Es. 17 luglio 2015

mercoledì 22 novembre 2017 09:02

int A[N] P[N] N = 8K

16K ADD R_0, #1, R_{prefix}

16K+1 ADD R_0, R_0, R_i

16K+2 Loop: LOAD $R_{base\,A}$, R_i, R_{ai}

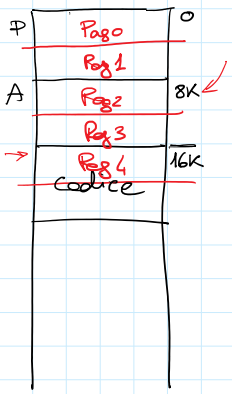
MUL R_{ai}, R_{prefix}, R_{prefix}

STORE $R_{base\,P}$, R_i, R_{prefix}

INC R_i

IF< R_i, R_N, loop

END

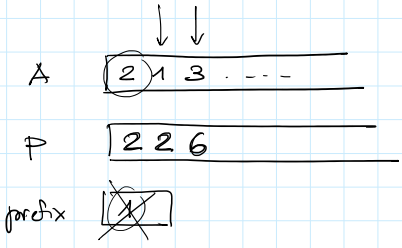


$R_{base\,A}$ = 8K

$R_{base\,P}$ = 0

R_N = 8K

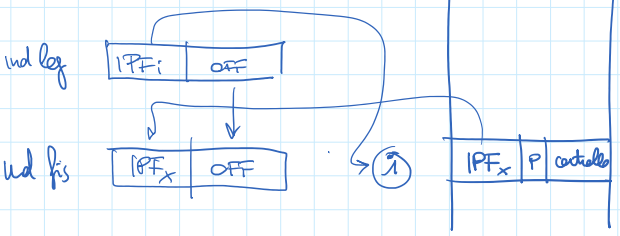
Mem 1M pag da 4K



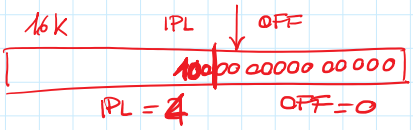
2
2
6

0	4
1	2
2	1
3	7
4	8

TAB RIL



$IPL_i \rightarrow IPF_x$



ind fisici del codice

8	0
1	
2	
⋮	
8	END

Ø

LOAD $R_{base\,A}$, R_i, ...

2	0

→

1	0
1	1

A[0]

A[1]

STORE $R_{base\,P}$, R_i, ...

0	0

→

4	0
4	1
4	2

P[0]

IPF	OFF		IPF · #PAG · OFF
8	0	ADD	8 · 4K + 0
8	1	ADD	8 · 4K + 1
8	2	LOAD	8 · 4K + 2
1	0	A[0]	1 · 4K + 0
8	3	MUL	
8	4	STORE	
4	0	P[0]	
8	5	INC	
8	6	IF<	
8	2	LOAD	
1	1	A[1]	
⋮			

Hp C1 set assoc 4 vie 1K set σ = 16

Hp: mem fisica $m = 4$

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

```
        ADD   R0, #1, Rprefix
        ADD   R0, R0, Ri
Loop:   LOAD  Rbase A, Ri, Rai
        MUL   Rai, Rprefix, Rprefix
        STORE Rbase B, Ri, Rprefix
        INC   Ri
        IFc   Ri, RN, Loop
        END
```

N 8K

$$\underline{\underline{2}}\left((2\tau + t_a) + \tau\right) +$$

$$N\left(5(2\tau + t_a) + 2(2\tau + t_a) + 50\tau + \tau + 2\tau\right)$$

- $5(2\tau + t_a)$: durata istruzioni del loop
- $2(2\tau + t_a)$: t_{exec} Load + store
- 50τ: MUL
- τ: INC
- 2τ: IFc

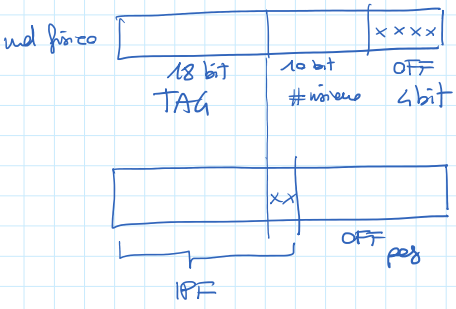
$$T_c = (6\tau + 2t_a) + N(67\tau + 7t_a)$$

