Corso di Laurea in Informatica
Architettura degli elaboratori
a.a. 2021-2022

Architettura degli Elaboratori 2021-2022

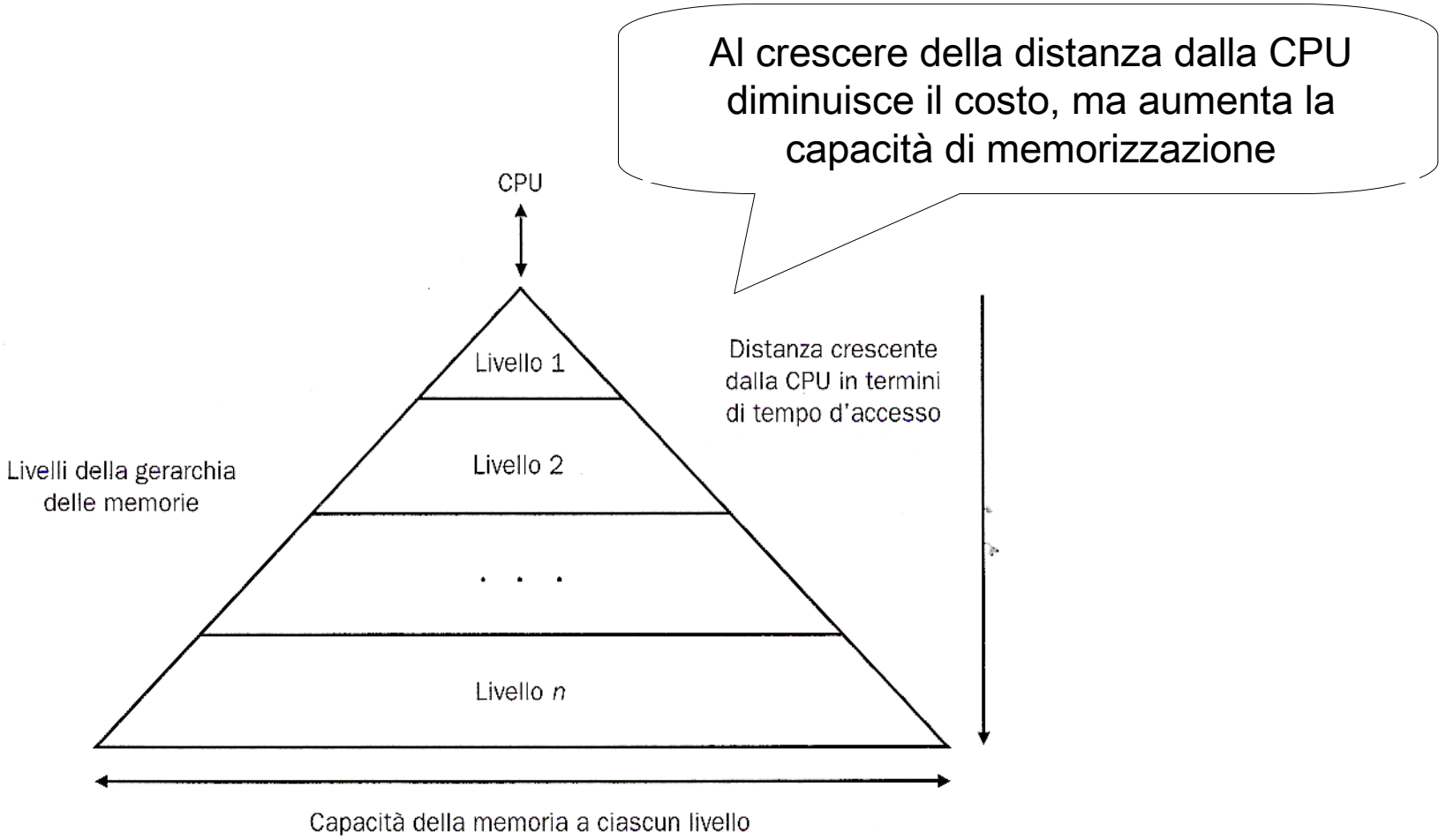
Gerarchie di memoria e cache

Prof. Elisabetta Fersini
elisabetta.fersini@unimib.it

Gerarchie di memoria

- Avere a disposizione una quantità illimitata di memoria e contemporaneamente veloce
- *Premessa*: un programma non accede a tutte le sue istruzioni e a tutti i suoi dati contemporaneamente con la stessa probabilità
- *Soluzione*: una gerarchia di memoria: consiste in un insieme di livelli di memoria, ciascuno caratterizzato da una diversa velocità e dimensione
- A parità di capacità, le memorie più veloci hanno un costo più elevato per singolo bit di quelle più lente

Gerarchie di memoria



Gerarchie di memoria

- La **memoria interna** alla CPU è costituita dai registri ed è caratterizzata da alta velocità e limitate dimensioni.
- La **memoria centrale** è caratterizzata da dimensioni molto maggiori della memoria interna alla CPU, ma con tempi di accesso più elevati. È accessibile in modo diretto tramite indirizzi.
  - Nei sistemi attuali un livello di **memorie cache** è stato inserito tra CPU e memorie centrali
- Le **memorie secondarie** sono ad alta capacità, bassi costi e non volatili

Principio di località

- Un programma, in un certo istante di tempo, accede soltanto a una porzione relativamente piccola del suo spazio di indirizzamento
- Rappresenta la base del comportamento dei programmi in un calcolatore
- Due tipi di località:
  - Temporale: quando si fa riferimento a un elemento c'è la tendenza a fare riferimento allo stesso elemento dopo poco tempo
  - Spaziale: quando si fa riferimento a un elemento c'è la tendenza a fare riferimento poco dopo ad altri elementi che hanno l'indirizzo vicino ad esso
- I programmi NON vedono la gerarchia ma referenziano i dati come se fossero sempre in memoria centrale

Gerarchia di memoria e principio di località

- Il principio di località viene sfruttato strutturando la memoria in modo gerarchico
- Velocità
  - Più è veloce, più è vicina al processore, più è costosa
- Dimensione
  - Più è grande, più è lontana dal processore, meno costosa

- Un livello di memoria più vicino al processore contiene un sottoinsieme di dati memorizzati in ogni livello sottostante e tutti i dati si trovano nel livello più basso

Definizioni (I)

- Blocco/linea: la più piccola quantità di informazione che può essere presente/assente in una gerarchia di memoria
- Hit (successo nell'accesso): l'informazione richiesta dal processore si trova in uno dei blocchi nel livello superiore di memoria
- Miss (fallimento nell'accesso): il dato non è presente nel livello immediatamente superiore ed occorre accedere al livello più distante.

