Lecture
9_5.2

Cache Memories

Paolo PRINETTO
Politecnico di Torino (Italy)
University of Illinois at Chicago, IL (USA)

Paolo.Prinetto@polito.it
prinetto@uic.edu

www.testgroup.polito.it
www.comitato-girotondo.org

License Information

Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

Goal

– This lecture focuses on cache memories, presenting the related most important architectural solutions.

Prerequisites

- **Module 0_4 : Digital Systems: Definitions and Taxonomies**

Homework

- **None**

Further readings

- **Students interested in making a reference to a text book on the arguments covered in this lecture can refer, for instance, to:**
  - ***G. Conte, A. Mazzeo, N. Mazzocca, P. Prinetto*: "Architettura dei calcolatori", *Città Studi, 2015* (Chapter 7 – Sistema delle memorie)**

Outline

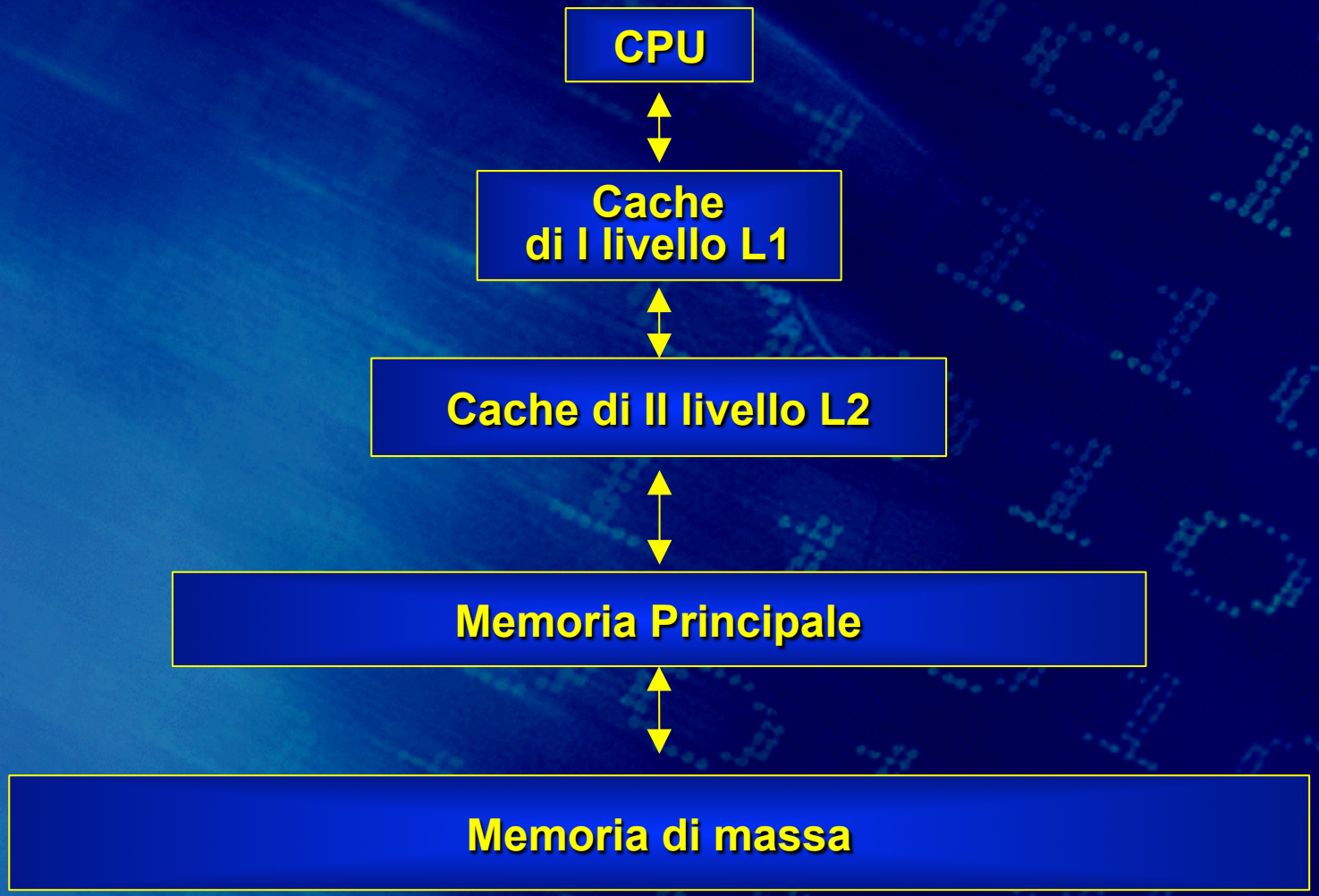
- Introduzione
- Il principio di località
- Struttura di una cache
- Parametri caratteristici
- Parametri fisici
- Funzione di traduzione (mapping)
- Algoritmi di sostituzione
- Aggiornamento della Memoria Principale
- Gestione della Coerenza
- Dimensionamento di una Cache
- Cache L1 e L2
- Esempi di cache.

Introduzione

- **Le memorie cache sono memorie di piccole dimensioni ma a elevata velocità, interposte tra il processore e la memoria principale.**

Rel. 30.05.2017

Gerarchia di Memorie



Outline

- Introduzione
- Il principio di località
- Struttura di una cache
- Parametri caratteristici
- Parametri fisici
- Funzione di traduzione (mapping)
- Algoritmi di sostituzione
- Aggiornamento della Memoria Principale
- Gestione della Coerenza
- Dimensionamento di una Cache
- Cache L1 e L2
- Esempi di cache.

Località dei riferimenti

- La presenza di una cache può migliorare le prestazioni di un sistema grazie alla ***località dei riferimenti*** osservabile nella maggioranza dei programmi

Principio di Località

- **Una delle principali proprietà dei programmi è la: *Locality of references (90/10 rule)***
  - **I Programmi tendono a riusare i dati e le istruzioni che hanno usato di recente**

Principio di Località

- Una regola empirica largamente verificata asserisce che un programma in media trascorre il 90% del suo tempo di esecuzione in una porzione di codice che rappresenta circa il 10% dell'intero programma (la stessa regola vale per i riferimenti sui dati)
- Molte istruzioni, quindi, in aree ben localizzate di un programma, vengono eseguite ripetutamente in un determinato periodo, e si accede al resto del programma relativamente di rado.

Conseguenza

- **Basandosi sulle ultime istruzioni eseguite da un programma è possibile predire con ragionevole accuratezza quali istruzioni e quali dati del programma verranno utilizzati nel prossimo futuro.**

Principio di Località

- **Si esprime in due forme*:***
  - ***Località Temporale***
  - ***Località Spaziale***

Località Temporale

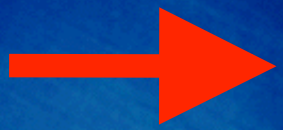
- **Rappresenta la probabilità (alta) che una istruzione eseguita di recente venga eseguita di nuovo entro breve tempo**
- **Se all'istante `t` il programma fa accesso a una cella di memoria, è molto probabile che il programma faccia nuovamente accesso alla stessa cella entro l'istante `t + D`**

Località Spaziale

- **Rappresenta la probabilità (alta) che istruzioni vicine a un'istruzione eseguita di recente siano anch'esse eseguite nel prossimo futuro.**
- **La vicinanza è espressa in termini di indirizzi delle istruzioni**
- **Se all'istante `t` il programma fa accesso alla cella di memoria di indirizzo `X`, è molto probabile che entro l'istante `t + D` il programma faccia accesso anche alla cella di indirizzo `X + E`.**

Principio di Località

Località Temporale



Portare un elemento (istruzioni o dati) nella cache quando viene richiesto per la prima volta, in modo che rimanga a disposizione nel caso di una nuova richiesta

Rel. 30.05.2017

Principio di Località

Località Temporale → **Portare un elemento (istruzioni o dati) nella cache quando viene richiesto per la prima volta, in modo che rimanga a disposizione nel caso di una nuova richiesta**

Località Spaziale → **Invece di portare dalla memoria principale alla cache un elemento alla volta, è conveniente portare un insieme (*block)* di elementi che risiedono in indirizzi adiacenti**

Rel. 30.05.2017

Principio di funzionamento

- **Se all'istante t (primo accesso a un blocco di memoria da parte del programma) viene caricato nella cache l'intero blocco, ci sono alte probabilità che per un certo tempo D il programma trovi nella cache tutte le parole di memoria cui deve fare accesso.**

Outline

- Introduzione
- Il principio di località
- Struttura di una cache
- Parametri caratteristici
- Parametri fisici
- Funzione di traduzione (mapping)
- Algoritmi di sostituzione
- Aggiornamento della Memoria Principale
- Gestione della Coerenza
- Dimensionamento di una Cache
- Cache L1 e L2
- Esempi di cache.

