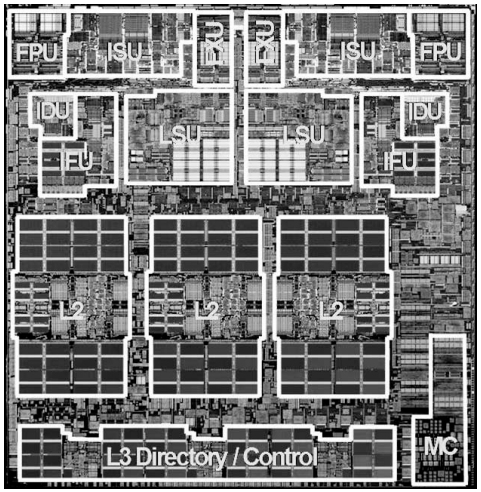
Università degli Studi di Cassino



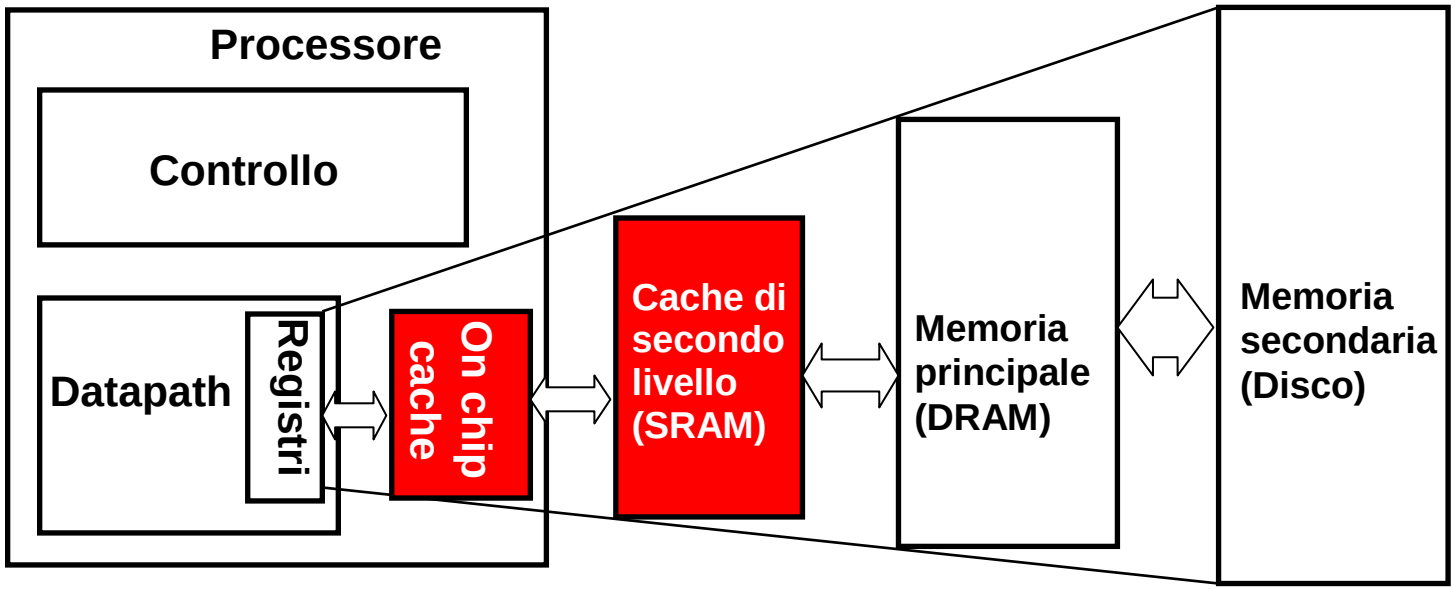
Corso di Calcolatori Elettronici II

Cache

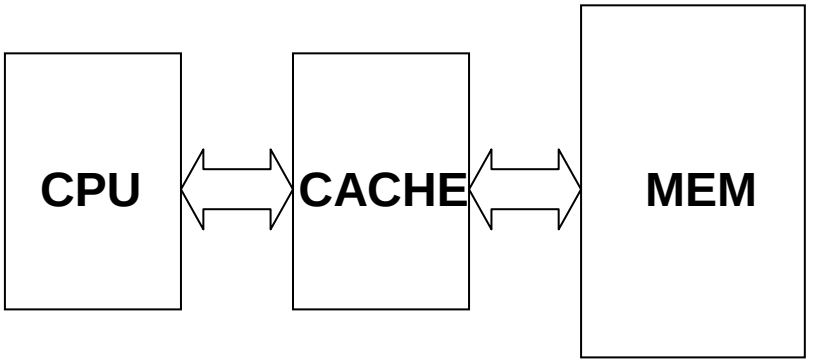
Anno Accademico 2007/2008

Francesco Tortorella

Cache: 1°livello della gerarchia di memoria



4 decisioni da prendere



1. Dove posizionare un blocco ?
2. Come reperire un blocco ?

} **Tecnica di indirizzamento**

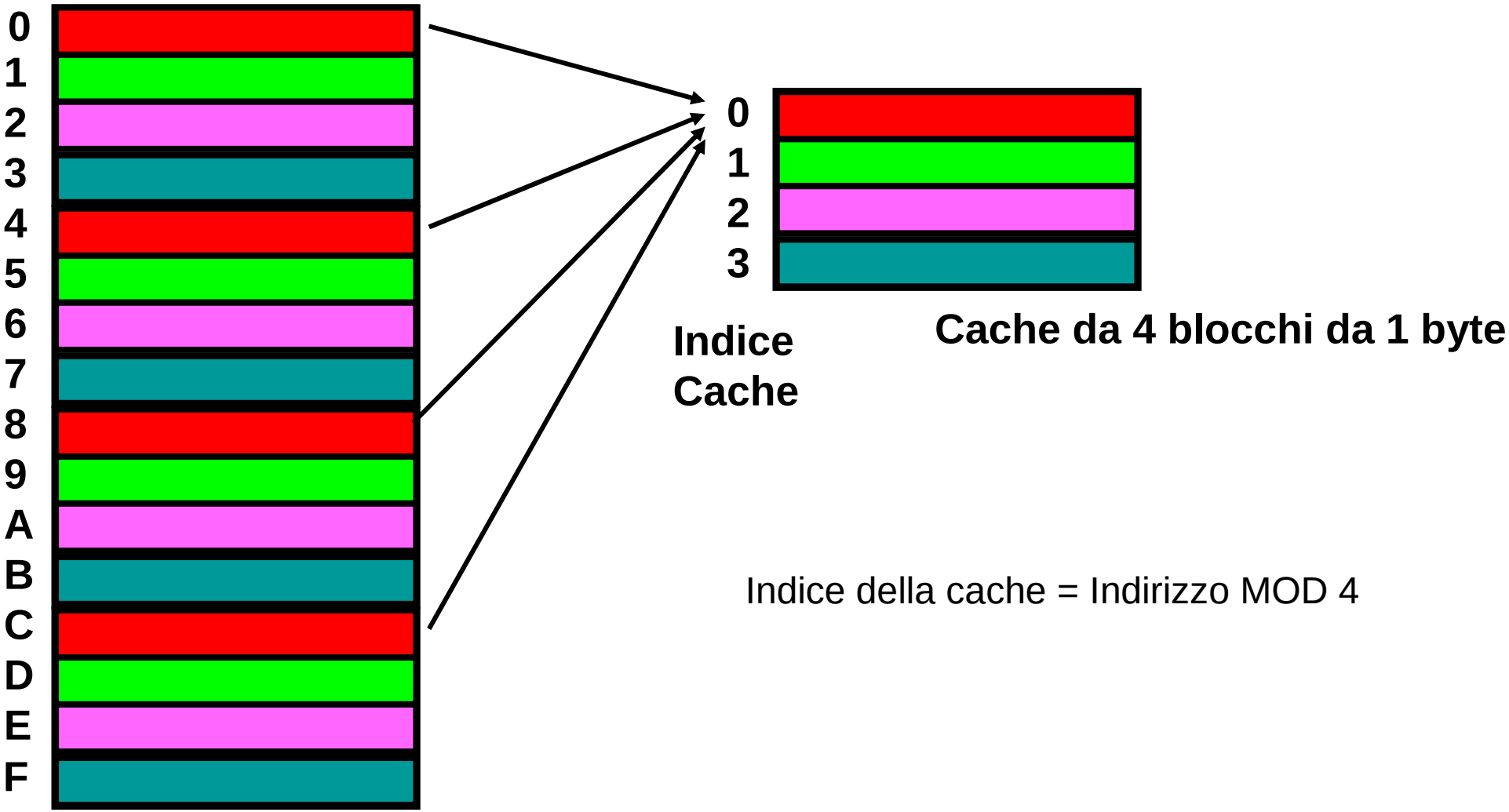
3. Quale blocco sostituire in corrispondenza di un miss ?

Algoritmo di sostituzione

4. Come gestire un'operazione di scrittura ?

Strategia di aggiornamento

Cache ad accesso diretto (Direct Mapped Cache)



Indice della cache = Indirizzo MOD 4

Problemi dell'accesso diretto

- Quale blocco di memoria è presente nella cache ?
- Come gestire un blocco formato da più di 1 byte ?