- Hit rate (frequenza di hit): frazione degli accessi alla memoria nei quali l'informazione richiesta è stata trovata nel livello superiore di memoria
- Miss rate (frequenza di miss): frazione degli accessi alla memoria nei quali l'informazione richiesta NON è stata trovata nel livello superiore di memoria (1 – HitRate)

Definizioni (II)

- Tempo di hit: tempo di accesso al livello superiore della memoria
  - Comprende anche il tempo necessario a stabilire se il tempo di accesso si risolva in un successo o in un fallimento

- Tempo di miss: il tempo necessario a sostituire un blocco del livello superiore con un nuovo blocco dal livello inferiore della gerarchia, e trasferire i dati di questo blocco al processore

- Remind: una gerarchia di memoria può essere composta da più livelli, ma i dati vengono trasferiti solo tra due livelli vicini

Definizioni (III)

- Frequenza di hit = Accessi risolti con successo / Numero totale di accessi
- Frequenza di miss = Accessi risolti senza successo / Numero totale di accessi

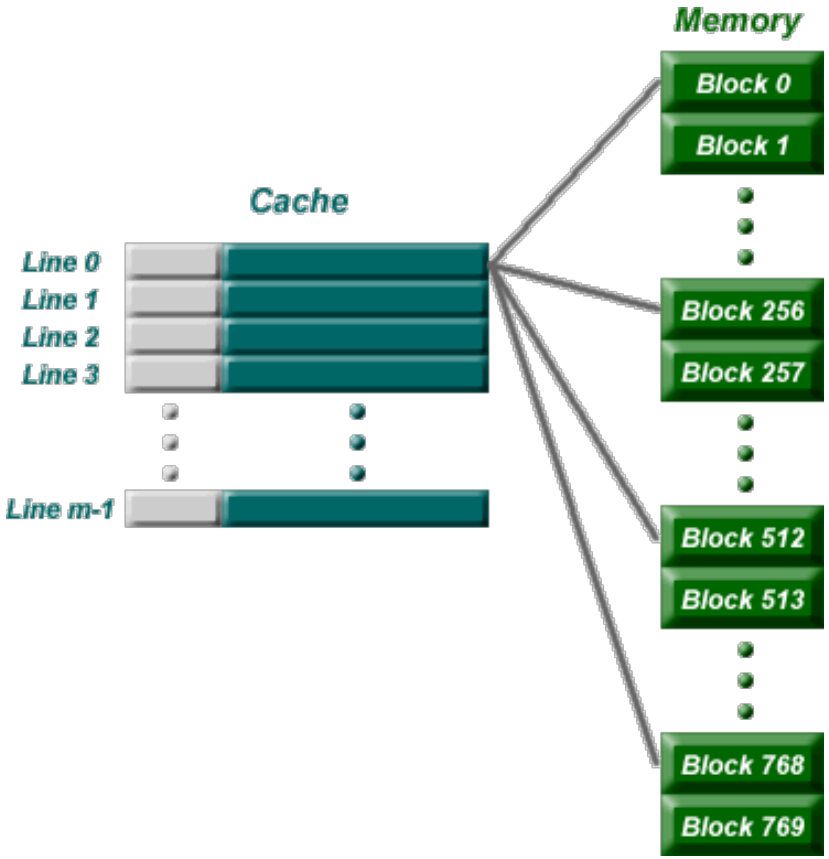
Cache

- Cache: il livello della memoria gerarchica che si trova tra il processore e la memoria principale
- Provenienza: termine francese: caché – significa nascosto
- La memoria cache e il suo utilizzo sono generalmente trasparenti al programmatore (quindi nascosta)
- L'algoritmo di caching si basa sui principi di località spaziale e temporale:
  - Mantiene i dati richiesti recentemente "vicino" alla CPU (località temporale)
  - Muove blocchi contigui di memoria che contendono il dato richiesto (località spaziale)

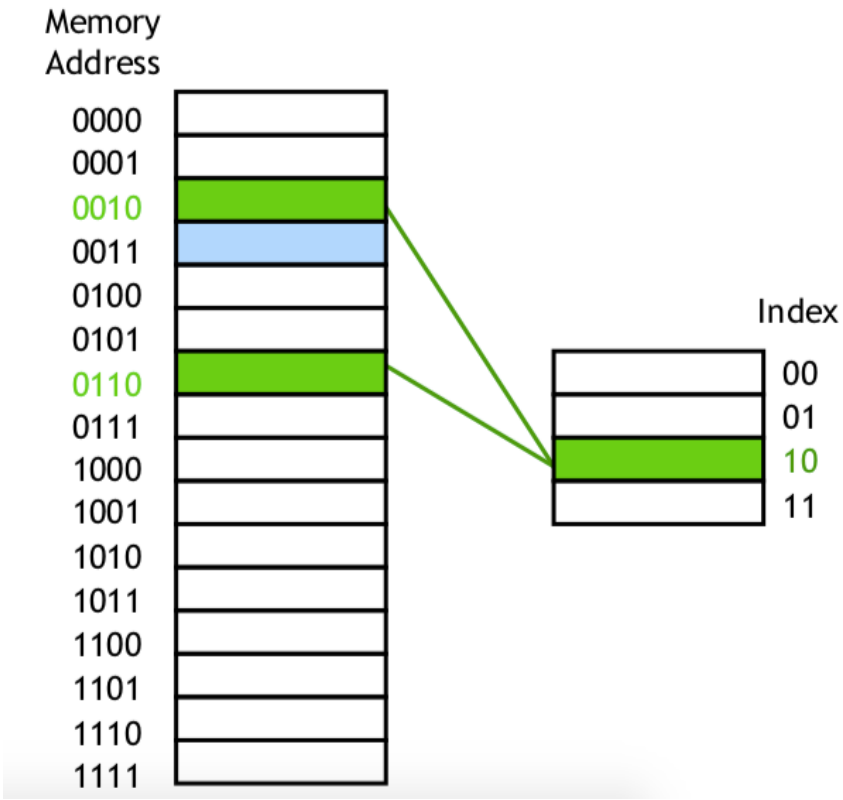
Cache

- Direct Mapped
  - A ciascun blocco della memoria corrisponde una specifica locazione nella cache
- Fully Associative
  - Ogni blocco può essere collocato in qualsiasi locazione della cache
  - Per ricercare un blocco nella cache è necessario cercarlo in tutte le linee della cache
  - La ricerca sequanziale è troppo lenta -> ricerca in parallelo (soluzione molto costosa
- Set Associative
  - Soluzione intermedia tra direct mapped e fully associative
  - Ciascun blocco della memoria ha a disposizione un numero fisso (>=2) di locazioni in cache

