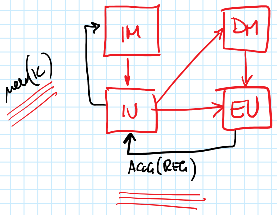
PROC "PIPELINE D-RISC"

martedì 6 dicembre 2016 11:14



DIPENDENZA LOGICA

istruzione I induce una dip logica su istruzione J quando I produce x che è letto da J e I produce x in U_A e J legge x in U_B

① la 1° cond. di Bernstein

② avviene in unità diverse

1) decodifico LOAD/STORE in IU utilizzando un registro che poi può essere modificato nella EU

```
....
INC  Ri              (EU)   W(Ri)
LOAD Rbase, Ri, R1   (IU)   R(Ri)
```

② ①

la INC induce una dip logica nella LOAD

IU cerca di leggere Ri (Cont = 1)

t	0	1	2	3	4	5	6	7	8	9
IM	INC	LOAD								
IU		INC	LOAD	LOAD						
DM				Ri	LOAD					
EU			INC			LOAD				

cont Ri ++ — scrive Ri

2) decodifico un salto condizionale o incondizionato ma "su registro"

IF< Ri, RN, loop

GOTO RK

dipendenza a distanza 1 — k = 1

```
i     INC Ri
i+1   IF< Ri, Rn, loop
```

	0	1	2	3	4	5	6	7	8
IM	INC	IF<							
IU		INC	IF<	IF<					
DM			Ri						
EU			INC						

distanza k = 2

```
INC Ri
ADD R1, R2, R3
IF< Ri, Rn, loop
```

	0	1	2	3	4	5	6
IM	INC	ADD	IF<				
IU		INC	ADD	IF<			
DM			Ri				
EU			INC	ADD			

k = 2

```
LOAD Rb, Rj, Rx
INC  Ri
ADD  R1 R2 R3
IF<  Ri, Rn, loop
```

	0	1	2	3	4	5	6	7
IM	LD	INC	ADD	IF<				
IU		LD	INC	ADD	IF<	IF<		
DM			LD		Ri			
EU				LD	INC	ADD		

k = 3

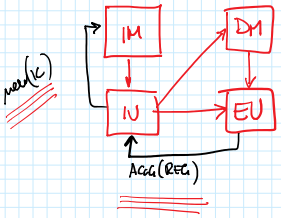
```
LOAD
INC
ADD
SUB Rx, R5, R6
IF<
```

LD	INC	ADD	SUB	IF<			
	LD	INC	ADD	SUB	IF<		
		LD		Ri			
			LD	INC	ADD	SUB	

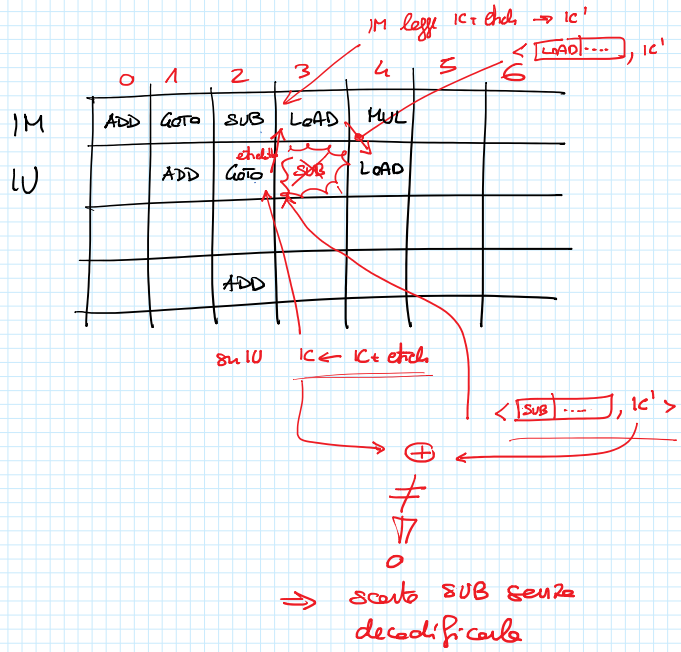
Solo le dip logiche $k \leq 2$ (possono avere) hanno un peso e il peso può aumentare se c'è una LOAD nella seq di istruzioni che porta alla dipendenza

