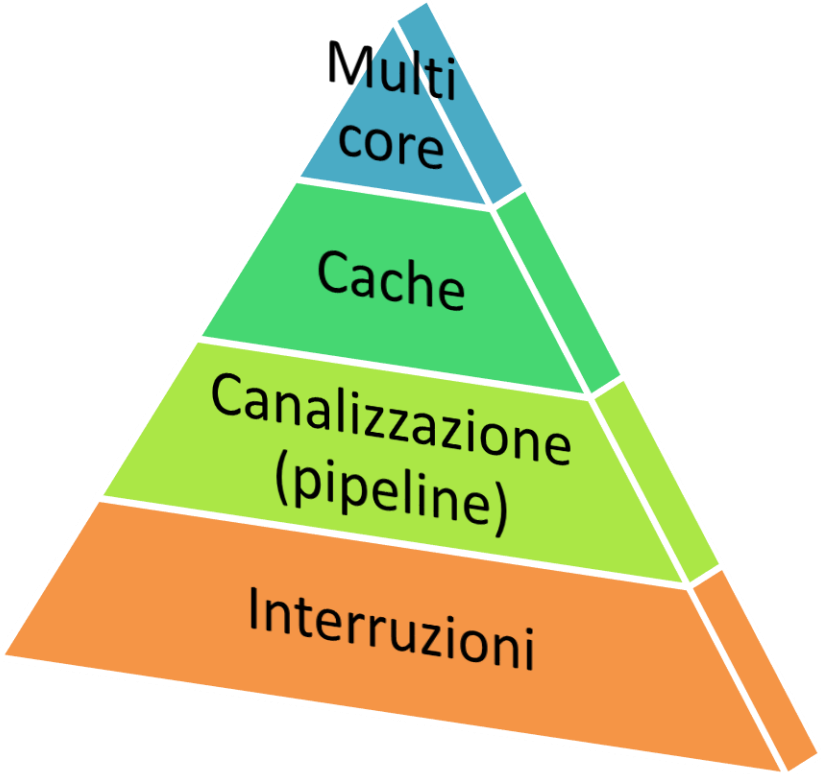
Architettura
degli Elaboratori Elettronici

Dott. Franco Liberati
liberati@di.uniroma1.it

ARGOMENTI DELLA LEZIONE

- ❑ Interruzioni
- ❑ Canalizzazione (pipeline)
- ❑ Cache
- ❑ Multi core (*cenni*)



Memoria cache

MEMORIA CENTRALE

Limiti

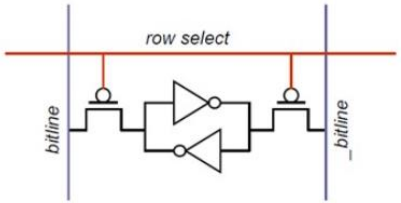
- Le prestazioni all'interno di un elaboratore elettronico sono limitate dai **tempi di accesso ai dati**
- **La memoria RAM ha tempi di propagazione del flusso informativo** (scrittura e lettura dati) **più lenti rispetto** alle altre unità funzionali (es.: unità di calcolo) e alle componenti (es.: i registri) **di una CPU**
  - tanto più se si considerano elaboratori canalizzati e se il riferimento è quindi alla singola fase piuttosto che all'intero colpo di clock

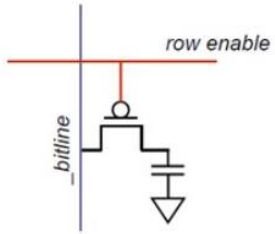
MEMORIA CENTRALE

Tipologie: memoria SRAM e DRAM

- La **Memoria Centrale** è realizzata utilizzando la tecnologia **DRAM** (*Dynamic Random Access Memory*, ovvero memoria dinamica ad accesso casuale)
- Ma esistono anche memorie **più veloci**: le **SRAM** (*Static Random Access Memory*)
- Il costo per bit delle DRAM è più basso rispetto a quello delle SRAM
  - La differenza di prezzo dipende dal fatto che le memorie DRAM utilizzano meno transistori per ogni bit da memorizzare: consentono quindi di raggiungere capacità superiori a parità di area di silicio
- Sono state sviluppate **SDRAM** che sono sincronizzate con il clock della macchina
- In seguito sono state adottate le **DDR** che consentono di operare non solo quando il segnale del clock è alto, ma anche quando è basso raddoppiando la velocità di trasferimento

SRAM	DRAM
Accessi più rapidi	Accesso più lenti
Costi più alti	Costi più bassi
Maggiori consumi di corrente (refresh frequenti dei dati memorizzati)	Minori consumi di correnti (i condensatori immagazzinano l'informazione per più tempo)
Circuiteria complessa	Circuiteria semplice
Bassa densità delle celle di memoria sul chip di memoria	Alta densità delle celle di memoria sul chip di memoria

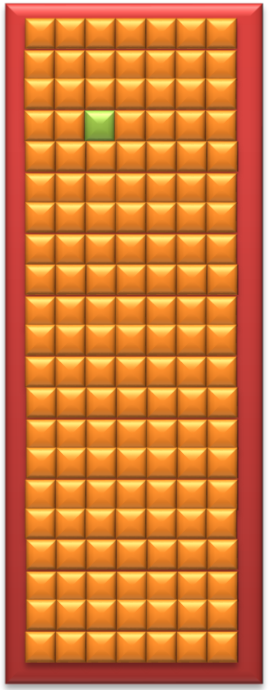




MEMORIA CENTRALE

Organizzazione e struttura della memoria principale

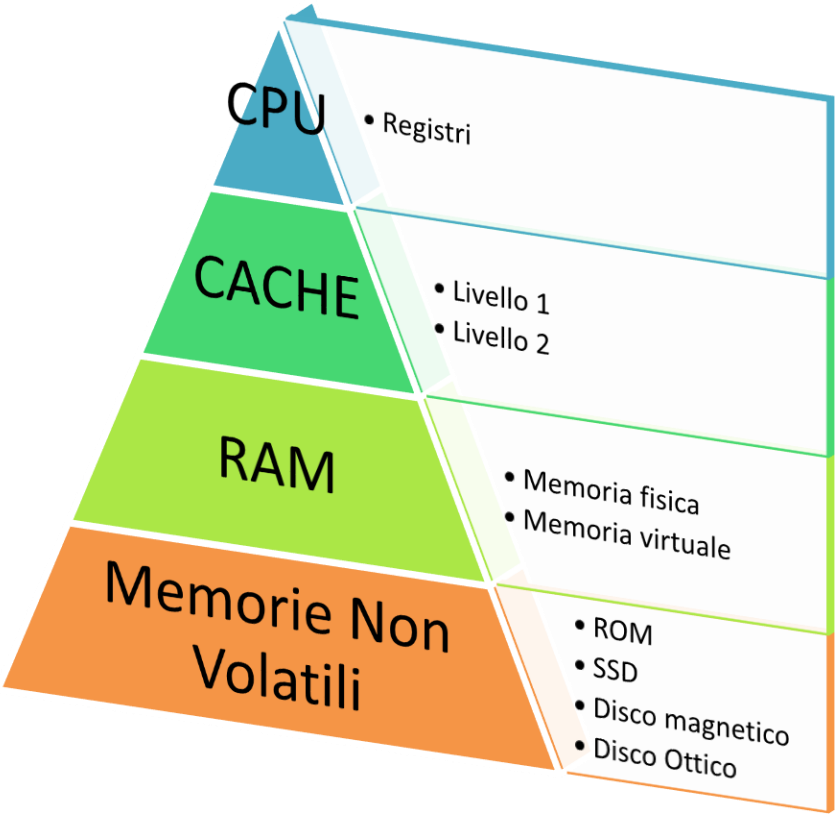
- La **memoria RAM** è divisa in **blocchi** di dimensioni uguali
- Attualmente la dimensione dei blocchi è di 8KB (1KB=1024byte); ma i Sistemi Operativi moderni consento di personalizzare questa dimensione in 4KB,8KB,16KB e 32KB
- Più blocchi sono stipati in un **banco**



MEMORIA CACHE

Generalità gerarchia di livelli di memoria

- A causa delle differenze di costo e della velocità, si definisce una **gerarchia di livelli di memoria**, con una memoria più veloce (e quindi più costosa) posta vicino al processore e una più lenta (e meno costosa) sita a maggiore distanza
- Nella gerarchia di solito si comprende anche le **memorie non volatili** cioè quei dispositivi che consento l'archiviazione permanente dei dati (es.: disco e nastro magnetico, supporti ottici, MSS) che sono memorie estremamente lente (a causa dei componenti elettromeccanici utilizzati per archiviare e reperire i dati)



