Architettura degli Elaboratori

Cache set-associative e multi-livello

SAPIENZA
UNIVERSITÀ DI ROMA

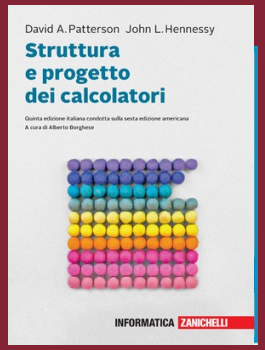
Alessandro Checco

checco@di.uniroma1.it

Special thanks and credits:

Andrea Sterbini, Iacopo Masi,
Claudio di Ciccio

[S&PdC]
5.1, 5.3 – 5.4



Argomento

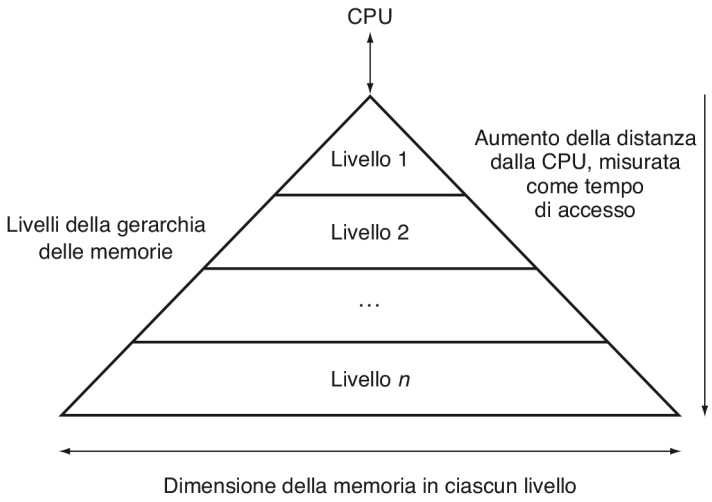
Problema:

la RAM è **molto più lenta** del processore (circa 10-100 volte più lenta)

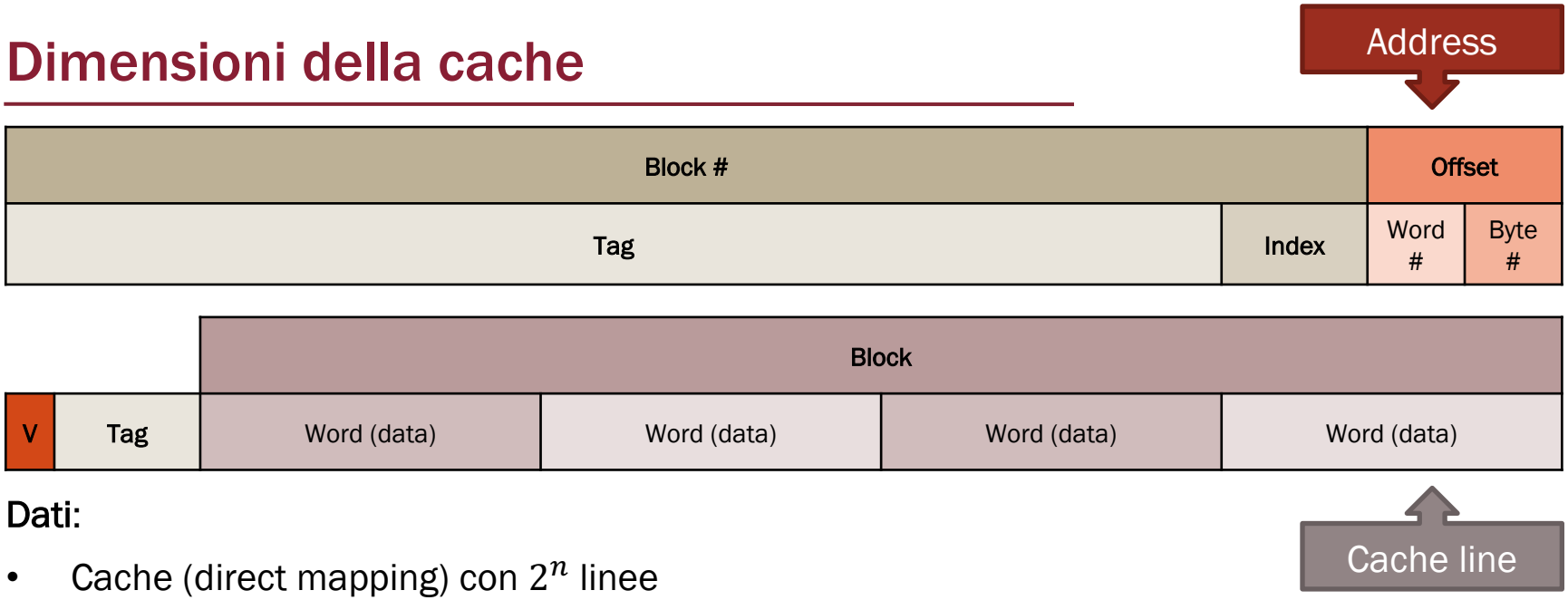
Una memoria più veloce è **molto più costosa**

1. **Principio di località temporale**
   - «un programma tende ad accedere allo stesso elemento in momenti vicini tra loro»
2. **Principio di località spaziale**
   - «un programma tende ad accedere successivamente elementi di memoria vicini tra loro»

- **Idea 1:**
  - mettiamo da parte in una **piccola** memoria **veloce** (**cache**) solo i **dati più usati**
- **Idea 2:**
  - quando memorizziamo un dato **mettiamo da parte anche i dati vicini**



Dimensioni della cache



Dati:

- Cache (direct mapping) con 2^n linee
- Blocchi di dimensione 2^m word da 4 byte ciascuna, cioè 2^{m+2} byte (2^{m+5} bit)
- 1 bit di validità

Dimensione del campo **tag:** $32 - n - m - 2$

Dimensione della **cache in bit:** $2^n \times [2^m \times 32 + (32 - n - m - 2) + 1]$