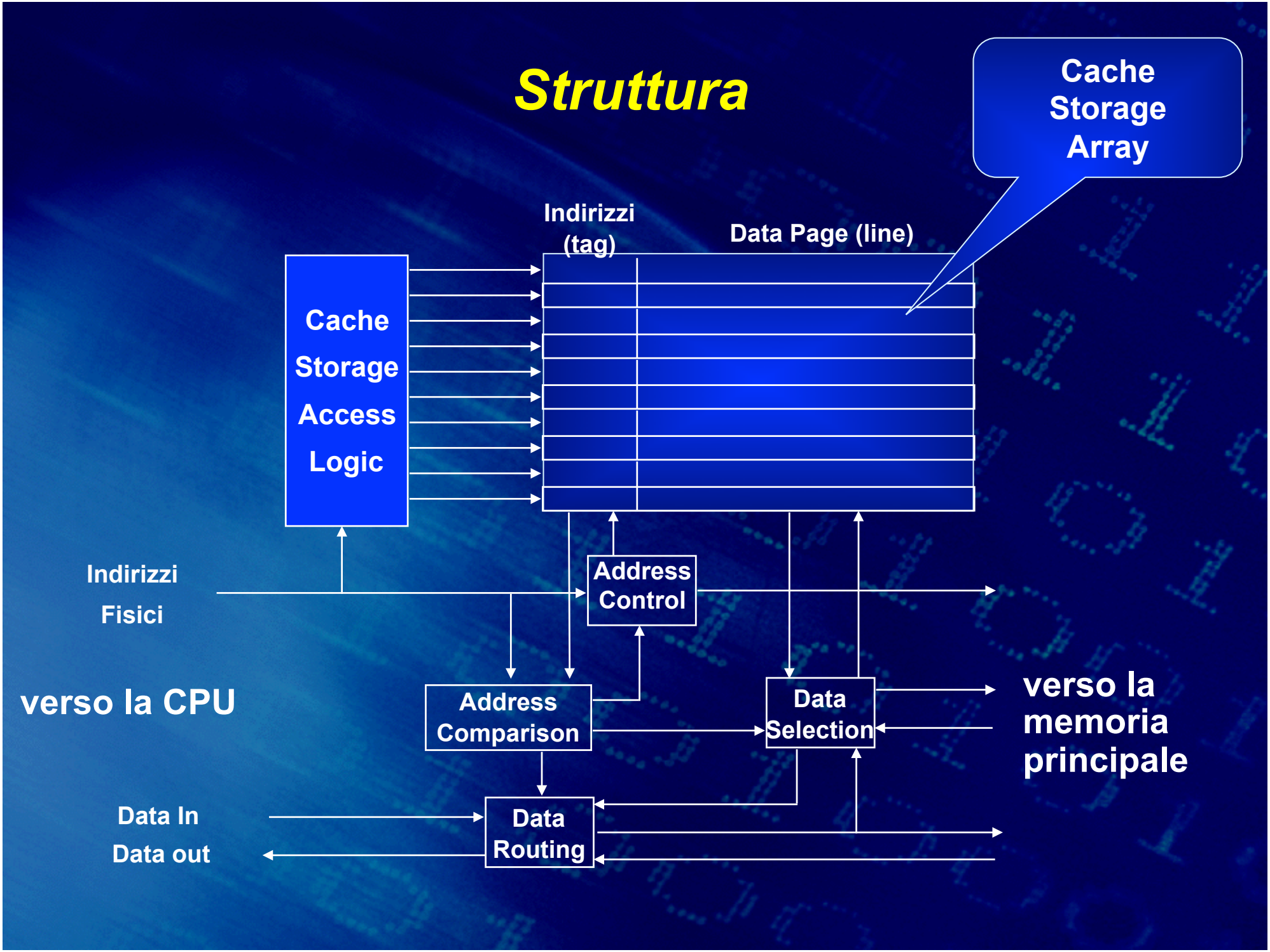
Struttura di una cache

- Una cache contiene al suo interno un certo numero di *linee*
- Ciascuna linea contiene un blocco di memoria
- A ogni linea è associato un campo *tag*, che identifica il blocco di memoria presente nella linea in quel momento

Struttura di una cache *(cont'd)*

- La cache contiene inoltre la logica per:
  - intercettare gli indirizzi prodotti dal processore
  - controllare al proprio interno la presenza di un blocco
  - eventualmente caricarne uno nuovo da memoria
  - gestire le operazioni di scrittura in memoria.

Struttura


Cache Storage Array
Indirizzi (tag)
Data Page (line)
Cache Storage Access Logic
Indirizzi Fisici
Address Control
verso la CPU
Address Comparison
Data Selection
verso la memoria principale
Data In
Data out
Data Routing

Principio di funzionamento

- **Ogni volta che il processore esegue un accesso alla memoria, la cache:**
  - **intercetta l'indirizzo**
  - **verifica se il blocco cui appartiene la parola è presente nella cache, controllando il valore dei tag**
  - **se sì (*hit*): estrae la parola dal blocco e la fornisce alla CPU al posto (e prima) della memoria principale**
  - **se no (*miss*): provvede a caricare nella cache l'intero blocco di cui la parola fa parte.**

Prestazioni

- **In caso di hit, la cache riduce i tempi di accesso di un fattore tipicamente compreso tra 2 e 6**

Prestazioni

- In caso di miss, la cache può rispondere in due possibili modi:
  - Accede alla memoria e carica l'intero blocco mancante; poi fornisce la parola richiesta. Il tempo di accesso è quindi superiore a quello della memoria.
  - Accede alla memoria e fornisce subito la parola richiesta; poi provvede al caricamento del blocco (load-through o early restart). Questa tecnica richiede un maggior costo dell'hardware della cache.

Posizione della cache

- La soluzione più seguita consiste nel fare della cache una parte della CPU, anziché una parte della memoria principale.
- I vantaggi che si ottengono sono:
  - si alleggerisce il carico del bus
  - la soluzione è compatibile con un'architettura multiprocessore.

Instruction Cache e Data Cache

- **Tipicamente vi sono cache separate per dati e istruzioni**
- **La cache per le istruzioni è in genere più semplice da gestire di quella per i dati, in quanto le istruzioni non possono essere modificate.**

Outline

- Introduzione
- Il principio di località
- Struttura di una cache
- Parametri caratteristici
- Parametri fisici
- Funzione di traduzione (mapping)
- Algoritmi di sostituzione
- Aggiornamento della Memoria Principale
- Gestione della Coerenza
- Dimensionamento di una Cache
- Cache L1 e L2
- Esempi di cache.

Parametri Caratteristici

- **Sono:**
  - *Parametri fisici*
  - *Funzione di traduzione (Mapping)*
  - *Algoritmo di Rimpiazzamento o Sostituzione*
  - *Metodo di Aggiornamento della Memoria Principale*
  - *Gestione della Coerenza*

Parametri fisici

- ***Block Size***
  - Un Blocco è un insieme di indirizzi contigui di una qualche dimensione
  - Per la proprietà di località spaziale conviene portare in cache un insieme di elementi facenti riferimento a indirizzi contigui

Parametri fisici

- ***Hit Time***
  - La memoria cache può memorizzare solo un piccolo sottoinsieme di blocchi presenti in memoria principale
  - Il numero di cicli di clock necessari per caricare un elemento che si trova in cache (in genere coincidente con 1 ciclo di clock)

Parametri fisici

- ***Miss Penalty***
  - Quando un elemento non viene trovato in cache si ha un miss
  - La Miss Penalty rappresenta il numero di cicli di clock necessari per caricare l'elemento dalla memoria principale nella cache (o eventualmente direttamente nel processore)
  - In genere,

$$t(\text{miss}) \cong 10 \cdot t(\text{hit})$$

Parametri fisici

- ***Access Time***
  - **Tempo di accesso alla memoria principale (o a una cache di livello superiore)**

Parametri fisici

- ***Transfer Time***
  - Tempo di trasferimento del blocco in cui è presente l'elemento referenziato dalla memoria principale alla memoria cache

Parametri fisici

- ***Miss Rate***
  - **Percentuale di miss (relativo all'architettura del sistema, alla funzione di mapping, agli algoritmi di gestione della coerenza, all'applicazione in esecuzione ...)**

Parametri fisici

- ***Cache size***
  - La scelta della dimensione ottimale è influenzata da:
    - costo
    - prestazioni: al crescere delle dimensioni, le cache diventano più lente.

Outline

- Introduzione
- Il principio di località
- Struttura di una cache
- Parametri caratteristici
- Parametri fisici
- Funzione di traduzione (mapping)
- Algoritmi di sostituzione
- Aggiornamento della Memoria Principale
- Gestione della Coerenza
- Dimensionamento di una Cache
- Cache L1 e L2
- Esempi di cache.