EFFETTI dei SALTI

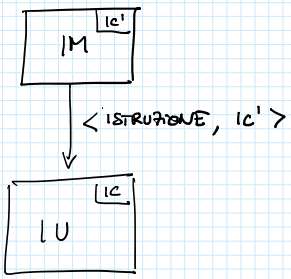
martedì 6 dicembre 2016 11:43



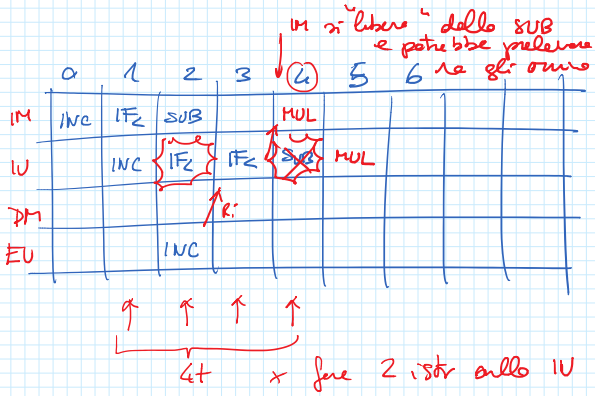

```
ADD  R1 R2 R3
GOTO etichetta
SUB  R4 R5 R6
INC
 :
etichetta : LOAD Rb, Ri, Rf
            MUL
```



⇒ scarto SUB senza decodificarla




```
INC Ri
IF< Ri, RN, loop
SUB
ADD
```



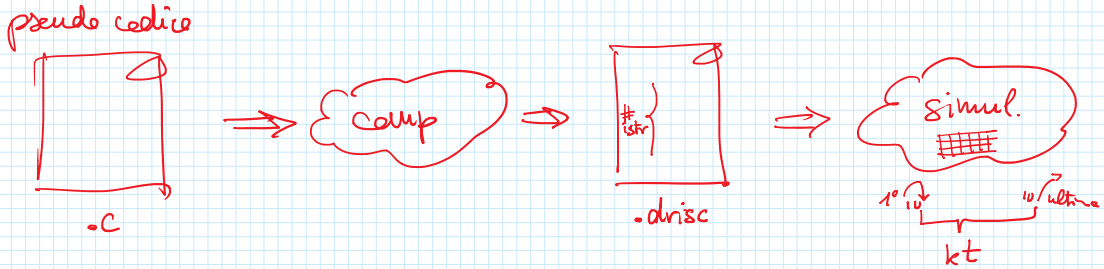

```
loop : MUL
```

$$\varepsilon(n) = \frac{T_{id}}{T(n)}$$

T_{id} INC + IF< ⇒ 2t

T ⇒ 4t

$\varepsilon = 50\%$



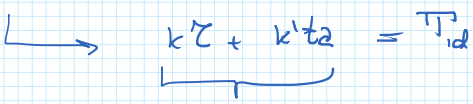
$$T_c = kt$$

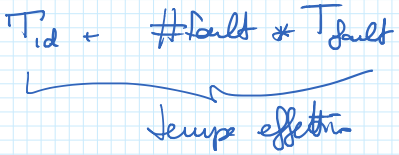
$$T_s = \frac{kt}{\#istr} = k't$$

$$\varepsilon = \frac{\#istr \cdot t}{k \cdot t} = \frac{\#istr}{k}$$

Proc monolitico

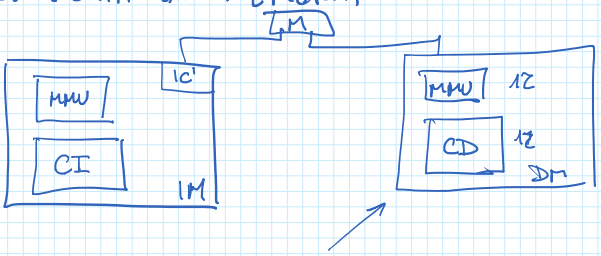
#istr
- 14 AL
- 6 LOAD
- 3 ST
- 4 SALT reg
- 5 SALT cond



