

7' Esercitazione

<https://politecnicomilano.webex.com/meet/gianenrico.conti>

2 maggio 2021

Gian Enrico Conti

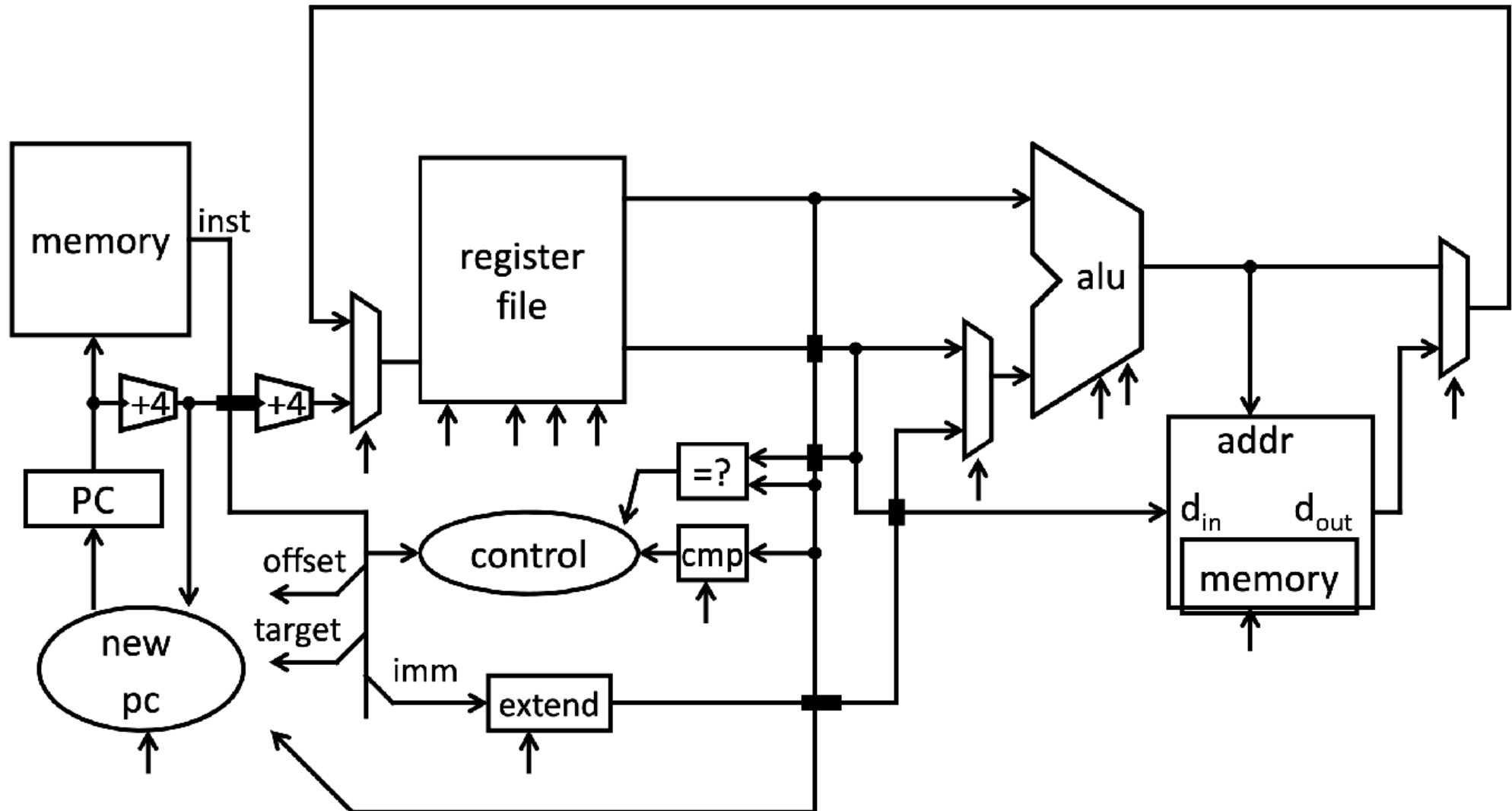
MIPS architecture (pipelined)

Recap

- **Architettura a singolo ciclo**
 - Funzionamento
 - Segnali di controllo

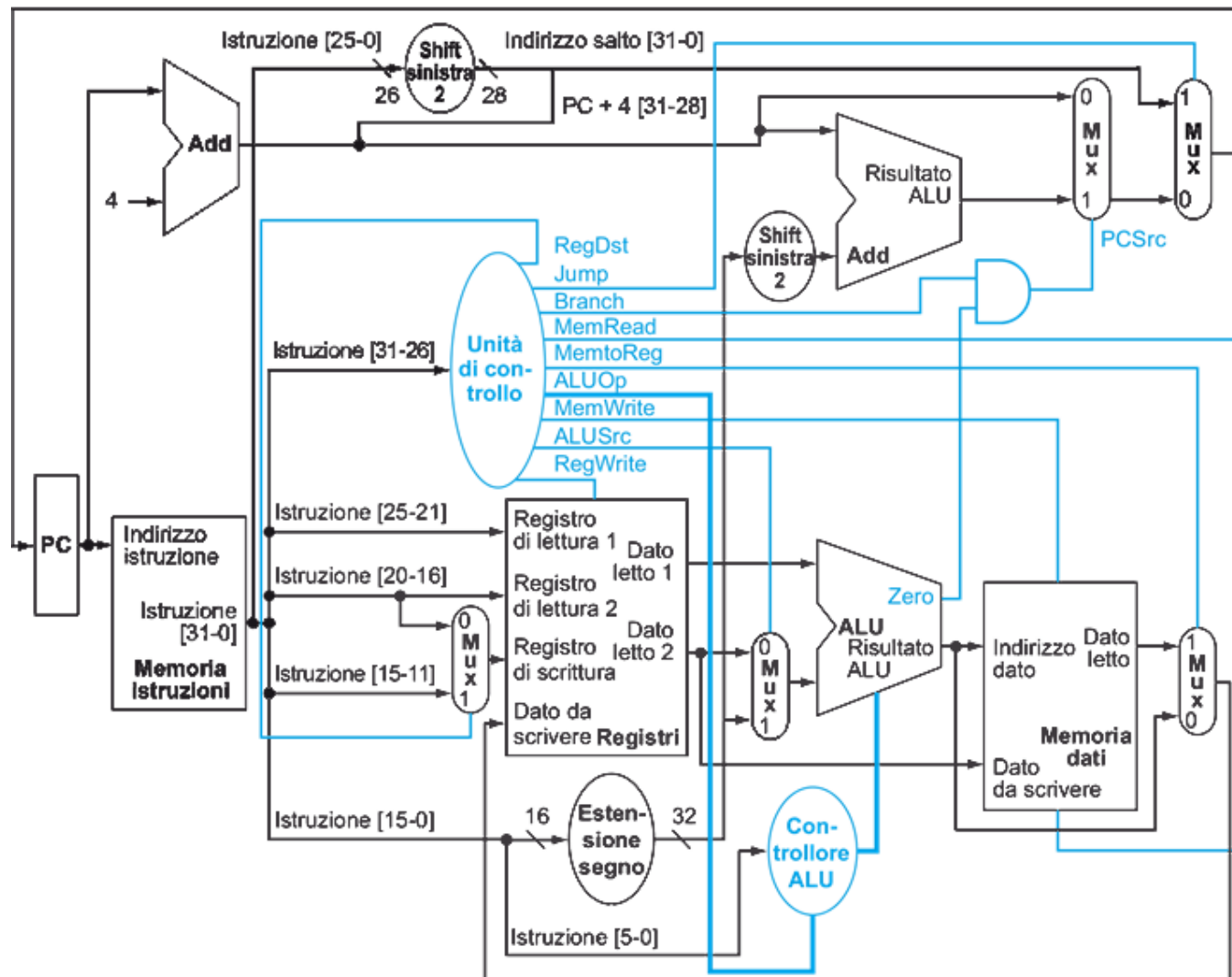
Architettura a singolo ciclo

- Schema semplificato dell'architettura MIPS a singolo ciclo



Architettura a singolo ciclo

- Schema semplificato dell'architettura MIPS a singolo ciclo

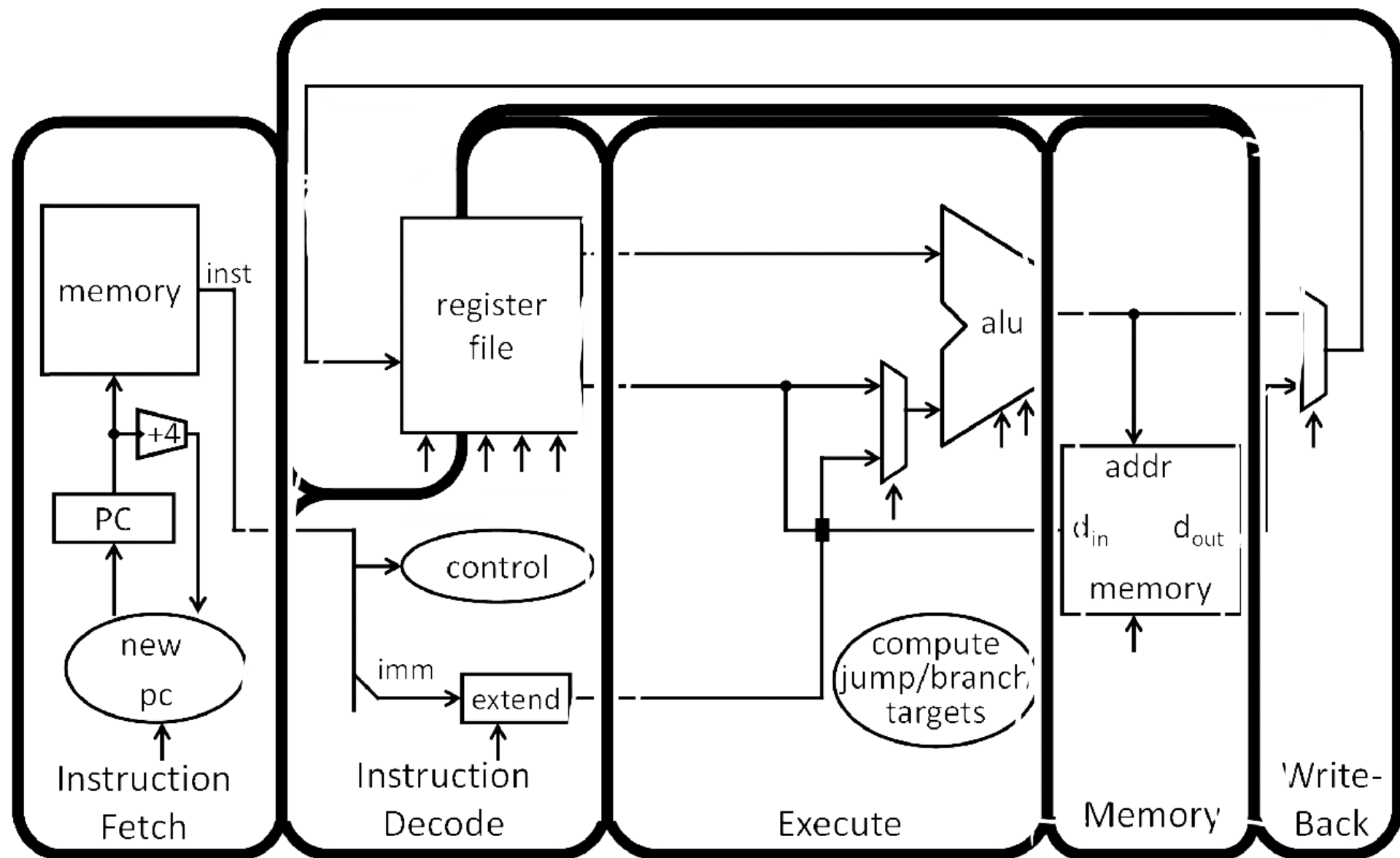


Outline

- **Architettura pipelined**

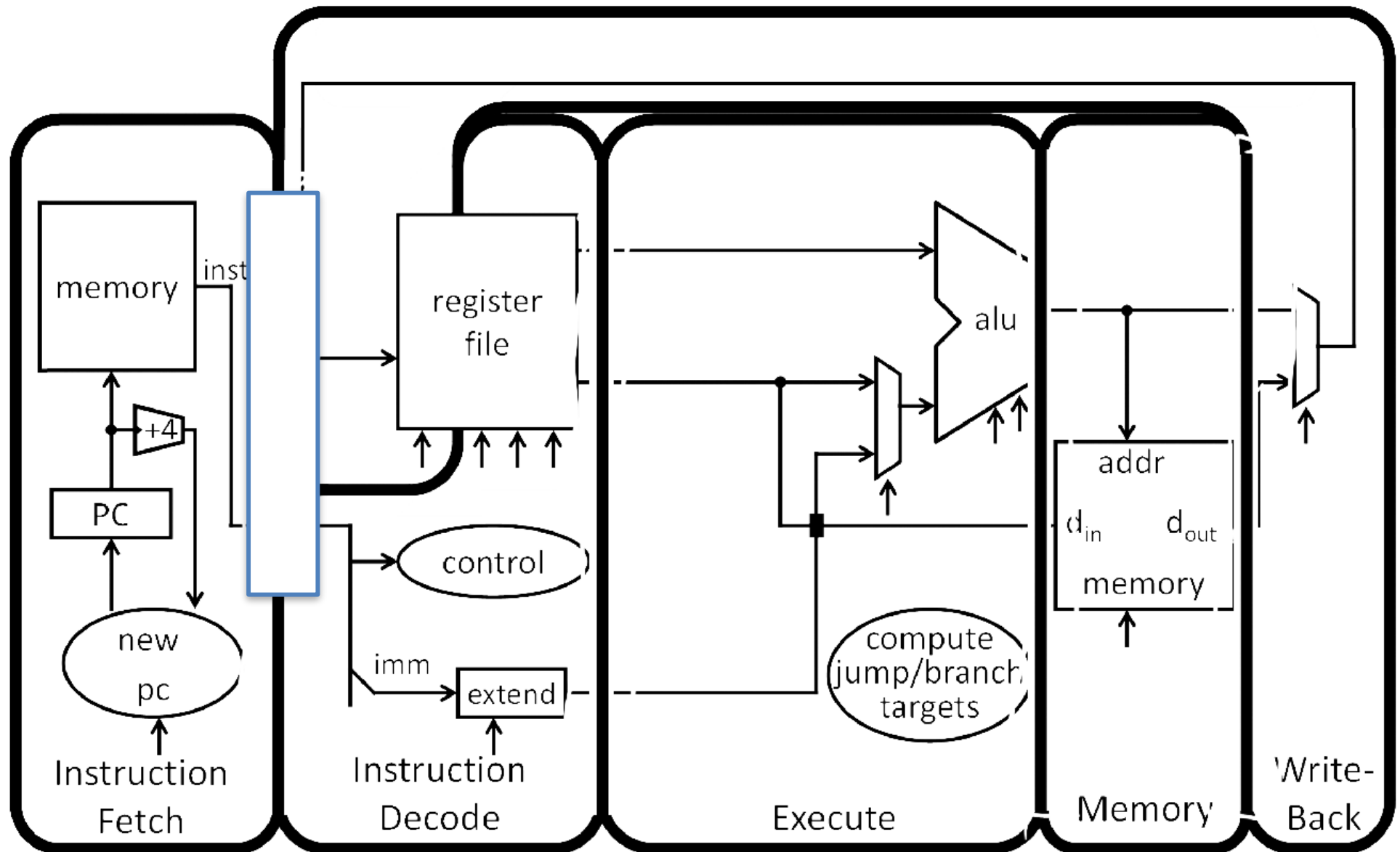
- Funzionamento
- Esecuzione sequenziale vs pipelined
- Conflitti all'interno della pipeline
 - Conflitti strutturali
 - Conflitti sui dati
 - Conflitti di controllo

Architettura pipelined: Zone (single cycle)



