venerdì 2 dicembre 2016 11:05

Note organizzative

1) Martedì 6 D.C non c'è lezione / ricevimento

Ricevimento Lun 5 ore 8-13

Lezione recuperata (se serve) 12 ore 14 qui 6

2) compitini

- orale compitino 10 gennaio 8R 9 Studio
- orale compitino fallito → compitino perso
  - rifiutato
  - non acc
- Vedi gennaio/febbraio
- occorre scrivere usare specificatamente "solo orale"

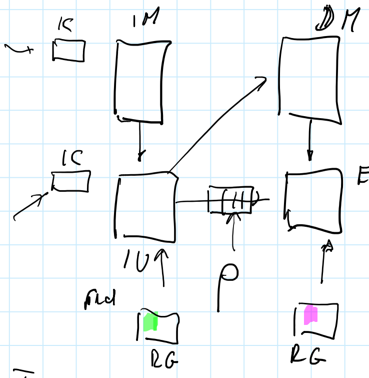
3) risultato scritto → numero orale
→ non numero all'orale

4) orale verrà sostenuto nello stesso appello dello scritto

5) secondo verifica info appelli a tutti

Uniprocessori e bubbloneri

Uniproc. Paralleli

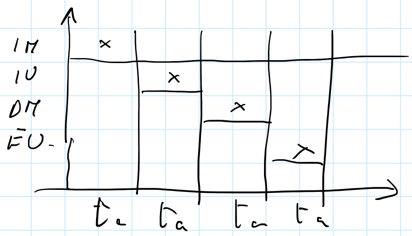


RR → t_a

LD

JMP → $2t_a$ eff. IC

ST → a) leggere reg
b) mem. DM



$t_c \geq 2\tau$

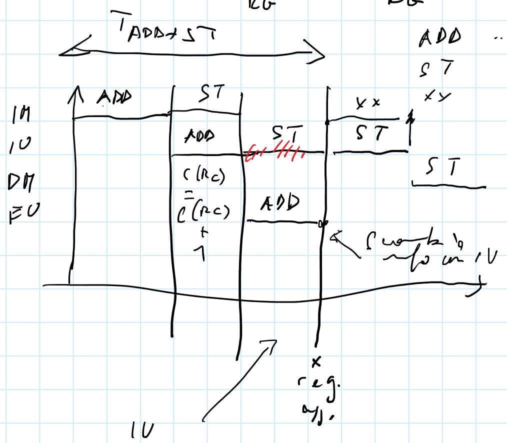
$\equiv$ t_{po} + tr ×

ADD Ra, Rb, Rc

ST Rc

LD Rc

IU deve essere certa che registro (sua copia) è aggiornato



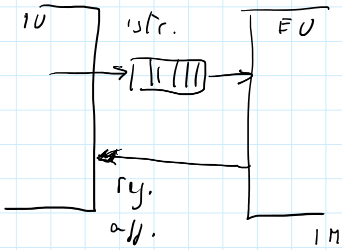
IU
blocca
Rc ≡ non coincide
read after write

ritardi dovute a
- salti
- dipendenze read after write
IU EU

Considerare ≡ contatore
ogni copia di registro nella IU ha associato un contatore (≥ 0)
- UI incrementa il contatore di R_i ogni volta che trasmette ad EU una richiesta che scrive in R_i
- UI decrementa contatore di R_i ogni volta che riceve un valore di R_i da EU
- UI può leggere R_i solo se $C(R_i) = 0$

