Corso di Laurea in Informatica
Architettura degli elaboratori
a.a. 2025-2026

Architettura degli Elaboratori 2025-2026

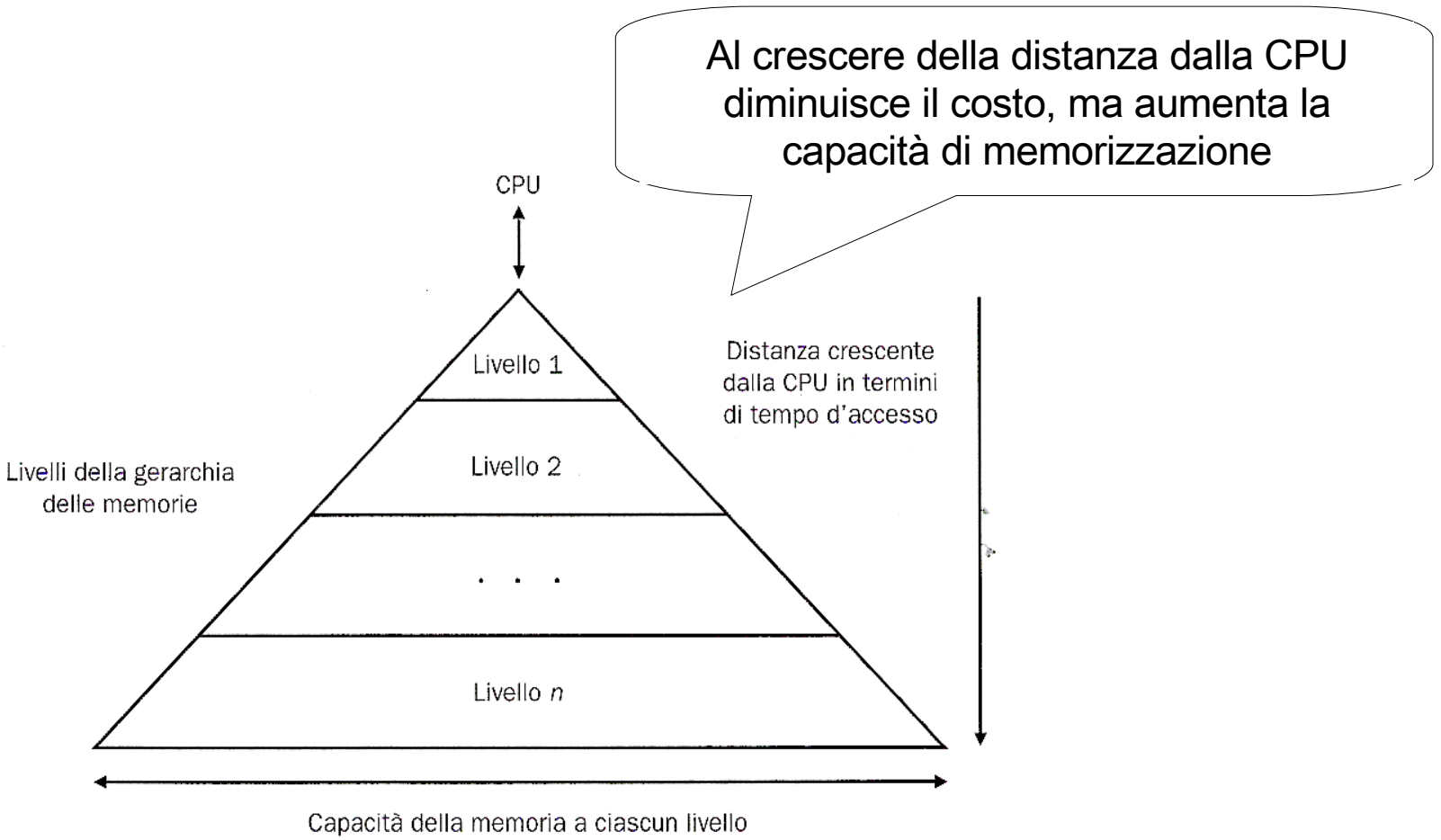
Gerarchie di memoria e cache

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 di memoria
- Miss (fallimento nell'accesso): il dato non è presente nel blocco ed occorre accedere al livello sottostante.

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

Definizioni (II)

- Tempo di hit: tempo di accesso al livello 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