⇒ Si divide l'indirizzo di memoria in 3 campi:
tag, index, e byte offset

index	riporta l'indice del blocco all'interno della cache;
offset	riporta lo spiazzamento dei dati di interesse all'interno del blocco
tag	riferisce se il dato al blocco indirizzato nella cache corrisponde all'indirizzo desiderato

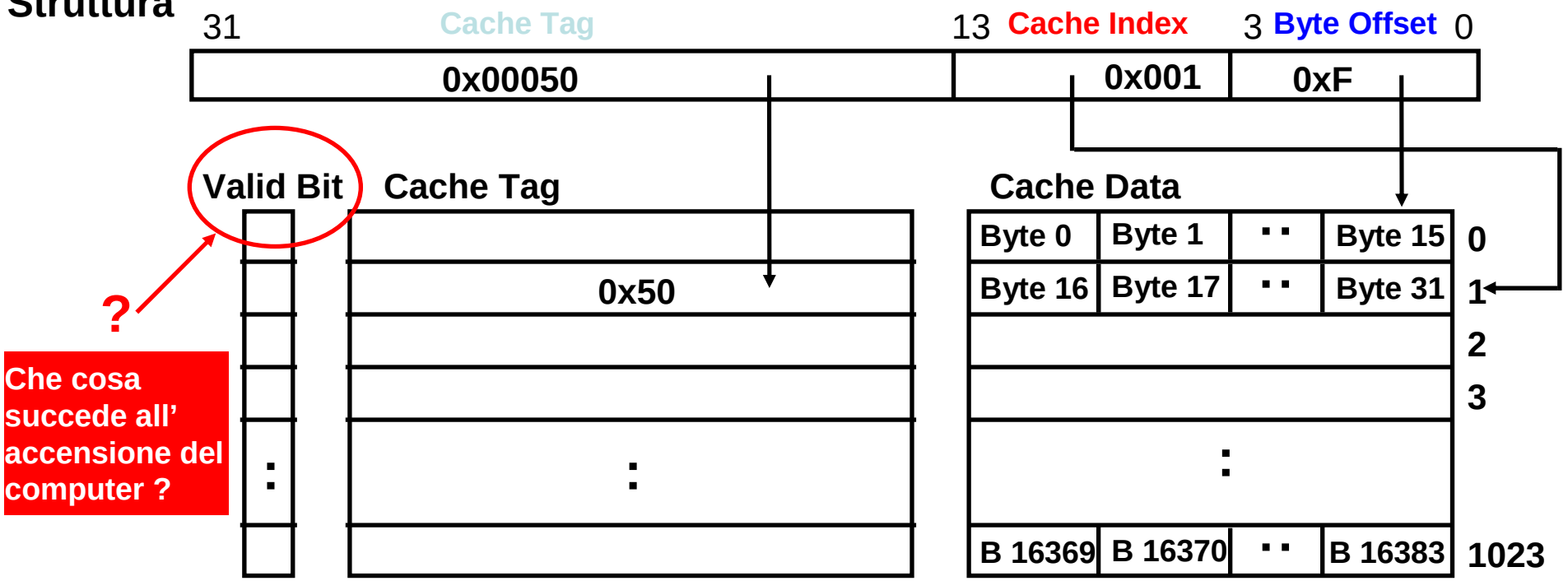
Esempio: cache a.d. da 16 KB (blocchi da 16 B)

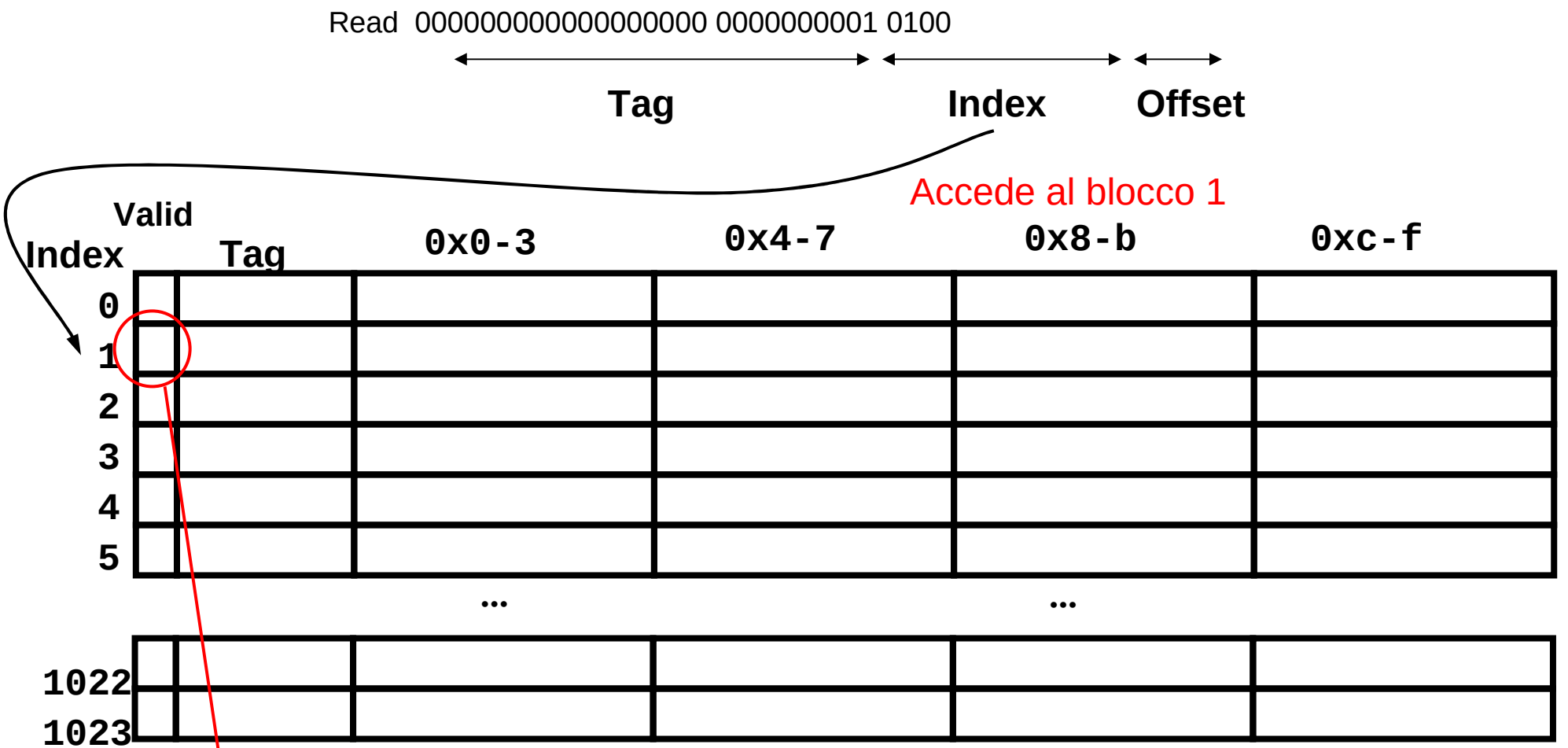
- dimensione dell'indirizzo di memoria: 32 bit
- dimensione della cache: 2^N bytes (N=14)
- dimensione del blocco (linea): 2^M bytes (M=4)
  - gli N bit meno significativi identificano il byte nella cache
  - i 32-N bit più significativi costituiscono il tag