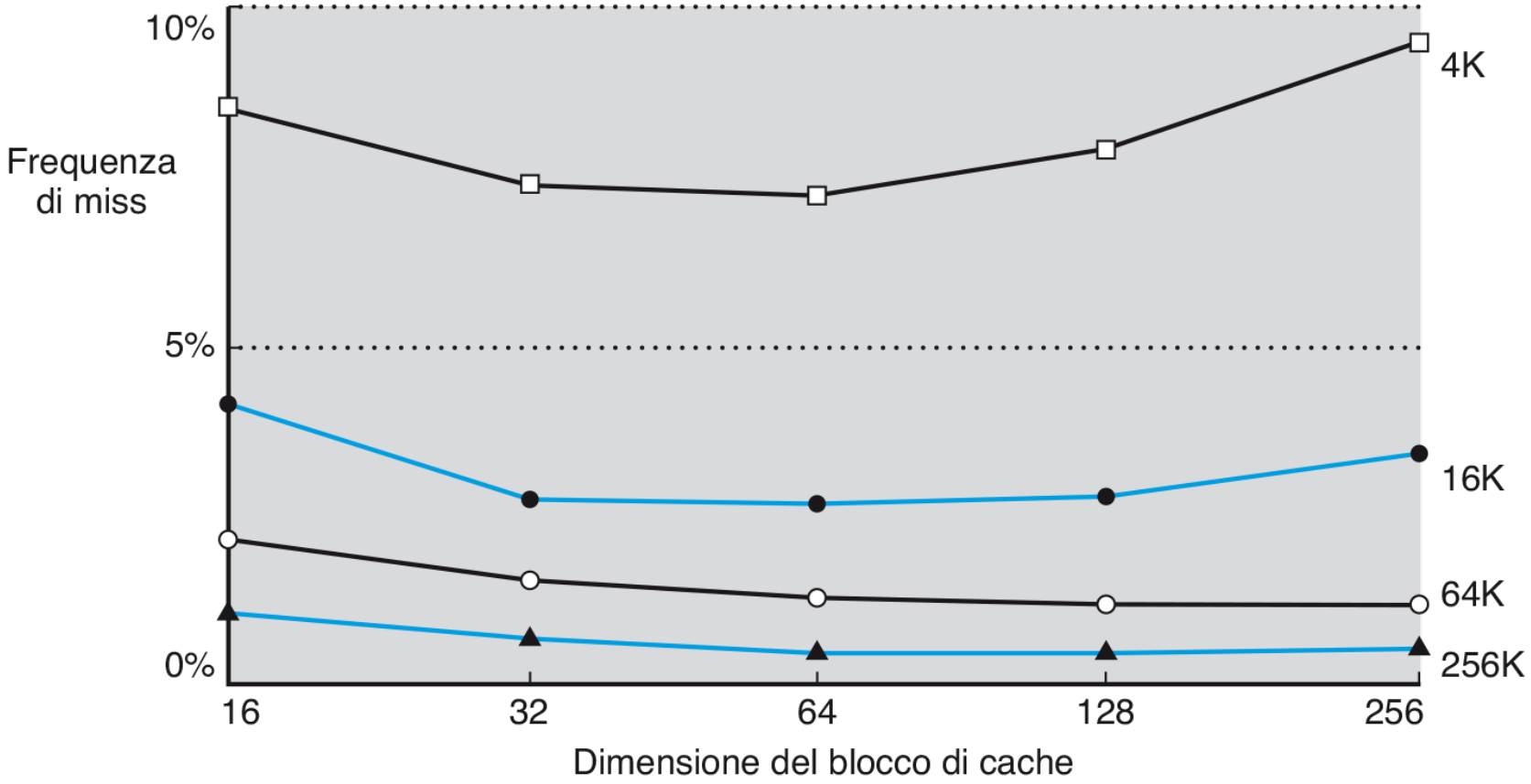
Esempio: cache con:

- blocchi da 16 word $\rightarrow m = 4$
- 256 linee $\rightarrow n = 8$

Dimensione del campo Tag: $32 - 8 - 4 - 2 = 18$ bit

Dimensione della cache in bit: $256 \times [16 \times 32 + 18 + 1] = 2^8 \times 531 = 132{,}75$ Kibibit

La dimensione del blocco non è la panacea



Altri tipi di mapping

Come migliorare la percentuale di HIT?

- **Direct mapping**
  - ogni blocco viene salvato in una linea fissata (che dipende dal numero di blocco)

    pro: hardware più semplice e meno costoso

    è facile determinare in quale linea cercare il dato

    #linea = #blocco % N

    contro: blocchi diversi sono mappati sulla stessa linea →

    se gli accessi a quei blocchi si alternano si verificano molti miss

    (**trashing**)

- **Fully-associative mapping**
  - un blocco può essere posto in una linea qualsiasi

    pro: massima flessibilità

    contro: hardware più complesso e più costoso

- **Set-associative mapping**
  - le linee sono organizzate in **S** insiemi (set)
  - ogni blocco è disposto in una qualsiasi delle linee del set fissato (che dipende dal #blocco)

    #set = #blocco % S

Attenzione: da qui in poi adottiamo la nomenclatura per il set-associative mapping usata in G.R. Wilson, in Embedded Systems and Computer Architecture, 2002

Associatività crescente

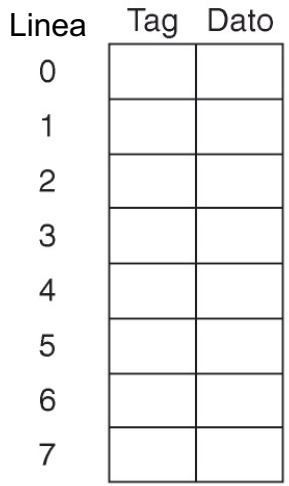
Cache con **8 blocchi** organizzati mediante diversi gradi di associatività

#vie = #linee per set

scelta del set è prefissata
#set = #blocco % S

se ho un solo set un blocco può essere posto in una linea qualsiasi

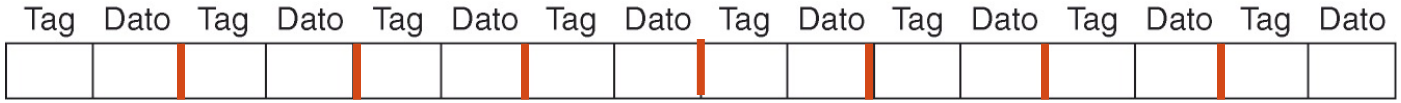
Cache set-associativa a una via (a mappatura diretta)



Cache set-associativa a due vie

Cache set-associativa a quattro vie

Cache set-associativa a otto vie (completamente associativa)



Associatività crescente – come si riempie [0 1 2 3 4 8]?

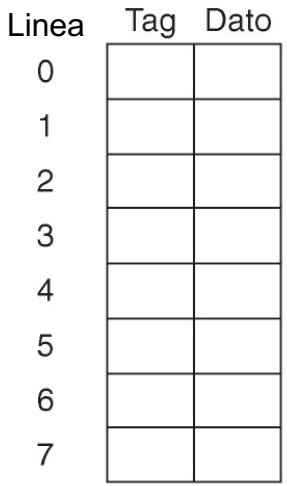
Cache con **8 blocchi** organizzati mediante diversi gradi di associatività

#vie = #linee per set

scelta del set è prefissata
#set = #blocco % S

se ho un solo set un blocco può essere posto in una linea qualsiasi

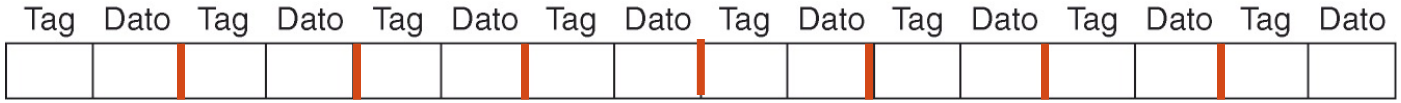
Cache set-associativa a una via (a mappatura diretta)



Cache set-associativa a due vie

Cache set-associativa a quattro vie

Cache set-associativa a otto vie (completamente associativa)

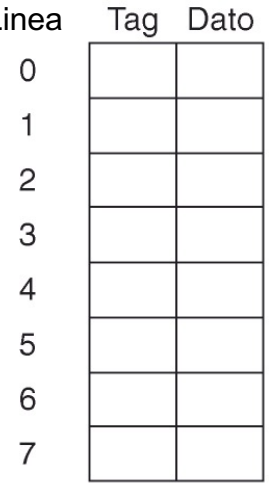


Associatività crescente

Cache con **8 blocchi** organizzati mediante diversi gradi di associatività.