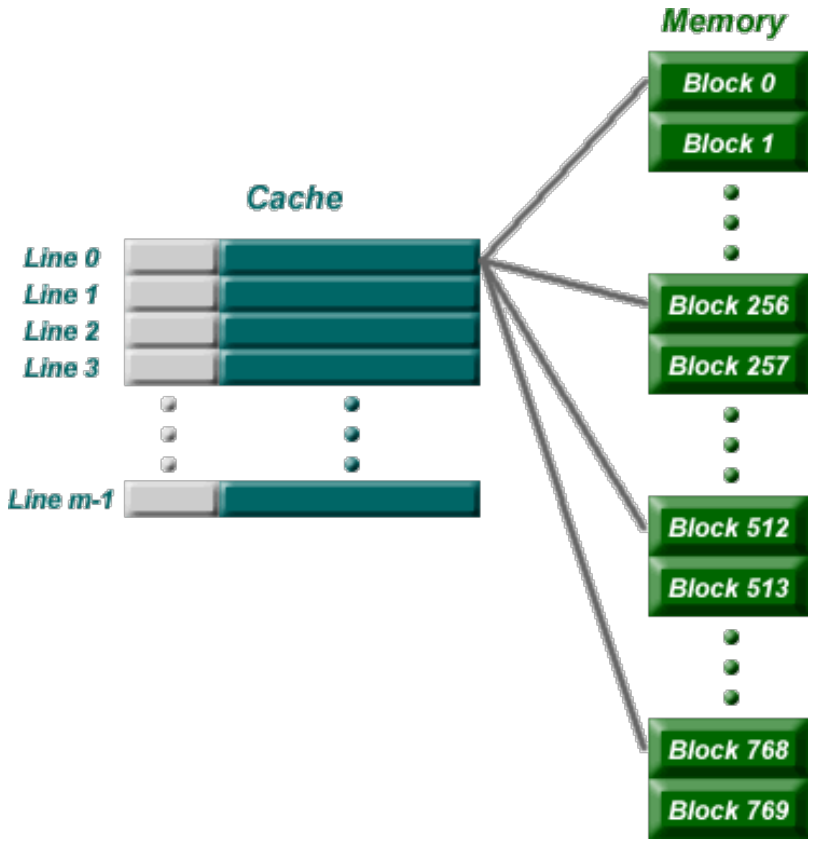
Cache

- 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)