MEMORIA CACHE

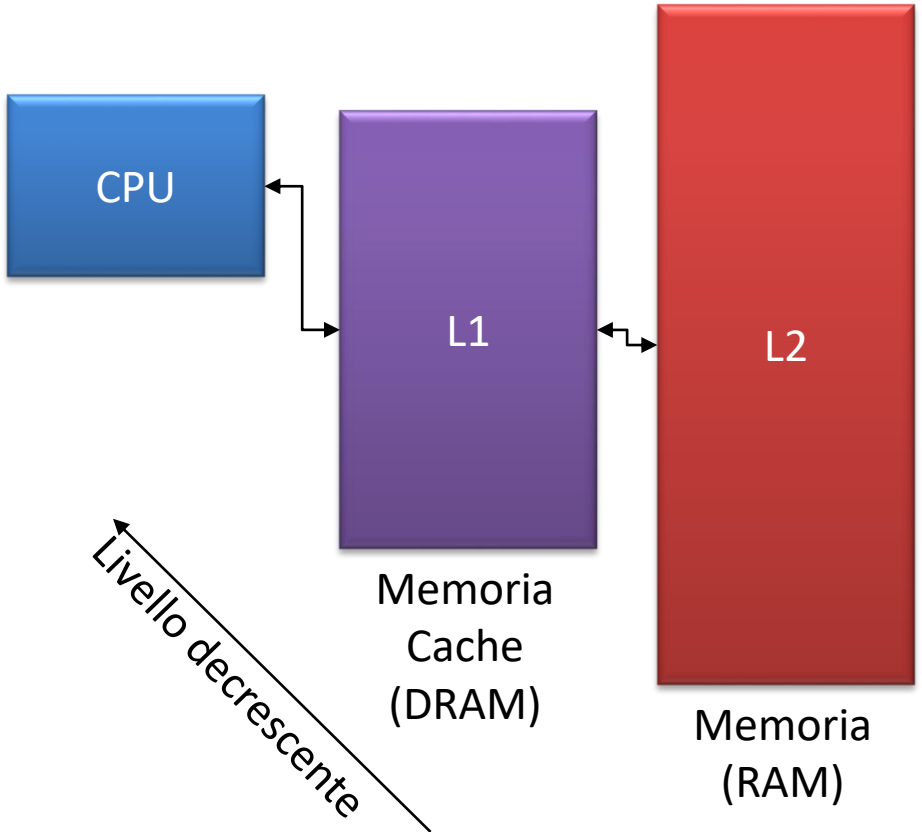
Principio di località

- L'introduzione della memoria cache si basa sul **principio di località** (verificato analizzando l'esecuzione di un gran numero di programmi):
  - **Località temporale**: quando si fa riferimento a un elemento di memoria, c'è la tendenza a far riferimento allo stesso elemento entro breve (es.: il riutilizzo di istruzioni e dati contenuti nei cicli)
  - **Località spaziale**: quando si accede a un elemento di memoria, c'è la tendenza a far riferimento entro breve tempo ad altri elementi che hanno indirizzo vicino (es.: eseguita un'istruzione si tende ad elaborare quella immediatamente successiva; quando si accede a dati organizzati in vettori o matrici, l'accesso a un dato è seguito dall'accesso al dato immediatamente successivo come avviene nella stampa di un vettore o nella scansione dei suoi elementi)

MEMORIA CACHE

Gerarchia di memoria non volatile

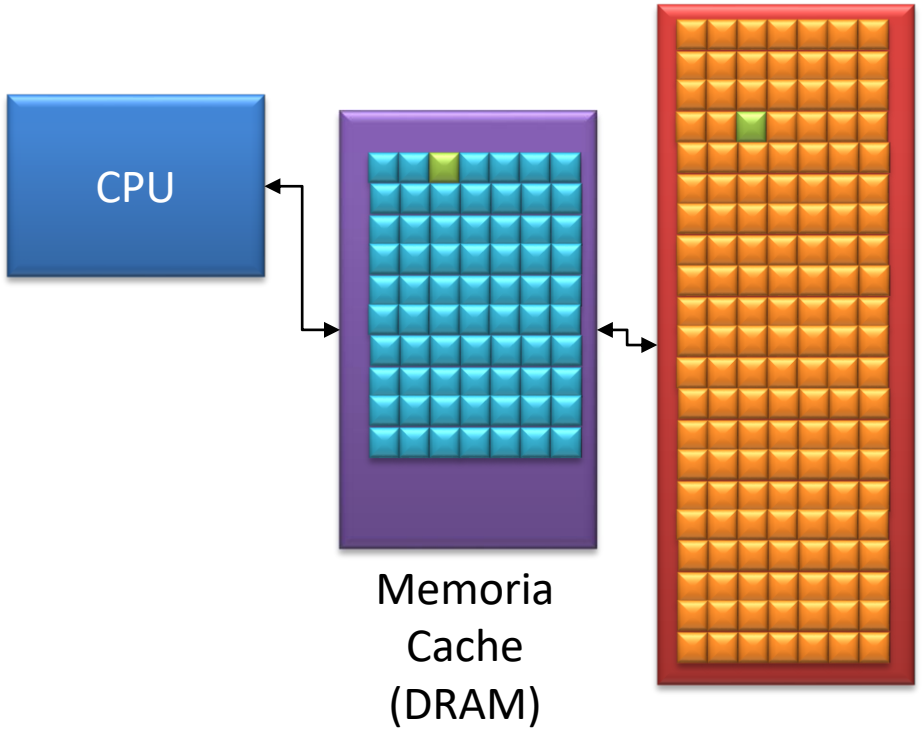
- In relazione al principio di località, la memoria di un calcolatore è realizzata come una **gerarchia di memoria**, che consiste in un insieme di livelli di memoria, ciascuno di diversa velocità e dimensione
- A parità di capacità, le memorie più veloci sono più costose di quelle più lente, perciò sono in genere più piccole: al livello più alto (cioè vicino alla CPU) ci sono memorie più piccole e veloci, a quello più basso memorie più lente e meno costose



MEMORIA CACHE

Gerarchia di memoria obiettivo e trasferimenti

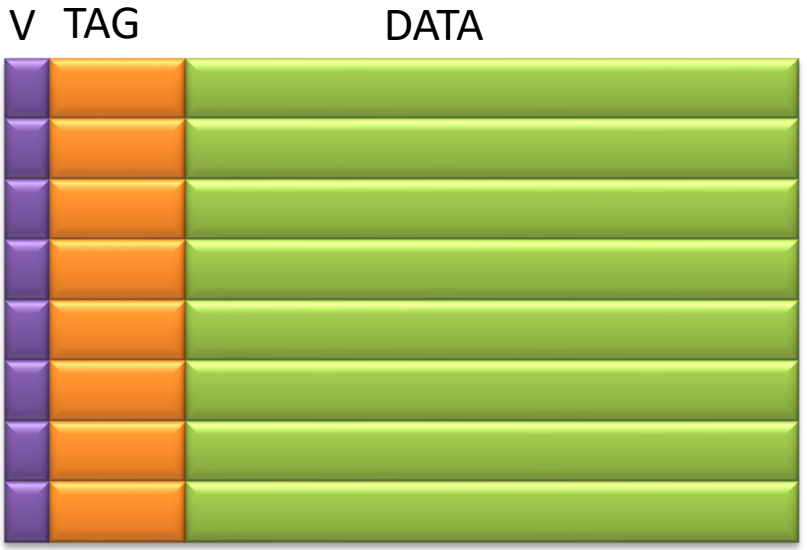
- L'**obiettivo della gerarchia di memoria** è quello di dare al programmatore l'illusione di poter usufruire di una **memoria** al tempo stesso **veloce** (idealmente, quanto la memoria al livello più basso) **e grande** (quanto quella al livello più alto)
- A causa della differente dimensione diventa necessario, durante l'esecuzione dei programmi, **trasferire informazione fra memorie di livelli diversi**
- Anche se una gerarchia di memoria è in genere composta da più livelli, le **informazioni** sono di volta in volta **copiate solo tra due livelli adiacenti**



MEMORIA CACHE

Organizzazione e struttura della memoria cache

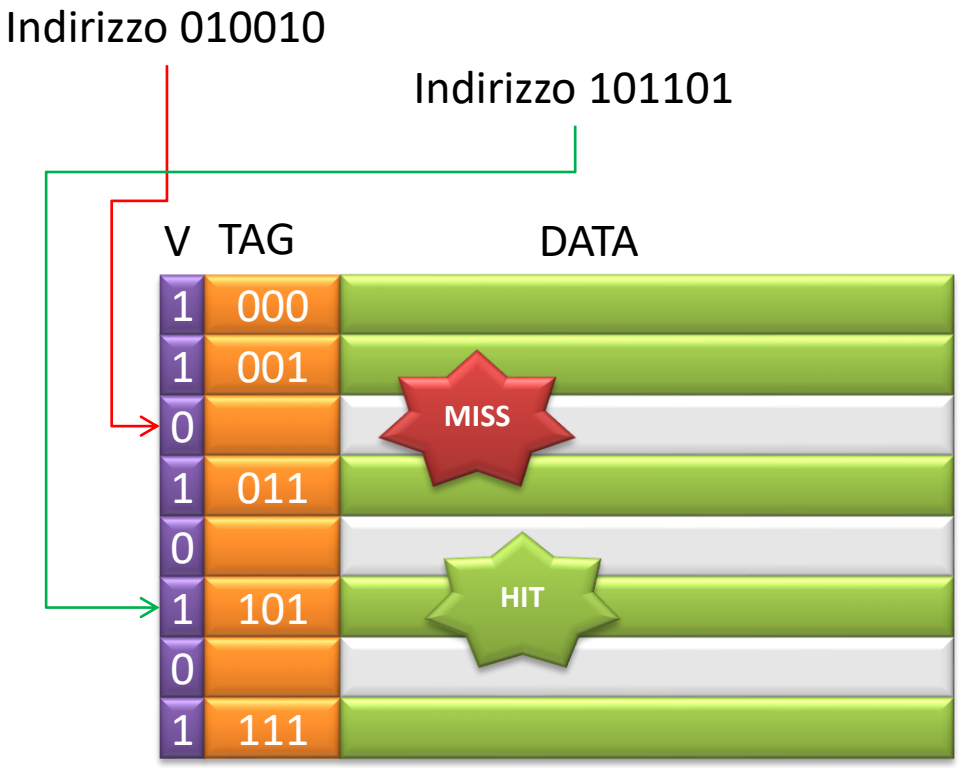
- La memoria cache è organizzata fisicamente in **linee** (***cache lines***)
- La **cache** contiene un certo numero di **linee** composte da:
  - 1 bit di **VALIDITÁ** che indica se la linea contiene dei dati
  - Il campo **TAG** (o **Etichetta**) che distingue quale parola (o blocco) della Memoria Centrale è nella linea
  - Il campo **DATA (parola** o **blocco)** cioè la parola (o il blocco) memorizzato nella linea
  - **Altri bit utili**: Dirty, Used, Writable



