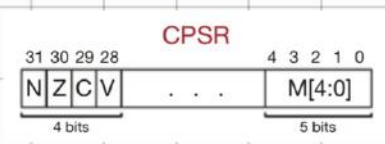
Flag dell'ALU

In ARM le flag di stato (o condizione) sono N, Z, C e V e sono memorizzate nei 4 bit più significativi del CPSR (Current Programm Status Register)

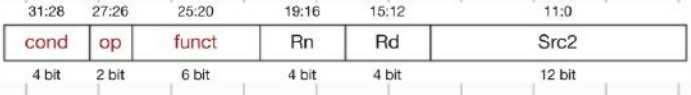


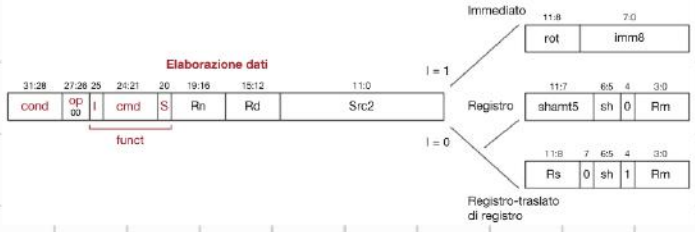
Flag	Nome	Descrizione
N	*Negative*	Il risultato dell'istruzione è negativo, cioè il bit 31 vale 1
Z	*Zero*	Il risultato dell'istruzione è zero
C	*Carry*	L'istruzione ha generato riporto (*carry*) di uscita
V	*oVerflow*	L'istruzione ha causato traboccamento (*overflow*)

Mnemonici di condizione

cond	Mnemonico	Nome	CondEse
0000	EQ	Uguale (*EQual*)	Z
0001	NE	Diverso (*Not Equal*)	$\overline{Z}$
0010	CS/HS	Attiva riporto/maggiore o uguale senza segno (*Carry Set/unsigned Higher or Same*)	C
0011	CC/LO	Disattiva riporto/minore senza segno (*Carry Clear/unsigned LOwer*)	$\overline{C}$
0100	MI	Meno/negativo (*MInus/negative*)	N
0101	PL	Più/positivo o nullo (*PLus/positive or zero*)	$\overline{N}$
0110	VS	Traboccamento/attiva traboccamento (*overflow/oVerflow Set*)	V
0111	VC	No traboccamento/disattiva traboccamento (*overflow/oVerflow Clear*)	$\overline{V}$
1000	HI	Maggiore senza segno (*unsigned HIgher*)	$\overline{Z}C$
1001	LS	Minore o uguale senza segno (*unsigned Lower or Same*)	Z OR $\overline{C}$
1010	GE	Maggiore o uguale con segno (*signed Greater than or Equal*)	$\overline{N \oplus V}$
1011	LT	Minore con segno (*signed Less Than*)	$N \oplus V$
1100	GT	Maggiore con segno (*signed Greater Than*)	$\overline{Z}\ (\overline{N \oplus V})$
1101	LE	Minore o uguale con segno (*signed Less than or Equal*)	Z OR $(N \oplus V)$
1110	AL (o niente)	Sempre/incondizionato (*ALways/unconditional*)	Ignorato

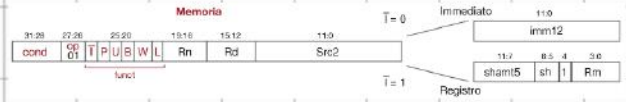
Istruzione di elaborazione dati





Il campo cond codifica l'eventuale esecuzione condizionata sulla base delle flag, mentre l'operazione che l'istruzione deve svolgere è codificata nei campi op (operation code) e funct. In questo caso op = 00, funct ha tre campi: I, cmd e S. I = 1 se Src2 è un immediato (o altrimenti). S = 1 quando l'istruzione imposta le flag di condizione, cmd indica la specifica operazione da svolgere.

Istruzioni di accesso a memoria



P	W	Modo di gestione indice
0	0	Post-indice
0	1	Non supportato
1	0	Spiazzamento
1	1	Pre-indice

L	B	Istruzione
0	0	STR
0	1	STRB
1	0	LDR
1	1	LDRB

Qui op = 01 e il campo funct è suddiviso in maniera diversa: $\bar{I}, P, U, B, W, L$. $\bar{I}$ determina se Src2 è un immediato o meno e U indica se deve essere sommato o sottratto. I due bit P (pre-index) e W (writeback) specificano il modo di gestione indice mentre L (load) e B (byte) specificano il tipo di accesso a memoria.

Bit	Significato $\bar{I}$	Significato U
0	Spiazzamento immediato in Src2	Sottrae lo spiazzamento dalla base
1	Spiazzamento a registro in Src2	Somma lo spiazzamento alla base

Istruzioni di salto

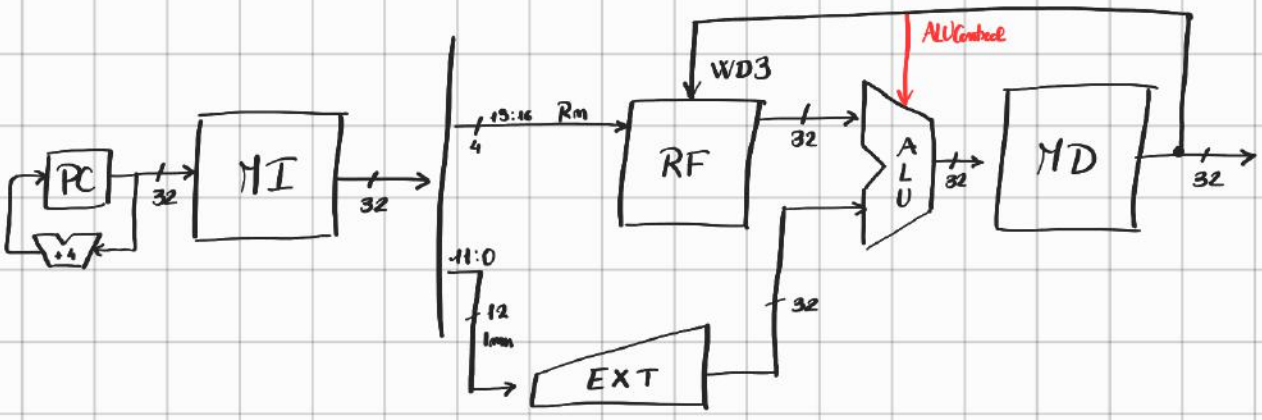
Usano un solo operando immediato a 24 bit signed. op = 10 mentre funct è di soli 2 bit: il più significativo è sempre 1, il meno significativo, L, indica il tipo di salto, 1 per BL e 0 per B.

L	B	Istruzione
0	0	STR
0	1	STRB
1	0	LDR
1	1	LDRB

Processore single cycle

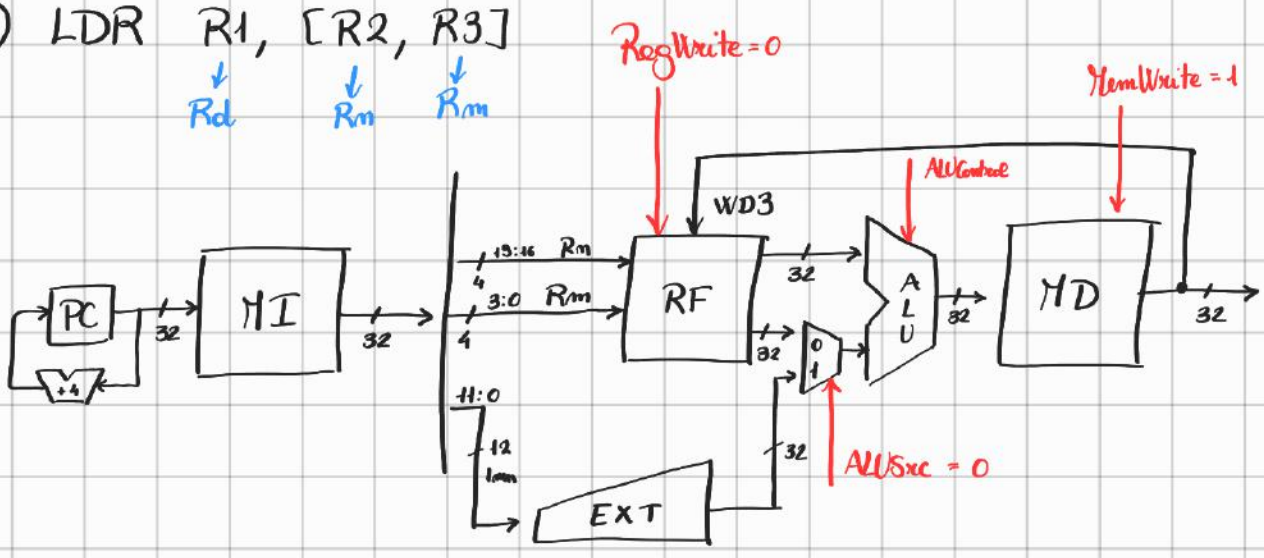
1) LDR R1, [R2, #OFF]