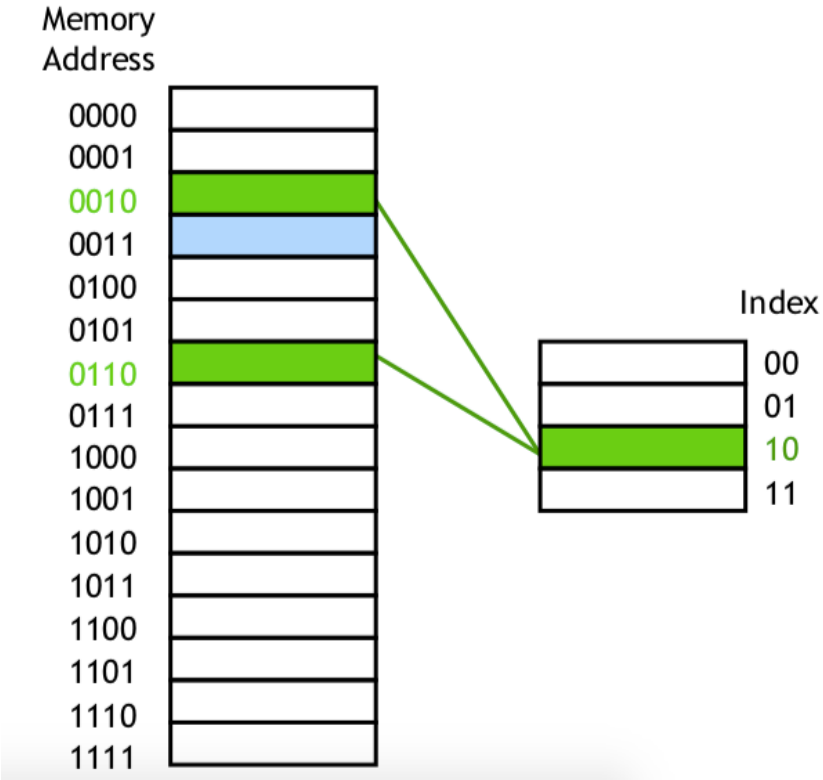
Cache

- 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

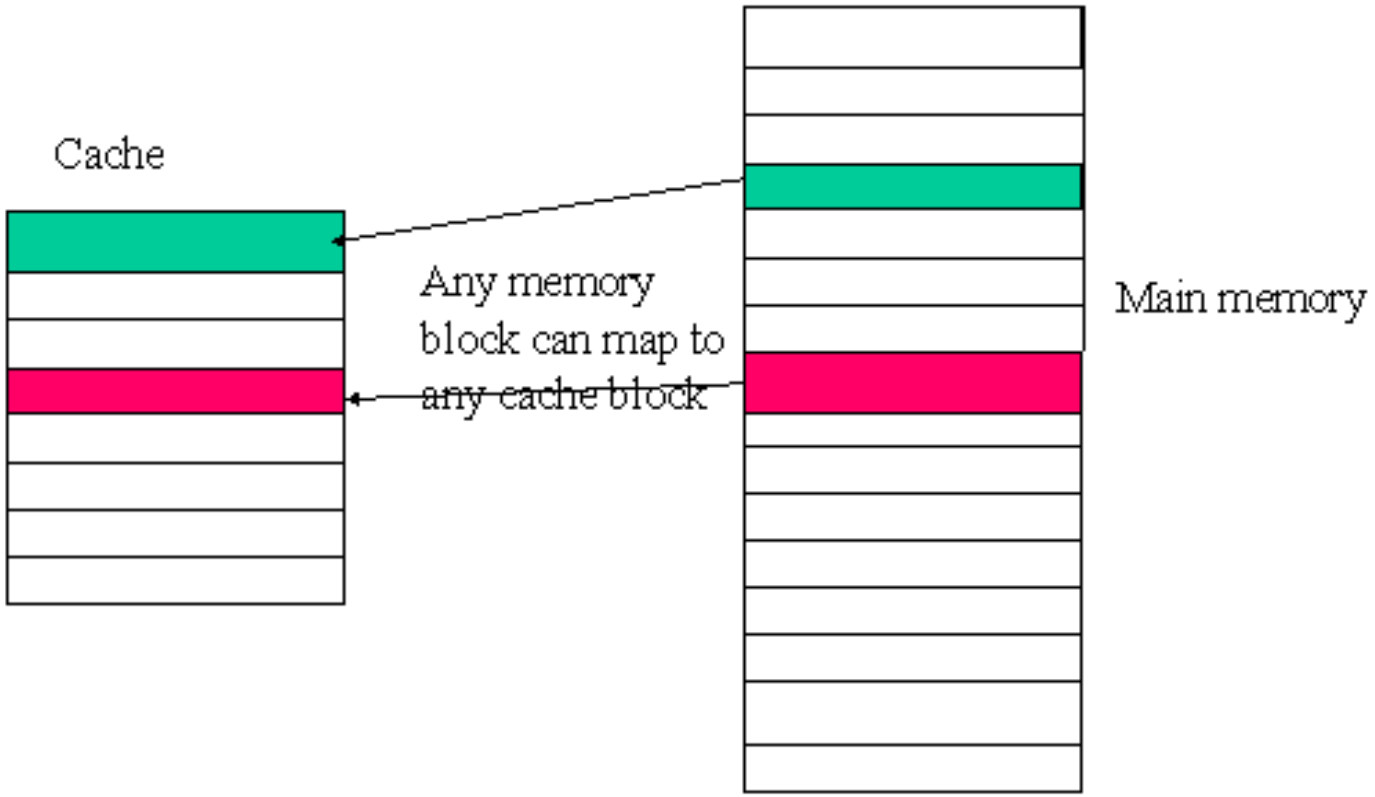
Direct Mapping



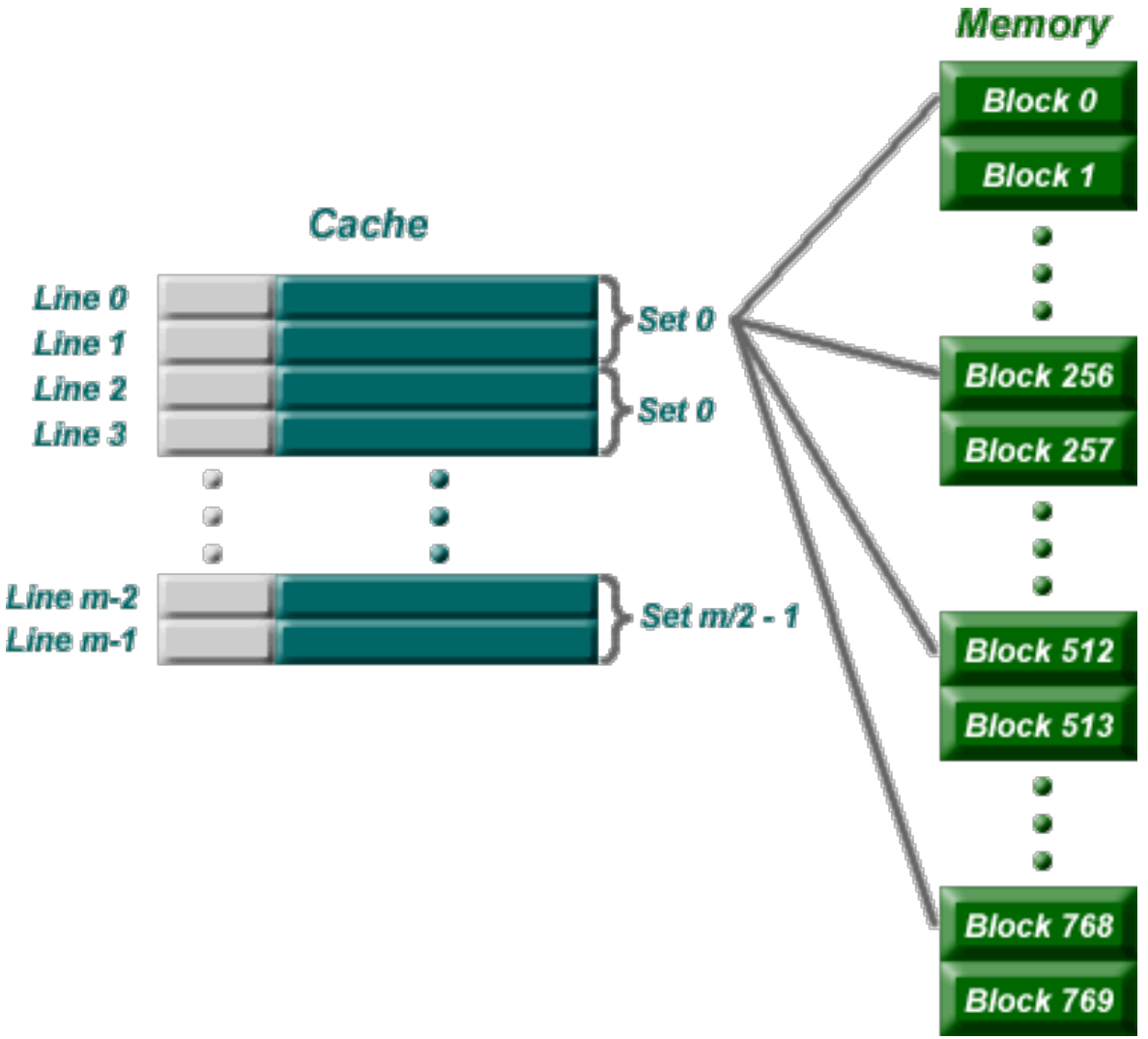
Direct Mapping



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
- **Indirizzo**:
  - 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
  - Offset: bit necessari per selezionare il byte richiesto nella parola

