


# Le memorie Cache associative

Prof. Alberto Borghese
Dipartimento di Informatica
alberto.borghese@unimi.it

Università degli Studi di Milano

Riferimento Patterson: 5.3, 5.4, 5.8




# Sommario

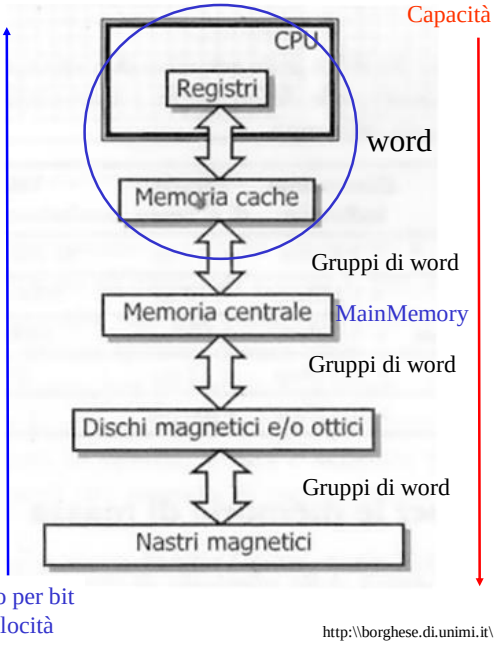
# Gerarchia di memorie

Livelli multipli di memorie con diverse dimensioni e velocità.

***Nel livello superiore troviamo un sottoinsieme dei dati del livello inferiore***.

***Ciascun livello vede il livello inferiore e viceversa.***

*Cache (memoria nascosta)*

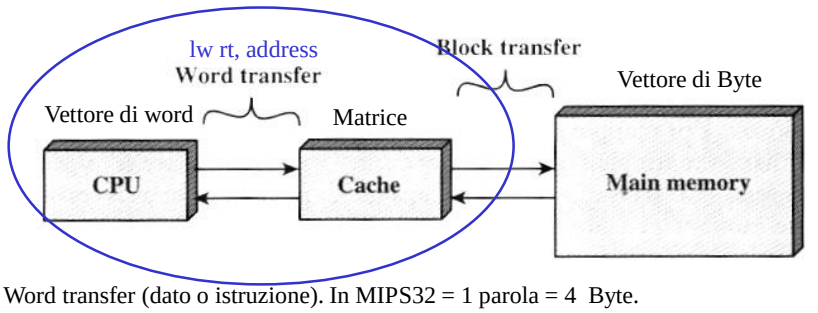





# Principio di funzionamento di una cache

**Scopo:** fornire alla CPU una velocità di trasferimento pari a quella della memoria più veloce con una capacità pari a quella della memoria più grande.

Una cache "disaccoppia" i dati utilizzati dal processore da quelli letti/scritti nella Memoria Principale.




Word transfer (dato o istruzione). In MIPS32 = 1 parola = 4 Byte.
Block transfer (più parole consecutive in MM – micro-blocco)

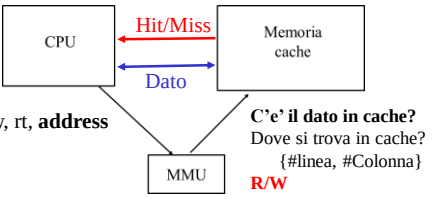
**La cache contiene una copia di parte del contenuto della memoria principale.**

# MMU, CPU e Cache

La MMU porta nella cache primaria i dati richiesti mentre il binomio processore-memoria sta lavorando.

1) Controlla se una parola è in cache (Hit).
2) Se si verifica una miss, porta una parola (e quelle vicine) in cache, prelevandole dal livello inferiore della gerachia.

- **Linea di cache** = 4 word (16 Byte)
- **Capacità cache** = 128 Byte (32 word)




lw $t0, 220($zero)

$204_{10}$ = 0000 0000 0000 0000 0000 0000 1101 1100$_2$

E gli altri bit?

#linea (3) #colonna (2)



# Determinazione di #linea, #colonna

La cache con linee di **ampiezza pari a 4 parole** (**micro-blocco** = 4 parole = 16 Byte) da 32 bit (**4 byte**), e **altezza di 8 linee**:
Il micro-blocco di dati della memoria principale che può essere contenuto in ogni linea di cache, ha dimensioni dim_linea = 4 parole * **4 byte** = 16 byte.
La capacità della cache sarà C = **8 linee** * dim_linea = **8** * 16 = 128 Byte (**macro-blocco** di MM).

lw $t0, 220($zero)
Indirizzo_cache = Indirizzo_Memoria principale *modulo* dimensione_macro-blocco_MM
220 / 128 Byte = **1** → mappiamo il 2° macro-blocco di MM sulla cache.
resto = 92 è l'indirizzo all'interno della cache (da trasformare in #linea, #colonna)

Numero linea = resto *modulo* dim_linea (capacità_micro-blocco)
92 / 16 = 5 → La word è contenuta nella linea #5, => 6ª linea della cache.
resto = 12 è l'indirizzo all'interno della linea della cache (numero di byte nella linea, da trasformare in #colonna)

12 / 4 = 3 → La word è la 4ª parola nella 6ª linea di cache. Resto = 0 è il numero di byte intra-word.

Il dato viene letto (trasferito nella CPU) assieme ai byte 221, 222, 223 della stessa 6ª linea della cache.