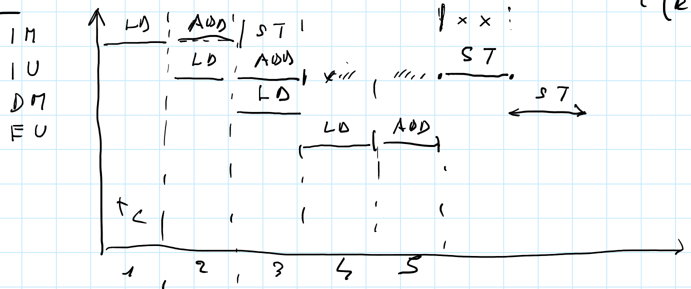
venerdì 2 dicembre 2016 11:37

Compero Livornesi e Fiorentini



LD R1 R2 R3 ⟶ c(R3) + 1 (IU)
ADD R3, R4, R5 ⟶ c(R5) + 1
ST R5, R5, R3 ⟶ ma incrementa contatore e può essere eseguita
xx

se c(R3) = ϕ
c(R5) = ϕ

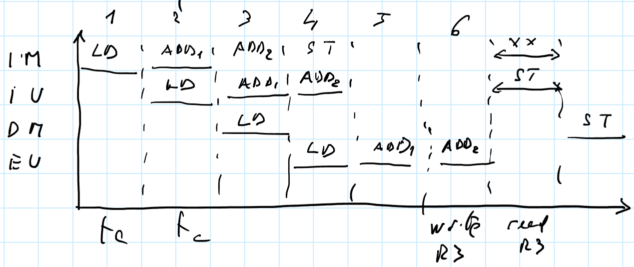


5 t_a per 3 ist.

LD R1, R2, R3
ADD₁ R5, R4, R2
ADD₂ R6 R7, R3
ST R6, R7, R3
xx

write (EU) R_3
read (IU)

ϕ



6 t_a

Non funzion.

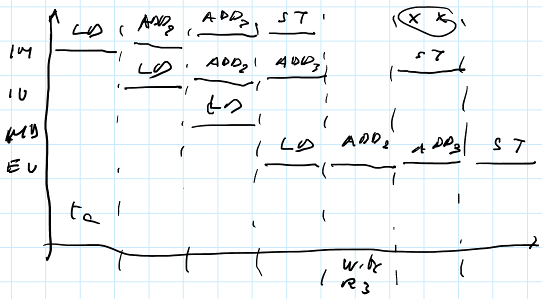
ADD₁ R5 R4, R2
ADD₂ R6, R7, R3
⟹ sono indipendenti
condizioni di Bernstein
$R_i \cap W_j = \phi$
$R_j \cap W_i = \phi$
$W_i \cap W_j = \phi$

R_i ≡ dominio lettura

W_i = dominio di scrittura

LD R1 R2 R3
ADD₂ R6 R7 R3
ADD R5 R4 R2
ST R6, R7, R3

1



5 t_a

DIPENDENZE IU – EU

Distanza tra due istruzioni in dipendenza (IU – EU)

determina il degrado ≡ aumenta al diminuire della distanza

Qualità di un codice ≡ – Distanza media tra le dip. ≡ Tra le istr. due loro in dip.

– Somma delle dist. delle dip.

LD → calcolo ind

ST → calc. ind + op

salti condiz. → degrado salto / → degrado lett.

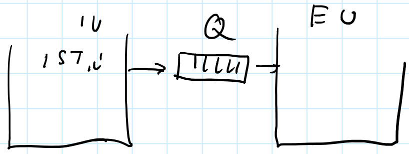
Δ ≡ → aumenta con il rischio dipend. / → inv. prop. distanza

$$T = n \times t_a + Salti + \Delta$$
$$= n \times t_c + n_s\, t_e + \Delta$$
$$\leq n \times t_a + n \times p_s\, t_e + \Delta$$
$$= n(t_a + p_s t_e) + \Delta$$
$$= n t_a (1 + p_s) + \Delta(p, \dots)$$