Direct Mapped Cache

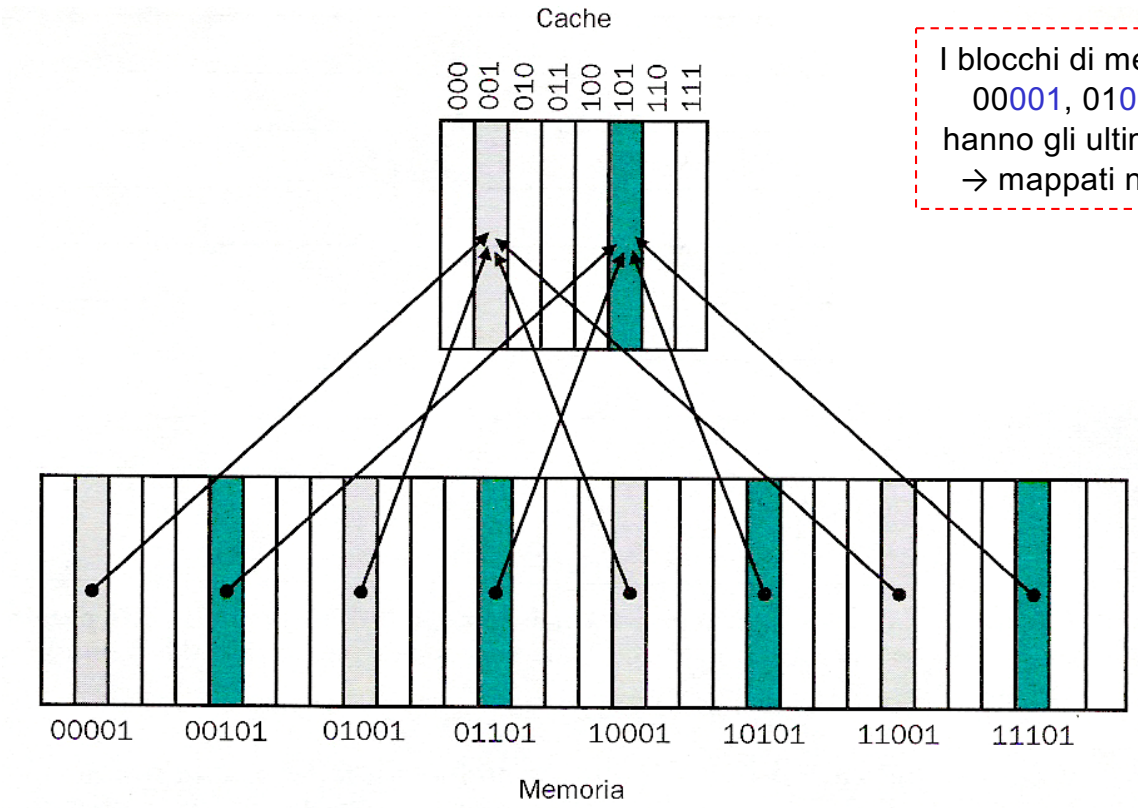
- Dato che il numero di blocchi nella cache è una potenza di 2, la posizione corrispondente della parola in cache è data dai log2(numero elementi nella cache) bits meno significativi dell'indirizzo in memoria principale.
- Esempio:
  - Numero di elementi nella cache: 8
  - Bit per indirizzare una locazione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

Contenuto di una linea di cache

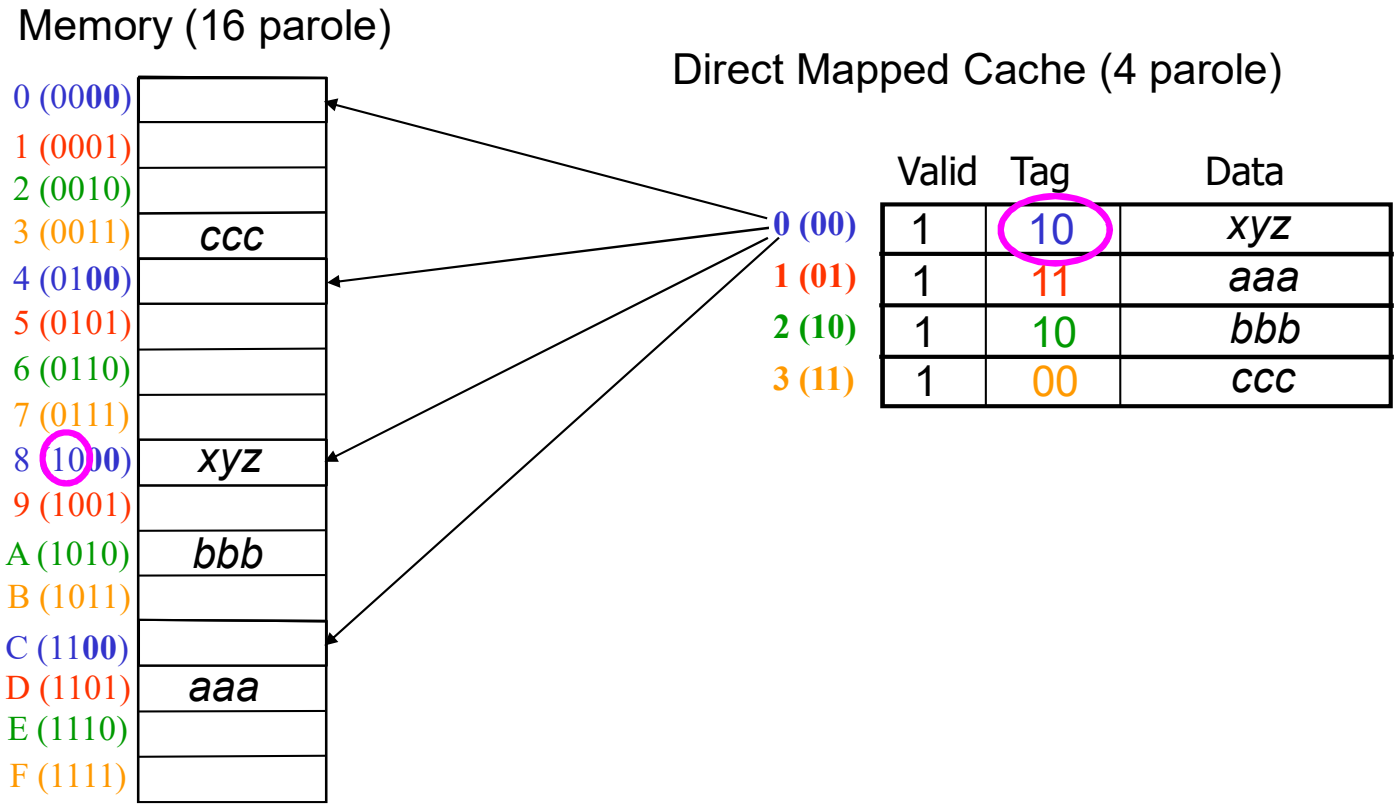
- In una cache ad indirizzamento diretto ogni linea di cache include:
  - Il **bit di validità**: indica se i dati nella linea di cache sono validi
    - All'avvio, tutte le linee sono non valide (*compulsory miss*)
  - Il **tag** (etichetta): consente di individuare in modo univoco il **blocco** in memoria che è stato ***mappato*** nella linea di cache