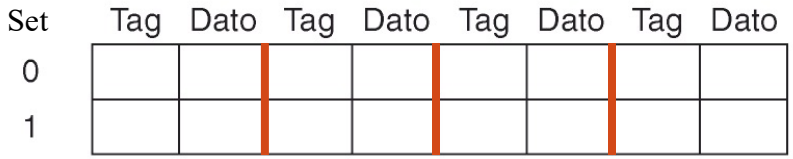
#set × #vie

Cache set-associativa a una via
(a mappatura diretta)



Cache set-associativa a due vie

Cache set-associativa a quattro vie

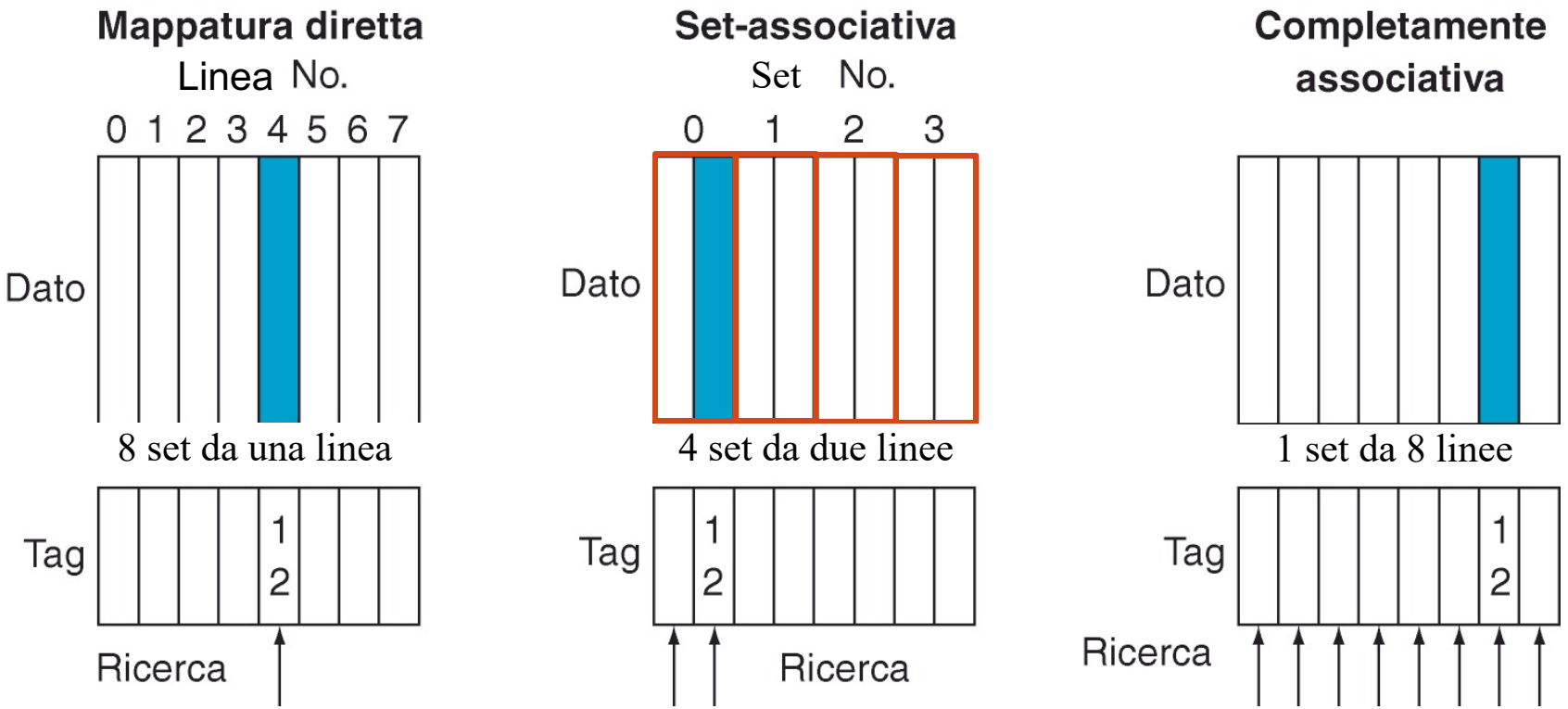


8×1 | 4×2 | 2×4

Associatività crescente

Immagine: David A. Patterson, John L. Hennessy, STRUTTURA E PROGETTO DEI CALCOLATORI, Zanichelli editore S.p.A. Copyright © 2015

Livelli di associatività



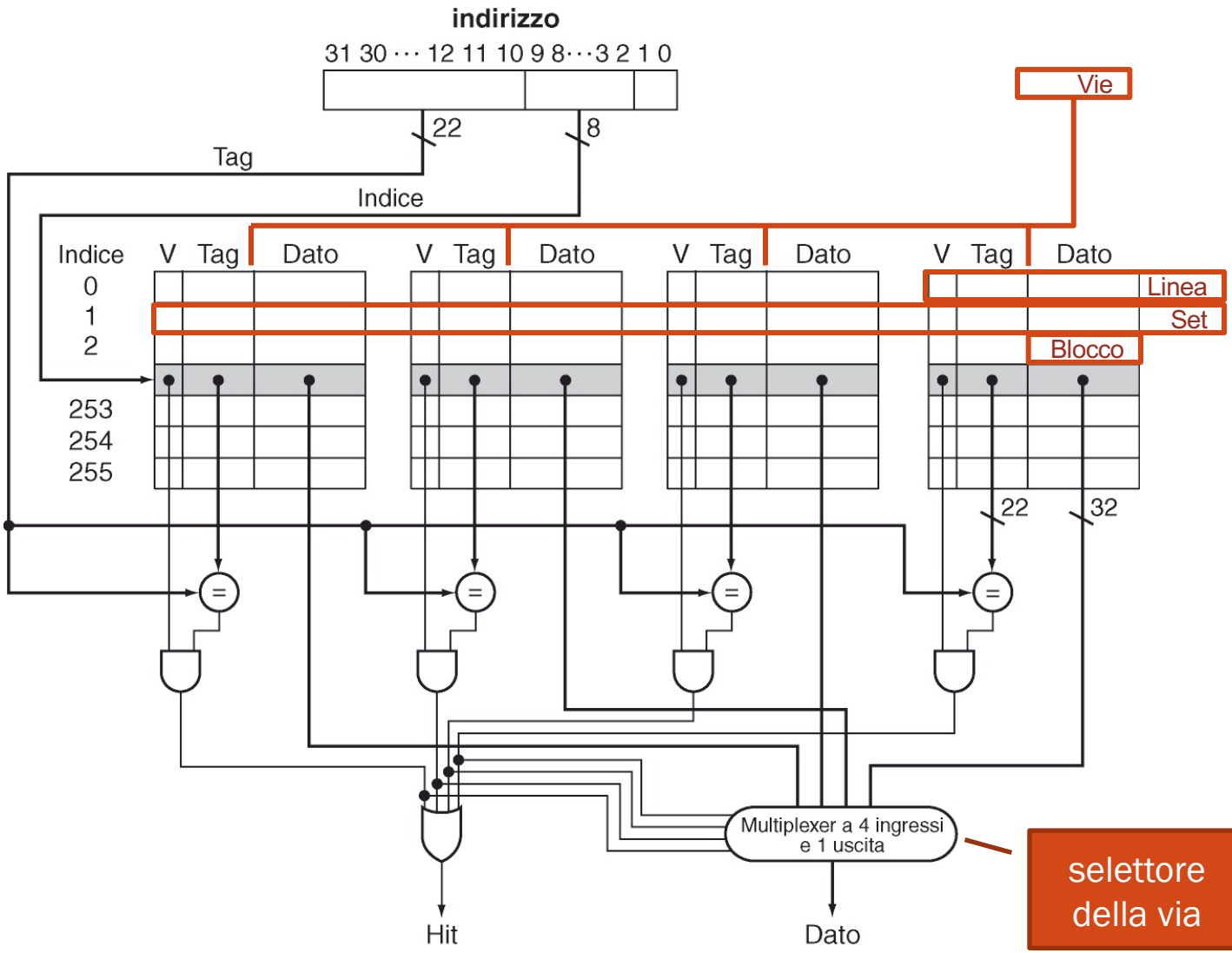
Associatività	Frequenza di miss
1	10,3%
2	8,6%
4	8,3%
8	8,1%

