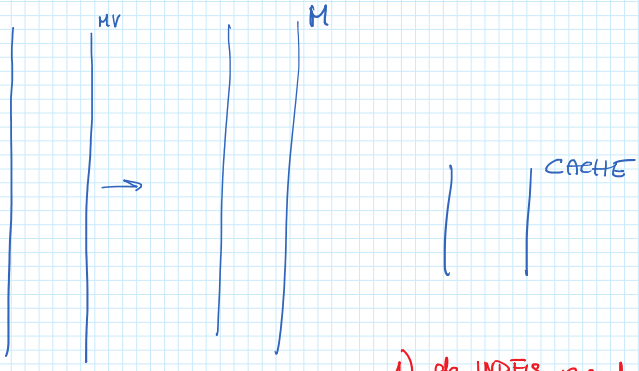
VEN 25 ore 14-16 aula C

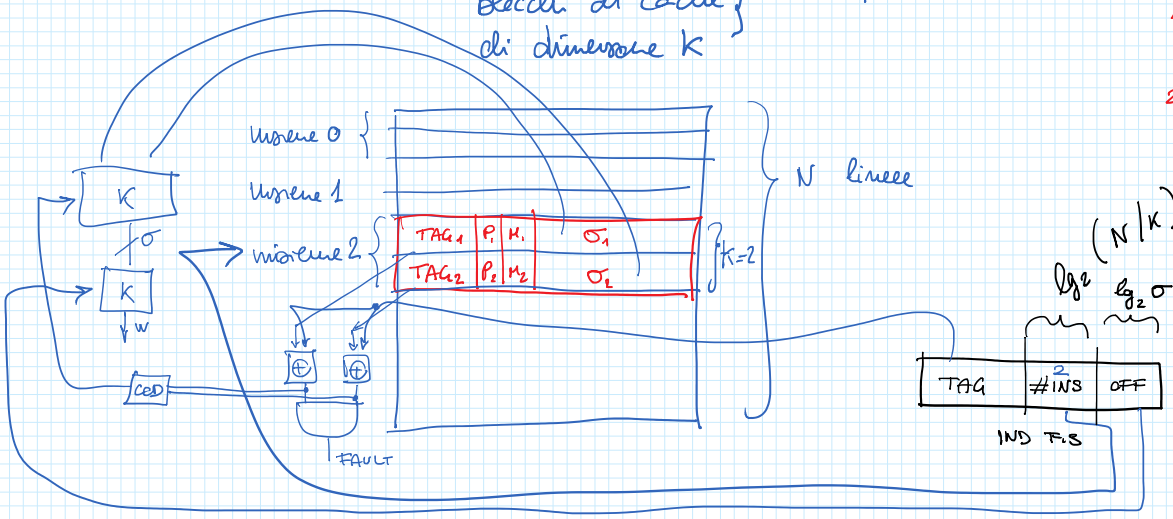
venerdì 18 novembre 2016 11:04

CACHE

DIRETTO ASSOC ASSOC. su INSIEMI



Insieme = { gruppo di linee / blocchi di cache } di dimensione K



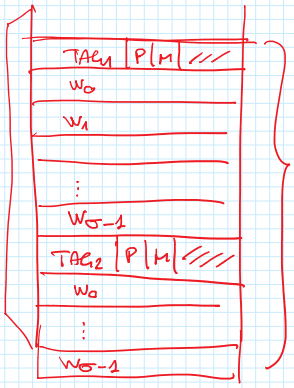
1) da INDFIS prendo #INS ⇒ decido quale insieme

2) prendo $TAG_1 \dots TAG_K$ delle linee dell'insieme e li confronto con TAG da INDFIS ⇒ decido quale linea oppure FAULT