p_s = prob. di eseguire salto > prob. di salto

distanza — dipende dalla lunghezza delle code tra IU ed EU

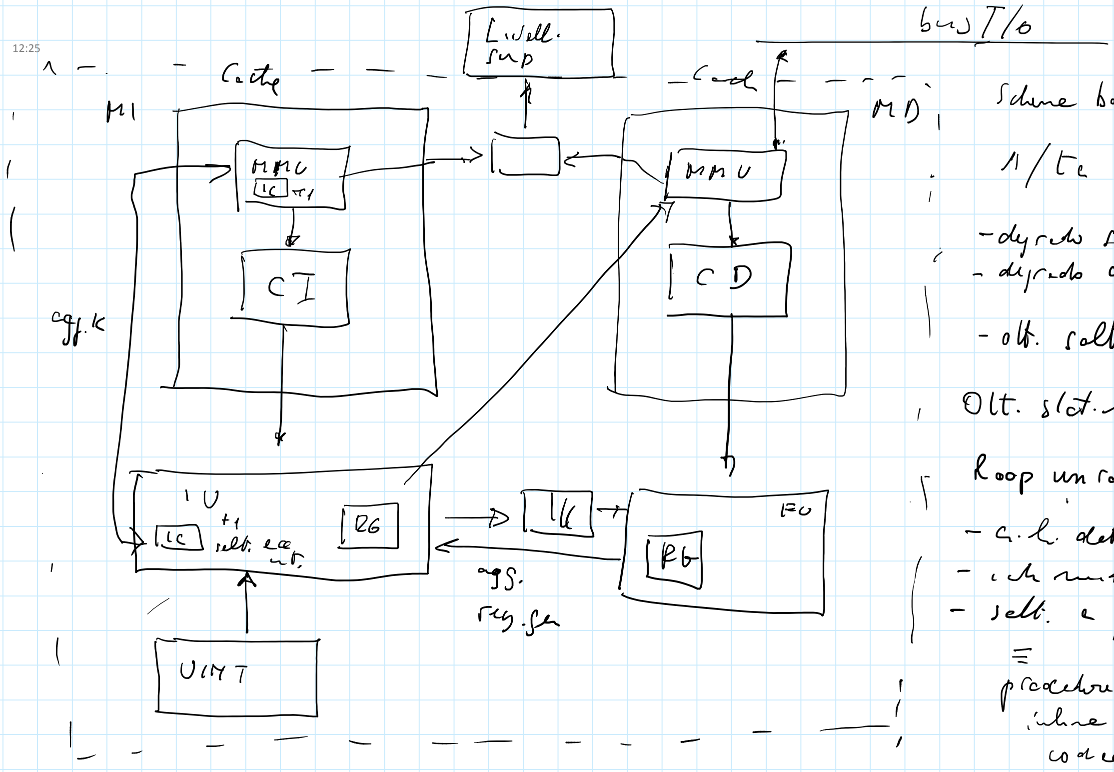
↳ degrado medio aumenta all'aumentare della lunghezza media di Q



$IST_i \xrightarrow{dip.} IST_J$

lunghezza media Q = $\rho^2 / (1-\rho)$ $\quad \rho$ = fatt. utilizzo $= \lambda / \mu$

tempo di attesa di IST_i ≡ degrado aumenta con l'aumentare della lunghezza media di Q



Schema base

1/Tc

- degrado salti
- degrado dipendenze
- ott. salti

Ott. statiche (comp.)

Loop unrolling / unfolding

- c. l. det. for ×
- while / rep ×
- salti a proc.

procedure ≡ macro
inline replacement
codice (no ricors.)

loop unrolling

for i = ∅, i < K, i++

for i = ∅, 2N, i++

{ ..... }×

2N salti

for J = ∅, N J++
{ i = 2J
{ ... }×
i = i+1
{ ... }×
}

N salti

DELAYED BRANCH

Modifica semantica linguaggio assembler

Istruzione che segue un branch viene comunque eseguita

senza delayed bran. | con delayed bra.

LOOP : ⋮ GOTO LOOP xx

LOOP : ⋮ GOTO LOOP y ← istruzione che sta tra LOOP e GOTO xx

caso ottimo

LOOP: ⋮ GOTO LOOP NOP xx

se ∄ una istruz. che si può muovere