Funzione di traduzione (mapping)

- **Specifica la corrispondenza tra i blocchi della memoria principale e quelli della cache, definendo in quale linea della cache viene memorizzato un determinato blocco di memoria.**

Ricerca di un Blocco in una Cache

- A ogni linea della cache è associato un campo Label
- Ai fini della ricerca di un blocco in una cache tutte le label che possono contenere informazioni su di esso vengono confrontate in parallelo
- Anche il dato viene letto parallelamente al confronto in modo da averlo immediatamente pronto in caso di hit

Funzione di traduzione (mapping)

- **Si deve garantire (a un prezzo accettabile) che la cache possa rapidamente verificare se contiene il dato corrispondente a un certo indirizzo.**
- **Possibili soluzioni:**
  - **memoria associativa (troppo costosa)**
  - **scansione sequenziale (troppo lenta)**

Meccanismi di mapping

- **I meccanismi di mapping sono tre:**
  - ***direct mapping***
  - ***fully associative mapping***
  - ***set associative mapping***

Direct Mapping

- Ogni blocco della memoria principale è messo in corrispondenza fissa con una linea k della cache.
- La funzione di trasformazione è

$$k = J \bmod N$$

dove:

  - N = numero di linee della cache
  - J = numero del blocco all'interno della memoria principale

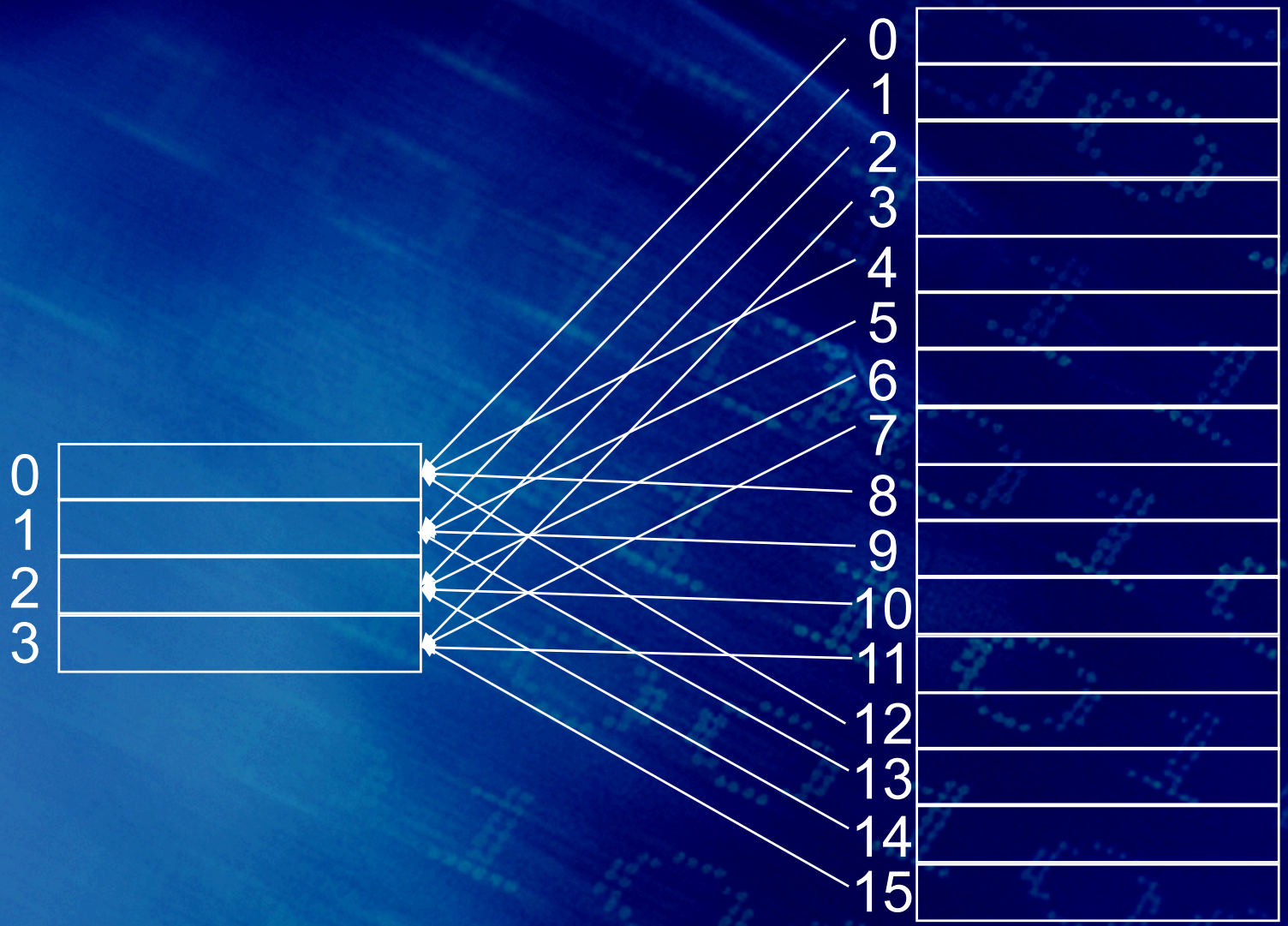
Esempio

- **Assumendo che**
  - **La cache sia composta da 4 linee**
  - **La memoria sia composta da 16 blocchi**

  si ha:

Rel. 30.05.2017

Esempio



Direct Mapping

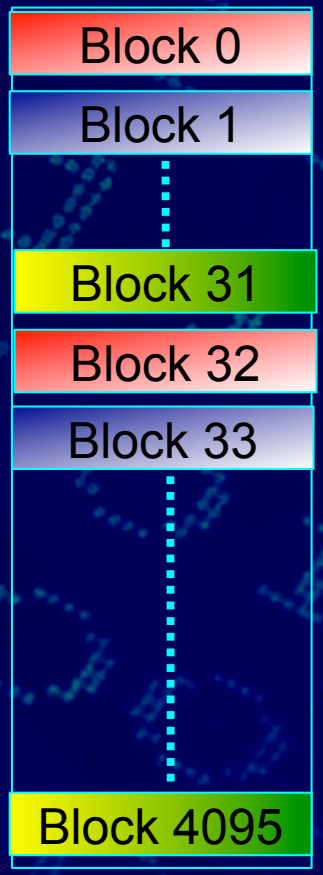
- **Vantaggi:**
  - **la funzione è facilmente implementabile in hardware**
- **Svantaggi:**
  - **se il programma fa accesso frequentemente a 2 blocchi corrispondenti alla stessa linea della cache, a ogni accesso si verifica un miss.**

Esempio

- **Nell'esempio seguente si farà riferimento a:**
  - **una memoria principale di 1 Mb, organizzata in 64 k x 16, a sua volta suddivisa in 4 k blocchi di 16 word; (64 k → 16 bit di indirizzo)**

Esempio

Cache

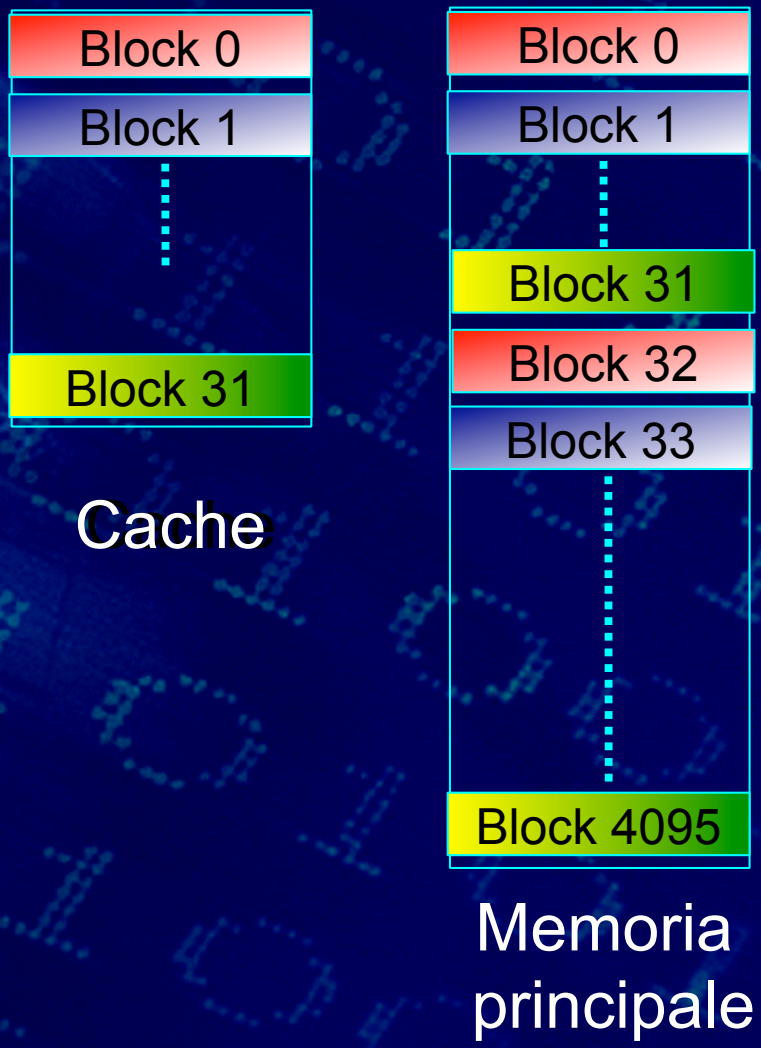


Memoria principale

Esempio

- **Nell'esempio seguente si farà riferimento a:**
  - **una memoria principale di 1 Mb, organizzata in 64 k x 16, a sua volta suddivisa in 4 k blocchi di 16 word; (64 k → 16 bit di indirizzo)**
  - **una cache di 8 kB, anche lei organizzata in 32 blocchi di 16 word ciascuno**