3) con OFF da INDFIS ⇒ decido quale delle σ parole

$\left(\frac{N}{K}\right)$

$\log_2$ $\log_2 \sigma$ bit



σ = 8

9 posizioni × linea

2 pos × insieme

18 w × insieme

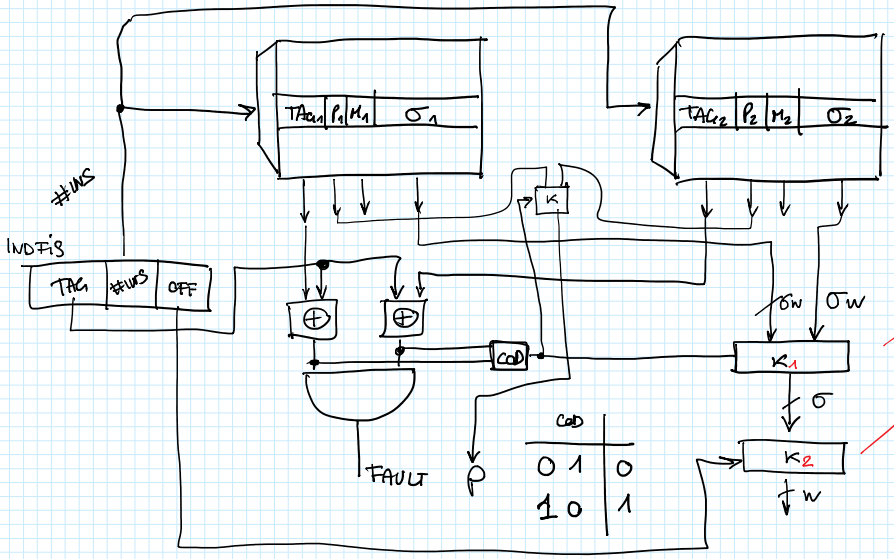
#INS = i

i · 18

1ª linee degli insiemi

2ª linee degli insiemi

K = 2



1) t_a

2) $t_\oplus$

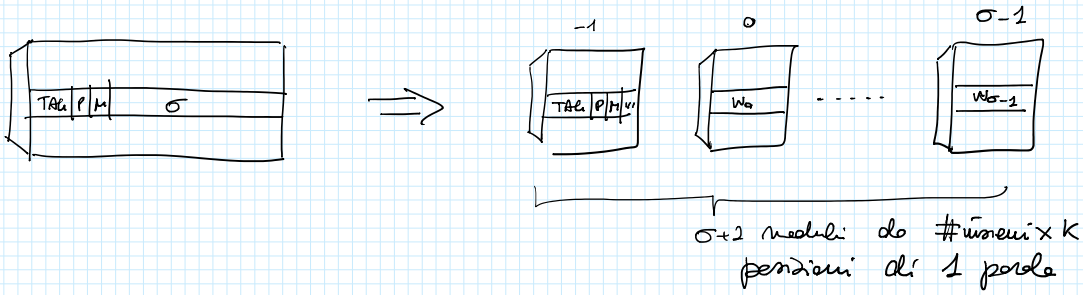
3) t_{cod}

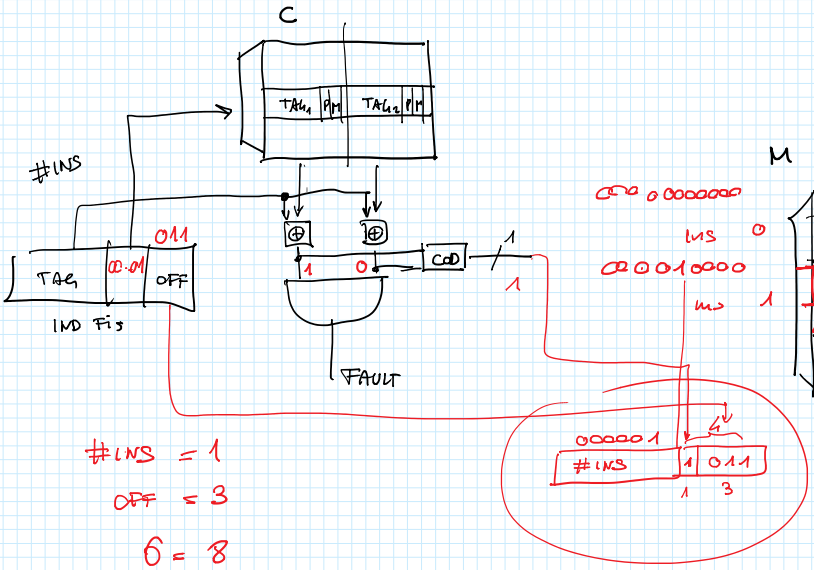
4) t_K } $2t_p$

5) t_K } $2t_p$

Taccesso alla cache senza fault

perolo M[INDFIS] sse { FAULT = 0, P = 1 }





linea 0 dell'ins. ∅
" 1 " " ∅
" 0 " " 1
" 1 " " 1

#INS = 1
~~OFF~~ = 3
6 = 8

scarta linea dell'ins 1
(da confrontatori)

1) t_a (x TAG)
2) $t_\oplus$
3) t_{cod}
4) t_2

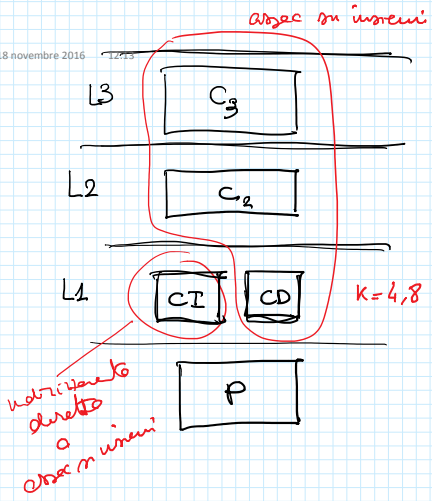
(nell'implementazione vale $2t_K$)

$t_a \rightarrow \lambda i.\ \big((C[\#INS].TAG_1 \oplus IND.TAG_1)\ AND\ (C[\#INS].TAG_2 \oplus IND.TAG_1)\big) \rightarrow F,$
$cod(C[\#INS].TAG_1 \oplus IND.TAG_1,\ C[\#INS].TAG_2 \oplus IND.TAG_1) \Rightarrow cod$

$t_a \rightarrow \lambda_{t1}.\ (F = 0)\quad M[\#INS \mid cod \mid (IND.OFF)] \rightarrow \ldots$

CACHE ASSOC m insiemi a k vie (#linee x insieme = k)

↳ tempo di accesso
- 1° colpo: $1t + t_{MM}$ → $2t$
- 2° colpo: $2t$ → $3t$

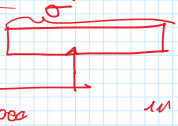


3° livello di cache 16 M
dati + istruzioni

2° livello di cache M
Dati e istruzioni insieme

1° livello di cache } 16k
Modello Harvard

1° livello { Solo dati del processo in esecuzione (codice)
al cambio di contesto si cancella (logicamente) il contenuto della cache (tutti i bit di presenza messi a ∅)
politica "on demand"
FAULT ⇒ richiesta della parte di cache la linea richiesta

potrebbe avere politica "prefetch"
se accedo IND1 ∈ 

tocca di portare in cache anche la linea che contiene (IND1 + 0⃗) se non già presente