NB 220 / 16 = 13 → Riempio interamente 8 linee di una prima cache virtuale + 5 linee della cache reale.

**Le diverse linee possono provenire da macro-blocchi diversi della MM.**

# Determinazione della posizione mediante shift

……..0000 0000 1101 11(00)  lw $t0, 220($zero)  220=128+64+16+8+4

Cache di 8 linee x 4 parole di 4 Byte ciascuna => Capacità della cache = 8x4x4 Byte = 128 Byte
**$\log_2$ 128 = 7 Numero di bit per indirizzare la cache.**

Indirizzo / capacità_cache (dimensione macro-blocco) 220 / 128 = 1 **TAG**

**R = 92 (indirizzo intra-cache)**
Resto / dim_linea (capacità micro-blocco) 92 / 16 = 5 **#Linea**
**$\log_2$ 8 = 3 Numero di bit per indirizzare la linea (indice)**
I 3 bit più significativi dell'indirizzo intra-cache indicano il numero di linea della cache indirizzata (per lettura / scrittura)

**R = 12 (numero di byte nella linea)**
Resto / dimensione della word (numero di Byte per word) 12 / 4 = 3 **#Numero word**
$\log_2$ 4 = 2 Numero di bit per indirizzare la parola nella linea
Questi bit indicano il numero di colonna (parola) all'interno della linea della cache.

I 25 bit in verde identificano il campo TAG. E' il numero di macro-blocco di MM associato all'indirizzo 220.



# Cosa rappresentano gli altri bit?

……..0000 0000 1101 11(00)  lw $t0, 220($zero)  220=128+64+16+8+4

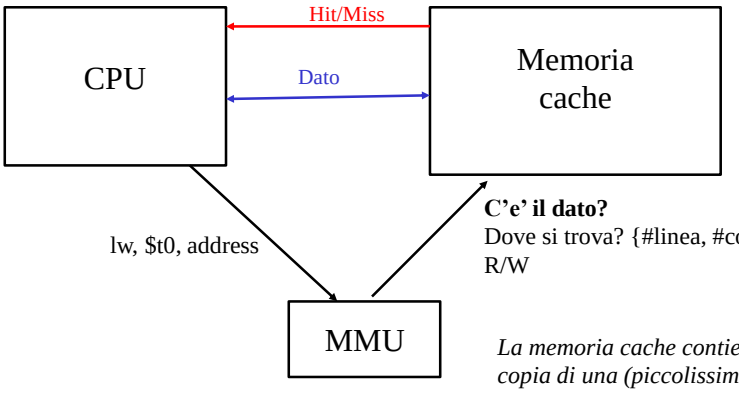
I 25 leading bit in verde identificano il campo **TAG**. E' il numero di macro-blocco di MM associato all'indirizzo 220.

I 2 bit meno significativi identificano i byte intra-word, e non interessano perché il trasferimento da cache a CPU è di parole intere.

lw $t0, 220($zero)
- Numero di linea = 5
- Numero di colonna = 3

Questa coppia, {numero linea, numero di colonna} è comune a tanti indirizzi, e.g.
- lw $t0 92($zero) - TAG = 0
- lw $t0, 348($zero) – TAG = 2
- lw $t0, 476($zero) – TAG = 3

# Le domande alla MMU sulla memoria cache




*La memoria cache contiene una copia di una (piccolissima) parte dei dati della memoria principale.*

Parte dati.
Selezione del dato gerarchica: {#linea, #colonna (word)}
Schema a 2 livelli (selezione della linea – cf. Register File + selezione della colonna)

Parte di controllo.
Hit/miss



# Come si può sapere se un dato è presente in cache?

Aggiungiamo a ciascuna delle linee della cache un campo **TAG**.

Il tag contiene i bit che costituiscono la parte più significativa dell'indirizzo e rappresenta il numero di macro-blocco di MM associato ai dati contenuti nella linea di cache (tutti i dati su una linea hanno lo stesso TAG). Il TAG è costituito da K bit:

$$K = M - \sup(\log_2 \text{Capacità\_cache (Byte)})$$

Nell'esempio precedente: $K = 32 - \sup(\log_2 128) = 25$ bit.
I bit per l'indirizzamento intra-cache sono 32-25 = 7 bit (128 Byte)

lw $t0, 220($zero)
$220_{10}$ = 0000 0000 0000 0000 0000 0000 1101 1100

Macro-blocco MM = TAG - etichetta associata a una linea

# Come si può sapere se un dato è presente in cache?

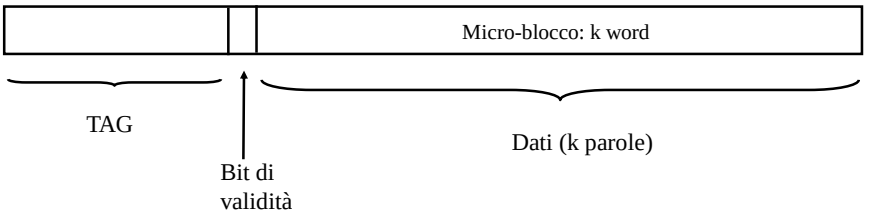
E all'accensione?

Come possiamo sapere che nessun dato e' ancora stato caricato in cache?

Viene introdotto un bit di validità, associate a ogni linea. La validità è della linea intera.