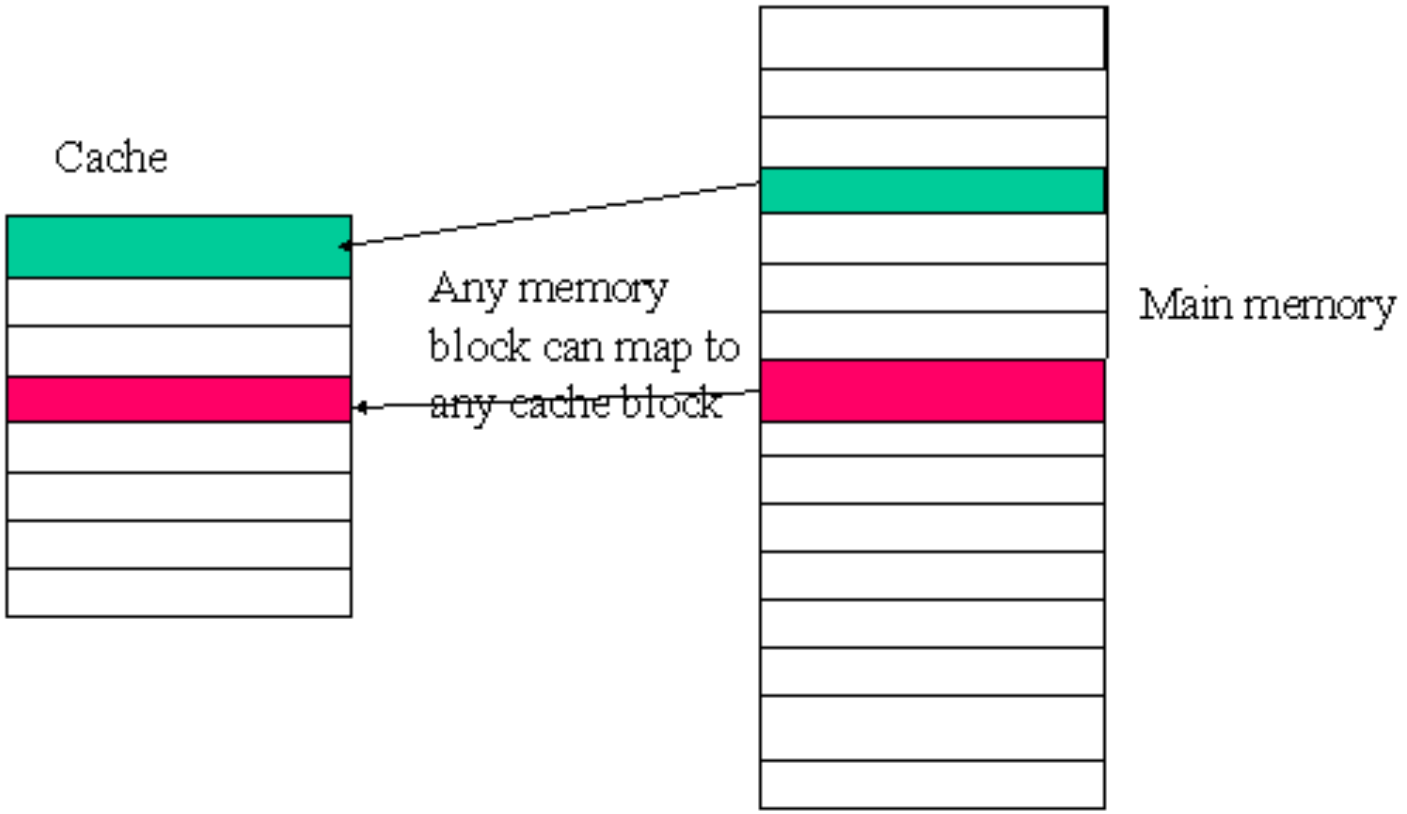
Direct Mapping



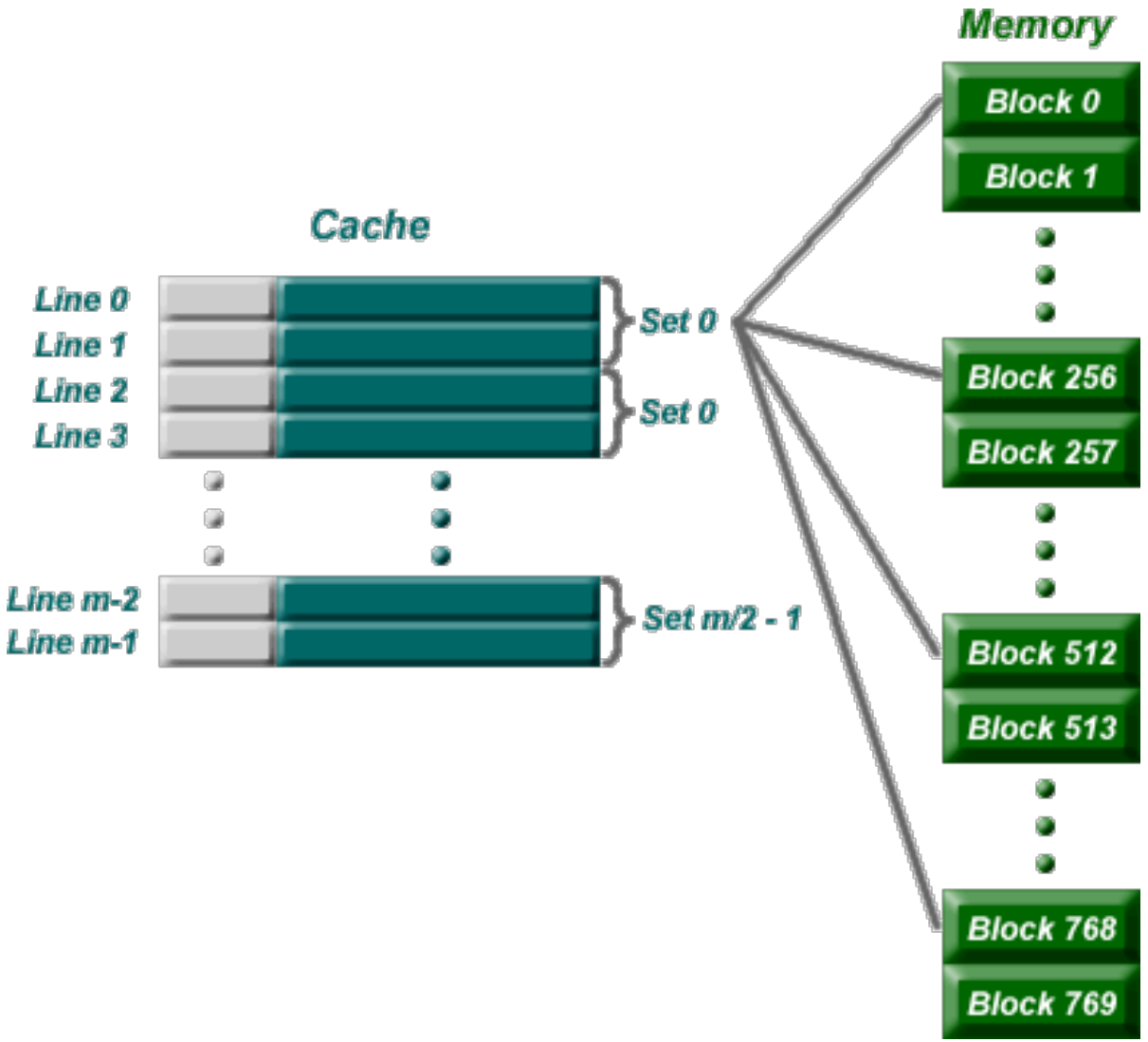
Direct Mapping



Fully associative



Set Associative Mapping



Direct Mapped Cache

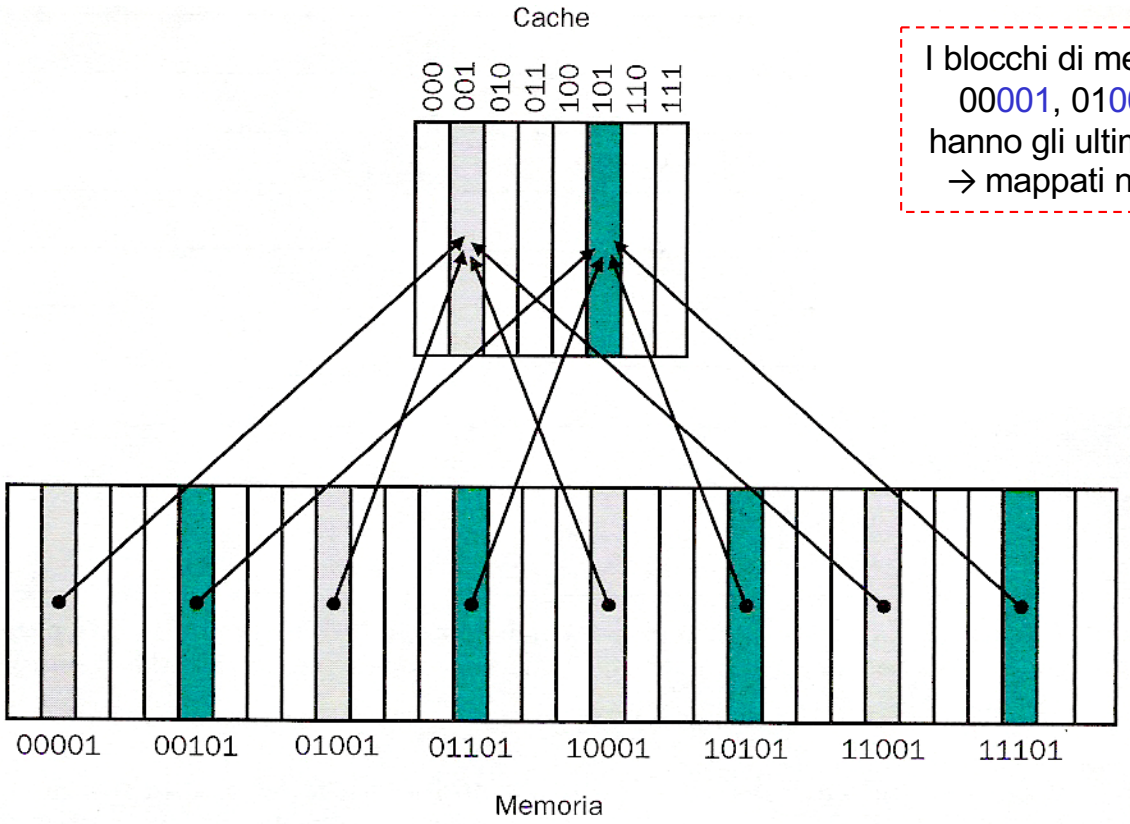
- Associa una sola locazione della cache a ogni parola della memoria definendo una corrispondenza tra l'indirizzo in memoria della parola e la locazione nella cache
- Per trovare il blocco che corrisponde ad un indirizzo della memoria principale:

(indirizzo del blocco) modulo (numero di blocchi nella cache)