Rd Rn Imm



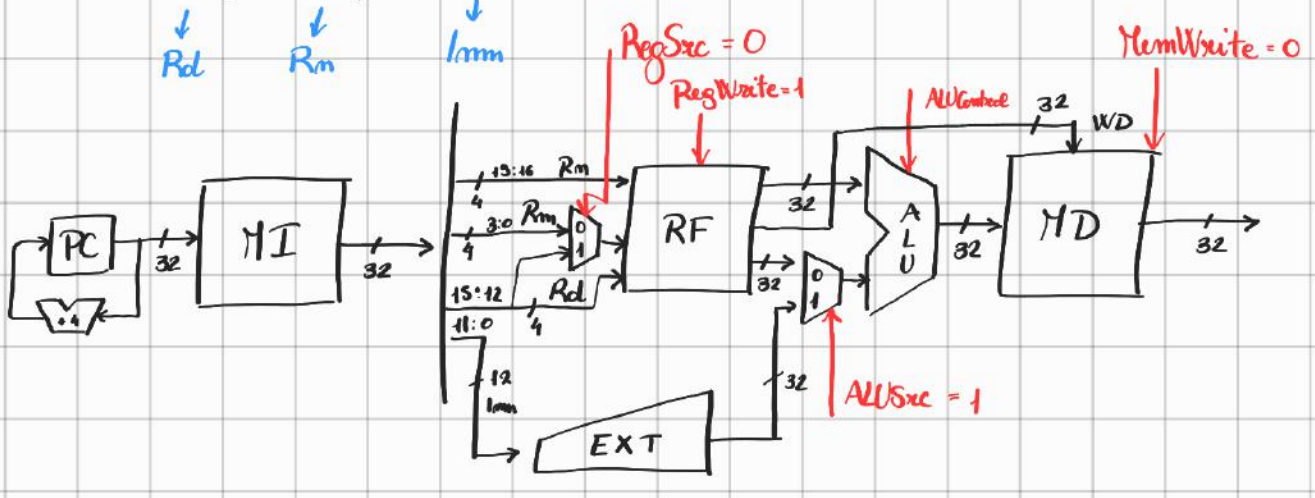
2) LDR R1, [R2, R3]

Rd Rn Rm



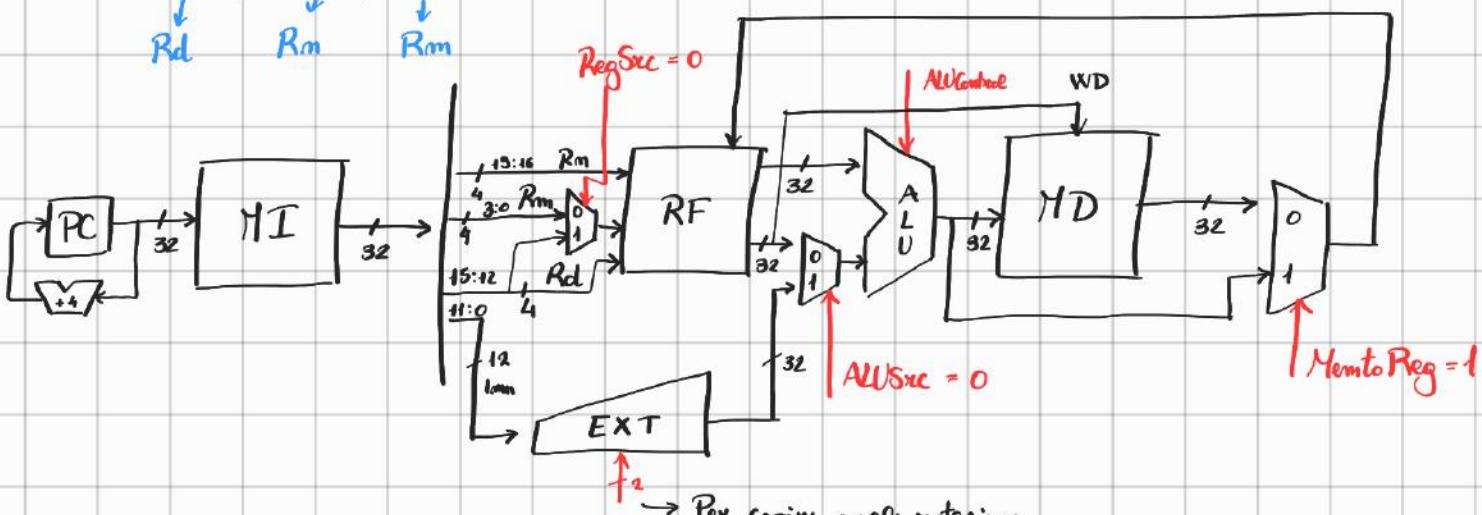
3) STR R1, [R2, #OFF]

Rd Rn Imm



4) ADD R1, R2, R3

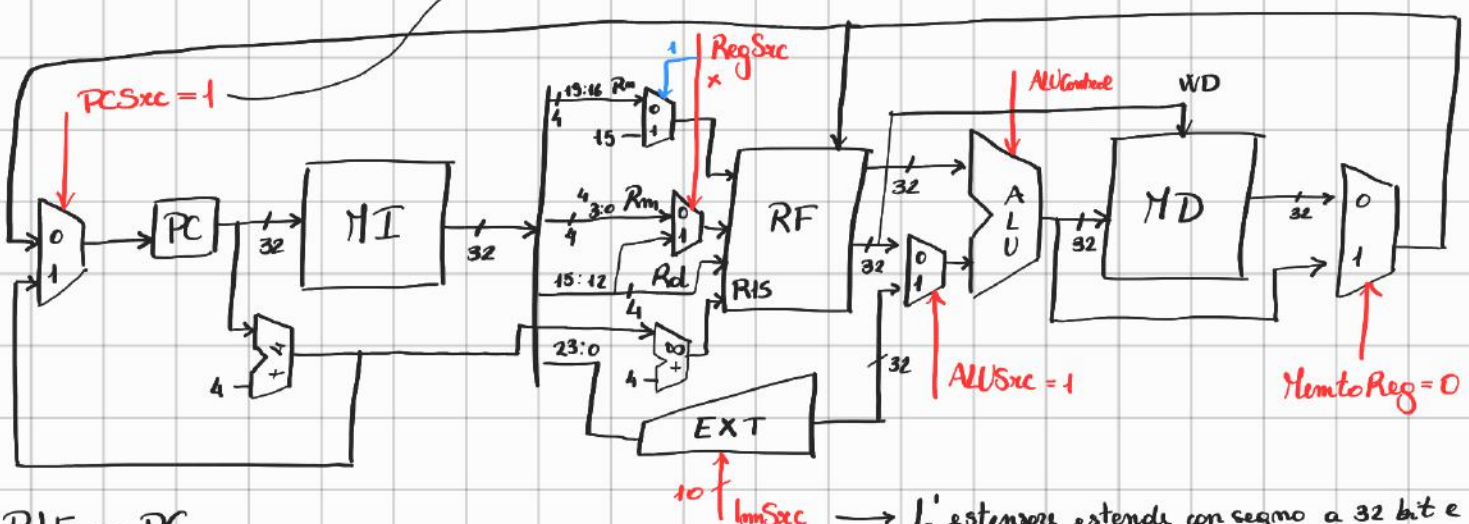
Rd Rm Rm



f2 → Per capire quale rotazione effettuare

5) B "Etichetta"

PCSrc = Branch | (Rd == 15 & regw = 1)