LOAD Rdest, Rindice, Rtarget ; FLAG-PREFETCH
STORE Rdest, Rindici, Rsource

2° livello { abbiamo dati & codice di processi diversi
on demand
prefetching

MODIFICHE in CACHE

venerdì 18 novembre 2016 12:31

2 STRATEGIE di gestione delle MODIFICHE

/

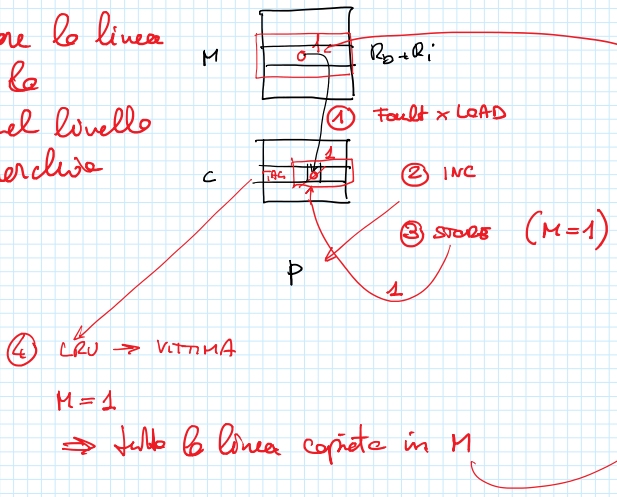
WRITE BACK

∀ STORE modifica l'indirizzo solo in cache

Quando dobbiamo liberare la linea soggetta a modifica, la copiamo in M / nel livello superiore della gerarchia di memoria

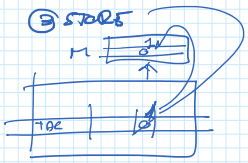
0

① LOAD R_b, R_i, R_1
② INC R_1
③ STORE R_b, R_i, R_1
1



WRITE THROUGH

∀ store modifica il dato in C e nel livello successivo della gerarchia di memoria



Scrittura in C è sincrona?

quella nel livello superiore è asincrona

venerdì 18 novembre 2016 12:42

for (i = 0 ; i < N ; i++) x[i] = 1;

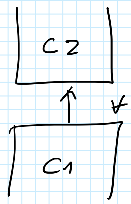
```
loop: STORE Rbasex, Ri, Runo      5τ + 2ta
      INC   Ri                    3τ + ta
      IF<   Ri, RN, loop          4τ + ta
      END ...
```

$12\tau + 4t_a = 12\tau + 8\tau = 20\tau$

↑ 2τ

STORE —— STORE

Hp C1 < CD assoc. su insiemi 2τ (1° schema); CI diretto 2τ



$t_{a_{C2}} \leq 20\tau$

$\forall\, 20\tau$

x il write through xché funzioni

$B(\text{STORE}) \leq B$ livello successivo della gerarchia di memoria

ASM

#istr * durata ⟹ T completamento del programma

↓

$T_{cho} + T_{ch} + T_{exec_{istr}}$

↳ T_c ideale $T_{c_{id}}$ (accessi in cache)

$$T_{completamento} = T_{c_{id}} + \#fault \cdot T_{trasf}$$

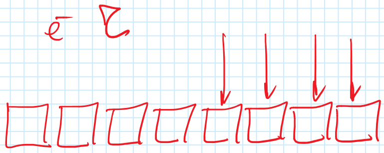
T_c

T_{trasf}: tempo di trattamento del fault

$$T_{trasf} = 2(T_{tr} + \tau) + \frac{\sigma}{m}\tau_M + \tau$$

1° schema

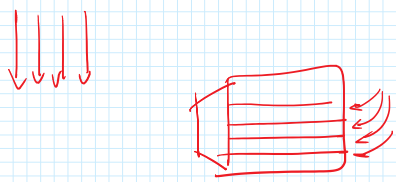
è τ



$\sigma = 8$

$m = 4$

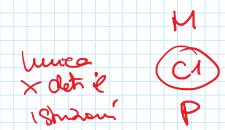
2° schema



è $m\tau$

Provate a valutare dato un codice ASM D-RISC quanti e quali sono gli accessi in memoria

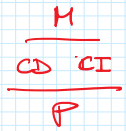
e quanti e quali sono i fault



Lunga x dati e istruzioni

associative in insiemi

K=2 σ=8



CD a indirizzamento diretto

CI assoc in insiemi K=2 σ=8

```
for (i=0; i<N; i++)
    A[i] = B[i] + C[N-i];
```