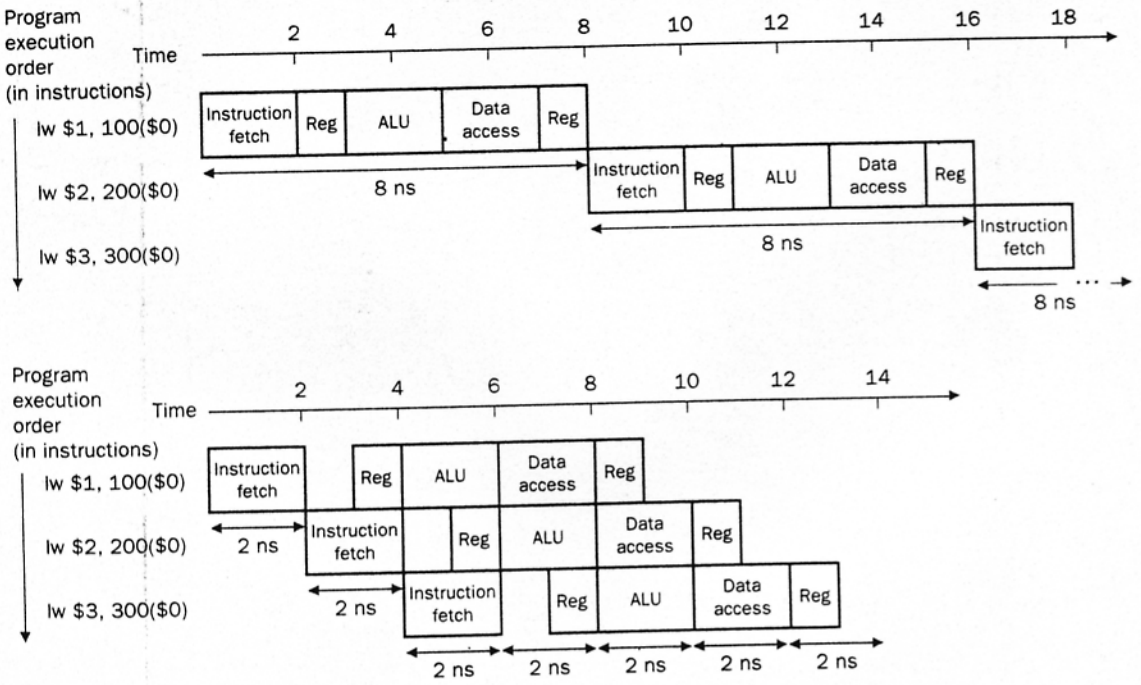
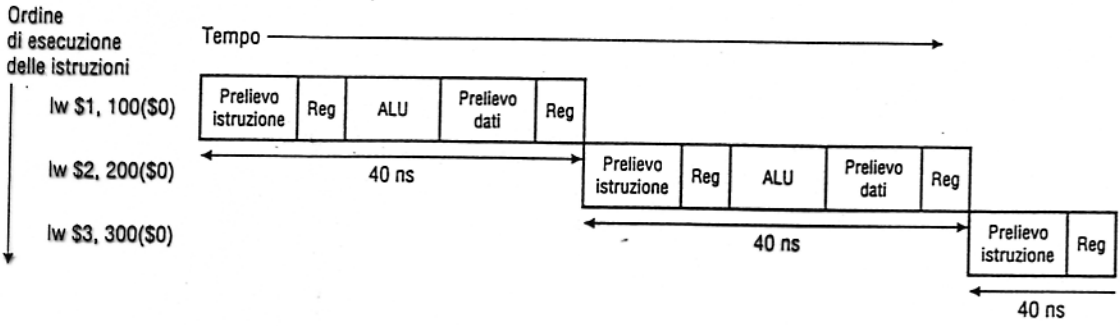


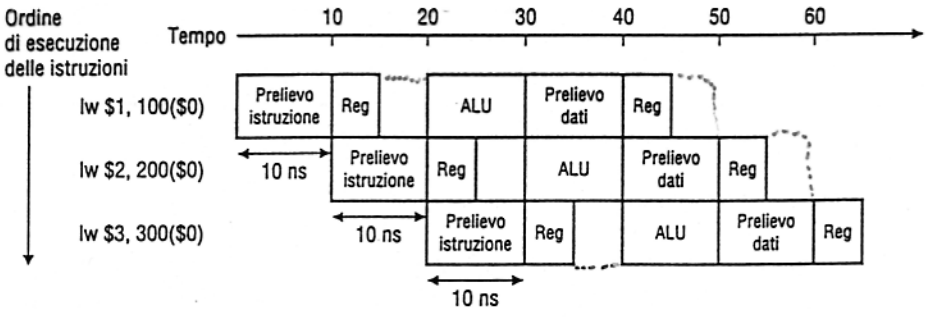

**FIGURE 6.3 Single-cycle, nonpipelined execution in top vs. pipelined execution in bottom.** Both use the same hardware components, whose time is listed in Figure 6.2. In this case we see a fourfold speedup on average time between instructions, from 8 ns down to 2 ns. Compare this figure to Figure 6.1. For the laundry, we assumed all stages were equal. If the dryer were slowest, then the dryer stage would set the stage time. The computer pipeline stage times are limited by the slowest resource, either the ALU operation or the memory access. We assume the write to the register file occurs in the first half of the clock cycle and the read from the register file occurs in the second half. We use this assumption throughout this chapter.

# CASO PROCESSORE

| Tipo di istruzione | Memoria istruzioni | Lettura registro | Operazione ALU | Memoria dati | Scrittura registro | Tempo totale |
|---|---|---|---|---|---|---|
| lw | 10 ns | 5 ns | 10 ns | 10 ns | 5 ns | 40 ns |
| sw | 10 ns | 5 ns | 10 ns | 10 ns | | 35 ns |
| add, sub, and, or | 10 ns | 5 ns | 10 ns | | 5 ns | 30 ns |
| beq | 10 ns | 5 ns | 10 ns | | | 25 ns |