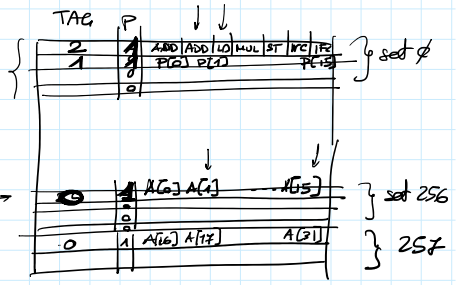
$$T_{c\,id} = (\quad 2\tau \qquad\qquad 2\tau \quad)$$

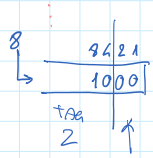
IPF	OFF
8	0
8	1
8	2
1	0
8	3
8	4
4	0
8	5
8	6
8	2
1	1

	TAG	SET	OFF	
ADD	2	0	0	fault
ADD	2	0	1	
LOAD	2	0	2	
A[0]	0	256	0	fault
MUL	2	0	3	
STORE	2	0	4	
P[0]	1	0	0	fault
INC	2	0	5	
IF	2	0	6	
LOAD	2	0	2	
A[1]	0	256	1	

1K set σ = 16 4 linee × insieme



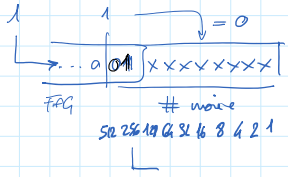




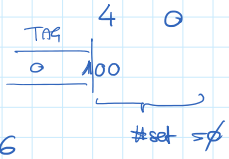
8 0
8 1
8 2
0 7

#set ∈ ∅

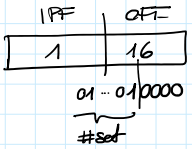
offcode
111



256



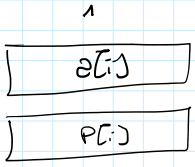
A[16]



256+1
257

	loc	riuso
cod	V	✓
a	V	—
p	V	—

```
prefix = 1
for(i=0 ; i<N ; i++) {
    prefix = prefix * a[i];
    p[i] = prefix;
}
```



N = 8K

A[N] P[N]

WS = { codice , 1 linea × A, 1 linea × P }

$$\#fault_{dati} = 2 \frac{8K}{16_{\sigma}} = 1k$$

8 istruzioni
$\sigma = 16$

$\#fault_{codice} = 1$

$$\#fault = (1K + 1) * T_{Resume}$$

da aggiungere al T_{cid}

LOAD R_{baseA}, R_i, R_{ai}, "PREFETCH" → leggerebbe $\left(\frac{N}{\sigma} - 1\right)$ fault

STORE $R_{basep}, R_i, R_{prefix}$, "PREFETCH" → leggerebbe $\left(\frac{N}{\sigma} - 1\right)$ fault

⇓

fault dati ⇒ 2

Se considerare che P è scritto ulteriormente nel prog senza leggerlo

⇒ fault di scrittura ≡ ∅

Hp "lavare" un ultimo costo molto meno del riempirlo dal livello superiore della gerarchia di memoria

⇓

#fault = 1 + 1
, Codice 1ª linea di A

5 8 Feb 2016

mercoledì 22 novembre 2017 09:03

```
for (i=1, i<1024, i++)
{     a[i]=b[i]*b[i]+c[1024-i]+c2;
      c[i]=c[i-1]+1;
      c2=c2+2;
      b[i]=c[i]+c2
}
```

Ritaglio schermata acquisito: 22/11/2017 10:36

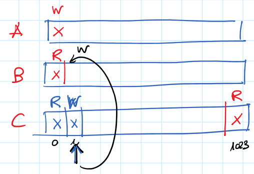
N > 1024

int A[N]

int B[N]

int C[N]

int c2



#fault "fisiologici"

— 1024 / σ = 2^3

2^{10} / 128 = 2^7

— 2^3

(sse non considero che A è in sola scrittura)

mi servono 8 linee da σ = 128

x contenere 1024

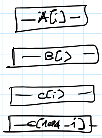
⇒ "non deallocare"

$\frac{N}{\sigma} \cdot \frac{1}{2}$

- calcolare il numero di fault generati dalla esecuzione del frammento assumendo che la dimensione della cache permetta di mantenere in memoria 2 pagine per istruzioni e 4 pagine per i dati,

Ritaglio schermata acquisito: 22/11/2017 10:36

WS
- codice
- linea di A
- linea di B
- 2 linee di C



set assoc a 8 vie

oppure

set assoc a 4 vie e #set (codice) ≠ #set (dati)

6 13 giugno 2016

mercoledì 22 novembre 2017 15:03

versione originale.