Cache set-associativa a 4 vie con blocco da 1 word

È necessario un comparatore per ciascuna via

Fully-associative

→ tanti comparatori quante sono le vie (invece di 4 comparatori me ne servono 1024)



Quanto è grande una cache? (update)

Per ogni **via** abbiamo una «tabella»

ogni tabella contiene S **set** con blocchi di N word

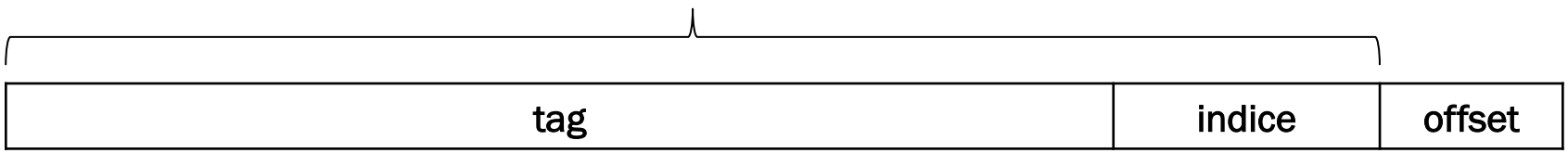
ogni linea contiene:

1 bit di **validità** (ed eventualmente i bit **Used** e **Dirty**)

il **tag**: 32 bit – bit di offset – bit di indice di linea (set)

il **blocco**: 8 [bit] × numero di byte = N x 32

I bit necessari per l'**offset** (di blocco e word) sono $\log_2(N \times 4)$ e per l'**indice** $\log_2(S)$



Esempio: cache a **4 vie**, con **8 set** e **blocchi da 4 word**

dimensione blocco = 4 word × 4 byte = 16 byte = 16 × 8 bit = **128 bit**

numero di bit per **offset** = $\log_2(4) + \log_2(4)$ = **4 bit**

numero di bit per **indice** = $\log_2(8)$ = **3 bit**

numero di bit per **tag** = 32 – 3 – 4 = **25 bit**

Dimensione della cache in bit:

4 vie × 8 set × (1 bit valid. + 128 bit blocco + 25 bit tag) = 32 × 154 = **4928 bit**

Politica di rimpiazzo della cache

Come scegliere quale blocco sostituire quando un set della cache è pieno?

Politica di rimpiazzo della cache

- **LRU** (Least Recently Used): sostituire il blocco **«più vecchio» (ultimo accesso)**
  - Approssimazione: si associa un bit (**Used**) a ciascuna linea
  - Ad ogni accesso si pone Used = 1
  - Lo si azzera allo scadere di un intervallo di tempo
    - Le linee con **Used = 1** sono **«più recenti»**
    - Le linee con **Used = 0** sono **«più vecchie»**
  - Può essere costoso mantenere questa informazione

- **LFU** (Least Frequently Used): sostituire il blocco **«meno usato»**
  - Si può associare un contatore a ciascun set ed aggiornarlo ad ogni accesso
  - Hardware necessario più complicato

- **Random**: sostituire un blocco **«a caso»** (per esempio in sequenza)

Tipi di MISS

Data una cache a **W vie** con **S set**,

un accesso alla cache può causare MISS per tre motivi:

1. **Cold start:** prima richiesta del dato/istruzione
2. **Conflict:** il blocco è stato sovrascritto per via del grado di associatività della cache
   - in una fully-associative cache, se richiesto dopo meno di W × S altri blocchi, sarebbe una HIT
3. **Capacity:** il blocco è stato sovrascritto per via della capienza della cache
   - anche in una fully-associative cache, se richiesto dopo più di W × S altri blocchi, causerebbe comunque una MISS

Quali parametri della cache hanno influenza sui tre tipi di MISS?

1. **Cold start:** la **dimensione del blocco** — prossimità spaziale
   - blocchi più grandi contengono più dati/istruzioni ⟶ il numero di cold miss diminuisce
2. **Conflict:** il **grado di associatività**
   - cache con più vie ⟶ meno conflitti
3. **Capacity:** la **dimensione della cache**
   - maggior numero di linee ⟶ maggior numero di richieste gestibili

Politiche di scrittura

Come aggiornare la memoria principale (RAM) quando il dato è modificato in cache?

Politiche di scrittura

- **Write through:** **ad ogni modifica** viene aggiornato il blocco in memoria (RAM)
  - **Pro:** in presenza di cache multiple la coerenza dei dati viene mantenuta
  - **Contro:** per la località degli accessi se avvengono più scritture nello stesso blocco si perde molto tempo
- **Write back:** il blocco viene aggiornato **solo quando viene sostituito**
  - **Pro:** la scrittura del blocco in memoria avviene raramente (è un blocco «vecchio»)
    → cache molto più veloce
  - **Contro:** il contenuto della cache non è più coerente con quello della memoria principale
    → complicando i sistemi multiprocessore e multicache

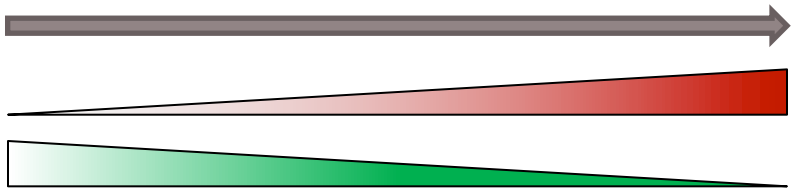
Ottimizzazione: con il bit **Dirty** (modificato) si può evitare di scrivere sulla RAM i blocchi non modificati

Influenza dei parametri della cache

Dimensione del blocco

↑ Tempo di MISS (trasferimento da RAM)

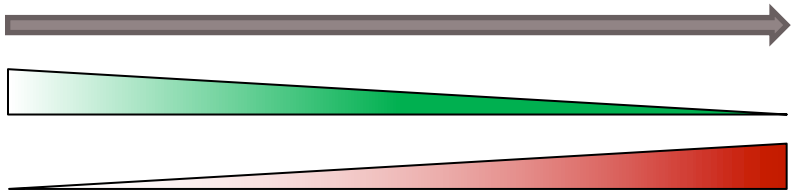
↓ Percentuale di MISS di tipo **cold start**



