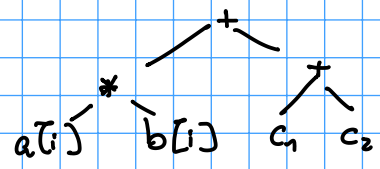
Analisi primo assegnamento

$a[i] = 2[i] * b[i] + c_1 + c_2$



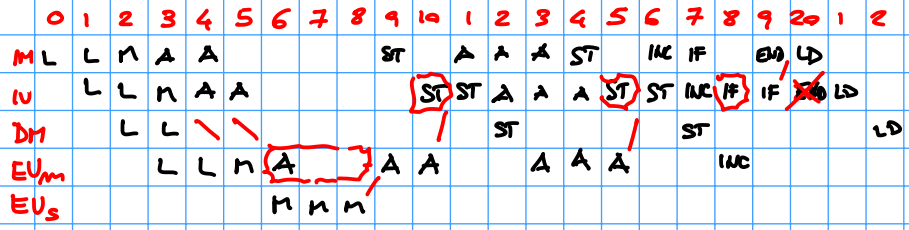
Dipendenze logiche

```
loop: LOAD   Ra, Ri, Rai
      LOAD   Rb, Ri, Rbi
      MUL    Rai, Rbi, Rm      ]
      ADD    Rc1, Rc2, Rcc     ] EU-EU
      ADD    Rcc, Rm, Rain     ]
      STORE  Ra, Ri, Rain      ] IU-EU
      ADD    Rc1, #1, Rc1
      ADD    Rc2, #2, Rc2
      ADD    Rc1, Rc2, Rbin    ]
      STORE  Rb, Ri, Rbin      ] IU-EU
      INC    Ri                ]
      IFL    Ri, RN, loop      ] IU-EU
      END
```

Compilazione standard

```
loop: LOAD   Ra, Ri, Rai       }
      LOAD   Rb, Ri, Rbi       }
      MUL    Rai, Rbi, Rm      }
      ADD    Rc1, Rc2, Rcc     } 1° statement
      ADD    Rcc, Rm, Rain     }
      STORE  Ra, Ri, Rain      }
      ADD    Rc1, #1, Rc1      } 2° statement
      ADD    Rc2, #2, Rc2      } 3° statement
      ADD    Rc1, Rc2, Rbin    }
      STORE  Rb, Ri, Rbin      } 4° statement
      INC    Ri                }
      IFL    Ri, RN, loop      } for
      END
```

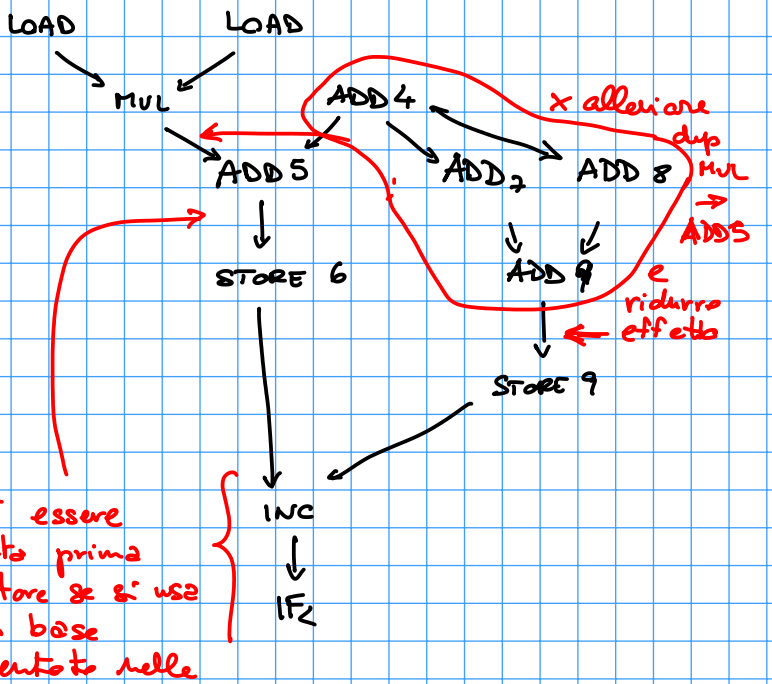
Simulazione dell'esecuzione di 1 iterazione



Tempo di esecuzione: $T = \frac{20}{12} t = \frac{5}{3} t$

Analisi Data flow del codice

```
loop: 1  LOAD   Ra, Ri, Rai
      2  LOAD   Rb, Ri, Rbi
      3  MUL    Rai, Rbi, Rm
      4  ADD    Rc1, Rc2, Rcc
      5  ADD    Rcc, Rm, Rain
      6  STORE  Ra, Ri, Rain
      7  ADD    Rc1, #1, Rc1
      8  ADD    Rc2, #2, Rc2
      9  ADD    Rc1, Rc2, Rbin
      10 STORE  Rb, Ri, Rbin
      11 INC    Ri
      12 IFL    Ri, RN, loop
      13 END
```



