



Introduzione alle memorie cache

Corso ACSO – prof. Cristina SILVANO
Politecnico di Milano

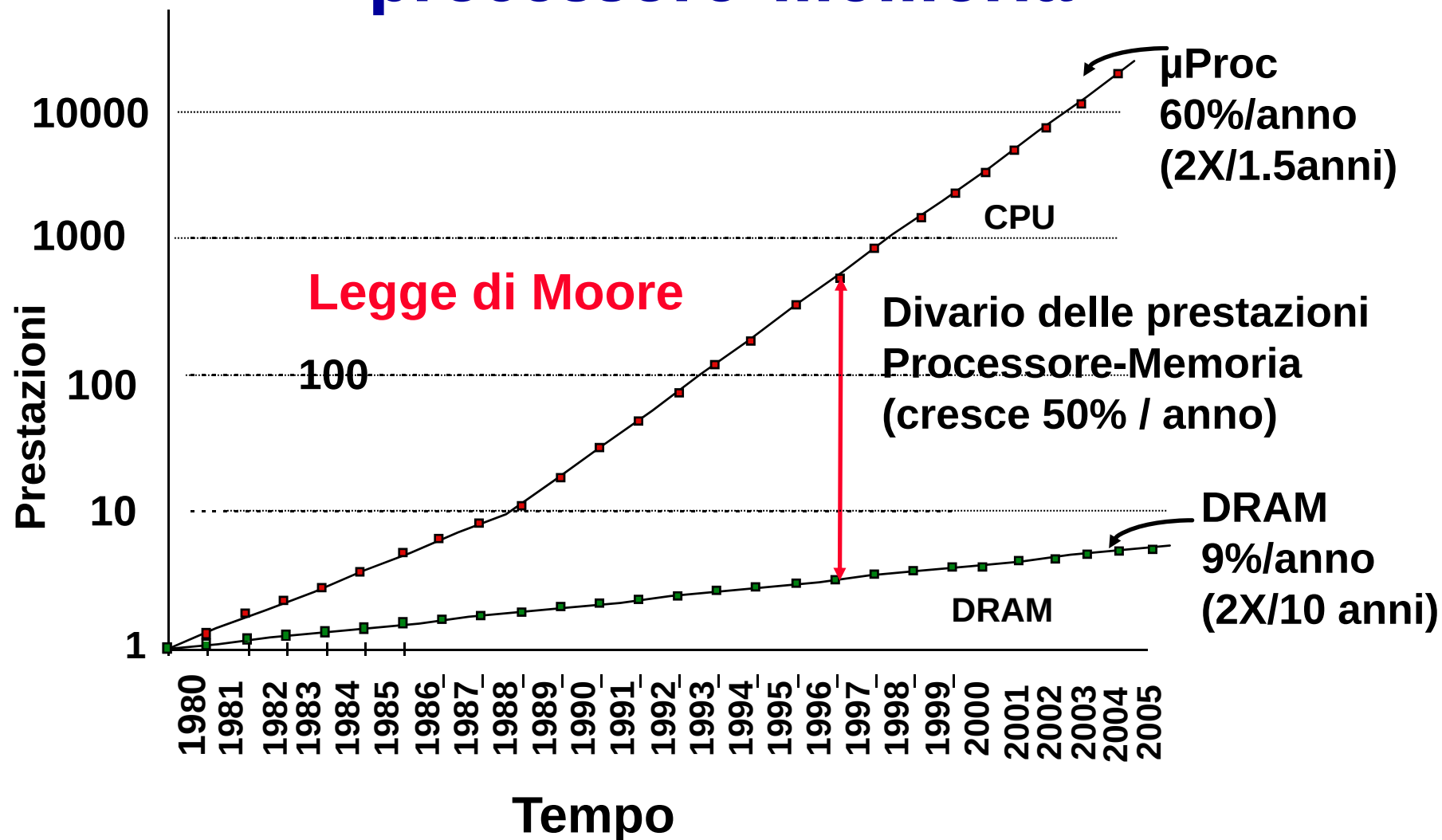
Sommario

- Obiettivo
- Livelli della gerarchia di memoria
- Memoria cache: concetti base
- Architettura di una memoria cache:
 - Cache a indirizzamento diretto
 - Cache completamente associativa
 - Cache set-associativa a n vie
- Sostituzione di un blocco (o cache line)

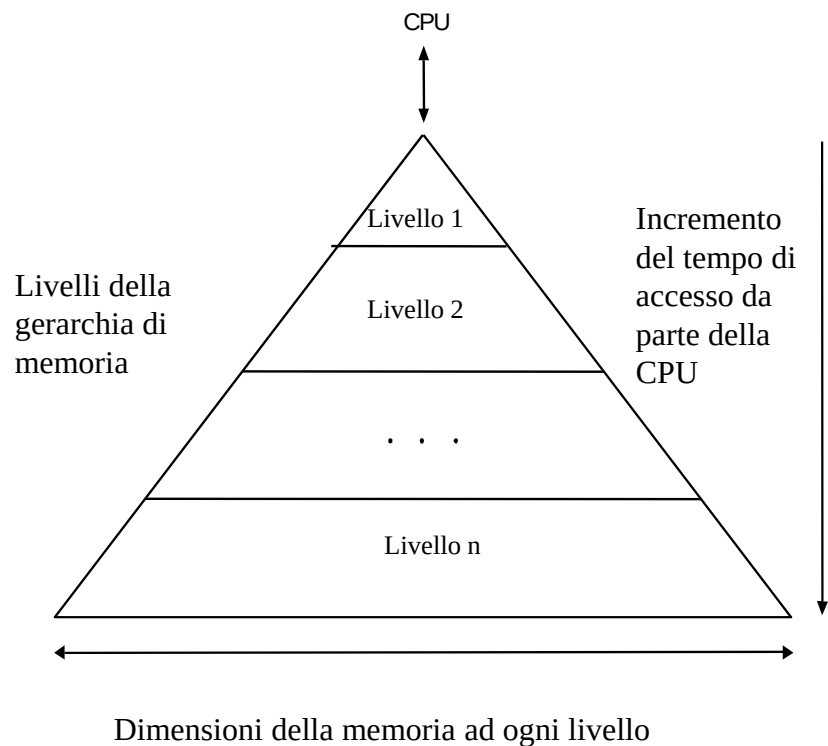
Obiettivo

- Migliorare le prestazioni di un calcolatore attraverso il sistema di memoria in modo da:
 - fornire agli utenti l'illusione di poter usufruire di una memoria contemporaneamente grande e veloce
 - fornire al processore i dati alla velocità con cui è in grado di elaborarli

Problema: divario delle prestazioni processore-memoria



Soluzione: gerarchia di memoria



- Utilizzare diversi livelli di memoria, ciascuno di diversa velocità e dimensione, realizzati con tecnologie diverse in modo da ottenere un buon compromesso costo/prestazioni.
- Obiettivo: fornire all'utente una quantità di memoria pari a quella disponibile nella tecnologia più economica, consentendo allo stesso tempo una velocità di accesso pari a quella garantita dalla tecnologia più veloce