MEMORIA CACHE

Accesso ai dati

- Quando la CPU **richiede un indirizzo**:
  - Se il dato richiesto dalla CPU fa parte di uno dei blocchi presenti nella cache di livello inferiore, si dice che la richiesta ha avuto **successo** (**HIT**)
  - Se invece il dato non si trova nel livello inferiore, si dice che la richiesta **fallisce** (**MISS**). In questo caso per trovare il blocco che contiene i dati richiesti, bisogna accedere al livello superiore della gerarchia e trasferire il blocco corrispondente al livello inferiore



MEMORIA CACHE

Prestazioni temporali

- Programma con 1000000 di accessi in memoria RAM e il tempo di accesso ad ogni istruzione in memoria RAM di 100ns
- Il tempo totale per l'accesso: 1000000·100ns = 100ms
- Se si usa una cache con tempo di accesso 1ns e la percentuale di miss è il 10%:
  - il 90% di 1000000 accessi (**HIT**) impiegano 1ns·1000000·90% = **0.9ms**
  - il 10% rimanente (**MISS**) impiega 100ns·1000000·10%=**10ms**
- In totale il **tempo di accesso medio in cache** è:

10ms + 0.9ms/1000000 = 10.9 ns

Con un aumento di velocità di 100ns/10.9ns; cioè circa nove volte

(N.B.: *si esclude da questo calcolo il tempo necessario per scrivere un blocco di dati nella memoria cache*)

MEMORIA CACHE

Progettazione di una cache

- Il **progetto di una gerarchia di memoria** richiede la risoluzione di quatto problemi:

1. Dove mettere una parola (o un blocco) che è portato dal livello superiore al livello inferiore (**Problema del piazzamento di un blocco/parola**)
2. Dove si trova la parola (o il blocco) che contiene il dato richiesto (**Problema della ricerca di un blocco/parola**)
3. Quale parola (o blocco) presente al livello inferiore deve essere sostituito da uno del livello superiore nel caso di fallimento (**Problema della sostituzione di un blocco/parola**)
4. Cosa succede nel caso di scrittura sulla parola (o sul blocco) presente nella cache (**Problema della strategia di scrittura**)

MEMORIA CACHE

Piazzamento del blocco/parola

- La CPU può, a priori, tentare di accedere a una qualunque parola nello spazio totale di indirizzamento (cioè ad un qualsiasi indirizzo dell'intera memoria RAM)
- Occorre quindi definire le possibili modalità per consentire ad ogni parola nello spazio totale di indirizzamento di trovare (dove necessario) una posizione della cache in cui possa essere trasferita cioè, **occorre definire una corrispondenza tra l'indirizzo in Memoria Centrale della parola e il rispettivo indirizzo nella Memoria Cache**
- A questo scopo, sono state definite tre soluzioni diverse:
  1. **Cache a indirizzamento diretto (direct mapped cache)**
  2. **Cache set-associativa a n vie (n-way set associative cache)**
  3. **Cache completamente associativa (fully associative cache)**

MEMORIA CACHE

Piazzamento parola: dettaglio

- **Cache direct mapped**:
  - ogni parola (o blocco) è posizionata in una linea fissata
    - **PRO**:hardware più semplice e meno costoso è facile determinare in quale linea cercare il dato

      #linea = #parola % N (dove N è la dimensione della cache)
    - **CONTRO**: blocchi diversi mappano nella stessa linea, se gli accessi a questi blocchi si alternano si ottengono molti **MISS** e si crea il fenomeno del **trashing**
- **Cache completamente associativa**:
  - un blocco può essere messo in una linea qualsiasi
    - **PRO**: massima flessibilità
    - **CONTRO**: hardware più complesso e più costoso
- **Cache set-associativa a n-vie :** le linee sono suddivise in S **gruppi** (***set***) formati ciascuno da *n* **elementi**, o **vie**, e ogni parola (o blocco) è disposta in una qualsiasi delle linee del set fissato

  #set = #parola % S (dove S è la dimensione del set)

MEMORIA CACHE

Piazzamento parola: direct mapped

- In una cache **a indirizzamento diretto** ogni locazione di memoria corrisponde esattamente a una locazione della cache
- La corrispondenza tra indirizzo di Memoria Centrale e locazione nella Memoria Cache è data da:

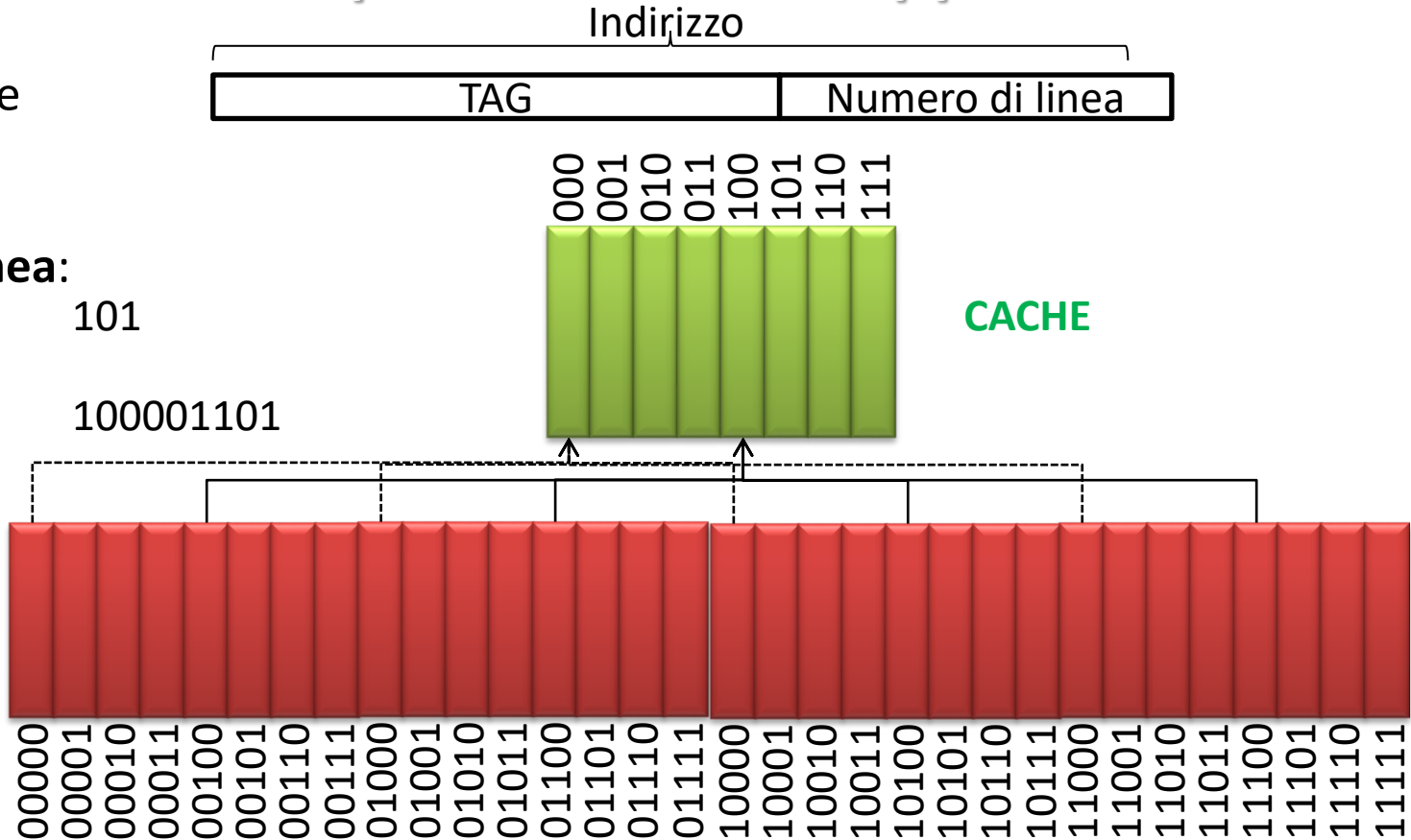
$$\textbf{(Ind. parola)}_{\textbf{cache}} = \textbf{(Ind. parola)}_{\textbf{mem}} \textbf{ modulo (numero di linee nella cache)}$$

