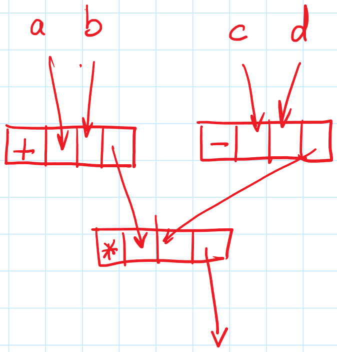
mercoledì 29 novembre 2017 08:45

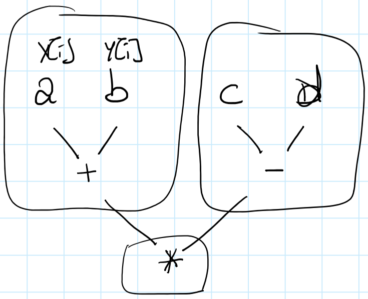
DATA FLOW



X[i] Y[i]

$(a+b) * (c-d)$

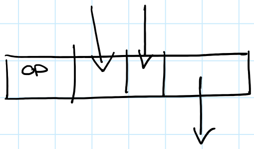
t_1 * t_2



②	LOAD	X[i]	
	LOAD	Y[i]	
	ADD		R_1
①	SUB	C-D	R_2
③	MUL		

GRAFO DATA FLOW

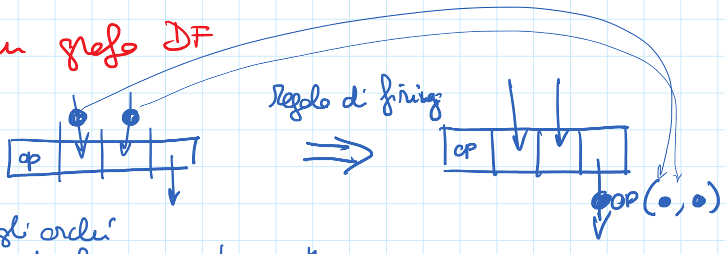
istruzioni DF



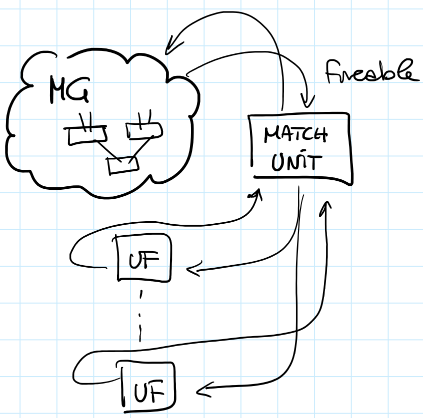
"Calcolo" di un grafo DF

fireable

Regola di firing



"tutti gli archi entranti hanno un token"

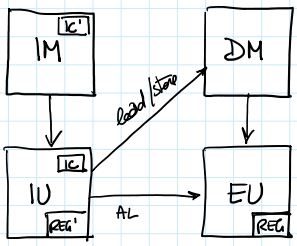


PROCESSORE D-RISC "PIPELINE" (HARVARD)

mercoledì 29 novembre 2017 09:35

Instruction memory — fetch

Instruction unit — decode + exec salti

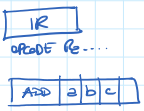


data memory — execute LD/ST

execution unit — execute AL / LD

ADD

IM → IU



IU decodifica → EU

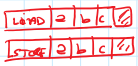
<+, a, b, c>

EU

$R_a + R_b \rightarrow R_c$

LOAD STORE

IM → IU



IU decodifica

R_a+R_b'