    contiene informazioni necessarie a verificare se una parola della cache corrisponde o meno alla parola cercata
  - Il **blocco di dati**: vero e proprio, formato da una o più parole

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

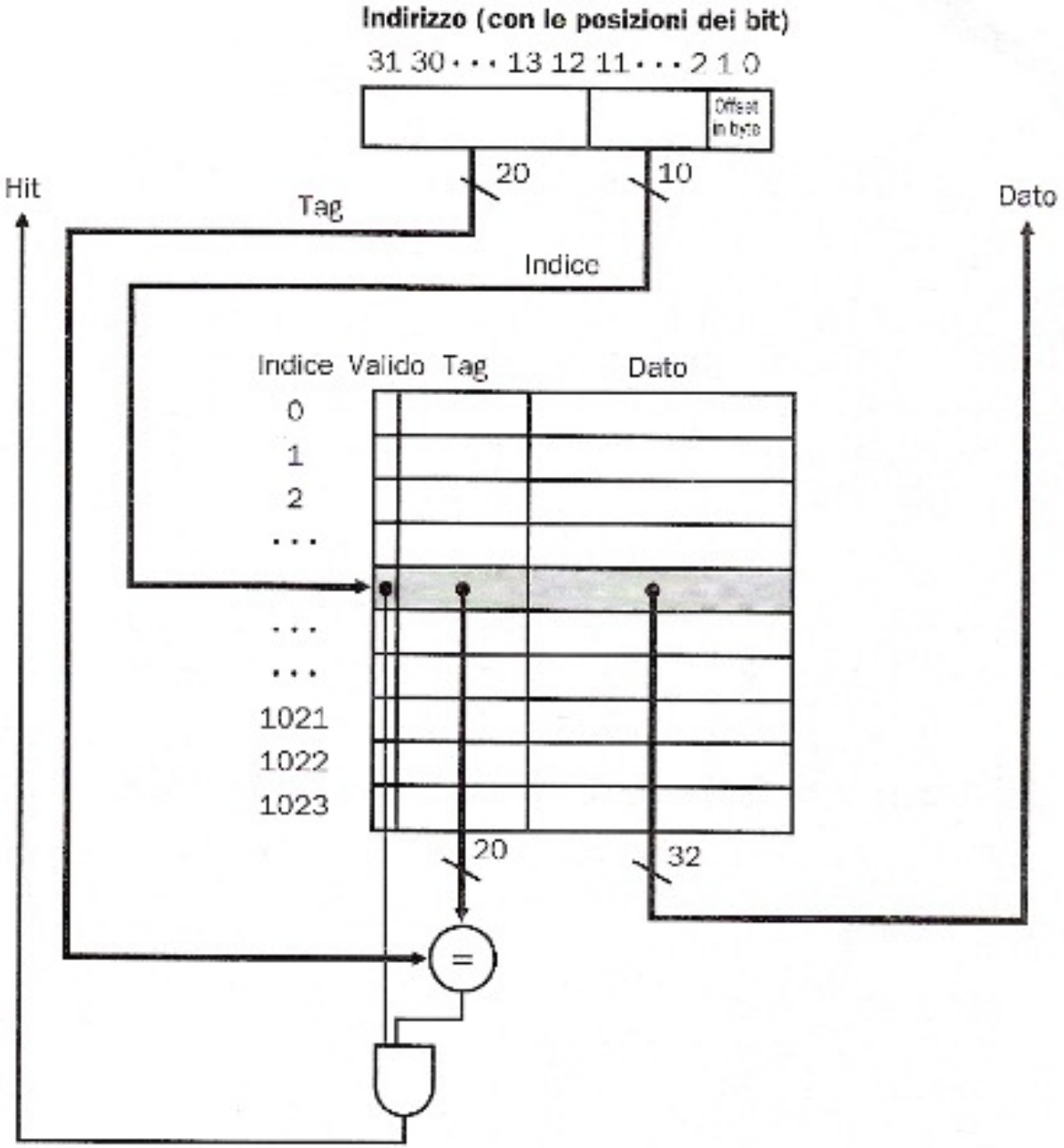

Memory (16 parole)
0 (0000)
1 (0001)
2 (0010)
3 (0011)
ccc
4 (0100)
5 (0101)
6 (0110)
7 (0111)
8 (1000)
xyz
9 (1001)
A (1010)
bbb
B (1011)
C (1100)
D (1101)
aaa
E (1110)
F (1111)
Direct Mapped Cache (4 parole)
Valid
Tag
Data
0 (00)
1 (01)
2 (10)
3 (11)
1
10
xyz
1
11
aaa
1
10
bbb
1
00
ccc