Dimensione della cache (numero di linee)

↓ Percentuale di MISS di **capacità**

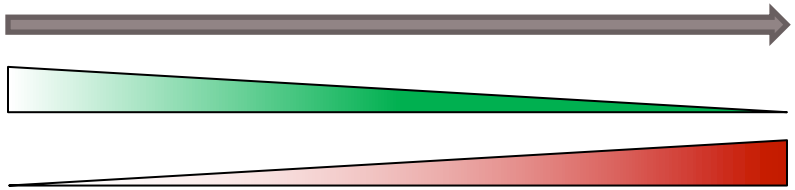
↑ Costo (più memoria)



Associatività (numero di vie)

↓ Percentuale di MISS di **conflitto**

↑ Costo (più comparatori)



IDEA: migliorare le performance del sistema usando più livelli di cache ottimizzate con parametri diversi (e costi diversi)

Diversa associatività

Sia data la sequenza di accessi a memoria: (0, 8, 0, 6, 8)

Calcolare il numero di HIT e MISS date le cache con

1. direct mapping:
   - 4 linee di blocchi da 1 word
2. set-associative mapping:
   - 2 vie, 2 set (4 linee) con blocchi da 1 word
   - politica di rimpiazzo LRU
3. fully-associative mapping:
   - 4 vie, 1 set (4 linee) con blocchi da 1 word
   - politica di rimpiazzo LRU

Diversa associatività

Sia data la sequenza di accessi a memoria: (0, 8, 0, 6, 8)

1 via, 4 set

MISS = 100%

Blocchi da 1 word, 4 linee, direct mapped					
#blocco	0	8	0	6	8
index	0	0	0	2	0
tag	0	2	0	1	2
HIT/MISS	MISS (cold start)	MISS (cold start)	MISS	MISS (cold start)	MISS

2 vie, 2 set

MISS = 80%

Blocchi da 1 word, 2 set, 2 vie con rimpiazzo LRU					
#blocco	0	8	0	6	8
index	0	0	0	0	0
tag	0	4	0	3	4
HIT/MISS	MISS (cold start)	MISS (cold start)	HIT	MISS (cold start)	MISS

4 vie, 1 set

MISS = 60%

Blocchi da 1 word, 1 set, 4 vie con rimpiazzo LRU					
#blocco	0	8	0	6	8
tag	0	8	0	6	8
HIT/MISS	MISS (cold start)	MISS (cold start)	HIT	MISS (cold start)	HIT

Diversa associatività

Sia data la sequenza di accessi a memoria: (0, 8, 0, 6, 8)

1 via, 4 set

MISS = 100%

Blocchi da 1 word, 4 linee, direct mapped					
#blocco	0	8	0	6	8
index	0	0	0	2	0
tag	0	2	0	1	2
HIT/MISS	MISS (cold start)	MISS (cold start)	MISS (conflict)	MISS (cold start)	MISS (conflict)

2 vie, 2 set

MISS = 80%

Blocchi da 1 word, 2 set, 2 vie con rimpiazzo LRU					
#blocco	0	8	0	6	8
index	0	0	0	0	0
tag	0	4	0	3	4
HIT/MISS	MISS (cold start)	MISS (cold start)	HIT	MISS (cold start)	MISS (conflict)

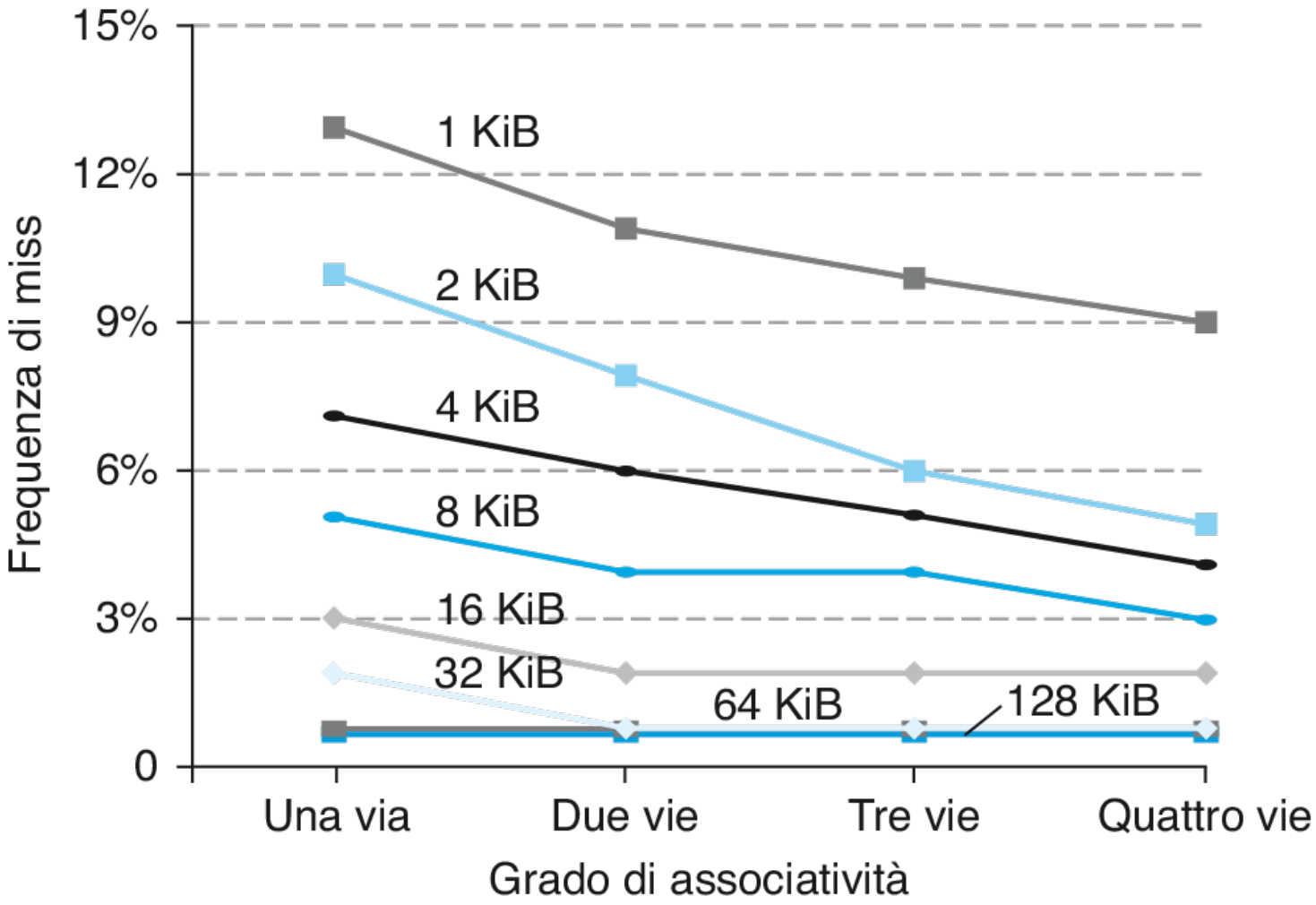
sarebbe MISS (capacity) se fosse MISS anche per una fully associative cache della stessa dimensione

4 vie, 1 set

MISS = 60%

Blocchi da 1 word, 1 set, 4 vie con rimpiazzo LRU					
#blocco	0	8	0	6	8
tag	0	8	0	6	8
HIT/MISS	MISS (cold start)	MISS (cold start)	HIT	MISS (cold start)	HIT

Miss rate v. grado di associatività



Cache multi-livello

In una memoria multi-livello:

- ogni **HIT** fornisce i dati a tutti i livelli superiori
- solo le **MISS** fanno richieste al livello subito inferiore
- ogni accesso impiega il tempo del livello più basso che ha risposto

Esempio di calcolo dei tempi se supponiamo che:

- HIT su L1: **2 ns**
- HIT su L2: **30 ns**
- accesso a RAM (ovvero MISS su L2 e HIT su RAM): **100 ns**
- clock della CPU da **2 GHz** (2 CC per ns) e **3 CPI** (CC per istr.)