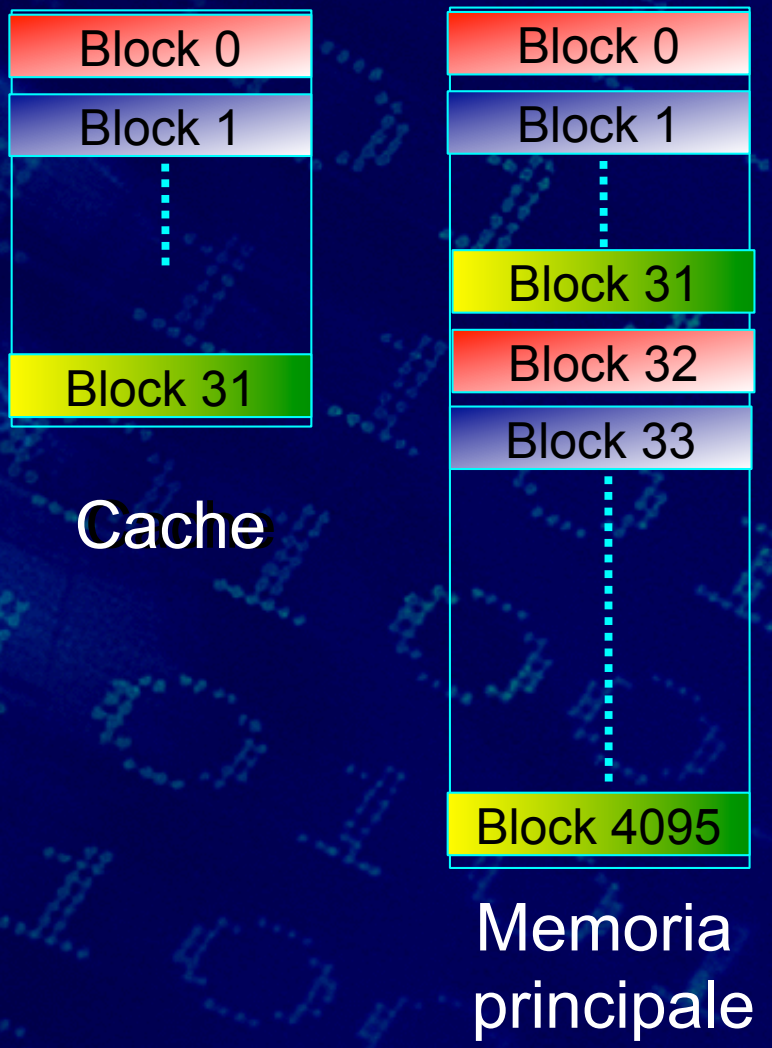
Esempio



Esempio

Quanti blocchi possono essere mappati nel blocco J-esimo della Cache?

4096 / 32 = 128 ⇒ 7 bit

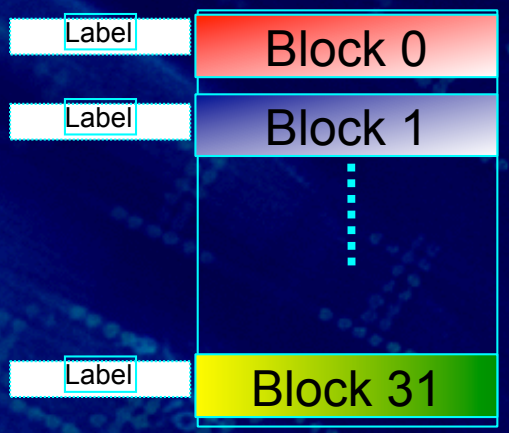


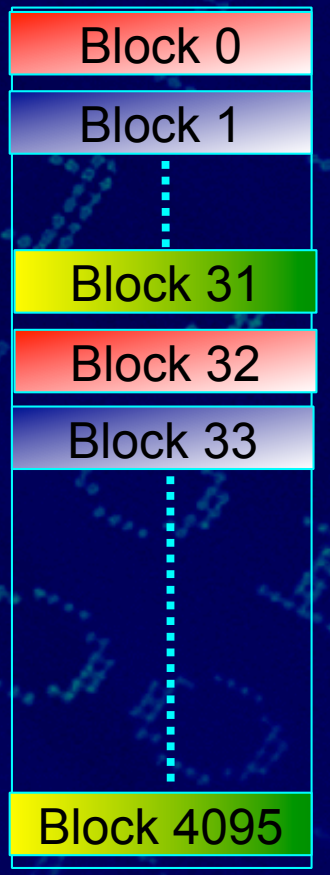
Esempio

Quanti blocchi possono essere mappati nel blocco J-esimo della Cache?

4096 / 32 = 128 ⇒ 7 bit

A ogni blocco della Cache viene associata una Label di 7 bit che identifica quale blocco della memoria esso contiene





Rel. 30.05.2017

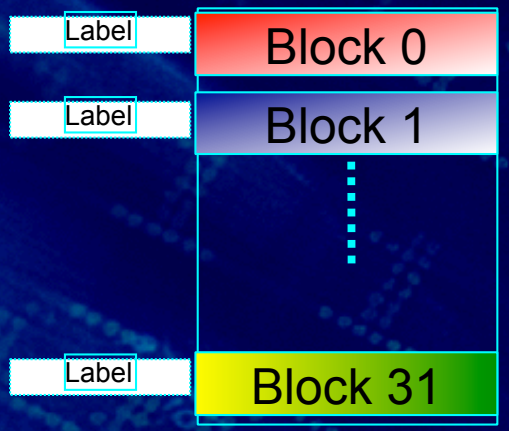
Direct Mapping: struttura dell'indirizzo

- L'indirizzo viene scomposto dalla cache in tre parti:

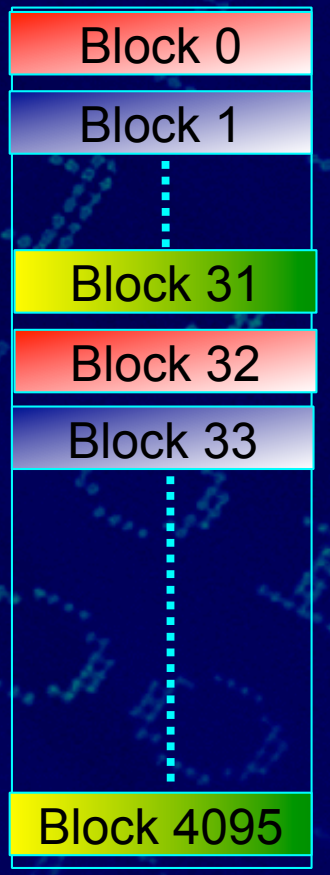
- La presenza o meno del blocco in cache viene verificata comparando il campo Label nell'address con il campo *Label* nella cache
- Il campo *Block* identifica il blocco all'interno della cache
- Il campo *Word* indirizza la word nel blocco

Esempio

Address in memoria principale



Cache



Memoria principale

Fully Associative Mapping

- **Ogni blocco della memoria principale può essere memorizzato in un qualsiasi blocco della cache**

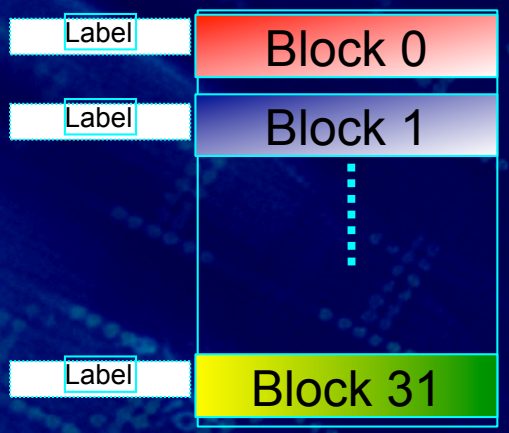
Fully Associative Mapping

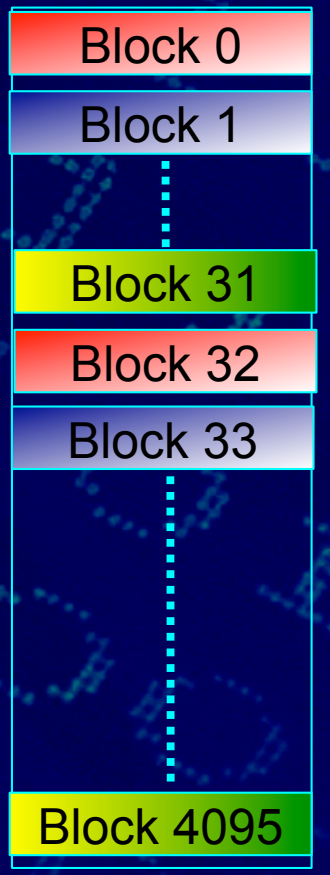
- **Vantaggi:**
  - **massima flessibilità nella scelta del blocco di cache da usare**
- **Svantaggi:**
  - **complessità dell'hardware di ricerca (di solito si adotta una memoria associativa).**

Esempio

Quanti blocchi possono essere mappati nel blocco J-esimo della Cache?

