

# Le memorie cache

M. Sonza Reorda

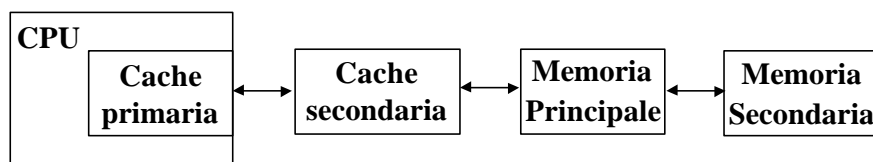
Politecnico di Torino  
Dip. di Automatica e Informatica

1

M. Sonza Reorda - a.a. 2001/2002

## Introduzione

Le memorie cache sono memorie di piccole dimensioni ma con elevata velocità interposte tra il processore e la memoria principale.



2

M. Sonza Reorda - a.a. 2001/2002

## Località dei riferimenti

La presenza di una cache può migliorare le prestazioni di un sistema per via della *località dei riferimenti* osservabile nella maggioranza dei programmi.

Si esprime in due forme:

- *località temporale*: se all'istante  $t$  il programma fa accesso ad una cella di memoria, è molto probabile che il programma faccia nuovamente accesso alla stessa cella entro l'istante  $t + D$
- *località spaziale*: 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  $X + e$ .

3

M. Sonza Reorda - a.a. 2001/2002

## Principio di funzionamento

Se all'istante  $t$  (primo accesso ad 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.

4

M. Sonza Reorda - a.a. 2001/2002

## Struttura di una cache

Una cache contiene al suo interno un certo numero di *linee*.

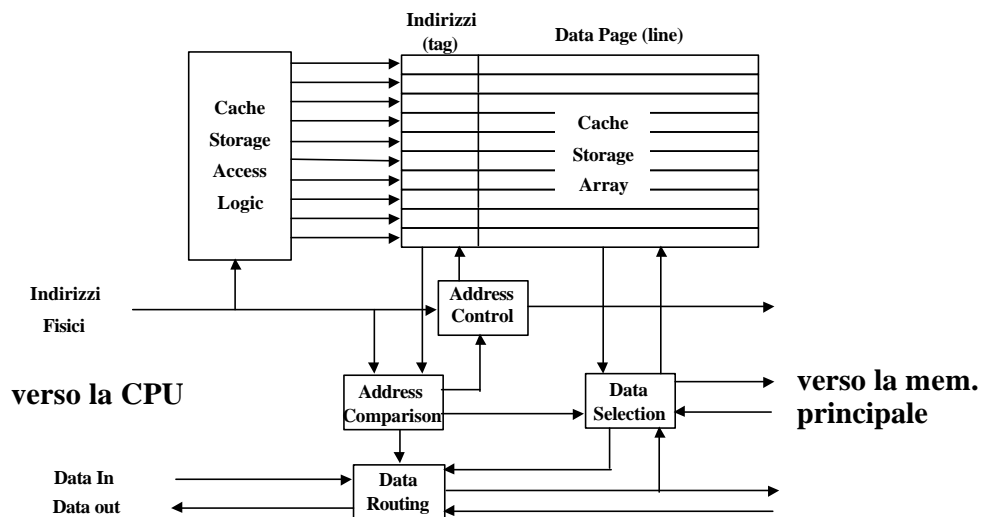
Una linea contiene un blocco di memoria. Ad ogni linea è associato un campo *tag*, che indica il blocco di memoria presente nella linea in quel momento.

La cache contiene inoltre la logica per intercettare gli indirizzi prodotti dal processore, controllare al proprio interno la presenza di un blocco, ed eventualmente caricarne uno nuovo da memoria.

5

M. Sonza Reorda - a.a. 2001/2002

## Struttura



6

M. Sonza Reorda - a.a. 2001/2002

## Funzionamento della cache

La cache si interpone tra processore e memoria principale.

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: estrae la parola dal blocco e la fornisce alla CPU al posto (e prima) della memoria principale (*hit*)
- se no: provvede a caricare nella cache l'intero blocco di cui la parola fa parte (*miss*).