Dati: perc. di MISS su L1 = 15% → % HIT su L1 = 85%

perc. di MISS su L2 = 20% → % HIT su L2 = 80%

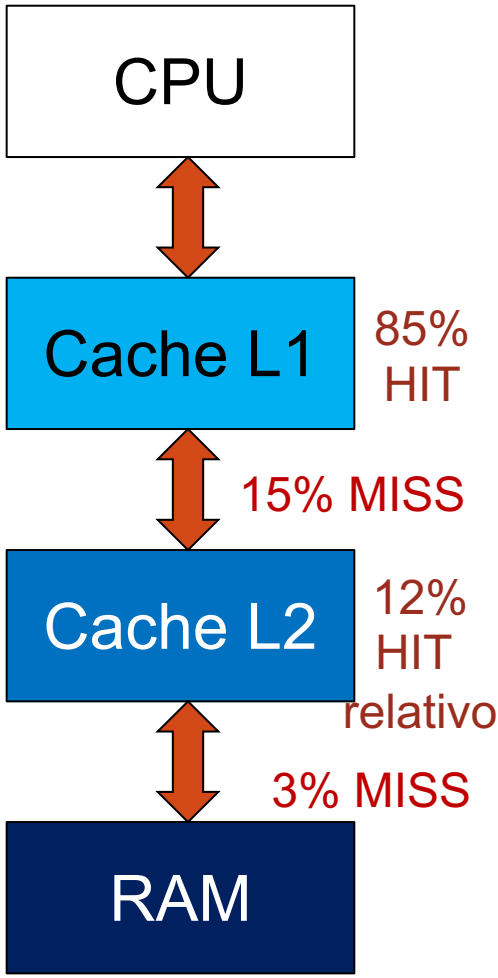
Il **tempo medio** di un **accesso in memoria** sarà

$0{,}85 \times 2\ ns + 0.15 \times 0.8 \times 30\ ns + 0.15 \times 0.2 \times 100\ ns =$

$= 1.7ns + 3.6ns + 3\ ns = \mathbf{8.3\ ns}$

Ovvero $8.3 \times 2 = \mathbf{16.6}$ clock per accesso (1 ogni 5.5 istruzioni)

Con la sola L1: $0.85 \times 2\ ns + 0.15 \times 100\ ns = 1.7\ ns + 15\ ns = \mathbf{16.7\ ns}$ (2 volte più lento)



Esercizio

Per una cache con:

- blocchi da **8** word = 32 byte = 256 bit
- **2** set
- **4** vie
- strategia di rimpiazzo **LRU**

1) Calcolare la **dimensione in bit** della cache

2) Calcolare nella sequenza di accessi in basso **quali siano HIT e quali siano MISS**

address	130	310	1800	150	519	342	1820	127	529	2000	325	516	1794	317
#blocco														
Indice linea														
tag														
HIT/MISS														

Esercizio

Per una cache con:

- blocchi da **8** word = 32 byte = 256 bit
- **2** set
- **4** vie
- strategia di rimpiazzo **LRU**

1) Calcolare la **dimensione in bit** della cache

Offset = $\log_2(32)$ = **5 bit** **Index** = $\log_2(2)$ = **1 bit**

Tag = 32 – offset – index = 32 – 5 – 1 = **26 bit**

DIM = 4 vie × 2 set × (tag + valid. + used + blocco) = 8 × (26+1+1+256) = 8×284 = **2272 bit**

2) Calcolare nella sequenza di accessi in basso **quali siano HIT e quali siano MISS**

address	**130**	**310**	**1800**	**150**	**519**	**342**	**1820**	**127**	**529**	**2000**	**325**	**516**	**1794**	**317**
#blocco	#blocco = ⌊ address / dim. blocco in byte ⌋ (/32)													
indice linea														
tag														
HIT/MISS														

$2^{3+2} = 2^3 \times 4$

Esercizio

Per una cache con:

- blocchi da **8** word = 32 byte = 256 bit
- **2** set
- **4** vie
- strategia di rimpiazzo **LRU**

1) Calcolare la **dimensione in bit** della cache

Offset = $\log_2(32)$ = **5 bit** **Index** = $\log_2(2)$ = **1 bit**

Tag = 32 – offset – index = 32 – 5 – 1 = **26 bit**

DIM = 4 vie × 2 set × (tag + valid. + used + blocco) = 8 × (26+1+1+256) = 8×284 = **2272 bit**

2) Calcolare nella sequenza di accessi in basso **quali siano HIT e quali siano MISS**

address	**130**	**310**	**1800**	**150**	**519**	**342**	**1820**	**127**	**529**	**2000**	**325**	**516**	**1794**	**317**
#blocco	**4**	**9**	**56**	**4**	**16**	**10**	**56**	**3**	**16**	**62**	**10**	**16**	**56**	**9**
indice linea	indice linea = #blocco % #set (%2)													
tag	tag = #blocco / #set (/2)													
HIT/MISS														

Esercizio

Per una cache con:

- blocchi da **8** word = 32 byte = 256 bit
- **2** set
- **4** vie
- strategia di rimpiazzo **LRU**

1) Calcolare la **dimensione in bit** della cache

Offset = $\log_2(32)$ = **5 bit** **Index** = $\log_2(2)$ = **1 bit**

Tag = 32 – offset – index = 32 – 5 – 1 = **26 bit**

DIM = 4 vie × 2 set × (tag + valid. + used + blocco) = 8 × (26+1+1+256) = 8×284 = **2272 bit**

2) Calcolare nella sequenza di accessi in basso **quali siano HIT e quali siano MISS**

address	**130**	**310**	**1800**	**150**	**519**	**342**	**1820**	**127**	**529**	**2000**	**325**	**516**	**1794**	**317**
#blocco	**4**	**9**	**56**	**4**	**16**	**10**	**56**	**3**	**16**	**62**	**10**	**16**	**56**	**9**
indice linea	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**
tag	**2**	**4**	**28**	**2**	**8**	**5**	**28**	**1**	**8**	**31**	**5**	**8**	**28**	**4**
HIT/MISS	HIT se stesso tag e stesso indice entro 4 (vie)													

Esercizio

Per una cache con:

- blocchi da **8** word = 32 byte = 256 bit
- **2** set
- **4** vie
- strategia di rimpiazzo **LRU**