- Il problema del piazzamento di un blocco è risolto in modo elementare: si esamina la configurazione degli ultimi *n* bit dell'indirizzo

MEMORIA CACHE

Piazzamento parola: direct mapped

- Esempio:
  - CACHE: 8 linee
- **Indirizzo**: ***2157*** (100001101101)
  - **Numero di linea**: 2157 % 8 = ***5*** 101
  - **Tag**: 2157/8 = ***269*** 100001101



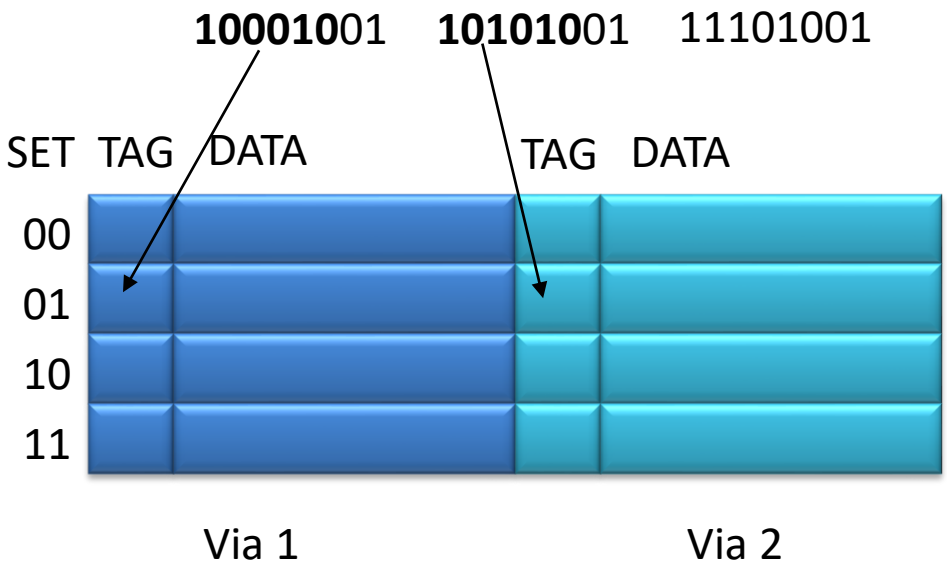
MEMORIA CACHE

Memoria cache set associativa a n-vie

- Una cache set-associativa a n vie è costituita da un **numero di insiemi** (set), ognuno dei quali comprende **n parole**
- In una cache set-associativa a n vie, ogni parola della memoria può essere trasferito in un numero prefissato n di posizioni all'interno di un insieme
- Si tratta di uno schema intermedio tra quello a indirizzamento diretto e quello completamente associativo
- In una cache set-associativa, l'insieme che contiene la parola è individuato da:

- **$(\text{Insieme})_{\text{cache}}$ = $(\text{Indirizzo parola})_{\text{mem}}$ modulo (numero di insiemi nella cache)**

- Poichè il numero degli insiemi nella cache una potenza di 2, l'operazione di modulo può essere effettuata considerando **$\log_2$(# insiemi nella cache) bit meno significativi**, che sono utilizzati come indice della cache. Il blocco può poi essere messo in un qualsiasi elemento dell'insieme: la ricerca deve quindi essere effettuata su tutti gli elementi dell'insieme

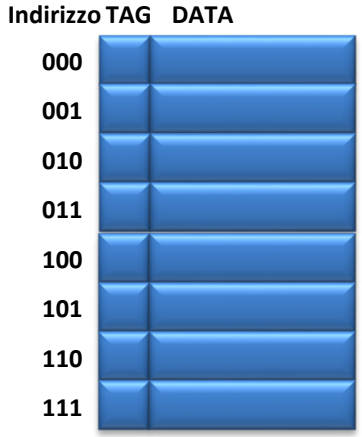


MEMORIA CACHE

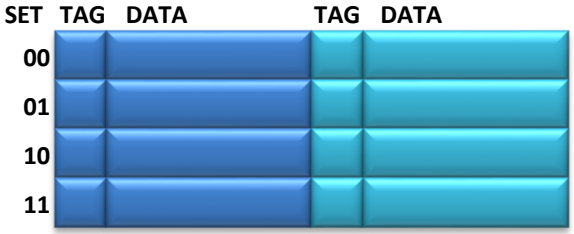
Memoria cache set associativa a n-vie: altri casi

- Definita la dimensione della cache, al crescere dell'associatività diminuisce il numero degli insiemi, mentre cresce il numero di elementi compresi in un insieme. A titolo di esempio, si considerino le possibili configurazioni di una cache composta da 8 blocchi

SET ASSOCIATIVA A 1 VIA
(accesso diretto)



SET ASSOCIATIVA A 2 VIE

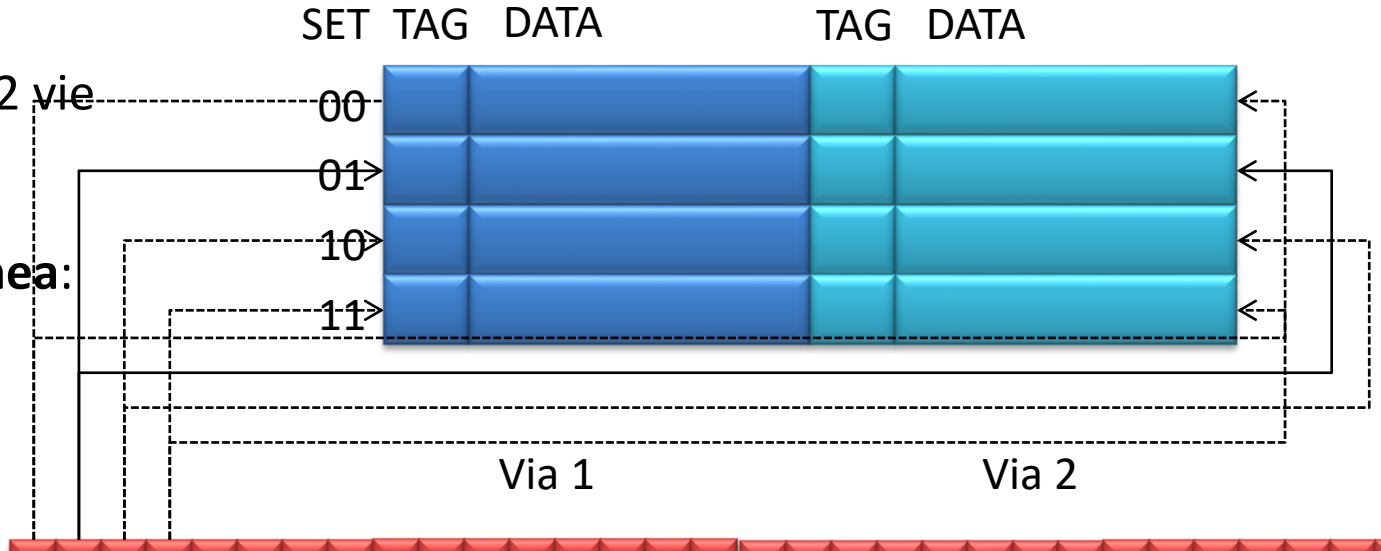


SET ASSOCIATIVA A 4 VIE

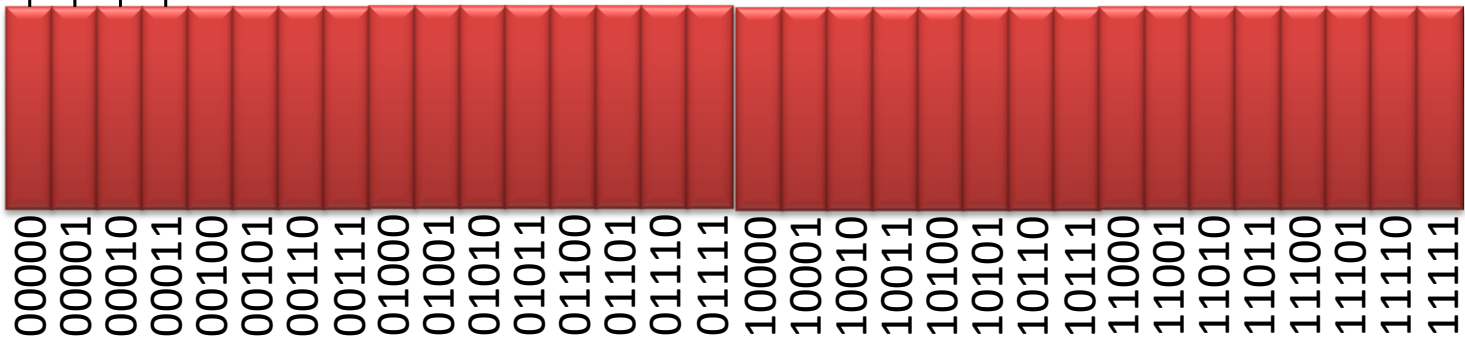
MEMORIA CACHE

Piazzamento parola: 4 insiemi e 2 vie

- Esempio:
  - CACHE: 4 set 2 vie
- **Indirizzo**: ***2157*** (100001101101)
  - **Numero di linea**: 2157 % 4 = ***1***
  - **Tag**: 2157/4 = ***539***