Livelli della gerarchia di memoria

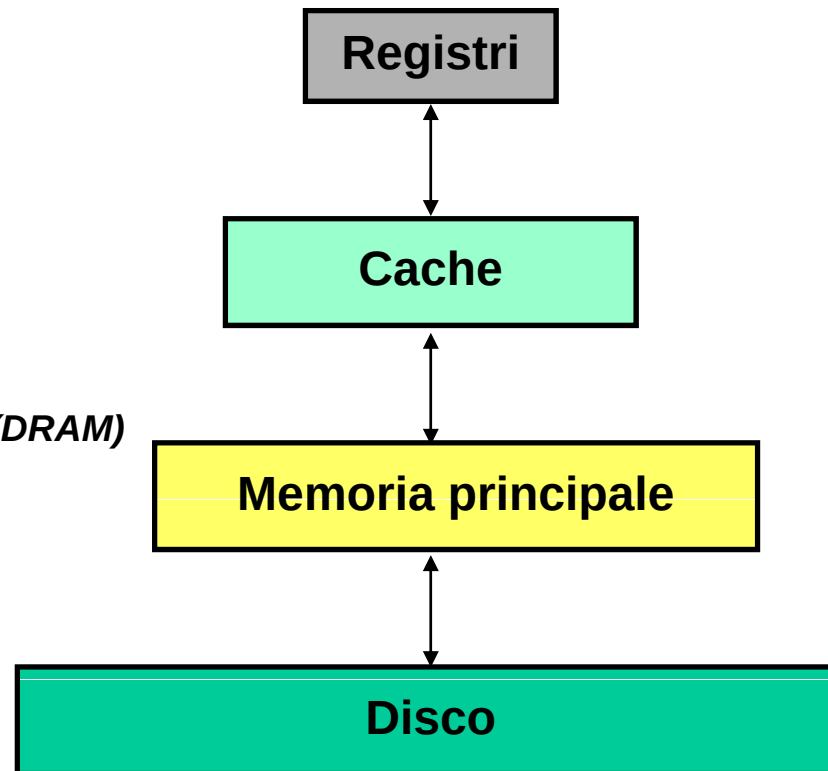
Capacità
Tempo di accesso

Registri CPU
< 1 KByte
0.25 – 0.5 ns

Cache (SRAM)
< 8 MByte
0.5 – 25 ns

Memoria principale (DRAM)
< 4 GByte
10 – 250 ns

Disco
G Byte,
10 ms



Livello superiore

+ veloce

+ costoso

+ grande

Livello inferiore

Soluzione: sfruttare la località dei riferimenti

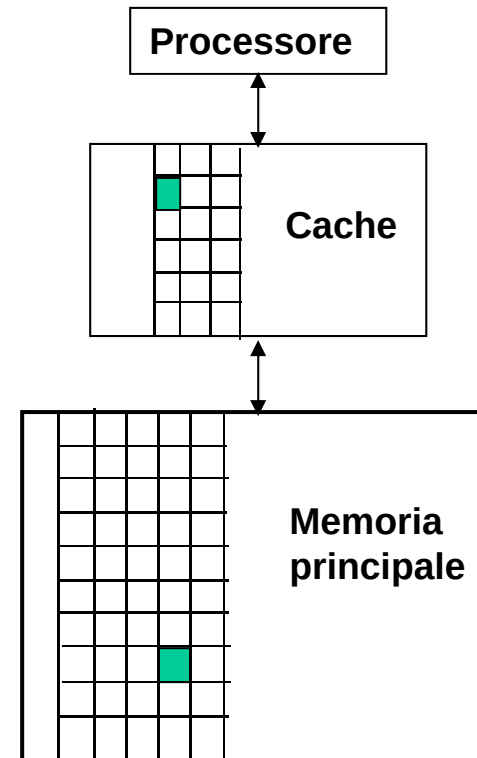
- **Località temporale:** quando si fa riferimento a un elemento di memoria, c'è la tendenza a far riferimento allo stesso elemento entro breve
(ad esempio: riutilizzo di istruzioni e dati contenuti nei cicli).
- **Località spaziale:** quando si fa riferimento a un elemento di memoria, c'è la tendenza a far riferimento entro breve tempo ad altri elementi che hanno indirizzo vicino a quello dell'elemento corrente
(ad esempio: sequenzialità istruzioni e accesso a dati organizzati in vettori o matrici).

Concetti base

- Una gerarchia di memoria è composta da più livelli, ma i dati vengono di volta in volta copiati solo tra *due livelli adiacenti*.
- ***Consideriamo i due livelli: cache e memoria principale***
- La cache (livello ***superiore***) è più piccola, veloce e costosa rispetto alla memoria principale (livello ***inferiore***).
- La **minima quantità di informazione** che può essere presente o assente nella cache è il **blocco o cache line**.
- Per sfruttare la località spaziale è necessario che la dimensione del blocco della cache sia un **multiplo** della dimensione della parola di memoria
 - Es.: dimensione blocco 128 bit = 4 parole da 32 bit
- Il numero di blocchi presenti in cache è dato da:
Num. blocchi in cache = Dim Cache / Dim. Blocco
 - Es.: Una cache da 64 K Byte e blocco da 128 bit (16 Byte) contiene 4 K blocchi

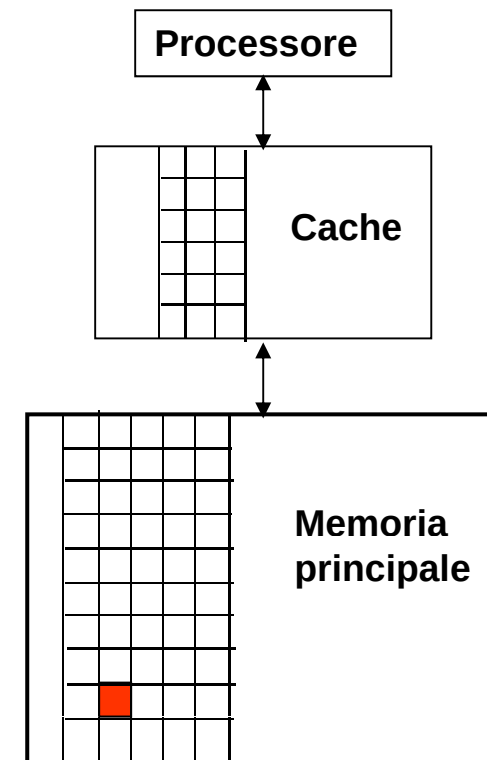
Successo nell'accesso alla cache

- Se il dato richiesto dal processore compare in uno dei blocchi presenti nel livello superiore, si dice che la richiesta ha successo (*hit*).



Fallimento nell'accesso alla cache

- Se il dato non si trova nel livello superiore, si dice che la richiesta fallisce (**miss**)
⇒ per trovare il blocco che contiene i dati richiesti, **bisogna accedere al livello inferiore della gerarchia.**
- In caso di fallimento nell'accesso ad un dato:
 - **stallo** della CPU;
 - richiesta del blocco contenente il dato cercato alla memoria;
 - copia in cache;
 - ripetizione dell'accesso in cache.



Gerarchia di memoria: definizioni

- **Hit (successo):** dati presenti in un blocco del livello superiore.
- **Hit Rate (frequenza dei successi):** numero di accessi a memoria che trovano il dato nel livello superiore rispetto al numero totale di accessi a memoria

$$\text{Hit Rate} = \frac{\# \text{ successi}}{\# \text{ accessi a memoria}}$$

- **Hit Time (tempo di successo):** tempo per accedere al dato nel livello superiore della gerarchia, che comprende anche il tempo necessario per stabilire se il tentativo di accesso si risolve in un successo o in un fallimento.

Gerarchia di memoria : definizioni

- **Miss (fallimento):** i dati devono essere recuperati dal livello inferiore.
- **Miss Rate (frequenza di fallimento) :**
numero di accessi a memoria che non trovano il dato nel livello superiore rispetto al numero totale di accessi a memoria
Miss Rate = $\frac{\text{\# fallimenti}}{\text{\# accessi a memoria}}$
- **Hit Rate + Miss Rate = 1**
- **Miss Penalty (penalizzazione di fallimento):** tempo necessario a sostituire un blocco nel livello superiore.
- **Miss Time (tempo di fallimento) :**
Miss Time = Hit Time + Miss Penalty
- Notare che: **Hit Time << Miss Penalty**

Tempo medio di accesso alla memoria

$$\mathbf{T_m = Hit\ Rate * Hit\ Time + Miss\ Rate * Miss\ Time}$$

Dove:

$$\mathbf{Miss\ Time = Hit\ Time + Miss\ Penalty}$$

$$\mathbf{Hit\ Rate + Miss\ Rate = 1}$$

$$\Rightarrow \mathbf{T_m = Hit\ Time + Miss\ Rate * Miss\ Penalty}$$

Struttura di una cache

Ogni posizione (entry) della cache include:

1. **Valid bit** che indica se questa posizione contiene o meno dati validi. Quando il calcolatore viene acceso tutte le posizioni della cache sono segnalate come NON valide
2. **Campo etichetta (tag)** che contiene il valore che identifica univocamente l'indirizzo di memoria corrispondente ai dati memorizzati
3. **Campo dati** che contiene una copia dei dati (blocco o cache line)

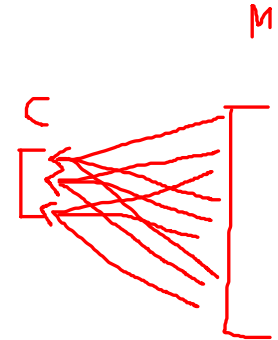


Problema del piazzamento di un blocco

- Problema: dato un indirizzo di un blocco nella memoria principale, determinare la sua posizione nella cache.
- Occorre stabilire una corrispondenza tra l'indirizzo in memoria del blocco e l'indirizzo nella cache.
- Questa corrispondenza dipende dall'architettura della cache:
 - **Cache a indirizzamento diretto**
 - **Cache completamente associativa**
 - **Cache set-associativa a n vie**

Cache a indirizzamento diretto (direct mapped)

- Numero di blocchi = Dim. Cache / Dim. Blocco
- Ogni locazione di memoria corrisponde a una e una sola locazione della cache:



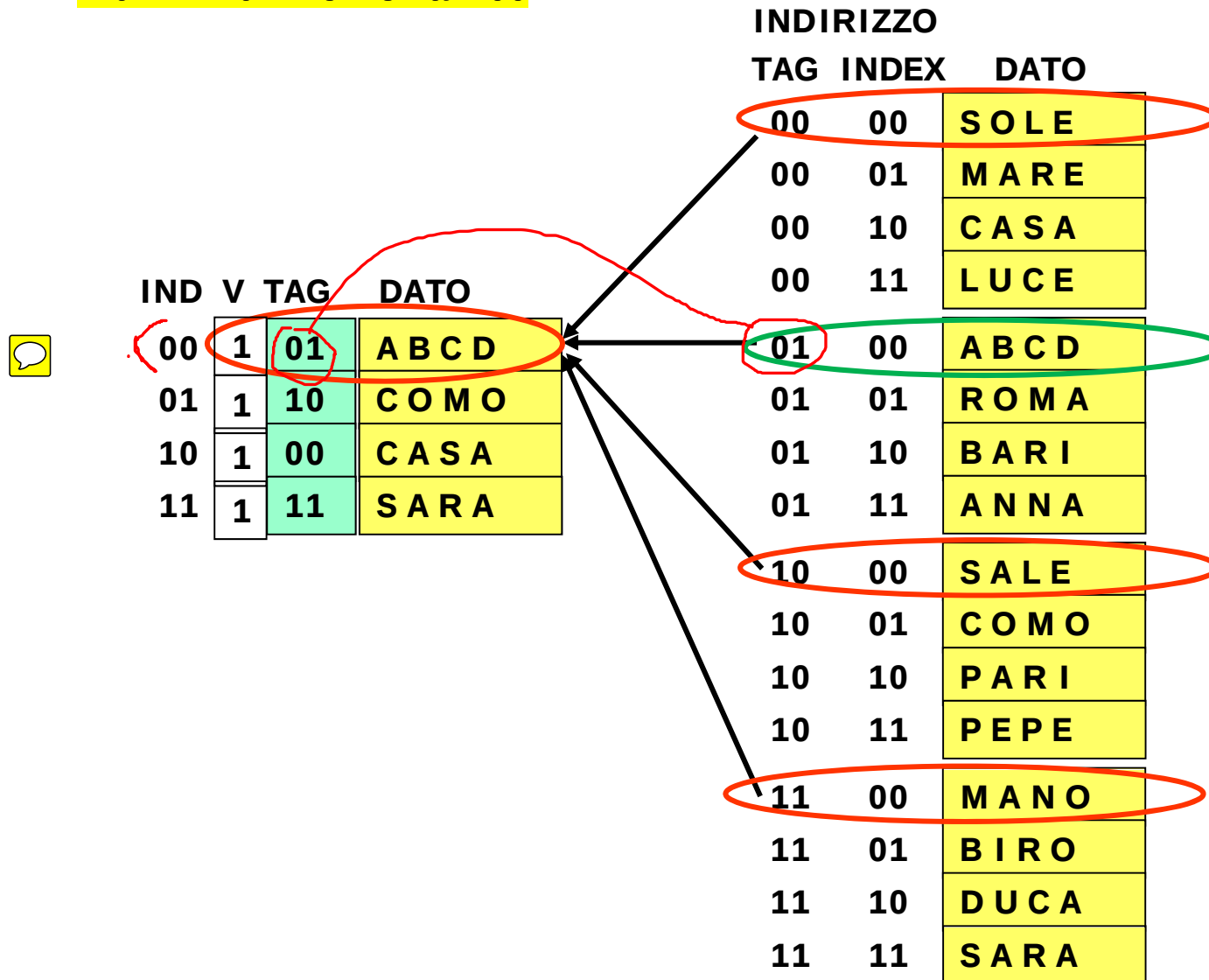
$$(\text{Ind. Bl.})_{\text{cache}} = (\text{Ind. Blocco})_{\text{mem}} \bmod (\text{Num. blocchi in cache})$$



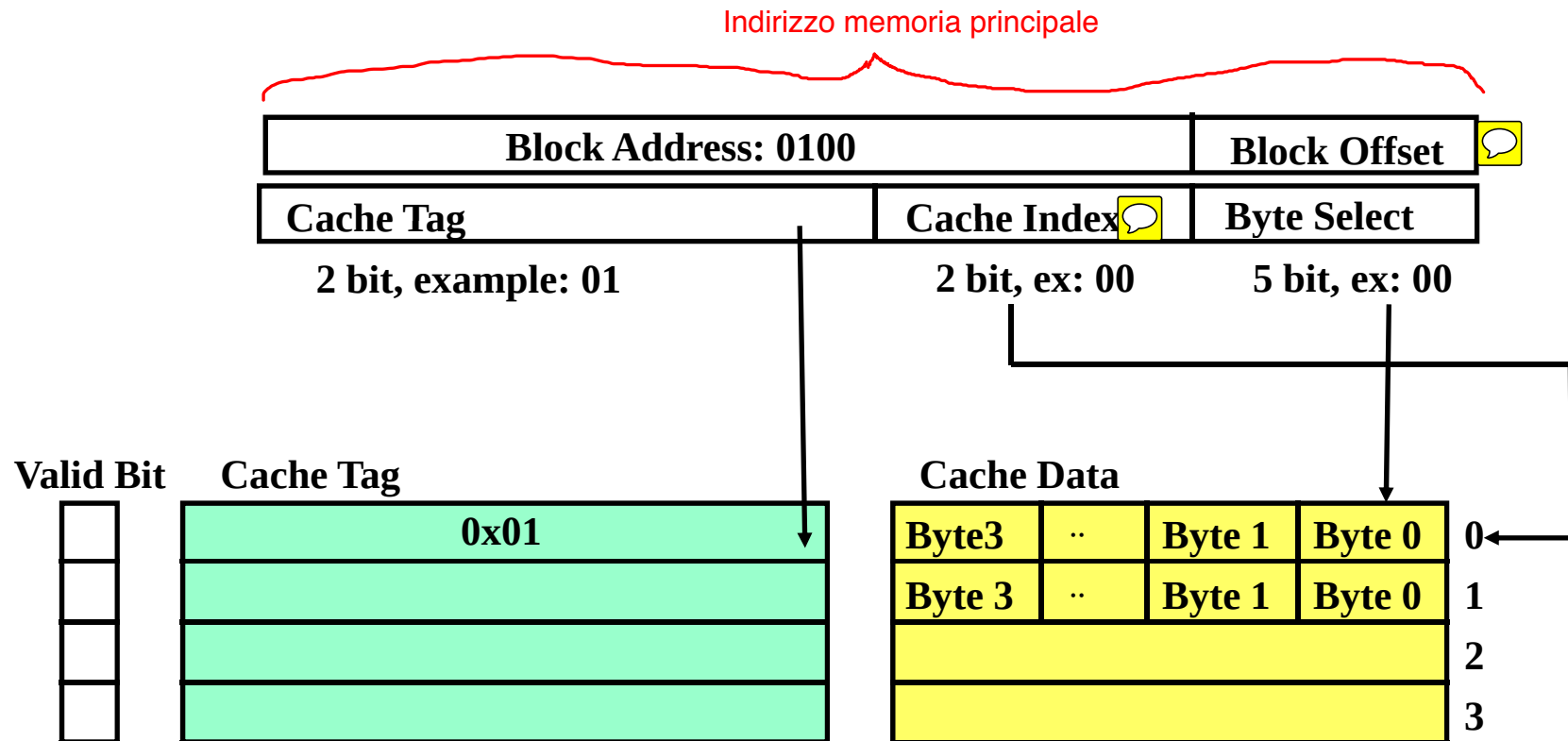
Una memoria cache ha la struttura di una normale memoria: il blocco al suo interno è direttamente accessibile tramite indirizzamento

Esempio: Cache Direct Mapped di 4 parole e memoria di 16 parole.