7

M. Sonza Reorda - a.a. 2001/2002

## Prestazioni

In caso di hit, la cache riduce i tempi di accesso di un fattore tra 2 e 6 (indicativamente).

In caso di miss, la cache risponde solo dopo aver caricato dalla memoria principale il blocco mancante. Il tempo di accesso è quindi superiore a quello della memoria.

In alcuni casi, in presenza di miss la cache fornisce subito la parola richiesta proveniente dalla memoria, e poi provvede al caricamento del blocco (*load-through* o *early restart*). La tecnica richiede un maggior costo dell'hardware della cache.

8

M. Sonza Reorda - a.a. 2001/2002

## 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

In alcuni casi vi sono cache separate per dati e istruzioni, oppure ve ne è solo una per le 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.

## Parametri Caratteristici

**Sono:**

- **Dimensione della Cache**
- **Funzione di Mapping**
- **Algoritmo di Rimpiazzamento**
- **Metodo di Aggiornamento della Memoria Principale**
- **Dimensione dei Blocchi.**

11

M. Sonza Reorda - a.a. 2001/2002

## Dimensione della Cache

**La scelta della dimensione ottimale è influenzata da:**

- **costo**
- **prestazioni: al crescere delle dimensioni, le cache diventano più lente.**

**Dimensioni frequenti vanno da 1K a 128K parole.**

12

M. Sonza Reorda - a.a. 2001/2002

---

## Funzione di traduzione (mapping)

Definisce in quale linea della cache è eventualmente posizionato un certo blocco di memoria.

Si deve garantire (ad un prezzo accettabile) che la cache possa rapidamente verificare se contiene il dato corrispondente ad un certo indirizzo. Possibili soluzioni:

- memoria associativa (troppo costosa)
- scansione sequenziale (troppo lenta)

I meccanismi di mapping sono tre:

- *direct mapping*
- *associative mapping*
- *set associative mapping*.

13

M. Sonza Reorda - a.a. 2001/2002

## Direct Mapping

Ogni blocco  $i$  della memoria principale è messo in corrispondenza fissa con una linea  $k$  della cache.

La funzione di trasformazione è

$$k = i \bmod N$$

dove  $N$  è il numero di linee della cache.

Vantaggi:

- la funzione è facilmente implementabile in hardware

Svantaggi:

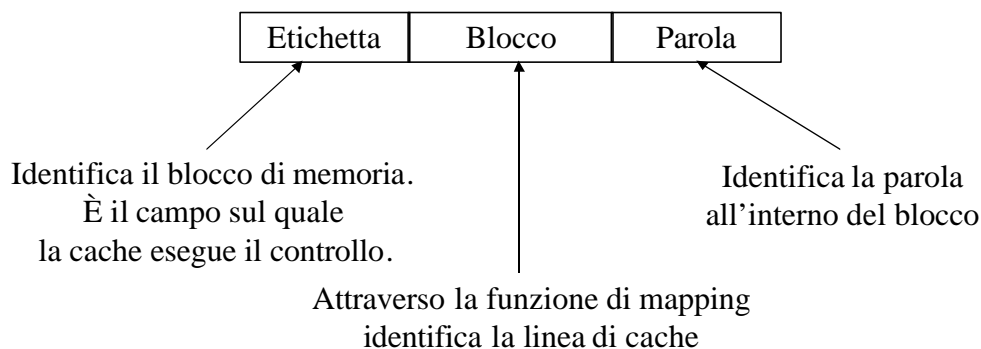
- se il programma fa accesso frequentemente a 2 blocchi corrispondenti alla stessa linea della cache, ad ogni accesso si deve trasferire uno dei 2 blocchi dalla memoria principale alla cache.