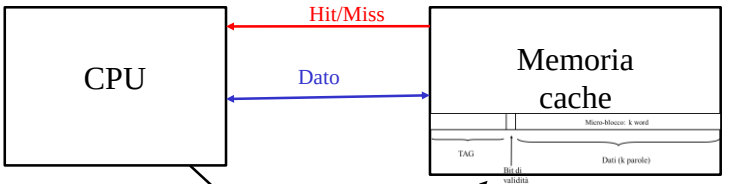


# Struttura di una linea di cache




Nel caso precedente, avremo linee di cache di lunghezza:
25 (lunghezza_campo_TAG) + 1 (bit di validità) + 4 (parole) * 4 (Byte/parola) * 8 (bit/Byte) = 156 bit.

# Le domande alle memorie cache




lw rt, address

**C'e' il dato?**
Dove si trova? {#linea, #colonna}
R/W

MMU

*La memoria cache contiene una copia di una (piccolissima) parte dei dati della memoria principale.*

lw $t0, 220($zero)
$220_{10}$ = 0000 0000 0000 0000 0000 0000 1 101 $1100_2$

Hit in cache se: "TAG_address = TAG_lineaCache & validità_lineaCache == 1"



# Sommario

Principio di funzionamento di una cache

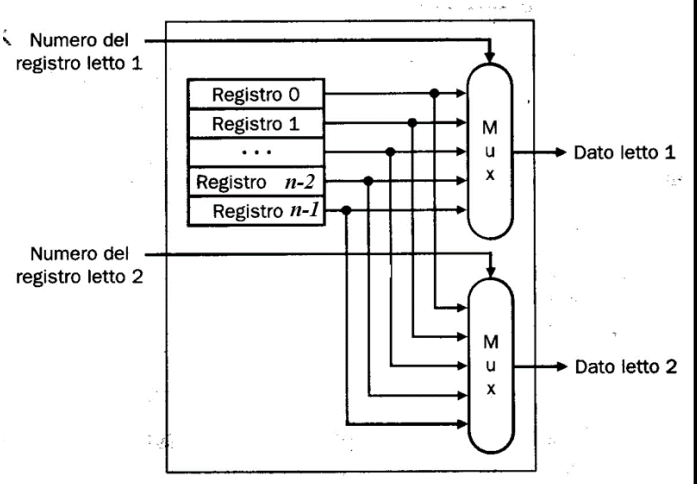
Circuito di lettura / scrittura di una cache a mappatura diretta

Cache associative

Politiche di sostituzione

# Porta di lettura di un dato (ampiezza 1 parola)

**Register file**
Pochi registri,
2 porte di lettura

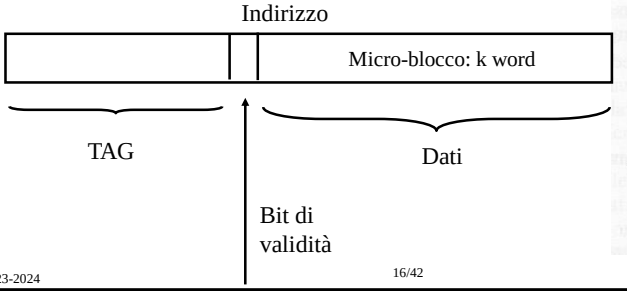


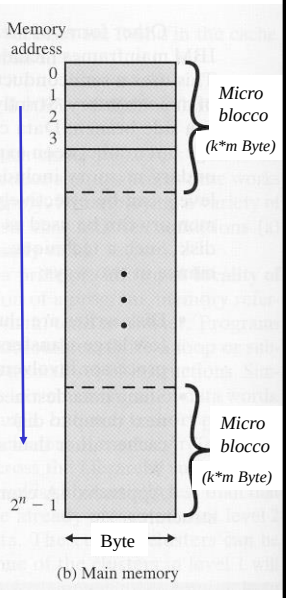

Cache: numero maggiore di celle di memoria. 1 sola porta di lettura. **Accesso a 2 livelli.**



# Come leggere dalla cache

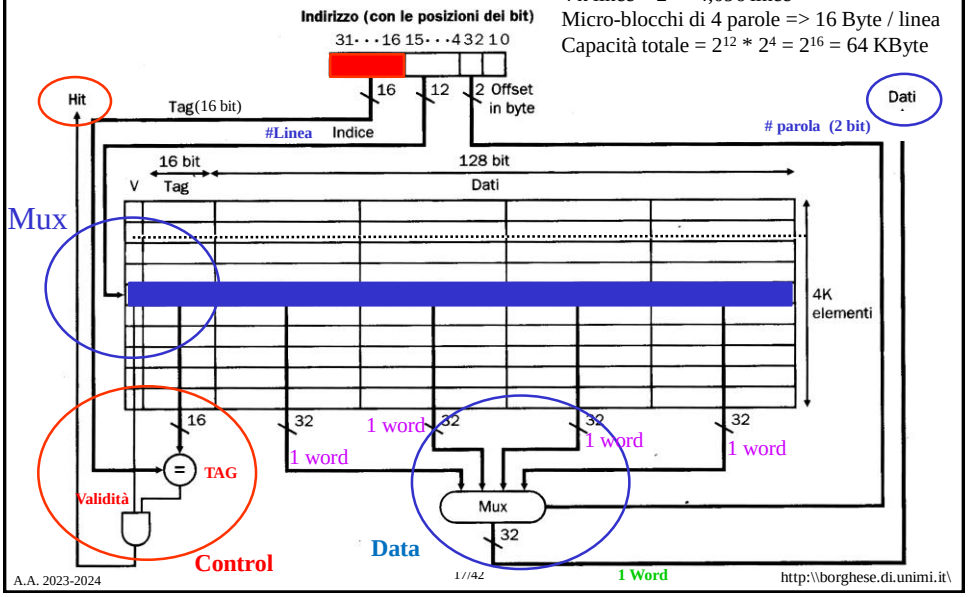
1) Individuare la linea della cache dalla quale leggere. Operazione analoga all'indirizzamento del register file.
2) Leggere la linea.
3) Confrontare il campo **tag dell'indirizzo** con il **campo tag della linea di cache** (tag è il numero di macro-blocco di MM)
4) Controllare il **bit di validità della linea di cache**.
5) Rispondere con hit/miss
6) *Selezionare la parola all'interno della linea* (per linee più ampie di una parola, occorre individuare una delle parola tra le k presenti nella linea di cache – micro-blocco).