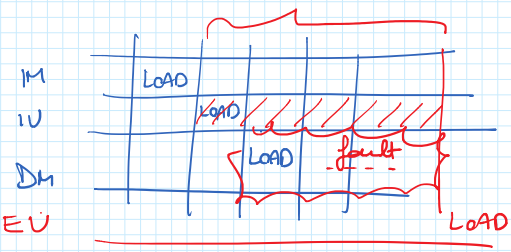


PIPELINE & GERARCHIA di MEMORIA

martedì 6 dicembre 2016 12:03



$t = 2\tau$



$T_{c_{id}} = kt$

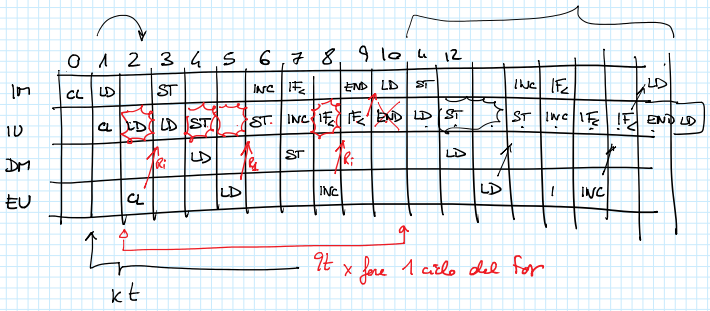
$T_c = T_{cid} + \#fault * T_{trap}$

Calcolo di T/Ts/Tc/E su processore PIPELINE di RISC

martedì 6 dicembre 2016 12:23

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

```
loop:  CLEAR  Ri
       LOAD   RboxB, Ri, R1
       STORE  RboxA, Ri, R1
       INC    Ri
       IFE    Ri, RN, loop
cont:  END
```



$T = 8t \cdot N$

$\varepsilon = \frac{4t \cdot N}{8t \cdot N} = \frac{1}{2} = 50\%$

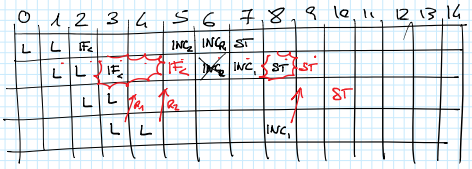
Al meglio $T_c = 1t + N * 4t + 1t$ (CLEAR, CORPO, END)

$T_c \approx N * 4t$

```
if(a[i] < b[i]) {
    a[i]++;
} else {
    b[i]++;
}
```

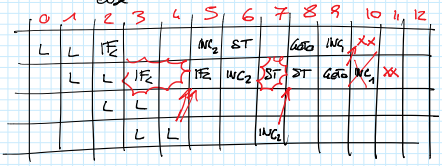
```
1          LOAD  RbA, Ri, R1
2          LOAD  RbB, Ri, R2
3          IFE   R1, R2, then
4   else:  INC   R2
5          STORE RbB, Ri, R2
6          GOTO  cont
7   then:  INC   R1
8          STORE RbA, Ri, R1
9   cont:  INC   Ri
10         IFE   Ri, RN, loop
```

then



5 istruzioni (9t) $\varepsilon = \frac{5}{9}$

else



6 istr (10t) $\varepsilon = \frac{6}{10} = \frac{3}{5}$

```
for( ) {
    if( )
        then
        else
}
```

p prob di eseguire then

(1-p) prob else

$T_c = N * (p(9t) + (1-p)10t)$ → + #fault × Ttransf

$T = N * (p \cdot 5t + (1-p)6t)$ = 3(....)

```
for(i=0; i<N; i++) {
    if(a[i] < b[i]) {
        a[i]++;
    } else {
        b[i]++;
    }
}
```

PROPRIETÀ

	Riuso	Località
A	No	Sì
B	No	Sì

WS

- codice (cache)
- 1 linea di A (∈A[i])
- 1 linea di B

⇓

#FAULT

Come ottimizzare questo fault?

N/0 fault × A

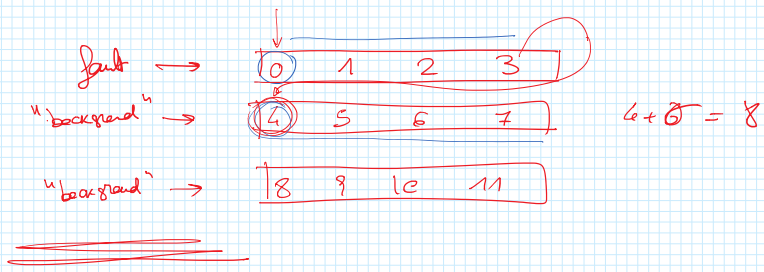
N/0 fault × B

(1-2) fault × codice

LOAD RbA, Ri, R1, prefetch

LOAD RbB, Ri, R2, prefetch ⇒ #fault 2 × dati + quelli del codice

fault → (0) 1 2 3


fault
0 1 2 3
"background"
4 5 6 7
"background"
8 9 10 11
6 + 0 = 8