Indirizzo

Per indirizzare una cache a mappatura diretta abbiamo bisogno di un indirizzo di memoria a n bit, esempio 32 bit:

- blocchi da **4 parole** (quindi servono **2 bit di offset**: $2^2=4$),
- 256 righe di cache (quindi servono **8 bit di indice**: $2^8=256$),
- il resto (22 bit) è il tag.

Tag (22 bit)	Indice (8 bit)	Offset (2 bit)


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
=

Gestione di cache hit e cache miss

- In caso di hit (il processore continua il processamento):
  - Accesso al dato dalla cache dati
  - Accesso all'istruzione dalla cache istruzioni

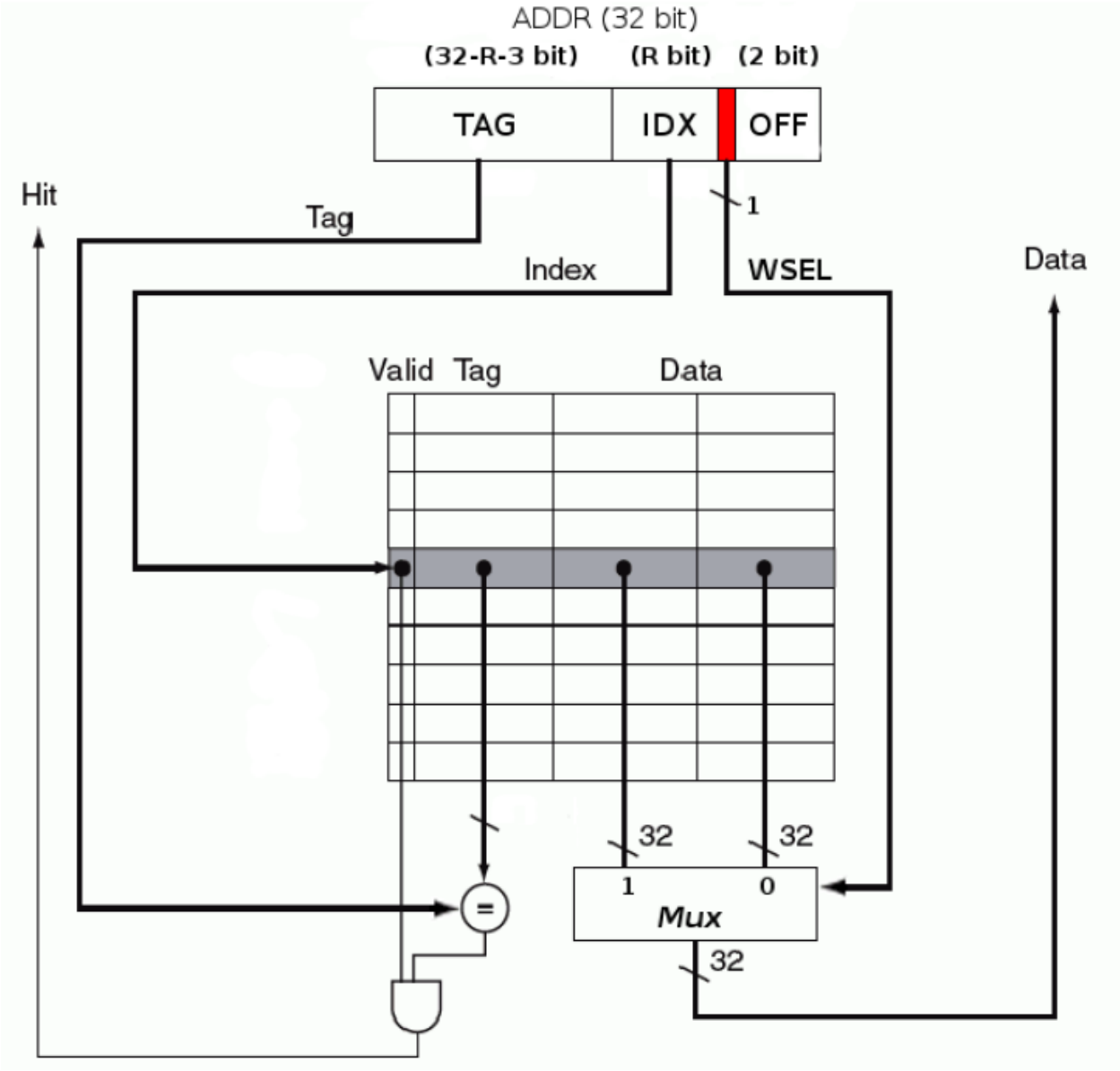
- In caso di miss:
  - Stallo del processore (come nel pipelining) in attesa di ricevere l'elemento dalla memoria
  - Invio dell'indirizzo al controller della memoria (simile ad un DMAC, identificato anche come MMU- Memory Management Unit)
  - Reperimento dell'elemento dalla memoria
  - Caricamento dell'elemento in cache
  - Ripresa dell'esecuzione