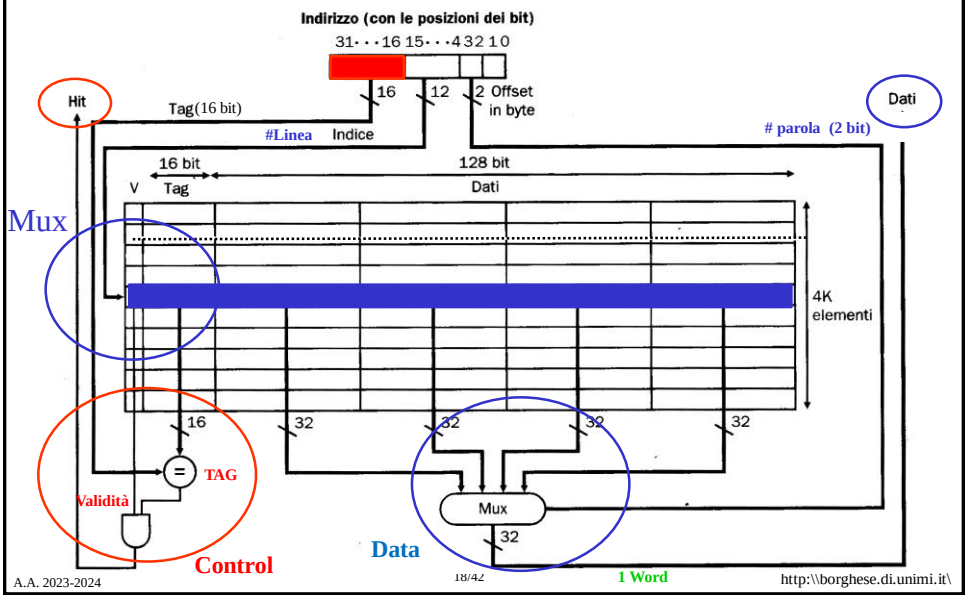
# Porta di lettura della cache, blocchi > 1 word

Parole di 32 bit (4 byte)
4 k linee – $2^{12}$ =4,096 linee
Micro-blocchi di 4 parole => 16 Byte / linea
Capacità totale = $2^{12} * 2^4 = 2^{16}$ = 64 KByte



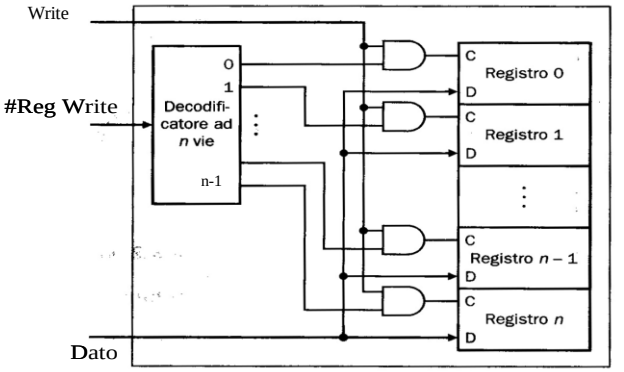



# Porta di scrittura nella cache, blocchi > 1 word?



# Porta di scrittura di una cache

Selezione della linea e poi della parola in scrittura.
Invio del dato (parola) a tutte le parole di tutte le linee della cache






# Sommario

Principio di funzionamento di una cache

Circuito di lettura / scrittura di una cache a mappatura diretta

Cache associative

Politiche di sostituzione

# Problemi con le cache a mappatura diretta

- Riempimento non ottimale (a macchia di leopardo): sostituzione del contenuto dell'intera linea della cache ogni volta che scrivo un singolo dato che appartiene a un blocco diverso di MM (anche nel caso di cache quasi vuota).
- Memoria associativa: il contenuto viene recuperato fornendo degli elementi dei dati, parte del contenuto, detti chiavi (e.g. ricerca nei data-base, ricerca attraverso ontologie WEB).
- Nelle memorie associative delle architetture si utilizza una parte dell'indirizzo come chiave, per recuperare il dato. Viene recuperato il dato che ha quella particolare chiave.

Occorre quindi sostituire il meccanismo di accesso diretto (parte dell'indirizzo -> numero di linea) con un meccanismo associativo che identifichi la linea associata alla chiave (parte dell'indirizzo = chiave di ricerca della linea).