Direct Mapped Cache

- Se il numero di blocchi nella cache è una potenza di 2, la **posizione** corrispondente della parola in cache è data dai log2(numero elementi nella cache) **bits meno significativi** dell'indirizzo in memoria principale.
- Esempio:
  - Numero di elementi nella cache: 8
  - Bit per indirizzare una locazione della cache: log2(8)=3
  - Indirizzo della parola di memoria= 0111 01010 0010 0100
  - Posizione in cache: 100 (4)

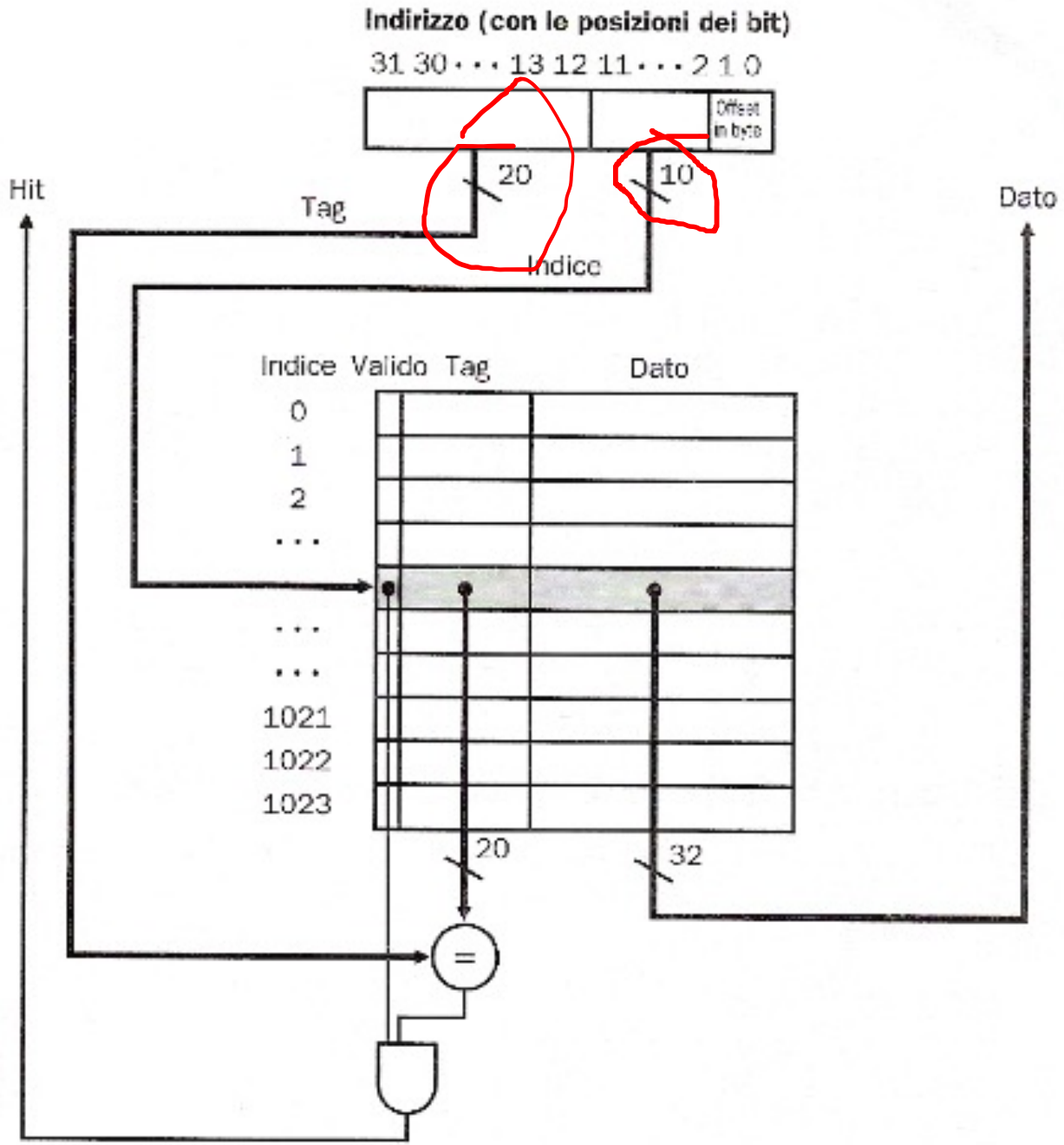
Corrispondenza tra indirizzi in memoria e in cache



