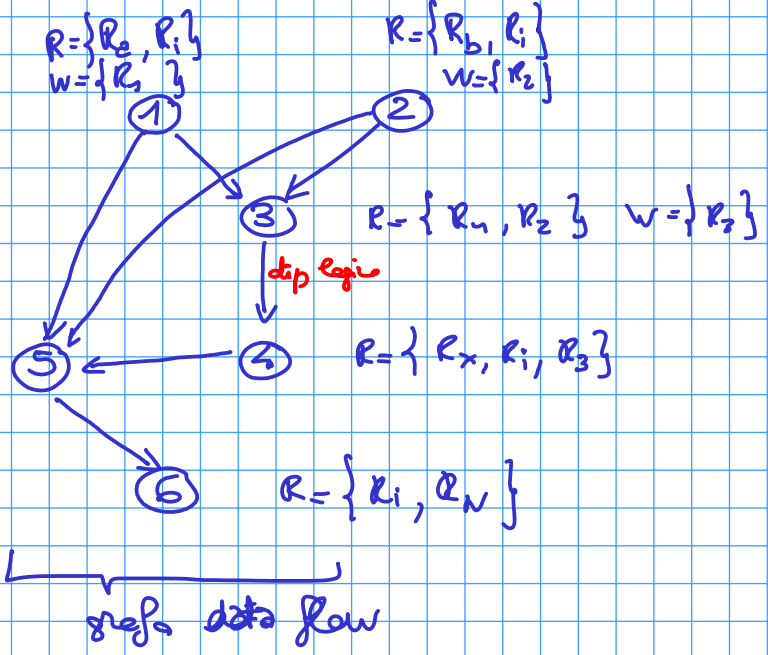


$$x[i] = a[i] + b[i]$$

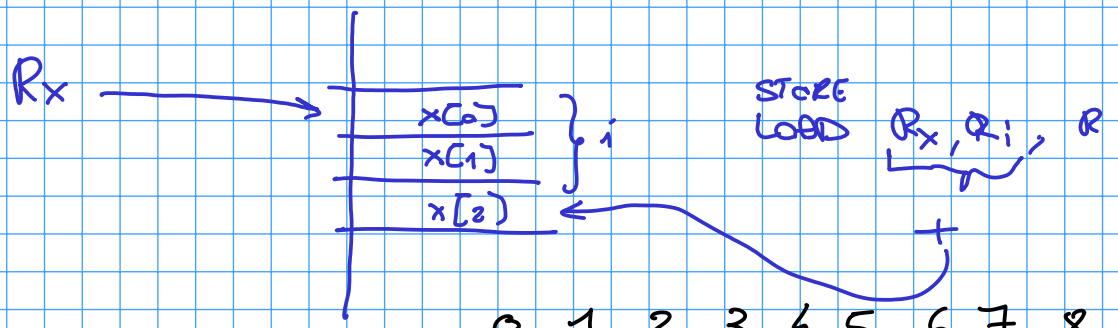
- loop: ① LOAD  $R_2, R_i, R_1$   $a[i]$   
 ② LOAD  $R_b, R_i, R_2$   $b[i]$   
 ③ ADD  $R_1, R_2, R_3$   
 ④ STORE  $R_x, R_i, R_3$   
 ⑤ INC  $R_i$   
 ⑥ IF $_c$   $R_i, R_N, loop$



- ②  
①  
③  
④  
⑤  
⑥

Calcola la stessa simultanea

### Utilizzare un Registro base modificato (decrementato di 1)



- loop: ① LOAD  $R_2, R_i, R_1$   
 ② LOAD  $R_b, R_i, R_2$   
 ③ ADD  $R_1, R_2, R_3$   
 ④ STORE  $R_x, R_i, R_3$   
 ⑤ INC  $R_i$   
 ⑥ IF $_c$   $R_i, R_N, loop$
- 6t  $E = \frac{6}{10}$

	0	1	2	3	4	5	6	7	8	9	10	11
in	L	L	A	S			I	IF $_c$			L	
IU		L	L	A	S		S	I	IF $_c$	IF $_c$	X	L
DM			L	L				S				L
EU				L	L	A			I			L