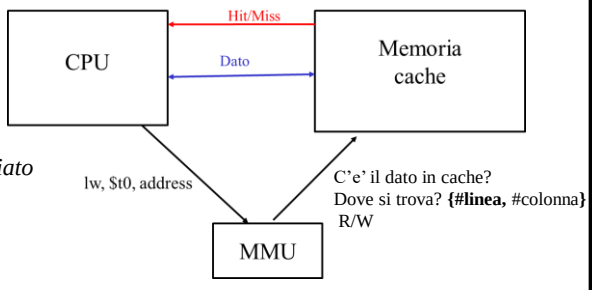
Occorre provare la chiave fornita dall'indirizzo su tutte le "serrature" della cache.

Non esiste più un numero d'ordine delle linee, una numerazione.



# Meccanismo di accesso

lw $t0, 220($zero)

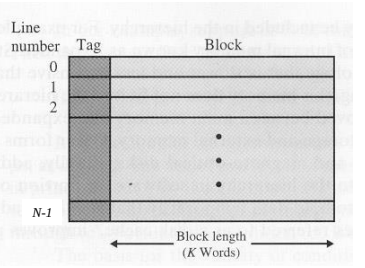



**Come ricavo il #linea?**

*Identifico la linea, confrontando la **chiave** contenuta nell'indirizzo inviato dalla CPU con la chiave di tutte le linee di cache.*

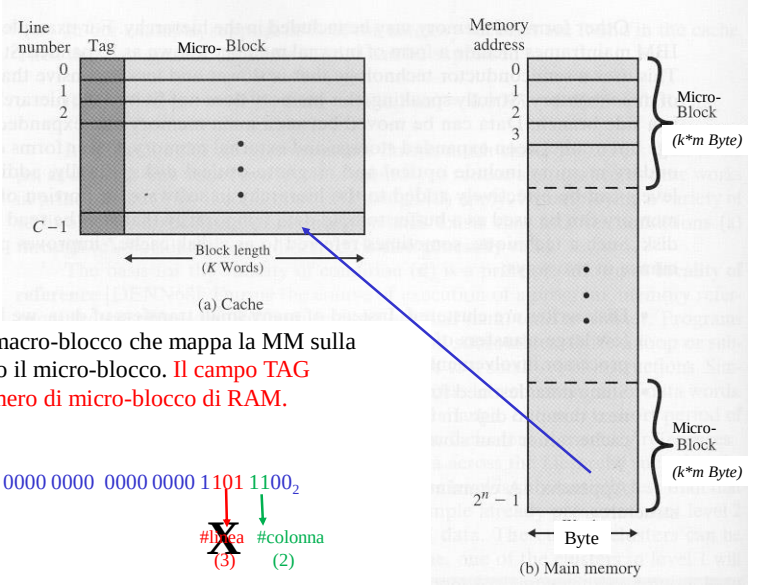
Come ricavo la chiave?

Come ricavo il numero di parola (word) nella linea?



# Memoria associativa

Numero linee: 8
Block length: 4




Non esiste più il macro-blocco che mappa la MM sulla cache, rimane solo il micro-blocco. Il campo TAG rappresenta il numero di micro-blocco di RAM.

lw $t0, 220($zero)
$220_{10}$ = 0000 0000 0000 0000 0000 0000 1101 1100$_2$

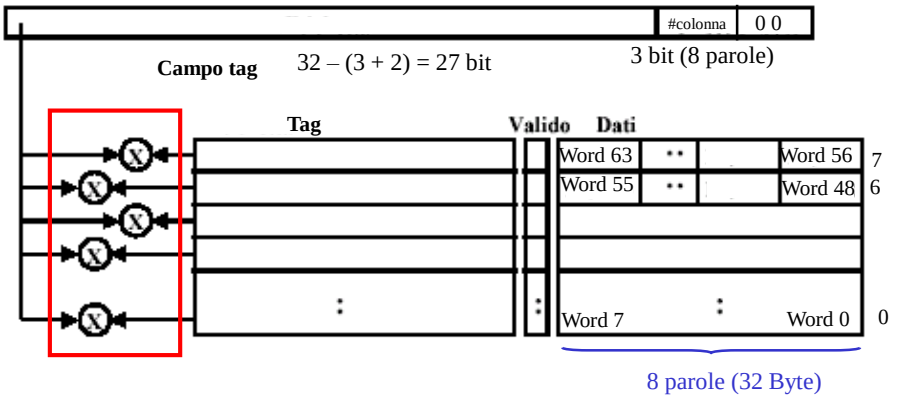
#linea (3) #colonna (2)

Non posso più ricavare il numero di linea direttamente dall'indirizzo. **Posso utilizzare il campo TAG come chiave.**