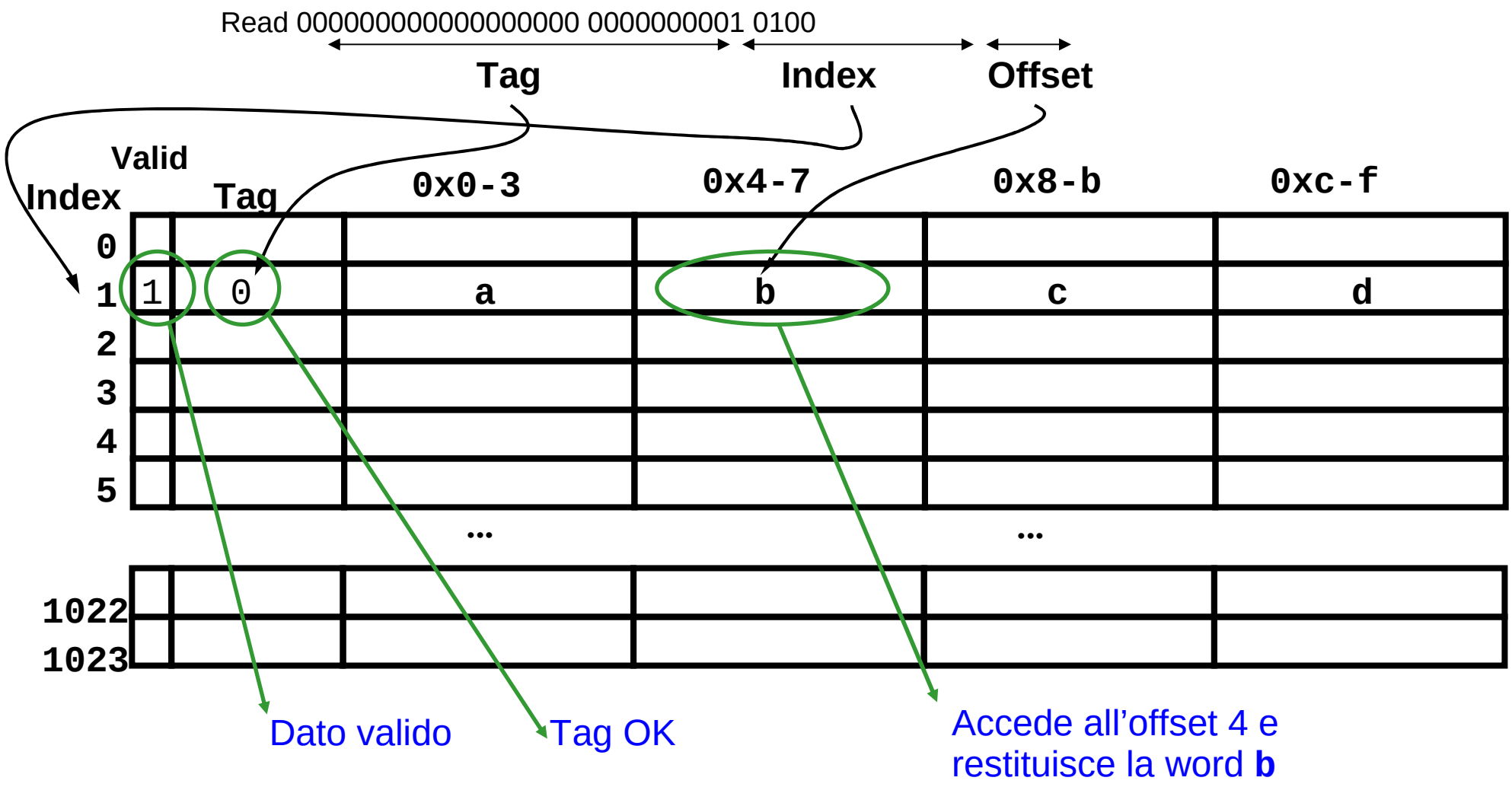
M per l'offset

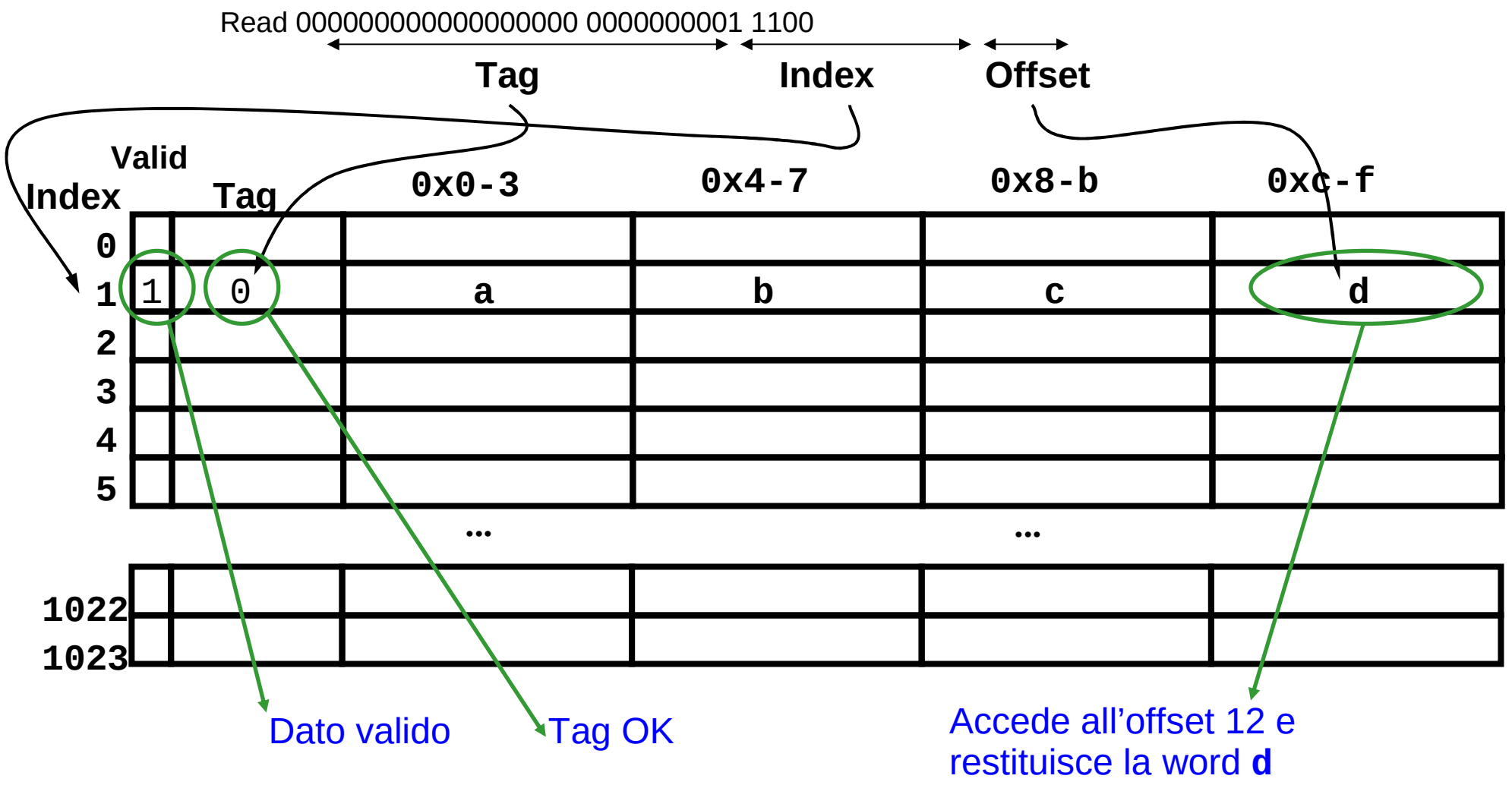
N-M per l'index

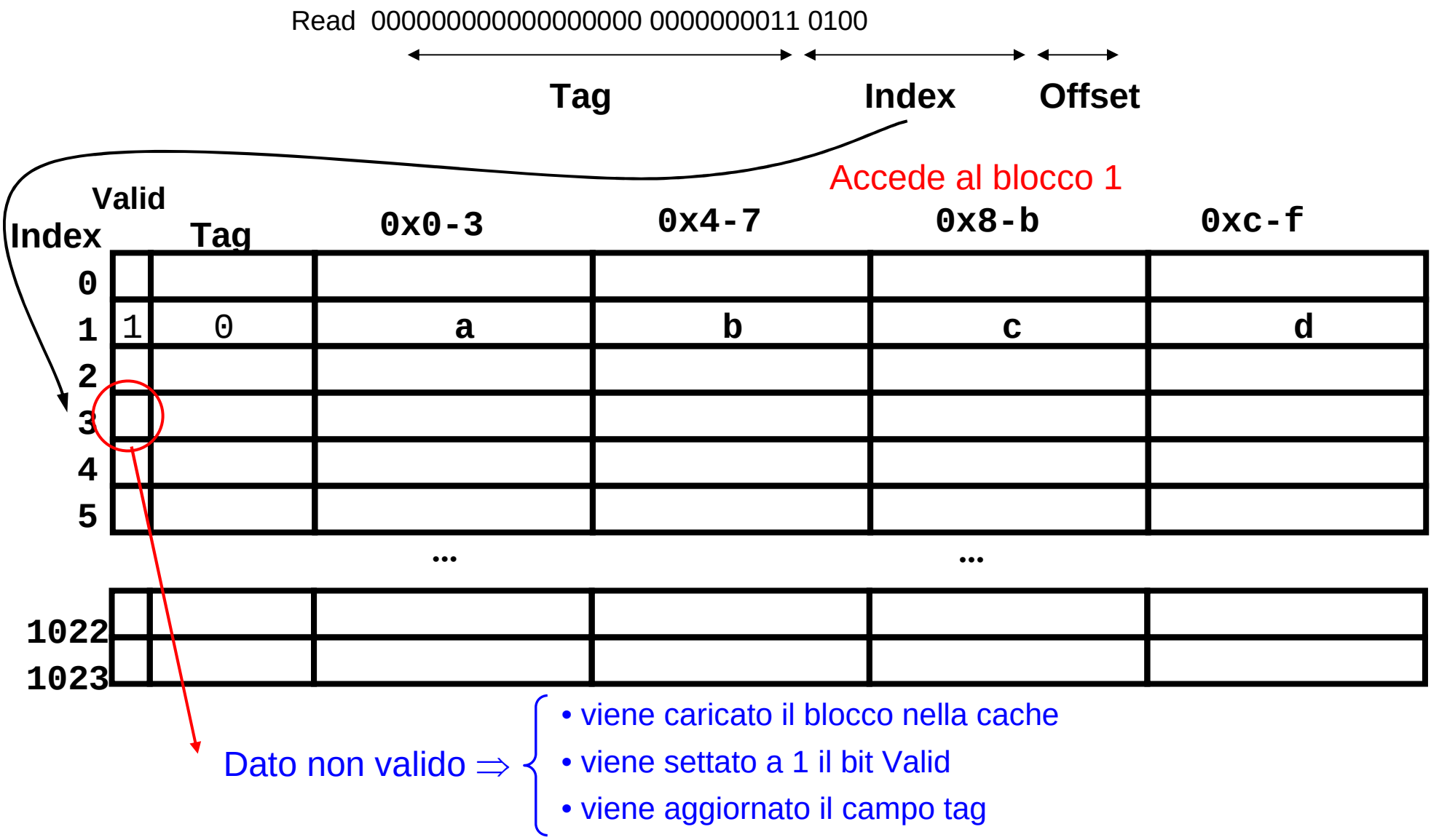
Struttura

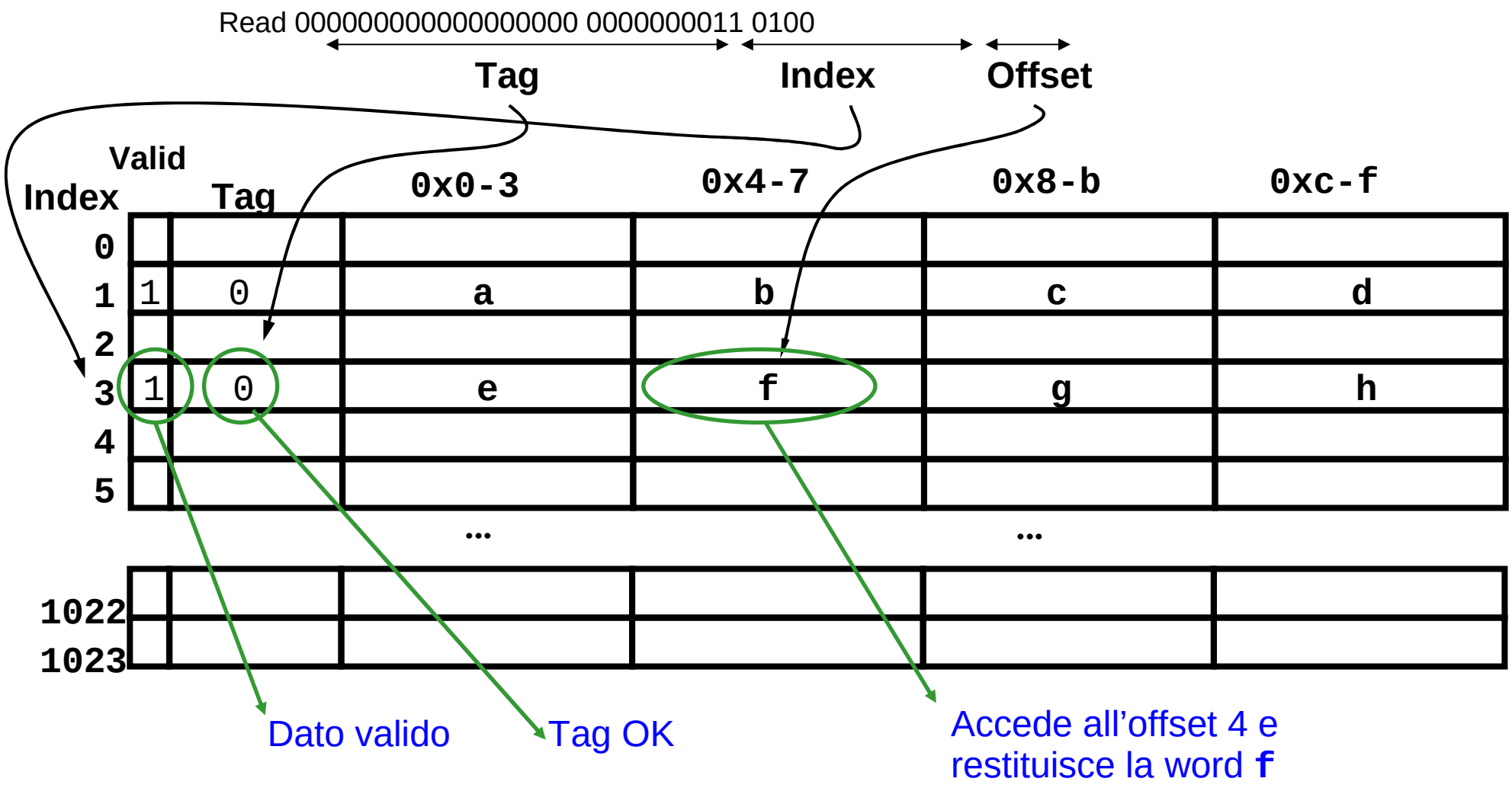


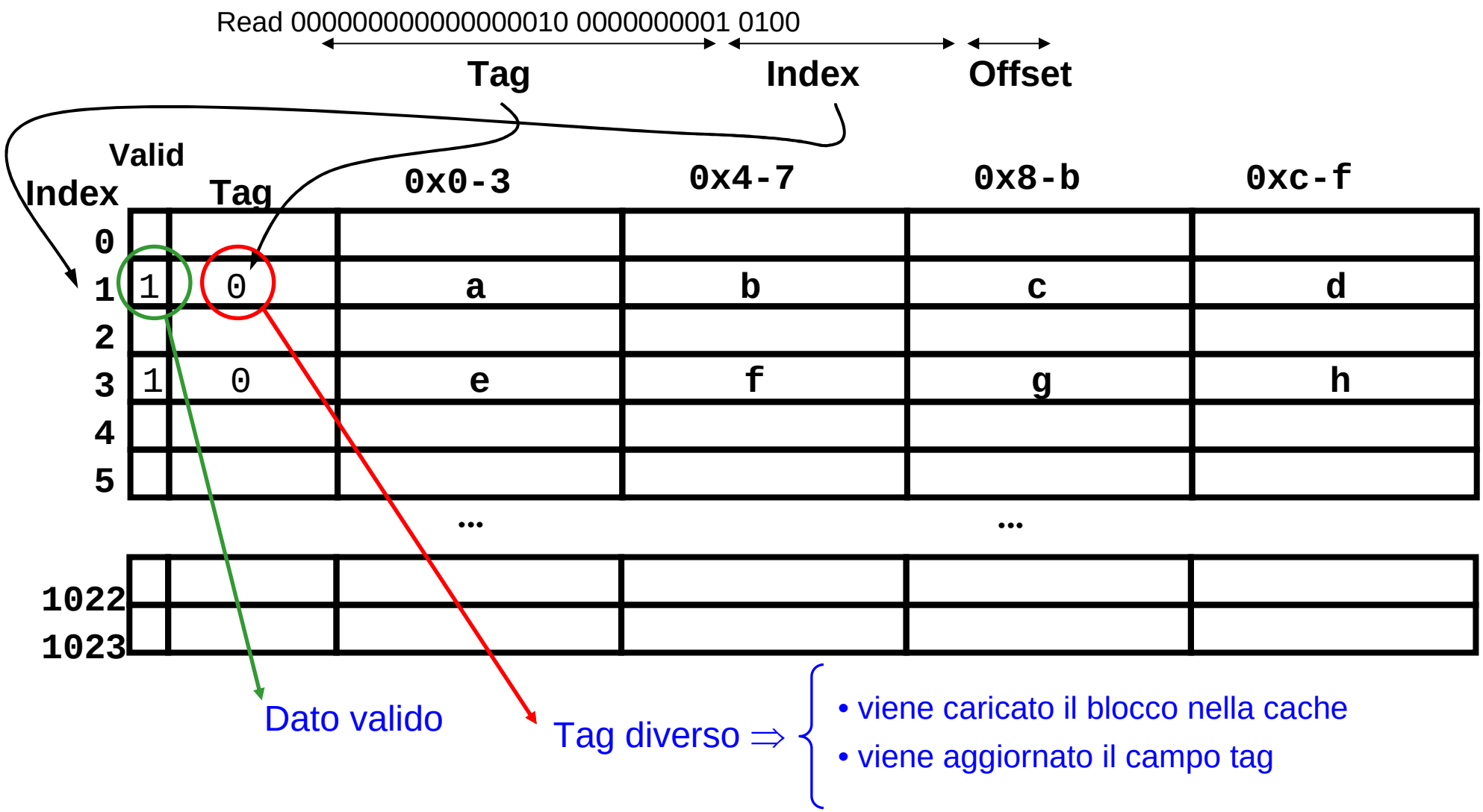


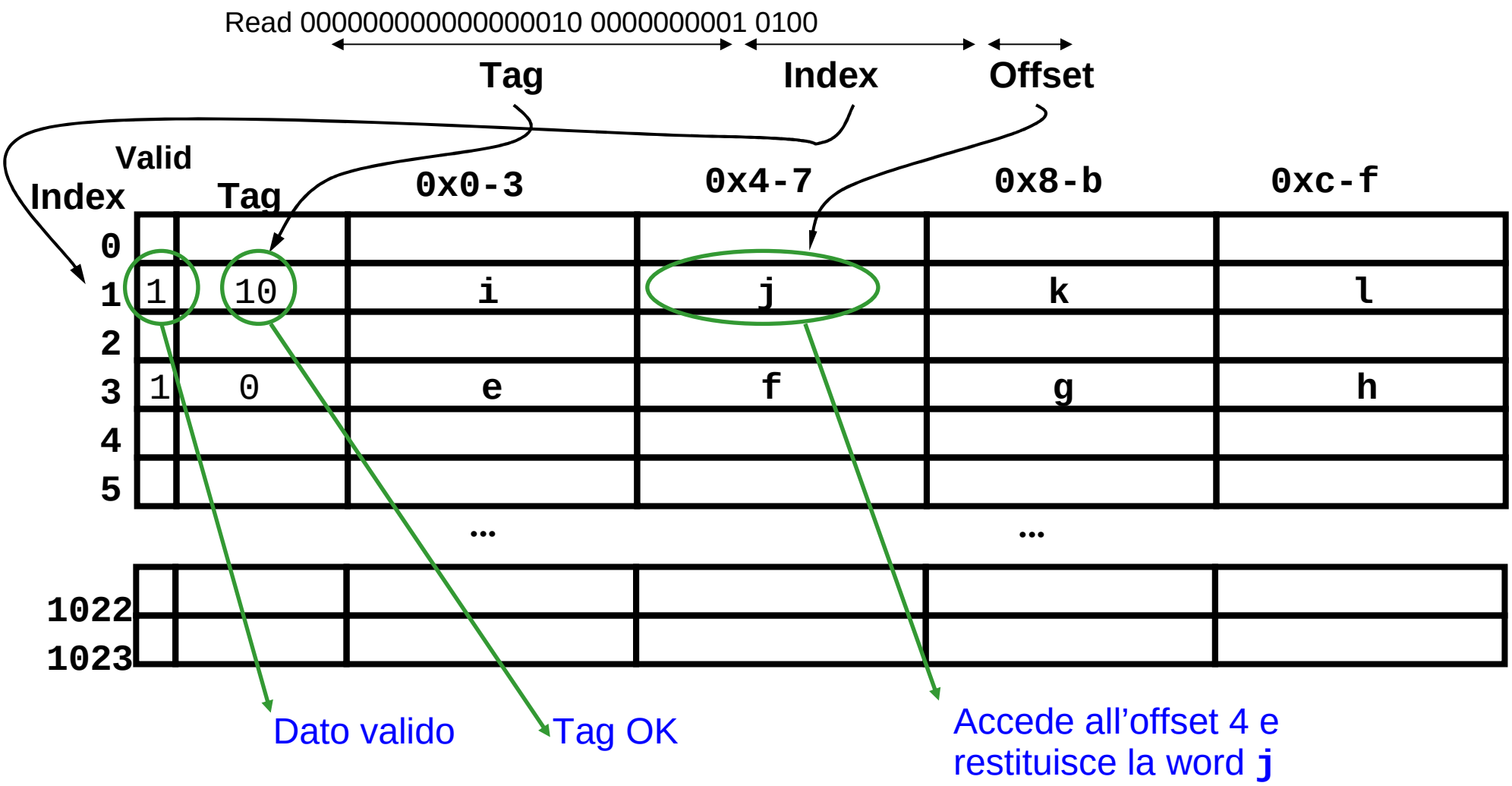








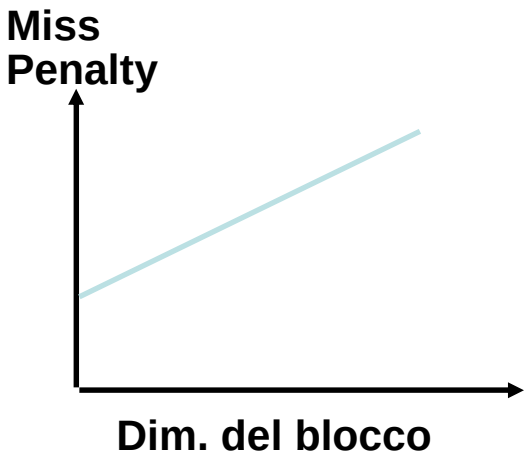


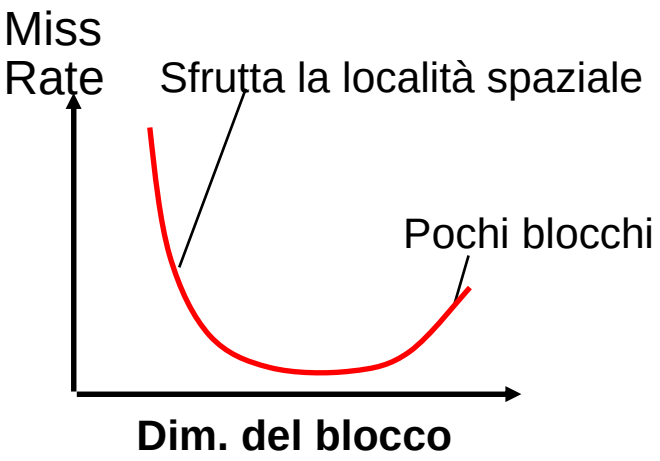


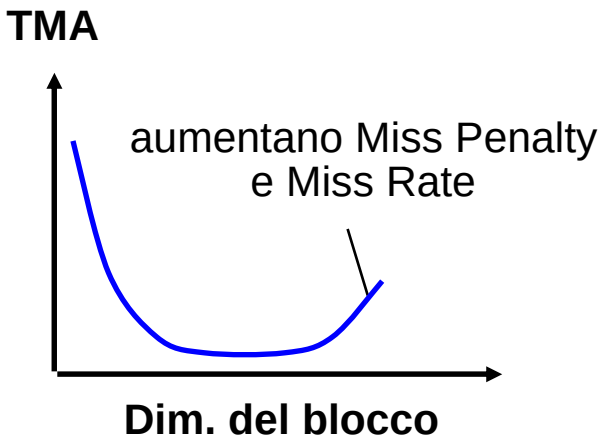
Scelta della dimensione del blocco

- In generale, una ampia dimensione per il blocco permette di sfruttare la località spaziale, MA:
  - blocchi di grossa dimensione comportano maggiori miss penalty:
    - è necessario più tempo per trasferire il blocco
  - se la dimensione del blocco è troppo grossa rispetto alla dimensione della cache, il miss rate aumenta
    - il numero di blocchi nella cache è insufficiente
- Bisogna minimizzare il tempo medio di accesso:

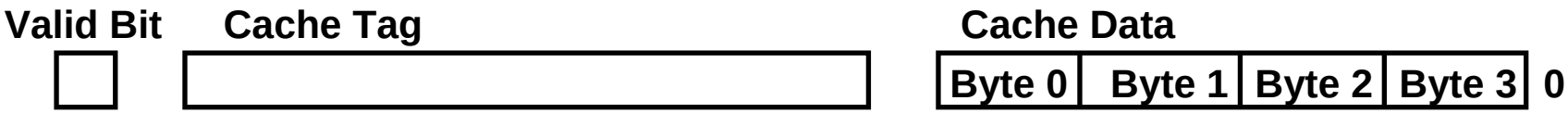
  Hit Time x (1 - Miss Rate) + Miss Penalty x Miss Rate







Cache a singolo blocco (1/2)



