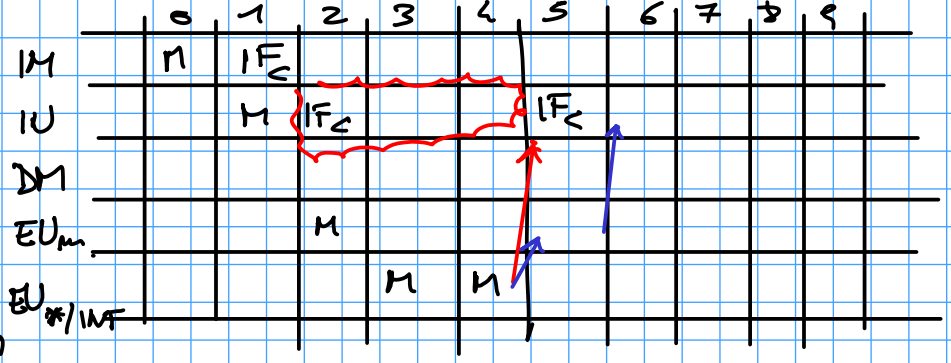
MUL R_3, R_5, R_6

IF< R_6, R_N, loop



2t

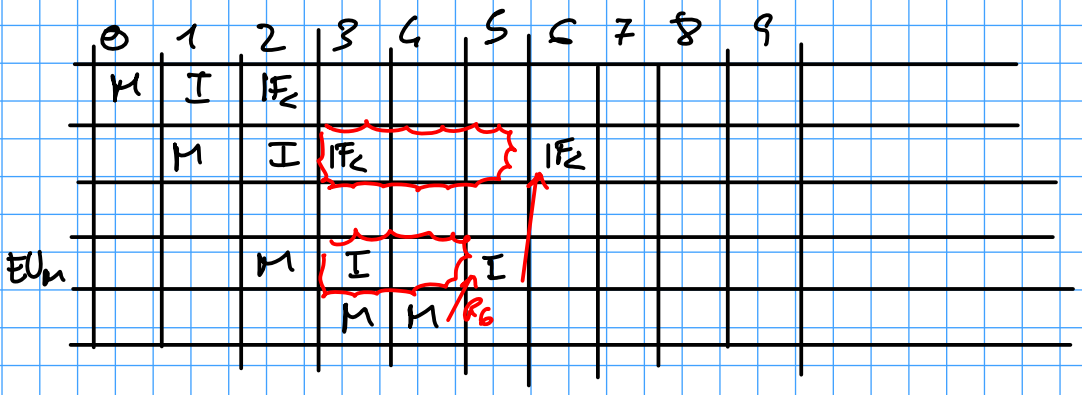
dip EU-EU

MUL R_3 R_5 R_6

INC R_6

dip IU-EU

IF< R_6, R_N, loop

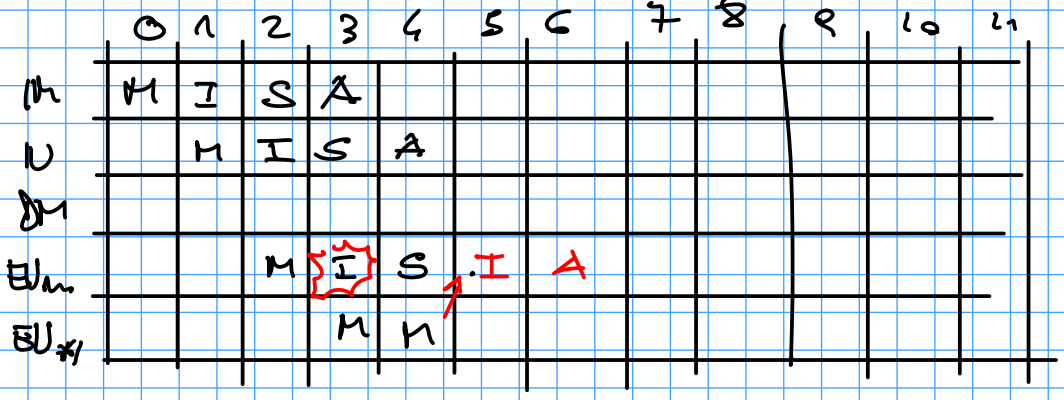


MUL R_3, R_5, R_6

INC R_6

SUB R_1, R_2, R_3

ADD R_8, R_9, R_{10}



Ottimizzazione del codice D-RISC

- processore monolitico
  - LOAD ... , PREFETCH / NON DEALLOCARE
- processore pipeline
  - salti
    - inlining
    - loop unrolling
    - delayed branche
  - dip logiche
    - riordino delle istruzioni D-RISC