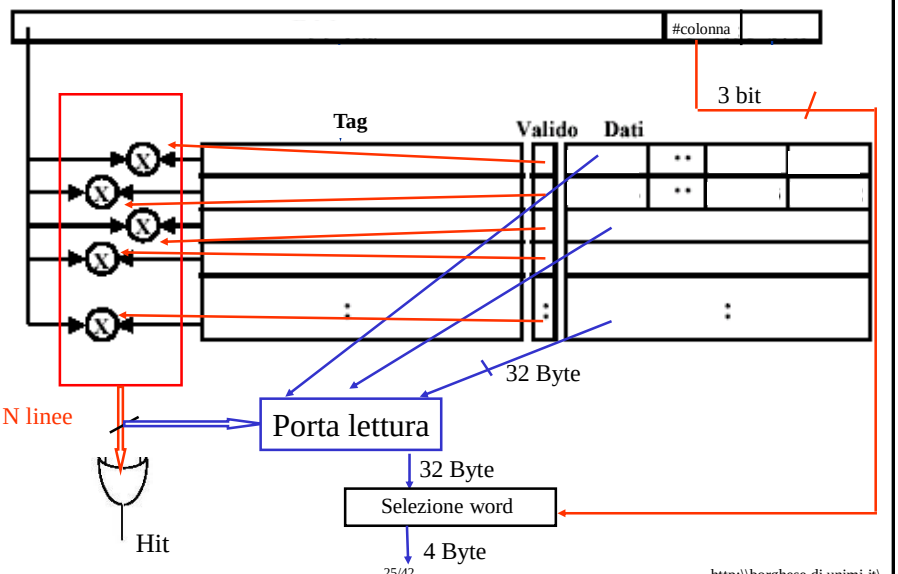
# Memorie associative

Memoria associativa con linee di **8 parole** => Dimensione della linea: 8 x 4 Byte = 32 Byte



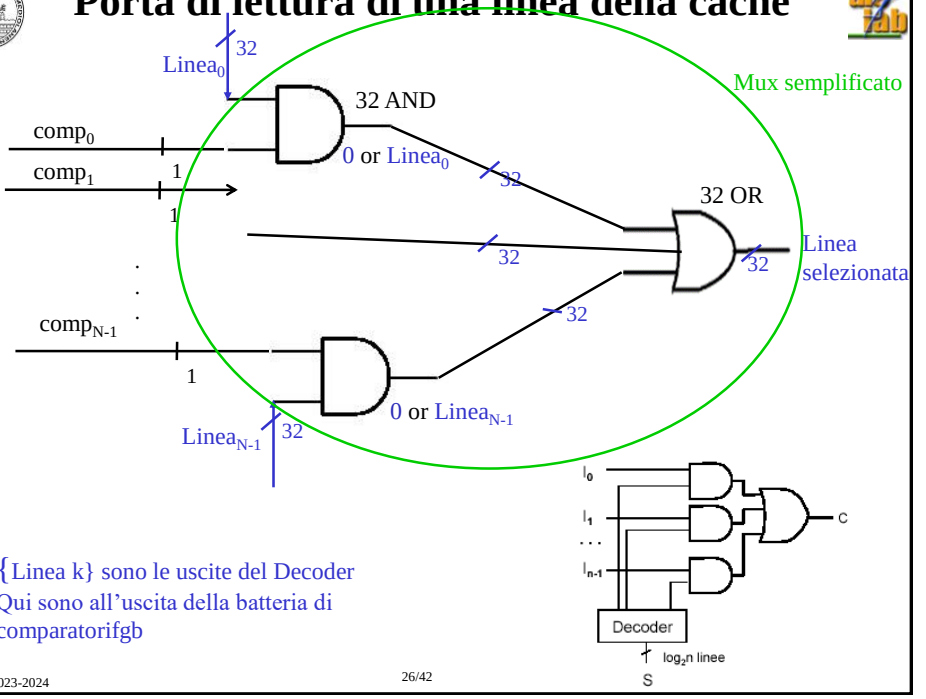

Numero di linee = 16 (la dimensione del campo TAG non dipende dal numero di line)
Capacità della cache: 32 x Numero_linee = ….
Il numero di linea non entra nell'indirizzamento
Consentono di caricare un blocco di Memoria Principale in una qualsiasi linea di cache.
E' una memoria completamente associativa.
Tramite una schiera di comparatori individuo in quale linea si trova il mio dato.

# Lettura di una memoria associativa





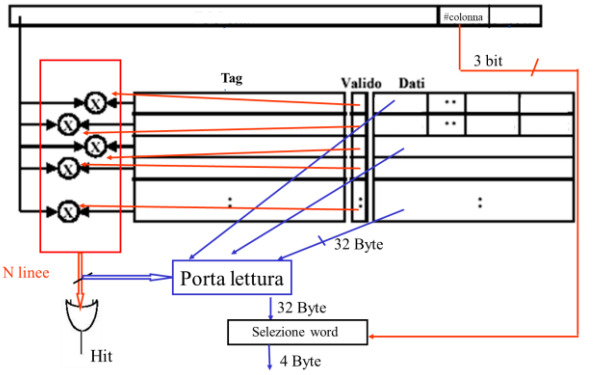

# Porta di lettura di una linea della cache




{Linea k} sono le uscite del Decoder
Qui sono all'uscita della batteria di comparatorifgb

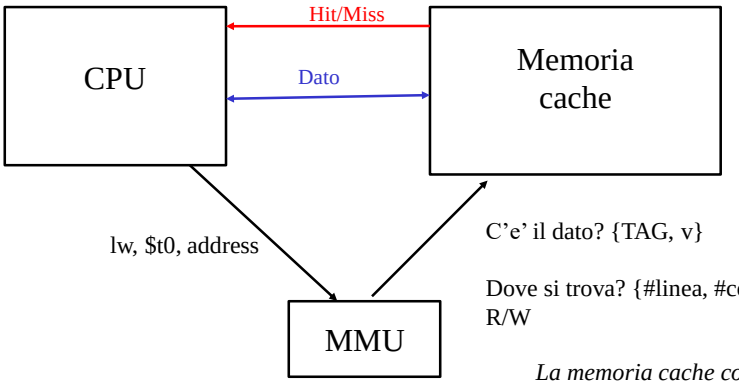
# Alcuni dettagli

- L'uscita del comparatore di una linea va in AND con il bit di validità di quella linea. Il segnale diventa quindi = 1 quando il dato è presente sulla linea (ha lo stesso TAG) ed è valido.
- Le uscite dagli N comparatori, ciascuno associato ad una linea diversa, possono avere al massimo un "1" -> esiste un'unica copia del dato della MM






# Le domande alla MMU sulla memoria cache




C'e' il dato? {TAG, v}

Dove si trova? {#linea, #colonna}
R/W

*La memoria cache contiene una copia di una (piccolissima) parte dei dati della memoria principale.*

Parte dati:
Schema a 2 livelli (selezione della linea mediante chiave + selezione della colonna mediante indice)
Parte di selezione del dato gerarchica: {#linea, #word (colonna)}

Parte di controllo: {TAG,v}

# Memorie n-associative

n-associative o set associative o a n vie.