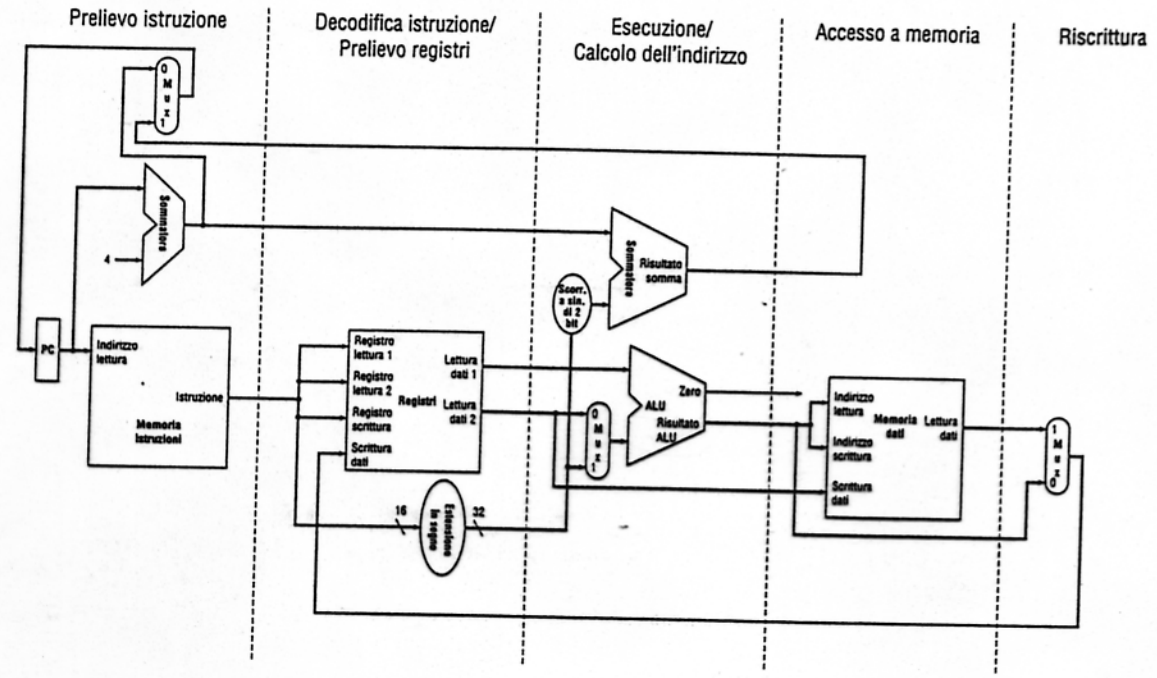


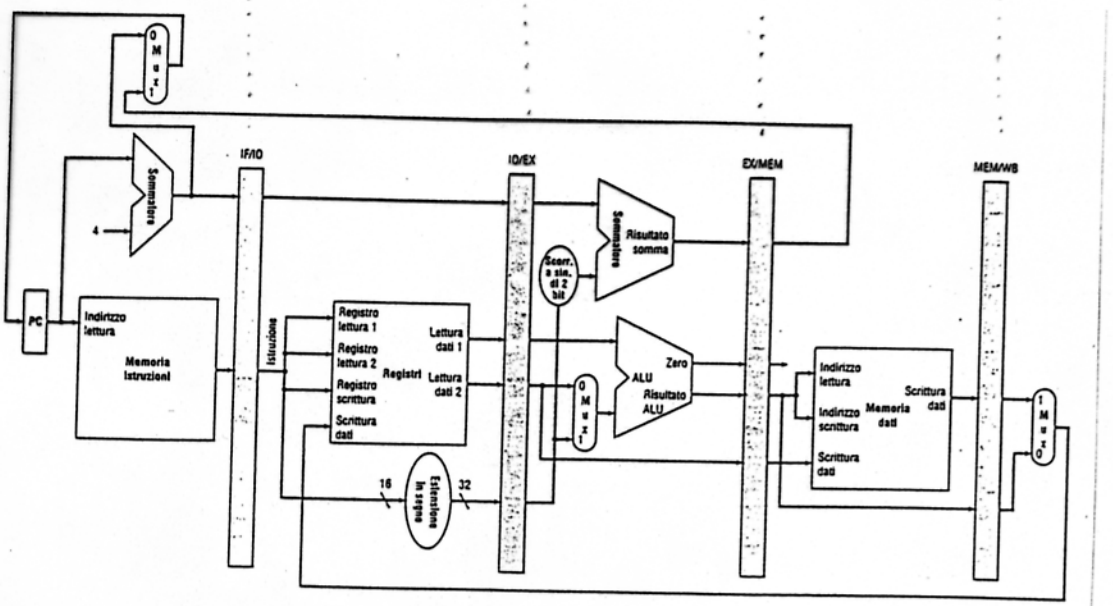

CPU a ciclo singolo



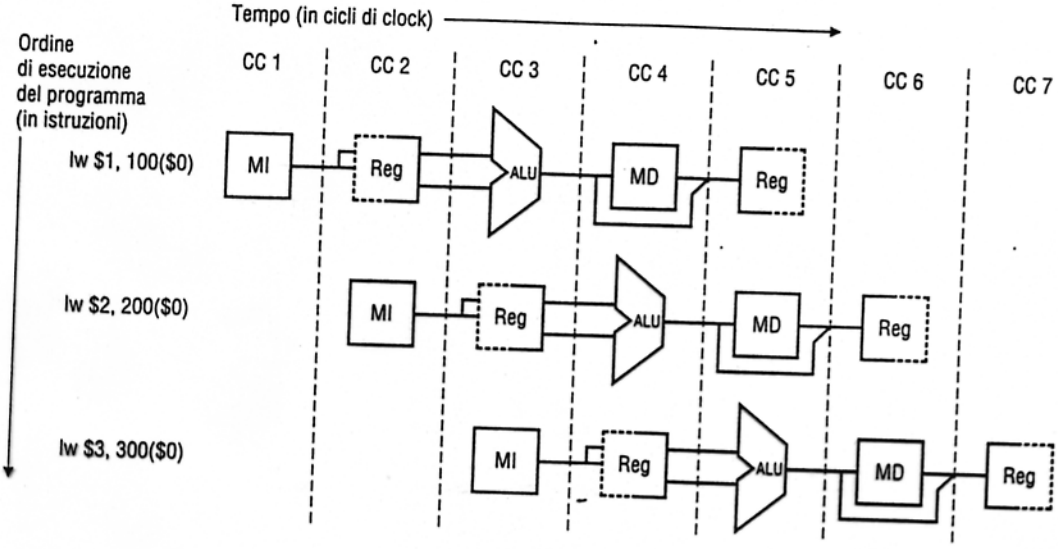

CPU a pipeline

# DAL CICLO SINGOLO AL PIPELINE



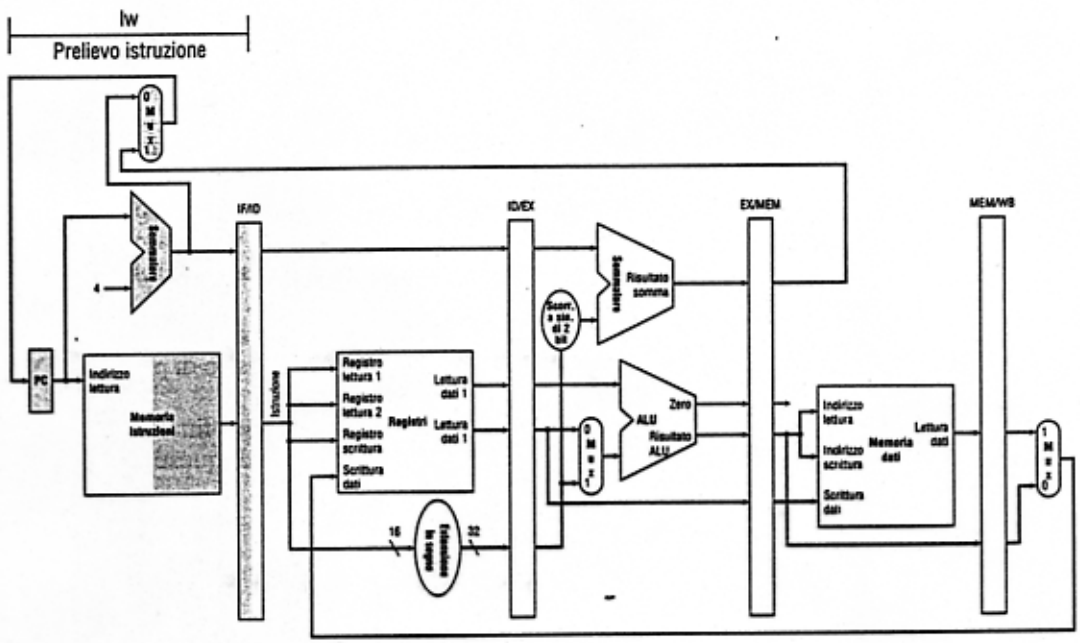
# RAPPRESENTAZIONE SCHEMATICA DELLE ATTIVITA' DELLA STRUTTURA A PIPELINE



ESECUZIONE ISTRUZIONE

lw $1, offset ($2) ------ $1 = MEMORIA DATI ($2+offset)

1° PASSO

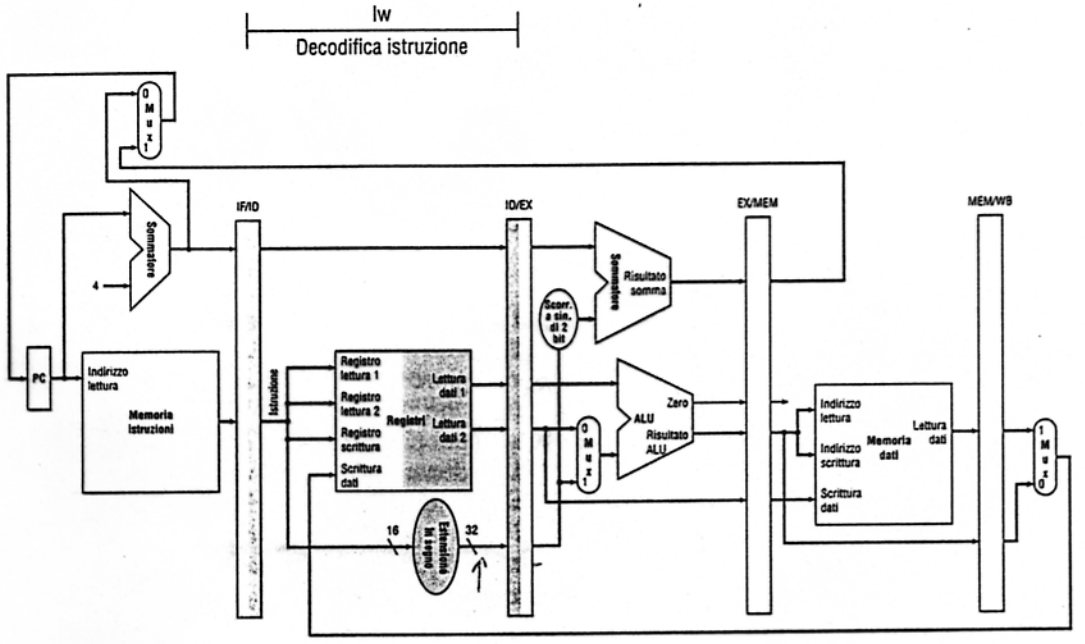



Mem data ← MEMORIA ISTR [PC]

PC ← PC+4

IF/ID ← { PC+4 ; Mem data }

# 2° PASSO

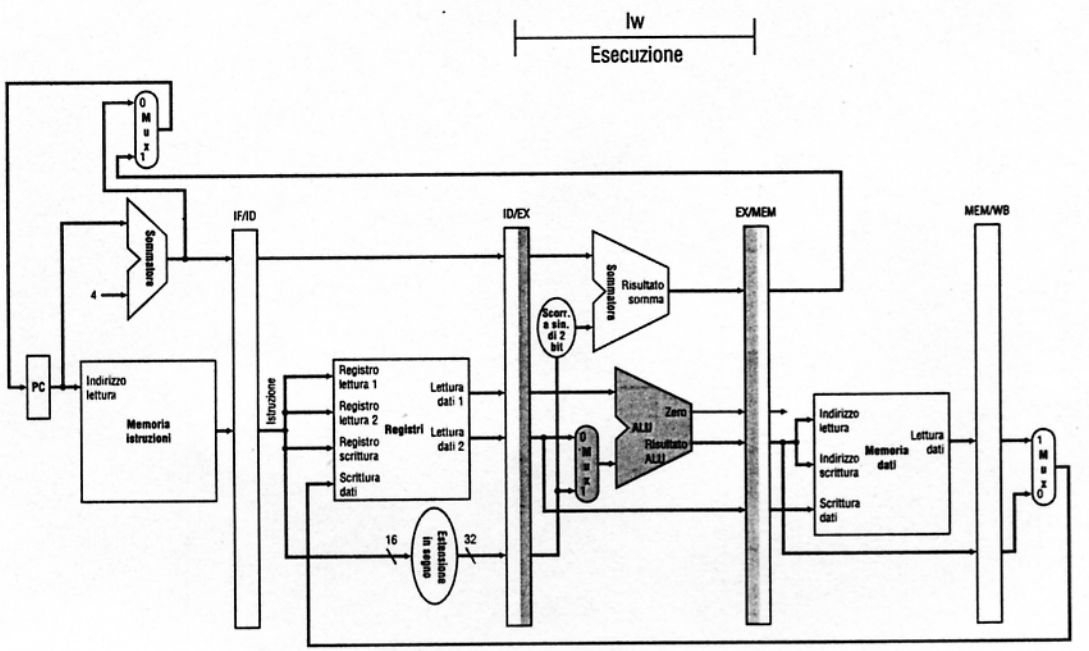



Lettura dati1 ← REGISTRO (RS) ---- $2

OUTestensione ← OFFSET

ID/EX ← { sottoinsieme di IF/ID (non interessa)
Lettura dati 1
Lettura dati 2 (non interessa)
OUTestensione

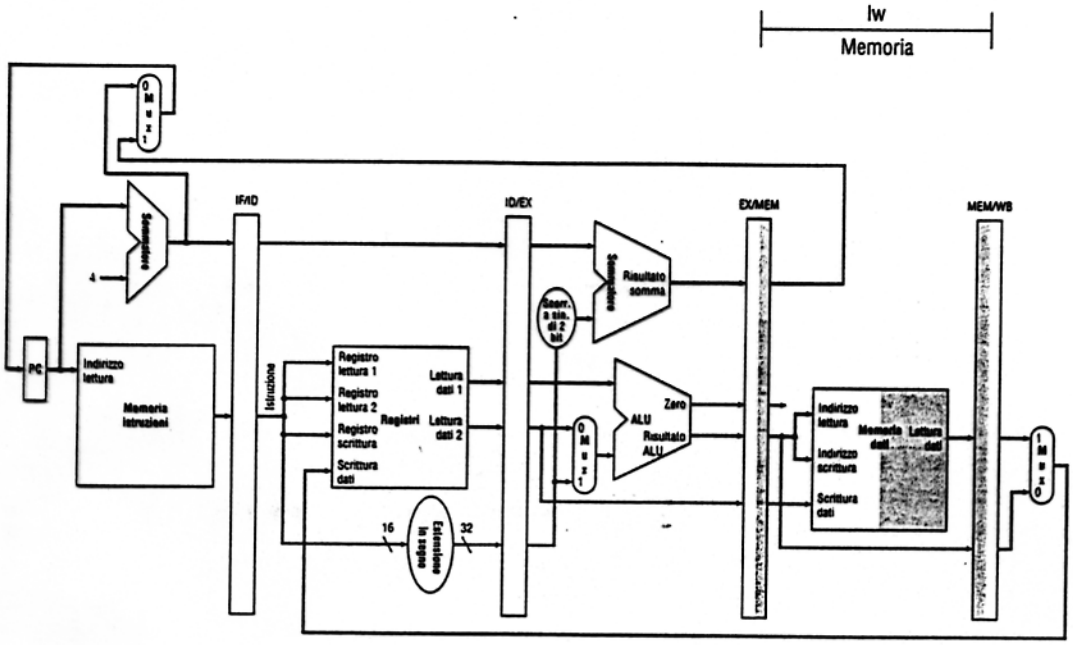
# 3° PASSO




Risultato ALU ← $2 + offset

EX/MEM ←
- Risultato somma (non interessa)
- zero (non interessa)
- Risultato ALU
- ID/EX. Lettura dati 2 (non interessa)