La posizione 00 della cache può essere occupata da dati provenienti dagli indirizzi di memoria --00



Cache a indirizzamento diretto di 4 parole e memoria di 16 parole



Indirizzamento nelle cache a indirizzamento diretto

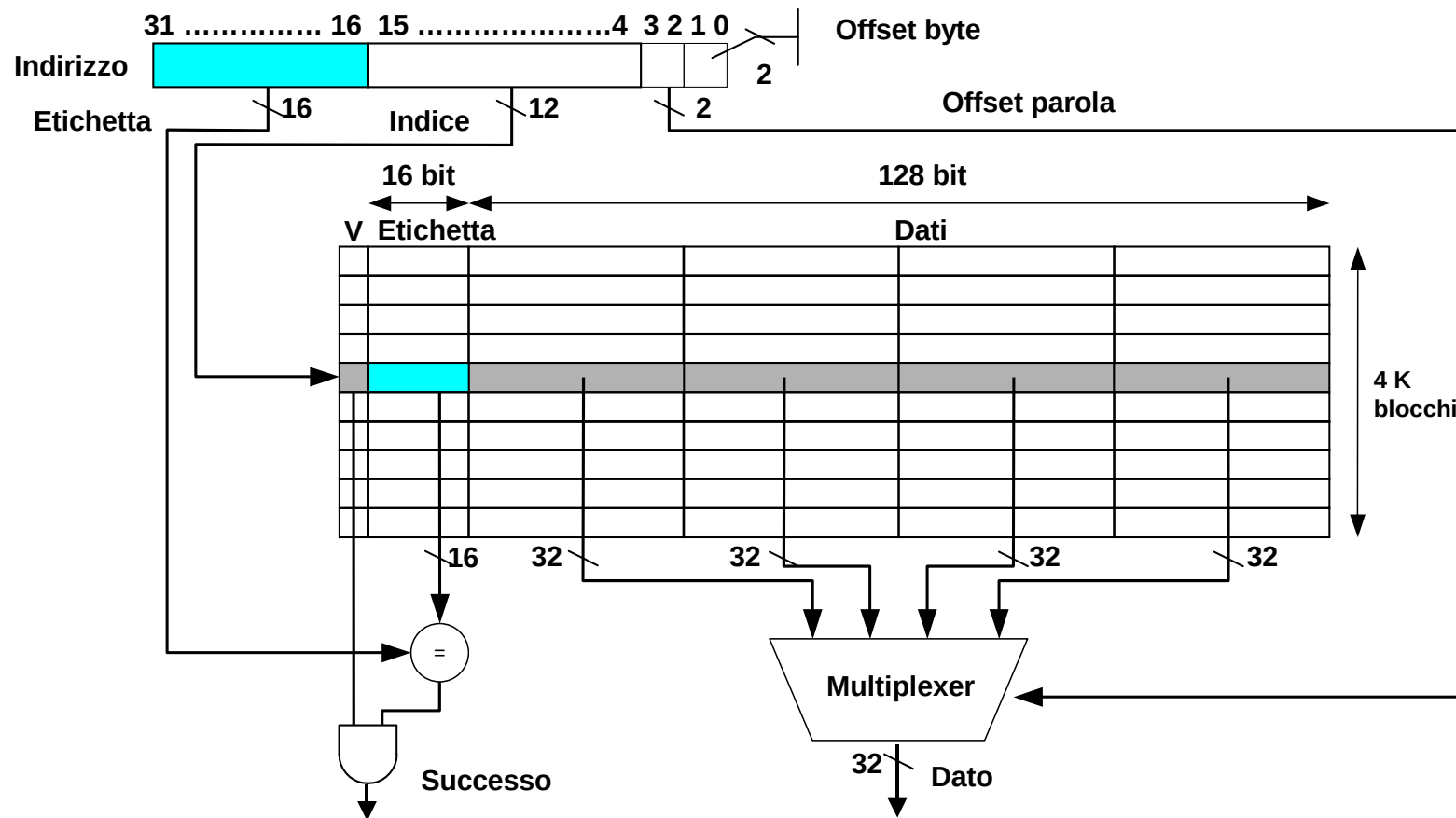
Etichetta	Indice	Sp. parola	Sp. byte
-----------	--------	------------	----------

- Indirizzo di memoria di **N** bit diviso in 4 campi:
 - Campo ***spiazzamento (offset) del byte*** nella parola individua il byte desiderato all'interno della parola: **B** bit
 - Se la parola non è indirizzabile al byte $B = 0$.
 - Campo ***spiazzamento (offset) della parola*** nel blocco individua la parola desiderata all'interno del blocco: **K** bit.
 - Se il blocco contiene un sola parola $K = 0$.
 - Campo ***indice (index)*** serve a identificare il blocco: **M** bit
 - Campo ***etichetta (tag)***, da confrontare con il contenuto del campo etichetta della cache, viene utilizzata per controllare il blocco selezionato dall'indice: **$N - (M + K + B)$** bit

Esempio:

- Indirizzo di memoria: **N = 32 bit**
- Cache a indirizzamento diretto da **64 K Byte** e blocco da **128 bit** (4 parole da 32 bit = 16 Byte)
- Numero di blocchi = Dim. Cache / Dim. Blocco =
64 K Byte / 16 Byte = **4 K blocchi**
- Struttura dell'indirizzo di memoria:
 - ***Spiazzamento del byte*: B = 2 bit**
 - ***Spiazzamento della parola*: K = 2 bit**
 - ***Indice*: M = 12 bit** 
 - ***Etichetta* : 16 bit** 