<"LD", ind, R_c'> → DM

<"ST", ind, val>

→ EU <"ld", c>

DM esegue op richiesta

nel LD invio a EU

<val>

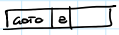
EU

IU → <"ld", c>

EU → <val>

} val → R_c

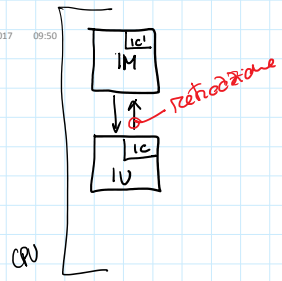
SALTI IM → IU

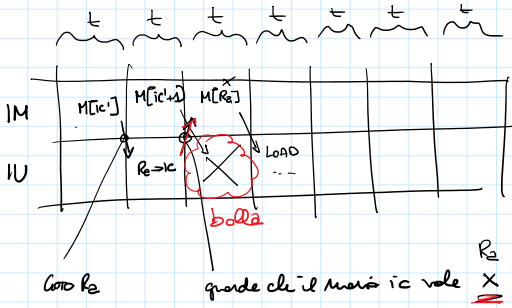


IU

R_a → IC

mando ad IM il nuovo valore di IC



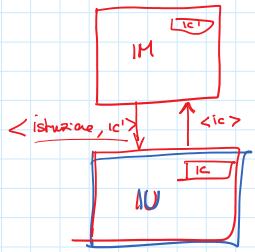


← tempo di servizio (o latenza, tanto IM ed IU sono sequenziali) di IM ed IU

GOTO R_2

cont: ADD R_2, R_b, R_c

: LOAD ...



while true {
  se IU mi rende new IC
  then rendo alla IU M[new IC]
    new IC → IC'
  else rendo alla IU M[IC']
  IC'++
}

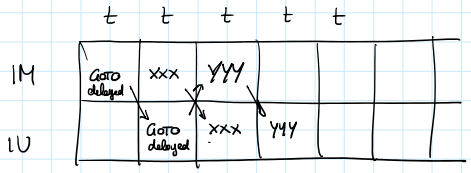
```
while (true) {
  ...
  if ( <istruzione, ic'>.IC' == IC')
    then (decodifica <istruzione, ic'>.istruzione)
    else nop
```

delayed branch

GOTO R_2, delayed = 1
XXX

R_2 → YYY

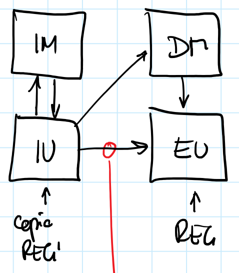
come se:
XXX
GOTO R_2
YYY



stessa bolla

esse "Bernstein"

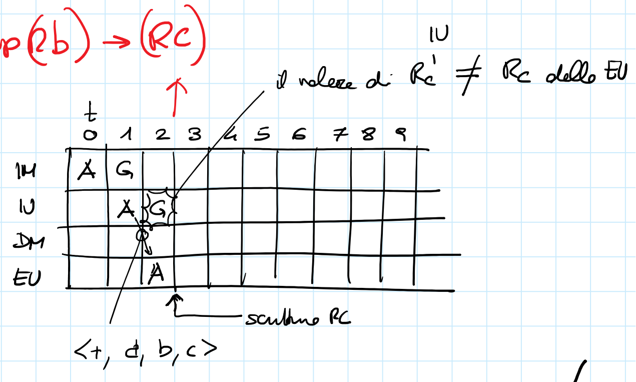
XXX non altera registri letti da istruzione di salto



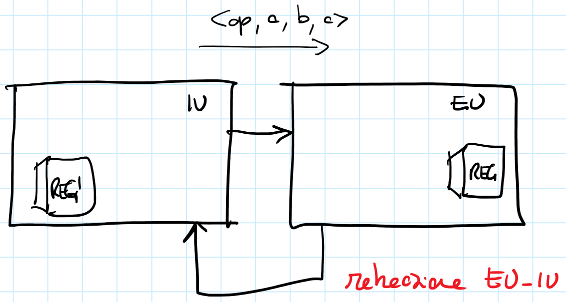
ADD Ra, Rb, Rc
GOTO Rc
XXX

YYY

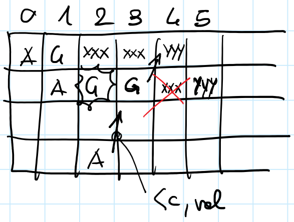
<op, a, b, c> ⇒ EU: fai (Ra) op (Rb) → (Rc)