4096 ⇒ 12 bit



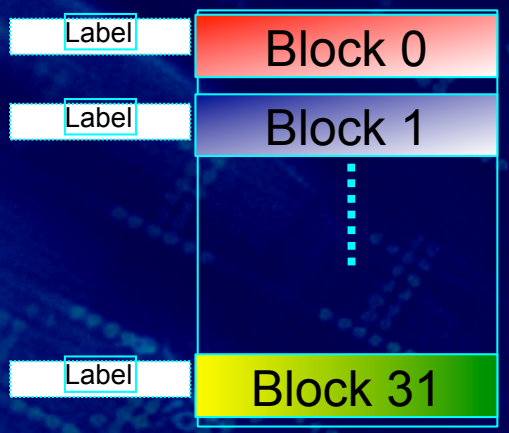


Esempio

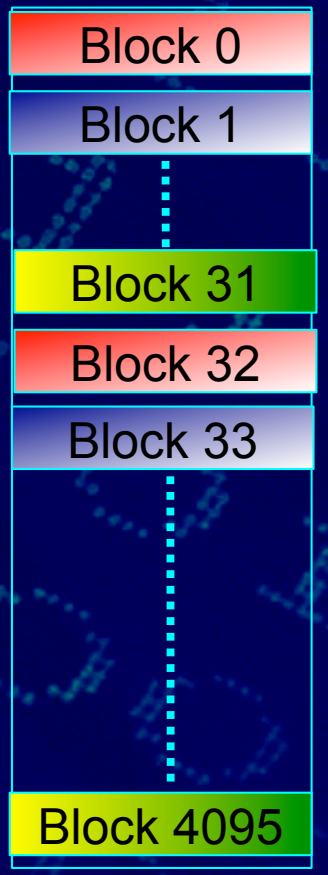
Quanti blocchi possono essere mappati nel blocco J-esimo della Cache?

4096 ⇒ 12 bit

A ogni blocco della Cache viene associata una Label di 12 bit che identifica quale blocco della memoria esso contiene



Cache



Memoria principale

Rel. 30.05.2017

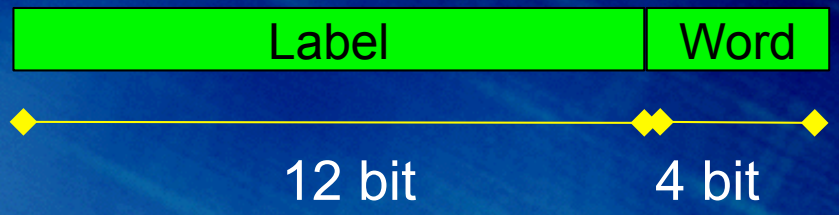
Struttura dell'indirizzo

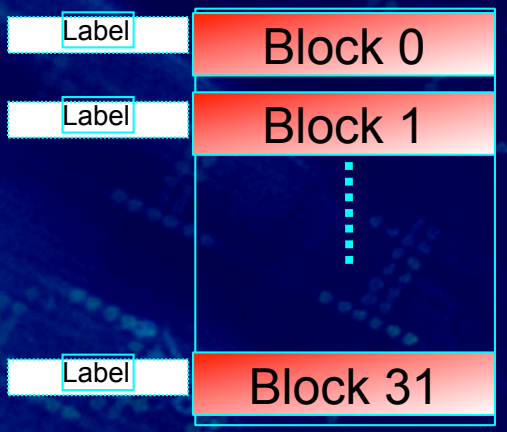
- L'indirizzo viene scomposto dalla cache in due parti:

- La presenza o meno del blocco in cache viene verificata comparando il campo Label nell'address con il campo Label nella cache
- Il campo word indirizza la word nel blocco (Block Offset)

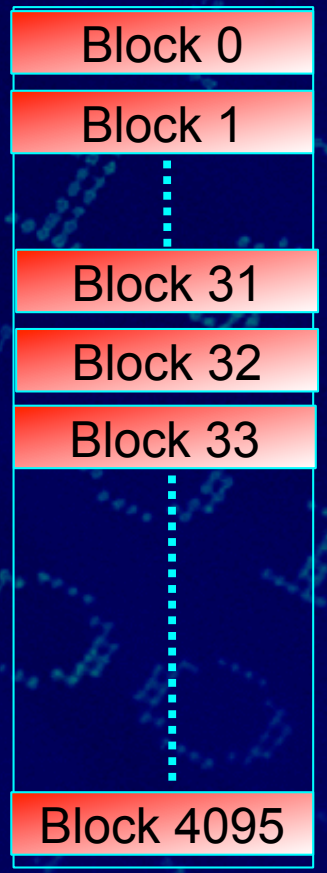
Esempio

Address in memoria principale





Cache



Memoria principale

Set Associative Mapping

- **Un blocco può essere posizionato in un ristretto insieme (set) di blocchi nella cache**
- **Un blocco viene prima mappato in un set e può essere inserito ovunque nel set**
- **La funzione di mapping del set in genere è una funzione in modulo**
- **Se ci sono n blocchi in ogni set la cache viene detta : n-way set associative**

Set Associative Mapping

- **Principio di funzionamento:**
  - **i blocchi della memoria principale sono suddivisi in S1 insiemi di dimensione D1**
  - **le linee della cache sono suddivise in S2 insiemi di dimensione D2<< D1**
  - **un blocco i è associato all'insieme di linee k se k=i mod S2**
  - **il blocco i può essere messo in una qualunque delle linee dell'insieme k.**