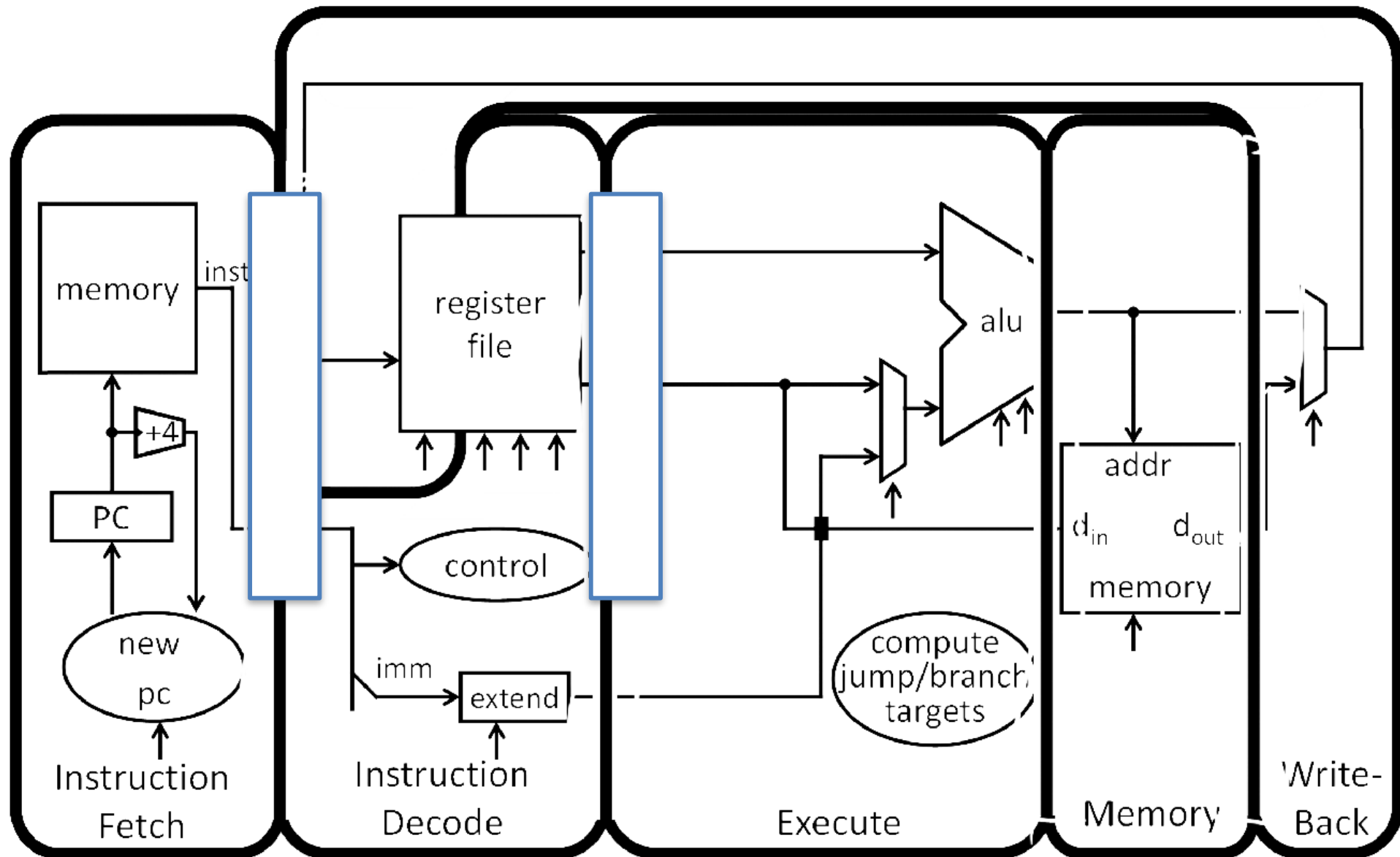
Architettura pipelined:

Aggiungendo dei registri tra le varie fasi si ottiene una architettura pipelined



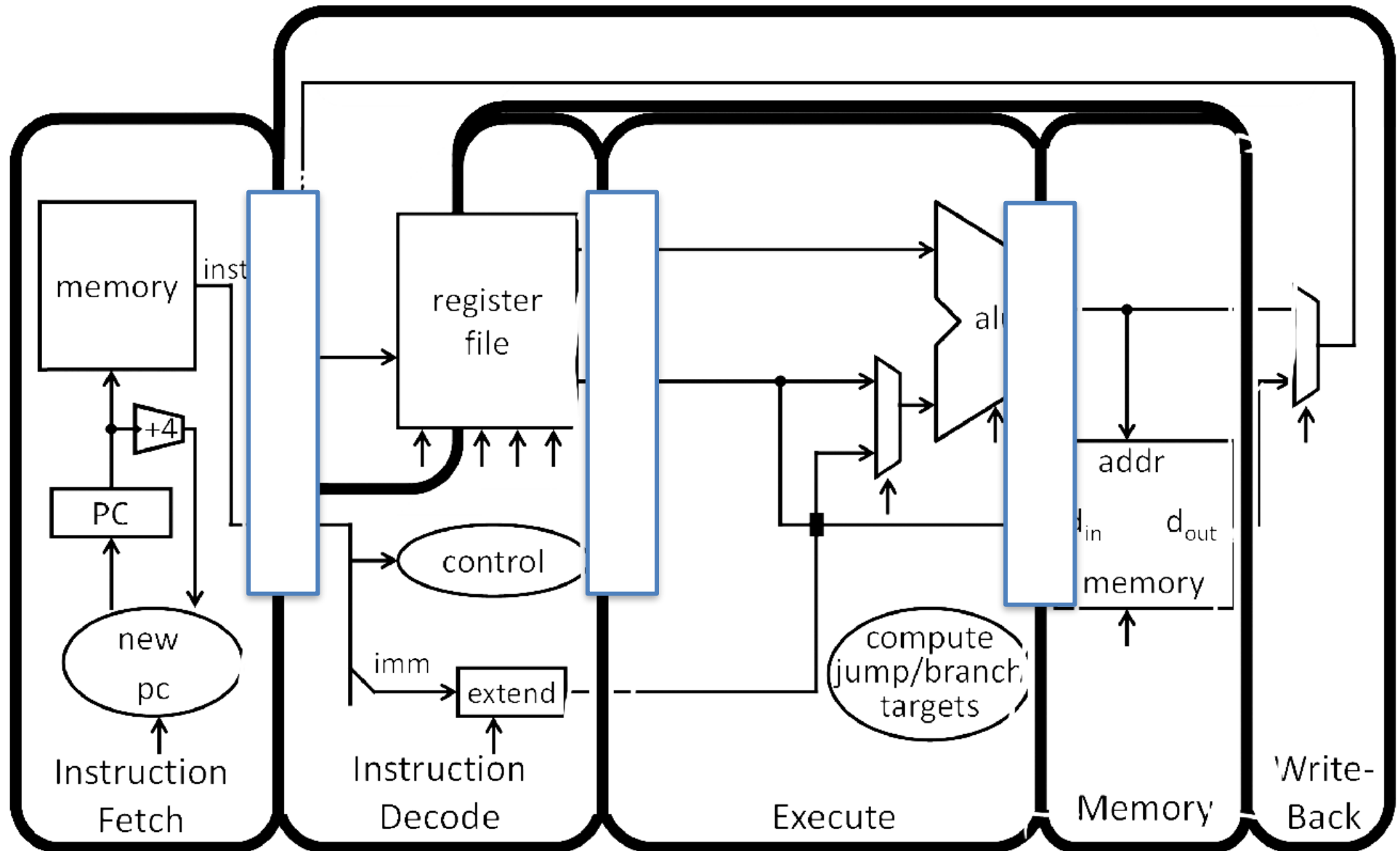
Architettura pipelined:

Aggiungendo dei registri tra le varie fasi si ottiene una architettura pipelined



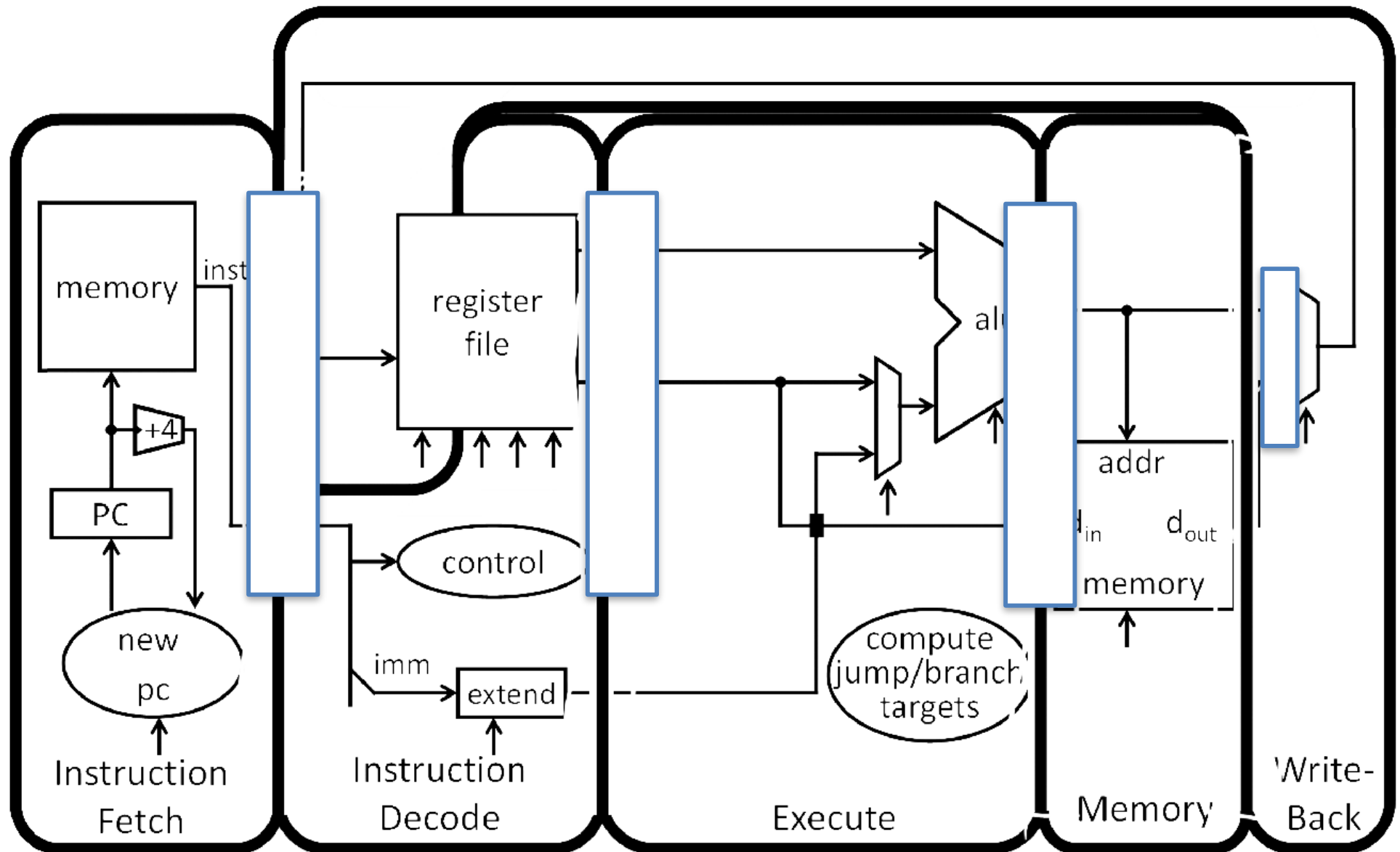
Architettura pipelined:

Aggiungendo dei registri tra le varie fasi si ottiene una architettura pipelined



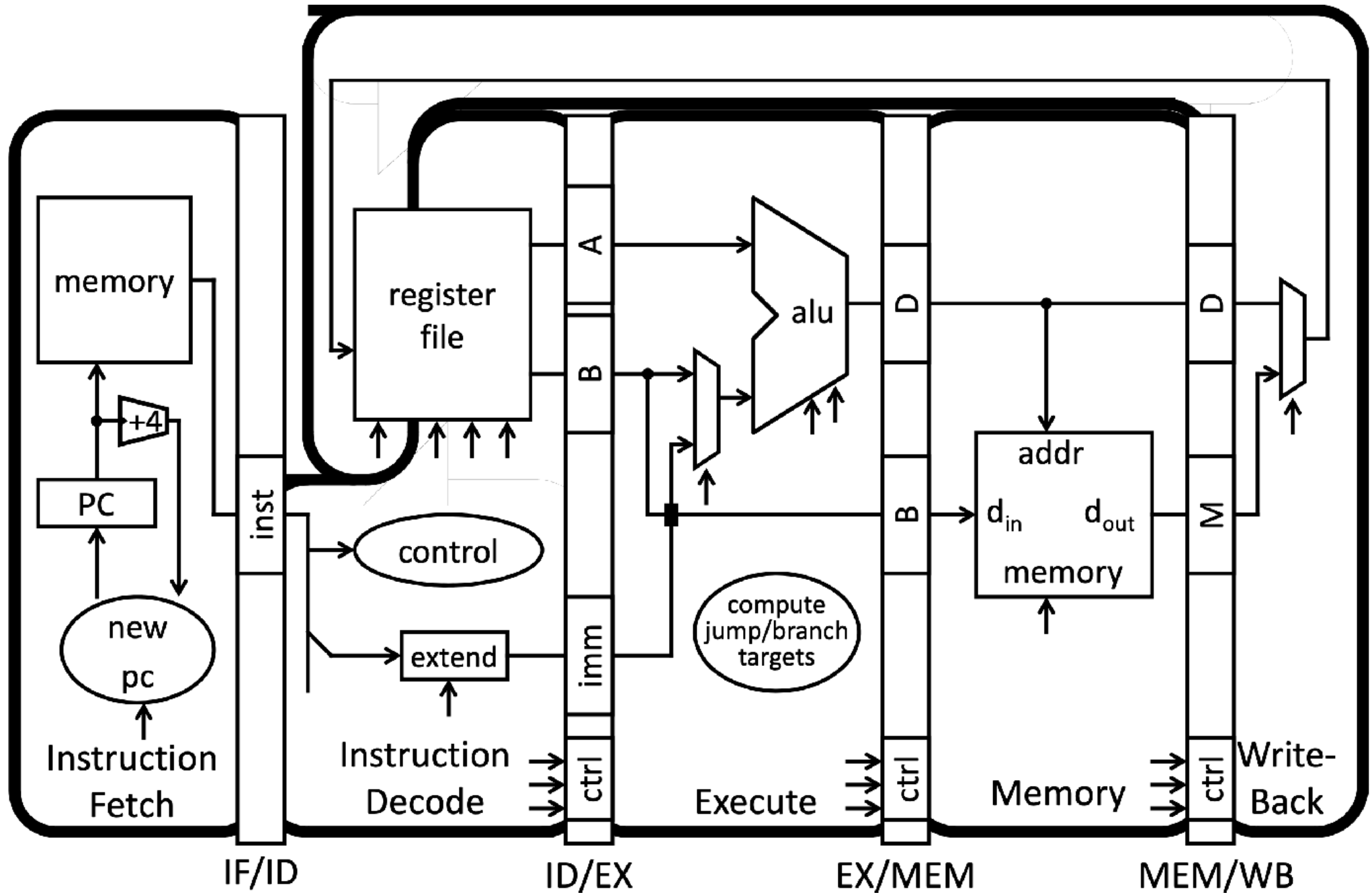
Architettura pipelined:

Aggiungendo dei registri tra le varie fasi si ottiene una architettura pipelined



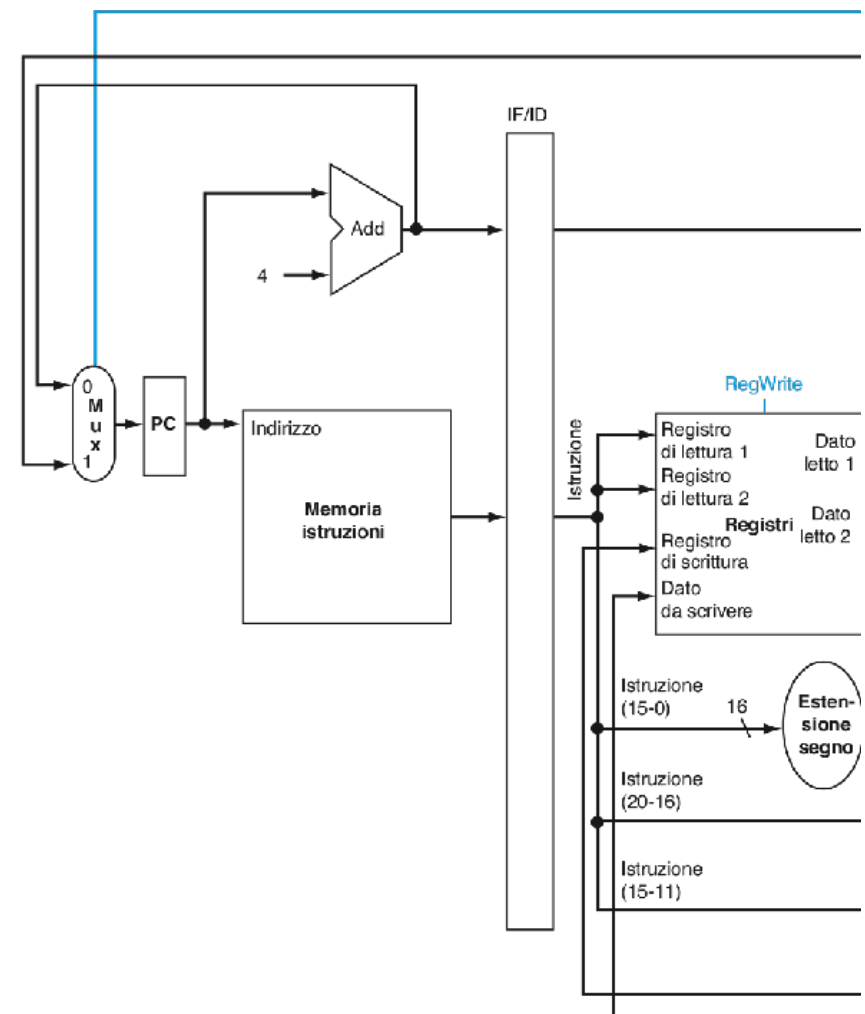
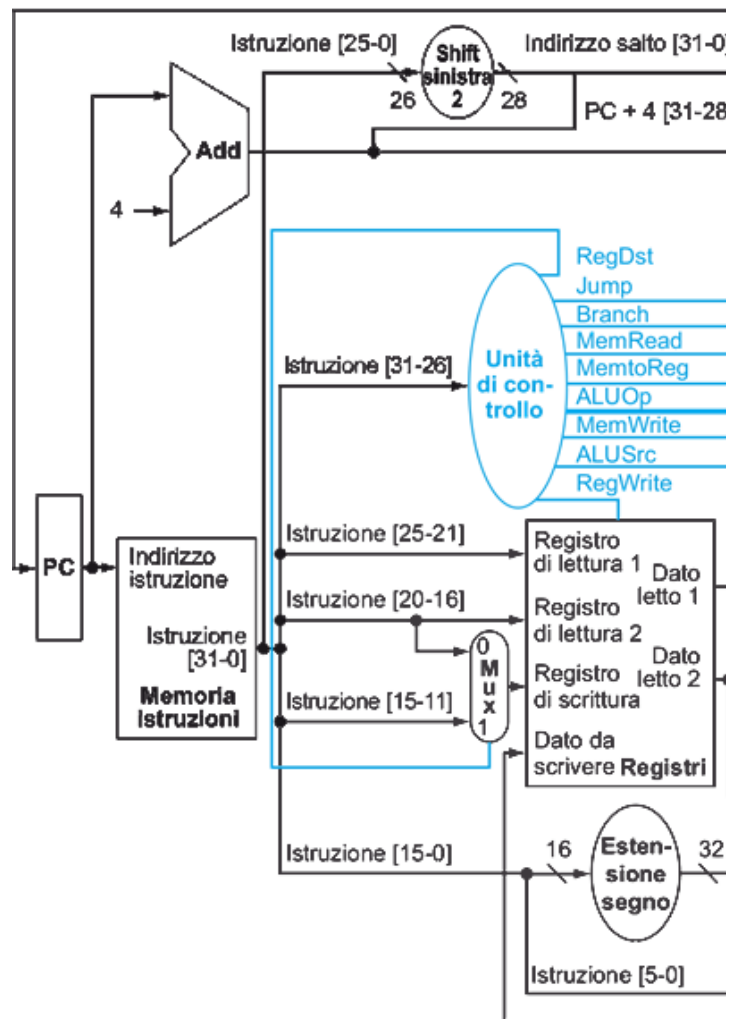
Architettura pipelined:

Aggiungendo dei registri tra le varie fasi si ottiene una architettura pipelined



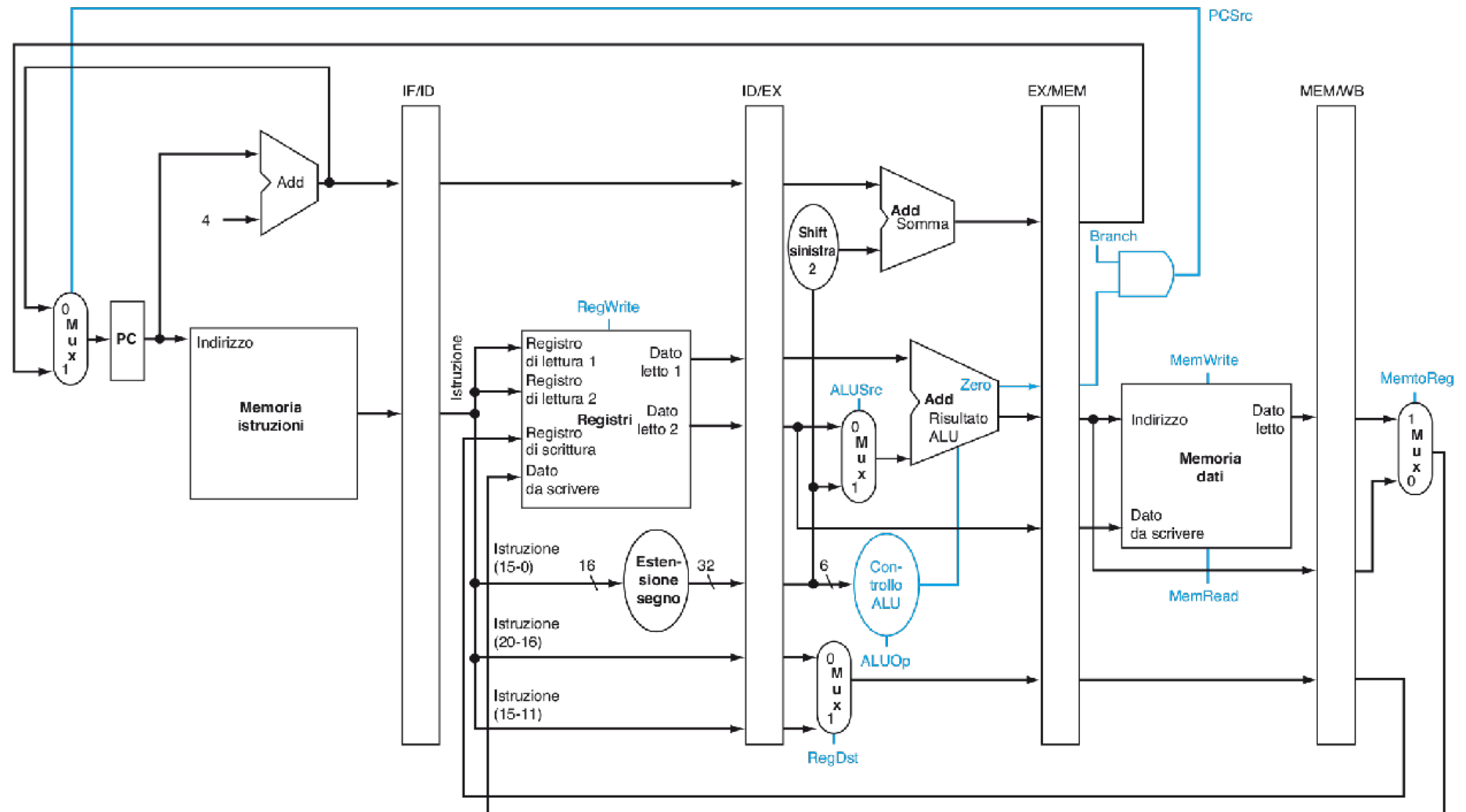
Architettura Pipelined

- Aggiungendo dei registri tra le varie fasi si ottiene una architettura pipelined (ns schema)



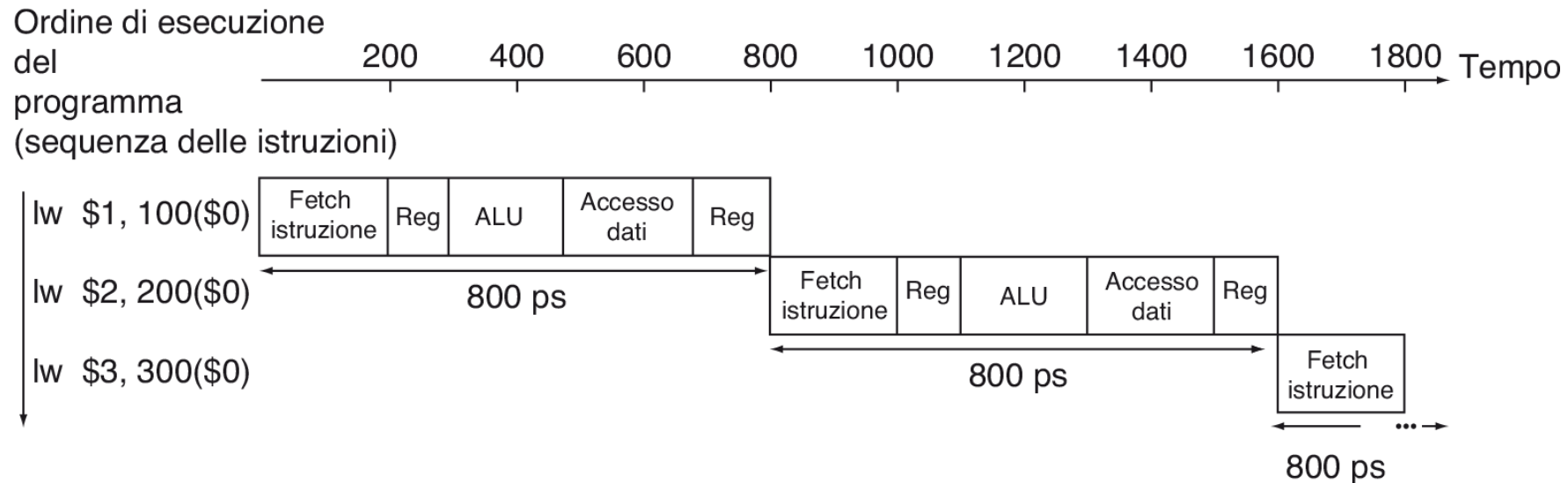
Architettura Pipelined

- Aggiungendo dei registri tra le varie fasi si ottiene una architettura pipelined in piu punti:



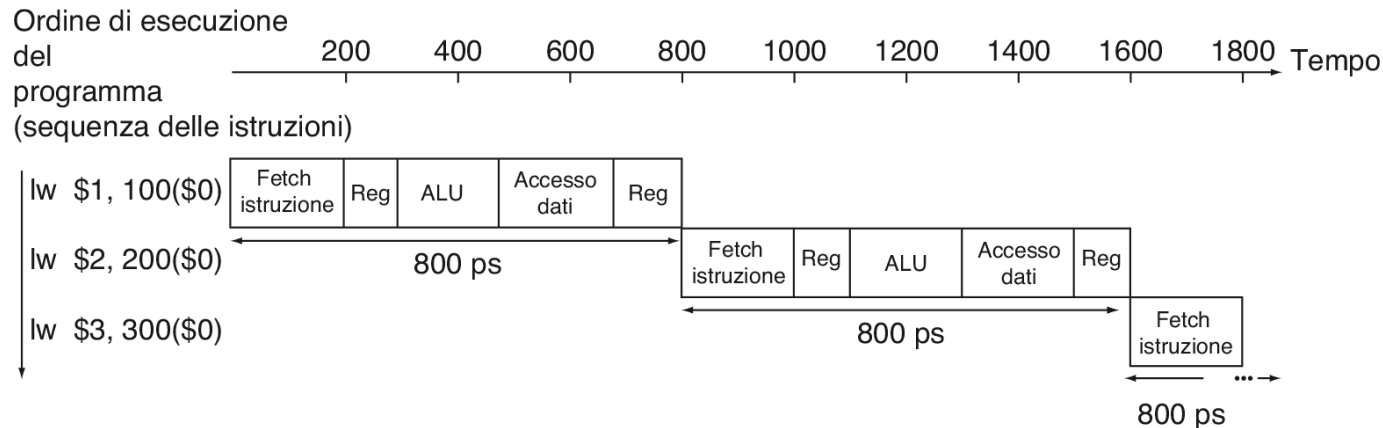
Esecuzione sequenziale vs pipelined

- Nell'esecuzione sequenziale ogni istruzione inizia la propria esecuzione solo al termine della precedente

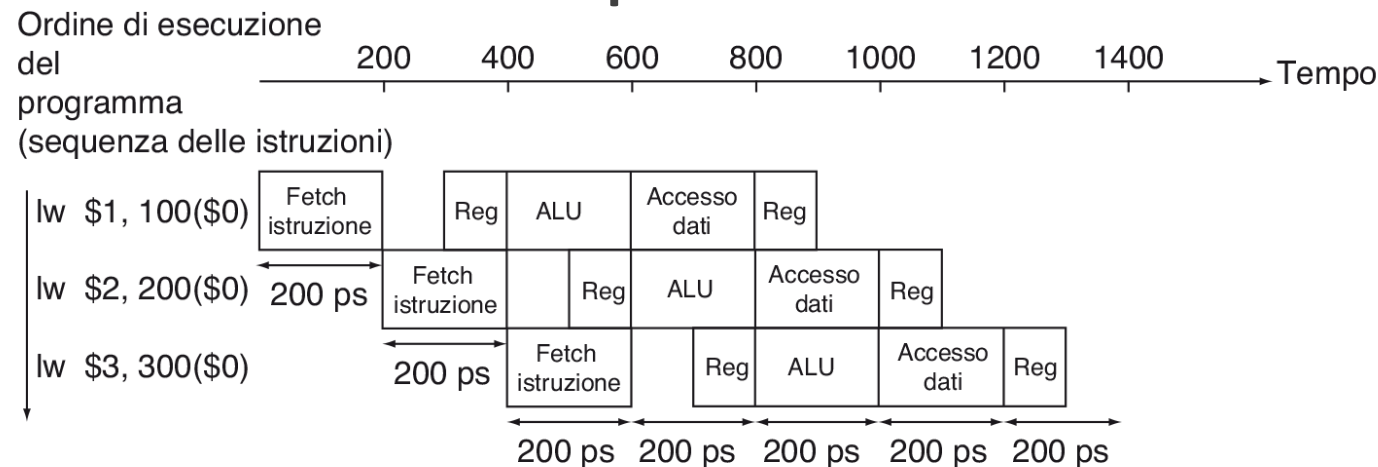


Esecuzione sequenziale vs pipelined

- Nell'esecuzione sequenziale ogni istruzione inizia la propria esecuzione solo al termine della precedente



- Nell'esecuzione pipelined si migliorano le prestazioni basata sulla sovrapposizione dell'esecuzione di più istruzioni appartenenti ad un flusso di esecuzione sequenziale.



CPI

- Per misurare l'efficienza di una certa architettura si utilizza un indicatore chiamato CPI (Cycles Per Instruction)

- $$\text{CPI} = \frac{\# \text{ TOTALE CICLI}}{\# \text{ TOTALE ISTRUZIONI}}$$

- Intuitivamente il CPI rappresenta il numero medio di cicli necessari per completare un istruzione

- **Attenzione!**

- $\text{CPI} \geq 1$ al massimo termina un istruzione per ciclo se non ci sono stalli

Terminologia

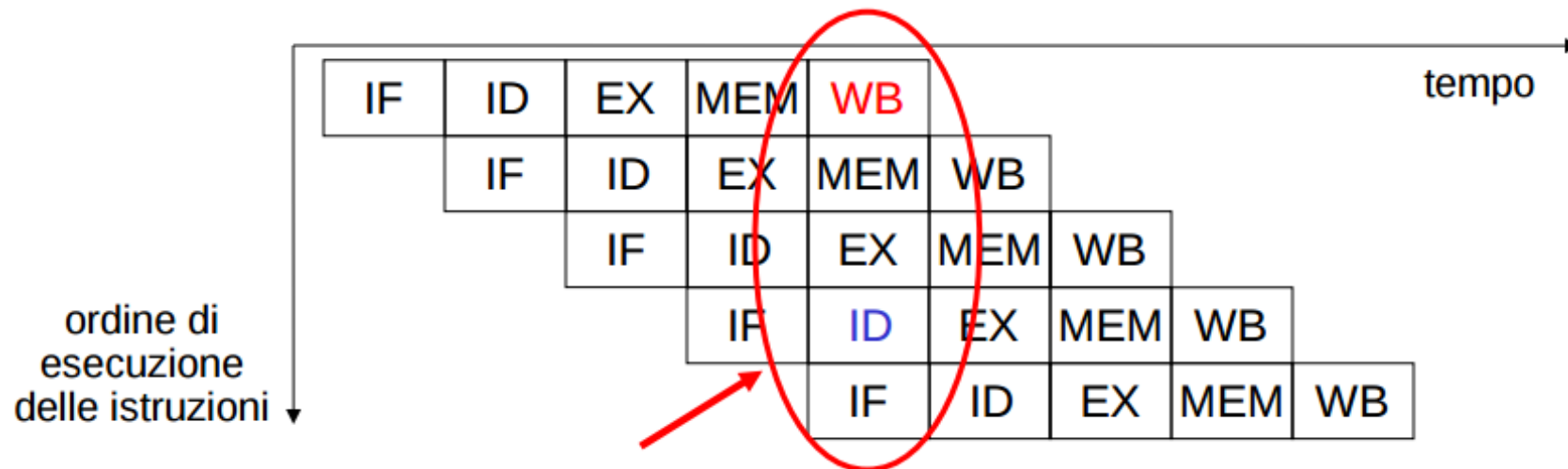
- Five stage “RISC” load-store architecture:
 - 1. Instruction fetch (IF)
 - get instruction from memory, increment PC
 - 2. Instruction Decode (ID)
 - translate opcode into control signals and read registers
 - 3. Execute (EX)
 - perform ALU operation, compute jump/branch targets
 - 4. Memory (MEM)
 - access memory if needed
 - 5. Writeback (WB)
 - update register file
 -

Conflitti all'interno della pipeline

- I conflitti sorgono nelle architetture con pipelining quando non è possibile eseguire un'istruzione nel ciclo immediatamente successivo
 - Conflitti strutturali
 - Tentativo di usare la stessa risorsa hardware da parte di diverse istruzioni in modi diversi nello stesso ciclo di clock
 - Conflitti sui dati
 - Tentativo di usare un risultato prima che sia disponibile
 - Conflitti di controllo
 - Nel caso di salti, decidere quale prossima istruzione da eseguire prima che la condizione sia valutata

Conflitto strutturali

- Nell'architettura MIPS pipeline non abbiamo conflitti strutturali
 - Memoria dati separata dalla memoria istruzioni
 - Banco dei registri progettato per evitare conflitti tra la lettura e la scrittura nello stesso ciclo
 - **Scrittura** del banco dei registri nella **prima metà** del ciclo di clock
 - **Lettura** del banco dei registri nella **seconda metà** del ciclo di clock

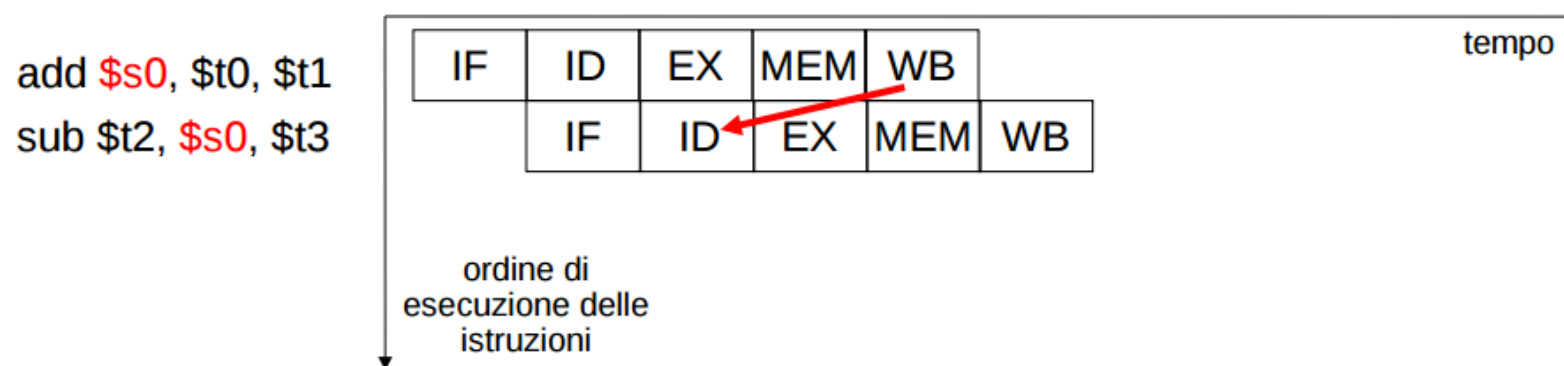


Conflitti sui dati

- Un'istruzione dipende dal risultato di un'istruzione precedente che è ancora nella pipeline