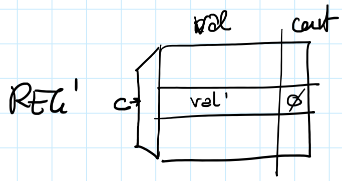


evidenzia il problema



nuovi valori dei reg
<c, val>





sse IU manda <op, - - c> ad EU
cont (c) ++

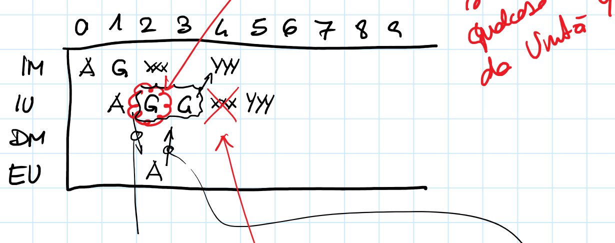
quando IU legge Rc
se cont (Rc) ≠ 0 ⇒ si blocca

ogni volta che arriva < val, x > dal canale retroazione EU→IU
val → Reg'[x]. valore
Reg'[x]. cont --

ADD Ra, Rb, Rc
GOTO Rc
XXX

YYY

bolla dovuta a "dipendenza logica" lettore di leggere in Unità X qualcosa scritto da Unità Y



IU 1) manda a EU <+, a, b, c>
2) cont (Rc) ++

EU → IU < val, c >
IU val → REG'[c]. valore
REG'[c]. cont --

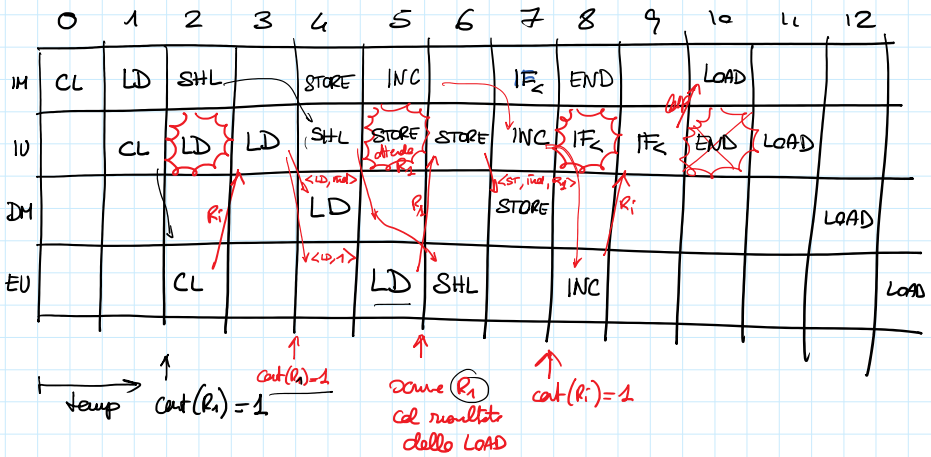
bolla "da salto"

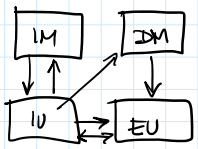
esecuzione del programma

```
for(i=0; i<N; i++) c[i] = b[i] × 2;
```

Tid = 6t

```
t  CLEAR Ri
loop: t  LOAD  Rbase B, Ri, R1
t  SHL   R1, #1, R1
t  STORE Rbase A, Ri, R1
t  INC   Ri
t  IF<   Ri, RN, loop
   END
```





$$\varepsilon = \frac{6t}{10t} = 0.6 \qquad \frac{T_{id}}{T(n)}$$

tempo di completamento del codice: init + 1ª iterazione } 10t in questo caso

2 problemi

bolle "di salto"

bolle "di dipendenza logica"

SOLUZIONI

usare delayed branch

⇓

Riorganizzare codice assembler D-RISC

loop: SUB R_i, #1, R_i

... ...

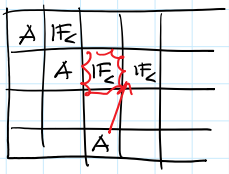
ADD R_B, R_D, R_C

$IF_<$ R_C, R_N, loop

loop+1

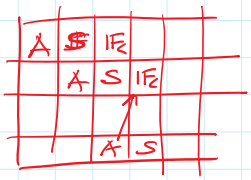
K=1

R_i --



3t x 2 ist

ε = 2/3



3t x 3 ist

ε = 1