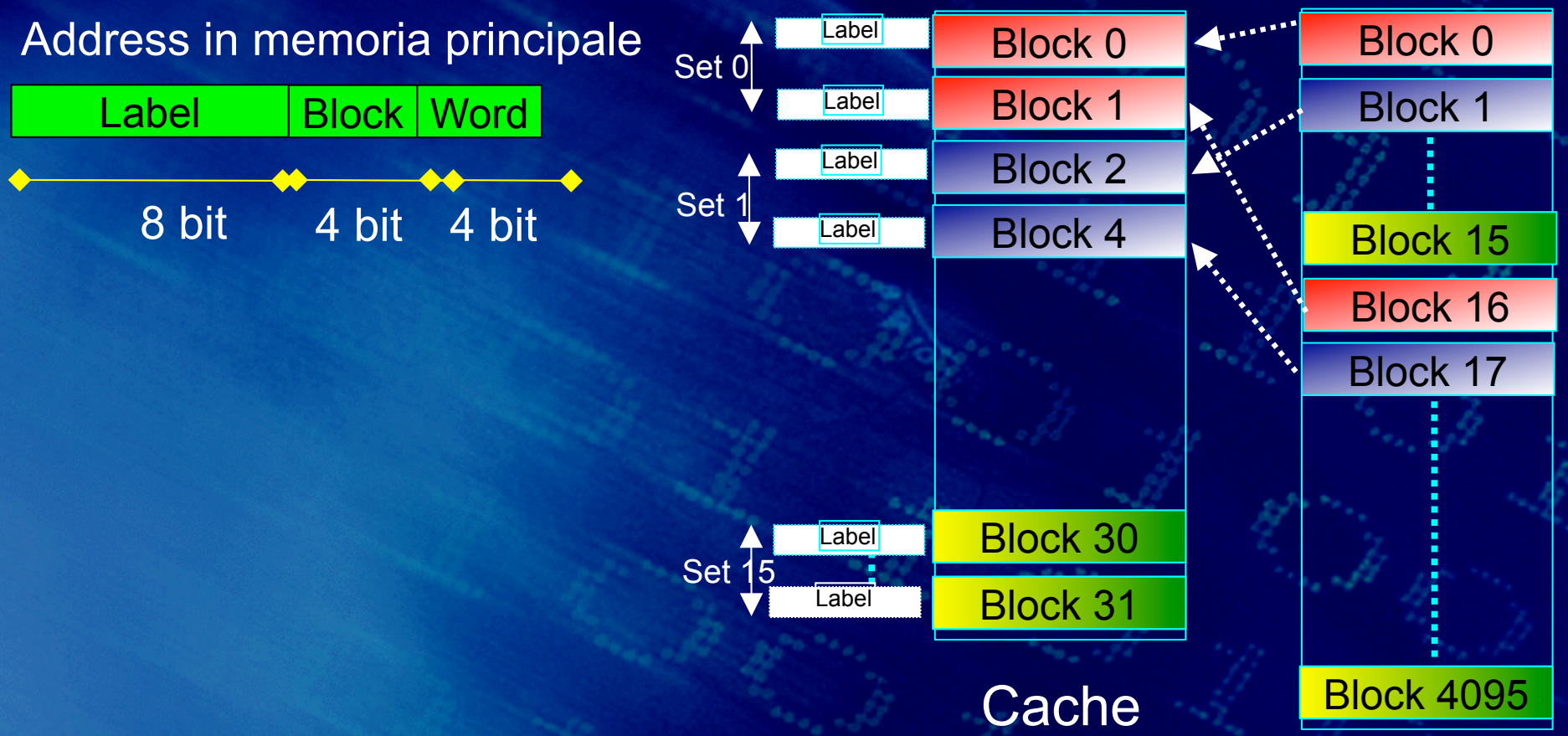
Set Associative Mapping

- **Se S2 = 1 si ha il direct mapping**
- **se S2 = N (N è la dimensione della cache) si ha il fully associative mapping.**

Struttura dell'indirizzo

- **L' indirizzo viene scomposto dalla cache in tre parti:**

Set Associative Mapping



Operazione di scrittura

- Nel caso di una operazione di scrittura, non è possibile cominciare l'operazione finchè non si ha un hit
- Visto che l'operazione di accesso all'area dati della cache, non può avvenire in parallelo con il confronto delle label, in genere, un'operazione di scrittura impiega più tempo di una di lettura.

Operazione di scrittura

- **Nel caso di una operazione di scrittura, occorre garantire la coerenza tra il contenuto della cache e quello della memoria principale**

Aggiornamento della Memoria Principale

- **La CPU ha normalmente un canale di connessione diretto con la memoria principale**
- **Questo permette l'implementazione di 2 diverse politiche di base per la gestione delle operazioni di scrittura:**
  - ***write-back***
  - ***write-through***

Write Back

- **L'informazione viene scritta solo nel blocco della cache.**
- **Per ogni blocco nella cache viene tenuto aggiornato un flag (*dirty bit*), settato se il blocco è stato modificato da quando è stato caricato nella cache.**
- **Quando un blocco viene eliminato dalla cache, esso viene copiato nella memoria principale solo se il suo dirty bit è settato.**

Write Through

- **La CPU esegue tutte le operazioni di scrittura, sia sul dato presente nella cache sia in quello presente nella memoria principale.**
- **La perdita di efficienza che ne deriva è limitata dal fatto che le operazioni di scrittura sono di solito molto meno numerose di quelle di lettura.**

Write Through

- **Quando una CPU attende il completamento di una scrittura si dice in *write stall***
- **Un'ottimizzazione comune per ridurre lo stallo è quello di utilizzare dei write buffers che permettono al processore di continuare l'esecuzione del programma mentre la memoria viene aggiornata**

Vantaggi e Svantaggi

- **Sia Write back sia Write Through hanno i loro pro e contro**

Write Through

+ La memoria principale è sempre allineata con la cache (importante per sistemi multiprocessore)
+ Più facilmente implementabile

– scritture alla velocità della memoria più lenta
– 1 scrittura per ogni modifica del blocco => occupa più banda (e quindi viene utilizzata per lo più tra cache L1 e cache L2)

Write Back

+ scritture alla velocità della memoria cache
+ scritture multiple sullo stesso blocco richiedono 1 sola scrittura nella memoria più lenta (meno occupazione di banda)

– nei sistemi multiprocessore si può avere inconsistenza tra le cache di diversi processori
– il ripristino dei dati dopo eventuali system failure può non essere possibile.

Outline

- Introduzione
- Il principio di località
- Struttura di una cache
- Parametri caratteristici
- Parametri fisici
- Funzione di traduzione (mapping)
- Algoritmi di sostituzione
- Aggiornamento della Memoria Principale
- Gestione della Coerenza
- Dimensionamento di una Cache
- Cache L1 e L2
- Esempi di cache.

Algoritmi di Sostituzione

- Quando si ha un cache miss su un blocco:

  IF (la cache ha blocchi disponibili per inserire un nuovo blocco)

  THEN si procede al caricamento dalla memoria nel set relativo al blocco

  ELSE si sostituisce 1 dei blocchi presenti nella cache con il nuovo blocco

Algoritmi di Sostituzione

- **I meccanismi di sostituzione sono diversi a seconda della struttura della cache.**

Considerazioni

- Una cache direct mapped possiede un solo blocco disponibile per ogni blocco in memoria principale
- Ne deriva che la copiatura di un nuovo blocco nella cache provoca sempre la sostituzione del blocco precedente

Considerazioni

- Una cache fully associative ha a disposizione tutta la cache per il caching dei blocchi: Un nuovo blocco che deve essere copiato nella cache fa uscire un altro già residente solo se l'intera cache è già piena
- Una cache set-associative ha per ogni blocco in memoria principale uno spazio pari al suo grado di associatività: Un nuovo blocco che deve essere copiato nella cache ne fa uscire un altro già residente solo se il set a esso associato è già pieno.

Sostituzione in Direct Mapped Cache

- Si procede direttamente alla sostituzione del blocco (se il dirty bit == 0, non si effettua la copia in memoria principale)

Sostituzione in Set e Fully Associative

- **IF (esiste nel set un blocco col dirty bit non settato)**

  THEN lo si sostituisce

  ELSE si identifica il "*miglior*" blocco da sostituire.

Algoritmi di Sostituzione

- In quest'ultimo caso, gli approcci possibili sono:
  - ***LRU (Least Recently Used):*** **il più ragionevole**
  - ***FIFO (First-In First-Out):*** **il più economico**
  - ***LFU (Least Frequently Used):*** **teoricamente il più efficace**
  - ***Random*****: il più usato ed efficiente.**

Algoritmi di Sostituzione

- **Random**
  - **Viene usato per distribuire uniformemente l'allocazione dei blocchi**
  - **Il blocco candidato viene scelto in modo pseudocasuale**
  - **Utilizzare un algoritmo pseudocausale e quindi riproducibile risulta particolarmente indicato per facilitare le operazioni di debugging**

Algoritmi di Sostituzione

- LRU
  - Riduce la probabilità di sostituire blocchi cui si farà accesso nel prossimo futuro
  - Il blocco sostituito è quello che non si usa da più tempo

LRU

- Il controllore ha bisogno di mantenere traccia degli accessi ai blocchi mentre l'elaborazione prosegue
- Viene utilizzato un contatore per ogni blocco

Rel. 30.05.2017

LRU

- Quando vi è un hit :
  - Il contatore del blocco dell'hit viene azzerato
  - I contatori con valori inferiori a quello dell'hit vengono incrementati
  - Gli altri rimangono invariati
- Se vi è un miss e vi è un blocco libero
  - Viene inserito il nuovo blocco con contatore=0
  - Tutti gli altri contatori vengono incrementati di 1

LRU

- Se vi è un miss e tutti i blocchi nel set sono occupati:
  - Si sposta un blocco con il valore più alto di conteggio
  - Si inserisce il nuovo blocco con il contatore a 0
  - Tutti gli altri contatori vengono incrementati di 1

Considerazioni

- Risultati sperimentali mostrano che l'algoritmo Random ha prestazioni pressocchè identiche a quello LRU quando la cache supera i 64 kB
- L'Hardware per implementare l'algoritmo Random è molto più semplice di quello di LRU