Istruzione
add \$s0, \$t0, \$t1
sub \$t2, \$s0, \$t3

- Nella pipeline queste istruzioni vengono rappresentate come



- Due possibili soluzioni a questo tipo di conflitti sono
 - NOP Inserimento di istruzioni NOP per evitare il conflitto
 - Scheduling Cambiamento dell'ordine di esecuzione delle istruzioni mantenendo equivalenza funzionale

Soluzioni ai conflitti sui dati

■ Soluzioni di tipo hardware

- Inserimento di bolle (**bubble**) o stalli nella pipeline
 - Si inseriscono dei tempi morti
 - Peggiora il throughput
- Propagazione o scavalcamiento (**forwarding** o bypassing)
 - Si propagano i dati in avanti appena sono disponibili verso le unità che li richiedono

■ Soluzioni di tipo software

- Inserimento di istruzioni **nop** (no operation)
 - Peggiora il throughput
- Riordino delle istruzioni
 - Spostare istruzioni “innocue” in modo che esse eliminino la criticità

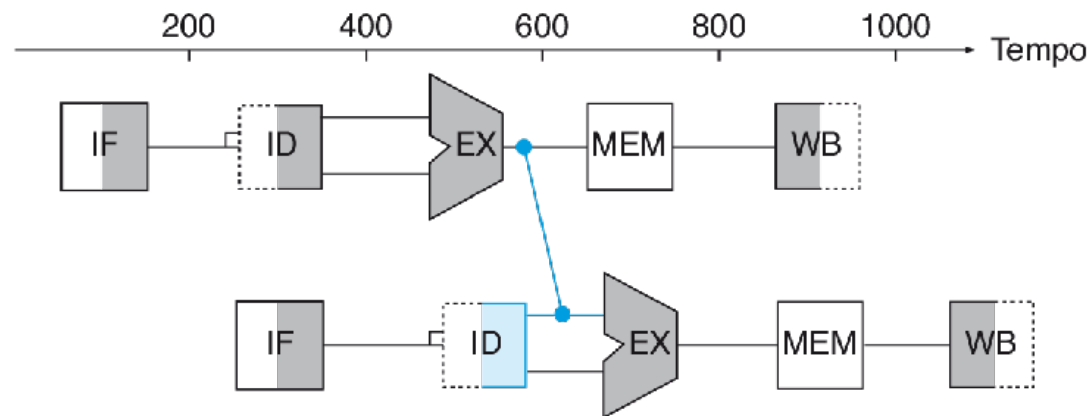
Propagazione

■ Propagazione di una istruzione aritmetica

Ordine di esecuzione
del programma
(sequenza
delle istruzioni)

add \$s0, \$t0, \$t1

sub \$t2, \$s0, \$t3

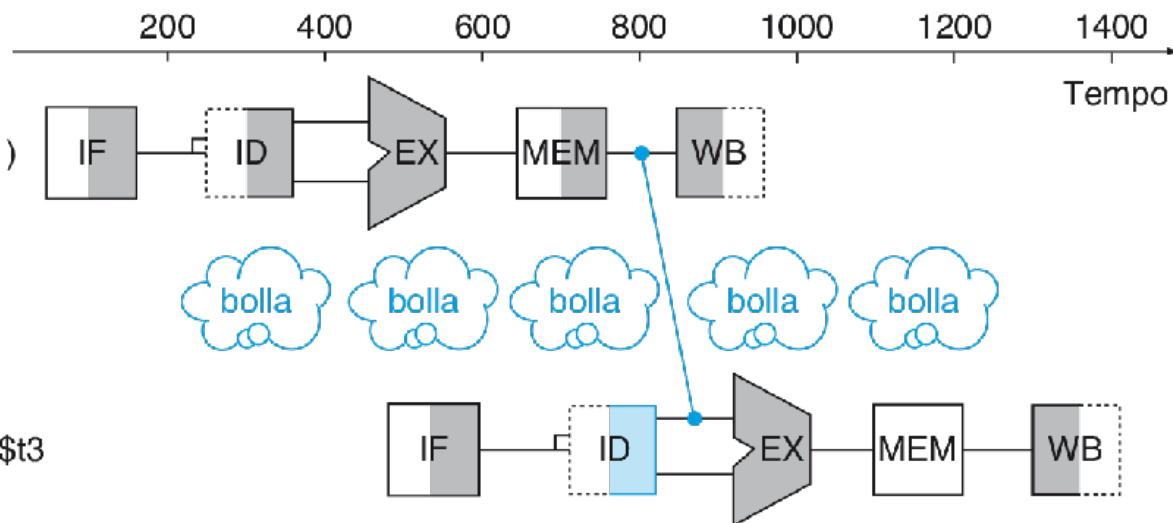


■ Propagazione di un'istruzione LW

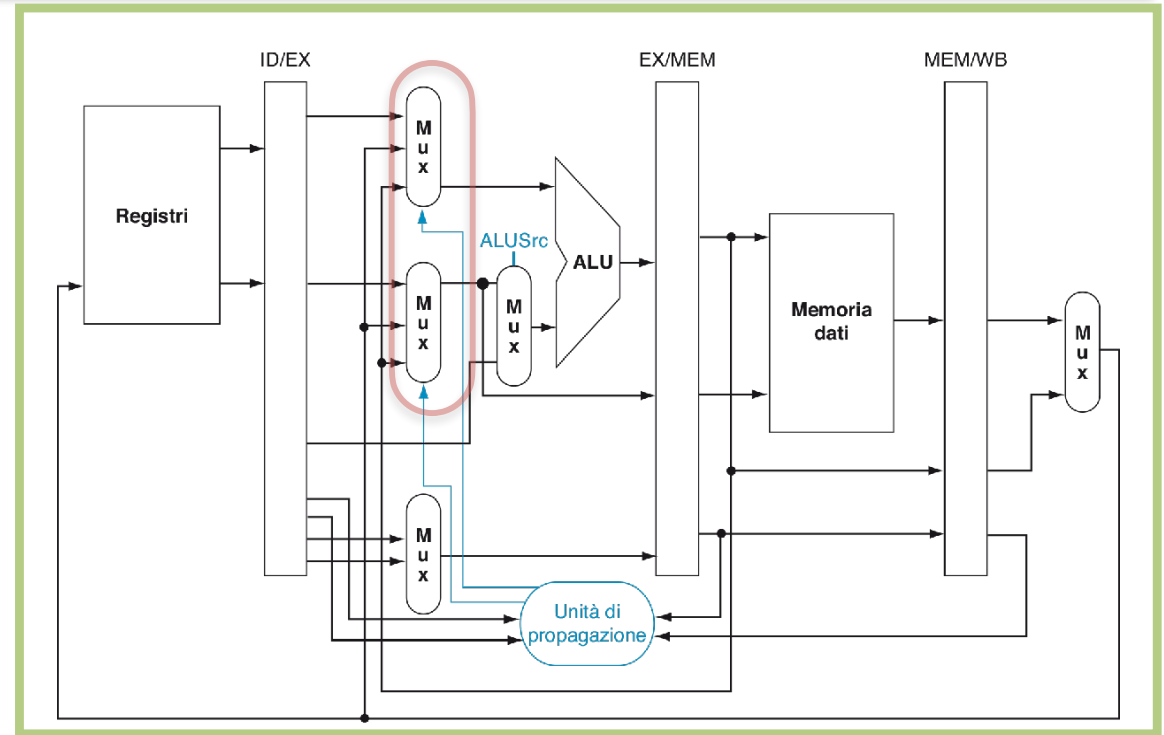
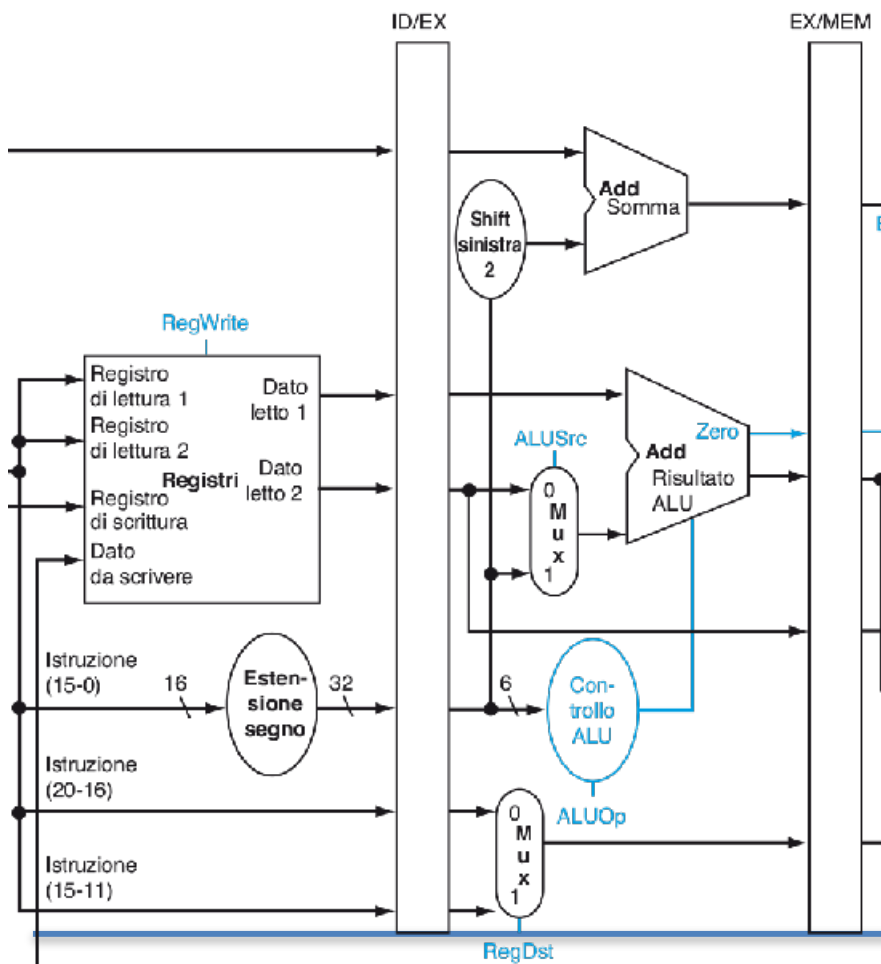
Ordine di esecuzione
del programma
(sequenza
delle istruzioni)

lw \$s0, 20(\$t1)

sub \$t2, \$s0, \$t3

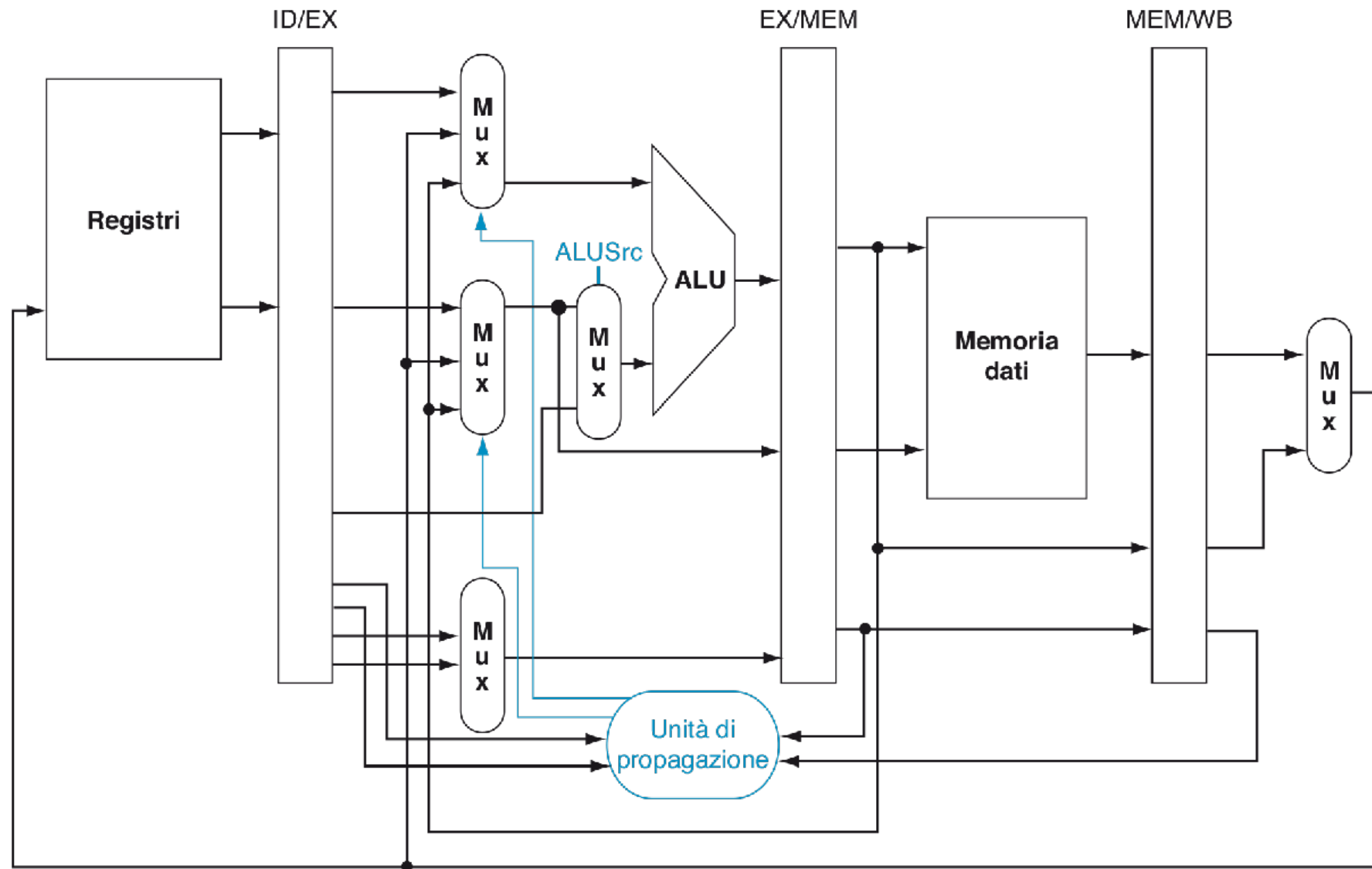


Architettura con forwarding



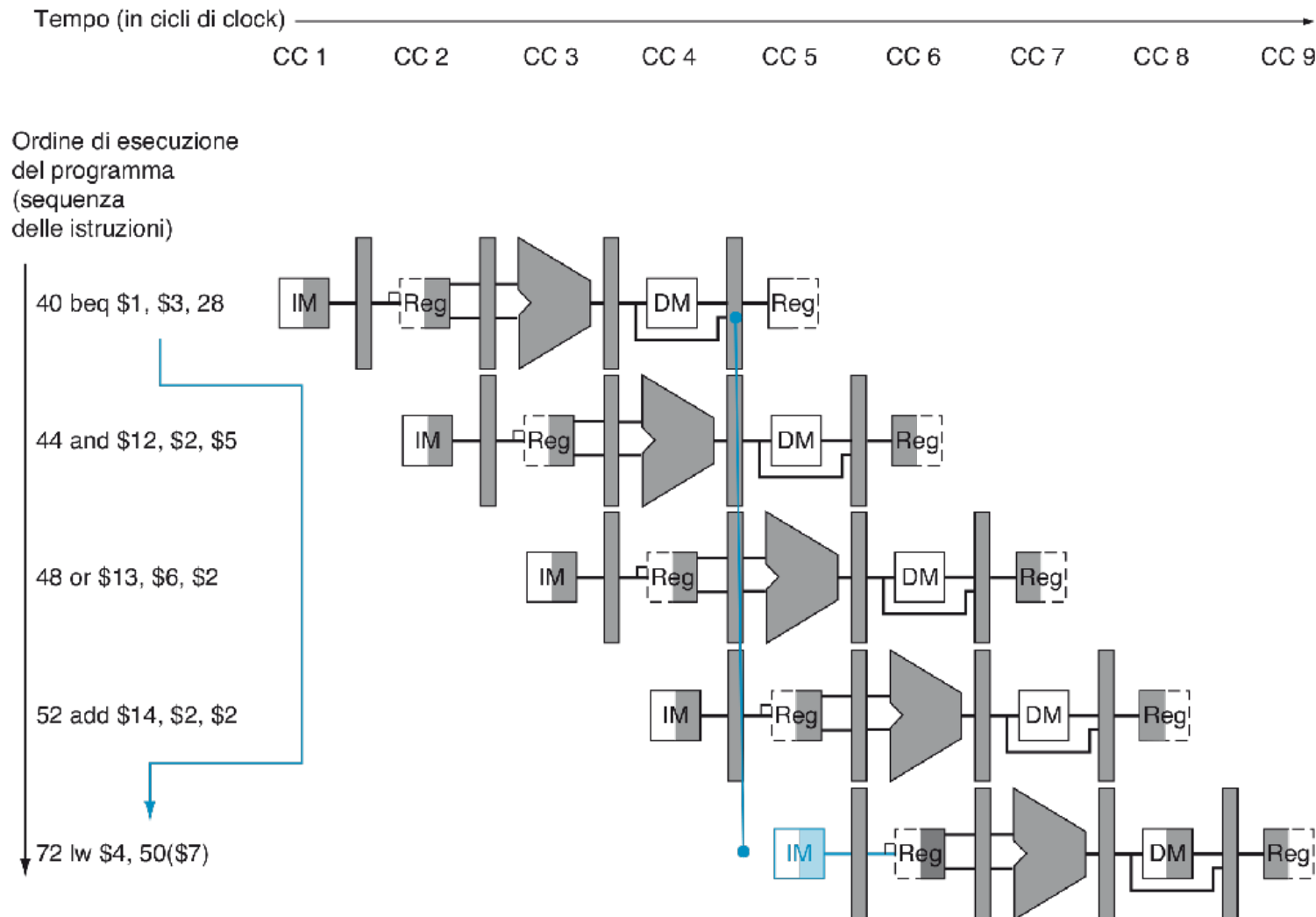
Forwarding: Si notino i MUX

Architettura con forwarding: U.di Prop. pilota i MUX



Conflitti condizionati

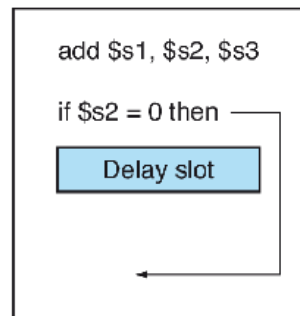
- Nel processore MIPS la decisione sul salto condizionato non viene presa fino al quarto passo (MEM) dell'istruzione beq