14

M. Sonza Reorda - a.a. 2001/2002

## Struttura dell'indirizzo

Nel caso di direct mapping, ciascun indirizzo viene scomposto dalla cache in tre parti:



15

M. Sonza Reorda - a.a. 2001/2002

## Associative Mapping

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

**Vantaggi:**

- massima flessibilità nella scelta del blocco di cache da usare

**Svantaggi:**

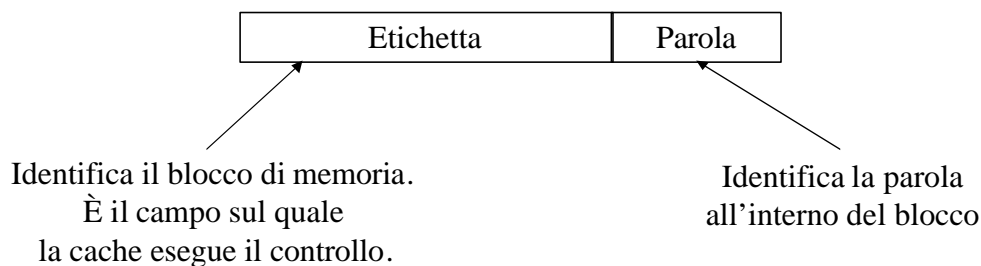
- complessità dell'hardware di ricerca (di solito si adotta una memoria associativa).

16

M. Sonza Reorda - a.a. 2001/2002

## Struttura dell'indirizzo

Nel caso di associative mapping, ciascun indirizzo viene scomposto dalla cache in due parti:



17

M. Sonza Reorda - a.a. 2001/2002

## Set Associative Mapping

**Caratteristiche:**

- i blocchi della memoria principale sono suddivisi in  $S_1$  insiemi di dimensione  $D_1$
- le linee della cache sono suddivise in  $S_2$  insiemi di dimensione  $D_2 \ll D_1$
- un blocco  $i$  appartiene all'insieme di linee  $k$  se  $k = i \bmod S_2$
- il blocco  $i$  può essere messo in una qualunque delle linee dell'insieme  $k$
- per accedere rapidamente ai blocchi in uno stesso insieme si usa una memoria associativa.

Se  $S_2 = 1$  si ha il *direct mapping*; se  $S_2 = N$  ( $N$  è la dimensione della cache) si ha l'*associative mapping*.

M. Sonza Reorda - a.a. 2001/2002

## Struttura dell'indirizzo

Nel caso di set associative mapping, ciascun indirizzo viene scomposto dalla cache in tre parti:



19

M. Sonza Reorda - a.a. 2001/2002

## Algoritmo di Rimpiazzamento

Nel caso in cui tutte le linee di cache in cui un blocco può venire memorizzato siano occupate, definisce quale deve venire utilizzata.

Viene scelto tra:

- **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.

20

M. Sonza Reorda - a.a. 2001/2002

## Aggiornamento della Memoria Principale

La CPU ha normalmente un canale di connessione diretto con la memoria principale; questo permette 2 possibili meccanismi di aggiornamento della memoria principale:

- *write-back*
- *write-through*.

21

M. Sonza Reorda - a.a. 2001/2002

## Write-Back

Per ogni blocco nella cache viene tenuto aggiornato un flag (*dirty bit*), che ricorda se il blocco è stato modificato da quando è stato caricato nella cache.

Quando un blocco viene eliminato dalla cache ed il dirty bit è settato, il blocco viene copiato dalla cache nella memoria principale.

**Svantaggi:**

- 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.

22

M. Sonza Reorda - a.a. 2001/2002

## Write-through

Ogni volta che la CPU esegue un'operazione di scrittura, la esegue sia sul dato nella cache che in quello 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.