Indirizzo (recap)

- 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:

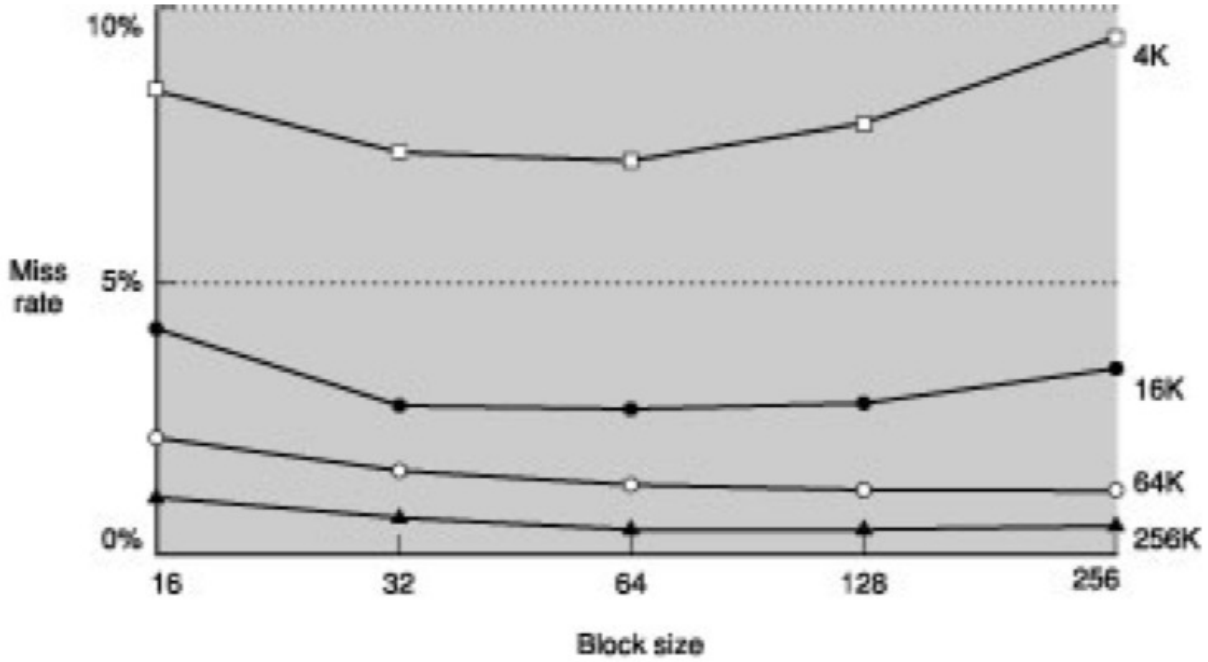
(indirizzo del blocco) modulo (numero di blocchi nella cache)

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

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

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

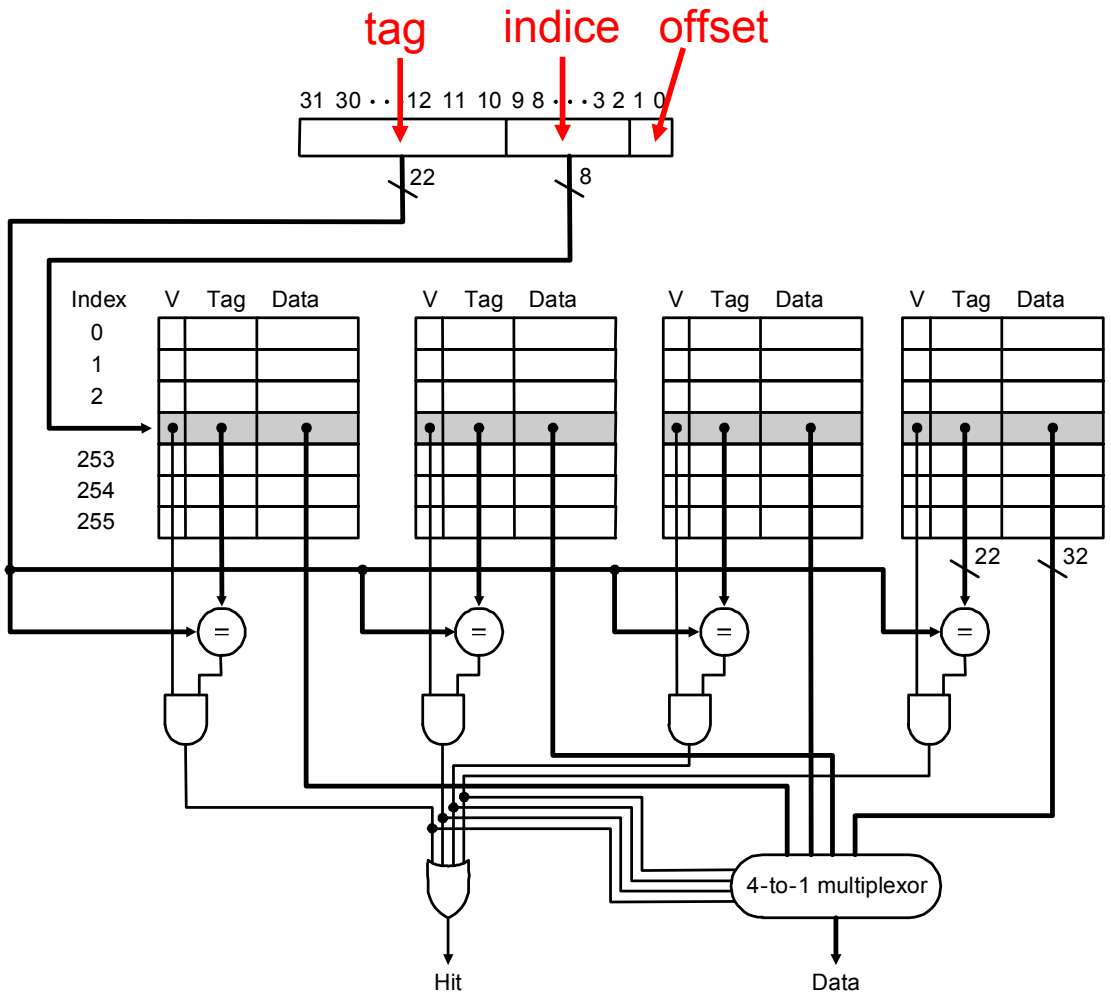
- Un'ampia dimensione per il blocco permette di sfruttare la località spaziale, MA:
  - blocchi di grossa dimensione comportano maggiori miss penalty*:
    - è necessario più tempo per trasferire il blocco
  - se la dimensione del blocco è troppo grossa rispetto alla dimensione della cache, il miss rate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