1) Calcolare la **dimensione in bit** della cache

Offset = $\log_2(32)$ = **5 bit** **Index** = $\log_2(2)$ = **1 bit**

Tag = 32 – offset – index = 32 – 5 – 1 = **26 bit**

DIM = 4 vie × 2 set × (tag + valid. + used + blocco) = 8 × (26+1+1+256) = 8×284 = **2272 bit**

2) Calcolare nella sequenza di accessi in basso **quali siano HIT e quali siano MISS**

address	130	310	1800	150	519	342	1820	127	529	2000	325	516	1794	317
#blocco	4	9	56	4	16	10	56	3	16	62	10	16	56	9
indice linea	0	1	0	0	0	0	0	1	0	0	0	0	0	1
tag	2	4	28	2	8	5	28	1	8	31	5	8	28	4
HIT/MISS	M	M	M	H	M	M	H	M	H	M	H	H	H	H

Esercizio

Per una cache con:

- blocchi da **8** word = 32 byte = 256 bit
- **2** set
- **4** vie
- strategia di rimpiazzo **LRU**

1) Calcolare la **dimensione in bit** della cache

Offset = $\log_2(32)$ = **5 bit** **Index** = $\log_2(2)$ = **1 bit**

Tag = 32 – offset – index = 32 – 5 – 1 = **26 bit**

DIM = 4 vie × 2 set × (tag + valid. + used + blocco) = 8 × (26+1+1+256) = 8×284 = **2272 bit**

2) Calcolare nella sequenza di accessi in basso **quali siano HIT e quali siano MISS**

address	130	310	1800	150	519	342	1820	127	529	2000	325	516	1794	317
#blocco	4	9	56	4	16	10	56	3	16	62	10	16	56	9
indice linea	0	1	0	0	0	0	0	1	0	0	0	0	0	1
tag	2	4	28	2	8	5	28	1	8	31	5	8	28	4
HIT/MISS	M	M	M	H	M	M	H	M	H	M	H	H	H	H

blocchi grandi riducono cold start

Esercizio con cache multilivello

SAPIENZA
UNIVERSITÀ DI ROMA

Esempio

Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 linee** (set)

Cache **L2** con blocchi da **16 word**, **2 vie** con **8 set** per via e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#														
	index														
	tag														
	H/M														
L2	block#														
	index														
	tag														
	H/M														

#blocco = $\lfloor$ address / dim. blocco $\rfloor$ (/8)

Esempio

Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 linee** (set)

Cache **L2** con blocchi da **16 word**, **2 vie** con **8 set** per via e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index														
	tag														
	H/M														
L2	block#														
	index														
	tag														
	H/M														

#blocco = $\lfloor$ address / dim. blocco $\rfloor$ (/8)

indice linea = #blocco % #set (%4)

Esempio

Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 linee** (set)

Cache **L2** con blocchi da **16 word**, **2 vie** con **8 set** per via e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag														
	H/M														
L2	block#														
	index														
	tag														
	H/M														

#blocco = $\lfloor$ address / dim. blocco $\rfloor$	(/8)
indice linea = #blocco % #set	(%4)
tag = #blocco / #set	(/4)

Esempio

Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 linee** (set)

Cache **L2** con blocchi da **16 word**, **2 vie** con **8 set** per via e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M														
L2	block#														
	index														
	tag														
	H/M														

#blocco = ⌊ address / dim. blocco ⌋ (/8)

indice linea = #blocco % #set (%4)

tag = #blocco / #set (/4)

HIT se e solo se: stesso tag e stesso indice entro 1 (via)

?	Qual è il block offset in L1 dato l'indirizzo 531?	(531 % 8) = 3
?	Qual è il word offset (nel blocco) in L1 dato l'indirizzo 531?	(531 % 8) / 4 = 0

Esempio

Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 linee** (set)

Cache **L2** con blocchi da **16 word**, **2 vie** con **8 set** per via e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M ?	M ?	M ?	M cold
L2	block#														
	index														
	tag														
	H/M														

Esempio

Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 linee** (set)

Cache **L2** con blocchi da **16 word**, **2 vie** con **8 set** per via e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M	M	M	M cold
L2	block#														
	index														
	tag														
	H/M														

Cache L1 fully-associative (1×4)			

I MISS in L1 sono di tipo conflict o capacity?

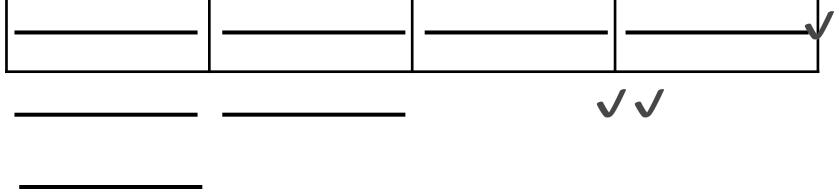
Esempio

Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 linee** (set)

Cache **L2** con blocchi da **16 word**, **2 vie** con **8 set** per via e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M	M	M	M cold
L2	block#														
	index														
	tag														
	H/M														

Cache L1 fully-associative (1×4)



- **137** causa un miss anche con cache fully associative (a parità di dimensioni di cache)
  - Quindi si ha un **capacity miss.**
- Lo stesso avviene con **11**.
  - Quindi si ha un **capacity miss.**
- **39** è presente con cache fully associative (a parità di dimensioni di cache).
  - Quindi si ha un **conflict miss.**

Esempio

Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 linee** (set)

Cache **L2** con blocchi da **16 word**, **2 vie** con **8 set** per via e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M cap	M cap	M conf	M cold
L2	block#														
	index														
	tag														
	H/M														

Cache L1 fully-associative (1×4)			
~~140~~	~~66~~	~~137~~	~~257~~ ✓
~~64~~	~~11~~	39 ✓✓	11
~~12~~	137		
13			

- **137** causa un miss anche con cache fully associative (a parità di dimensioni di cache)
  - Quindi si ha un **capacity miss.**
- Lo stesso avviene con **11**.
  - Quindi si ha un **capacity miss.**
- **39** è presente con cache fully associative (a parità di dimensioni di cache).
  - Quindi si ha un **conflict miss.**

Esempio

Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 linee** (set)

Cache **L2** con blocchi da **16 word**, **2 vie** con **8 set** per via e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M cap	M cap	M conf	M cold
L2	block#														
	index														
	tag														
	H/M														

#blocco = ⌊ address / dim. blocco ⌋ (/64)

Esempio

Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 linee** (set)

Cache **L2** con blocchi da **16 word**, **2 vie** con **8 set** per via e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M cap	M cap	M conf	M cold
L2	block#	17	8	17	32	8	1		4	1		17	1	4	1
	index														
	tag														
	H/M														

#blocco = ⌊ address / dim. blocco ⌋ (/64)

indice linea = #blocco % #set (%8)

Esempio

Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 linee** (set)

Cache **L2** con blocchi da **16 word**, **2 vie** con **8 set** per via e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M cap	M cap	M conf	M cold
L2	block#	17	8	17	32	8	1		4	1		17	1	4	1
	index	1	0	1	0	0	1		4	1		1	1	4	1
	tag														
	H/M														

#blocco = $\lfloor$ address / dim. blocco $\rfloor$	(/64)
indice linea = #blocco % #set	(%8)
tag = #blocco / #set	(/8)

Esempio

Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 linee** (set)