# 4° PASSO

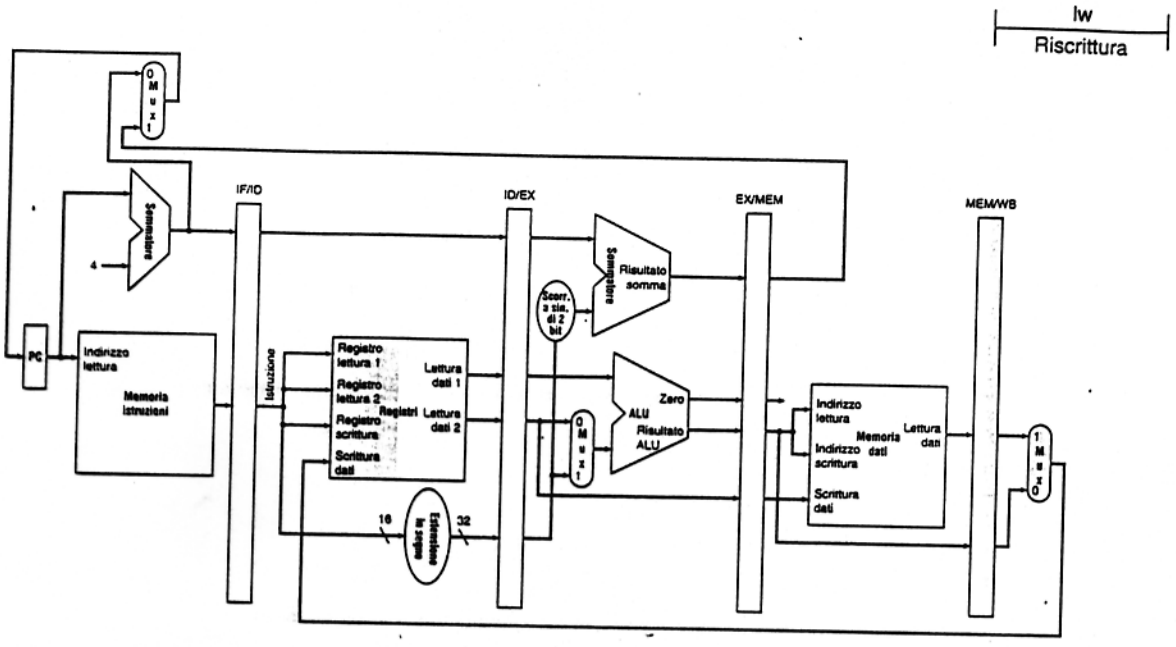



LeHura dati ← MEMORIA DATI [indirizzo lettura]

MEM/WB ← { LeHura dati ; EX/EM. Risultato ALU

(non interessa)
potrebbe contenere
il risultato di una
operazione sul
contenuto di due regist
che deve essere memorizz
zato in un terzo regist

5° passo




Registro scrittura ← MEM/WB. Lettura dati

?

dove è registrato l'identificativo del registro scrittura? ERRORE

SCA CORRETTA

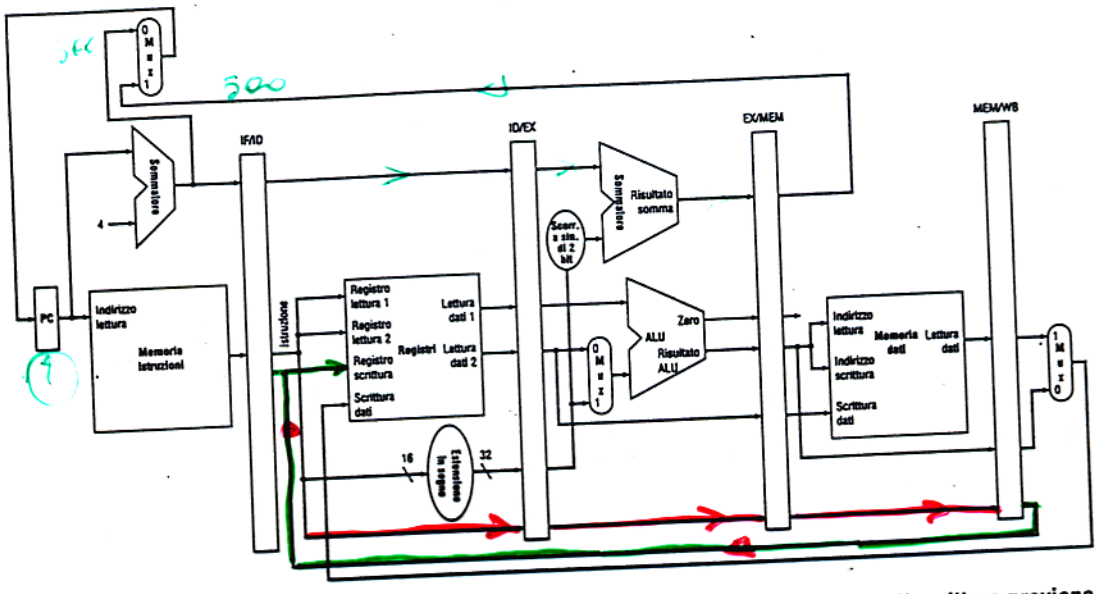



**FIGURA 6.12** L'unità di calcolo corretta per l'istruzione di caricamento. Il numero di registro di scrittura proviene ora dal registro di pipeline MEM/WB insieme ai dati. Il numero di registro è fatto passare attraverso lo stadio di pipeline ID finché non raggiunge il registro di pipeline MEM/WB. Il nuovo percorso è mostrato in colore.

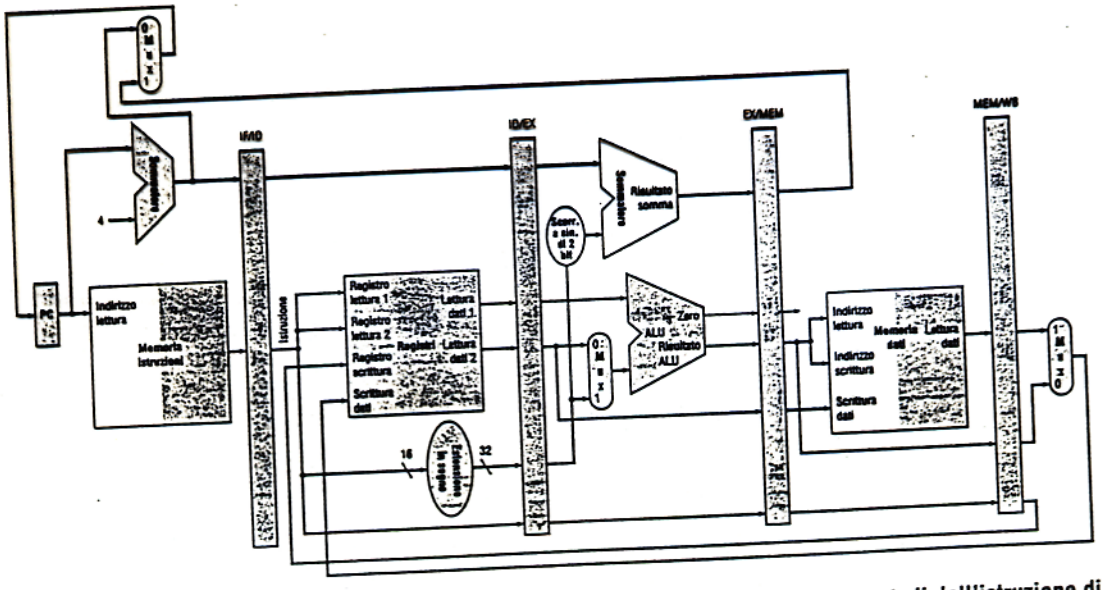
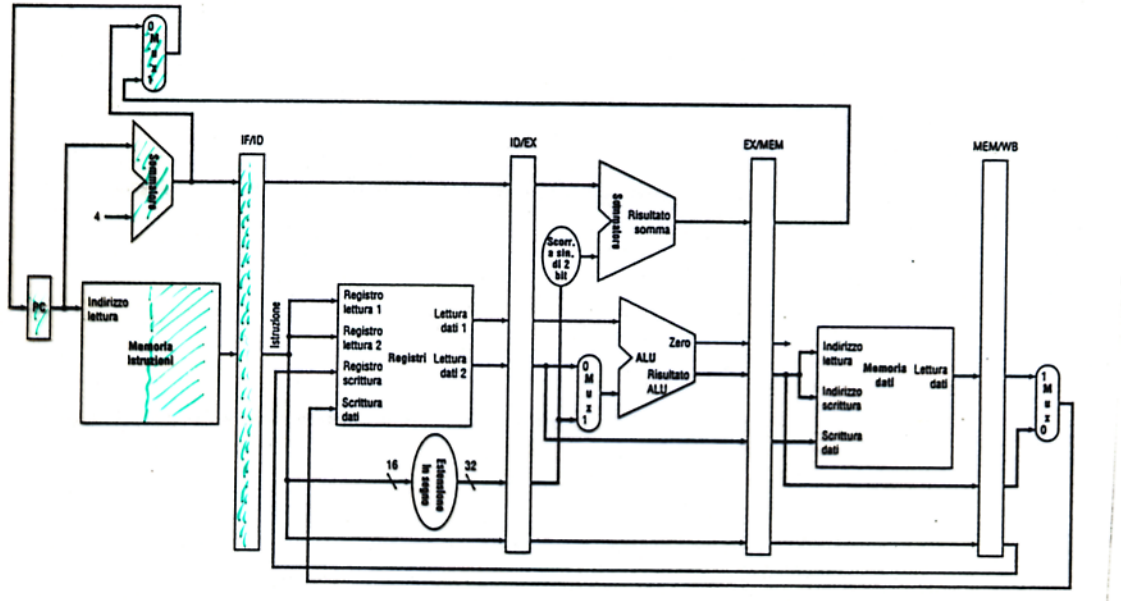



**FIGURA 6.13** La porzione di unità di calcolo nella figura 6.12 che è utilizzata in tutti i cinque stadi dell'istruzione di caricamento.