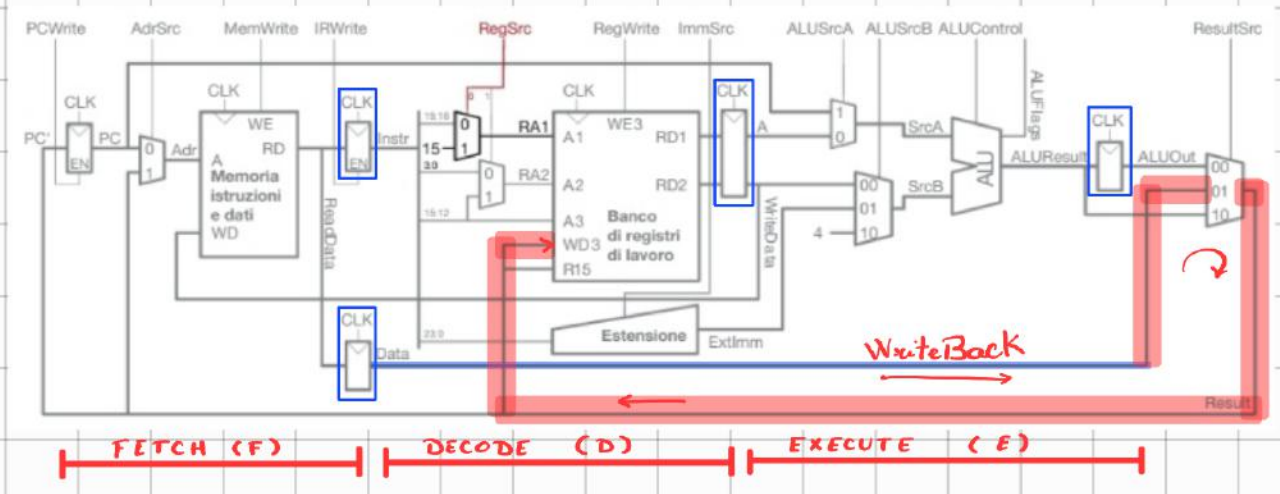


R15 == PC

ImmSrc → L'estensore estende con segno a 32 bit e shifta a sinistra di due posizioni ovvero moltiplica per 4.

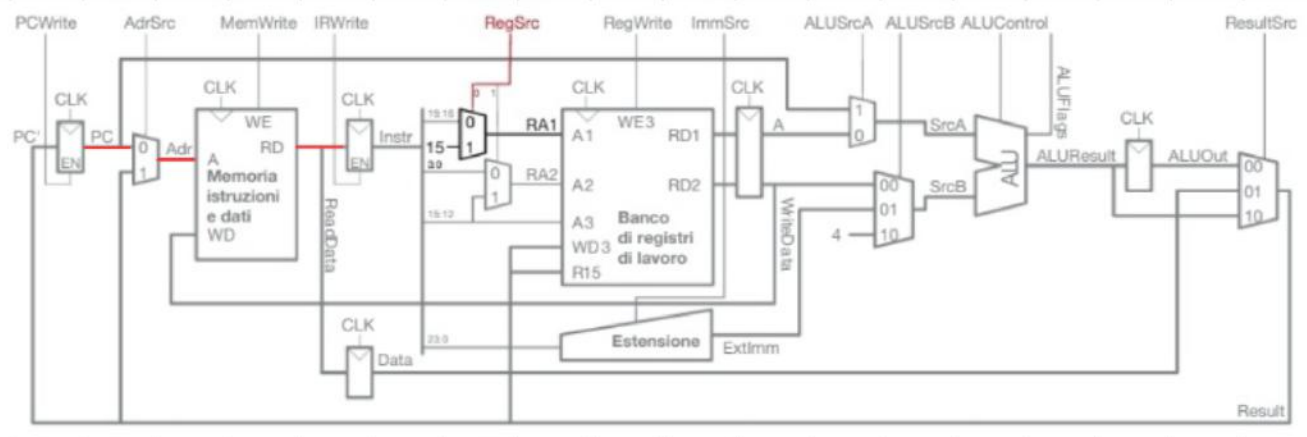
Processore Multi-Cycle

Registri non architetturali: separano le fasi di Fetch, Decode, Execute e WriteBack

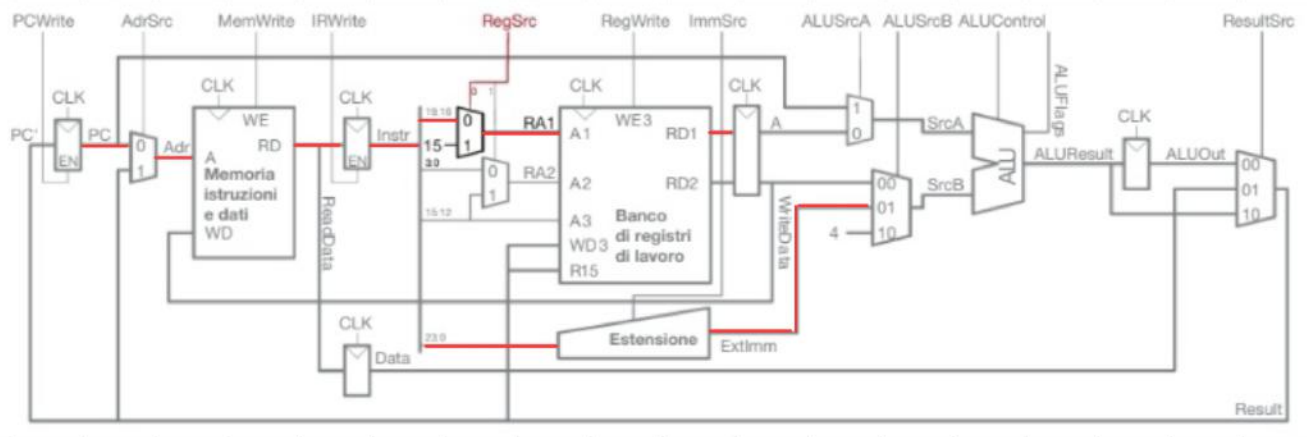


1) LDR R0, [R1, #OFF] 5 cicli

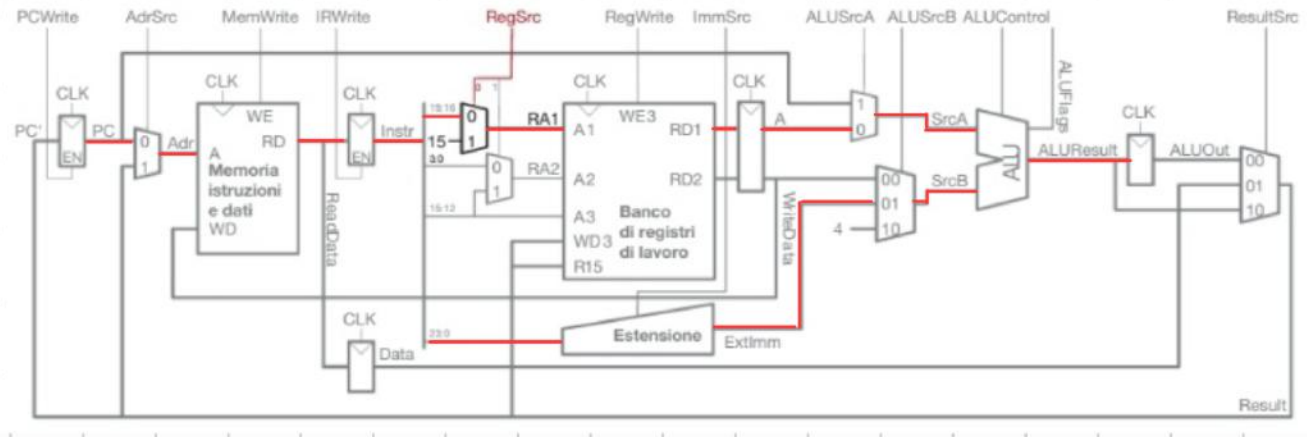
Ciclo 1:



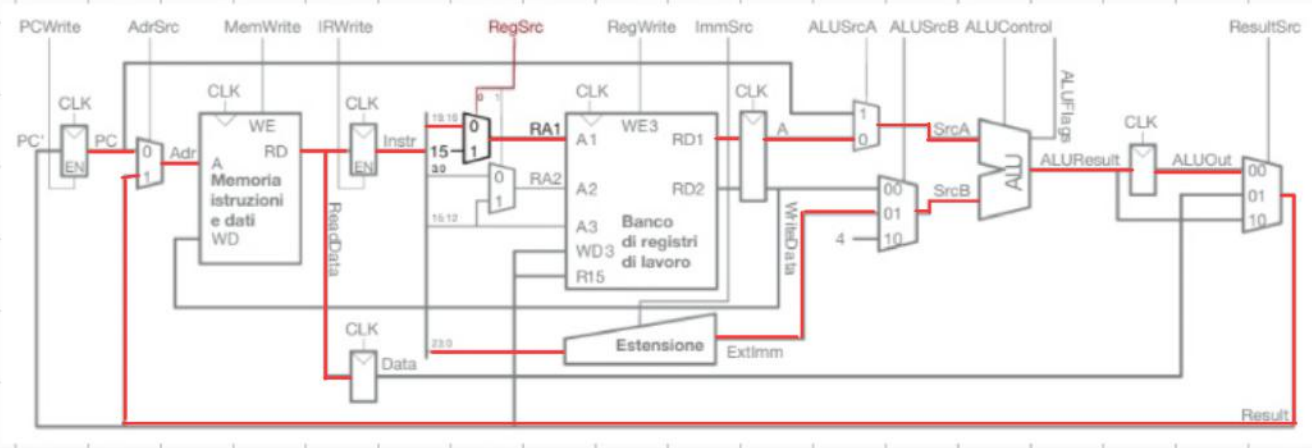
Ciclo 2:



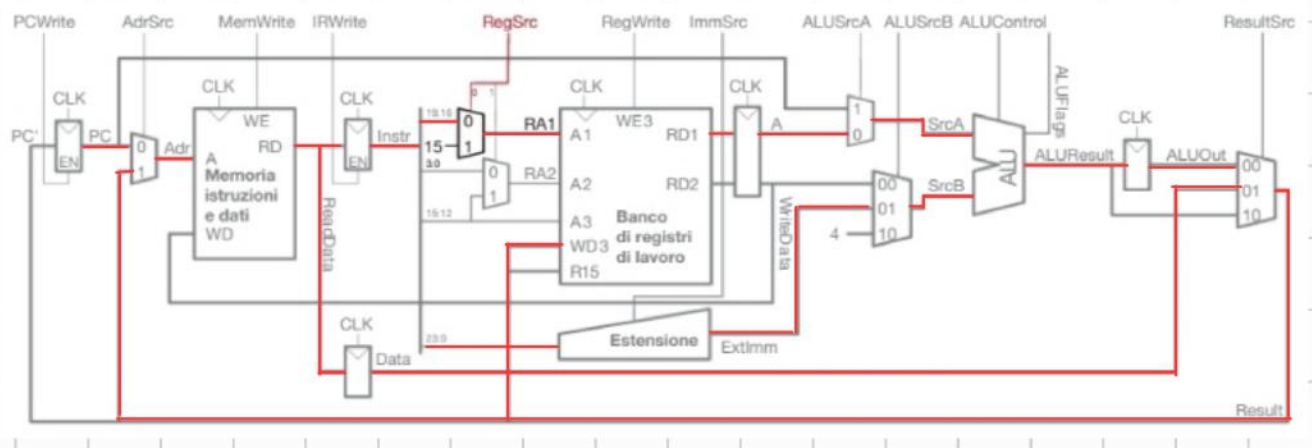
Ciclo 3:



Ciclo 4:

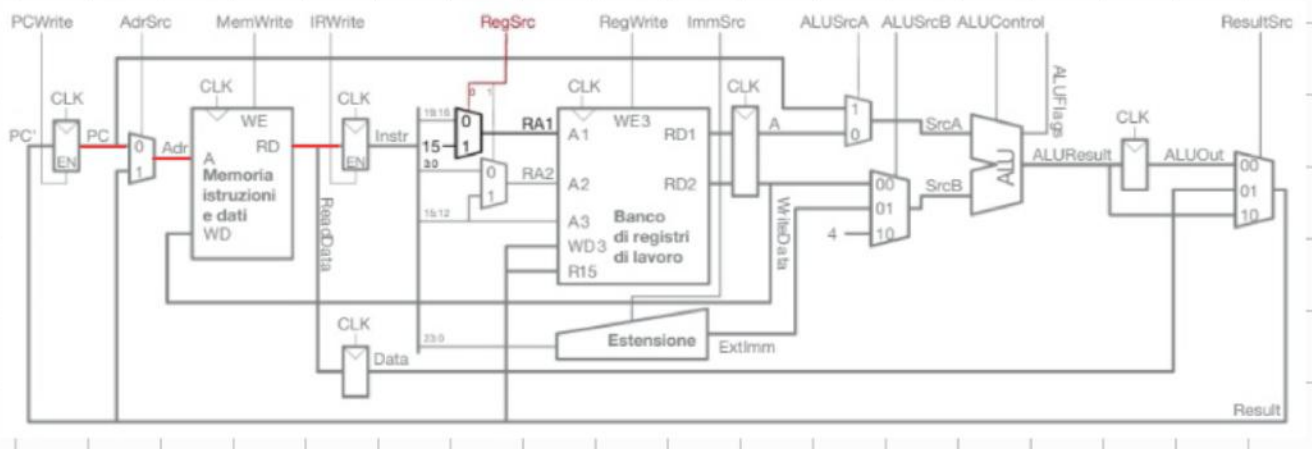


Ciclo 5:

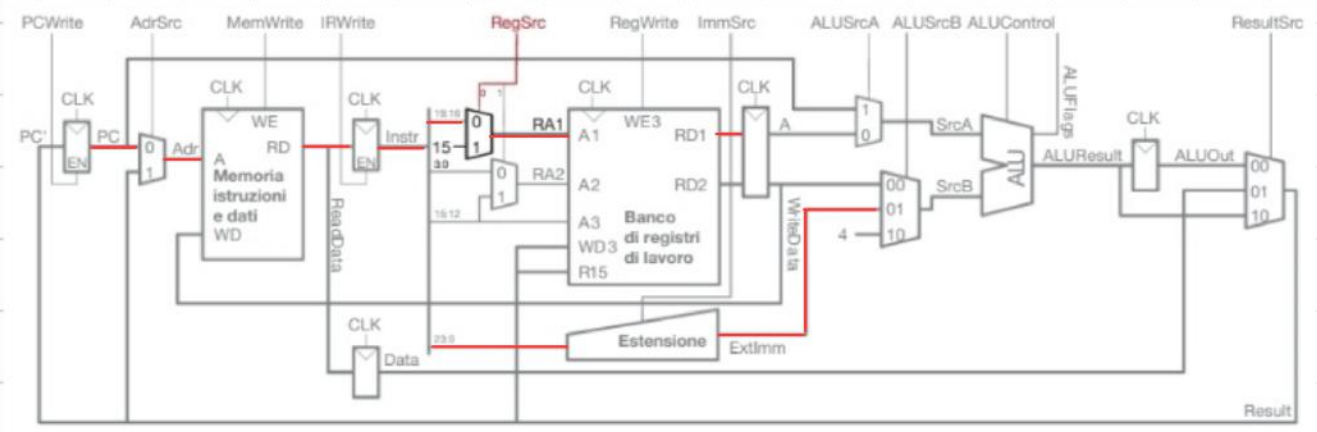


2) ADD R0, R1, #OFF 4 cicli

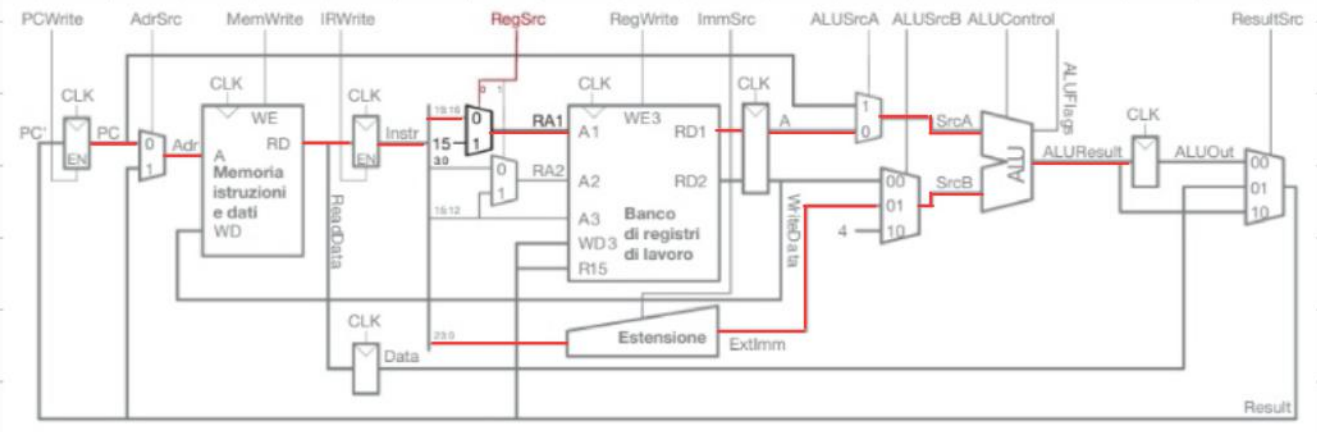
Ciclo 1:



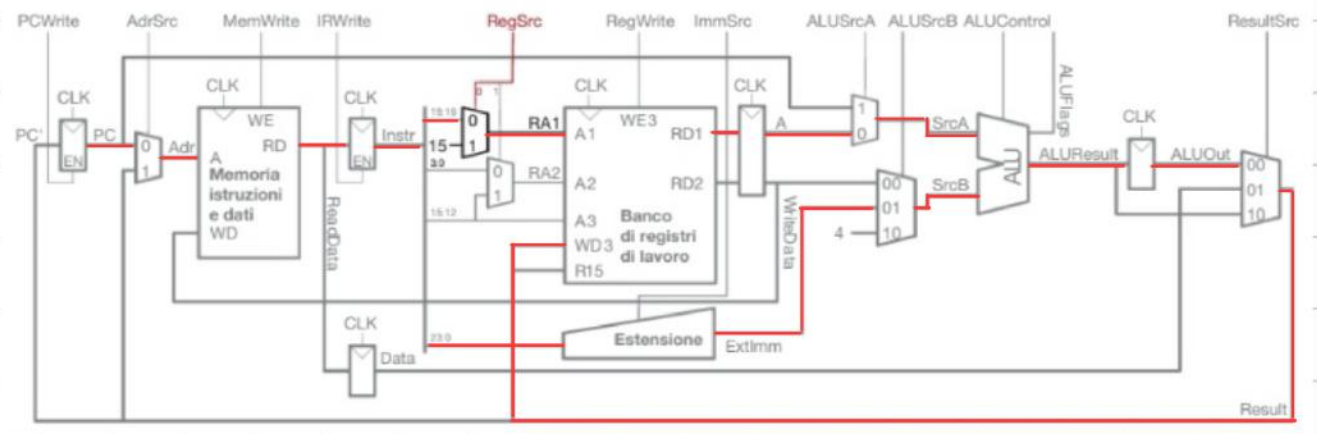
Ciclo 2:



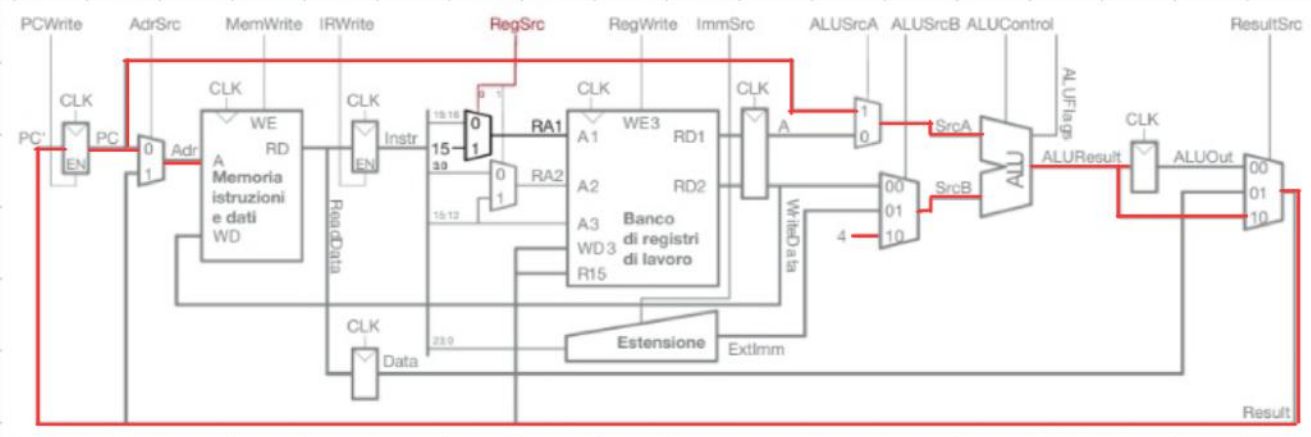
Ciclo 3:



Ciclo 4:

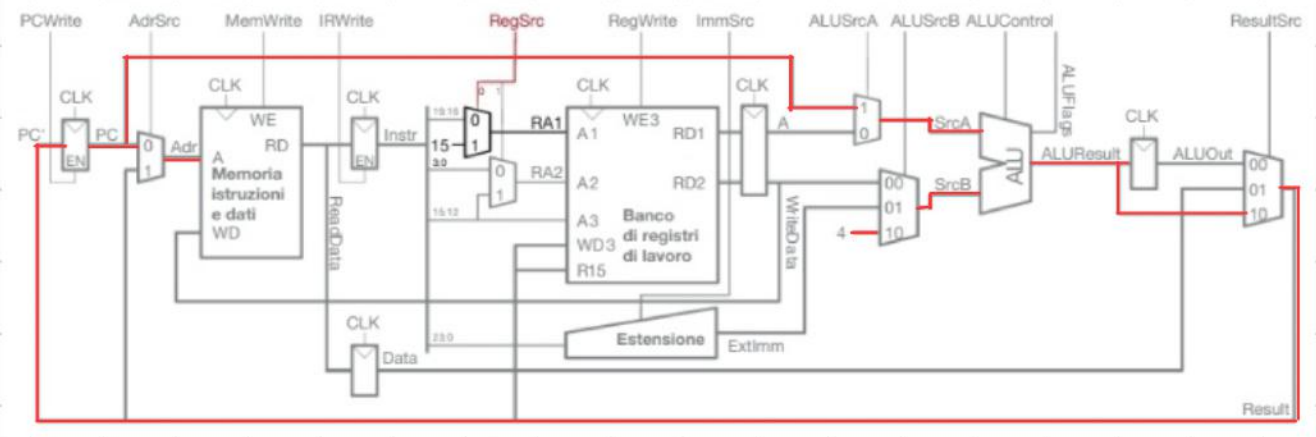


3) PC + 4

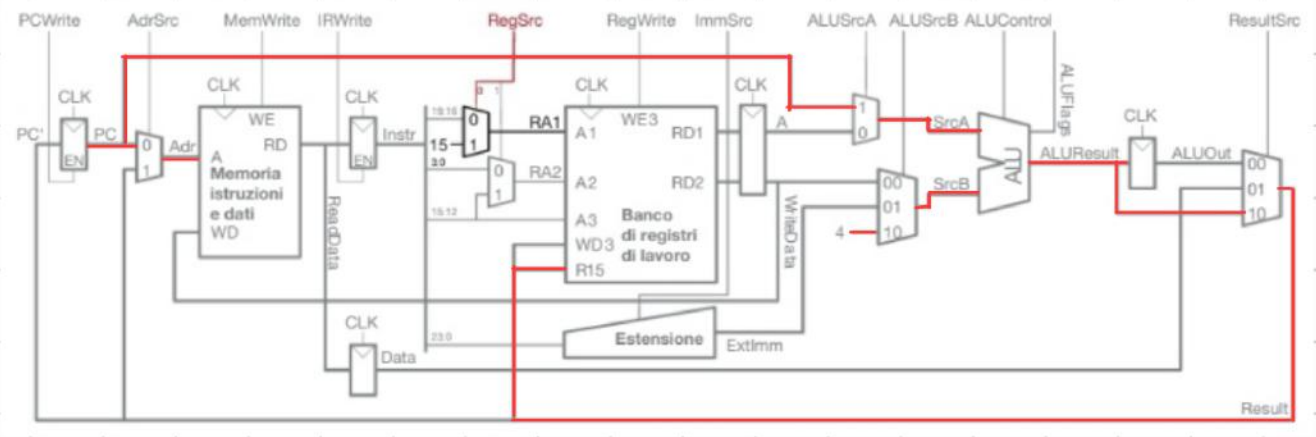


4) (PC + 4) + 4

Ciclo 1:

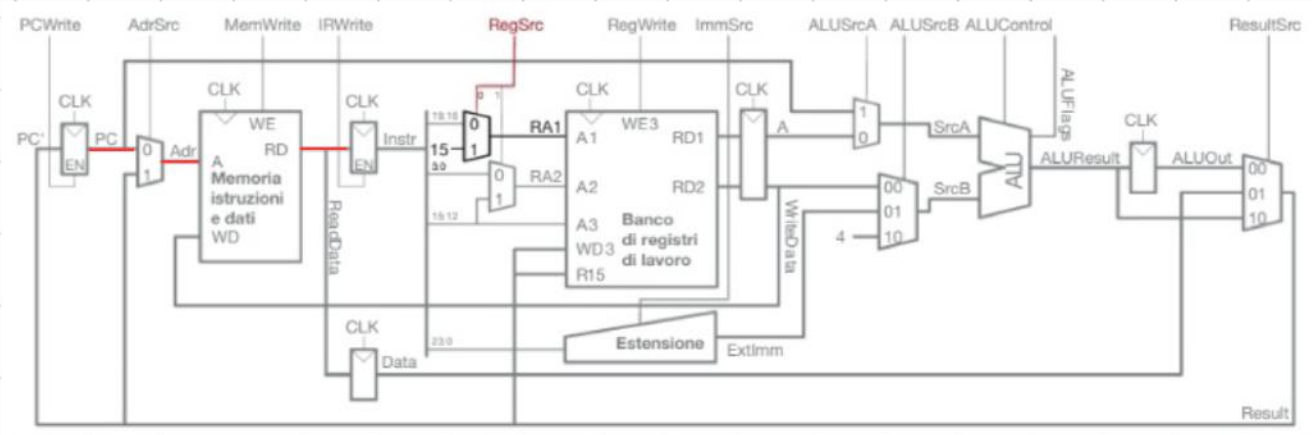


Ciclo 2:

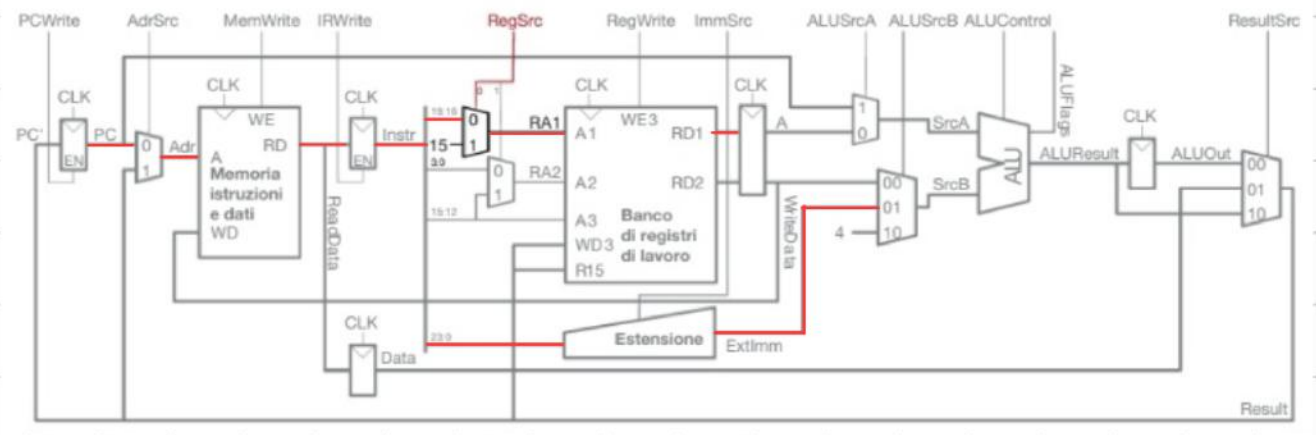


5) STR R0, [R1, #OFF] 4 cicli

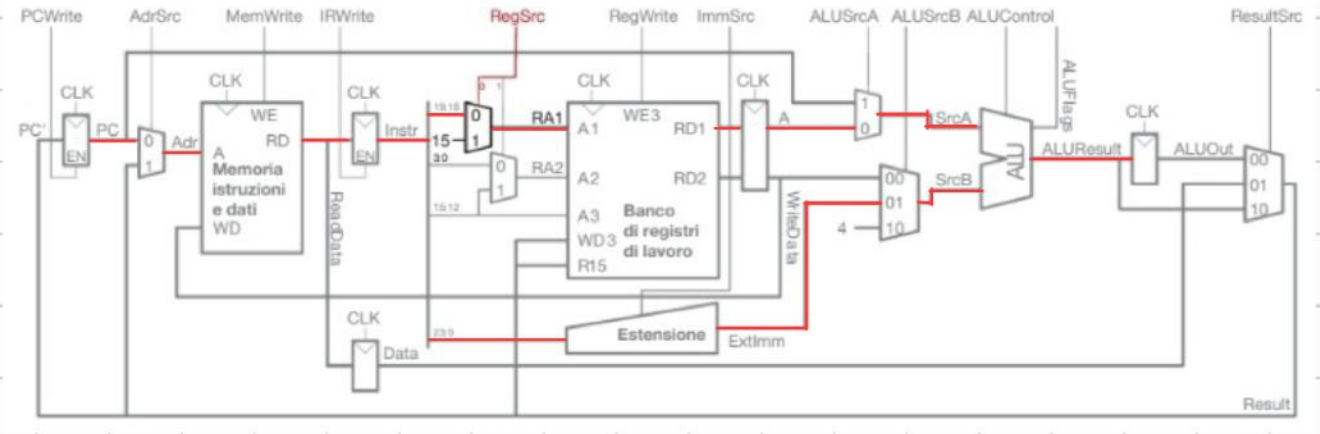
Ciclo 1:



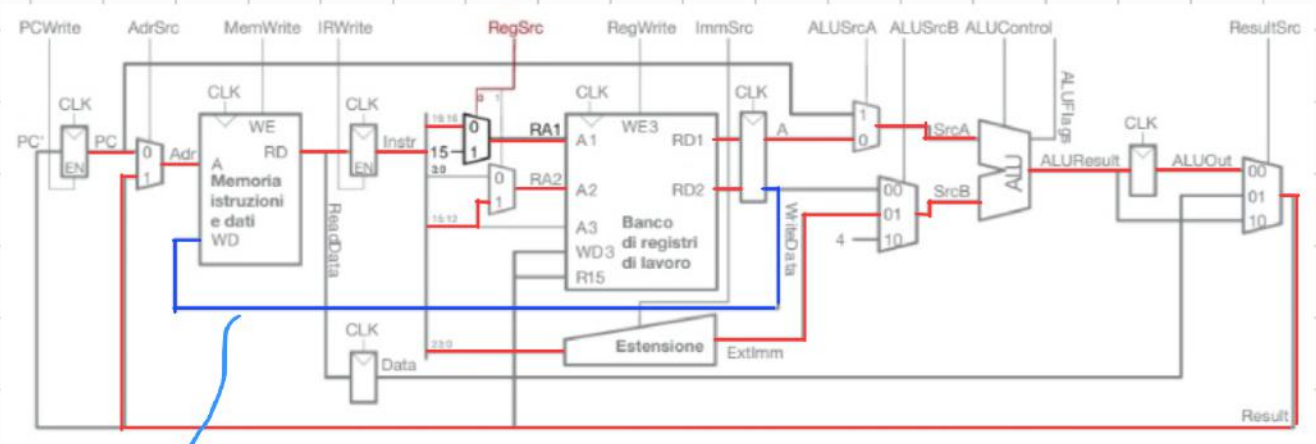
Ciclo 2:



Ciclo 3:



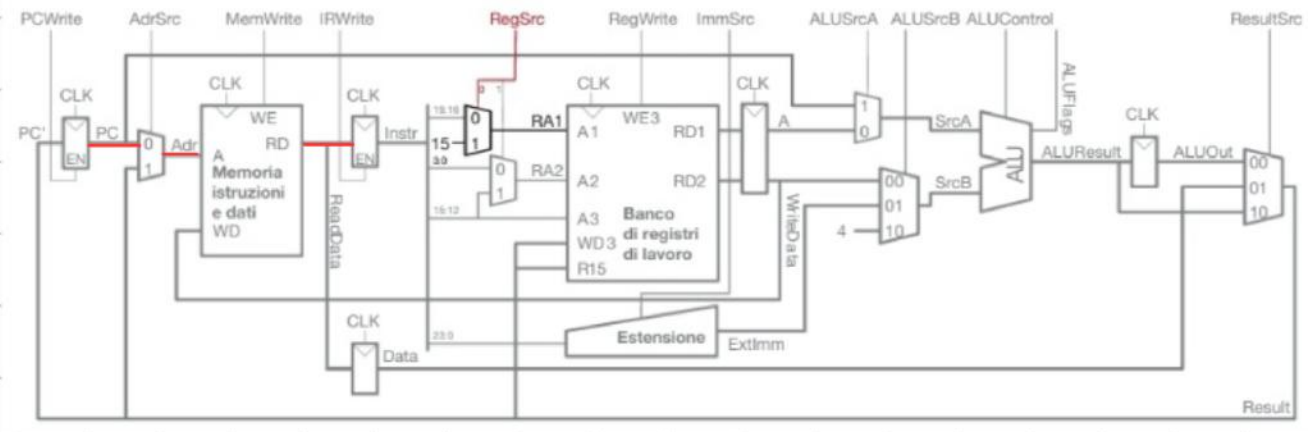
Ciclo 4:



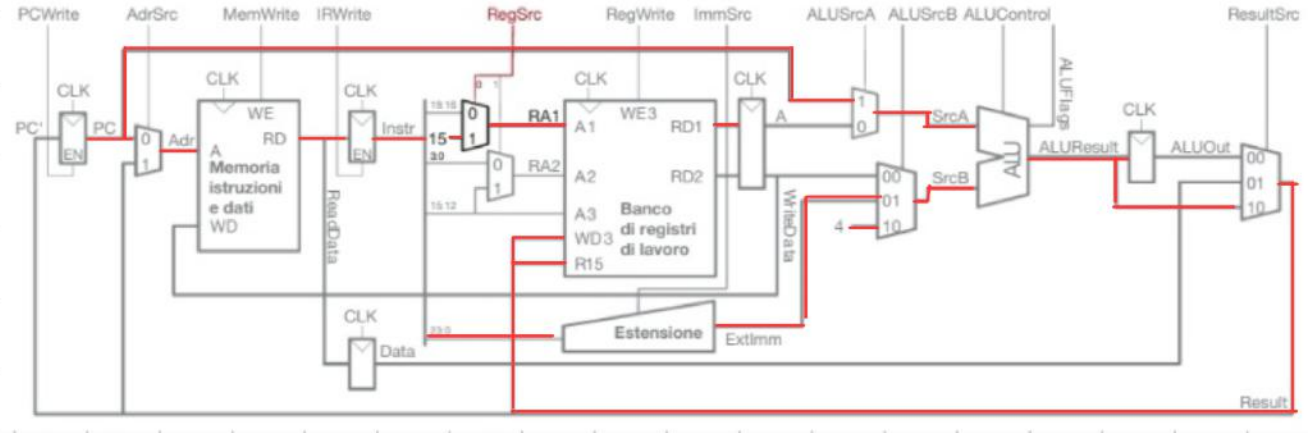
Avviene nella fase di EXECUTE

6) B LOOP 3 cicli

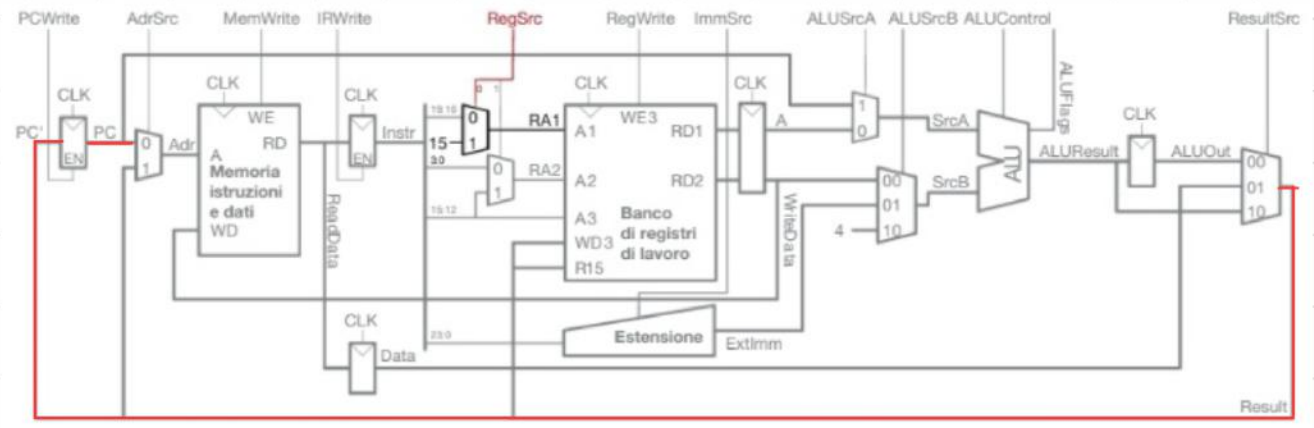
Ciclo 1:



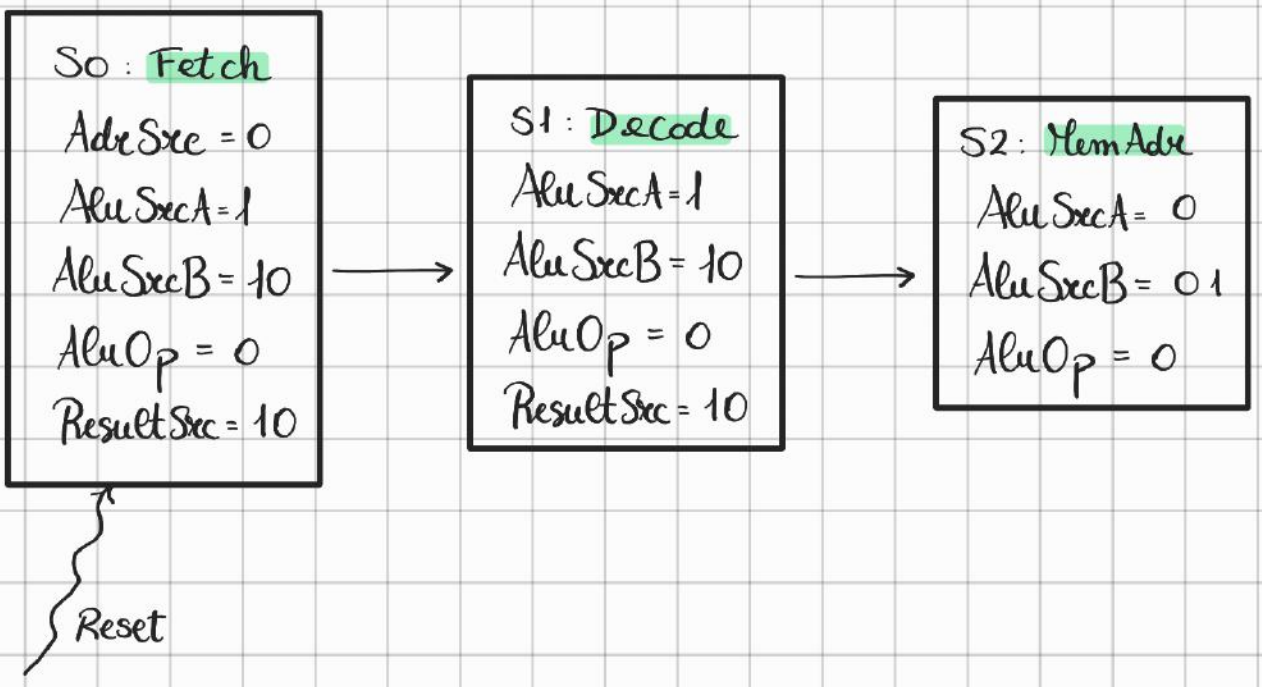
Ciclo 2:



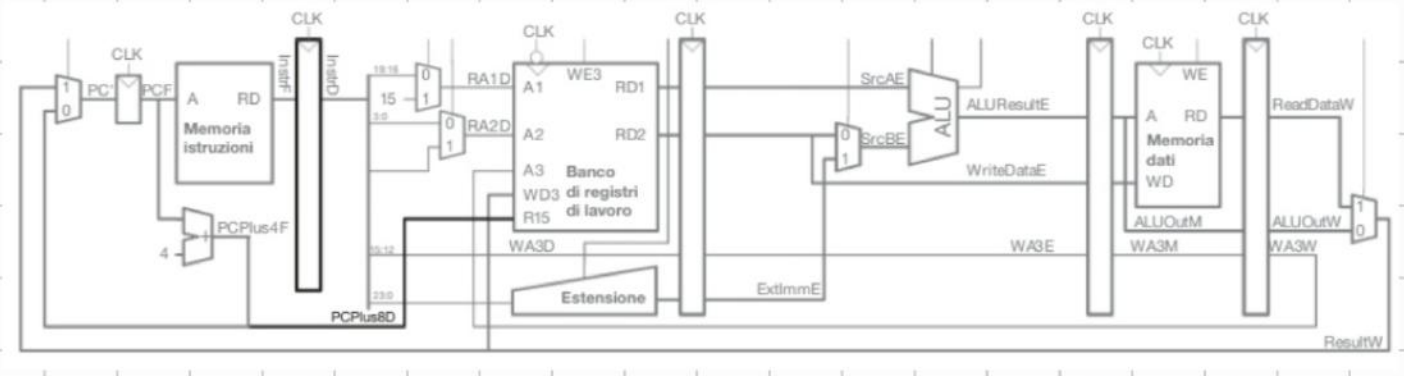
Ciclo 3:



Unità di controllo multi-cycle (FSM)

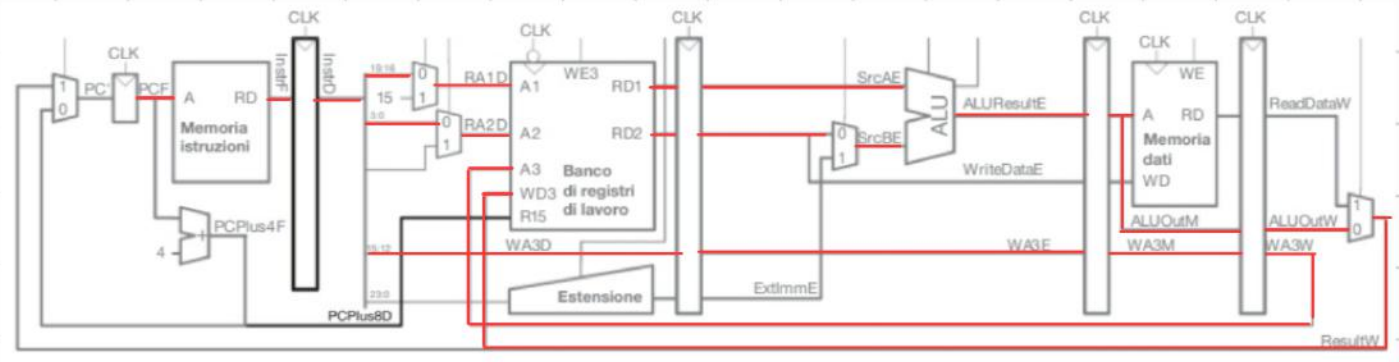


Processore Pipeline

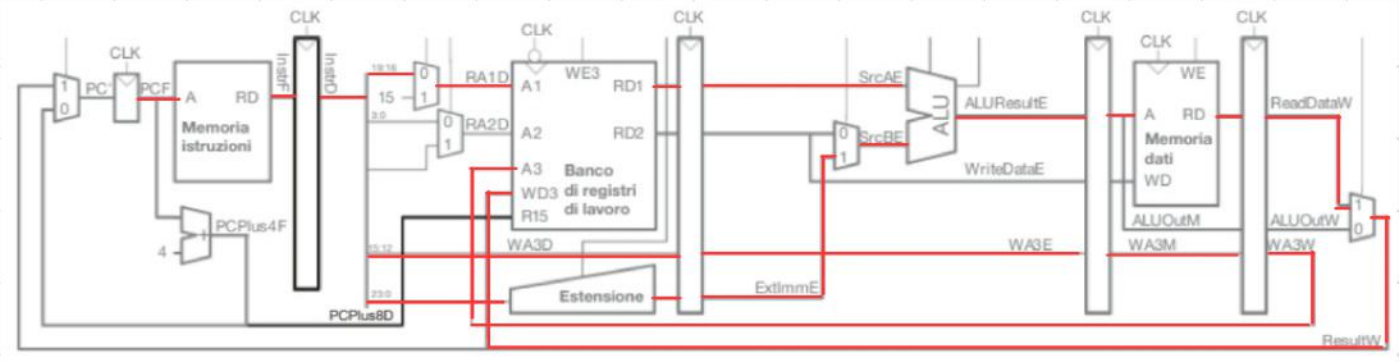


Fetch | Decode | Execute | Memory | Writeback

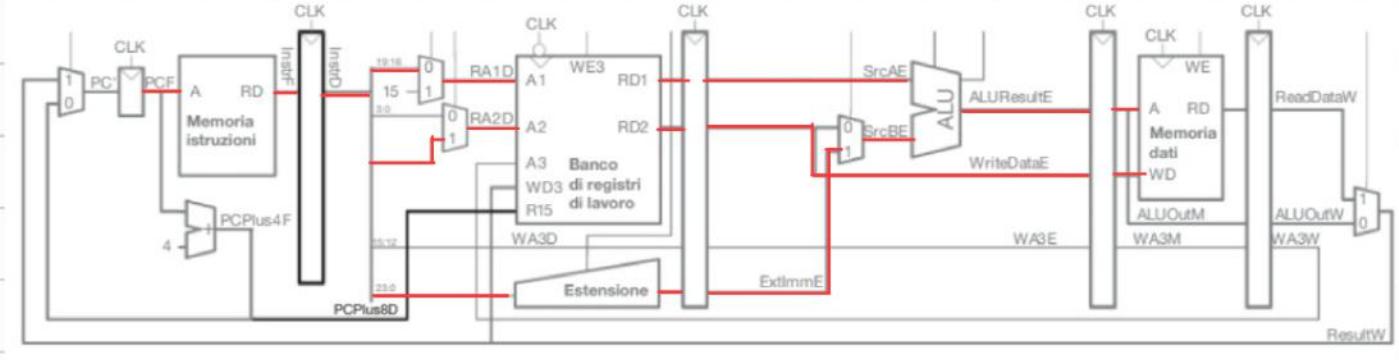
1) ADD R0, R1, R2 5 cicli



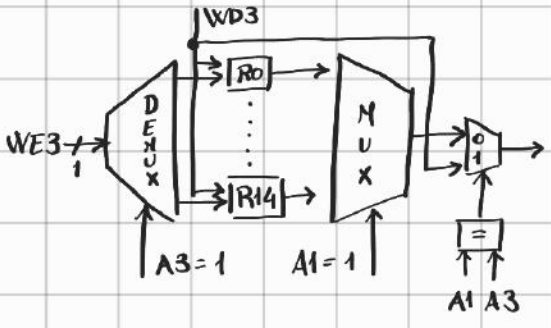
2) LDR R0, [R1, #OFF] 5 cicli



3) STR R0, [R1, #OFF] 4 cicli



Comportamento Register file



Di norma se A1 = A3 (o A2 = A3) si può nello stesso ciclo: leggere il valore corrente di un registro e scrivere un nuovo valore nello stesso registro. Nel processore pipeline conviene modificare il RF per poter scrivere un valore x in un registro e leggere lo stesso valore nello stesso ciclo. In questo modo stiamo aggiungendo un write-to-read forwarding path.

Dipendenze sui dati

F = Fetch
D = Decode
E = Execute
M = Memory
WB = WriteBack

1) Istruzione ≠ da LDR:

- distanza 1: risolte con forwarding M → E
- distanza 2: risolte con forwarding WB → E
- distanza 3: risolte con RF modificato
- distanza > 3: no problem!

2) LDR:

- distanza 1: risolte con stallo + forwarding WB → E
- distanza 2: risolte con forwarding WB → E
- distanza 3: risolte con RF modificato
- distanza > 3: no problem!