La memoria è suddivisa in n insiemi, o banchi, ciascuno di k linee, posti in parallelo.

***Cache:*** *è* l'insieme dei banchi più i circuiti che li gestiscono.

***Insieme (banco):*** cache elementare ad accesso diretto.

***Capacità della cache:*** #parole = #banchi * (#linee / banco) * (#parole / linea).

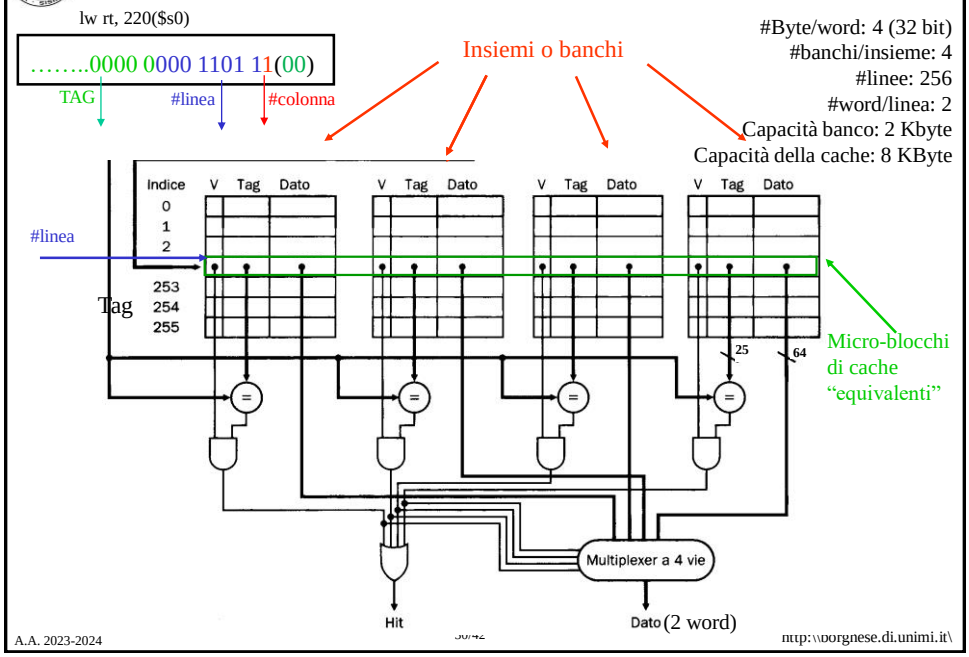
***Micro-blocco (linea di cache):*** #parole (byte) della MM adiacenti, contenute in una linea di uno dei banchi della cache.

La corrispondenza tra Memoria Principale e linea di un banco è a mappatura diretta.
La corrispondenza tra Memoria Principale e i diversi banchi è associativa.

Associatività. Per cercare un dato non devo più analizzare tutte le linee di una cache, ma un'unica linea, ma devo analizzare quella linea sui diversi banchi.