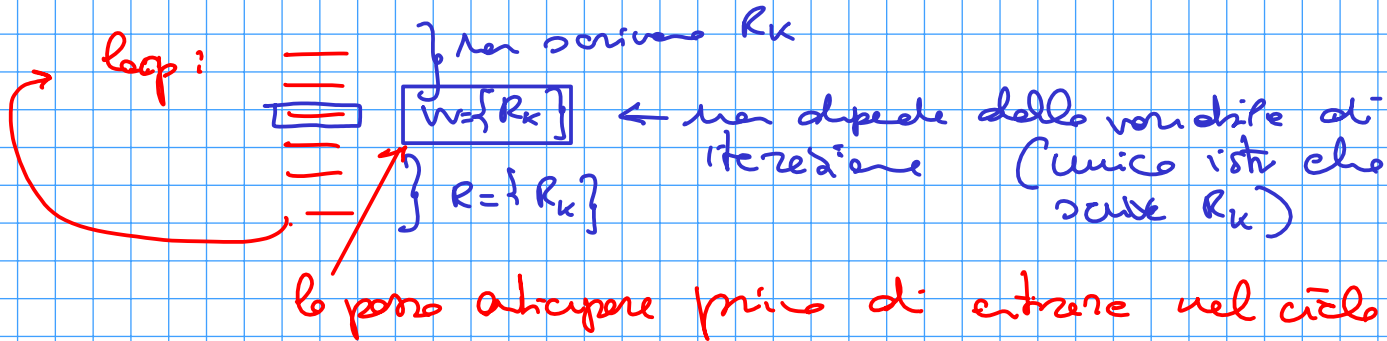
10t

- loop: LOAD  
 LOAD  
 ADD  
 INC  $R_i$   
 IF $_c$   $R_i, R_N, loop$ , delayed  
 STORE  $R_x, R_i, R_3$
- 6t  $E = \frac{6}{8} = \frac{3}{4}$

	0	1	2	3	4	5	6	7	8	9	10	11
	L	L	A	I	IF $_c$			ST	L			
		L	L	A	I	IF $_c$		IF $_c$	ST	L		
			L	L						ST	L	
				L	L	A	I					

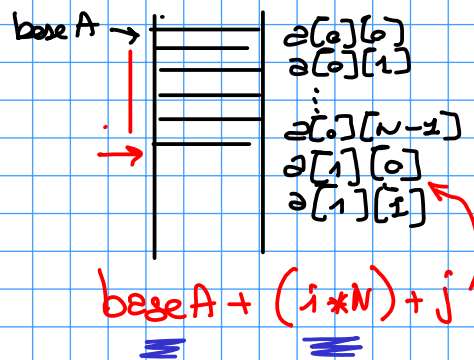
8t

# Rimozione degli INVARIANTI



```

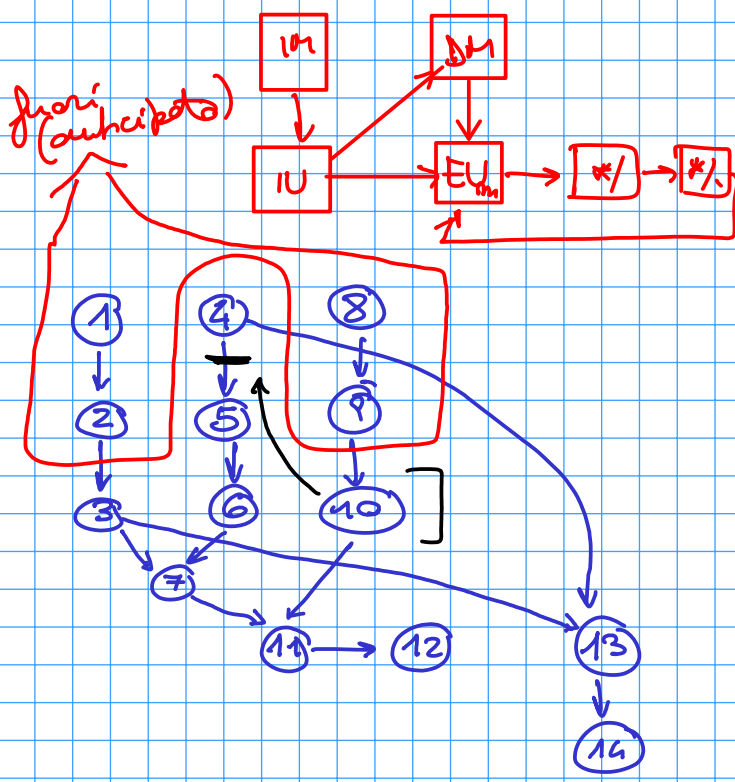
for(i ..... )
  for(j ..... )
    c[i][j] = 0
    [
      for(k ..... )
        c[i][j] += a[i][k] * b[k][j];
    ]
  
```



- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14

```

loop:
  MUL R1, RN, Rai
  ADD Rai, RbaseA, Rai
  LOAD Rai, Rk, R2
  MUL Rk, RN, Rbk
  ADD Rbk, RbaseB, Rbk
  LOAD Rbk, Rj, R2
  MUL R1, R2, R3
  MUL R1, RN, Rci
  ADD Rci, RbaseC, Rci
  LOAD Rci, Rj, R4
  ADD R4, R3, R4
  STORE Rci, Rj, R4
  INC Rk
  IF< Rk, RN, loop
  