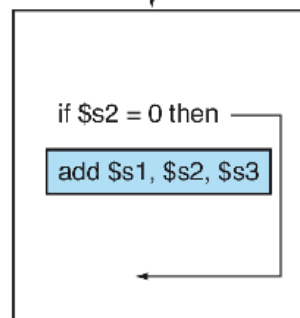
Conflitti condizionati

- Per avere un'esecuzione corretta da parte del processore si possono attuare due soluzioni:
 - NOP Inserimento di istruzioni NOP finché non conosco il risultato della branch
 - Delay Slot Cambio l'ordine di esecuzione delle istruzioni in modo da mantenere equivalenza funzionale e non inserire le NOP

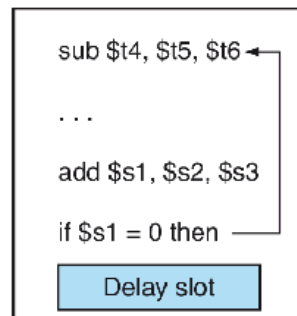
a. Da prima



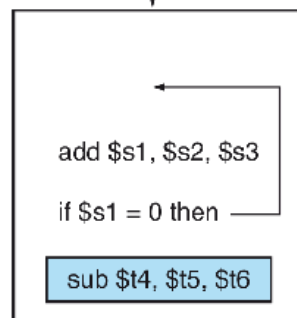
Diviene



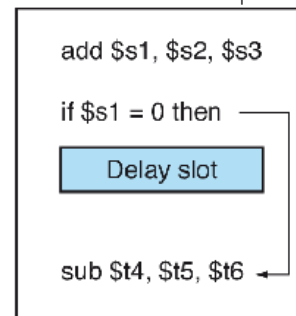
b. Dall'indirizzo di salto



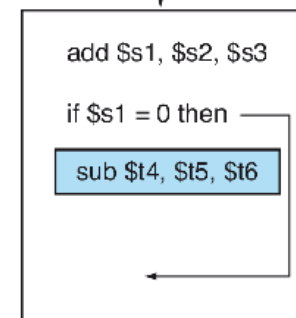
Diviene



c. Dall'indirizzo di salto, nel caso di fallimento della predizione



Diviene



Soluzioni ai conflitti di controllo

■ Soluzioni di tipo hardware

- Inserimento di bolle (**bubble**) o stalli nella pipeline (3 cicli)
 - Si inseriscono dei tempi morti
 - Peggiora il throughput
- Ridurre i ritardi associati ai salti condizionati
 - Comparatore in fase di **decode**
 - Calcolo dell'indirizzo di destinazione in fase di **decode**
- Predizione Statica
 - Si assume **branch taken** o **branch not taken**
 - In caso di errore si **invalida** l'istruzione in esecuzione
- Predizione Dinamica
 - Comparatore in fase di **decode**
 - Tabella della storia dei salti (**branch prediction buffer**)
 - Predizione a 1 o 2 bit

- Dipendenze
- Uso di:

NOP

Stalli

- Calcolo del CPI(clock cycles per instruction)

$$CPI = \frac{\# \text{ TOTALE CICLI}}{\# \text{ TOTALE ISTRUZIONI}}$$

Esercizio 1

Dato il seguente codice assembly:

```
ADD R5, R6, R7  
LW  R6, 200(R5)  
SUB R3, R1, R2  
ADD R4, R3, R4
```

1. Identificare le dipendenze dati del seguente codice
2. Risolvere i conflitti presenti utilizzando le NOP + Calcolare il CPI
3. Risolvere i conflitti presenti utilizzando gli STALLI e calcolare il CPI
4. Risolvere i conflitti presenti RIORDINANDO le istruzioni e introducendo STALLI dove necessario. Calcolare infine il CPI
5. Risolvere i conflitti presenti assumendo che l'architettura supporti la PROPAGAZIONE e calcolare il CPI
6. Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 4

Esercizio 1-1 Identificare le dipendenze dati

```
ADD  R5, R6, R7
LW   R6, 200(R5)
SUB  R3, R1, R2
ADD  R4, R3, R4
```

1. Dipendenze dati del codice:

Dipendenza **RAW** su R5 tra ADD1 e LW2

Dipendenza **RAW** su R3 tra SUB3 e ADD4

Dipendenza **WAR** su R6 tra ADD1 e LW2

Esercizio 1-2 Risolvere i conflitti presenti utilizzando le NOP

1. Tabella

2. Righe: le istruzioni

3. Colonne molte... avremo piu cicli...

4. "Aspetto" introducendo eventuali "NOP" come righe

CICLO	1	2	3	4	..							
ADD1												
LW2												
SUB3												
ADD4												

Nota: NOP e' messa lato codice

Esercizio 1-2 Risolvere i conflitti presenti utilizzando le NOP

ADD R5, R6, R7
LW R6, 200(R5)
SUB R3, R1, R2
ADD R4, R3, R4

1 istruzione ADD:

CICLO	1	2	3	4	5	6	7	8	9	10	11	12
ADD1	F	D	E	M	W							

Esercizio 1-2 Risolvere i conflitti presenti utilizzando le NOP

```
ADD  R5, R6, R7
LW   R6, 200(R5)
SUB  R3, R1, R2
ADD  R4, R3, R4
```

2' istruzione LW deve aspettare la W (di R5) per la sua fase D:

Ma dove la metto?

CICLO	1	2	3	4	5	6	7	8	9	10	11	12
ADD1	F	D	E	M	W							

Nota: posso aggiungere NOP a SX, ma ogni NOP "sposta" in diagonale, dovendo rispettare:

FDEMW

Esercizio 1-2 Risolvere i conflitti presenti utilizzando le NOP

ADD R5, R6, R7
LW R6, 200(R5)
SUB R3, R1, R2
ADD R4, R3, R4

Qui NON va bene:

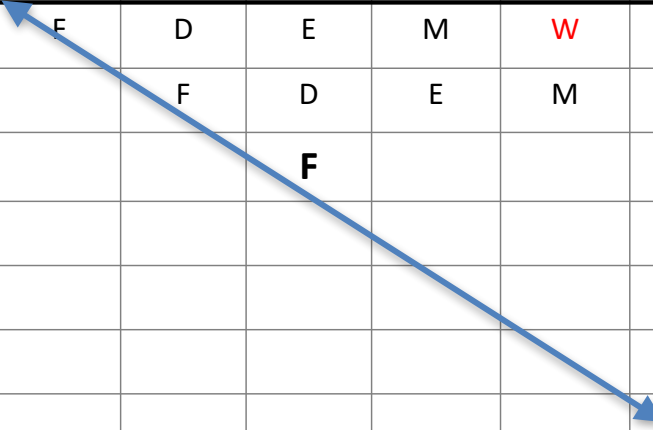
CICLO	1	2	3	4	5	6	7	8	9	10	11	12
ADD1	F	D	E	M	W							
NOP		F	D	E	M	W						
			F	???	D							

Esercizio 1-2 Risolvere i conflitti presenti utilizzando le NOP

```
ADD  R5, R6, R7
LW   R6, 200(R5)
SUB  R3, R1, R2
ADD  R4, R3, R4
```

Aggiungo F, in diagonale...

CICLO	1	2	3	4	5	6	7	8	9	10	11	12
ADD1	F	D	E	M	W							
NOP		F	D	E	M	W						
NOP			F									



Esercizio 1-2 Risolvere i conflitti presenti utilizzando le NOP

ADD R5, R6, R7
LW R6, 200(R5)
SUB R3, R1, R2
ADD R4, R3, R4

2. Quindi 2 NOP:

CICLO	1	2	3	4	5	6	7	8	9	10	11	12
ADD1	F	D	E	M	W							
NOP		F	D	E	M	W						
NOP			F	D	E	M	W					
LW2				F	D	E	M	W				
SUB3												
NOP												
NOP												
ADD4												

Esercizio 1-2 Risolvere i conflitti presenti utilizzando le NOP

ADD R5, R6, R7
LW R6, 200(R5)
SUB R3, R1, R2
ADD R4, R3, R4

2. Risolvere i conflitti presenti utilizzando le NOP

CICLO	1	2	3	4	5	6	7	8	9	10	11	12
ADD1	F	D	E	M	W							
NOP		F	D	E	M	W						
NOP			F	D	E	M	W					
LW2				F	D	E	M	W				
SUB3					F	D	E	M	W			
NOP						F	D	E	M			
NOP							F	D	E	M	W	
ADD4								F	D	E	M	W

Esercizio 1-2: CPI

Calcolare il CPI:

Dalla tabella precedente... 12 "passi" ..

Ma le istruzioni "vere" sono 4..

$$CPI = \frac{\# \text{ TOTALE CICLI}}{\# \text{ TOTALE ISTRUZIONI}} = \frac{12}{4} = 3$$

Esercizio 1-3

Risolvere i conflitti presenti utilizzando gli STALLI

Passi:

- 1) Tabella
- 2) Uno stallo e' ***interno*** al μP
- 3) NON aggiungo righe
- 4) Simbolo X

Esercizio 1-3


1 passo:

CICLO	1	2	3	4	5	6	7	8	9	10	11	12
ADD1	F	D	E	M	W							
LW2												
SUB3												
ADD4												

Esercizio 1-3

2' riga LW, ma "D" deve attendere "W" della 1:

CICLO	1	2	3	4	5	6	7	8	9	10	11	12
ADD1	F	D	E	M	W							
LW2		F			D							
SUB3												
ADD4												



Ma LW inizia subito dopo..

Esercizio 1-3

Quindi il μp avr  due "X" (stalli)

CICLO	1	2	3	4	5	6	7	8	9	10	11	12
ADD1	F	D	E	M	W							
LW2		F	X	X	D							
SUB3												
ADD4												

Esercizio 1-3

Quindi il μp avra' due "X" (stalli) e quindi riga 1 e 2:

CICLO	1	2	3	4	5	6	7	8	9	10	11	12
ADD1	F	D	E	M	W							
LW2		F	X	X	D	E	M	W				
SUB3												
ADD4												

Il Fetch di SUB e' stato fermato dagli stalli...

Esercizio 1-3

Sub 3 procede e produce **W**

CICLO	1	2	3	4	5	6	7	8	9	10	11	12
ADD1	F	D	E	M	W							
LW2		F	X	X	D	E	M	W				
SUB3					F	D	E	M	W			
ADD4												

Esercizio 1-3

Add 4 esegue fetch, ma "D" deve aspettare "W"

CICLO	1	2	3	4	5	6	7	8	9	10	11	12
ADD1	F	D	E	M	W							
LW2		F	X	X	D	E	M	W				
SUB3					F	D	E	M	W			
ADD4						F	X	X	D	E	M	W

Quindi 2 stalli

Esercizio 1-3: CPI

Calcolare il CPI:

Dalla tabella precedente... 12 "passi" ..

Ma le istruzioni "vere" sono 4..

$$CPI = \frac{\# \text{ TOTALE CICLI}}{\# \text{ TOTALE ISTRUZIONI}} = \frac{12}{4} = 3$$

Esercizio 1-4

4. Risolvere i conflitti presenti RIORDINANDO le istruzioni e introducendo STALLI dove necessario.

ADD R5, R6, R7
LW R6, 200(R5)
SUB R3, R1, R2
ADD R4, R3, R4

CICLO	1	2	3	4	5	6	7	8	9
ADD1	F	D	E	M	W				
SUB3		F	D	E	M	W			
LW2			F	X	D	E	M	W	
ADD4					F	D	E	M	W

Spostato SUB.

Esercizio 1-4:CPI

4. Risolvere i conflitti presenti RIORDINANDO le istruzioni e introducendo STALLI dove necessario. **Calcolare infine il CPI**

$$CPI = \frac{\# \text{ TOTALE CICLI}}{\# \text{ TOTALE ISTRUZIONI}} = \frac{9}{4} = 2.25$$

Esercizio 1-5

Risolvere i conflitti presenti assumendo che l'architettura supporti la PROPAGAZIONE

Logica: non aspetto W, mi basta M...

Esercizio 1-5

Risolvere i conflitti presenti assumendo che l'architettura supporti la PROPAGAZIONE

Logica: non aspetto W, mi basta M:

CICLO	1	2	3	4	5	6	7	8
ADD1	F	D	E	M	W			
LW2		F	D	E	M	W		
SUB3			F	D	E	M	W	
ADD4				F	D	E	M	W

Esercizio 1-5:CPI

Risolvere i conflitti presenti assumendo che l'architettura supporti la PROPAGAZIONE
e **calcolare il CPI**

$$CPI = \frac{\# \text{ TOTALE CICLI}}{\# \text{ TOTALE ISTRUZIONI}} = \frac{8}{4} = 2$$

Esercizio 1-6: Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 4

CICLO	1	2	3	4	5	6	7	8
ADD1	F	D	E	M	W			
LW2		F	D	E	M	W		
SUB3			F	D	E	M	W	
ADD4				F	D	E	M	W

Segnali:

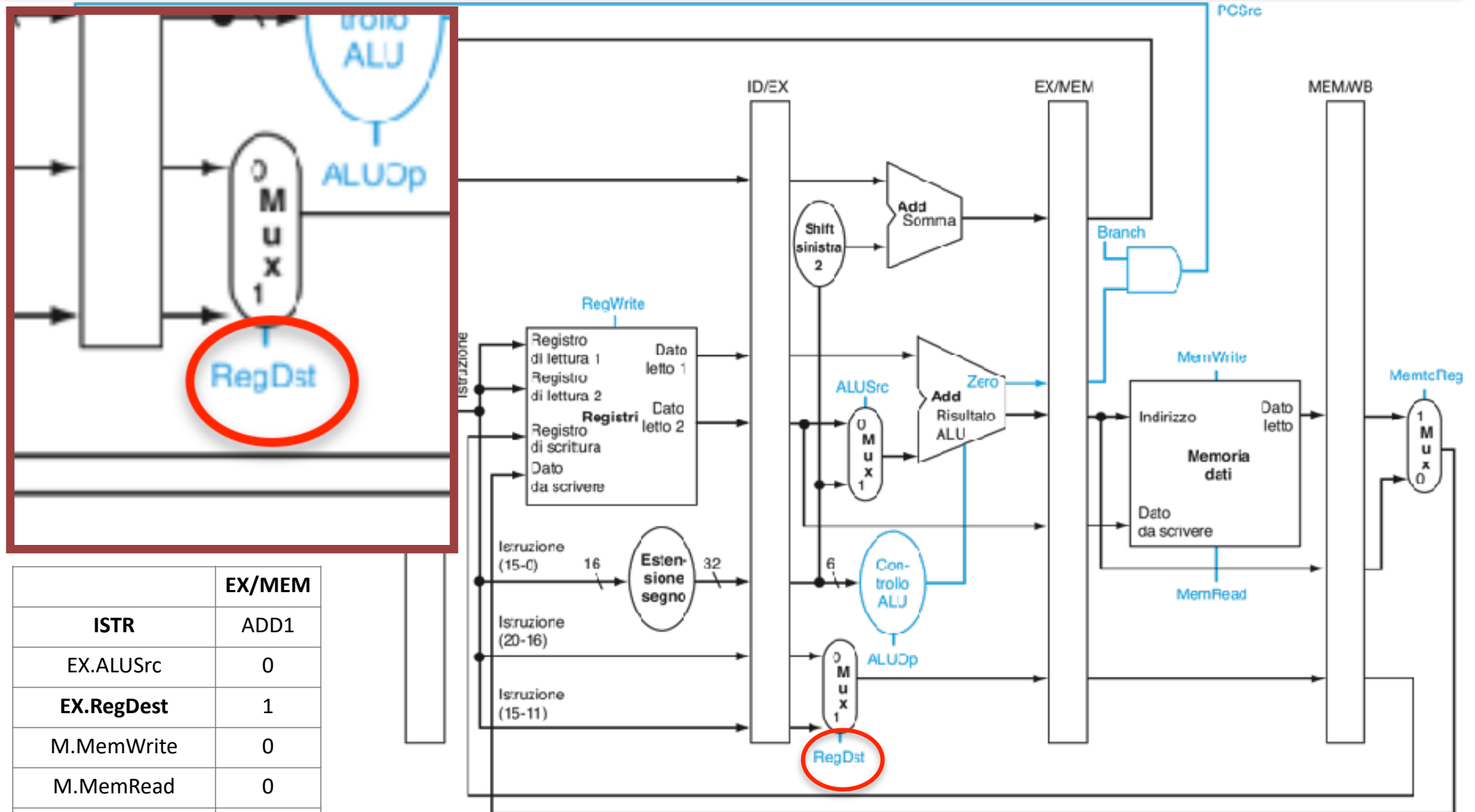
EX.ALUSrc
EX.RegDest
M.MemWrite
M.MemRead
M.Branch
WB.MemToReg
WB.RegWrite

The diagram illustrates the MIPS processor architecture across five stages: IF/ID, ID/EX, EX/MEM, and MEM/WB. A detailed inset shows the ALUSrc logic, which uses a 2-to-1 multiplexer to select between register data and an immediate value from the instruction. The main diagram shows the Register File, ALU, Branch logic, Memory, and a final multiplexer for the PC. The ALU performs operations based on ALUOp, and the Branch logic determines if a branch should be taken based on ALU results and instruction flags. The Memory stage handles data reads and writes, and the final MEM/WB stage updates the PC.