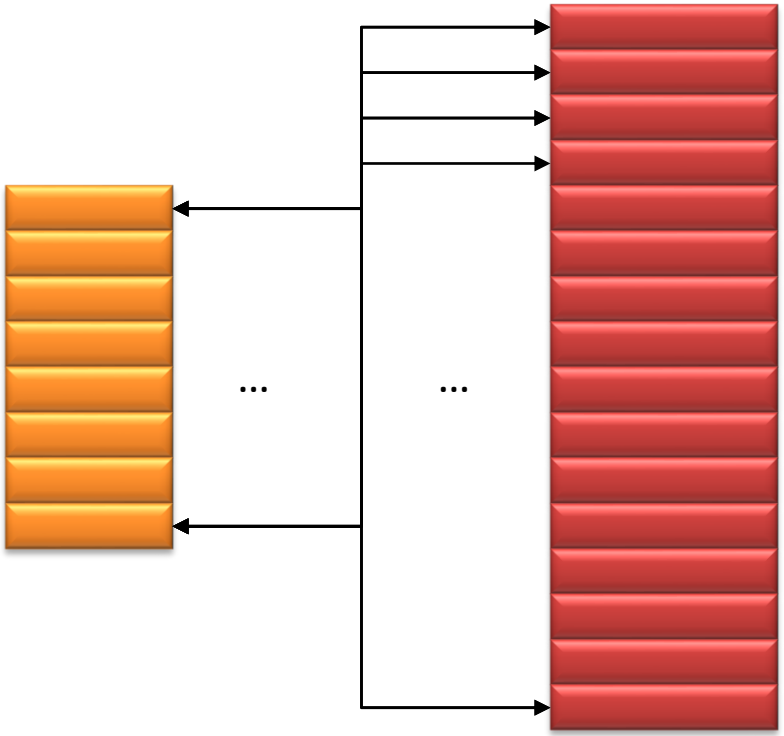
MEMORIA



MEMORIA CACHE

Cache completamente Associativa

- In una **cache completamente associativa** (***fully associative***) una parola nella Memoria Centrale può essere messo in una qualsiasi posizione della Memoria Cache
- La parola può essere messa in un posto qualsiasi della Cache quindi, al momento della ricerca, tutti i blocchi della Cache dovranno essere esaminati



MEMORIA CACHE

Ricerca di una parola in una cache

- Poiché ogni linea della cache può contenere più parole, per sapere se una parola di memoria si trova nella cache bisogna aggiungere nella linea di cache una **etichetta** (tag), che contiene informazioni necessarie per verificare se una delle parole presenti nella cache corrisponde, o meno, alla parola cercata
- Oltre all'etichetta è presente **un bit di validità (validity bit)** per indicare se la linea di cache è occupata da un dato
  - Il validity bit è ovviamente indipendente dalla particolare filosofia scelta per il piazzamento, ed è quindi presente in tutte le soluzioni

MEMORIA CACHE

Ricerca di una parola in una cache

- All’inizio una cache è vuota e le informazioni nelle etichette non hanno nessun significato
- Una volta riempita una linea della cache si aggiorna il relativo bit di validità, l’etichetta e si scrive una parola (o un blocco)

MEMORIA CACHE

Ricerca di una parola in una cache direct mapped

- Si consideri ora in maggior dettaglio come si svolgono alcune **sequenze di accesso in una cache mappata direttamente**, supponendo di partire dall'accensione della macchina, quando tutti **i bit di validità** sono negativi (N) e i contenuti delle etichette prive di significato
- La ricerca della parola con indirizzo di memoria 10**110** è fatta esaminando la linea con indice 110. Poiché il bit di validità è N allora c'è un MISS
- L'accesso è gestito trasferendo il dato dalla memoria e portando a valore positivo (S) la linea con indice 110 e l'etichetta diventa 10

Indice	Validità	Tag	Blocco
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		
111	N		

Indice	Validità	Tag	Blocco
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	S	10	*Parola corrispondente*
111	N		

MEMORIA CACHE

Aggiornamento dopo un MISS in direct mapped

- Quando c'è un MISS (es.: chiamando 10**110**) si attiva una interruzione che trasferisce la parola cercata dalla RAM alla cache, e aggiorna l'etichetta della linea (nell'esempio, la linea con indice **110**)
- Questo provoca uno stallo all'elaborazione del programma (non si può procedere avanti fino a quando non si ha il dato da elaborare)

Indice	Validità	Tag	Blocco
110	S	10	*Blocco o parola di memoria10110*

MEMORIA CACHE

Ricerca di una parola in una cache direct mapped

- Gestendo la richiesta del dato contenuto all'indirizzo 01**001** si popola la seconda linea
- Gestendo la richiesta del dato contenuto all'indirizzo 00**000** si popola la prima linea
- Gestendo la richiesta del dato contenuto all'indirizzo 10**111** si popola l'ultima linea

Indice	Validità	Tag	Blocco
000	S	00	*Blocco o parole di memoria 00000*
001	S	01	*Blocco o parole di memoria01001*
110	S	10	*Blocco o parole di memoria10110*
111	S	10	*Blocco o parole di memoria 10111*

MEMORIA CACHE

Ricerca di una parola in una cache direct mapped

- Si supponga ora di avere già "popolato" la cache con vari trasferimenti dalla memoria
- Se si cerca di leggere la parola di memoria 10**010** il bit di validità del blocco 010 è positivo e l'etichetta è uguale quindi si ha un **HIT**
- Se si cerca di leggere la parola di memoria 10**101** il bit di validità del blocco 101 è positivo, ma l'etichetta è diversa da quella voluta (01 invece di 10): si ha quindi un fallimento o **MISS**

Indice	Validità	Tag	Blocco
000	S	00	*Blocco o parole di memoria 00000*
001	S	01	*Blocco o parole di memoria01001*
010	S	10	*Blocco o parole di memoria 10010*
011	S	11	*Blocco o parole di memoria11011*
100	S	11	*Blocco o parole di memoria 11100*
101	S	01	*Blocco o parole di memoria01101*
110	S	10	*Blocco o parole di memoria10110*
111	S	10	*Blocco o parole di memoria 10111*

Indice	Validità	Tag	Blocco
000	S	00	*Blocco o parole di memoria00000*
001	S	01	*Blocco o parole di memoria01001*
010	S	10	*Blocco o parole di memoria 10010*
011	S	11	*Blocco o parole di memoria11011*
100	S	11	*Blocco o parole di memoria11100*
101	S	01	*Blocco o parole di memoria 01101*
110	S	10	*Blocco o parole di memoria10110*
111	S	10	*Blocco o parole di memoria10111*

MEMORIA CACHE

Aggiornamento dopo un MISS in direct mapped

- Quando c'è un MISS (es.: chiamando 111**01**) si attiva una interruzione che trasferisce la parola cercata dalla RAM alla cache, e aggiorna l'etichetta della linea
- Questo provoca uno stallo all'elaborazione del programma (non si può procedere avanti fino a quando non si hail dato da elaborare!)

Indice	Validità	Tag	Blocco
000	S	00	*Blocco o parole di memoria00000*
001	S	01	*Blocco o parole di memoria01001*
010	S	10	*Blocco o parole di memoria 10010*
011	S	11	*Blocco o parole di memoria11011*
100	S	11	*Blocco o parole di memoria11100*
101	S	01	*Blocco o parole di memoria 01101*
110	S	10	*Blocco o parole di memoria10110*
111	S	10	*Blocco o parole di memoria10111*

Indice	Validità	Tag	Blocco
000	S	00	*Blocco o parole di memoria00000*
001	S	01	*Blocco o parole di memoria01001*
010	S	10	*Blocco o parole di memoria 10010*
011	S	11	*Blocco o parole di memoria11011*
100	S	11	*Blocco o parole di memoria11100*
101	S	11	*Blocco o parole di memoria 11101*
110	S	10	*Blocco o parole di memoria10110*
111	S	10	*Blocco o parole di memoria10111*

MEMORIA CACHE

Schema cache:sfruttamento località spaziale

- Lo logica utilizza nella realizzazione della memorie cache descritta finora non sfrutta il principio di località spaziale degli accessi in quanto su una linea giace una sola parola
- Per trarre vantaggio dalla **località spaziale** è necessario che il campo data sia maggiore della dimensione della parola di memoria, in modo che contenga più di una sola parola
- **Quando si verifica un fallimento, dalla memoria centrale, pertanto, sono prelevate più parole adiacenti perché hanno una elevata probabilità di essere richieste a breve**
- È necessario quindi un campo aggiuntivo dell'indirizzo che rappresenti lo **spiazzamento** (offset) della parola nel blocco.