} cause di degrado delle prestazioni

INLINING

↳ eliminare i salti relativi alla chiamata di procedure / funzione

R_9 → int inc(int x) { return (x+1); }
R_9 ↑

{ $R_{ret} = R_{10}$

inc: INC R_9
GOTO R_{10}
XXX

{
y[i] = inc(z);

{ ADD R_z, R_0, R_9
CALL R_{inc}, R_{10}
STORE R_y, R_i, R_9

Sistruzioni 7t

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IM	A	C	St	INC	G	XXX	ST									
IU		A	C	~~X~~	INC	G	~~XXX~~	ST								
DM									ST							
EU			A			INC										

inline int inc (int x) { ... }
{
y(i) = inc(z);

{ y[i] = ~~(x=z)~~ z+1;
~~(x+1)~~

1 { INC R_z
→ STORE R_y, R_i, R_z

3t

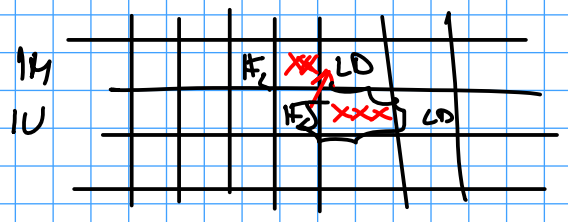
I	ST				
	I	ST	ST		
		I			

LOOP UNROLLING

```
for (i=0; i<N; i++)
    x[i] = a[i] + b[i];
```

loop: { LOAD, LOAD, ADD, STORE } $\in$ (corpo)

$\in$(for) { INC, IF< ... loop, xxx



(N è pari)

```
for (i=0; i<N; i+=2) {
    corpo (i)      x[i] = a[i] + b[i];
    corpo (i+1)    x[i+1] = a[i+1] + b[i+1];
}
```

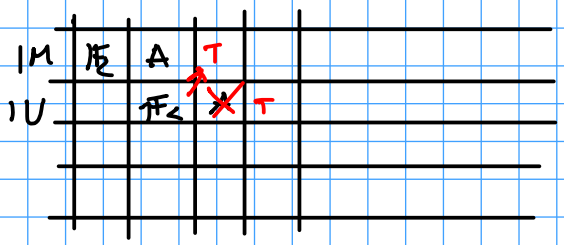
2

fattore di unrolling

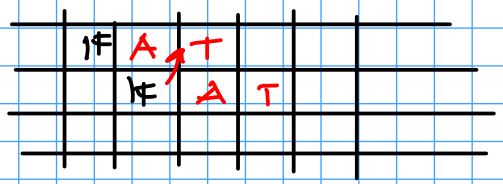
DELAYED BRANCH

delay slot { IF< R_i, R_N, loop, "delayed"
ADD R_1 R_2 R_3 } se il salto è preso esegui comunque la ADD e poi salta

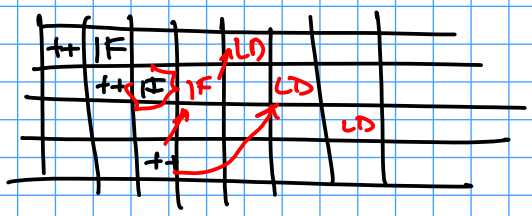
no delayed



con delayed



```
for( i . . . . )
    x[i] = a[i] + b[i] ;
```




```
loop:  LOAD   Re, Ri, R1      ①
       LOAD   Rb, Ri, R2      ②
       ADD    R1, R2, R3      ③
R(R3)  STORE  Rc, Ri, R3      ④
       INC    Ri              ⑤
       IF<     Ri, RN, loop , delayed
```

ADD — R(R1, R2) — W(R3)

STORE ④
IF< Ri, RN+1, loop , delayed
INC Ri

→ LOAD Re, Ri, R1
loop: LOAD ←
~~ADD~~
STORE
INC Ri
k=2 IF< Ri, RN delayed ←
→ LOAD Ra, Ri, R1