I blocchi di memoria con indirizzo
00001, 01001, 10001, 11001
hanno gli ultimi $\log_2 8 = 3$ bit uguali
→ mappati nello stesso blocco

Campi della cache

- Tag (etichetta): contiene informazioni necessarie a verificare se una parola della cache corrisponde o meno alla parola cercata
- Indice: utilizzato per selezionare il blocco della cache
- Bit di validità: un campo che indica se il blocco di memoria associato contiene un dato valido (true/false)

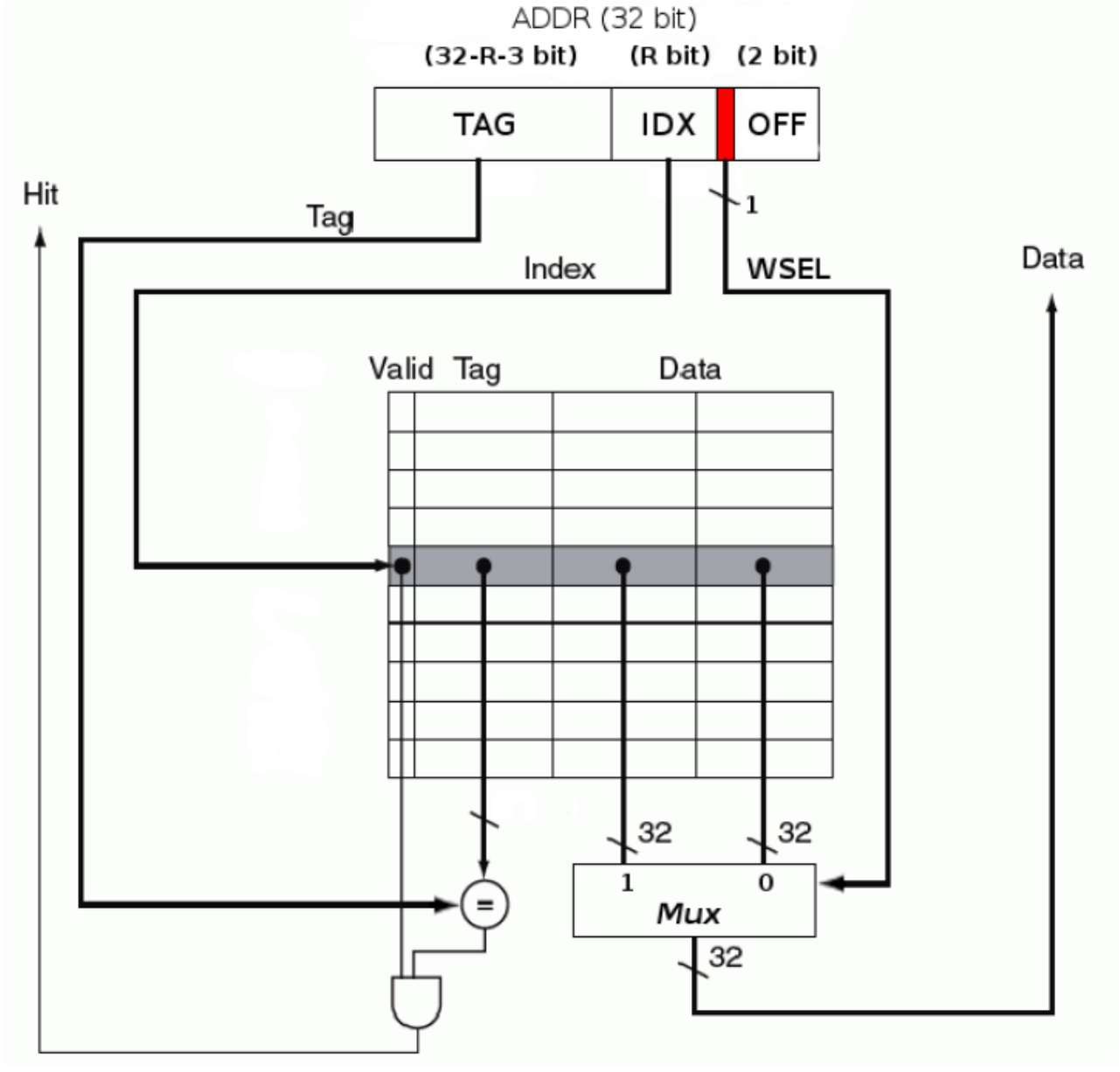

Indirizzo (con le posizioni dei bit)
31 30 · · · 13 12 11 · · · 2 1 0
Offset in byte
20
10
Hit
Tag
Indice
Dato
Indice Valido Tag
Dato
0
1
2
· · ·
· · ·
· · ·
1021
1022
1023
20
32
=

Campi della cache

- I bit meno significativi dell'indirizzo del dato richiesto indicano la posizione (indice) del dato nella cache
- Osservazione: più blocchi di memoria sono destinati alla stessa posizione nella cache! Come riconoscere il dato richiesto?
- I bit più significativi dell'indirizzo del dato richiesto rappresentano il Tag
- Esempio:

indirizzo: 00101 -> Tag: 00 Indice: 101

2 – Word Direct Mapped Cache



Esempio

- Si consideri la seguente situazione:
  - Indirizzo su 32 bit
  - Cache a mappatura diretta
  - La dimensione della cache è di 2^n blocchi, dove *n* bit vengono usati per l'indice
  - La dimensione del blocco della cache è di 2^m parole, ossia 2^{m+2} byte, per cui *m* bit vengono usati per individuare una parola all'interno di un blocco, mentre 2 bit per individuare un byte all'interno di una parola
- La dimensione del campo tag è: 32 - (n+m+2)
- Il numero totale di bit contenuti in una cache a mappatura diretta è:

$$2^n \times (\text{dimensione\_blocco} + \text{dimensione\_tag} + \text{bit\_validità})$$

$$2^n \times (2^m \times 32 + (32-(n+m+2)) + 1)$$

Mappatura di un indirizzo

- Consideriamo ora una cache con 64 blocchi di 16 byte ciascuno. L'indirizzo 1200 (in byte) può essere ricavato come:

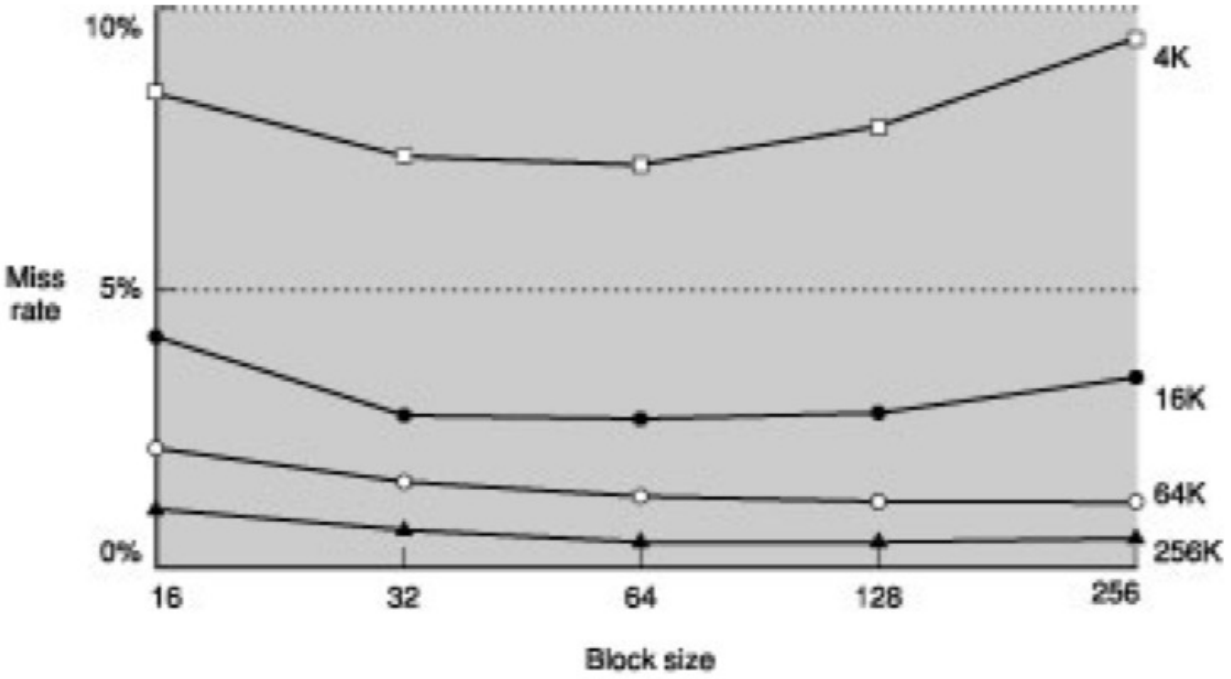
(indirizzo del blocco) modulo (numero di blocchi nella cache)

$$\frac{1200}{16} = 75 \qquad mod \qquad 64$$

$$indirizzo\ del\ blocco = \frac{Indirizzo\ del\ dato\ in\ byte}{Byte\ per\ blocco} = 75$$

Dimensione del blocco di cache

- La dimensione del blocco di cache è in stretta relazione con la frequenza di miss.



Scelta della dimensione del blocco

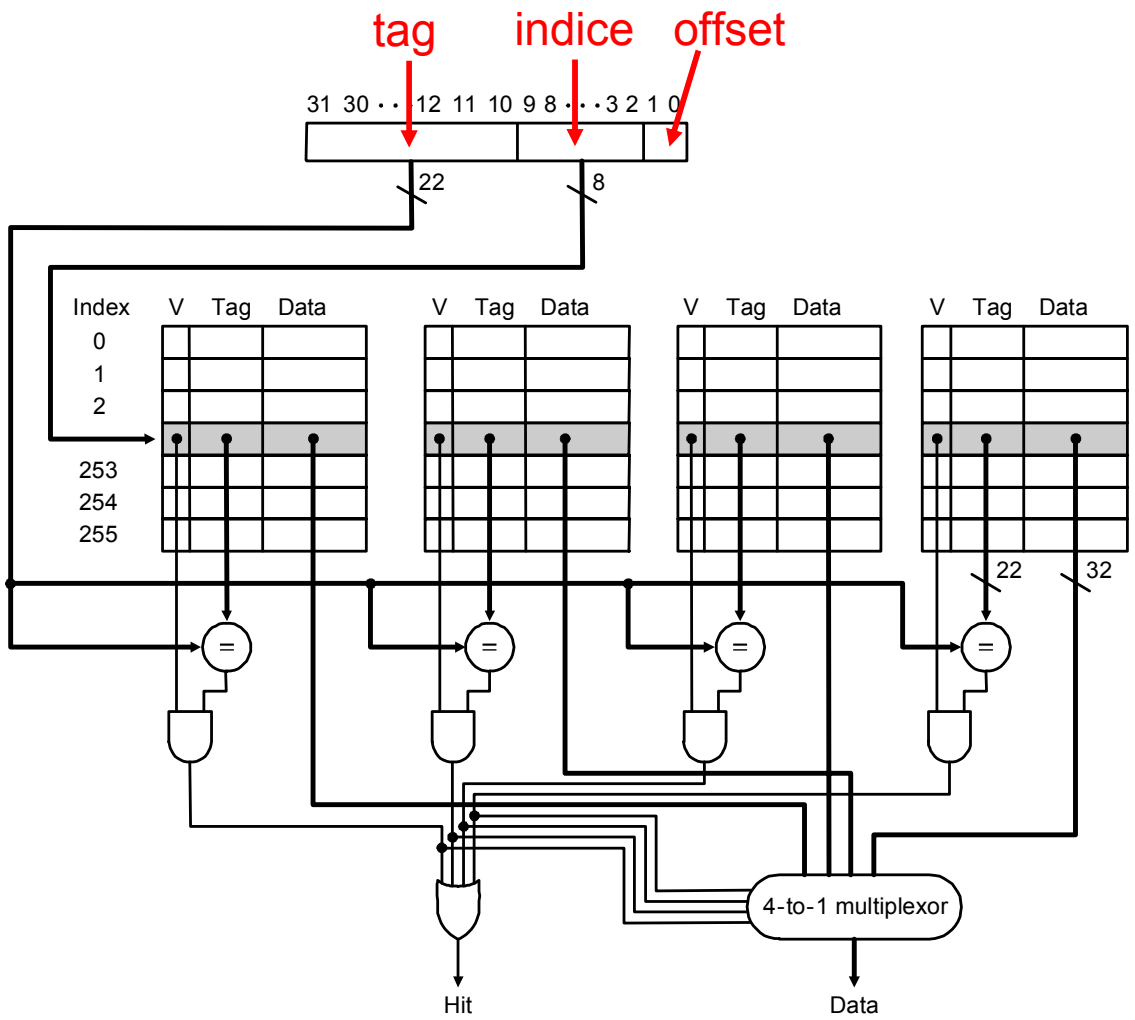
- Un'ampia dimensione per il blocco permette di sfruttare la località spaziale, MA:
  - blocchi di grossa dimensione comportano maggiori miss penalty*:
    - è necessario più tempo per trasferire il blocco
  - se la dimensione del blocco è troppo grossa rispetto alla dimensione della cache, il miss rate aumenta
    - Il numero di blocchi nella cache è insufficiente