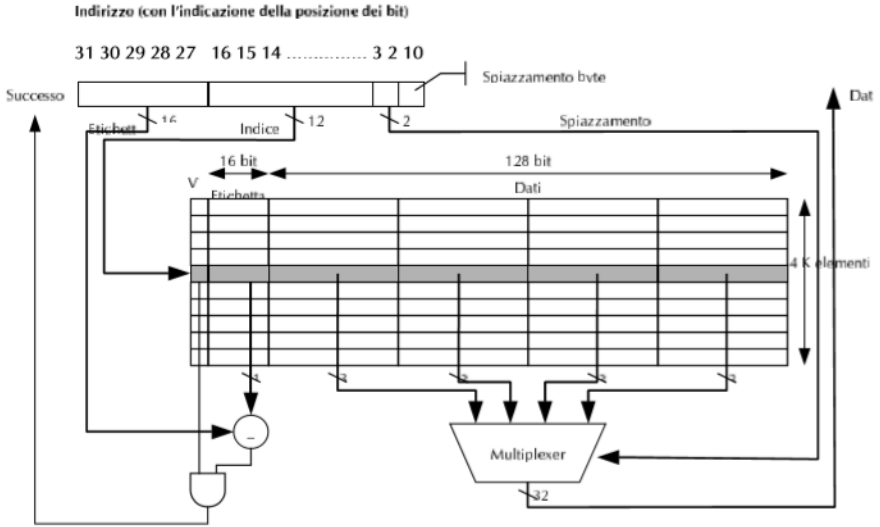
MEMORIA CACHE

Prelievo di blocco in una cache direct mapped

- La struttura per prelevare più parole adiacenti (un blocco) è costituita da:
  - L'**etichetta**, da confrontare con il contenuto del campo etichetta della cache, viene infatti utilizzata per controllare tutti i blocchi nell'insieme selezionato dall'indice (cache set-associativa), il blocco selezionato dall'indice (cache a indirizzamento diretto) oppure tutti i blocchi (cache completamente associativa).
  - L'**indice** serve a identificare l'insieme (cache set-associativa) oppure il blocco (cache a indirizzamento diretto). In una cache completamente associativa, il campo indice non serve poiché c'è un solo insieme.
  - Lo **spiazzamento** (offset) nel blocco indica l'indirizzo della parola da prelevare all'interno del blocco
- Questa struttura viene utilizzata sia per cache a indirizzamento diretto, sia per quelle associative e quella set-associative: diverso è l'uso che se ne fa nei diversi casi.

Etichetta	Indice	Spiazzamento

Si consideri una cache a indirizzamento diretto da 64Kbyte e blocco da 128 bit (quattro parole):

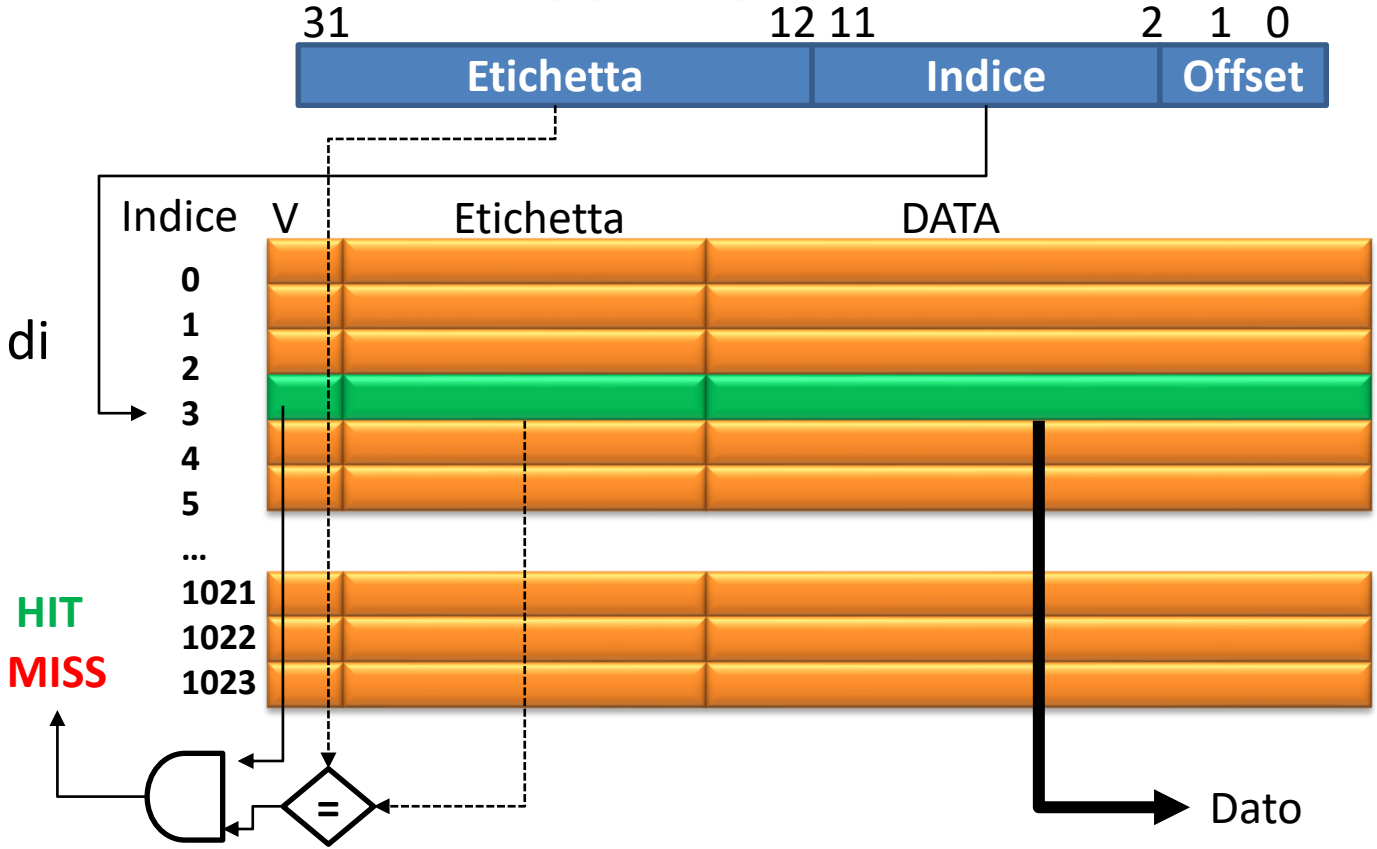


La cache contiene 4K (2^{12}) blocchi: *12* bit sono utilizzati per l'*indice* della cache.
Ogni blocco è composto da *4* parole (*4 x 32 bit = 16* byte): è necessario un campo aggiuntivo da *2* bit (i bit 3–2) per lo *spiazzamento della parola nel blocco*. Tali bit controllano il multiplexer in modo da selezionare la parola richiesta tra le *4* parole che si trovano nel blocco individuato.
I *2* bit meno significativi dell'indirizzo *(spiazzamento o offset)* specificano un byte all'interno di una parola da *32* bit: rimangono *32 – 12 – 2 – 2 = 16* bit per l'*etichetta*.

MEMORIA CACHE

Schema cache direct mapped per blocchi

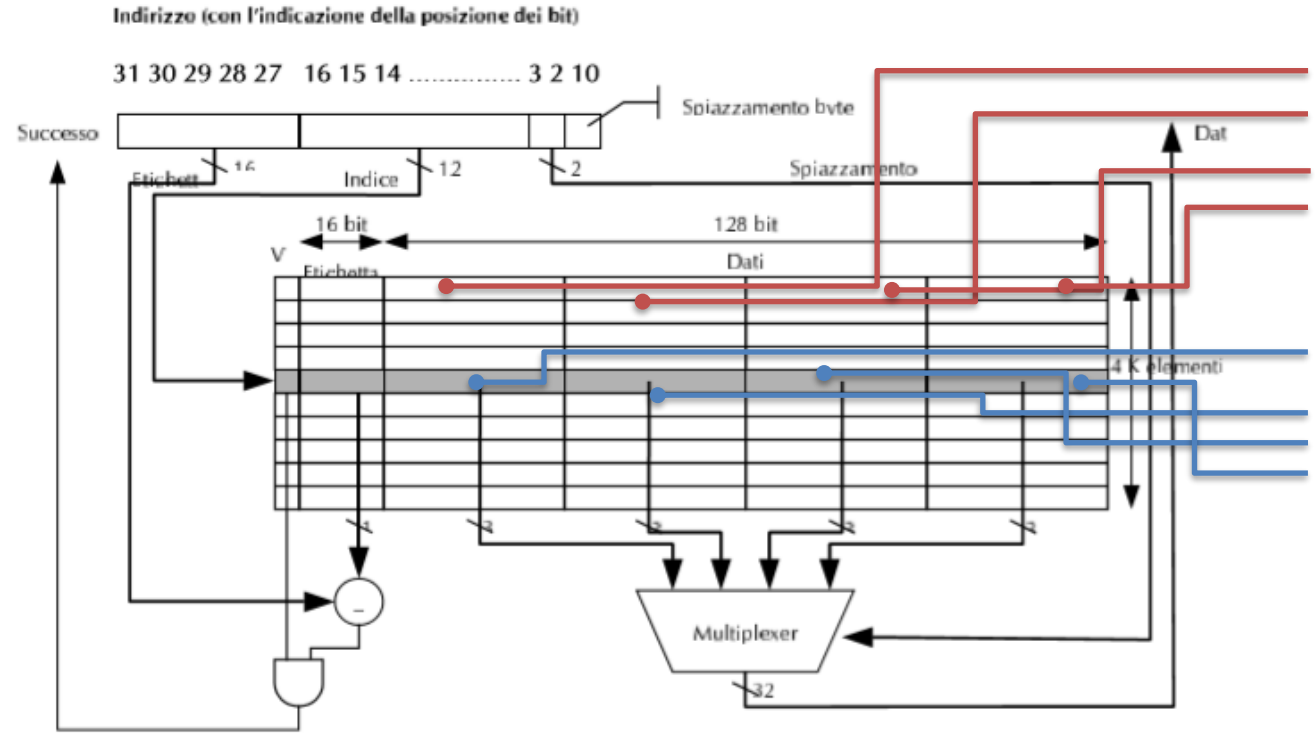
- Una cache a indirizzamento diretto con 1024 linee a campo DATA con blocco di 4 parole di 8bit può essere schematizzata come segue