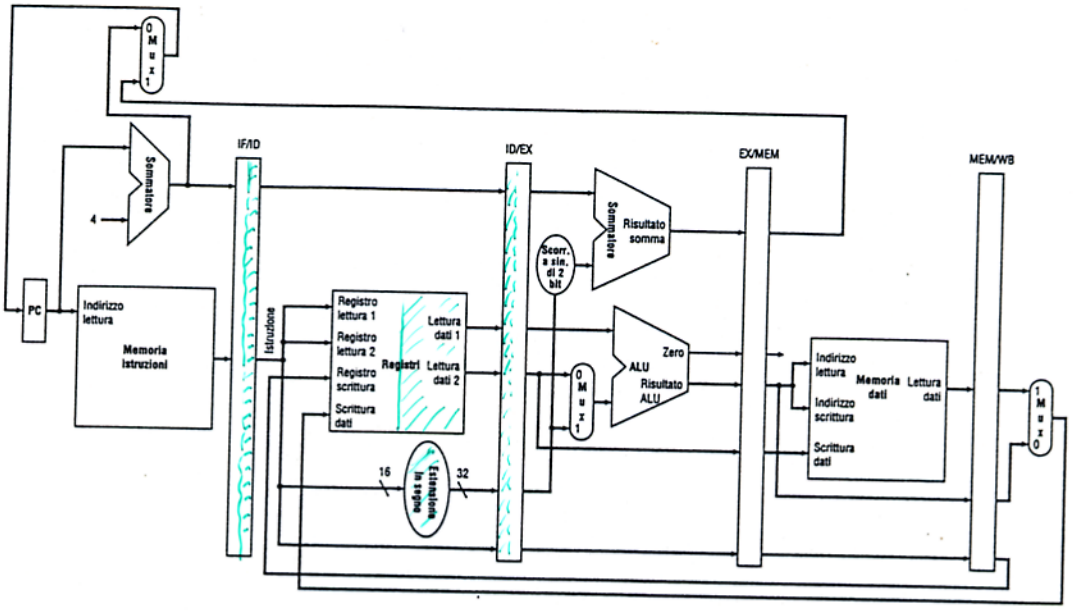
ESEMPIO DI ISTRUZIONE CHE NECESSITA DI QUATTRO PASSI

SW $1, offset($2) ---- MEMORIADATI ($2+OFFSET) ← $1

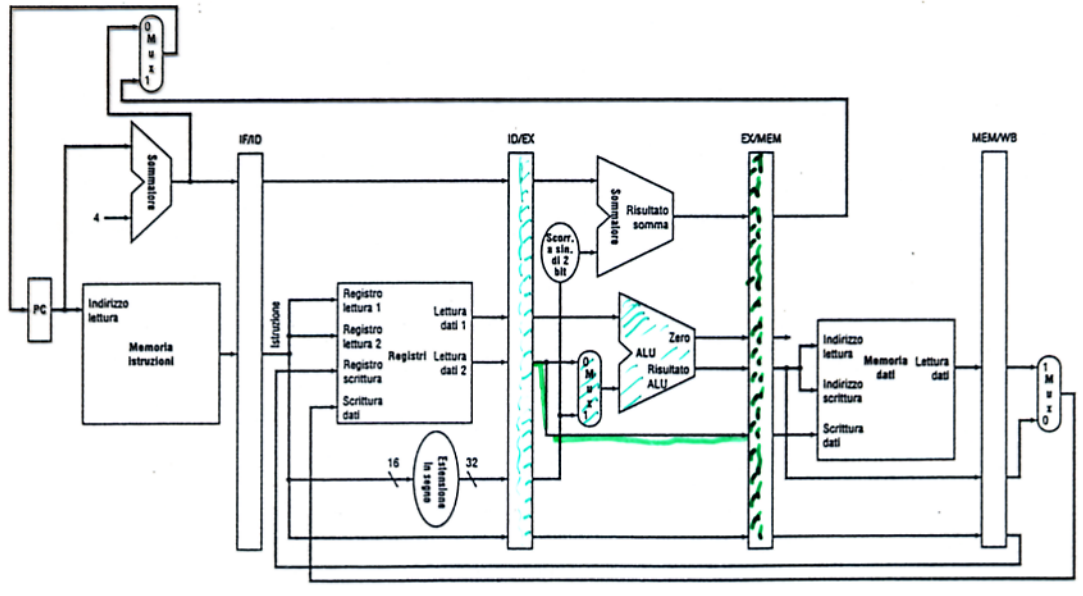
1° PASSO : fetch



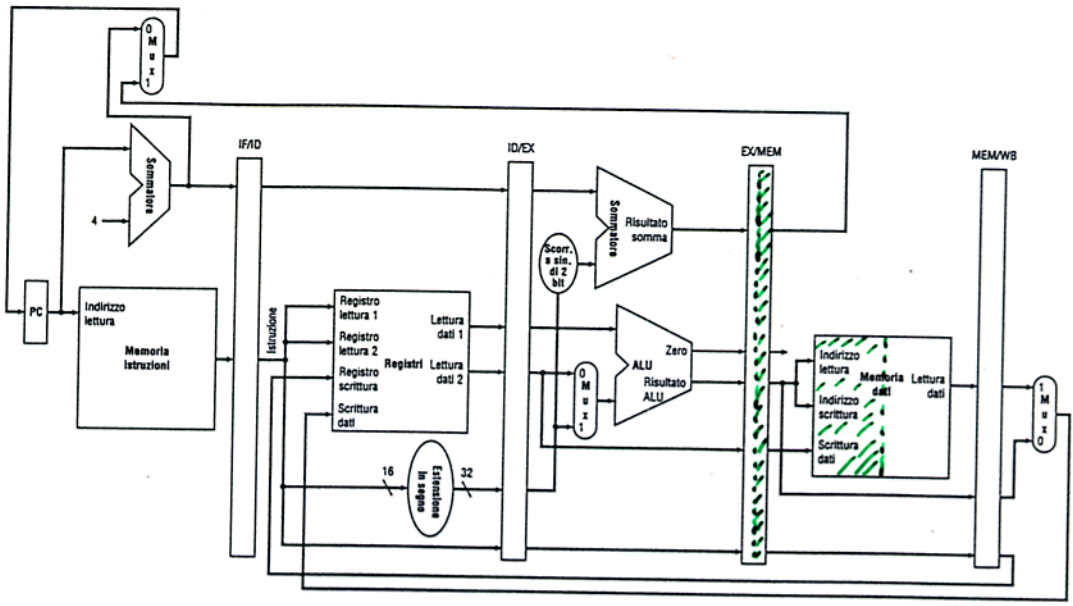
# 2° PASSO: lettura registro - preparazione offset



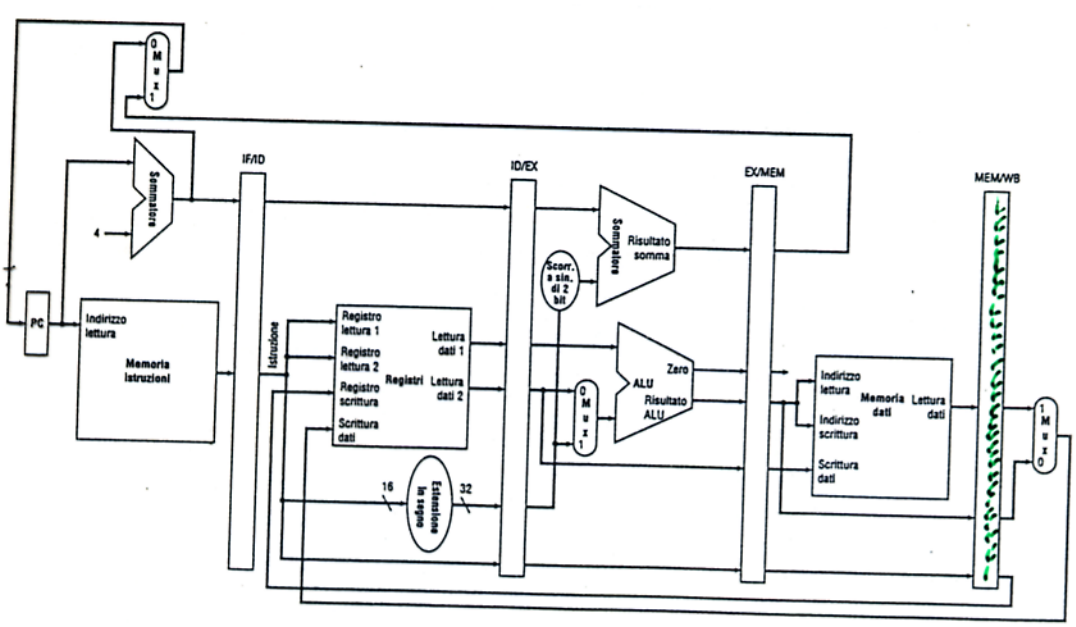

# 3° PASSO: esecuzione somma



# 4° PASSO : memorizzazione dato




# 5° PASSO: non accade nulla .



# RAPPRESENTAZIONE GRAFICA DELLE ATTIVITA' DEL PIPELINE

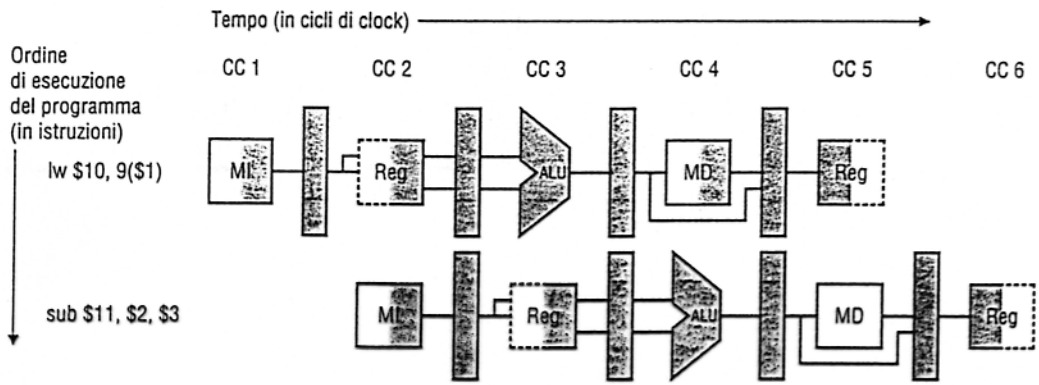



**FIGURA 6.14 Diagramma di due istruzioni della pipeline multiciclo. La figura 6.15 mostra il metodo tradizionale per rappresentare tale diagramma, mentre le figure da 6.16 a 6.18 danno i diagrammi delle stesse istruzioni per la pipeline a ciclo di clock singolo.**