	EX/MEM
ISTR	ADD1
EX.ALUSrc	0
EX.RegDest	1
M.MemWrite	0
M.MemRead	0

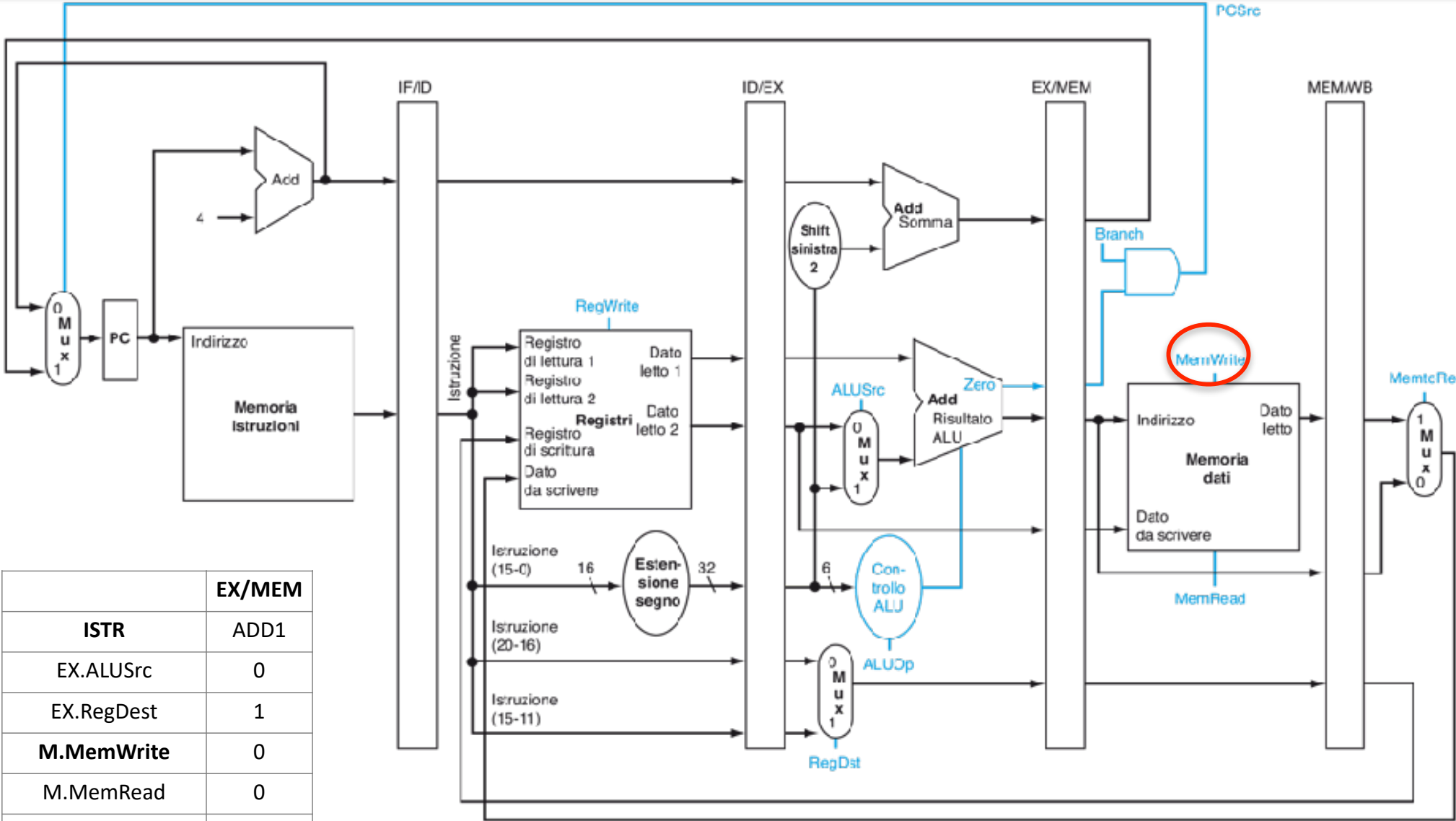
	EX/MEM
ISTR	ADD1
EX.ALUSrc	0
EX.RegDest	1
M.MemWrite	0
M.MemRead	0
M.Branch	0
WB.MemToReg	0
WB.RegWrite	1

Esercizio 1-6: Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 4



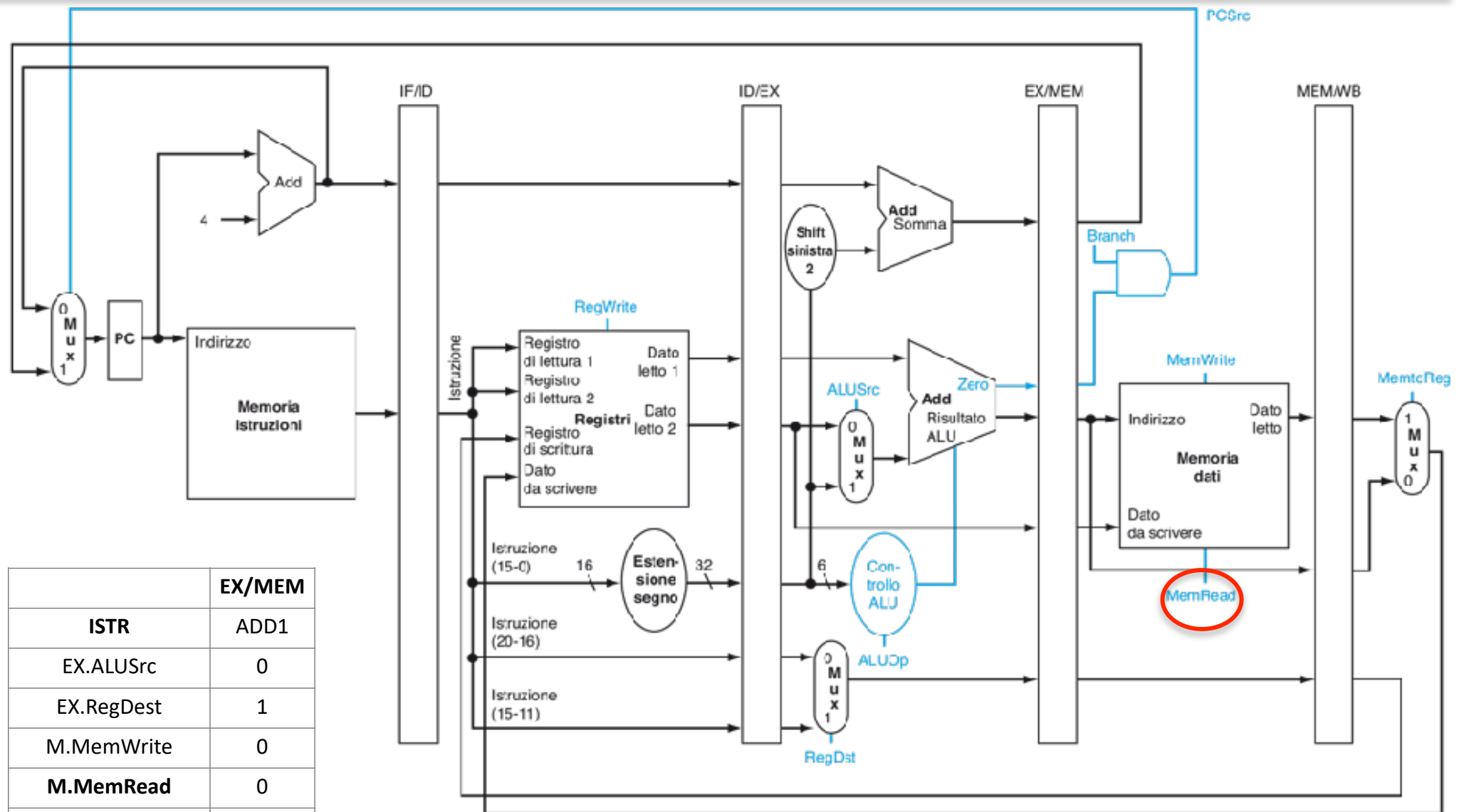
	EX/MEM
ISTR	ADD1
EX.ALUSrc	0
EX.RegDest	1
M.MemWrite	0
M.MemRead	0
M.Branch	0
WB.MemToReg	0
WB.RegWrite	1

Esercizio 1-6: Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 4

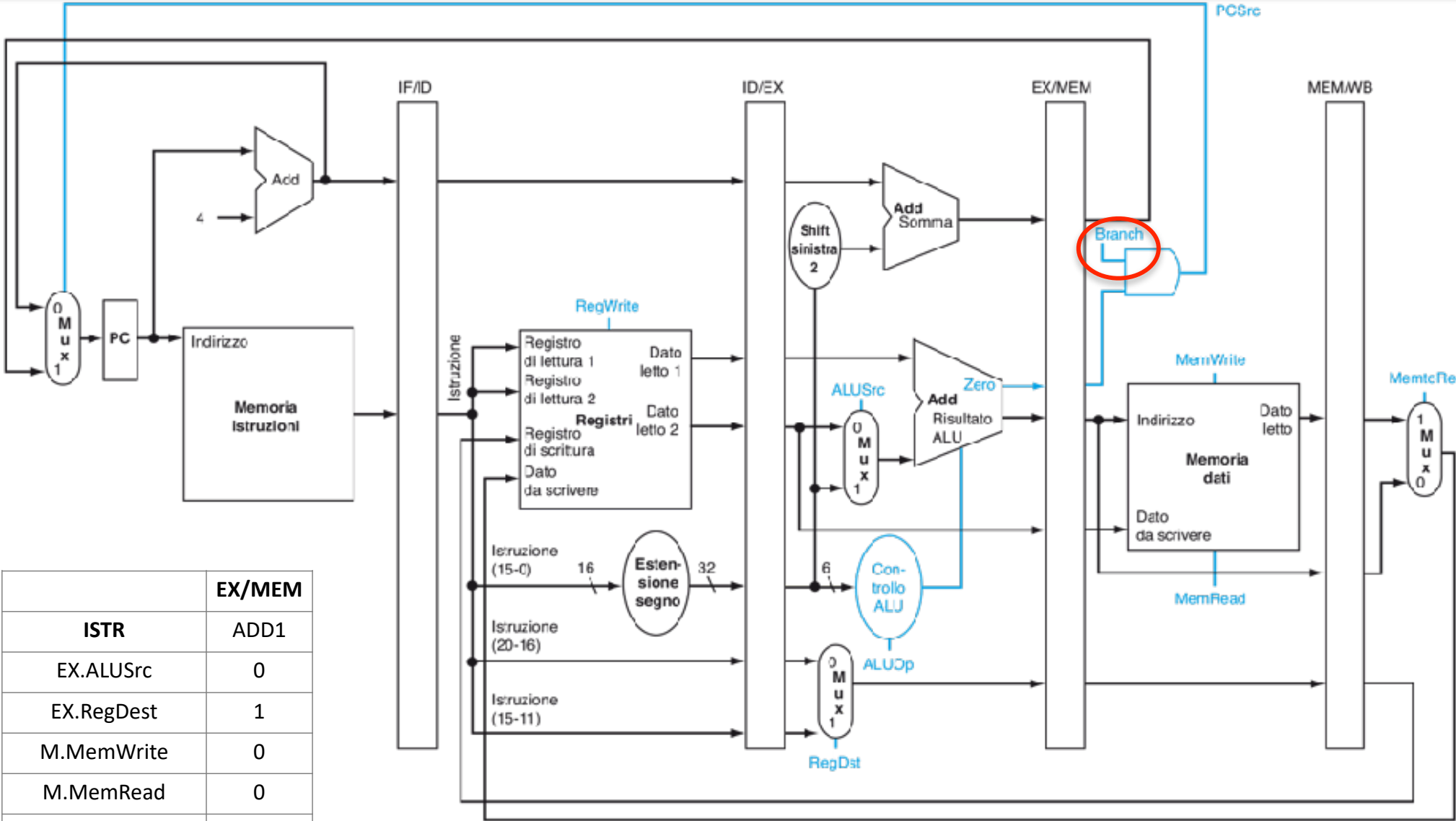


	EX/MEM
ISTR	ADD1
EX.ALUSrc	0
EX.RegDest	1
M.MemWrite	0
M.MemRead	0
M.Branch	0
WB.MemToReg	0
WB.RegWrite	1