```



① Loop: MUL  $R_i, R_N, R_{ai}$   
 ② ADD  $R_{ai}, R_{baseA}, R_{ai}$  ← EU-EU  
 ③ LOAD  $R_{ai}, R_K, R_2$  ← U-EU  
 ④ MUL  $R_K, R_N, R_{bx}$   
 ⑤ ADD  $R_{bx}, R_{baseB}, R_{bx}$  ←  
 ⑥ LOAD  $R_{bx}, R_j, R_2$   
 ⑦ MUL  $R_1, R_2, R_3$   
 ⑧ MUL  $R_i, R_N, R_{ci}$   
 ⑨ ADD  $R_{ci}, R_{baseC}, R_{ci}$  ←  
 ⑩ LOAD  $R_{ci}, R_j, R_4$   
 ⑪ ADD  $R_4, R_3, R_4$   
 ⑫ STORE  $R_{ci}, R_j, R_4$   
 ⑬ INC  $R_K$   
 ⑭ IFC  $R_K, R_N, loop$

	0	1	2	3	4	5	6	7	8	9	10				
IM	L	M	A	L	M						M				
IV		L	M	A	L						L	M			
DM			L								L				
EU			L	M	A				A		L	M			
					M	M	M	M				M	M	M	M

MUL  $R_K, R_N, R_{bx}$   
 LOAD  $R_{ai}, R_K, R_1$   
 LOAD  $R_{ci}, R_j, R_4$   
 ADD  $R_{bx}, R_{baseB}, R_{bx}$   
 LOAD  $R_{bx}, R_j, R_2$

⑦ ⑪ ⑫ ⑬ ⑭

	0	1	2	3	4	5	6	7	8
IM	M	L	L	A	L				
IV		M	L	L	A	L	L	L	L
DM			L	L					
EU		M	L	L	A	A			
EU <sub>7</sub>			M	M	M	M			

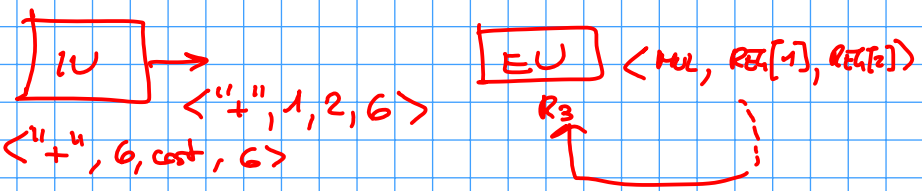
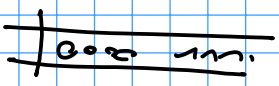
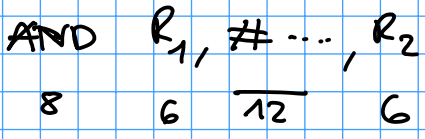
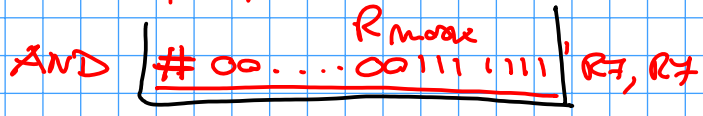
Si consideri il seguente frammento di codice, che lavora sui vettori di indice A, B, C e D, ciascuno da N=8K posizioni:

```

for(i=0; i<N; i++) {
    a[i]=a[i]+b[i]*c[i];
    b[i]=b[i]+c[i]+cost;
    j=b[i] mod 128;
    d[j]=d[j]+cost;
}
    
```

Si supponga che il risultato della compilazione venga eseguito su un processore D-RISC pipeline con EU parallela (EU slave che esegue moltiplicazione o divisione in 4t). La gerarchia di memoria comprende una cache di primo livello con linee (blocchi) da  $\sigma=64$  parole e la memoria centrale interallacciata (m moduli).

Si chiede di



$$T_{id} = kt$$

```

PREFETCH
for(i=0; i<N; i++) {
    a[i]=a[i]+b[i]*c[i];
    b[i]=b[i]+c[i]+cost;
    j=b[i] mod 128;
    d[j]=d[j]+cost;
}
NON DEALLOCARE
    
```

WORKING SET

codice	loc	RUSO
	✓	✓
A	✓	—
B	✓	—
C	✓	—
D	—	✓

Tutto  
 } 1 linea di cache  
 x A, B, C  
 Tutto

```

loop : LOAD R3, R1, R1
      LOAD R4, R1, R2
      MUL R3, R2, R3
      LOAD R5, R1, R1
      ADD R3, R4, R5
      STORE R3, R1, R5
      ADD R1, R2, R6
      ADD R6, Rcost, R6
      STORE R6, R1, R6
      MOD R6, #128, R7
      LOAD R7, R7, R8
      ADD R8, Rcost, R8
      STORE R8, R1, R8
      INC R1
      IFZ R1, RN, loop
    
```