* MISS PENALTY = differenza tra il tempo di accesso al livello inferiore e il tempo di accesso alla cache

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

Cache 4-way Set Associative



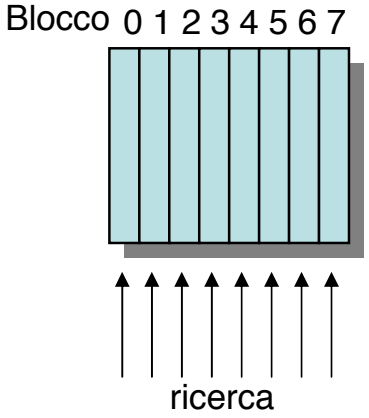
Svantaggi della cache set associative

- 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

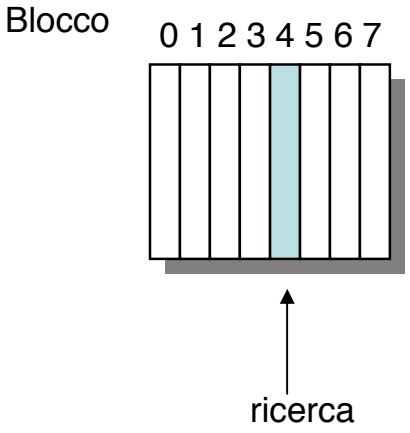
- In una cache ad accesso diretto, il blocco è disponibile prima della decisione Hit/Miss

Confronto tra le tecniche di indirizzamento

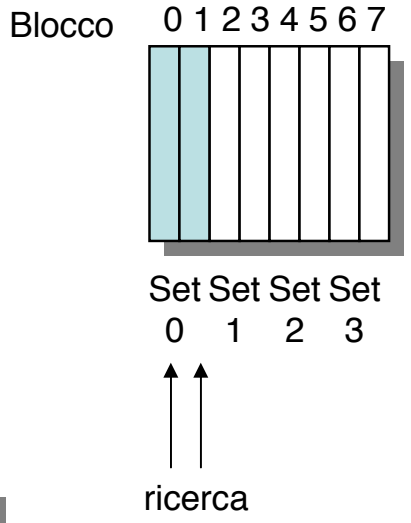
Fully associative:
il blocco 12 può essere disposto ovunque

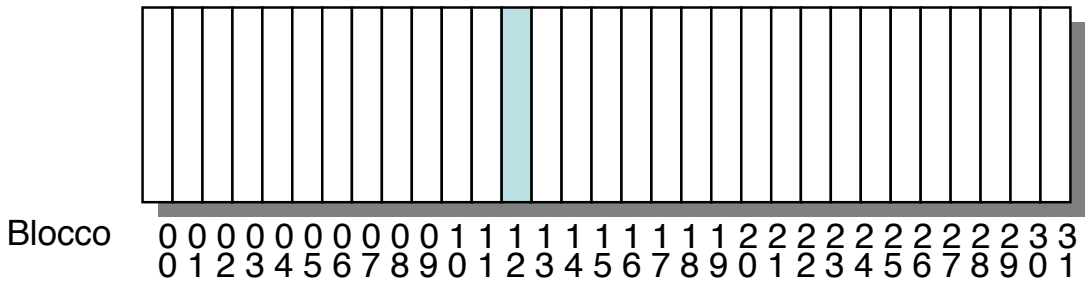


Direct mapped:
il blocco 12 può essere disposto solo sul blocco 4 (12 mod 8)



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





Dimensione del tag e associatività

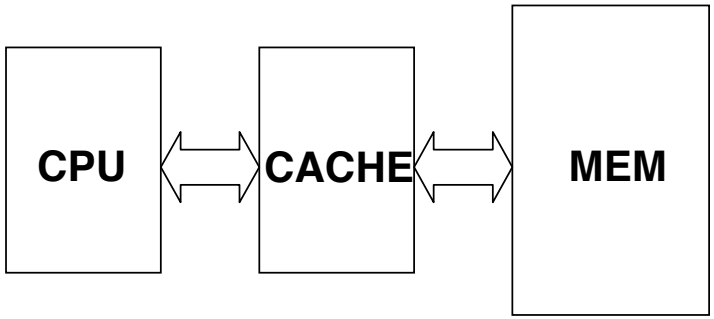
- Aumentando il grado di associatività
  - Aumenta il numero dei comparatori ed il numero di bit per il tag
- Esempio
  - Cache con 4K blocchi, blocco di 4 parole, indirizzo a 32 bit
  - $r = \log_2(4*4)=4 \rightarrow n-r = (32-4) = 28$ bit per tag e indice
  - Cache ad indirizzamento diretto
    - $s = \log_2(4K) = 12$
    - Bit di tag totali = (28-12)*4K = 64K
  - Cache set-associativa a 2 vie
    - $s = \log_2(4K/2) = 11$
    - Bit di tag totali = (28-11)*2*2K = 68K
  - Cache set-associativa a 4 vie
    - $s = \log_2(4K/4) = 10$
    - Bit di tag totali = (28-10)*4*1K = 72K
  - Cache completamente associativa
    - s = 0
    - Bit di tag totali = 28*4K*1 = 112K

Incremento dell'associatività

- Principale vantaggio
  - Diminuzione del miss rate
- Principali svantaggi
  - Maggior costo implementativo
  - Incremento dell'hit time
- La scelta tra cache ad indirizzamento diretto, set- associativa e completamente associativa dipende dal costo dell'associatività rispetto alla riduzione del miss rate

Gestione della cache

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

Algoritmi per la sostituzione di blocchi

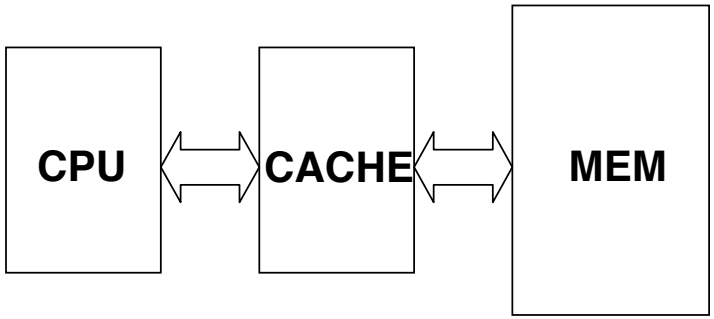
- Nella cache ad accesso diretto: se il blocco di memoria è mappato in una linea di cache già occupata (conflict miss), si elimina il contenuto precedente della linea e si rimpiazza con il nuovo blocco
- Quale blocco sostituire in caso di (capacity) miss nelle cache set- o fully- associative?
  - In caso di cache completamente associativa: ogni blocco è un potenziale candidato per la sostituzione
  - In caso di cache set-associativa a N vie: bisogna scegliere tra gli N blocchi del set