Esercizio 1-6: Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 4

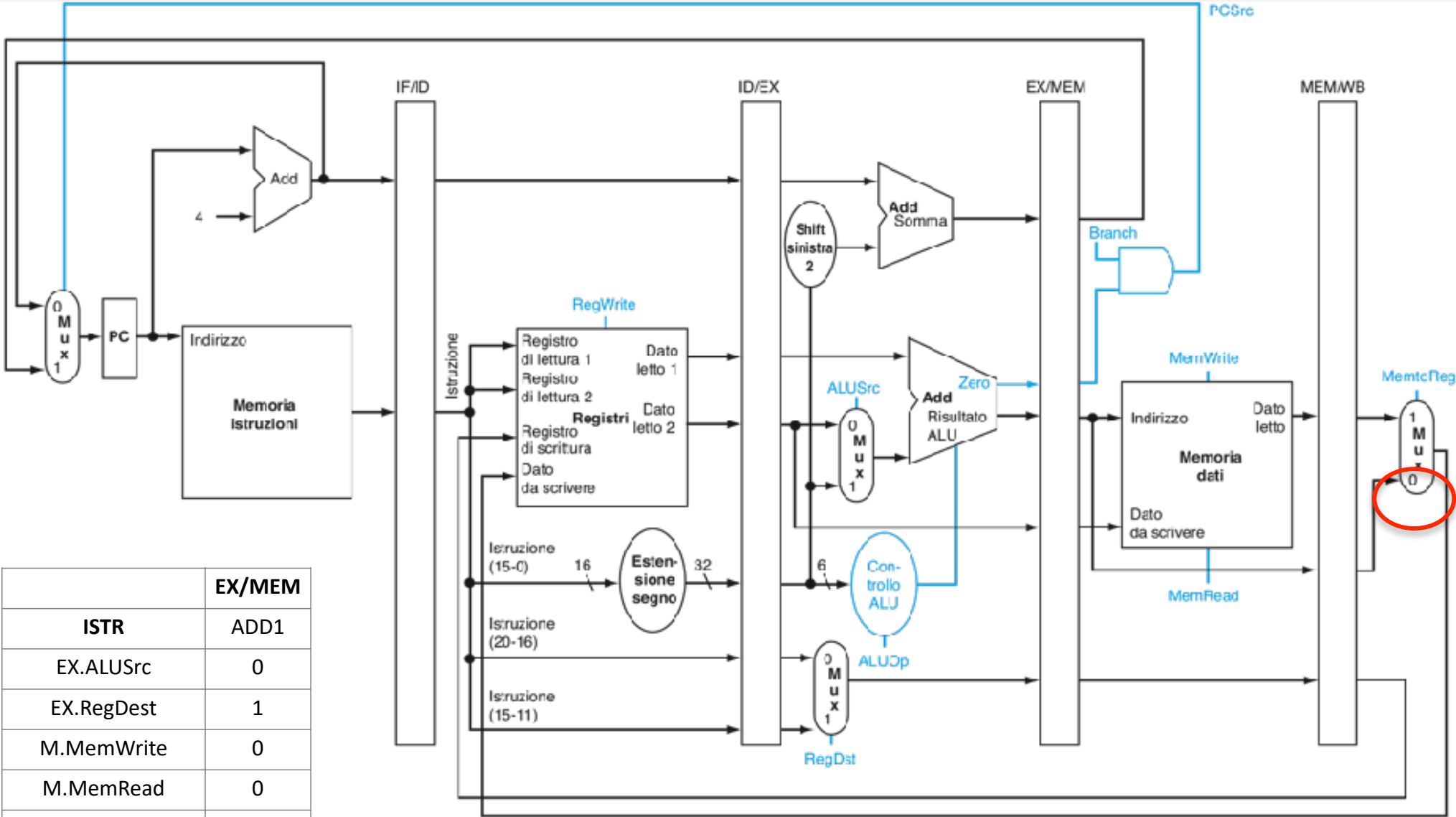


Esercizio 1-6: Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 4

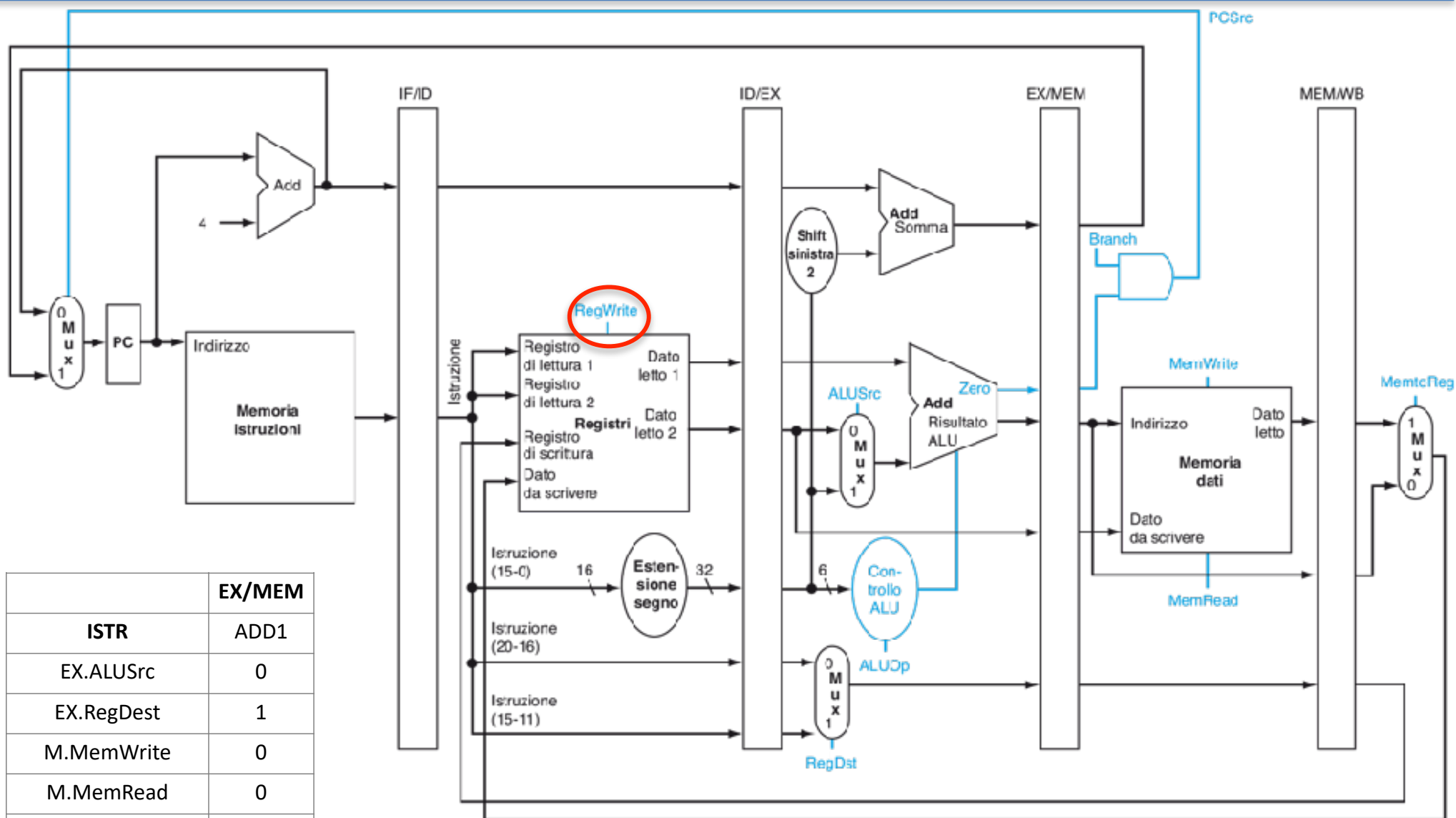


	EX/MEM
ISTR	ADD1
EX.ALUSrc	0
EX.RegDest	1
M.MemWrite	0
M.MemRead	0
M.Branch	0
WB.MemToReg	0
WB.RegWrite	1

Esercizio 1-6: Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 4

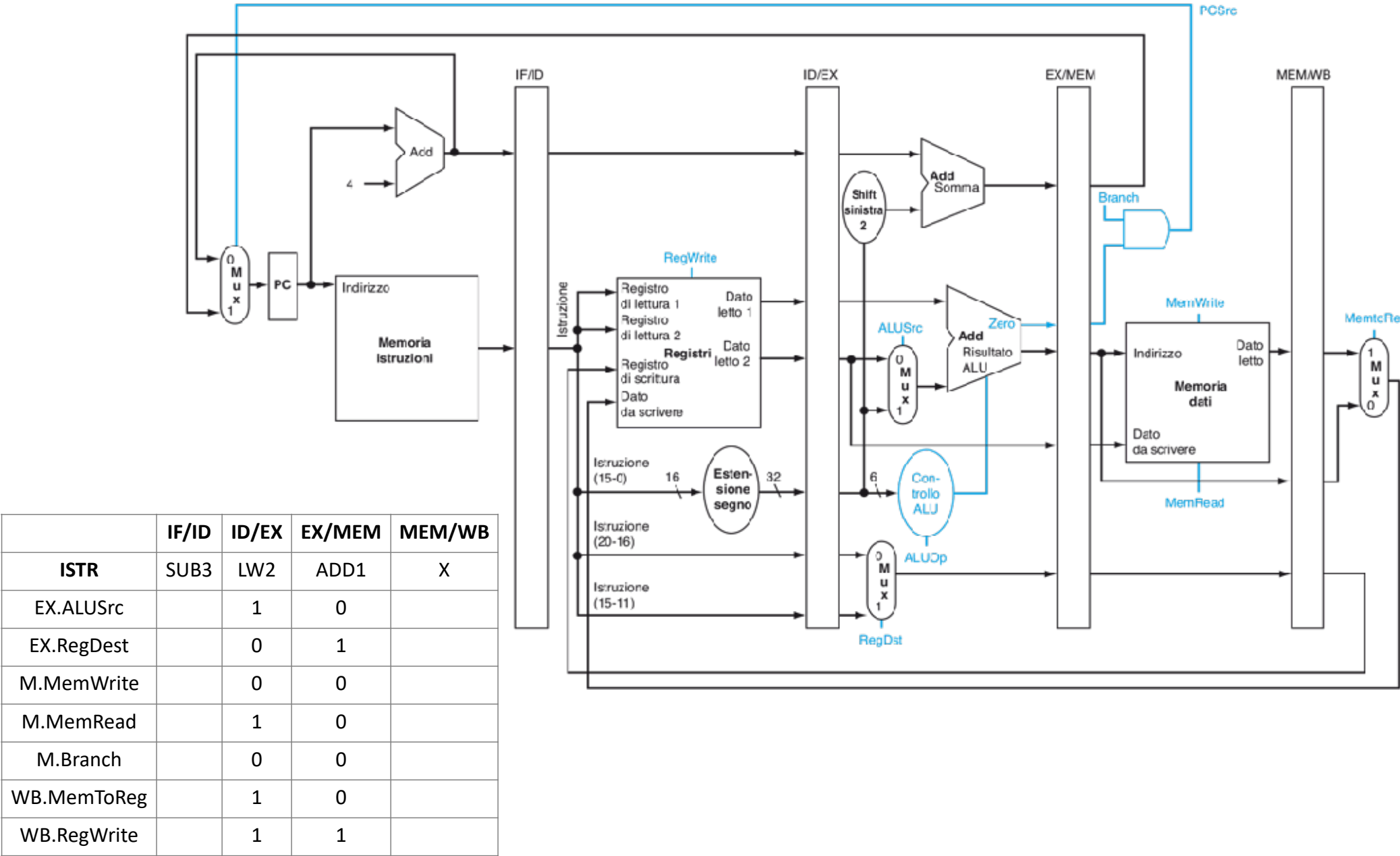


Esercizio 1-6: Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 4

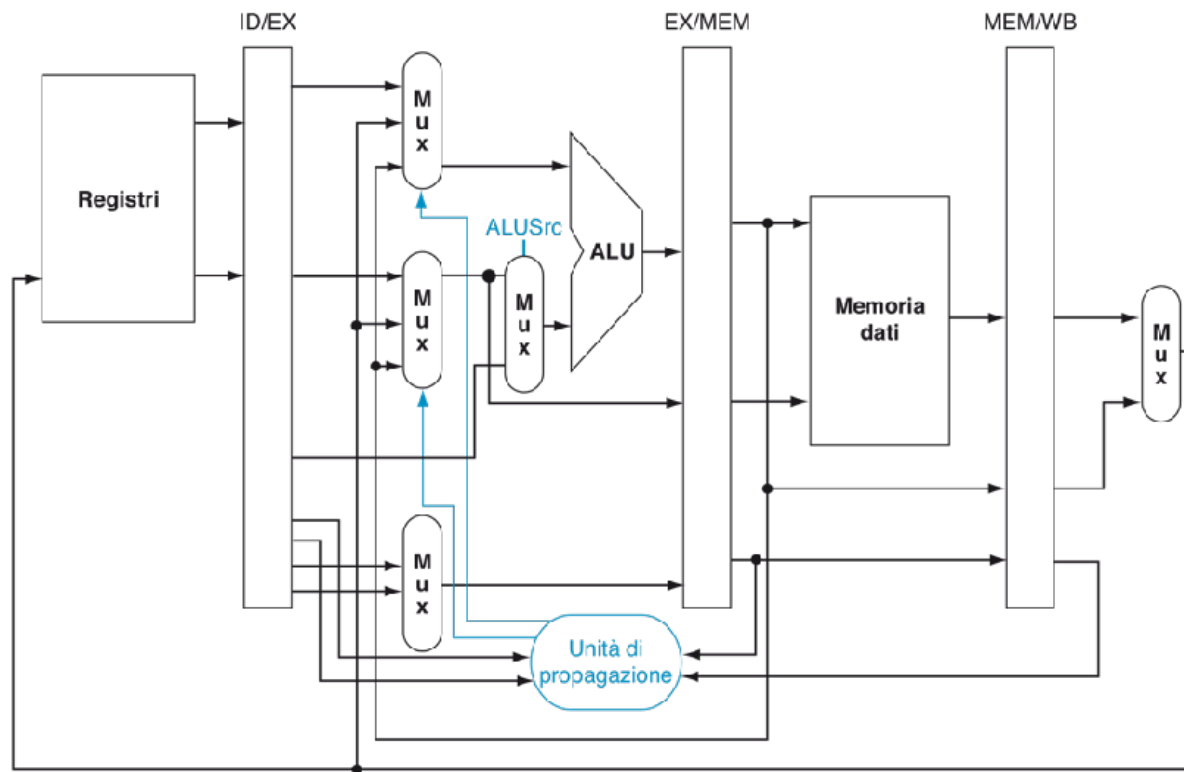


	EX/MEM
ISTR	ADD1
EX.ALUsrc	0
EX.RegDest	1
M.MemWrite	0
M.MemRead	0
M.Branch	0
WB.MemToReg	0
WB.RegWrite	1

Esercizio 1-6: Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 4



Esercizio 1-6 B: Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 4



	Valore
PropagaA	10
PropagaB	00

Esercizio 2

Dato il seguente codice assembly:

```
ADD  R5, R6, R7  
LW   R6, 200(R5)  
SUB  R5, R6, R7
```

1. Identificare le dipendenze dati del seguente codice
2. Risolvere i conflitti presenti utilizzando le NOP e calcolare il CPI
3. Risolvere i conflitti presenti utilizzando gli STALLI e calcolare il CPI
4. Risolvere i conflitti presenti RIORDINANDO le istruzioni e introducendo STALLI dove necessario. Calcolare infine il CPI
5. Risolvere i conflitti presenti assumendo che l'architettura supporti la PROPAGAZIONE e calcolare il CPI
6. Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 6

Esercizio 2-1

Dato il seguente codice assembly:

```
ADD  R5, R6, R7
LW   R6, 200(R5)
SUB  R5, R6, R7
```

Identificare le dipendenze

Dipendenza **RAW** su R5 tra ADD1 e LW2

Dipendenza **RAW** su R6 tra LW2 e SUB3

Dipendenza **WAW** su R5 tra ADD1 e SUB3

Dipendenza **WAR** su R6 tra ADD1 e LW2

Dipendenza **WAR** su R5 tra LW2 e SUB3

Esercizio 2-2

Dato il seguente codice assembly:

```
ADD  R5, R6, R7
LW   R6, 200(R5)
SUB  R5, R6, R7
```

Risolvere i conflitti presenti utilizzando le NOP e calcolare il CPI

CICLO	1	2	3	4	5	6	7	8	9	10	11
ADD1	F	D	E	M	W						
NOP		F	D	E	M	W					
NOP			F	D	E	M	W				
LW2				F	D	E	M	W			
NOP					F	D	E	M	W		
NOP						F	D	E	M	W	
SUB3							F	D	E	M	W

$$CPI = \frac{\# \text{ TOTALE CICLI}}{\# \text{ TOTALE ISTRUZIONI}} = \frac{11}{3} = 3.67$$

Esercizio 2-3

Dato il seguente codice assembly:

```
ADD  R5, R6, R7
LW   R6, 200(R5)
SUB  R5, R6, R7
```

Risolvere i conflitti presenti utilizzando gli STALLI e calcolare il CPI

CICLO	1	2	3	4	5	6	7	8	9	10	11
ADD1	F	D	E	M	W						
LW2		F	X	X	D	E	M	W			
SUB3					F	X	X	D	E	M	W

$$CPI = \frac{\# \text{ TOTALE CICLI}}{\# \text{ TOTALE ISTRUZIONI}} = \frac{11}{3} = 3.67$$

Esercizio 2-4

Dato il seguente codice assembly:

```
ADD    R5, R6, R7  
LW     R6, 200(R5)  
SUB    R5, R6, R7
```

Risolvere i conflitti presenti RIORDINANDO le istruzioni e introducendo STALLI dove necessario. Calcolare infine il CPI

Non è possibile introdurre nessun ordinamento per migliorare le prestazioni quindi l'esecuzione rimane identica al punto precedente.

Esercizio 2-5

Dato il seguente codice assembly:

```
ADD  R5, R6, R7
LW   R6, 200(R5)
SUB  R5, R6, R7
```

Risolvere i conflitti presenti assumendo che l'architettura supporti la PROPAGAZIONE e calcolare il CPI

CICLO	1	2	3	4	5	6	7	8
ADD1	F	D	E	M	W			
LW2		F	D	E	M	W		
SUB3			F	X	D	E	M	W

$$CPI = \frac{\# \text{ TOTALE CICLI}}{\# \text{ TOTALE ISTRUZIONI}} = \frac{8}{3} = 2.67$$

Esercizio 2-6

Dato il seguente codice assembly:

```
ADD    R5, R6, R7
LW     R6, 200(R5)
SUB    R5, R6, R7
```

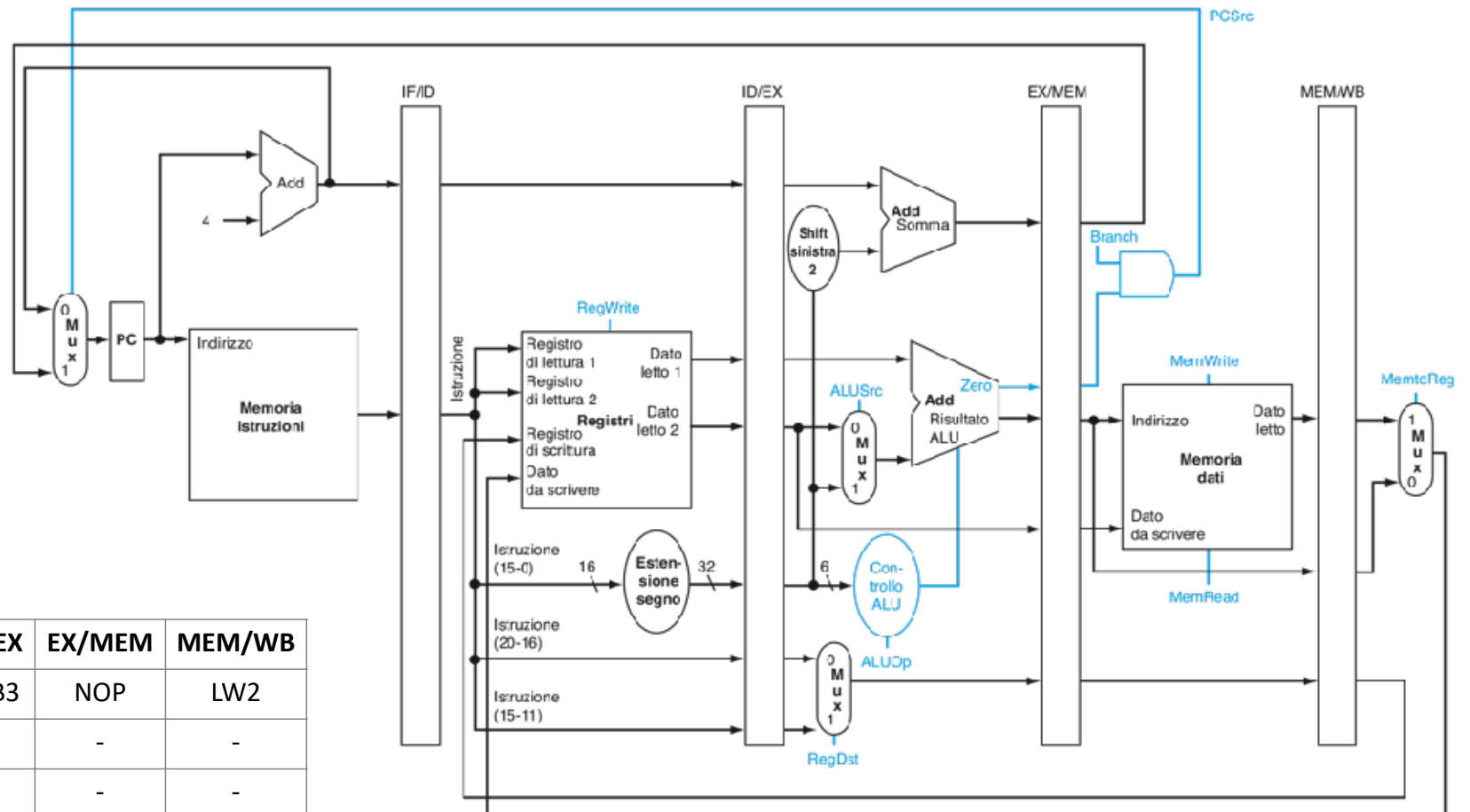
Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 6