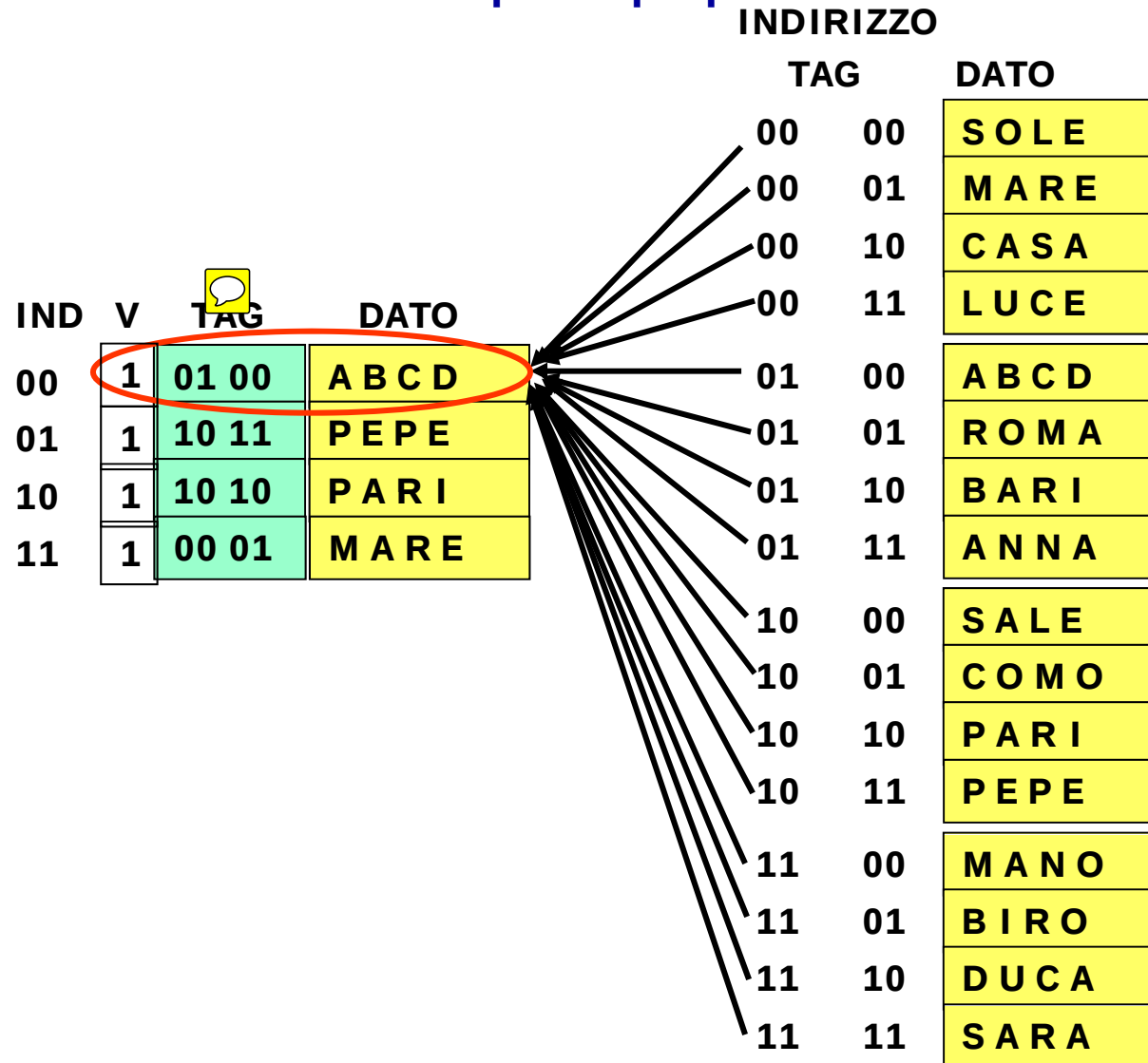
Cache a indirizzamento diretto da 64KByte e blocco da 128 bit



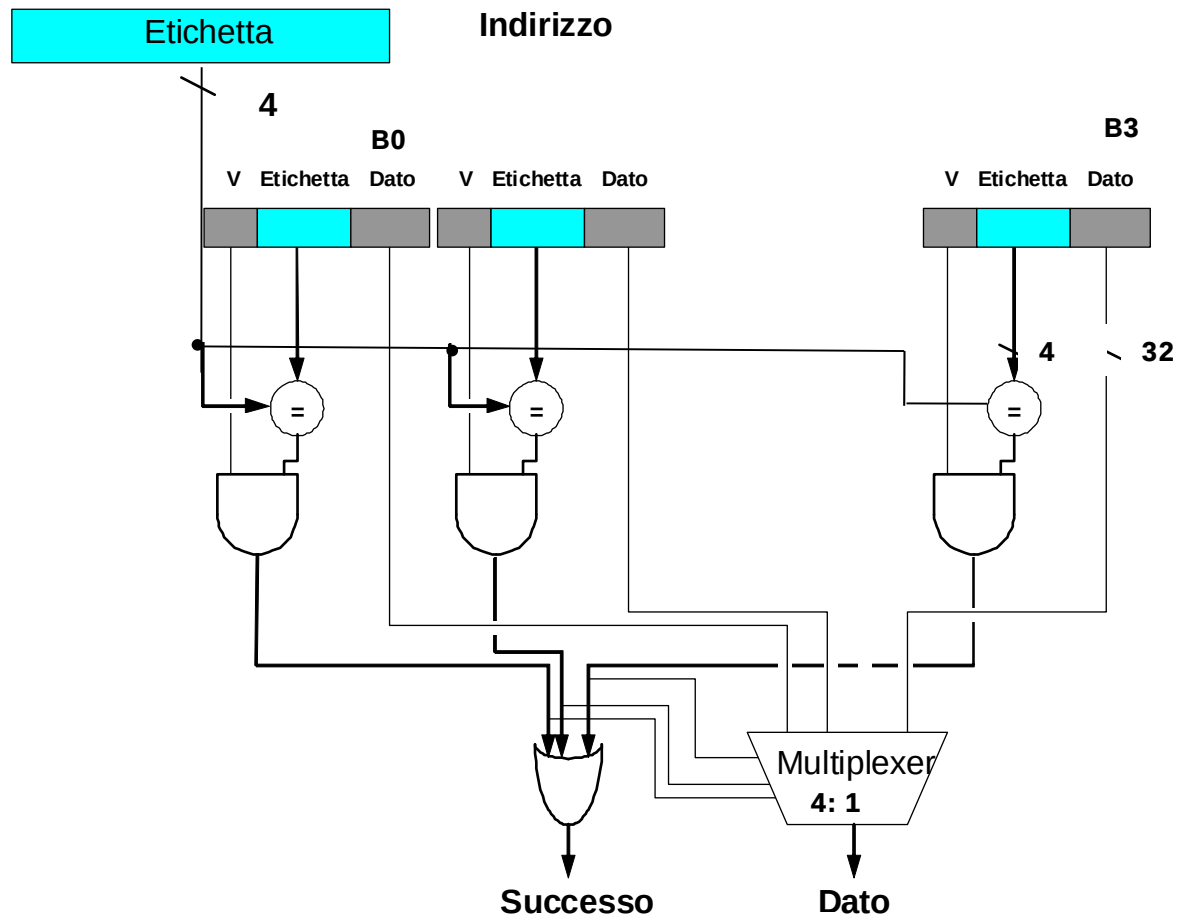
Cache completamente associativa

- In una cache *completamente associativa* (*fully associative*) un blocco di memoria può essere messo in una qualsiasi posizione della memoria cache.
- Poiché il blocco può essere messo in un posto qualsiasi della cache $\Rightarrow$ tutti i blocchi della cache debbono essere esaminati durante la ricerca.
- Nell'indirizzo non esiste il campo indice
- Numero di blocchi = $\text{Dim. Cache} / \text{Dim. Blocco}$