23

M. Sonza Reorda - a.a. 2001/2002

## 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 4 o 8 parole.

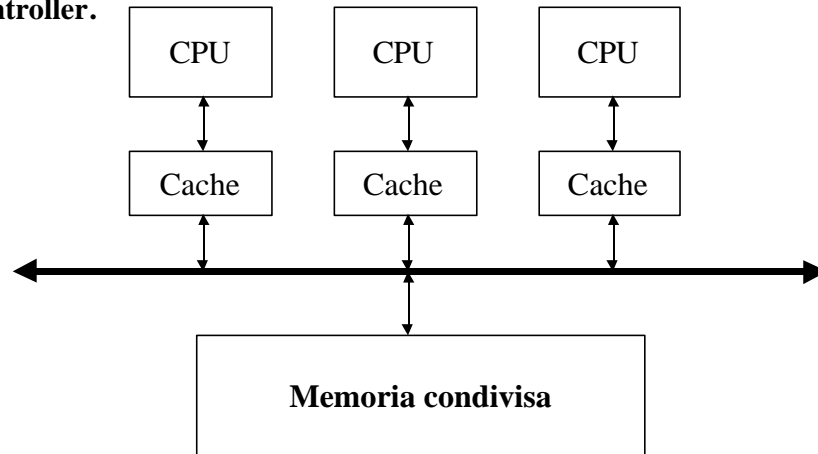
24

M. Sonza Reorda - a.a. 2001/2002

## Coerenza della cache

È un problema nei sistemi a multiprocessore con memoria condivisa, nei quali ogni processore ha una sua cache.

Problemi analoghi si possono avere se il sistema utilizza un DMA controller.



25

M. Sonza Reorda - a.a. 2001/2002

## 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.

26

M. Sonza Reorda - a.a. 2001/2002

---

## Soluzioni

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

Inoltre, per garantire la coerenza delle cache si possono usare le seguenti soluzioni:

- *Bus Watching con Write-through*: il controllore di ciascuna cache rileva le operazioni di Write-through sul bus, e invalida (attraverso il bit di validità) le linee corrispondenti nella propria cache;
- *Hardware Transparency*: ogni operazione di Write-through scatena automaticamente l'aggiornamento (flush) delle altre cache;
- *Non-cacheable Memory*: la memoria condivisa non può essere trasferita nelle cache.

27

M. Sonza Reorda - a.a. 2001/2002

## 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

$$t_{\text{medio}} = hC + (1-h)M$$

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

28

M. Sonza Reorda - a.a. 2001/2002

---

## Cache di primo e secondo livello

Può essere conveniente avere due livelli di cache:

- una cache di primo livello (CPL), più piccola e veloce
- una cache di secondo livello (CSL), più lenta ma più grande.

Aggiungendo una cache di secondo livello

- in caso di hit nella CPL il tempo di accesso dipende solo dalla velocità della CPL
- in caso di miss nella CPL il tempo di accesso
  - viene comunque ridotto se la parola si trova nella CSL
  - resta pari al tempo di accesso alla memoria principale in caso contrario.

29

M. Sonza Reorda - a.a. 2001/2002

## Esempio: Motorola 68000

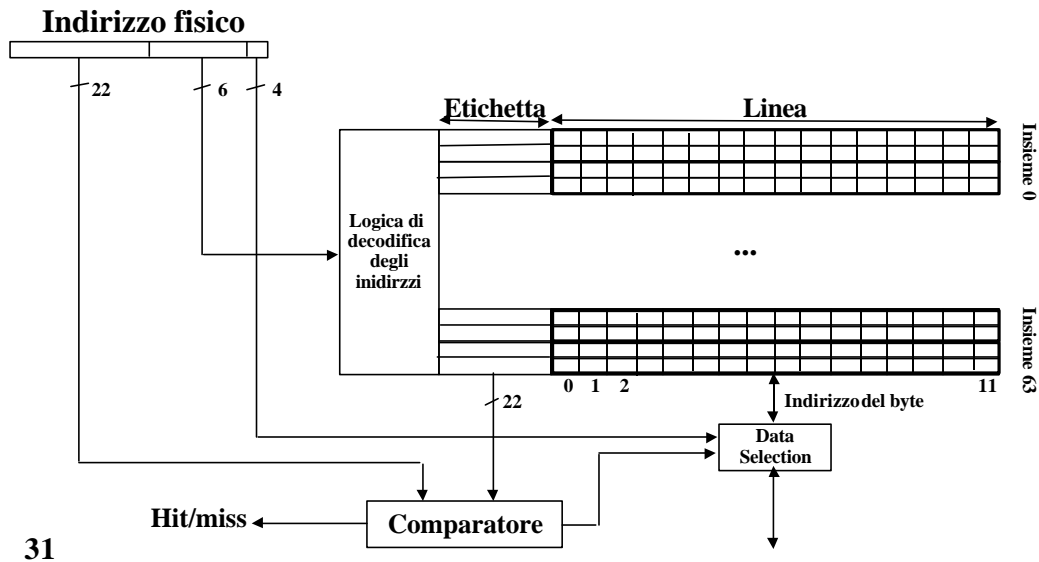
Caratteristiche:

- 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.

30

M. Sonza Reorda - a.a. 2001/2002

## Architettura



M. Sonza Reorda - a.a. 2001/2002

## 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.

32

M. Sonza Reorda - a.a. 2001/2002

---

## Esempio: 80486

L'80486 possiede una cache on-chip con le seguenti caratteristiche:

- **dimensione: 8Kbyte**
- **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.**

33

M. Sonza Reorda - a.a. 2001/2002

## Metodo pseudo-LRU

Per implementare una strategia LRU con 4 linee per insieme sarebbero richiesti 6 bit. Per risparmiare hardware, nel 486 vi sono solo 3 bit, e si implementa una strategia pseudo-LRU.

Ad ogni insieme (composto da 4 linee L0, L1, L2 e L3) sono associati 3 bit B0, B1 e B2.

Ogni volta che si fa accesso ad un insieme, i 3 bit vengono aggiornati secondo le seguenti regole:

- **accesso ad L0 o L1: B0  $\rightarrow$  1**
- **accesso ad L0: B1  $\rightarrow$  1**
- **accesso ad L1: B1  $\rightarrow$  0**
- **accesso ad L2 o L3: B0  $\rightarrow$  0**
- **accesso ad L2: B2  $\rightarrow$  1**
- **accesso ad L3: B2  $\rightarrow$  0.**

34

M. Sonza Reorda - a.a. 2001/2002

---

## Metodo pseudo-LRU (II)

All'atto di caricare un blocco in memoria, la scelta su dove metterlo nella cache viene fatta in base al seguente processo di decisione:

- se c'è una linea nell'insieme che non è valida, il nuovo blocco viene inserito al suo posto; altrimenti
- si testa B0: se ne deduce se l'ultima linea utilizzata è stata L0 o L1, oppure L2 o L3
- si testa B1:
  - se B0=0 e B1=0 si rimpiazza L0
  - se B0=0 e B1=1 si rimpiazza L1
  - se B0=1 e B2=0 si rimpiazza L2
  - se B0=1 e B2=1 si rimpiazza L3.

35

M. Sonza Reorda - a.a. 2001/2002

## 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
- l'istruzione INVD causa lo scaricamento della cache sulla memoria esterna (*cache flush*), e segnala ad 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.

36

M. Sonza Reorda - a.a. 2001/2002

---

## Esempio di calcolo: specifiche

Si consideri una cache con le seguenti caratteristiche:

- 64KByte di dati
- meccanismo di direct mapping
- blocchi di 4 byte
- indirizzi su 32 bit.

Si vuole determinare la struttura della cache (numero dei blocchi, dimensione del campo tag).

37

M. Sonza Reorda - a.a. 2001/2002

## Esempio: struttura della cache

I blocchi sono in numero pari a  $64\text{KByte}/4=16\text{K}=2^{14}$ .

Il campo tag deve poter identificare il blocco presente nella linea. La memoria è composta da  $2^{30}$  blocchi. Ogni linea corrisponde quindi a  $2^{16}$  blocchi di memoria. Il tag ha quindi dimensione pari a 16 bit.

La cache ha quindi le seguenti dimensioni:

$$2^{14} \cdot (32 + 16) = 2^{14} \cdot 48 = 768\text{Kbit} = 96\text{KByte}$$

38

M. Sonza Reorda - a.a. 2001/2002