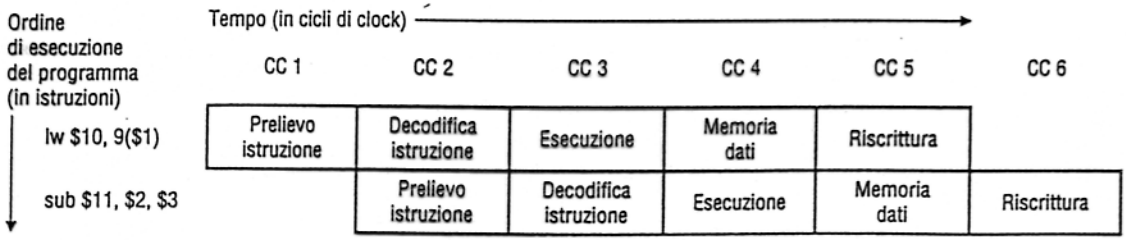



**FIGURA 6.15 Diagramma tradizionale delle due istruzioni nella figura 6.14 per la pipeline multiciclo.**

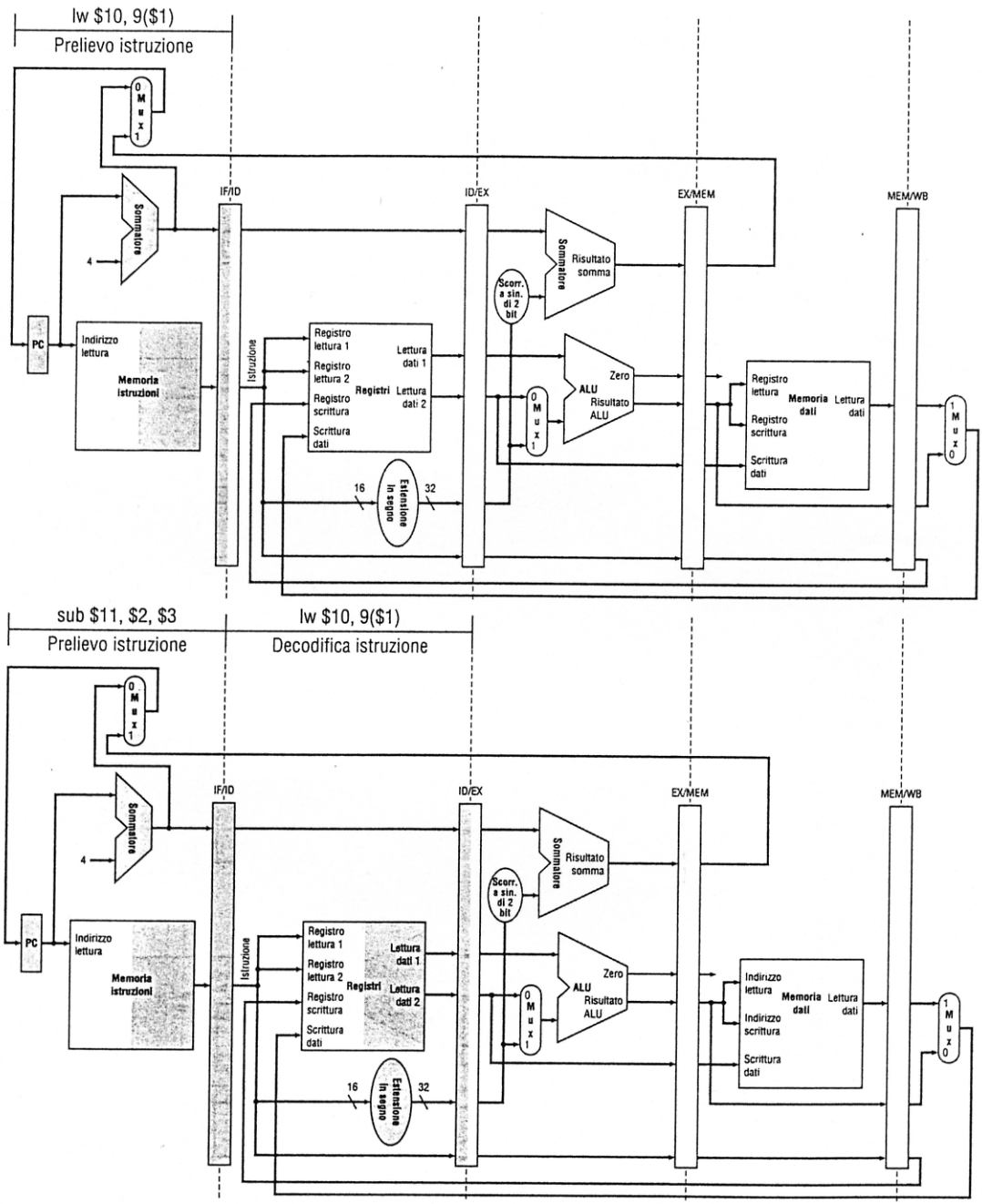



FIGURA 6.16 I diagrammi di una pipeline a ciclo singolo per i cicli di clock 1 (diagramma in alto) e 2 (diagramma in basso). Le parti dell'unità di calcolo messe in evidenza sono quelle attive durante tale ciclo di clock. Il caricamento viene prelevato nel ciclo di clock 1 e decodificato nel ciclo di clock 2, con la sottrazione prelevata nel secondo ciclo di clock.

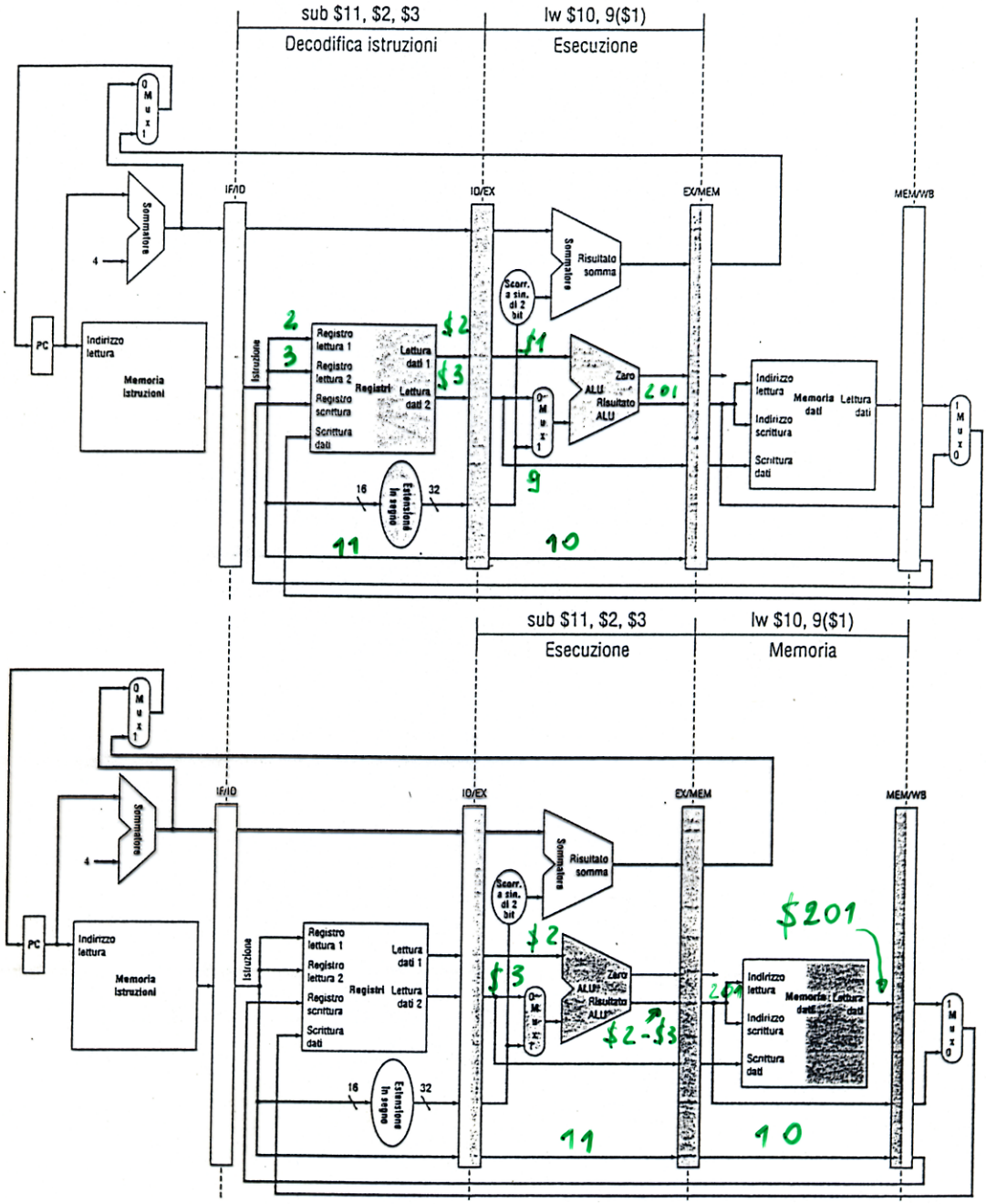



FIGURA 6.17 I diagrammi di una pipeline a ciclo singolo per i cicli di clock 3 (diagramma in alto) e 4 (diagramma in basso). Nel terzo ciclo di clock nel diagramma in alto, `lw` entra nello stadio EX. Contemporaneamente, `sub` entra nello stadio ID. Nel quarto ciclo di clock (unità di calcolo inferiore), `lw` prosegue nello stadio MEM, leggendo la memoria all'indirizzo trovato in EX/MEM all'inizio del ciclo di clock 4. Allo stesso tempo, l'ALU sottrae e pone la differenza in EX/MEM alla fine del ciclo di clock.

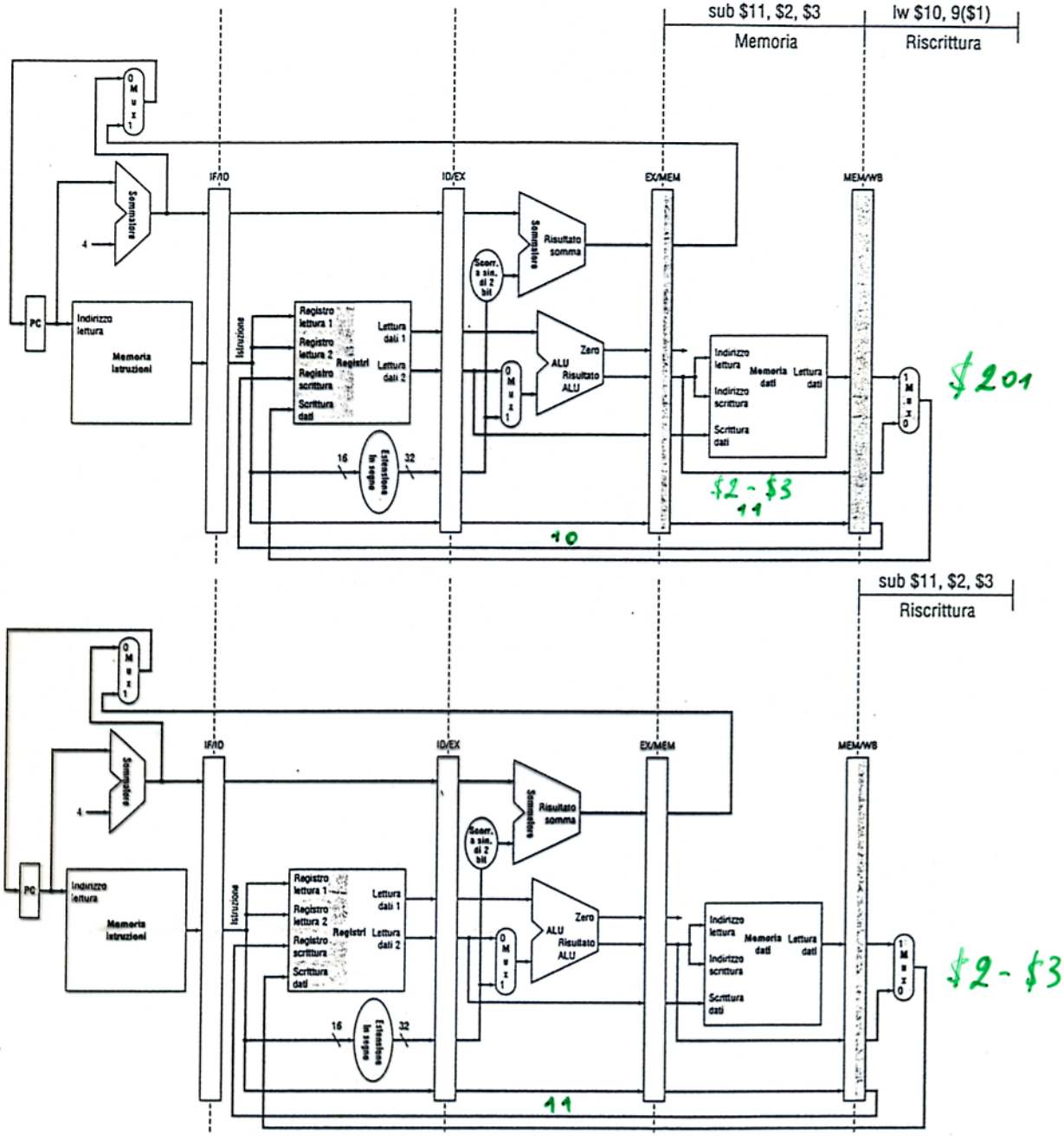
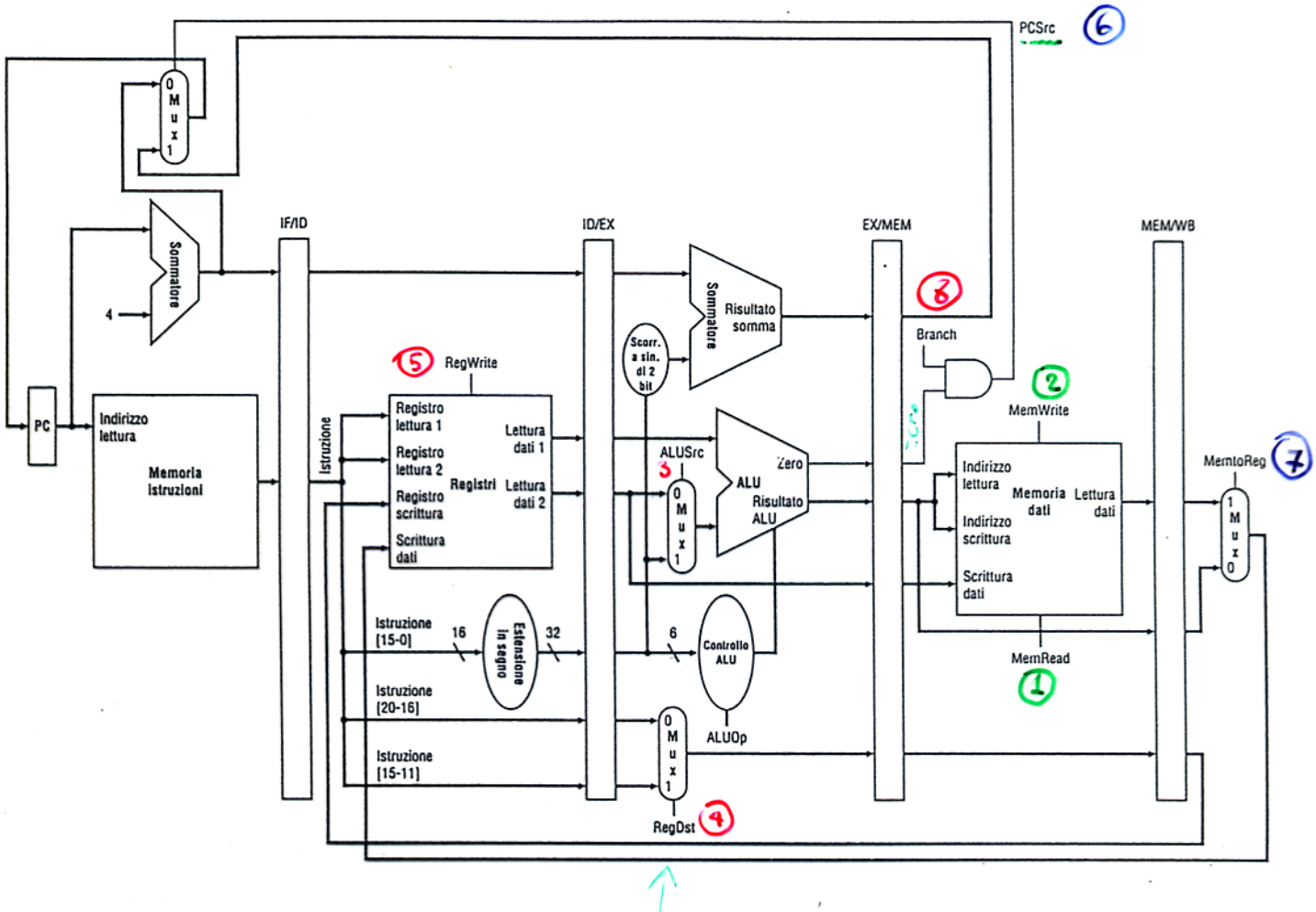