- Dimensione della cache = 4 bytes dimensione del blocco = 4 bytes
  - cache con 1 solo blocco
- Se si realizza l'accesso ad un dato, è probabile che presto vi sarà ancora un accesso, ma è improbabile che questo nuovo accesso si verificherà immediatamente.
  - L'accesso successivo produrrà ancora un miss
    - si trasferiscono continuamente dati verso la cache, scaricandoli prima che possano essere nuovamente usati
    - Effetto Ping Pong

Cache a singolo blocco (2/2)

Nella cache a singolo blocco i miss sono interamente dovuti a conflitto sull'indice di cache (*Conflict Miss*), in quanto indirizzi di memoria diversi sono mappati sullo stesso indice.

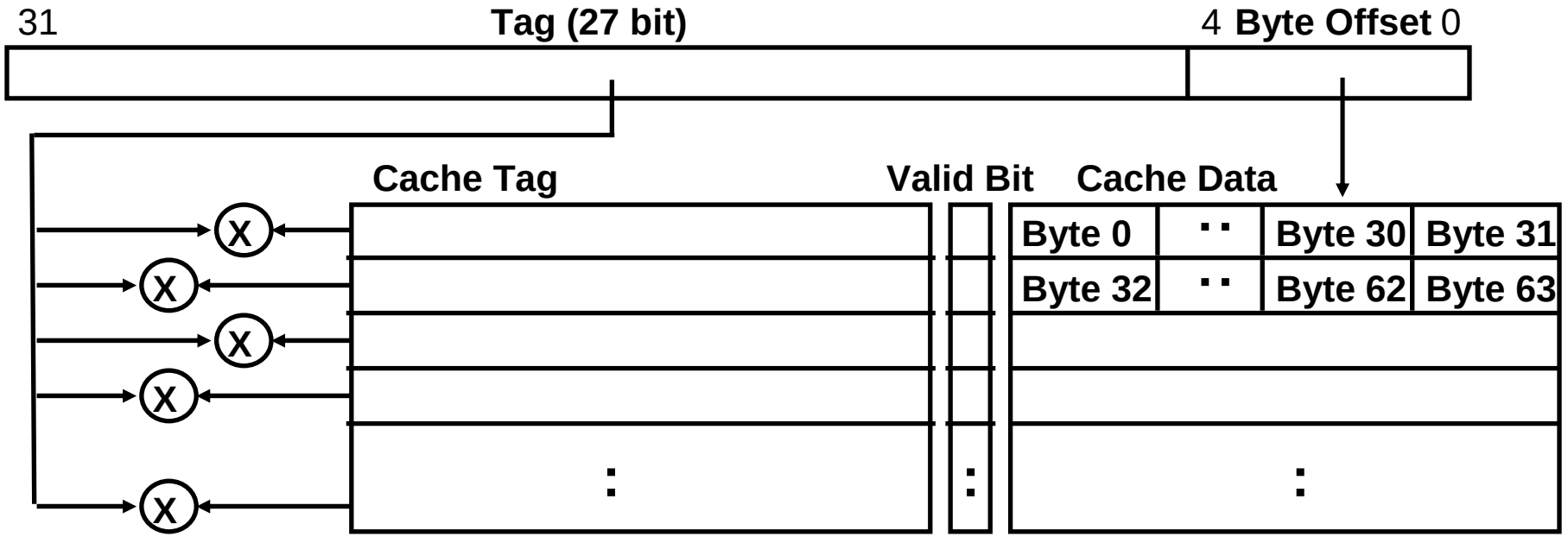
Possibili soluzioni:

- aumentare la dimensione della cache
- aumentare il numero di words per indice
- realizzare un accesso indipendente dall'indice di cache
  - (cache *fully associative*)

Cache Fully Associative (1/2)

- non viene impiegata alcuna indicizzazione
- vengono confrontati in parallelo i tag di tutti i blocchi appartenenti alla cache