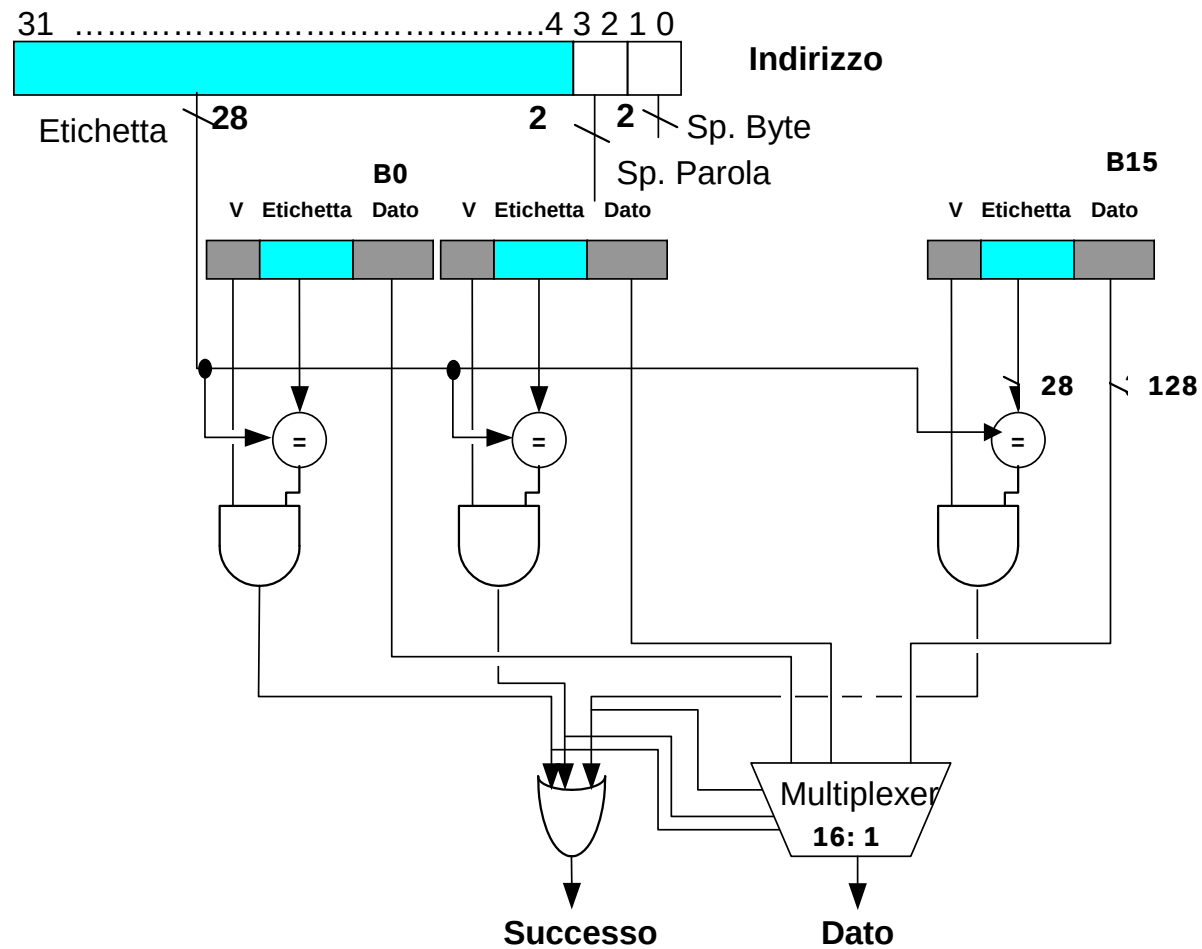
Esempio: Cache Fully Associative di 4 parole e memoria di 16 parole. La posizione 00 della cache può essere occupata da dati provenienti da una qualunque posizione di memoria



Cache compl. associativa di 4 parole e memoria di 16 parole



Cache compl. associativa da 256 Byte e blocco da 128 bit



Cache set-associativa a n vie

- Suddivisa in **n vie parallele** (Way)

Numero di blocchi = Dim. Cache / Dim. Blocco

Numero di blocchi in una via = Num. Blocchi / n

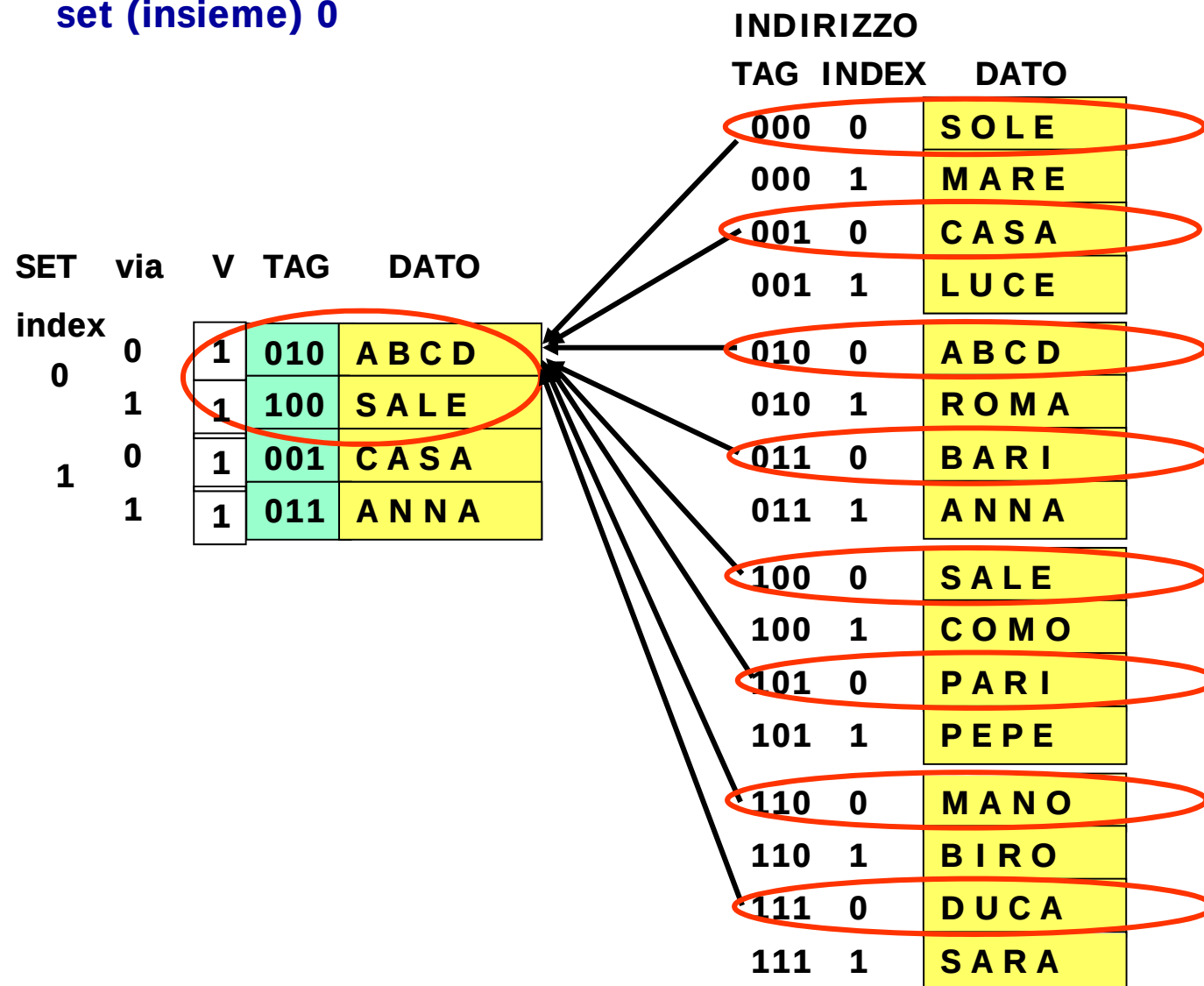
- Il **blocco** può essere **messo in una qualsiasi delle n vie**, ma **sempre nello stesso indirizzo** $\Rightarrow$ la **ricerca nelle vie** deve essere effettuata **contemporaneamente**, accedendo contemporaneamente allo stesso indirizzo in ogni via. Gli n blocchi letti formano un **set (insieme)**

Numero di insiemi = Dim. Cache / (Dim. Blocco $\times$ n)