Algoritmi per la sostituzione di blocchi

- Politica di sostituzione **Random** – Scelta casuale
- Politica di sostituzione **Least Recently Used** (**LRU**)
  - Sfruttando la località temporale, il blocco sostituito è quello che non si utilizza da più tempo
  - Ad ogni blocco si associa un contatore all'indietro, che viene portato al valore massimo in caso di accesso e decrementato di 1 ogni volta che si accede ad un altro blocco
- Politica di sostituzione **First In First Out** (**FIFO**)
  - Si approssima la strategia LRU selezionando il blocco più vecchio anziché quello non usato da più tempo

Gestione della cache

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

Gestione dei miss in lettura

- Se un blocco non è presente nella cache bisogna mettere in stallo l'intera CPU
- In generale al verificarsi di un miss nella cache delle istruzioni sono necessari i seguenti passi:
  1. Inviare PC – 4 (uscita della ALU) alla memoria
  2. Lettura dalla memoria
  3. Scrittura nella cache (dato, tag e bit di validità)
  4. Riavviare l'esecuzione dell'istruzione che ha causato il miss.

Cache - Accesso in scrittura

- Scrivere un dato nella cache significa creare un'incoerenza, se non si aggiornano i livelli inferiori della gerarchia di memorie.
- Tale aggiornamento richiede lo stallo della CPU.
- Due tecniche risolutive:
  - Write-through:
  - Write-back
  - Utilizzo di un write buffer

Cache: Write-through

- Write-through: i dati sono scritti nel blocco della cache e nel blocco del livello inferiore.
- Vantaggi
  - E' la soluzione più semplice da implementare
  - Si mantiene la coerenza delle informazioni nella gerarchia di memorie
- Svantaggi
  - Le operazioni di scrittura vengono effettuate alla velocità della memoria di livello inferiore → diminuiscono le prestazioni
  - Aumenta il traffico sul bus di sistema

Cache: Write-back

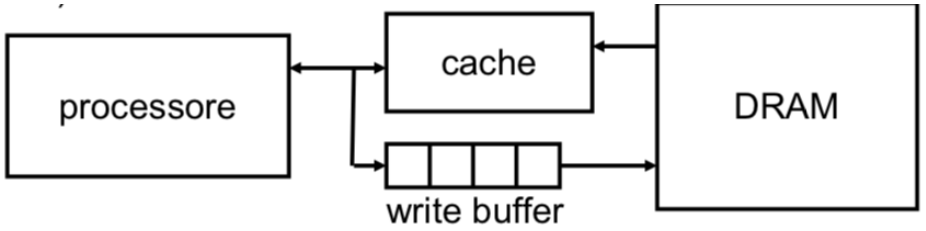
- Write-back (o copy-back): i dati sono scritti solo nel blocco della cache.

  Il blocco modificato viene scritto nel livello inferiore della gerarchia solo quando deve essere sostituito.

- Vantaggi
  - Le scritture avvengono alla velocità della cache
  - Scritture successive sullo stesso blocco alterano solo la cache
- Svantaggi
  - Ogni sostituzione del blocco può provocare un trasferimento in memoria

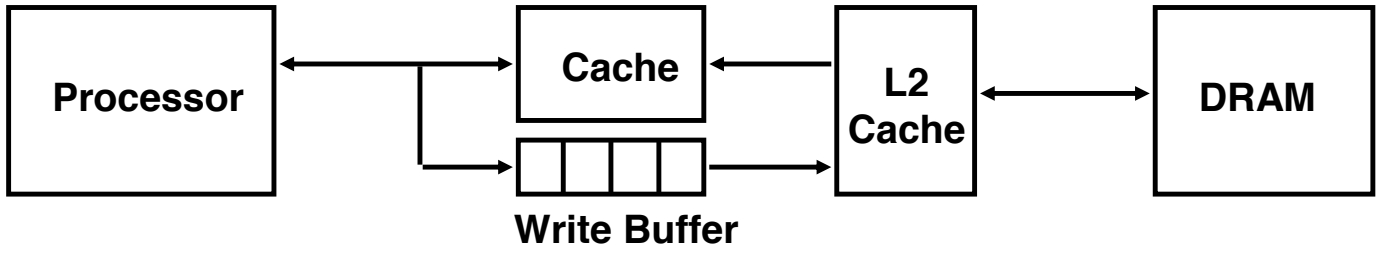
Cache: write-through con write buffer

- *Write buffer* interposto tra la cache e la memoria di livello inferiore
  - Il processore scrive i dati sia nella cache e nel write buffer
  - Il controller della memoria scrive il contenuto del write buffer in memoria
- Il write buffer è gestito in modalità FIFO
  - Numero tipico di elementi del buffer: 4
  - Efficiente se la frequenza di scrittura << 1/write cycle della DRAM
  - Altrimenti, il buffer può andare in saturazione ed il processore deve aspettare che le scritture giungano a completamento (*write stall*)



Cache: write-through con write buffer

- *Write buffer* interposto tra la cache e la memoria di livello inferiore
  - Il processore scrive i dati sia nella cache e nel write buffer
  - Il controller della memoria scrive il contenuto del write buffer in memoria
- Il write buffer è gestito in modalità FIFO
  - Numero tipico di elementi del buffer: 4
  - Efficiente se la frequenza di scrittura << 1/write cycle della DRAM
  - Altrimenti, il buffer può andare in saturazione ed il processore deve aspettare che le scritture giungano a completamento (*write stall*)



Write miss

- Le scritture possono indurre **write miss**
- Soluzioni possibili:
  - **Write allocate**: il blocco viene caricato in cache e si effettua la scrittura
  - **No-write allocate**: il blocco viene scritto direttamente nella memoria di livello inferiore, senza essere trasferito in cache

Osservazione (I)

- Una cache con blocchi di dimensioni maggiori sfruttano maggiormente la località spaziale diminuendo la frequenza di miss
- La frequenza di miss torna a crescere se la dimensione dei blocchi diventa troppo grande rispetto alla dimensione della cache
- Quindi i blocchi vengono scaricati dalla cache prima ancora che molti dati in essi contenuti siano stati utilizzati
- Quindi la località spaziale tra le parole di un blocco diminuisce e il miglioramento legato alla frequenza di miss si riduce

Osservazione (II)

- Inoltre, cresce anche il costo di una miss: la penalità di una miss è determinata dal tempo necessario a prelevare un blocco dal livello sottostante e a scriverlo nella cache
- Il tempo per prelevare un blocco è dato dalla somma tra la latenza per ottenere la prima parola del blocco e il tempo di trasferimento del resto del blocco