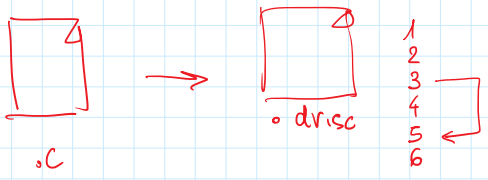
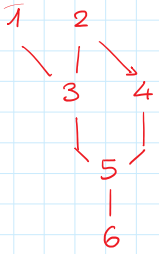
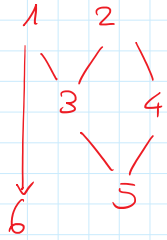
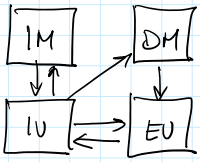
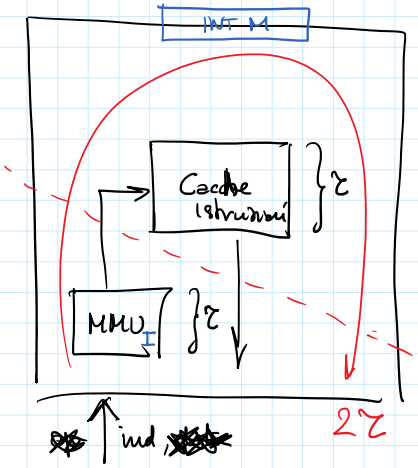


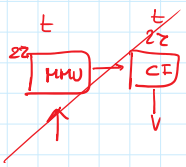


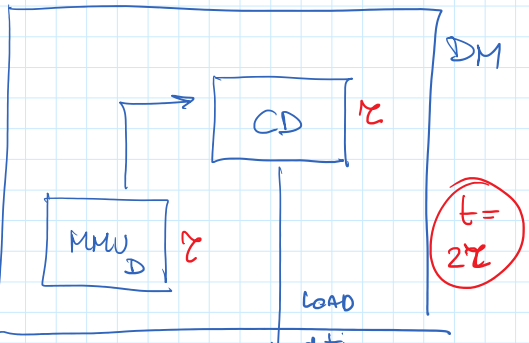


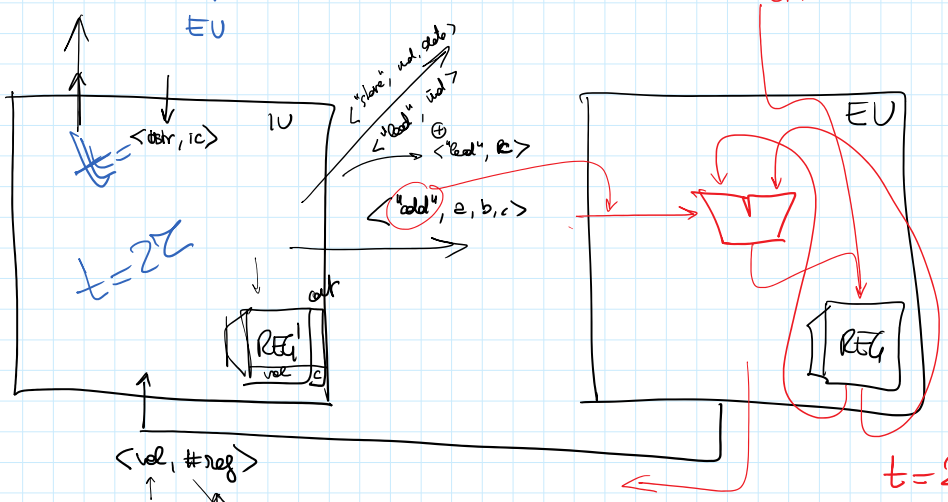


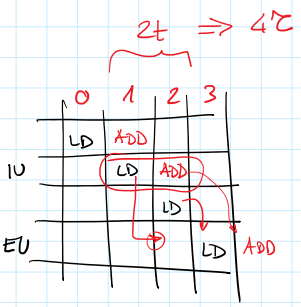
t ? kτ?