FIGURA 6.18 I diagrammi di una pipeline a ciclo singolo per i cicli di clock 5 (diagramma in alto) e 6 (diagramma in basso). Nel ciclo di clock 5, `lw` viene completata con la scrittura dei dati contenuti in MEM/WB nel registro 10 e `sub` invia la differenza scritta precedentemente in EX/MEM a MEM/WB. Nel ciclo di clock successivo, `sub` scrive il valore di MEM/WB in un registro.

# PROGETTO SCO

## SEGNALI DI CONTROLLO o "TASK




| | Nome del segnale | Effetto quando non asserito (0) | Effetto quando asserito (1) |
|---|---|---|---|
| 1 | MemRead | Nessuno | Il contenuto della memoria dati all'indirizzo di lettura viene scritto sull'uscita lettura dati. |
| 2 | MemWrite | Nessuno | Il contenuto della memoria dati all'indirizzo di scrittura viene sostituito da quello all'ingresso scrittura dati. |
| 3 | ALUSrc | Il secondo operando dell'ALU è la seconda uscita del banco dei registri. | Il secondo operando dell'ALU sono i 16 bit estesi in segno dell'istruzione. |
| 4 | RegDst | Il numero del registro di destinazione per la scrittura del registro viene dal campo rt. (lw) | Il numero del registro di destinazione per la scrittura del registro viene dal campo rd. (R) |
| 5 | RegWrite | Nessuno | Il registro indicato dal numero del registro scrittura viene scritto con il valore all'ingresso scrittura dati. |
| 6 | PCSrc *1 | Il PC viene sostituito dall'uscita del sommatore che calcola il valore PC + 4. | Il PC viene sostituito dall'uscita del sommatore che calcola la destinazione del salto condizionato. |
| 7 | MemtoReg | Il valore fornito all'ingresso registro scrittura dati proviene dall'ALU. | Il valore fornito all'ingresso registro scrittura dati proviene dalla memoria dati. |

8 Branch  PC ← PC+4  se zero = 0 allora PC ← PC+4

se zero = 1 allora PC ← PC+4 + offset*4

*1 "generato" da Branch

| | EX | | | | M | | | WB | |
|---|---|---|---|---|---|---|---|---|---|
| | Linee di controllo stadio di esecuzione | | | | Linee di controllo stadio di memoria | | | Linee di controllo stadio di riscrittura | |
| Istruzione | RegDst | ALUOp1 | ALUOp0 | ALUSrc | Branch | MemRead | MemWrite | RegWrite | MemtoReg |
| Tipo R | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| `lw` | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| `sw` | X | 0 | 0 | 1 | 0 | 0 | 1 | 0 | X |
| `beq` | X | 0 | 1 | 0 | 1 | 0 | 0 | 0 | X |

**FIGURA 6.22** I valori di queste linee di controllo sono gli stessi della figura 5.23, riorganizzati in tre gruppi corrispondenti agli ultimi tre stadi della pipeline.

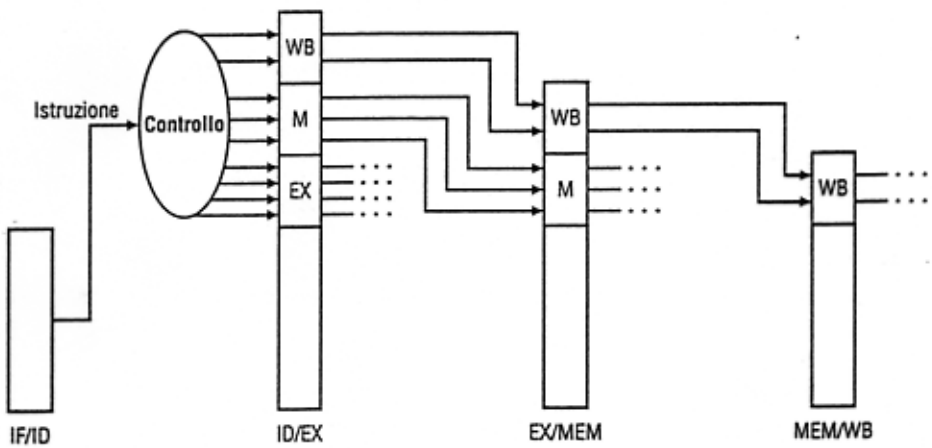
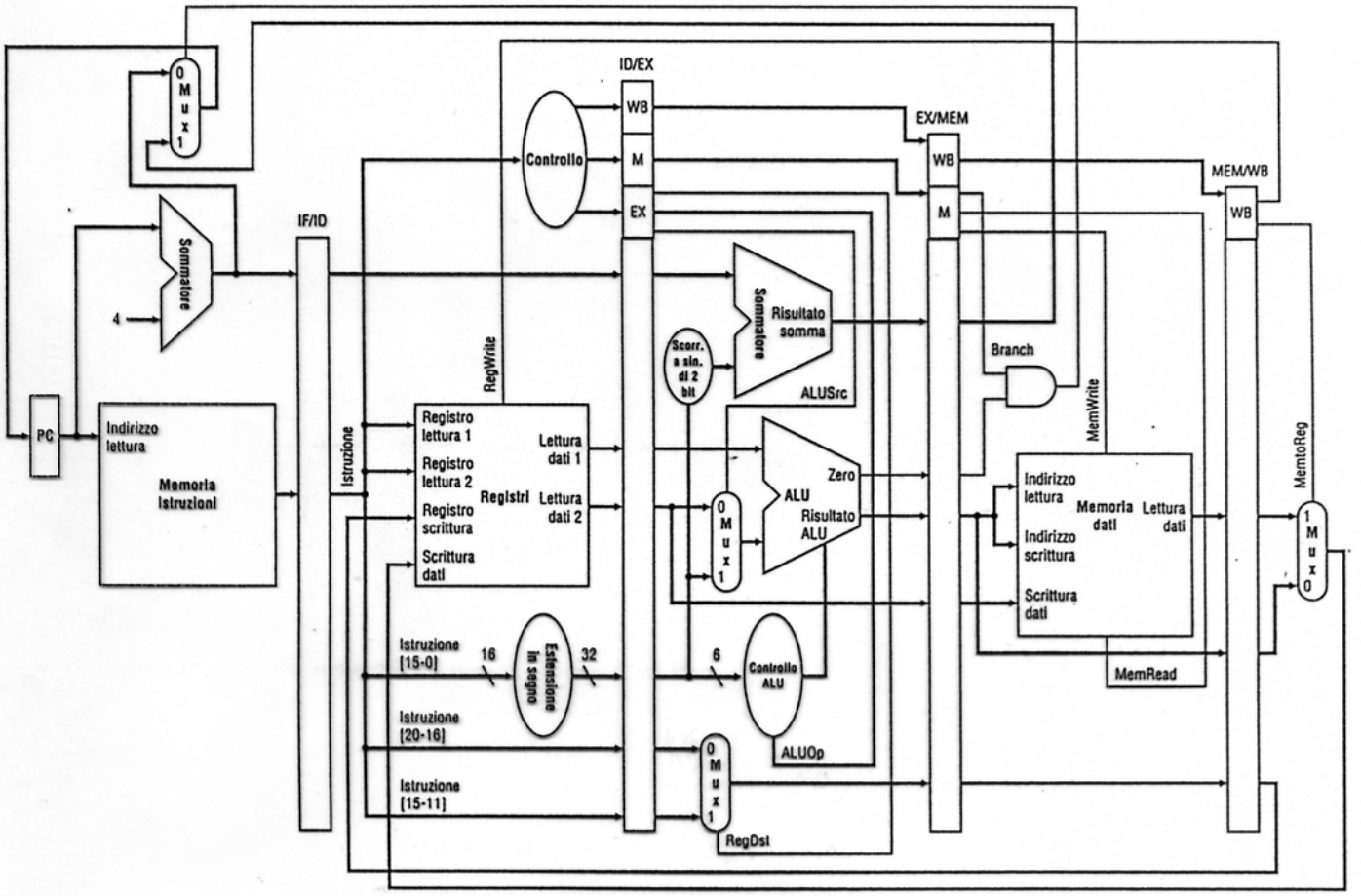