- Esempio:
  - indirizzo da 32 bit
  - dimensione di blocco = 32 byte -> tag da 27 bit
  - N blocchi -> necessari N comparatori da 27 bit

Ex: 0x01

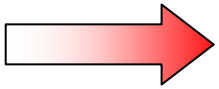


Cache Fully Associative (2/2)

- Per una cache fully associative Conflict Miss = 0 per definizione
- miss generati solo dalla capacità insufficiente della cache (*Capacity Miss*)
- Problemi:
  - hardware molto complesso
  - praticamente realizzabile solo con un piccolo numero di blocchi

Cache Set Associative

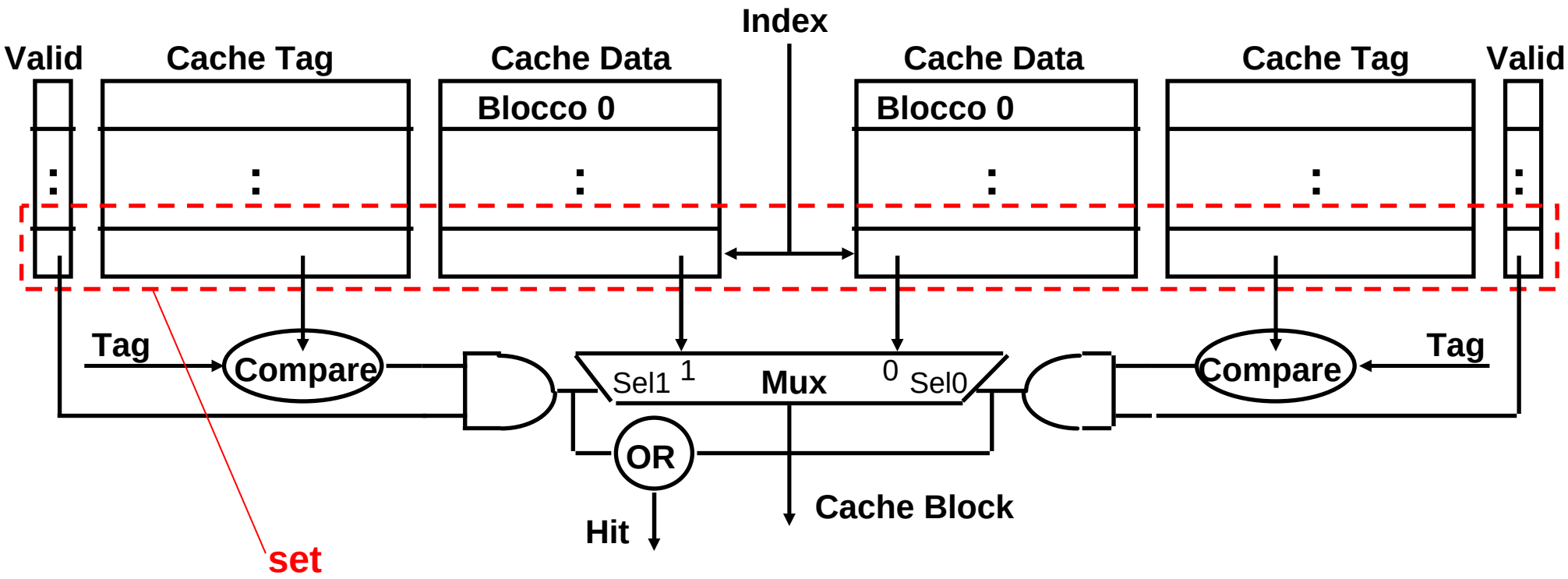
- I blocchi appartenenti alla cache sono raggruppati in *set*
- in una cache set associative ad N vie (N-way set associative) ogni set raggruppa N blocchi
- ogni indirizzo di memoria corrisponde ad un unico set della cache (accesso diretto tramite indice) e può essere ospitato in un blocco qualunque appartenente a quel set
- stabilito il set, per determinare se un certo indirizzo è presente in un blocco del set è necessario confrontare in parallelo i tag di tutti i blocchi