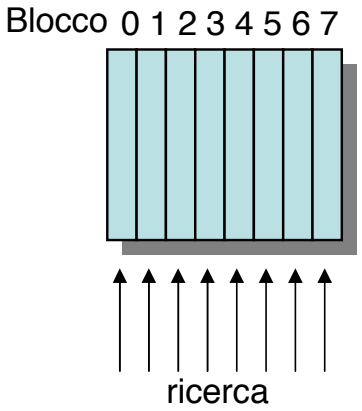
Svantaggi della cache set associative

- 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

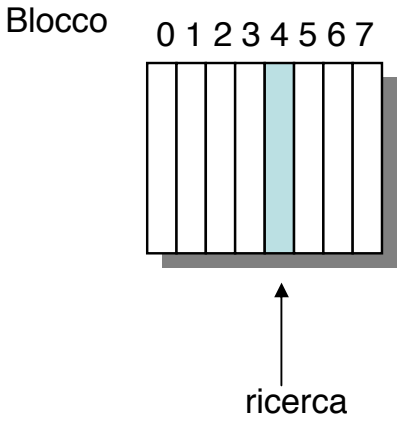
- In una cache ad accesso diretto, il blocco è disponibile prima della decisione Hit/Miss

Confronto tra le tecniche di indirizzamento

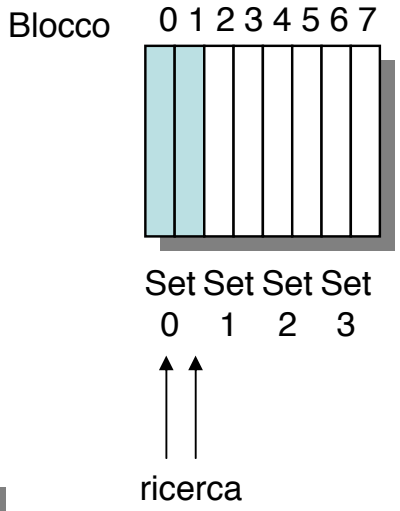
Fully associative:
il blocco 12 può essere disposto ovunque

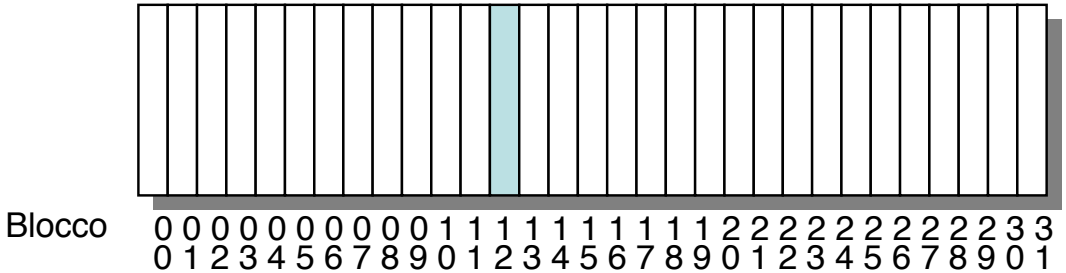


Direct mapped:
il blocco 12 può essere disposto solo sul blocco 4 (12 mod 8)



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





Memoria di livello inferiore

Incremento dell’associatività

Vantaggi dell'Associatività Elevata:

1. **Riduzione dei Conflitti:** Con un'associatività più alta, i blocchi di memoria hanno più posizioni candidate, riducendo la probabilità che due blocchi "competano" per lo stesso slot, diminuendo la probabilità di conflitto.
2. **Maggiore Flessibilità:** Le cache più associative sfruttano meglio lo spazio disponibile.
3. **Prestazioni Migliori:** Riducendo i miss, si riduce anche la frequenza di accesso alla memoria principale, diminuendo la latenza media e migliorando il throughput del processore.

Incremento dell'associatività

Svantaggi dell'Associatività Elevata:

1. **Aumento della Complessità Hardware**: cache più associative richiedono **più comparatori** e **logica** per determinare se un blocco è presente, aumentando i costi di produzione e il consumo energetico. Questa logica aggiuntiva può **rallentare** leggermente l'accesso ai dati.
2. **Maggiore Consumo di Energia**: con più associatività, ogni accesso alla cache implica la verifica di più tag, **aumentando il consumo energetico.**