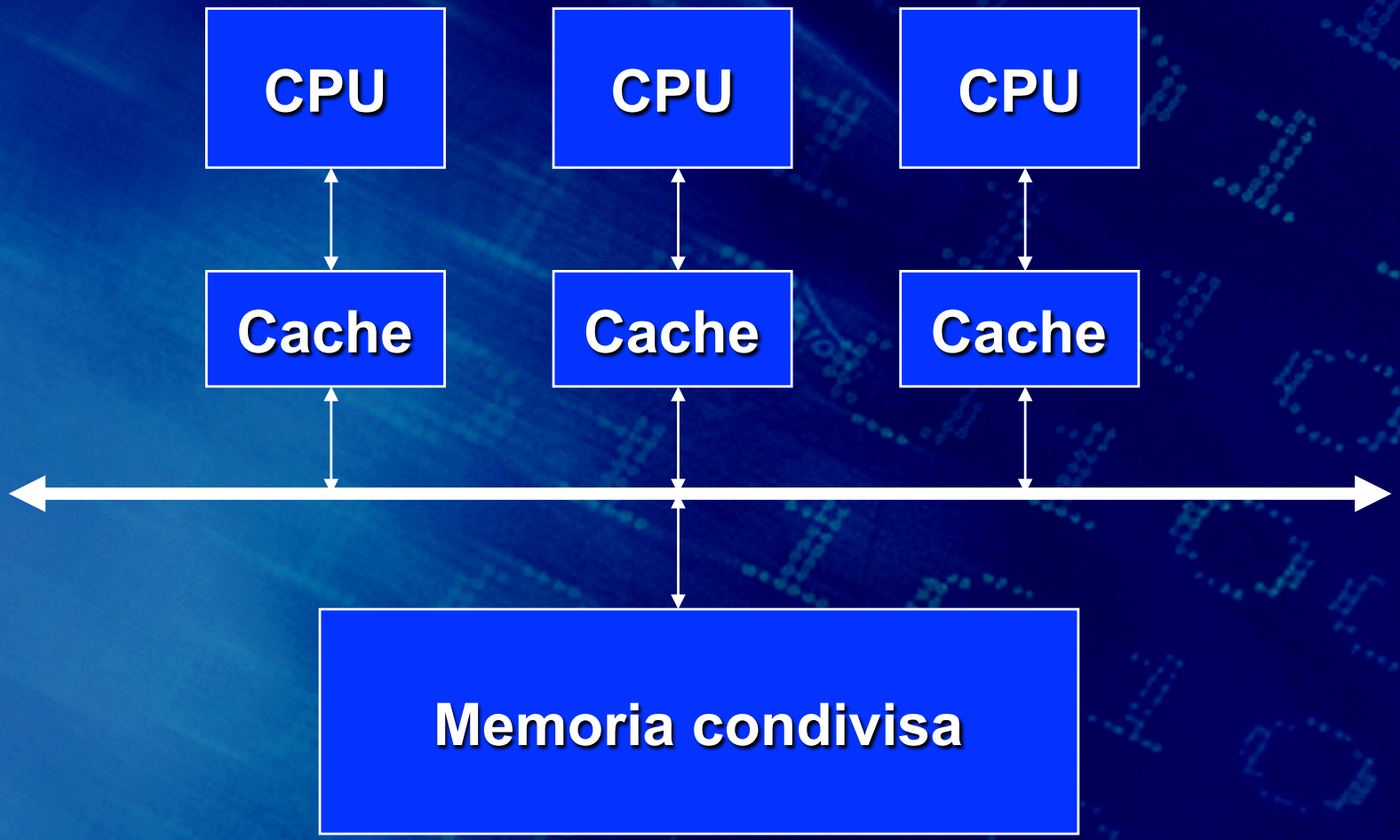
Outline

- Introduzione
- Il principio di località
- Struttura di una cache
- Parametri caratteristici
- Parametri fisici
- Funzione di traduzione (mapping)
- Algoritmi di sostituzione
- Aggiornamento della Memoria Principale
- Gestione della Coerenza
- Dimensionamento di una Cache
- Cache L1 e L2
- Esempi di cache.

Coerenza della cache

- È un problema nei sistemi multiprocessor con memoria condivisa, nei quali ogni processore ha una sua cache
- Problemi analoghi si possono avere se il sistema utilizza un DMA controller

Coerenza della cache



Bit di Validità

- **Per ottenere la coerenza delle cache si introduce per ogni linea di ogni cache un bit di validità**
- **Se è disattivato, significa che il blocco presente in quella linea ha un valore diverso dal corrispondente blocco nella memoria principale. In tal caso ogni accesso al blocco comporta un miss.**

Soluzioni

- **Nei sistemi multiprocessore si usa di solito il meccanismo di write-through.**

Soluzioni

- **Inoltre, per garantire la coerenza delle cache si possono usare le seguenti soluzioni:**
  1. ***Bus Watching con Write-through*: il controllore ci ciascuna cache rileva le operazioni di Write-through sul bus, e invalida (attraverso il bit di validità) le linee corrispondenti nella propria cache**
  2. ***Hardware Transparency*: ogni operazione di Write-through scatena automaticamente l'aggiornamento (flush) delle altre cache**
  3. ***Non-cacheable* Memory: la memoria condivisa non può essere trasferita nelle cache.**

Outline

- Introduzione
- Il principio di località
- Struttura di una cache
- Parametri caratteristici
- Parametri fisici
- Funzione di traduzione (mapping)
- Algoritmi di sostituzione
- Aggiornamento della Memoria Principale
- Gestione della Coerenza
- Dimensionamento di una Cache
- Cache L1 e L2
- Esempi di cache.

Dimensionamento di una Cache

- D = dimensione della cache
- l = lunghezza del blocco
- N = numero di set
- k = grado di associatività (numero di linee per set)

$$D = k * l * N$$

- I gradi di libertà nel progetto sono quindi 3:

$$l , k , N$$

Dimensionamento

- **Il progetto non prevede un'unica soluzione ma uno schema logico può essere il seguente:**
  - **L'andamento del miss-rate con la dimensione della cache è di tipo a ginocchio dove la posizione di quest'ultimo dipende dal carico e dal processore (in genere raddoppiando la dimensione i miss diminuiscono di circa il 30 %)**

Dimensionamento

- Fissata la dimensione D si sceglie un grado di associatività piccolo (k=2,4)
- Si provano successivamente soluzioni con differenti dimensioni di linea l variando proporzionalmente anche il numero di set N in modo da mantenere costante la dimensione scelta
- Si fissa infine il grado di associatività

Dimensioni dei Blocchi

- Al crescere delle dimensioni del blocco si verificano 2 fenomeni:
  - dapprima la hit ratio cresce
  - poi comincia a decrescere.
- Valori frequenti sono da 4 a 16 parole.

Prestazioni

- Si definiscano le seguenti grandezze:
  - h: hit ratio della cache
  - C: tempo di accesso alla cache
  - M: penalità di fallimento, ossia tempo di accesso in memoria quando il dato non è in cache. In generale dipende dalla dimensione dei blocchi.
- Il tempo medio di accesso in memoria per la CPU sarà

$$t_{medio} = h\,C + (1-h)\,M$$

- Valori normali per h sono dell'ordine di 0,9.

Outline

- Introduzione
- Il principio di località
- Struttura di una cache
- Parametri caratteristici
- Parametri fisici
- Funzione di traduzione (mapping)
- Algoritmi di sostituzione
- Aggiornamento della Memoria Principale
- Gestione della Coerenza
- Dimensionamento di una Cache
- Cache L1 e L2
- Esempi di cache.

Cache di primo e secondo livello

- Può essere conveniente avere due livelli di cache:
  - una cache di primo livello (L1), più piccola e veloce
  - una cache di secondo livello (L2), più lenta ma più grande.

Cache di primo e secondo livello

- **Aggiungendo una cache di secondo livello:**
  - **in caso di hit in L1, il tempo di accesso dipende solo dalla velocità di L1**
  - **in caso di miss in L1, il tempo di accesso**
    - **viene comunque ridotto se la parola si trova in L2**
    - **resta pari al tempo di accesso alla memoria principale in caso contrario.**