CICLO	1	2	3	4	5	6	7	8
ADD1	F	D	E	M	W			
LW2		F	D	E	M	W		
SUB3			F	X	D	E	M	W

ISTR				
EX.ALUSrc				
EX.RegDest				
M.MemWrite				
M.MemRead				
M.Branch				
WB.MemToReg				
WB.RegWrite				

Esercizio 2-6

Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 6



	IF/ID	ID/EX	EX/MEM	MEM/WB
ISTR	X	SUB3	NOP	LW2
EX.ALUSrc		0	-	-
EX.RegDest		1	-	-
M.MemWrite		0	0	-
M.MemRead		0	0	-
M.Branch		0	0	-
WB.MemToReg		0	0	1
WB.RegWrite		1	0	1

Esercizio 3

Dato il seguente codice assembly:

```
ADD  R3, R6, R7
SUB  R5, R3, R1
LW   R6, 100(R3)
SUB  R7, R3, R6
```

1. Identificare le dipendenze dati del seguente codice
2. Risolvere i conflitti presenti utilizzando le NOP e calcolare il CPI
3. Risolvere i conflitti presenti utilizzando gli STALLI e calcolare il CPI
4. Risolvere i conflitti presenti RIORDINANDO le istruzioni e introducendo STALLI dove necessario. Calcolare infine il CPI
5. Risolvere i conflitti presenti assumendo che l'architettura supporti la PROPAGAZIONE e calcolare il CPI
6. Risolvere i conflitti presenti assumendo che l'architettura supporti la PROPAGAZIONE e RIORDINO e calcolare il CPI
7. Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 4
8. Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 6

Esercizio 3-1

Dato il seguente codice assembly:

```
ADD    R3, R6, R7
SUB     R5, R3, R1
LW      R6, 100(R3)
SUB     R7, R3, R6
```

1. Identificare le dipendenze dati del seguente codice

Dipendenza **RAW** su R3 tra ADD1 e SUB2

Dipendenza **RAW** su R3 tra ADD1 e LW3

Dipendenza **RAW** su R3 tra ADD1 e SUB4

Dipendenza **RAW** su R6 tra LW3 e SUB4

Dipendenza **WAR** su R6 tra ADD1 e LW3

Esercizio 3-2

Dato il seguente codice assembly:

```
ADD  R3, R6, R7
SUB  R5, R3, R1
LW   R6, 100(R3)
SUB  R7, R3, R6
```

Risolvere i conflitti presenti utilizzando le NOP e calcolare il CPI

CICLO	1	2	3	4	5	6	7	8	9	10	11	12
ADD1	F	D	E	M	W							
NOP		F	D	E	M	W						
NOP			F	D	E	M	W					
SUB2				F	D	E	M	W				
LW3					F	D	E	M	W			
NOP						F	D	E	M	W		
NOP							F	D	E	M	W	
SUB4								F	D	E	M	M

$$CPI = \frac{\# \text{ TOTALE CICLI}}{\# \text{ TOTALE ISTRUZIONI}} = \frac{12}{4} = 3$$

Esercizio 3-3

Dato il seguente codice assembly:

```
ADD  R3, R6, R7
SUB  R5, R3, R1
LW   R6, 100(R3)
SUB  R7, R3, R6
```

Risolvere i conflitti presenti utilizzando gli STALLI e calcolare il CPI

CICLO	1	2	3	4	5	6	7	8	9	10	11	12
ADD1	F	D	E	M	W							
SUB2		F	X	X	D	E	M	W				
LW3					F	D	E	M	W			
SUB4						F	X	X	D	E	M	W

$$CPI = \frac{\# TOTALE CICLI}{\# TOTALE ISTRUZIONI} = \frac{12}{4} = 3$$

Esercizio 3-4

Dato il seguente codice assembly:

```
ADD  R3, R6, R7
SUB  R5, R3, R1
LW   R6, 100(R3)
SUB  R7, R3, R6
```

Risolvere i conflitti presenti RIORDINANDO le istruzioni e introducendo STALLI dove necessario. Calcolare infine il CPI

CICLO	1	2	3	4	5	6	7	8	9	10	11
ADD1	F	D	E	M	W						
LW3		F	X	X	D	E	M	W			
SUB2					F	D	E	M	W		
SUB4						F	X	D	E	M	W

$$CPI = \frac{\# TOTALE CICLI}{\# TOTALE ISTRUZIONI} = \frac{11}{4} = 2.75$$

Esercizio 3-5

Dato il seguente codice assembly:

```
ADD  R3, R6, R7
SUB  R5, R3, R1
LW   R6, 100(R3)
SUB  R7, R3, R6
```

Risolvere i conflitti presenti assumendo che l'architettura supporti la PROPAGAZIONE e calcolare il CPI

CICLO	1	2	3	4	5	6	7	8	9
ADD1	F	D	E	M	W				
SUB2		F	D	E	M	W			
LW3			F	D	E	M	W		
SUB4				F	X	D	E	M	W

$$CPI = \frac{\# \text{ TOTALE CICLI}}{\# \text{ TOTALE ISTRUZIONI}} = \frac{9}{4} = 2.25$$

Esercizio 3-6

Dato il seguente codice assembly:

```
ADD  R3, R6, R7
SUB  R5, R3, R1
LW   R6, 100(R3)
SUB  R7, R3, R6
```

Risolvere i conflitti presenti assumendo che l'architettura supporti la PROPAGAZIONE e RIORDINO e calcolare il CPI

CICLO	1	2	3	4	5	6	7	8
ADD1	F	D	E	M	W			
LW3		F	D	E	M	W		
SUB2			F	D	E	M	W	
SUB4				F	D	E	M	W

$$CPI = \frac{\#TOTALE\ CICLI}{\#TOTALE\ ISTRUZIONI} = \frac{8}{4} = 2$$

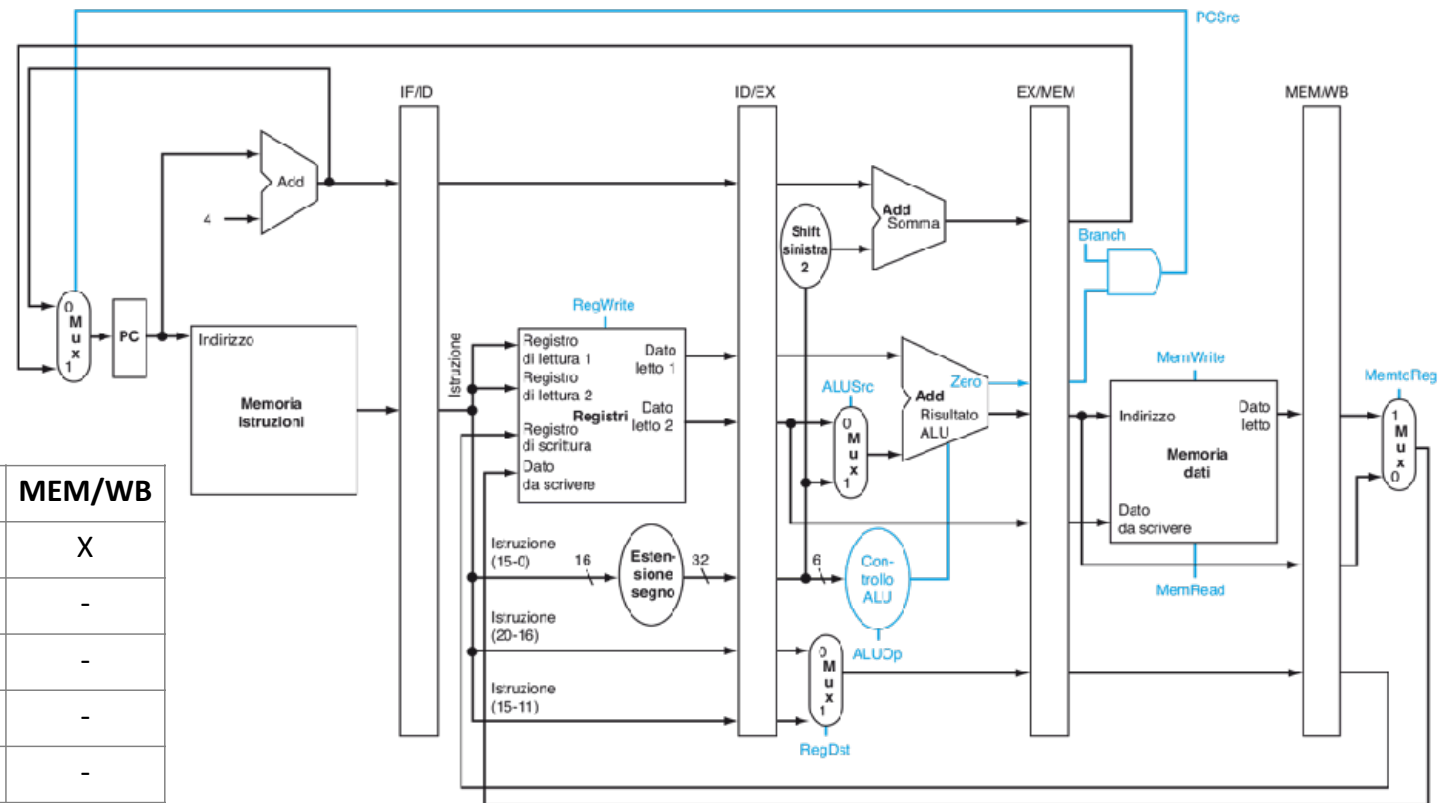
Esercizio 3-7

Dato il seguente codice assembly:

```
ADD    R3, R6, R7
SUB    R5, R3, R1
LW     R6, 100(R3)
SUB    R7, R3, R6
```

Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 4

	IF/ID	ID/EX	EX/MEM	MEM/WB
ISTR	SUB2	LW3	ADD1	X
EX.ALUSrc	-	1	-	-
EX.RegDest	-	0	-	-
M.MemWrite	-	0	0	-
M.MemRead	-	1	0	-
M.Branch	-	0	0	-
WB.MemToReg	-	1	0	-
WB.RegWrite	-	1	1	-



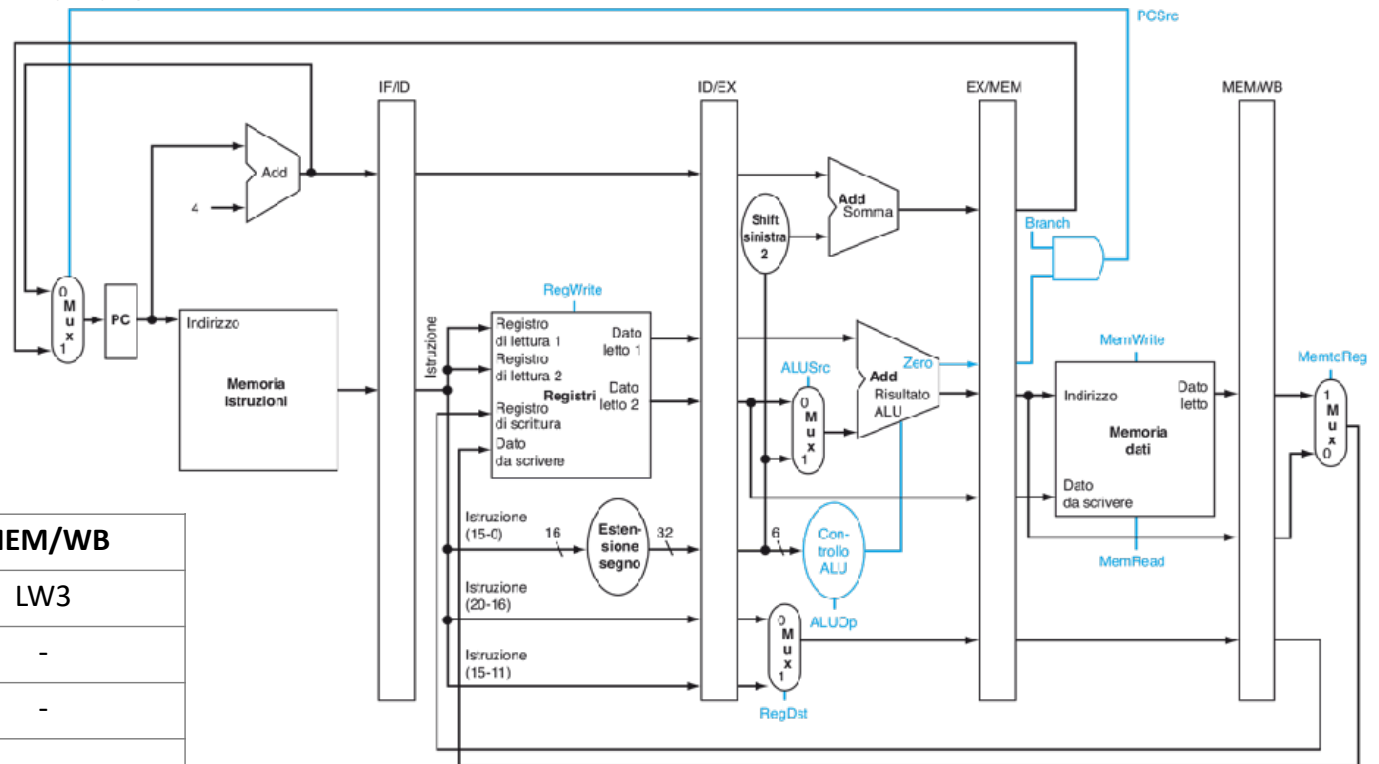
Esercizio 3-8

Dato il seguente codice assembly:

```
ADD  R3, R6, R7
SUB  R5, R3, R1
LW   R6, 100(R3)
SUB  R7, R3, R6
```

8. Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 6

	IF/ID	ID/EX	EX/MEM	MEM/WB
ISTR	X	SUB4	SUB2	LW3
EX.ALUSrc	-	0	-	-
EX.RegDest	-	1	-	-
M.MemWrite	-	0	0	-
M.MemRead	-	0	0	-
M.Branch	-	0	0	-
WB.MemToReg	-	0	0	1
WB.RegWrite	-	1	1	1



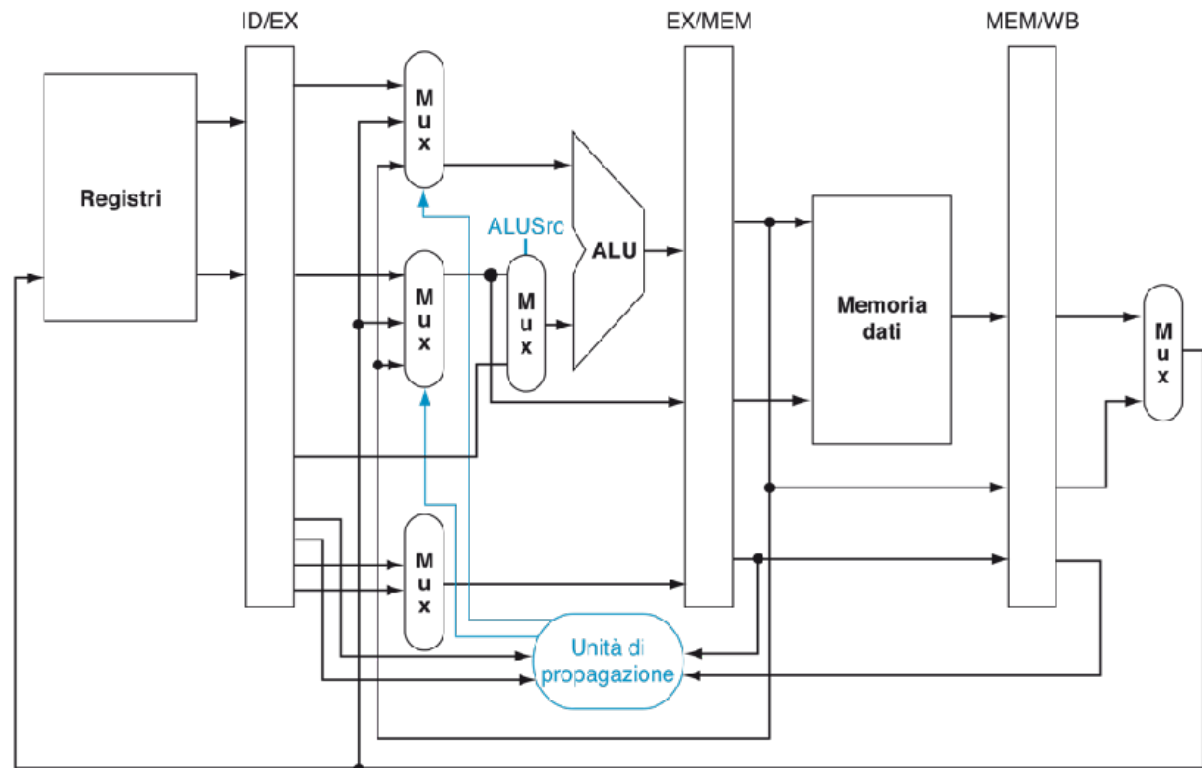
Esercizio 3-8-B

Dato il seguente codice assembly:

```
ADD  R3, R6, R7
SUB  R5, R3, R1
LW   R6, 100(R3)
SUB  R7, R3, R6
```

8. Mostrare i segnali di controllo dell'architettura con propagazione nel ciclo 6

	Valore
PropagaA	00
PropagaB	01



Next Session

Es 4 5 6 8