Vengono combinati gli approcci ad accesso diretto e fully associative

Esempio: Cache 2-way Set Associative

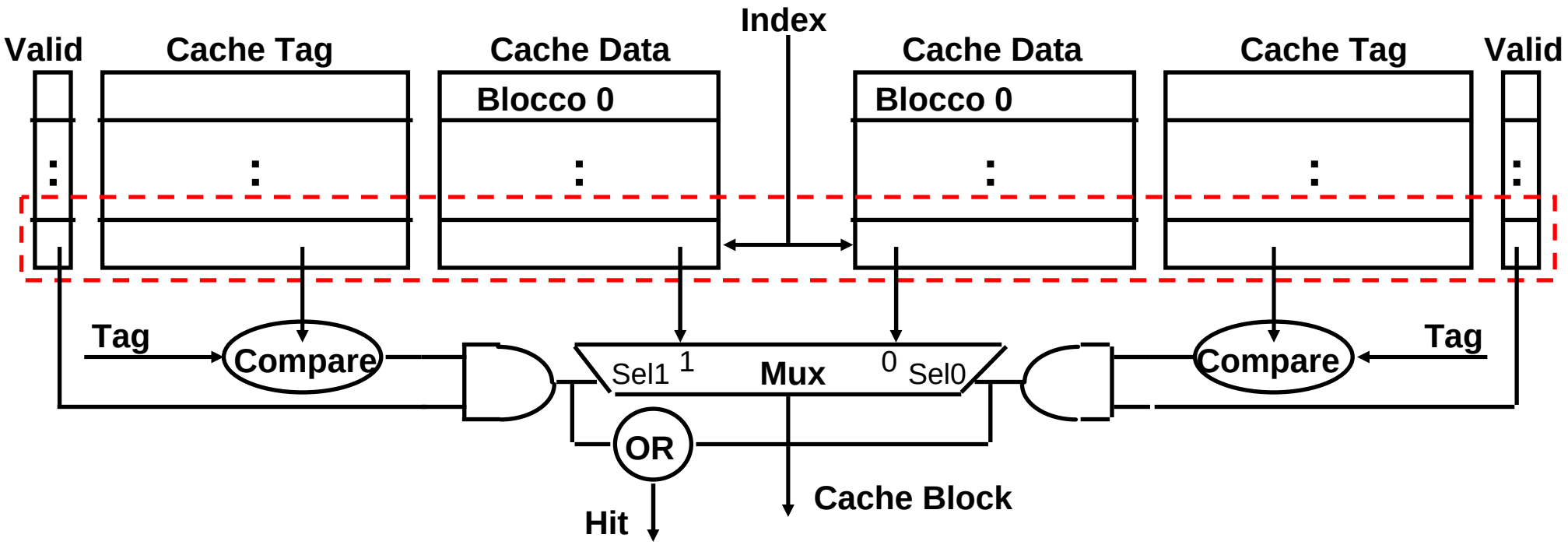
- tramite il campo indice dell'indirizzo viene selezionato un set
- i due tag nel set sono confrontati in parallelo
- il blocco viene selezionato sulla base del risultato dei confronti



Svantaggi della cache set associative

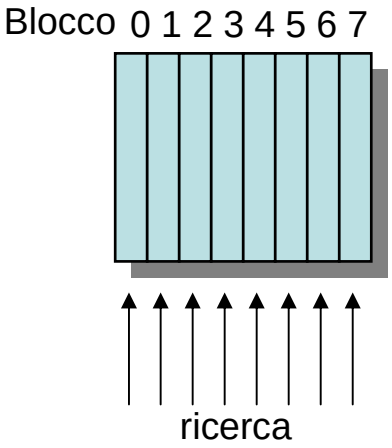
- Cache N-way set associative a confronto con cache ad accesso diretto:
  - N comparatori invece di 1
  - ulteriore ritardo fornito dal multiplexer
  - il blocco è disponibile dopo la decisione Hit/Miss e la selezione del set

- In una cache ad accesso diretto, il blocco è diponibile prima della decisione Hit/Miss

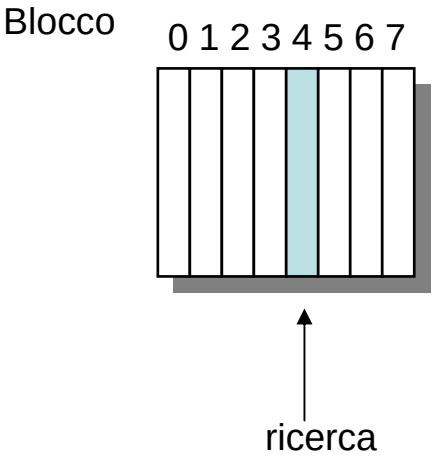


Confronto tra le tecniche di indirizzamento

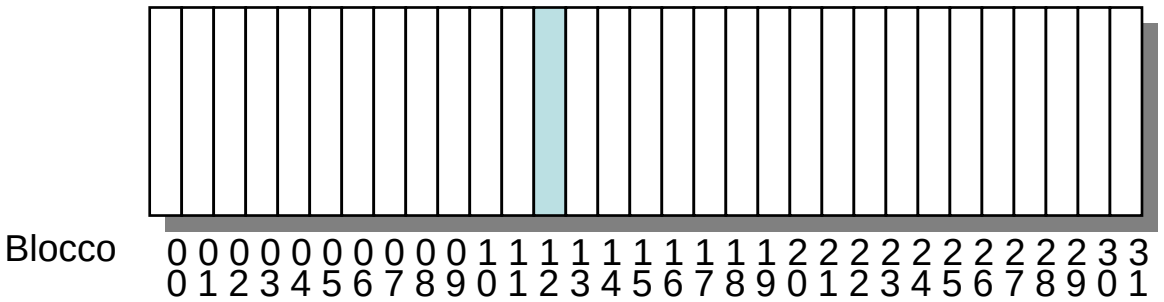
Fully associative:
il blocco 12 può essere disposto ovunque



Direct mapped:
il blocco 12 può essere disposto solo sul blocco 4 (12 mod 8)



Set associative:
il blocco 12 può essere disposto ovunque nel set 0 (12 mod 4)



Memoria di livello inferiore

Algoritmi per la sostituzione di blocchi (1/2)

- In corrispondenza di un miss, è necessario introdurre nella cache un nuovo blocco, eventualmente sostituendo un blocco già presente.
- Nella cache ad accesso diretto non c'è scelta, in quanto i miss sono generati da conflitto sull'indice (conflict miss).
- Nelle cache associative (N-way Set Associative o Fully Associative) c'è la possibilità di scegliere quale blocco sostituire (capacity miss).
- Nel caso ci sia un blocco non valido, si sostituisce questo.
- Altrimenti, due schemi possibili:
  - **LRU**
  - **Random**

Algoritmi per la sostituzione di blocchi (2/2)

- **LRU**
  - Viene sostituito il blocco che è stato utilizzato meno di recente (*Least Recently Used*). Solitamente, è il blocco che ha minore probabilità di essere impiegato di nuovo.
  - E' quindi necessario tenere memoria dei cache hit relativi ad ogni blocco.
- **Random**
  - LRU difficile da gestire in caso di grado di associatività elevato (possibile per una cache 2-way set associative, proibitivo per un'associatività > 4)
  - Alternativa: si sostituisce un blocco a caso.
  - Vantaggi
    - facile da implementare (soprattutto in HW)
    - comportamento predicibile
    - non presenta comportamenti di tipo worst case

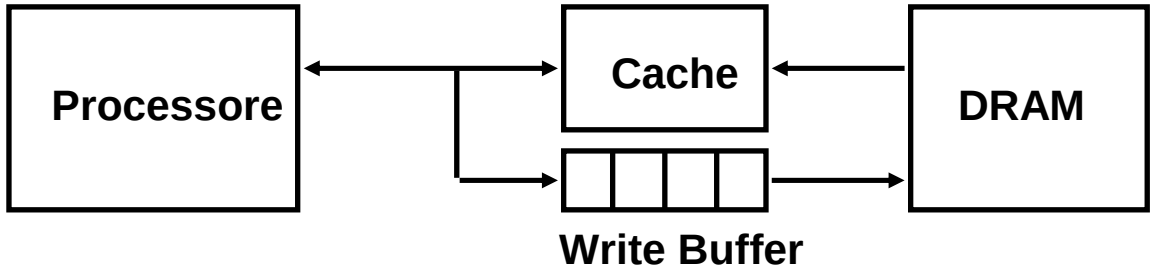
Gestione delle operazioni di scrittura

- In conseguenza di un'operazione di scrittura, effettuata su un blocco presente nella cache, i contenuti di quest'ultima saranno diversi dai contenuti della memoria di livello inferiore.
- Soluzione più semplice: il dato viene scritto sia nel blocco della cache sia nel blocco contenuto nella memoria di livello inferiore. (**Write through**)
- Problema: in questo modo le operazioni di scrittura vengono effettuate alla velocità della memoria di livello inferiore !
- Alternative:
  - modalità **Write Back**
  - utilizzo di un **Write Buffer**