Domanda 2

Si consideri un array A di interi positivi con $N=2^K$ elementi. L'array è ordinato, dal valore più basso (prima posizione) a quello più alto (ultima posizione). Assumendo di avere a disposizione un processore D-RISC pipeline con EU monolitica e cache di primo livello (sia dati che istruzioni) associativa su insiemi a 4 vie, con linee da 32 parole (sigma = 32) e prefetching. Si calcolino il numero massimo di fault nell'esecuzione di una ricerca binaria e nell'esecuzione di una ricerca esaustiva sull'array (all'inizio della ricerca, nessuna pagina dell'array è allocata in cache dati).

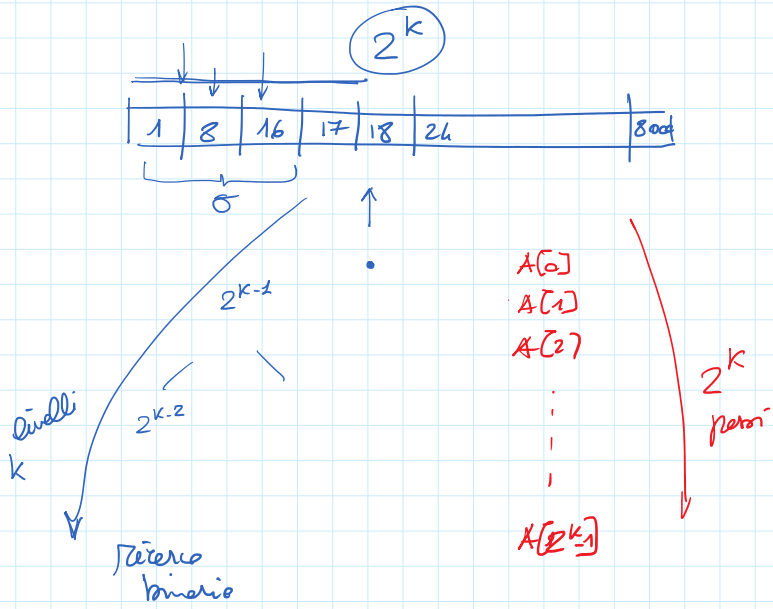
Lo pseudo codice delle due procedure di ricerca è il seguente:

```
/* ricerca binaria: cerca x cominciando da metà e poi continuando nella metà alta o bassa */
int start = 0;
int stop  = n;
int found = -1;
while(stop - start > 1) {
        int i = start + (stop-start)/2;
        if(A[i] == x) {found = i; break;}
        if(x < A[i]) { stop = i; } else { start = i; }
}
if(A[start] == x) found = start;
if(A[stop] == x)  found = stop;

/* ricerca esaustiva: cerca x controllando le celle una per una fino a che l'array è finito o l'elemento è stato trovato */
int found = -1;
for(int i=0; i<n; i++)
        if(A[i] == x) { found = i; break; }
```

Successivamente si fornisca il tempo di completamento della prima iterazione while della ricerca binaria nell'ipotesi che l'elemento cercato non sia stato trovato.

Ritaglio schermata acquisito: 22/11/2017 10:48



#fault?

n k fault

un caso di prefetch

1 fault

T_c

2^k

$T_{calcolo}$ log

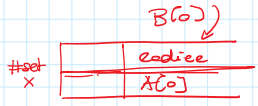
T_{fault} log

$T_{calcolo}$ lineare

T_{fault} costante

$m = 4$ γ_m T_{tr}

set assoc 2 vie $\sigma = 16$ 1k set



⇓ che opera su vettori

Fornire un codice ASM D-RISC in cui il #fault > #fault ginodicici

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

#fault ginodagici

$2 \frac{N}{\sigma}$ ∅ 1-2

A B X codice

#fault veri ?

4 codice + 2 1° linea di A 1° linea di B

③ fault stesso ordine

1) MV

2) ind log → ind fis

3) ? #set

8

mercoledì 22 novembre 2017 09:03

10

mercoledì 22 novembre 2017 09:03