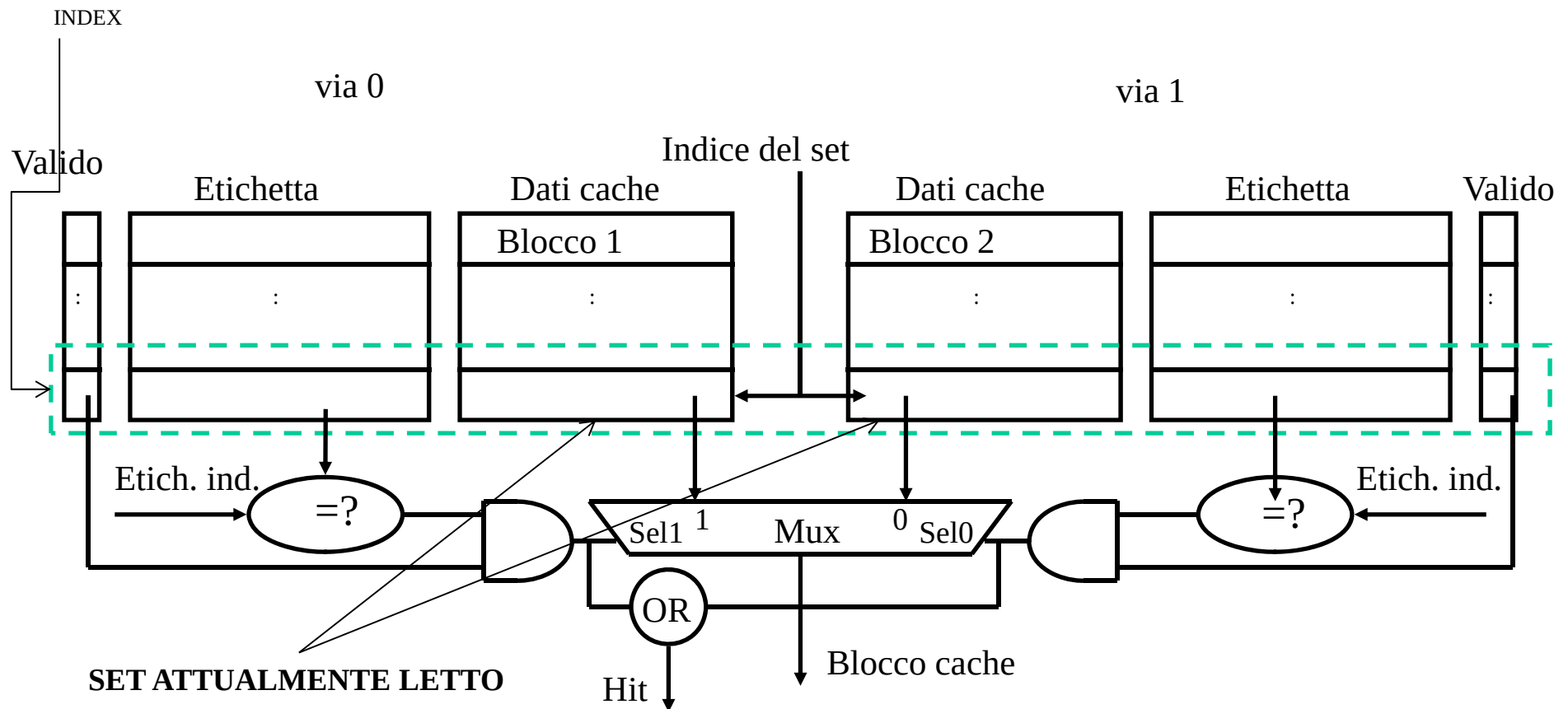
- Ogni via ha la struttura di una normale memoria indirizzabile in modo diretto, l'**associatività sta nel fatto che le vie possono essere usate alternativamente senza vincolo** (ogni set è associativo al proprio interno).
- Ogni blocco della memoria corrisponde ad un unico *posto* in ogni via (è l'indirizzo del suo set nella cache):

(indir in ogni via)_{cache} = (ind. blocco)_{mem} modulo (num. blocchi in via)

Esempio: Cache Set Associative a 2-vie di 4 parole e memoria di 16 parole. Il set 0 della cache può essere occupato da dati provenienti dagli indirizzi di memoria ---0. Il blocco può essere messo in uno qualsiasi dei 2 elementi del set (insieme) 0



Esempio: cache set-associativa a 2 vie



Entrambe le vie sono lette contemporaneamente, usando l'index come indirizzo.
La coincidenza tra etichetta (generata dalla CPU) ed etichetta memorizzata
in uno dei due blocchi del set garantisce lo hit.

Indirizzamento nelle cache set associative a n vie

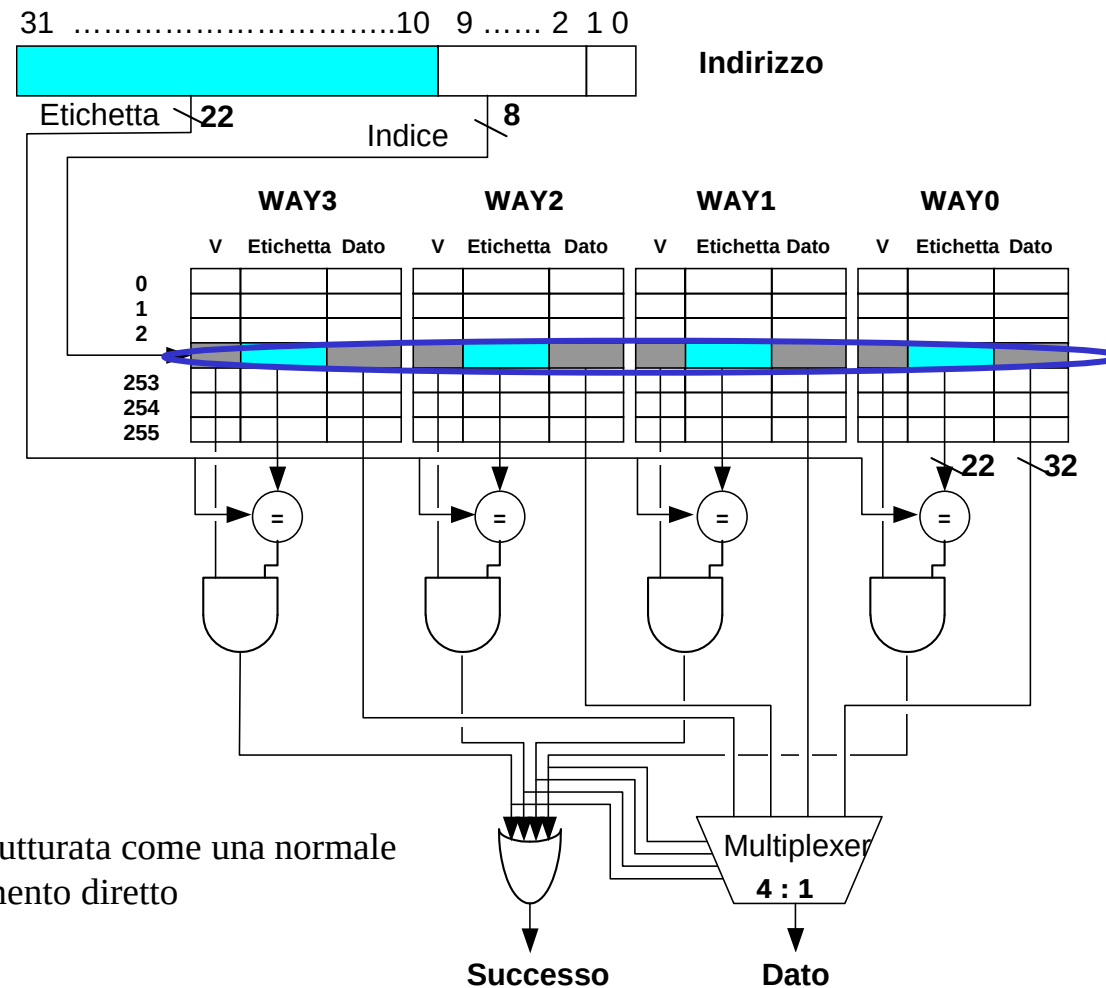
Etichetta	Indice	Sp. parola	Sp. byte
-----------	--------	------------	----------

- Indirizzo di memoria di N bit diviso in 4 campi:
 - Campo ***spiazzamento (offset) del byte*** nella parola individua il byte desiderato all'interno della parola: B bit
 - Campo ***spiazzamento (offset) della parola*** nel blocco individua la parola desiderata all'interno del blocco: K bit
 - Campo ***indice (index)*** serve a identificare la posizione unica e fissa corrispondente a ogni insieme: M bit (indirizzamento diretto dentro le vie, in parallelo)
 - Campo ***etichetta (tag): $N - (M + K + B)$ bit*** (individuazione del blocco associativamente dentro l'insieme –set- indirizzato)

Esempio:

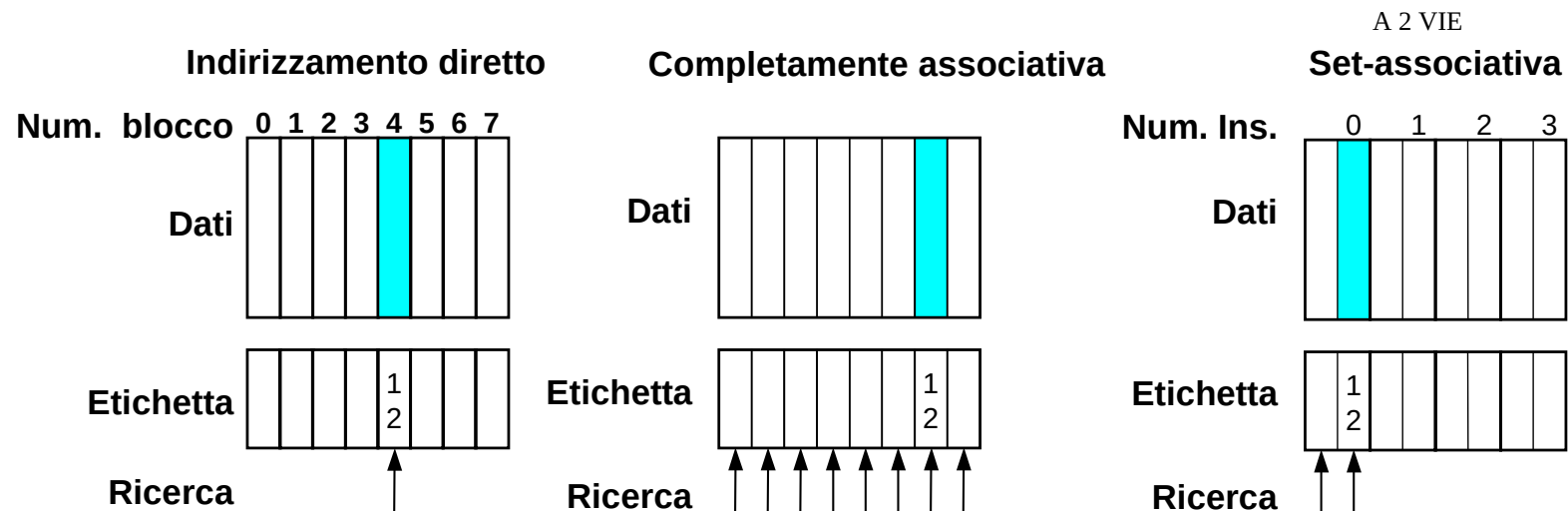
- Indirizzo di memoria: **N = 32 bit**
- Cache set associativa a **n = 4 vie** da **4KByte** e blocco da **32 bit** (1 parola = 4 Byte)
- Numero di blocchi = Dim. Cache / Dim. Blocco =
4 K Byte / 4 Byte = **1 K blocchi**
- Numero di insiemi: Dim. Cache / (Dim. Blocco x n) = 4 K Byte / (4 Byte x 4) = **2⁸ insiemi**
- Struttura dell'indirizzo di memoria:
 - ***Spiazzamento del byte*: B = 2 bit**
 - ***Spiazzamento della parola*: K = 0 bit**
 - ***Indice*: M = 8 bit**
 - ***Etichetta* : 22 bit**

Cache set-associativa a 4 vie da 4 KB e blocco da 32 bit



Ogni via (WAY) è strutturata come una normale memoria a indirizzamento diretto

Sommario: Identificazione del blocco



- Indirizzamento diretto:
 - Calcolo posizione, verifica etichetta e verifica bit valido
- Indirizzamento completamente associativo:
 - Confronta etichetta in ogni blocco e verifica bit valido
- Indirizzamento set-associativo
 - Identifica insieme, confronta etichette dell'insieme e verifica bit valido

Incremento dell'associatività

- Principale vantaggio:
 - Riduzione del miss rate
- Principali svantaggi:
 - Maggiore costo implementativo
 - Incremento di hit time.
- La scelta tra cache a indirizzamento diretto, completamente associative e set associative dipende dal costo di implementazione dell'associatività (sia in tempo sia in hardware addizionale) rispetto alla riduzione del miss rate.

Problema della sostituzione di un blocco

- In una cache a **indirizzamento diretto** c'è un solo candidato alla sostituzione.
- Quando si verifica un fallimento in una cache **completamente associativa**, bisogna decidere quale blocco sostituire: ogni blocco è un potenziale candidato per la sostituzione.
- Se la cache è **set-associativa**, bisogna scegliere tra i blocchi compresi nell'insieme.
- Principali strategie utilizzate per la scelta del blocco da sostituire nelle cache associative:
 - **Casuale**
 - **LRU (Least Recently Used)** si sostituisce il blocco usato meno recentemente
 - **FIFO (First In First Out)**

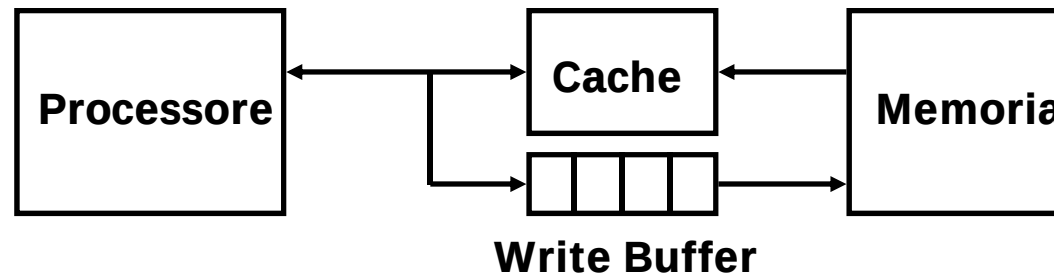
Problema della strategia di scrittura

- Possibili strategie per la gestione delle scritture:
 - **Write-through:** l'informazione viene scritta nel blocco della cache *e* nel blocco della memoria principale.
 - **Write-back:** l'informazione viene scritta solo nel blocco della cache. Il blocco modificato è scritto nella memoria principale *solo* quando viene sostituito a causa di un miss.
 - Al termine della scrittura nella cache il blocco di cache diventa **dirty (modified)** e la memoria principale conterrà un valore diverso da quello presente nella cache: **la memoria principale e la cache non sono coerenti (consistenti).**

Problema della strategia di scrittura

- **Vantaggi write-back:**
 - I singoli blocchi possono essere scritti dal processore alla frequenza a cui la cache, e non la memoria principale, è in grado di accettarle.
 - Scritture multiple all'interno dello stesso blocco richiedono una sola scrittura nella memoria principale.
- **Vantaggi write-through:**
 - Più semplice da realizzare, anche se per essere efficace in un sistema veloce, una cache write-through deve essere dotata di un **buffer di scrittura (write buffer)** in modo da non dover attendere il livello inferiore di memoria.
 - I fallimenti in lettura sono meno costosi, infatti non richiedono mai la scrittura nel livello inferiore.

Buffer di scrittura (Write buffer)



Hit e Miss in lettura e scrittura

- **Hit in lettura**

- il dato è presente in cache

- **Miss in lettura**

- *stallo* della CPU, richiesta del blocco contenente il dato cercato alla memoria, copia in cache (scrittura in cache), ripetizione dell'operazione di lettura in cache

- **Hit in scrittura:**

- Scrittura del dato sia in cache sia in memoria (**write-through**)
- Scrittura del dato solo nella cache (**write-back**): la copia in memoria avviene in un secondo momento quando il blocco viene sostituito

- **Miss in scrittura:**

- *stallo* della CPU, richiesta del blocco contenente il dato cercato alla memoria, copia in cache (scrittura in cache), ripetizione dell'operazione di scrittura

Progetto di una memoria cache

- Metodo di **indirizzamento**
 - come scegliere il blocco della cache in cui copiare un blocco di memoria centrale (*mapping o piazzamento del blocco*)
- Metodo di **identificazione**
 - come localizzare un blocco di memoria centrale all'interno della cache
- Metodo di **scrittura** (write back o write through)
 - come comportarsi quando si deve scrivere una parola contenuta in un blocco della memoria cache
- Metodo di **sostituzione**
 - come si sostituiscono blocchi della cache per liberare spazio nel caso di miss

Conclusioni

- Analizzati i vantaggi legati all'introduzione di una gerarchia di memoria in un calcolatore.
- Analizzati i concetti base e le architetture delle memorie cache.