Modalità Write Back

- I dati sono scritti solo sul blocco presente nella cache
- Il blocco modificato è trascritto in memoria principale solo quando viene sostituito
  - Il blocco può essere in due stati: non modificato (clean) o modificato (dirty)
- Vantaggi:
  - Scritture successive sulla stessa locazione alterano solo la cache
- Svantaggi:
  - Ogni sostituzione di blocco (dovuti p.es. a read misses) può provocare un trasferimento in memoria

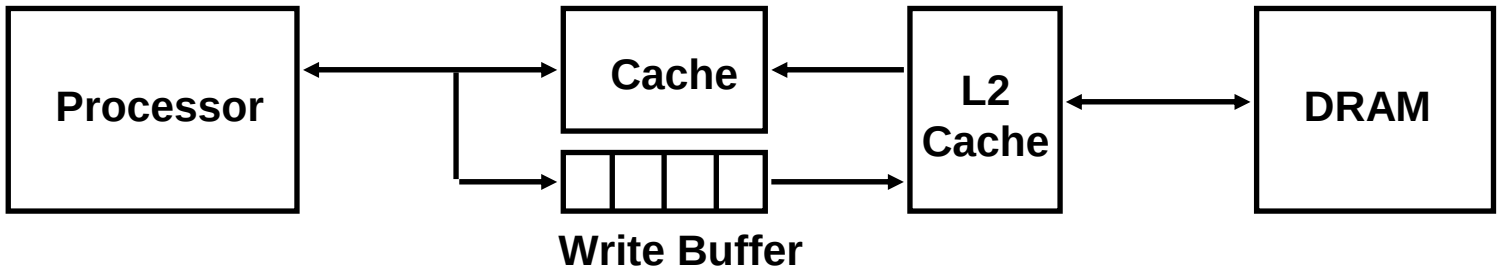
Write Through con Write Buffer



- Viene disposto un buffer per la scrittura (Write Buffer) tra la cache e la memoria
  - il processore scrive i dati nella cache e nel buffer
  - il controller della memoria scrive i contenuti del buffer in memoria
- Il Write Buffer funziona con strategia FIFO:
  - numero tipico di elementi: 4
  - funziona se la frequenza di scrittura << 1 / write cycle della DRAM
  - altrimenti, il buffer può andare in saturazione.

Saturazione del Write Buffer

- Nel caso in cui la frequenza di scrittura è superiore a 1 / DRAM write cycle (es.: ciclo della CPU troppo breve o troppe istruzioni di scrittura successive), il buffer andrà in overflow qualunque sia la sua dimensione.
- Possibili soluzioni:
  - modificare la modalità della cache in write back
  - inserire una cache di secondo livello (L2) di tipo write back



Gestione del write miss

- In corrispondenza di un read miss, bisogna trasferire il blocco relativo nella cache così da poter completare la lettura. Come operare in presenza di un write miss ?
- Opzione 1: stessa tecnica impiegata nel read. L'intero blocco viene trasferito nella cache dove viene poi modificato (Write Allocate).

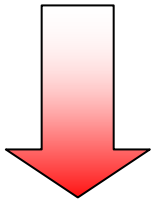
- Opzione 2: viene modificata solo la memoria di livello inferiore; la cache non subisce alcuna modificata (No Write Allocate)

Cache a più livelli: perché ?

- Obiettivo: minimizzare il tempo medio di accesso:
  = Hit Time x (1 - Miss Rate)
  + Miss Penalty x Miss Rate
- Finora si è cercato di diminuire il Miss Rate agendo su:
  - dimensioni del blocco
  - dimensioni della cache
  - grado di associatività
- Come diminuire il Miss Penalty?
  - Inizialmente, il Miss Penalty era intorno ai 10 cicli di clock di processore
  - Attualmente, per un processore a 500 MHz, con una DRAM che richiede 200 ns per un ciclo completo di lettura/scrittura $\Rightarrow$ 100 cicli di clock
  - Soluzioni possibili:
    - organizzazione interleaving della memoria principale
    - introduzione di una **cache di secondo livello** (L2 Cache)

Analisi del TMA con cache a più livelli

- TMA = L1 hit time * L1 hit rate + L1 miss penalty * L1 miss rate
- Nel valutare il miss penalty per la cache L1 bisogna ora considerare che un miss penalty su L1 comporta un accesso su L2

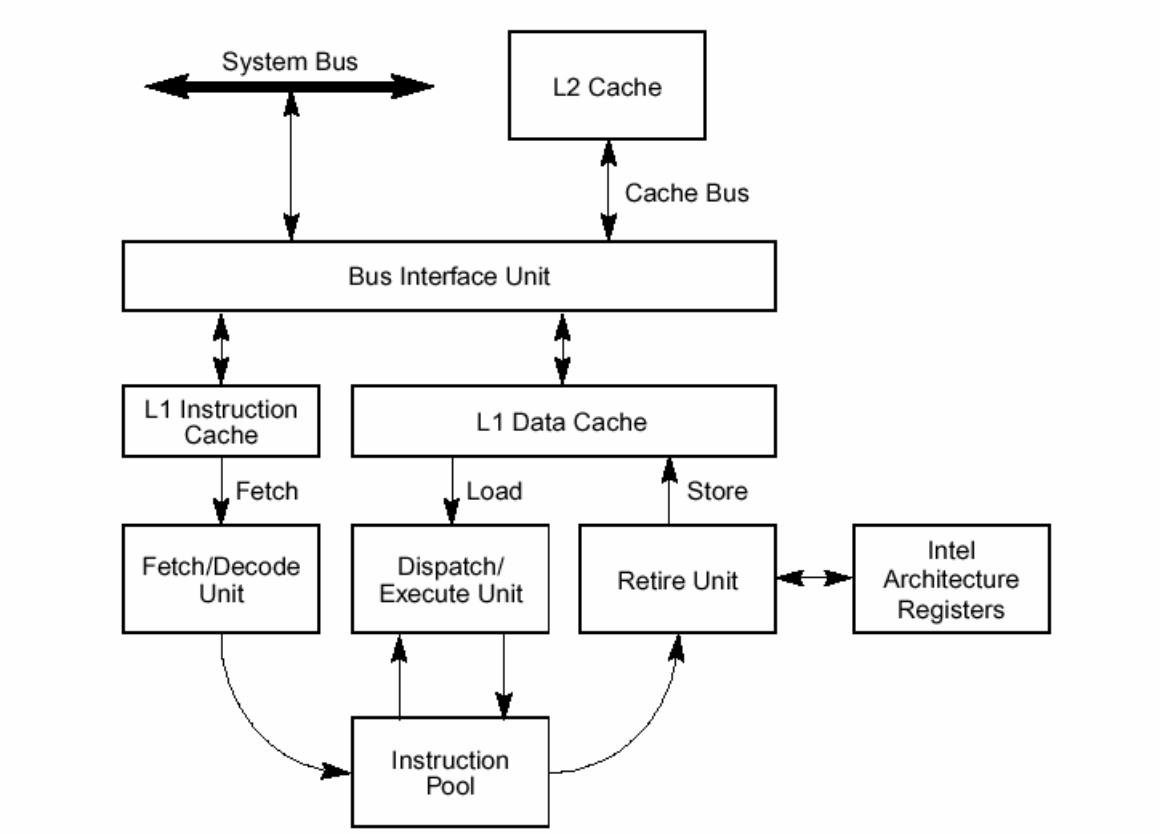


TMA =
L1 hit time * L1 hit rate +
(L2 hit time * L2 hit rate + L2 miss penalty * L2 miss rate)* L1 miss rate

Un po' di conti

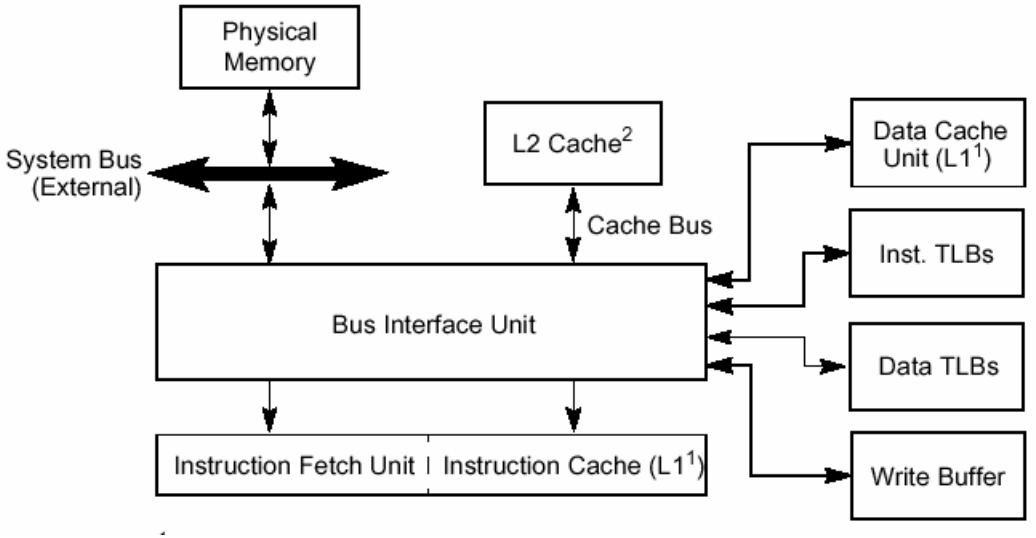
- Ipotesi:
  - L1 hit time = 1 ciclo
  - L1 hit rate = 90%
  - L2 hit time = 4 cicli
  - L2 hit rate = 90%
  - tempo di ciclo della memoria principale = 100 cicli
- Senza cache L2:
  TMA = 1*0.9 + 100 * (1-0.9)
  = 0.9 + 100 * 0.1 = **10.9** cicli di clock
- Con la cache L2:
  TMA = 1*0.9 + (4*0.9 + 100*0.1) *(1-0.9)
  = 0.9 + (13.6) * 0.1 = **2.26** cicli di clock

Esempio: la cache nell'Intel Pentium



Livelli della cache

- Cache a 2 livelli (L1 e L2)
- Cache di primo livello (L1):
  - strettamente connessa con le unità del processore
  - divisa in due sezioni: Instruction Cache e Data Cache
- Cache di secondo livello (L2):
  - strettamente connessa con la cache di primo livello attraverso un apposito bus (*processor cache bus*)
  - contiene sia dati che istruzioni



Caratteristiche della cache

- L1 Instruction Cache:
  - 16 Kbyte
  - 4-way set associative
  - dimensione del blocco pari a 32 byte
- L1 Data Cache
  - 16 Kbyte
  - 2-way set associative
  - blocco da 32 byte
- Cache di secondo livello (L2):
  - 256 Kbyte, 512 Kbyte o 1 Mbyte
  - 4-way set associative
  - dimensione del blocco pari a 32 byte

Nel Pentium 4:

- **on-die**
- **8-way set-associative**

Caching Methods (1/2)

- **Uncacheable (UC)**

  l'area non viene allocata sulla cache. Tutte le operazioni di lettura e scrittura vengono realizzate verso la memoria centrale. Utile per locazioni di memoria corrispondenti a device di I/O memory mapped.