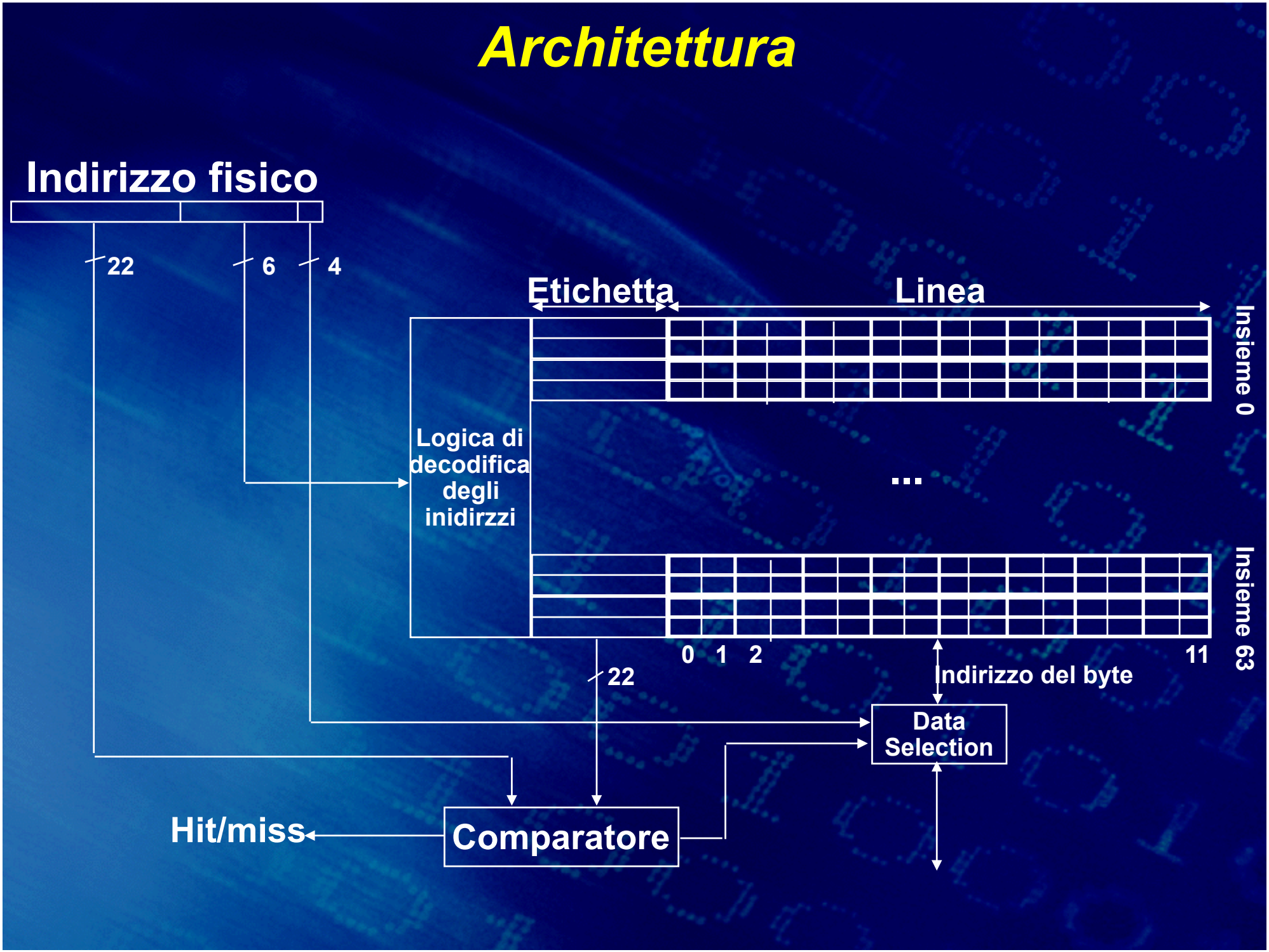
Outline

- Introduzione
- Il principio di località
- Struttura di una cache
- Parametri caratteristici
- Parametri fisici
- Funzione di traduzione (mapping)
- Algoritmi di sostituzione
- Aggiornamento della Memoria Principale
- Gestione della Coerenza
- Dimensionamento di una Cache
- Cache L1 e L2
- Esempi di cache

Esempio: Motorola 68000

- Caratteristiche:
  - due cache on-chip da 4 kB ciascuna, una per i dati e l'altra per le istruzioni
  - mapping di tipo set associative: la cache è divisa in 64 insiemi da 4 linee ciascuno; ogni linea contiene 4 long-word (16 byte)
  - 1 bit di validità per ogni blocco; 1 dirty bit per ogni long-word
  - meccanismo di rimpiazzamento all'interno dell'insieme: casuale
  - meccanismo di gestione delle operazioni di scrittura: write-through o write-back, selezionabile via software.

Architettura



Motorola 68000: Algoritmo per l'accesso alla memoria

- Attraverso i bit 4:9 si seleziona un insieme
- i bit 10:31 sono confrontati con quelli delle 4 linee appartenenti all'insieme selezionato
- se il blocco è presente nella cache si usano i bit 0:3 per selezionare il byte desiderato
- se il blocco non è presente, viene caricato dalla memoria principale al posto di uno dei 4 appartenenti all'insieme. La scelta fra i 4 è casuale.

Esempio: 80486

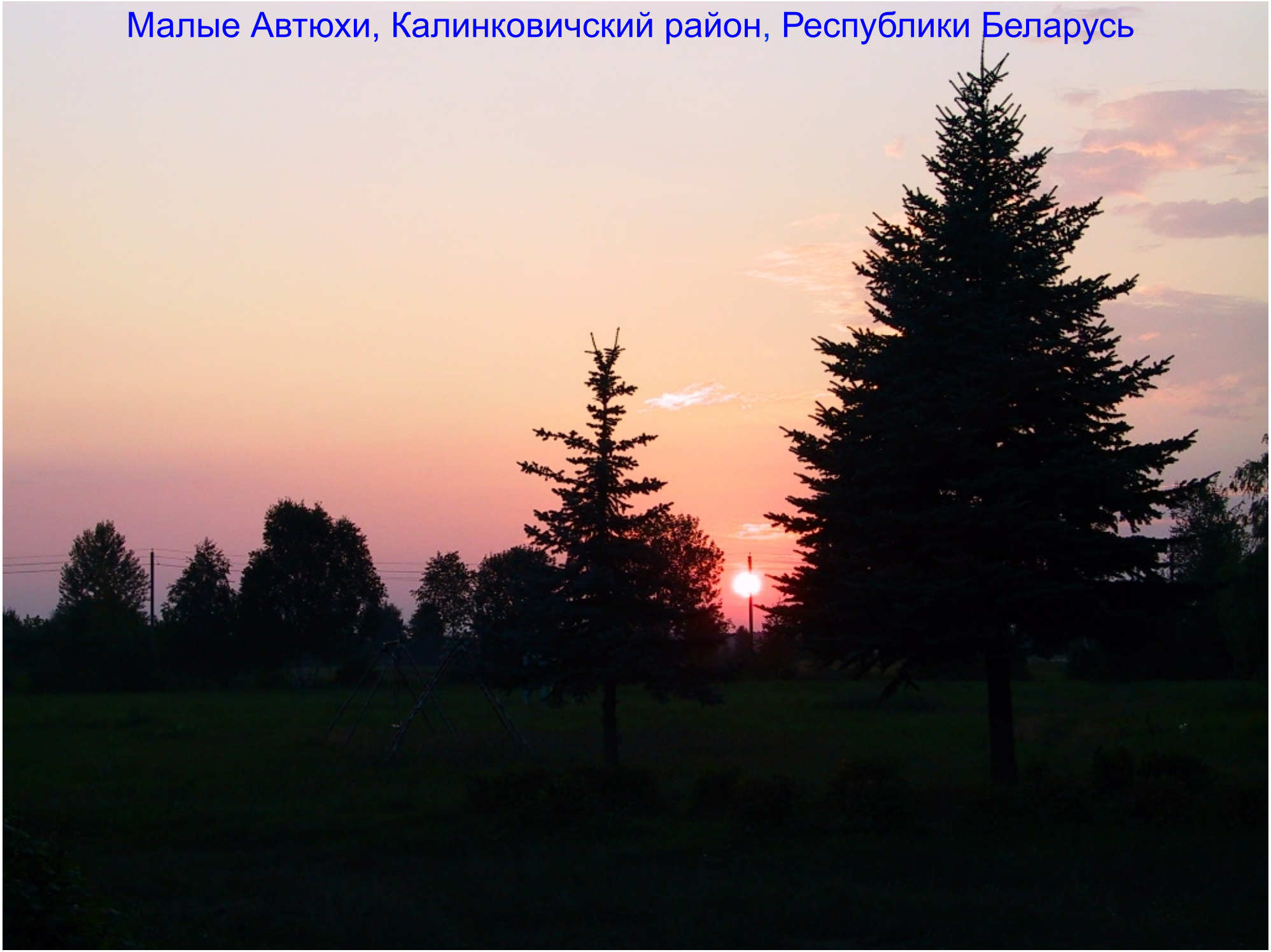
- L'80486 possiede una cache on-chip con le seguenti caratteristiche:
  - dimensione: 8 kB
  - dimensione dei blocchi: 16 byte
  - meccanismo di aggiornamento: write-through
  - organizzazione set associative, con 128 insiemi composti da 4 linee ciascuno
  - eventuale cache secondaria esterna
  - algoritmo di rimpiazzamento: pseudo-LRU
  - i risultati sperimentali indicano una hit-ratio del 96% per la generica applicazione DOS, e del 92% per le applicazioni UNIX e OS/2.

Gestione delle Cache

- L'80486 prevede un supporto per la gestione delle cache da parte del programmatore:
  - tra i bit di controllo scrivibili via software vi sono:
    - CD: abilita (CD=0) e disabilita (CD=1) l'uso della cache
    - NW: abilita (NW=0) e disabilita (NW=1) il meccanismo di write-through

Gestione delle Cache

- L'istruzione INVD causa lo scaricamento della cache sulla memoria esterna (cache flush), e segnala a eventuali cache esterne di fare altrettanto
- L'istruzione WBINVD ha la stessa funzione, ma segnala anche ad un'eventuale cache esterna di tipo write-back che prima di fare il flush deve eseguire la scrittura in memoria del proprio contenuto.


Малые Автюхи, Калинковичский район, Республики Беларусь