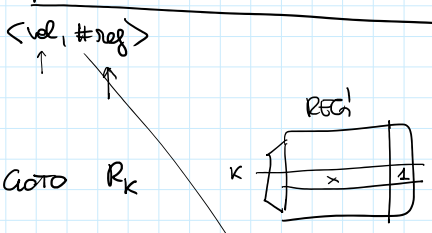












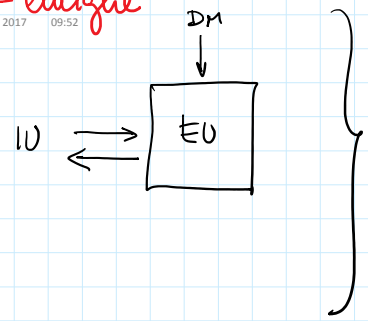
Es. (RDY in EU = 0) (=1) nop, y

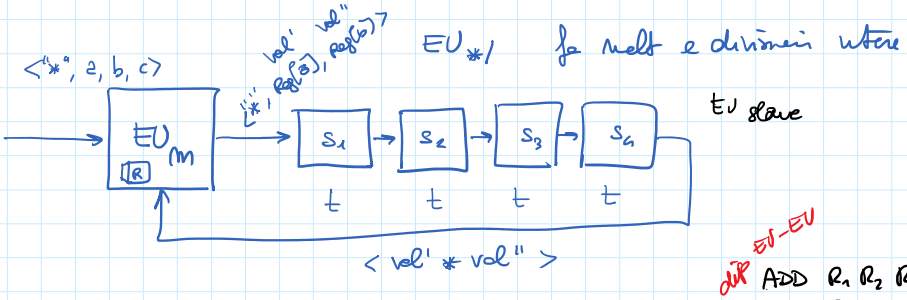
IC + DATAIN da EU val → IC

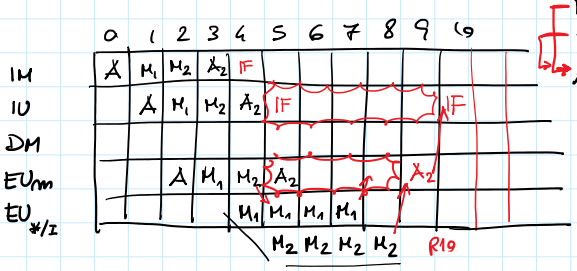
IC + REG(IR.R2) → IC

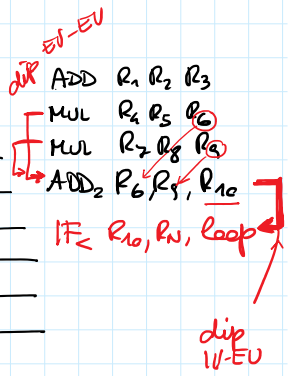
AL - "lunghe"

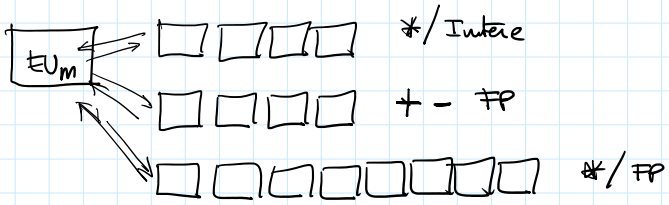
lunedì 4 dicembre 2017 09:52







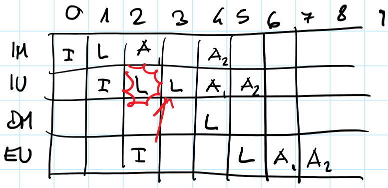




DIPENDENZE EU-EU

out of order

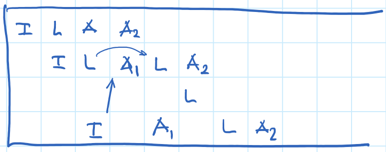
lunedì 4 dicembre 2012 10:19



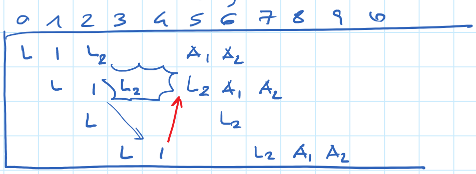
INC Ri
LOAD Rbase, R1, R2
ADD Rc, Rd, Rd
ADD Rd, R2, R2

out of order

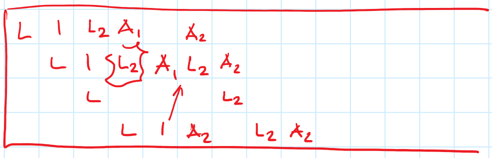
→ sospendere l'esecuzione della LOAD
→ analizzare l'istr successiva
sse non serve cose che richiede la LOAD (ADD non usa cose scritte dalla LOAD)
⇒ esegue l'istr successiva