- **Write Combining (WC)**

  l'area non viene allocata sulla cache. Le operazioni di scrittura possono essere ritardate e combinate nel write buffer per ridurre gli accessi in memoria. Adatto per aree di memoria adibite a video frame buffer.

- **Write Through (WT)**

  l'area viene allocata sulla cache. Le operazioni di scrittura vengono effettuate sia sulla cache che in memoria.

Caching Methods (2/2)

- **Write Back (WB)**

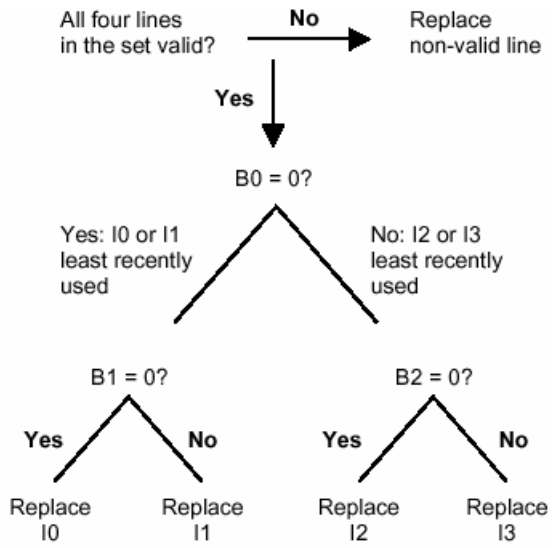
  l'area viene allocata sulla cache. Le operazioni di scrittura aggiornano la cache, ma non sono inoltrate subito alla memoria. L'aggiornamento in memoria avviene in corrispondenza della deallocazione del blocco dalla cache o su richiesta del meccanismo che si occupa di mantenere la coerenza della cache.

- **Write Protected (WP)**

  l'area viene allocata sulla cache. Le operazioni di lettura accedono ai dati della cache, quando possibile. Un read miss provoca l'aggiornamento della cache. Le operazioni di scrittura, invece, sono inoltrate direttamente alla memoria e rendono non validi i blocchi coinvolti.

Organizzazione della cache

- La cache dati e la cache istruzioni possono essere accedute contemporaneamente.
- I tag di entrambe le cache sono *triple ported*. Ciò permette accessi contemporanei a due diversi dati (istruzioni) da parte di diverse unità del processore. Una porta è dedicata alle attività di snooping.
- I dati sono organizzati in maniera interleaved su 8 banchi da 4 byte.
- Entrambe le cache hanno un controllo basato sulla parità:
  - un bit di parità per ogni tag;
  - un bit di parità ogni 8 byte per la cache istruzioni;
  - un bit di parità per ogni byte per la cache dati.
- L'algoritmo di sostituzione utilizzato è uno pseudo LRU che richiede 3 bit per ogni set.



Protocollo per la consistenza della cache dati (prot. MESI) (1/2)

E' costituito da un insieme di regole grazie alle quali si definisce uno stato per ogni blocco nella cache che riporta la validità del dato. Lo stato è determinato sulla base delle attività di lettura/scrittura che si riscontrano sul bus (*snooping*), generate sia dal processore, sia da altri master del bus.

E' formato da 4 stati (**M**, **E**, **S**, **I**) che definiscono se un blocco è valido, se è disponibile in altre cache, e se è stato modificato:

Cache Line State	M (Modified)	E (Exclusive)	S (Shared)	I (Invalid)
This cache line is valid?	Yes	Yes	Yes	No
The memory copy is…	…out of date	…valid	…valid	—
Copies exist in caches of other processors?	No	No	Maybe	Maybe
A write to this line …	…does not go to bus	…does not go to bus	…causes the processor to gain exclusive ownership of the line	…goes directly to bus

Protocollo per la consistenza della cache dati (prot. MESI) (2/2)

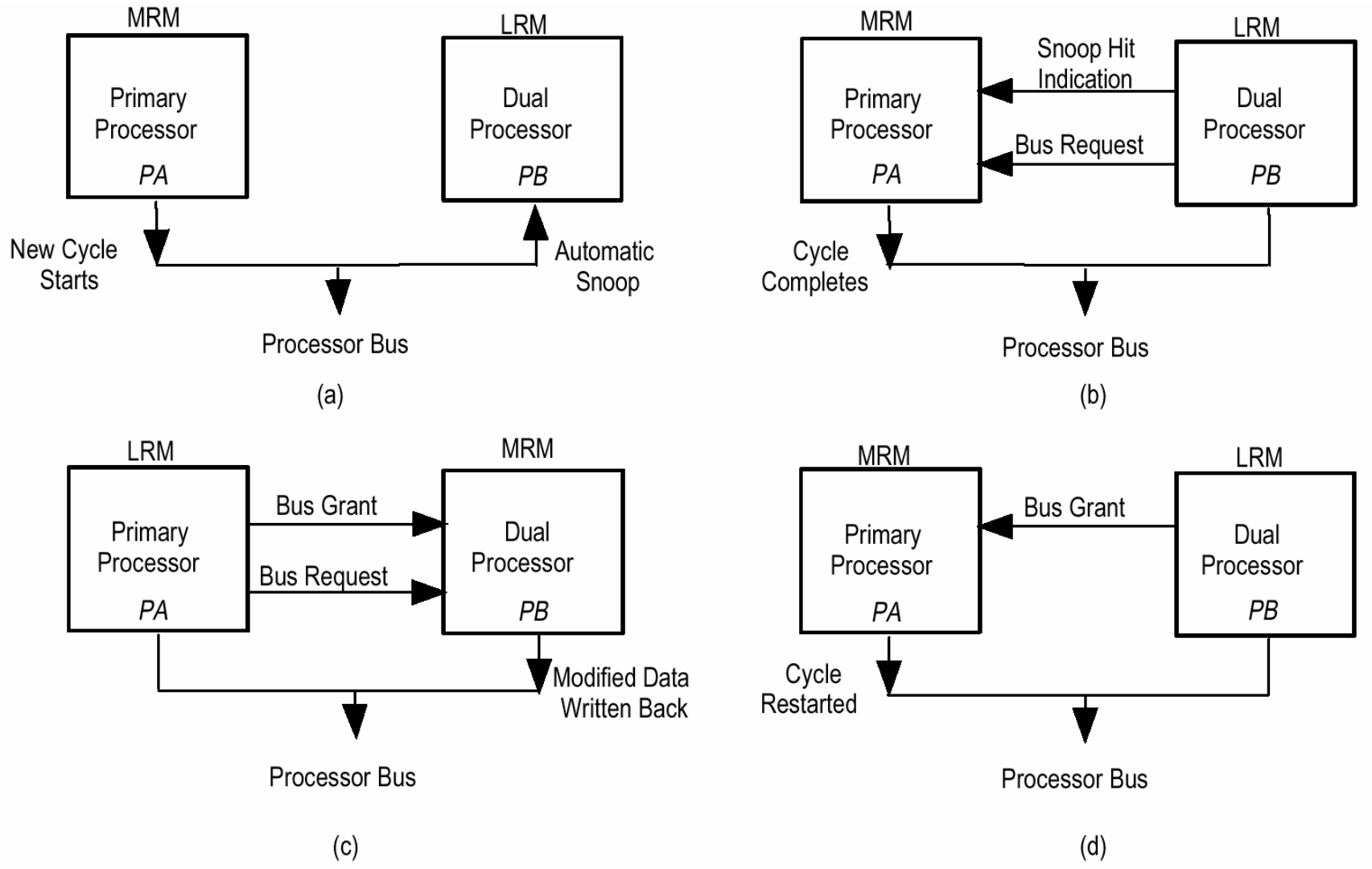
M - Modified: Un blocco in stato M è disponibile all'interno di una sola cache e risulta essere modificato rispetto alla memoria centrale. Un accesso ad un blocco in stato M non richiede cicli sul bus e non modifica lo stato.

E - Exclusive: Un blocco in stato E è disponibile all'interno di una sola cache e risulta essere uguale al dato in memoria centrale. Un accesso ad un blocco in stato E non richiede cicli sul bus. Un'operazione di scrittura rende il blocco MODIFIED.

S - Shared: Il blocco è potenzialmente comune ad altre cache. La lettura di un blocco in stato S non genera attività di bus, mentre una scrittura genera un write through sul bus che può invalidare il blocco all'interno di altre cache. Il dato nella cache viene comunque aggiornato.

I - Invalid: Il blocco non è disponibile nella cache. La lettura di un blocco in stato I genera un MISS che forza un trasferimento del dato dalla memoria centrale. La scrittura, invece, causa un ciclo di write-through sul bus.

Snooping



Protocollo per la consistenza della cache istruzioni

- Viene utilizzato un sottinsieme del protocollo MESI.
- Accessi alla cache istruzioni generano un Hit (Shared) o un Miss (Invalid).
-
- Nel caso di un read hit, si accede alla cache e non si genera attività sul bus. Nel caso di un read miss, si genera l'aggiornamento del blocco con trasferimento dalla memoria centrale.