Cache **L2** con blocchi da **16 word**, **2 vie** con **8 set** per via e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M cap	M cap	M conf	M cold
L2	block#	17	8	17	32	8	1		4	1		17	1	4	1
	index	1	0	1	0	0	1		4	1		1	1	4	1
	tag	2	1	2	4	1	0		0	0		2	0	0	0
	H/M	M cold	M cold	HIT	M cold	HIT	M cold		M cold	HIT		HIT	HIT	HIT	HIT

#blocco = $\lfloor$ address / dim. blocco $\rfloor$ (/64)

indice linea = #blocco % #set (%8)

tag = #blocco / #set (/8)

HIT se e solo se: stesso tag e stesso indice entro 2 (vie)

Esempio

Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 linee** (set)

Cache **L2** con blocchi da **16 word**, **2 vie** con **8 set** per via e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M cap	M cap	M conf	M cold
L2	block#	17	8	17	32	8	1		4	1		17	1	4	1
	index	1	0	1	0	0	1		4	1		1	1	4	1
	tag	2	1	2	4	1	0		0	0		2	0	0	0
	H/M	M cold	M cold	HIT	M cold	HIT	M cold		M cold	HIT		HIT	HIT	HIT	HIT

Il tempo impiegato è dovuto a 2 L1 HIT, 7 L2 HIT e 5 L2 MISS

Avendo L1 HIT = 1 ns, L2 HIT = 10 ns, L2 MISS = 100 ns e clock = 1GHz con 1 CPI

Tempo totale: $2 \times 1\text{ ns} + 7 \times 10\text{ ns} + 5 \times 100\text{ ns} = 572\text{ ns}$

Tempo medio: $572\text{ ns} / 14 = 40.8\text{ ns}$ ovvero 40.8 colpi di clock medi per accesso

NOTA: si potrebbe eliminare la MISS di conflitto cambiando l'associatività di L1

Cambio di setup

Cambiando il setup della cache L1 (ma mantenendo la stessa dimensione totale in blocchi):

- blocco da 2 a 4 word (16 byte) — per **diminuire** i **MISS** da **cold start**
- da 1 a 2 vie — per **diminuire** i **MISS** da **conflict/capacity**
- da 4 a 2 set — per mantenere il num. totale di blocchi

													M (cap)	M (conf)	M (cold)
	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	**block#**	70	33	68	128	32	5	128	19	6	19	68	5	19	6
	index	0	1	0	0	0	1	0	1	0	1	0	1	1	0
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M cap	HIT	HIT	HIT
L2	**block#**	17	8	17	32	8	1		4	1		17			
	index	1	0	1	0	0	1		4	1		1			
	tag	2	1	2	4	1	0		0	0		2			
	H/M	M cold	M cold	HIT	M cold	HIT	M cold		M cold	HIT		HIT			

Totale: 5 L1 HIT + 4 L2 HIT + 5 L2 MISS

Tempo: 5 × 1 ns + 4 × 10 ns + 5 × 100 ns = 545 ns (invece di 572ns: **quasi 5% in meno**)

Esercizio d'esame (appello 2020-07-06)

Sia data una CPU con processore a **2GHz** e **4 CPI** (Clock per Instruction) che adoperi indirizzi da 32 bit e memoria strutturata su due livelli di **cache** (L1, L2), il cui setup è come segue:
L1 è una cache **set-associativa** a **4 vie** con **2 set** (linee) e **blocchi da 4 word**; adopera una politica di rimpiazzo **LRU**. Ricordiamo che consideriamo la linea come l'insieme del blocco in cache con tag e bit di controllo, mentre il set è l'insieme di linee con il medesimo indice.
L2 è una cache **direct-mapped** con **8 linee** e **blocchi da 16 word**.

1) Supponendo che all'inizio nessuno dei dati sia in cache, indicare quali degli accessi in memoria indicati di seguito sono HIT o MISS in ciascuna delle due cache. Per ciascuna **MISS** indicare se sia di tipo Cold Start (**Cold**), Capacità (**Cap**) o Conflitto (**Conf**). Utilizzare la tabella sottostante per fornire i risultati ed indicare la metodologia di calcolo più in basso.

	Address	180	184	178	675	984	540	184	803	798	466	112	790
L1	Block#												
	Index												
	Tag												
	HIT/MISS												
	Miss type												
L2	Block#												
	Index												
	Tag												
	HIT/MISS												
	Miss type												

2) Calcolare le dimensioni in bit (compresi i bit di controllo ed assumendo che ne basti uno per la LRU) delle due cache: (a) L1 e (b) L2.

3) Assumendo che gli accessi in **memoria** impieghino **200 ns**, che gli **hit** nella cache **L1** impieghino **1 ns** e gli **hit** nella cache **L2** impieghino **20 ns**, calcolare (a) il **tempo totale** per la sequenza di accessi, (b) il tempo **medio** per la sequenza di accessi, e (c) **quante istruzioni** vengono svolte nel tempo medio calcolato.

Esercizio d'esame (appello 2020-07-06)

	Address	180	184	178	675	984	540	184	803	798	466	112	790
	Block#	11	11	11	42	61	33	11	50	49	29	7	49
	Index	1	1	1	0	1	1	1	0	1	1	1	1
	Tag	5	5	5	21	30	16	5	25	24	14	3	24
	HIT/MISS	MISS	HIT	HIT	MISS	MISS	MISS	HIT	MISS	MISS	MISS	MISS	HIT
L1	**Miss type**	cold			cold	cold	cold		cold	cold	cold	cold	

L1		×	
Num set	2		
Blocco da	4	4	byte
Num vie	4		
L2			
Num set	8		
Blocco da	16	4	byte
Num vie	1		

Esercizio d'esame (appello 2020-07-06)

	Address	180	184	178	675	984	540	184	803	798	466	112	790
	Block#	11	11	11	42	61	33	11	50	49	29	7	49
	Index	1	1	1	0	1	1	1	0	1	1	1	1
	Tag	5	5	5	21	30	16	5	25	24	14	3	24
	HIT/MISS	MISS	HIT	HIT	MISS	MISS	MISS	HIT	MISS	MISS	MISS	MISS	HIT
L1	**Miss type**	cold			cold	cold	cold		cold	cold	cold	cold	
	Block#	2			10	15	8		12	12	7	1	
	Index	2			2	7	0		4	4	7	1	
	Tag	0			1	1	1		1	1	0	0	
	HIT/MISS	MISS			MISS	MISS	MISS		MISS	HIT	MISS	MISS	
L2	**Miss type**	cold			cold	cold	cold		cold		cold	cold	

L1		×	
Num set	2		
Blocco da	4	4	byte
Num vie	4		
L2			
Num set	8		
Blocco da	16	4	byte
Num vie	1		
Freq	2	GHz	
CPI	4	CPI	
HIT L1	1	ns	
HIT L2	20	ns	
MEM	200	ns	

Tempi

Totale	1424
Medio	118.67
Ist. es.	59.33

Dimensioni (bit)

L1	
Used	1
Valid	1
Dirty	1
Offset	4
Index	1
Tag	27
Blocco	128
Totale	**1248**
Tot (-V)	**1240**
Tot (+U/D)	**1256**
Tot (+U+D)	**1264**
L2	
Used	1
Valid	1
Dirty	0
Offset	6
Index	3
Tag	23
Blocco	512
Totale	**4288**
Tot (-V)	**4280**
Tot (+D)	**4296**