MEMORIA CACHE

Prelievo di un blocco in una cache direct mapped

Si consideri una cache a indirizzamento diretto da 64Kbyte e blocco da 128 bit (quattro parole):

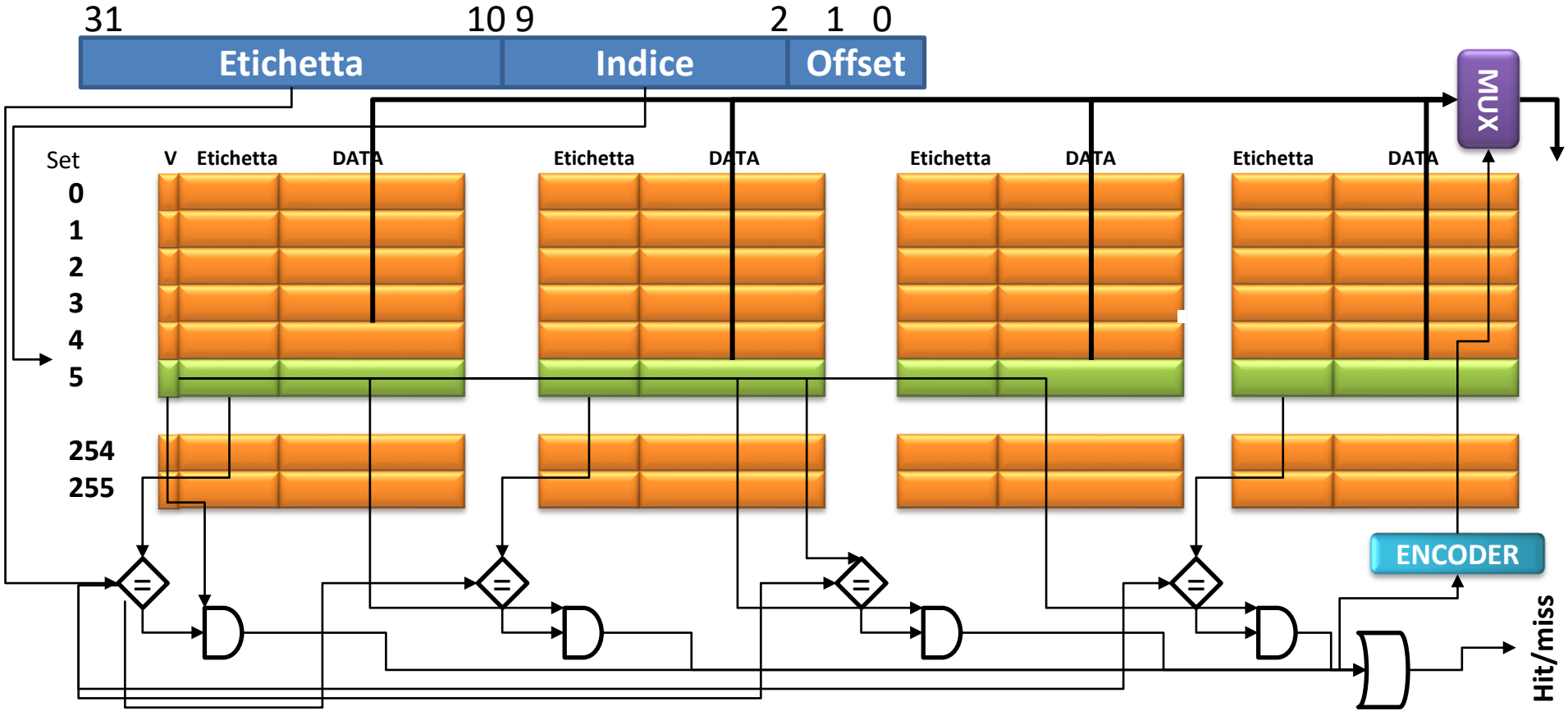


1101100010010010101000000000 00
1101100010010010101000000000 01
1101100010010010101000000000 10
1101100010010010101000000000 11

1101100010010010101000000100 00
1101100010010010101000000100 01
1101100010010010101000000100 10
1101100010010010101000000100 11

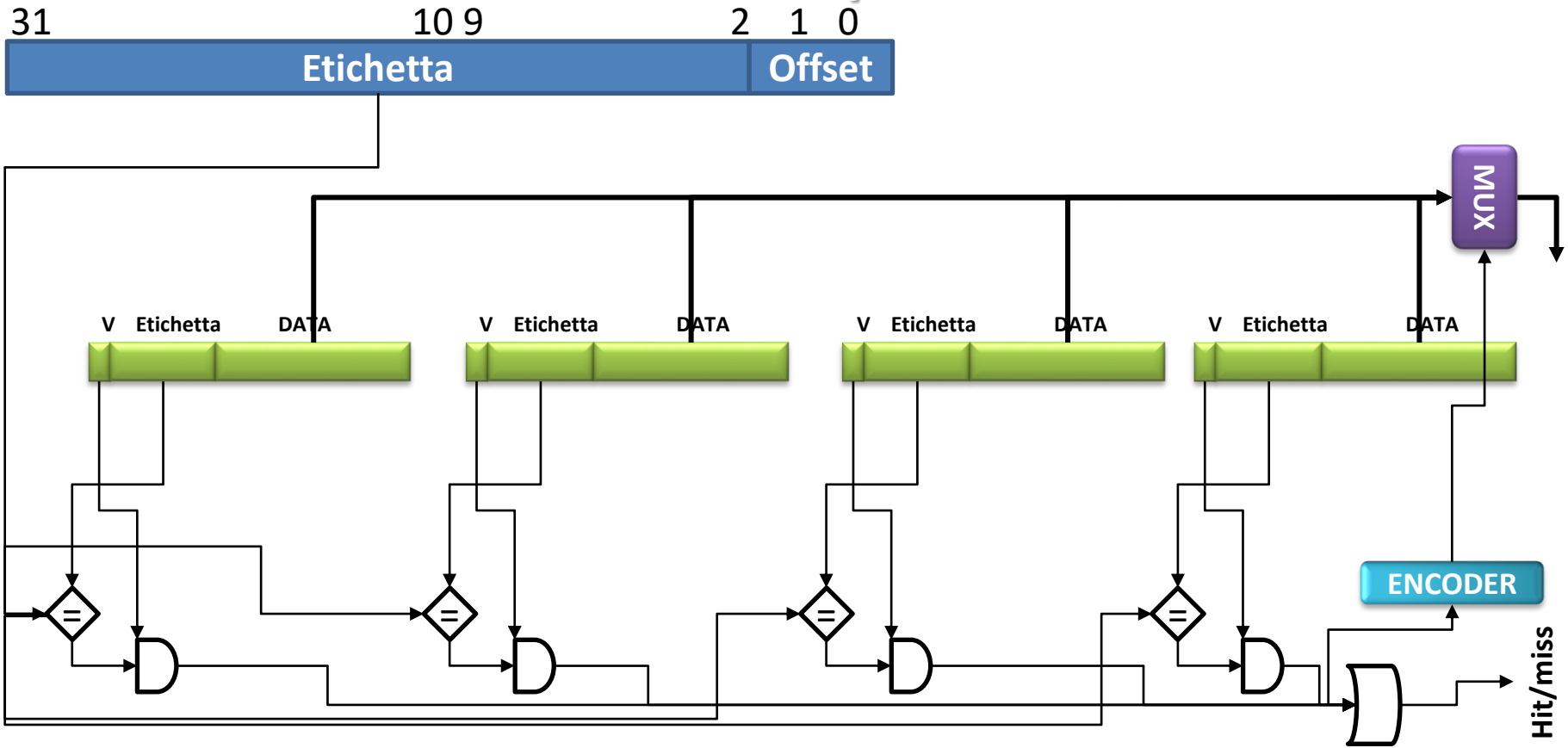
MEMORIA CACHE

Schema memoria cache set associativa a 4-vie



MEMORIA CACHE

Schema memoria cache completamente associativa



MEMORIA CACHE

Rimpiazzo del blocco: politiche

- Quando si verifica un fallimento nell'accesso alla cache, nel caso di **cache a indirizzamento diretto** c'è un solo candidato alla sostituzione: il problema si risolve quindi immediatamente
- In una **cache completamente associativa**, invece, bisogna decidere quale blocco sostituire: ogni blocco è un potenziale candidato per la sostituzione. Occorre stabilire una politica di sostituzione
- Se la **cache è set-associativa** l'insieme interessato è identificato immediatamente ma occorre stabilire una politica di sostituzione limitatamente ai blocchi compresi nell'insieme