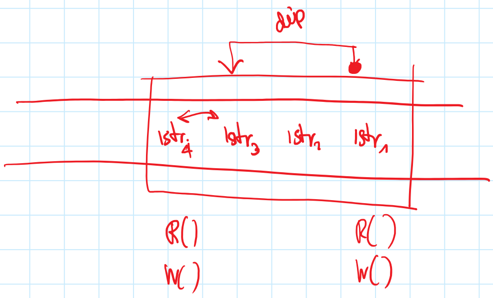
LD
INC
LOAD2
ADD
ADD



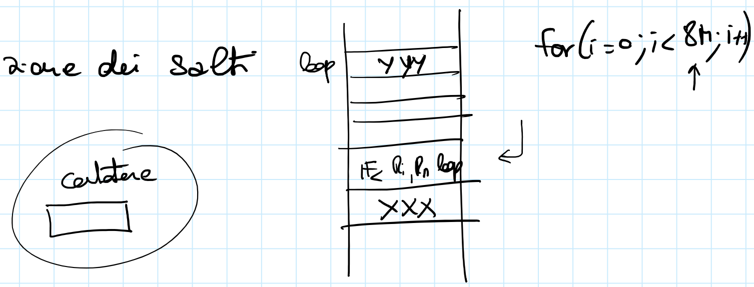
senza out of order



con out of order



Predizione dei salti

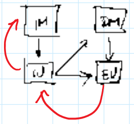


costo area sul chip ≈ 10-15%

produce un incremento delle prestazioni ≈ 7-8%

lunedì 4 dicembre 2017 10:40

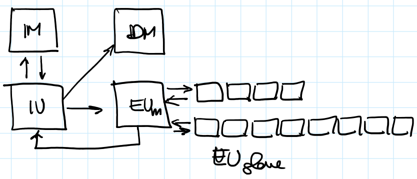
a)



① Telecomandi ⇒ dipendenza logica (IU–EU)

② EU_m – EU slave
⇒ dipendenze logiche (EU–EU)
⇒ supporto del AL–lunghe

③ esecuzione out-of-order
⇒ permette di mitigare l'effetto delle dip



SW

⇒ problema dipendenze logiche

tecniche statiche | tecniche dinamiche (out of order) (((delayed branch)))

– spostamento di codice

≡
≡ ⟲ (se non cambia la semantica)
≡ ⟧ dip IU-EU

– loop unrolling } + iterazioni di un ciclo senza salti presi (manualmente)

– trasformazioni di codice

loop : LOAD R_b, R_i, R_1
≡
INC R_i
IF< R_i R_N loop
cont :

⇓

loop : LOAD
≡
INC R_i
IF< R_i R_N, loop, delayed
[LOAD R_b, R_i, R_1]

loop : ≡
INC R_i
IF< R_i R_N loop

⇓

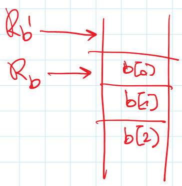
loop : ≡
INC R_i
IF= R_i R_N exit
≡
INC R_i
IF< R_i R_N loop

loop : ≡
INC R_i
≡
INC R_i
IF< R_i, R_N, loop

(sse # iterazioni pari)

LOAD ...
...
..
STORE R_b, R_i, ..
INC R_i
IF<

LOAD R_{base}, R_i, R_1
⋮
INC R_i
IF< R_i R_N, loop ⟧ dip IU-EU

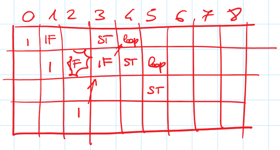


usiamo R_b' — registro base che punta alla loc precedente a quella di B[0]

INC R_i
STORE $R_{b'}$, R_i, ...
IF< R_i R_N, loop

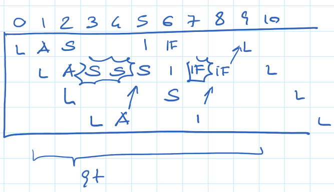
⇒

[INC R_i]
IF< R_i R_N loop, delayed
STORE R_b', R_i, —
≡



for( )
x[i] = y[i] * b

LOAD R_{basey}, R_i, R_1
ADD R_1, R_b, R_2
STORE R_{basex}, R_i, R_2
INC R_i
IF< R_i, R_N, loop



LOAD
INC
ADD
IF< delayed
STORE R_b'