**FIGURA 6.23** Le linee di controllo per gli ultimi tre stadi della pipeline. Si noti che quattro delle nove linee di controllo sono utilizzate nella fase EX e le restanti cinque linee di controllo passate al registro di pipeline EX/MEM ampliato per contenere anche le linee di controllo; tre linee sono utilizzate durante lo stadio MEM; le ultime due sono passate al registro MEM/WB per essere utilizzate per lo stadio WB.

caso particolare di "Mealy" ritardata

# SCO - SCA


0
M
u
x
1
Sommatore
4
IF/ID
ID/EX
EX/MEM
MEM/WB
Controllo
WB
M
EX
PC
Indirizzo lettura
Memoria Istruzioni
Istruzione
RegWrite
Registro lettura 1
Registro lettura 2
Registro scrittura
Scrittura dati
Registri
Lettura dati 1
Lettura dati 2
Scorr. a sin. di 2 bit
Sommatore
Risultato somma
ALUSrc
Branch
MemWrite
MemtoReg
Zero
ALU
Risultato ALU
Indirizzo lettura
Indirizzo scrittura
Scrittura dati
Memoria dati
Lettura dati
MemRead
Istruzione [15-0]
16
Estensione in segno
32
6
Controllo ALU
ALUOp
Istruzione [20-16]
Istruzione [15-11]
RegDst

lw $10, 9($1)
sub $11, $2, $3
and $12, $4, $5
or $13, $6, $7
add $14, $8, $9

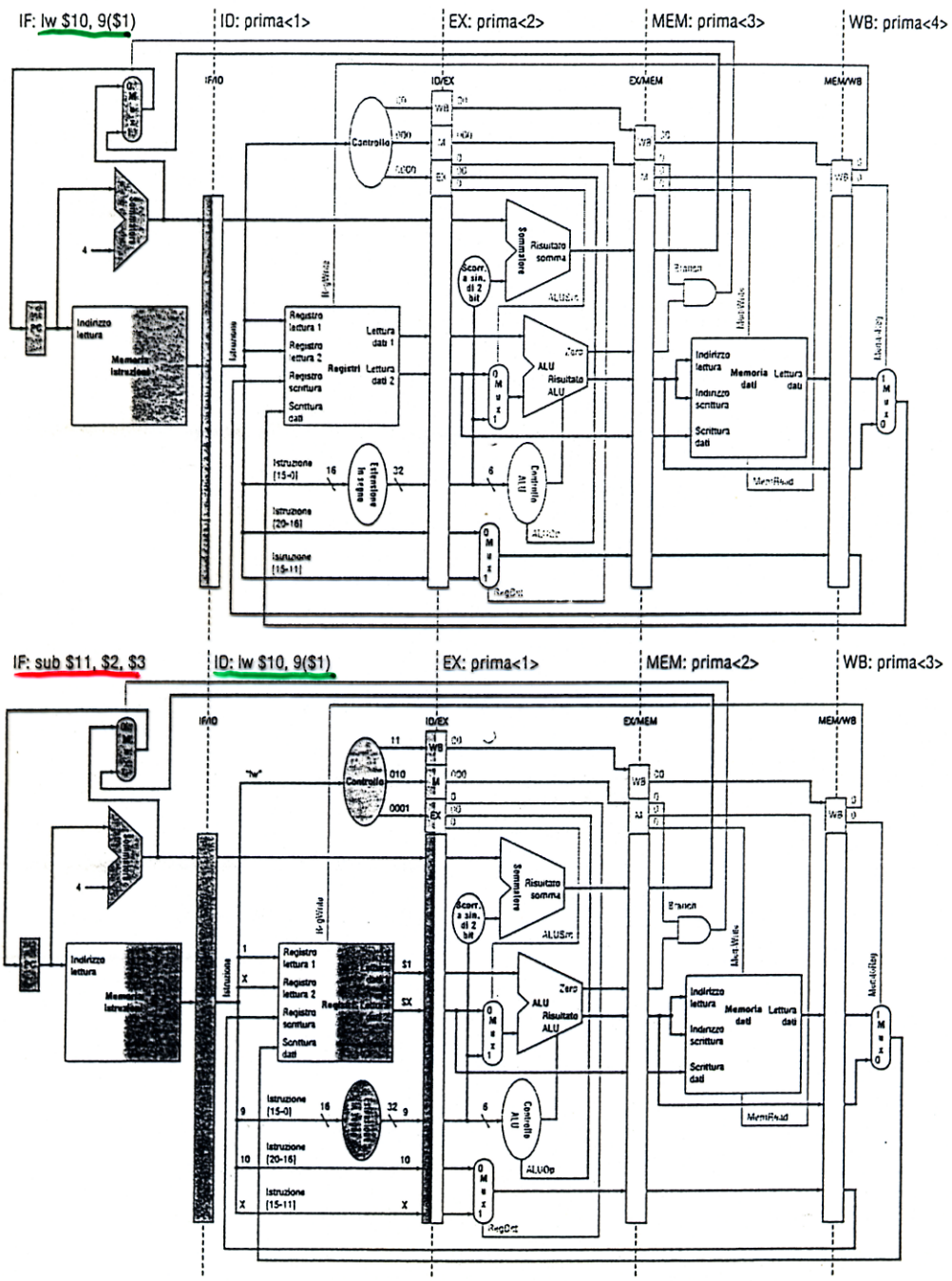



FIGURA 6.25 Cicli di clock 1 e 2. L'unità di calcolo superiore mostra quali unità sono attive durante il primo ciclo di clock, quella inferiore mostra invece quelle attive durante il secondo ciclo. L'espressione «prima<*i*>» indica l'*i*-esima istruzione prima dell'istruzione `lw`. L'istruzione `lw` nell'unità di calcolo superiore è nello stadio IF. Alla fine del ciclo di clock l'istruzione `lw` è nel registro di pipeline IF/ID. Nel secondo ciclo di clock, descritto nell'unità di calcolo inferiore, l'istruzione `lw` entra nello stadio ID e l'istruzione `sub` entra nello stadio IF. Si noti che i valori dei campi dell'istruzione e i registri selezionati vengono mostrati nello stadio ID. Di conseguenza il registro `$1` e la costante 9, gli operandi della `lw`, vengono scritti nel registro di pipeline ID/EX. «X» significa che tale campo della `lw` non è utilizzato. Il numero 10, che rappresenta il numero del registro di destinazione della `lw`, viene anch'esso scritto nel registro ID/EX. La parte superiore dei registri di pipeline mostra i valori di controllo per l'istruzione `lw` da utilizzarsi nei rimanenti stadi della pipeline.

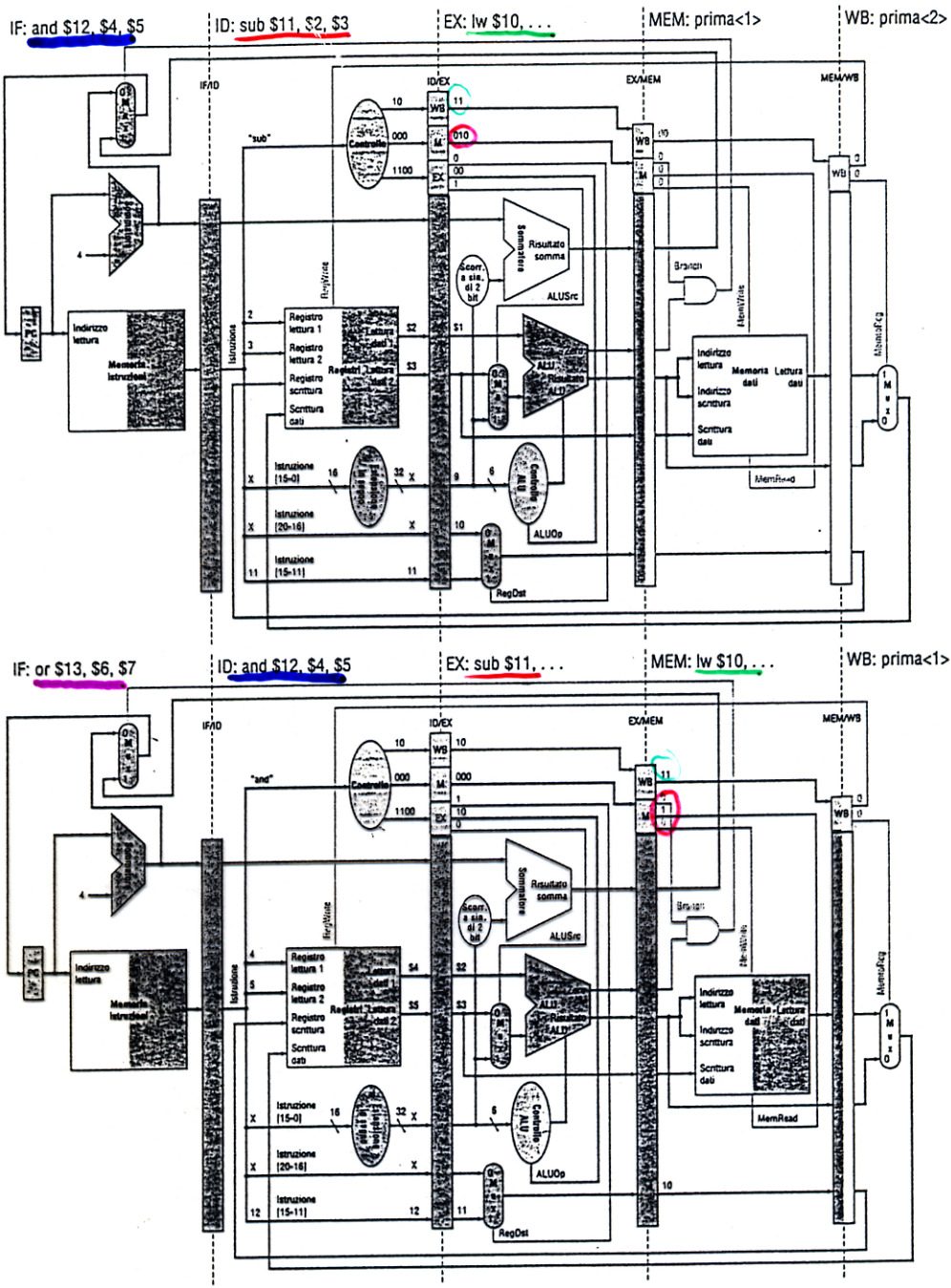



FIGURA 6.26 Cicli di clock 3 e 4. Nel diagramma superiore, `lw` entra nello stadio EX nel terzo ciclo di clock, sommando `$1` e 9 per creare l'indirizzo effettivo nel registro di pipeline EX/MEM. Contemporaneamente, `sub` entra nello stadio ID, leggendo i registri `$2` e `$3`, mentre l'istruzione `and` inizia la fase IF. Nel quarto ciclo di clock (unità di calcolo inferiore), `lw` procede con lo stadio MEM, leggendo la memoria utilizzando il valore in EX/MEM come indirizzo effettivo. Nello stesso ciclo di clock, l'ALU sottrae `$3` da `$2`, pone la differenza in EX/MEM, `and` legge i registri `$4` e `$5` durante lo stadio ID, e l'istruzione `or` entra nello stadio IF. I due diagrammi mostrano i segnali di controllo creati nello stadio ID ed eliminati man mano che vengono utilizzati negli stadi successivi della pipeline.