Codice ottimizzato

```
LOAD   Ra, Ri, Rai
LOAD   Rb, Ri, Rbi
MUL    Rai, Rbi, Rm
ADD    Rc1, Rc2, Rcc
ADD    Rc1, #1, Rc1
ADD    Rc2, #2, Rc2
a) ADD    Rc1, Rc2, Rbin
b) ADD    Rcc, Rm, Rain
c) INC    Ri
   STORE  Ra', Ri, Rain
   IF<    Ri, RN, loop, delayed
   STORE  Rb, Ri, Rbin
```

Simulazione di una iterazione

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IM	L	L	M	A	A	A	A	A	I	ST	IF	ST	L			
IU		L	L	M	A	A	A	A	A	I	ST	ST	IF	ST	L	
DM			L	L								ST		ST	L	
EUm				L	L	M	A	A	A	A	A	I				L
EU3							M	M	M							

simulazione

$T = \frac{13t}{12}$

Possiamo anticipare b) prima di a) e c) prima di b)

```
....
ADD    Rc2, #2, Rc2
ADD    Rcc, Rm, Rain
INC    Ri
ADD    Rc1, Rc2, Rbin
STORE  ...
IF     delayed
STORE  ....
```

Simulazione

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
IM	L	L	M	A	A	A	A	I	A	ST	IF	ST	L				
IU		L	L	M	A	A	A	A	I	A	ST	IF	ST	L			
DM			L	L								ST	ST		L		
EUm				L	L	M	A	A	A	A	I	A				L	
EU3							M	M	M								

Guadagno delle predizioni

(su tempo ideale)

$$\frac{\frac{5}{3}t}{t} = \frac{5}{3} \simeq 1.6 \times$$

$T = t$

Analisi del WORKING SET

	LOCALITÀ	RIUSO
pagine del codice	✓	✓
1 pagina × A	✓	–
1 pagina × B	✓	–

anche con una cache con solo 4 pagine disponibili è sufficiente e garantisce l'insorgere dei soli fault fisiologici

dunque $N_{fault} = \frac{N}{\sigma} + \frac{N}{\sigma} + 1 \simeq \frac{2N}{\sigma}$

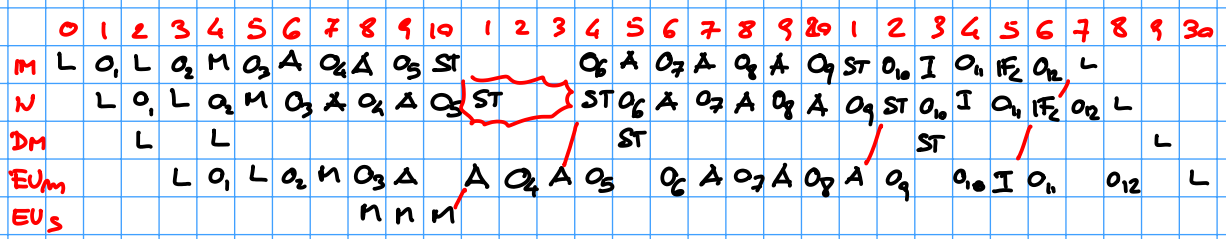
A B codice

(soli fault fisiologici)

Esecuzione multithreaded (interleaving a 2 vie)

Codice "standard"

```
loop:  LOAD   Ra, Ri, Rai
       LOAD   Rb, Ri, Rbi
       MUL    Rai, Rbi, Rm
       ADD    Rc1, Rc2, Rcc
       ADD    Rcc, Rm, Rain
       STORE  Ra, Ri, Rain
       ADD    Rc1, #1, Rc1
       ADD    Rc2, #2, Rc2
       ADD    Rc1, Rc2, Rbin
       STORE  Rb, Ri, Rbin
       INC    Ri
       IF2    Ri, RN, loop
       END
```



$$T = \frac{27\,t}{(12+12)} = \frac{27}{24}\,t$$

molto migliore del $\frac{5}{3}t$ iniziale

maggior parte delle bolle "coperte" dall'interleaving

Per finire:

T_c del codice ottimizzato

N iterazioni × 12 istruzioni × $1t$ + #fault * T_{transf}

$$= 12Nt + \frac{2N}{\sigma} T_{transf}$$

T_{transf} dipende da hw di sottosistema di M

Hp: cache L1 set assoc con numero $\sigma = 64$

M interallacciata $m = 8$

$$T_{transf} = 2T_{tr} + \frac{\sigma}{m}\tau_M + m\tau = 2T_{tr} + 8\tau_M + 8\tau$$