MEMORIA CACHE

Rimpiazzo del blocco: politiche

- Le principali strategie utilizzate per la scelta del blocco da sostituire sono sostanzialmente tre:
  1. **Blocco utilizzato meno di recente** (***Least Recently Used - LRU***), il blocco sostituito è quello che è rimasto inutilizzato da più lungo tempo. A questo scopo, nei termini più semplici ad ogni blocco si associano dei contatori (i bit **TIMEUSED**) verso il basso che al momento della scrittura nel blocco sono posti al valore massimo e che sono poi decrementati di un'unità ogni volta che si effettua una lettura in un blocco diverso. La sostituzione toccherà quindi il blocco associato al contatore col valore più basso
  2. **Blocco meno usato** (***Least Frequently Used, LFU***). Si può associare un contatore (i bit **NUMBUSED**) a ogni blocco ed aggiornarlo ad ogni accesso
  3. **Sostituzione casuale** (***random***), la scelta tra i blocchi candidati è effettuata a caso, eventualmente utilizzando dei componenti hardware di supporto per l'identificazione del blocco

MEMORIA CACHE

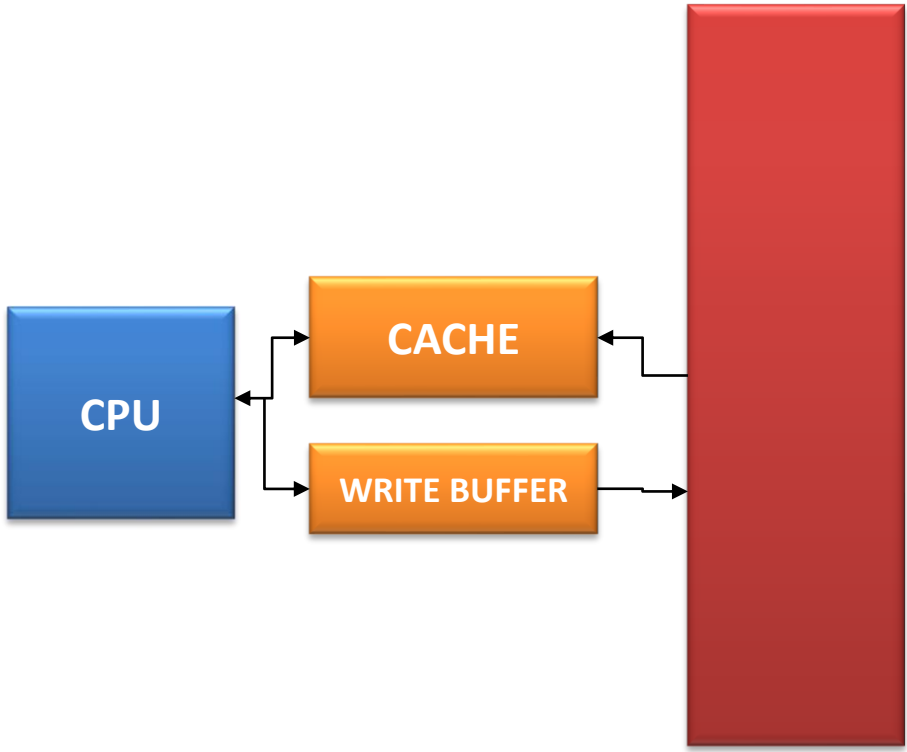
Aggiornamento della memoria: politiche

- Quando un dato è modificato in cache la memoria di livello inferiore deve essere aggiornata
- Politiche di scrittura
  - **Write Through**: ad ogni modifica è aggiornato il blocco in memoria
    - **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 (mitigabile con un buffer di scrittura)
  - **Write Back:** il blocco viene aggiornato solo quando è sostituito
    - **PRO**: la scrittura del blocco in memoria avviene raramente (è un blocco «vecchio») quindi la cache è molto più veloce
    - **CONTRO**:il contenuto della cache non è più coerente con quello della RAM (complicando i sistemi multiprocessore e multi-cache)
    - Ottimizzazione:con il bit **Dirty** (modificato) si può evitare di scrivere i blocchi non modificati

MEMORIA CACHE

Aggiornamento della memoria: politiche

- Write Through è realizzato con buffer di scrittura per non aumentare troppo i tempi di scrittura dovuti alle inferiori prestazioni della memoria di livello inferiore
  - È necessario un **buffer di scrittura (write buffer)** tra Cache e Memoria
    - Processore: scrive i dati in cache e nel buffer di scrittura
    - Controllore di memoria: scrive i contenuti del buffer in memoria



MEMORIA CACHE

Miglioramento cache: rimpiazzo del blocco

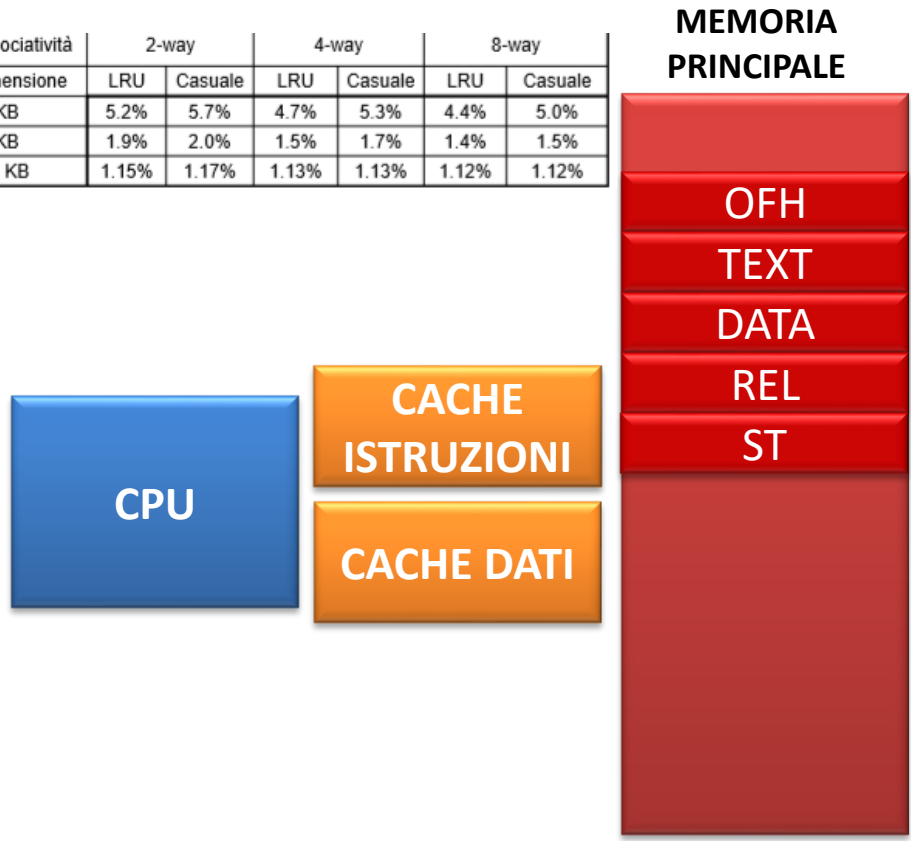
- Nei calcolatori moderni sono presenti più livelli di cache :
  - una cache di 1° livello, ormai sempre integrata nello stesso chip del processore, ad accesso rapidissimo;
  - una cache di 2° livello, talvolta esterna al chip del processore, ad accesso rapido;
  - a volte anche una cache di 3° livello.
- Lo scopo è di portare nella cache di 1° livello (più vicina al processore) le istruzioni ed i dati cui su cui il processore accederà nell'immediato futuro, spostando in una cache meno veloce (e meno costosa), ma più veloce della RAM, ciò che forse servirà più avanti (in modo da limitare le conseguenze negative dei miss)

MEMORIA CACHE

Miglioramento della cache: riduzione miss

- Aumentando le dimensioni del blocco di cache, il numero di miss si riduce fino a un certo punto
  - In alcuni casi però le prestazioni peggiorano perché nella sequenza di istruzioni sono presenti istruzioni di salto che riducono l'efficienza in termini di località spaziale
- Conviene introdurre cache separate per istruzioni e dati (**split cache**) riproponendo così la macchina di Harvard
  - Le operazioni di lettura/scrittura possono essere svolte in modo indipendente in ognuna delle due cache quindi di fatto si raddoppia la larghezza di banda della memoria
  - Le caratteristiche di località sono molto diverse per istruzioni e per dati

Associatività	2-way		4-way		8-way	
Dimensione	LRU	Casuale	LRU	Casuale	LRU	Casuale
16 KB	5.2%	5.7%	4.7%	5.3%	4.4%	5.0%
64 KB	1.9%	2.0%	1.5%	1.7%	1.4%	1.5%
256 KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%



Fine