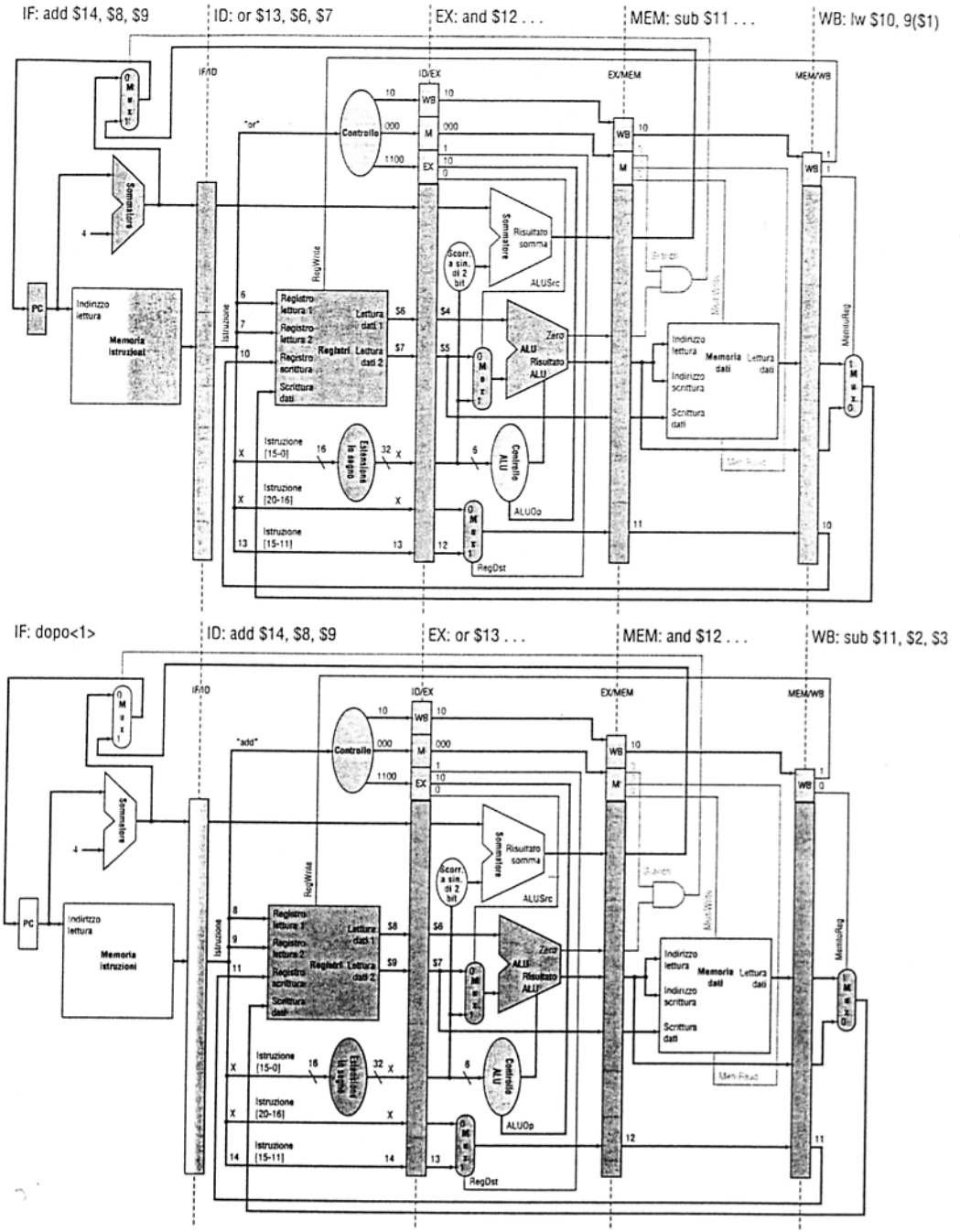



**FIGURA 6.27** Cicli di clock 5 e 6. Con l'istruzione `add` che entra nello stadio IF, nell'unità di calcolo superiore, tutte le istruzioni sono in esecuzione. L'istruzione finale in questo esempio, «dopo <*i*>» indica l'*i*-esima istruzione dopo l'istruzione `add`. Scrivendo i dati nel registro MEM/WB nel registro 10, `lw` viene completata; sia i dati sia il numero del registro sono in MEM/WB. In seguito `sub` invia la differenza presente in EX/MEM a MEM/WB e il resto delle istruzioni procedono. Nel ciclo di clock successivo, `sub` seleziona il valore di MEM/WB da scriversi nel registro numero 11, trovato anch'esso in MEM/WB. Le istruzioni rimanenti seguono la stessa linea: l'ALU effettua l'OR tra `$6` e `$7` per l'istruzione `or` nello stadio EX, e i registri `$8` e `$9` vengono letti nello stadio ID per l'istruzione `add`.

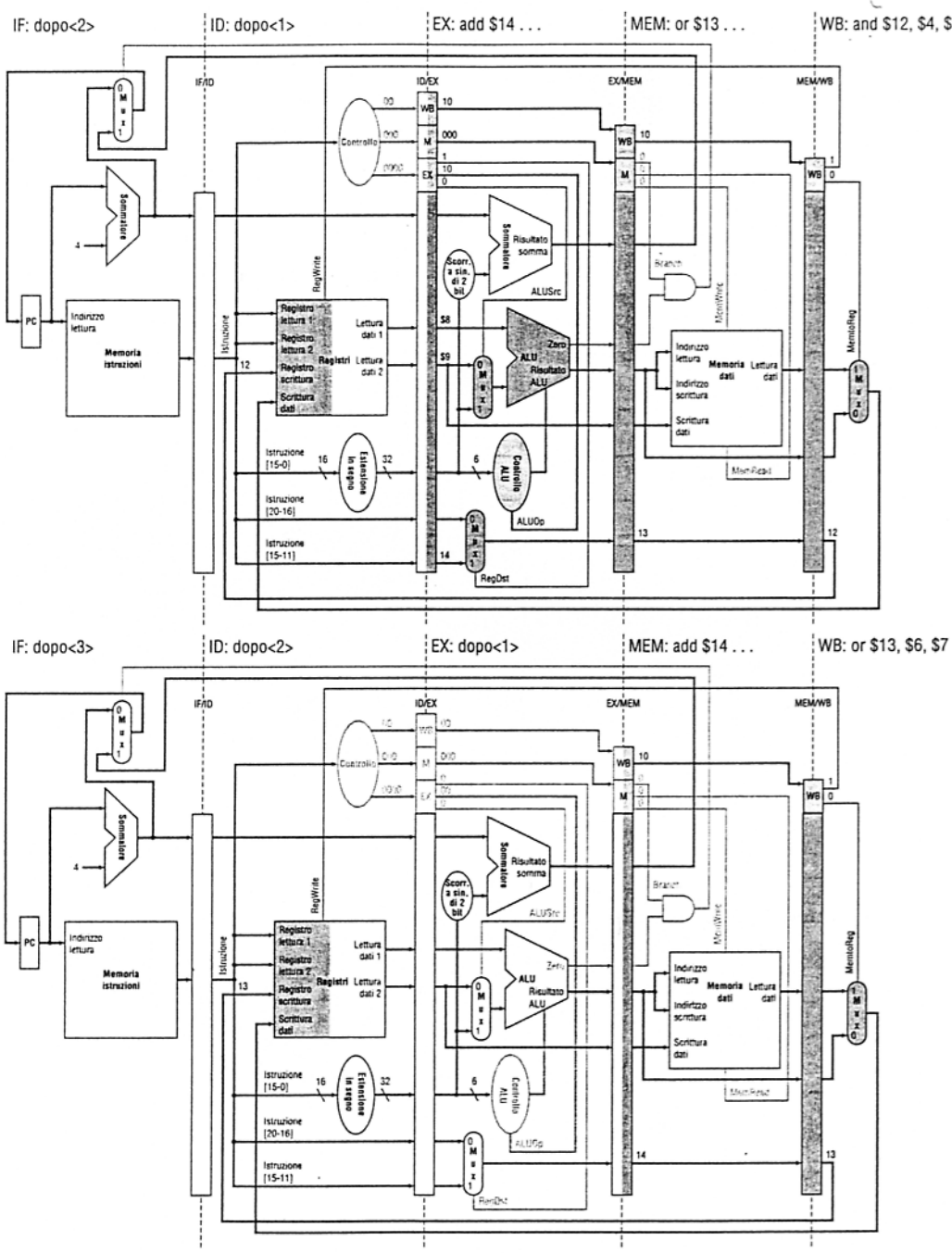



FIGURA 6.28 Cicli di clock 7 e 8. Nell'unità di calcolo superiore, l'istruzione `add` si trova nello stadio EX, sommando i valori corrispondenti ai registri `$8` e `$9`. Il risultato viene poi passato dal registro EX/MEM a MEM/WB nello stadio MEM per l'istruzione `or` e lo stadio WB scrive i risultati di MEM/WB nel registro `$12` per completare l'istruzione `and`. Si noti che i segnali di controllo sono non asseriti (ovvero posti a 0) nello stadio ID, poiché nessuna istruzione è in esecuzione. Nel ciclo di clock successivo (disegno in basso), lo stadio WB scrive il risultato nel registro `$13`, completando così l'istruzione `or`, e lo stadio MEM copia il contenuto di EX/MEM in MEM/WB.

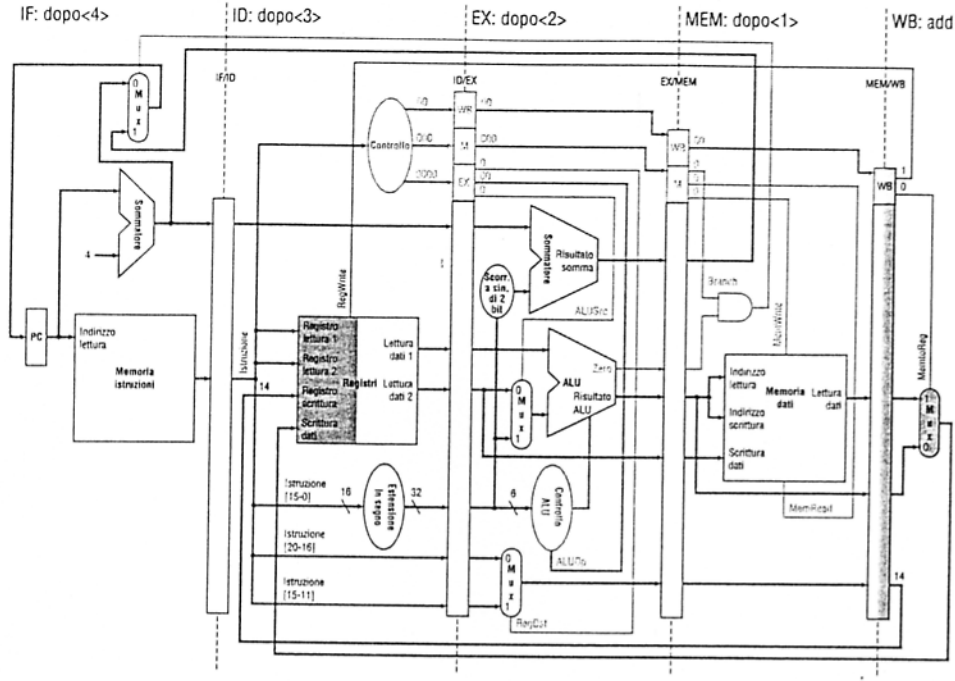



FIGURA 6.29 Ciclo di clock 9. Lo stadio WB scrive la somma presente in MEM/WB nel registro `$14`, concludendo l'istruzione `add` e la sequenza delle cinque istruzioni.