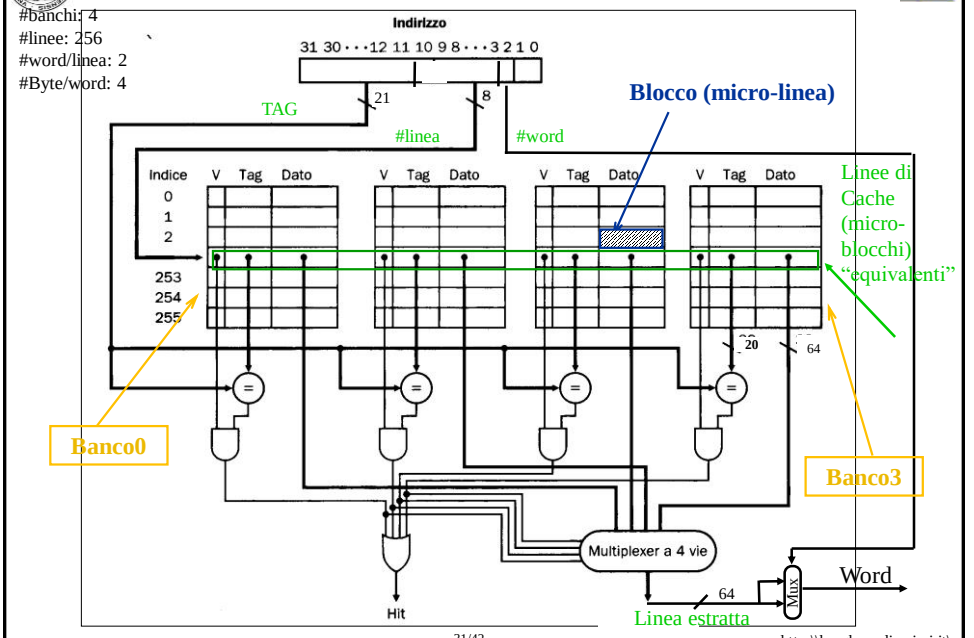


# Memorie n-associative: struttura



# Memorie n-associative: struttura completa

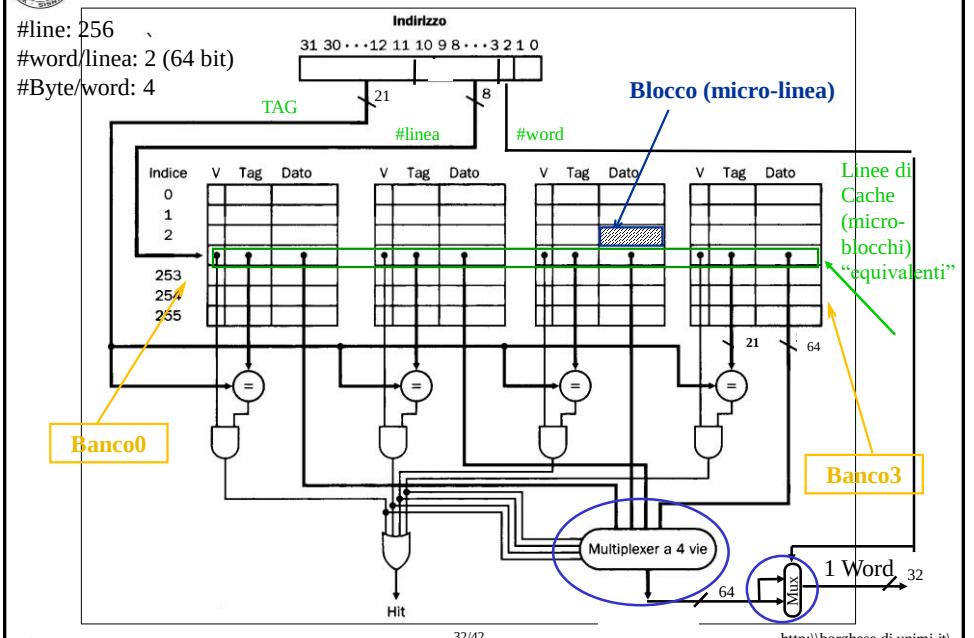
#banchi: 4
#linee: 256
#word/linea: 2
#Byte/word: 4





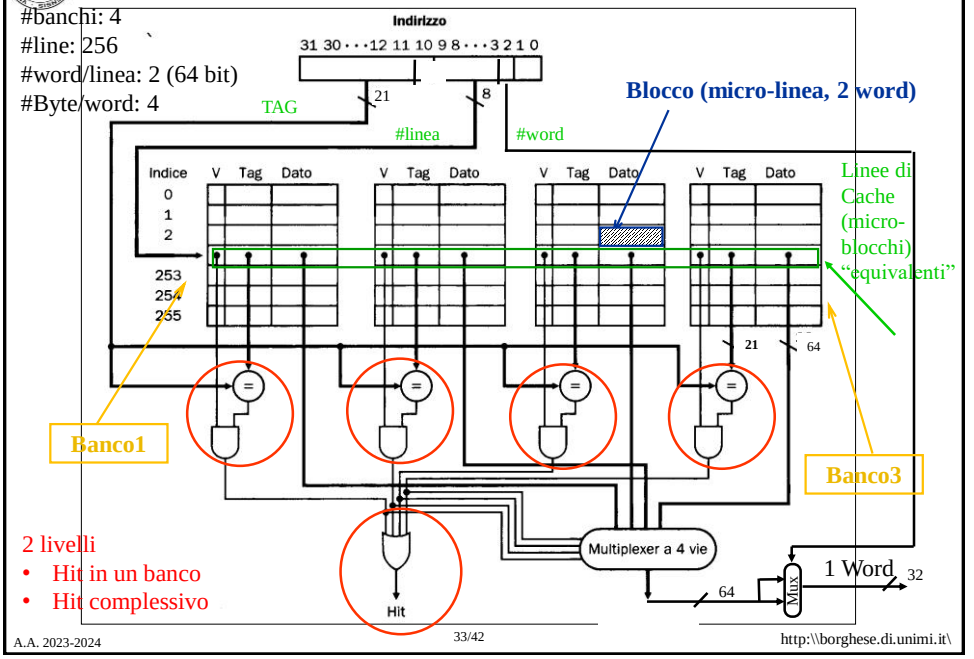

# Memorie set-associative: data-path

#line: 256
#word/linea: 2 (64 bit)
#Byte/word: 4



# Memorie set-associative: control path

#banchi: 4
#line: 256
#word/linea: 2 (64 bit)
#Byte/word: 4




2 livelli
- Hit in un banco
- Hit complessivo



# Accesso a cache ad n-vie

1) **INDICE**. Se la parola richiesta è memorizzata in cache, si trova in una particolare linea di uno dei banchi. Questa linea è individuata dall'**indice (di linea)**.

2) **NUMERO COLONNA.** Estrae la parola dalla linea.

**3) TAG** – contiene numero del macro-blocco della MM a cui appartiene il dato. Cerca il tag dell'indirizzo all'interno del **TAG della linea** identificata a mappatura diretta, nei **diversi banchi**.

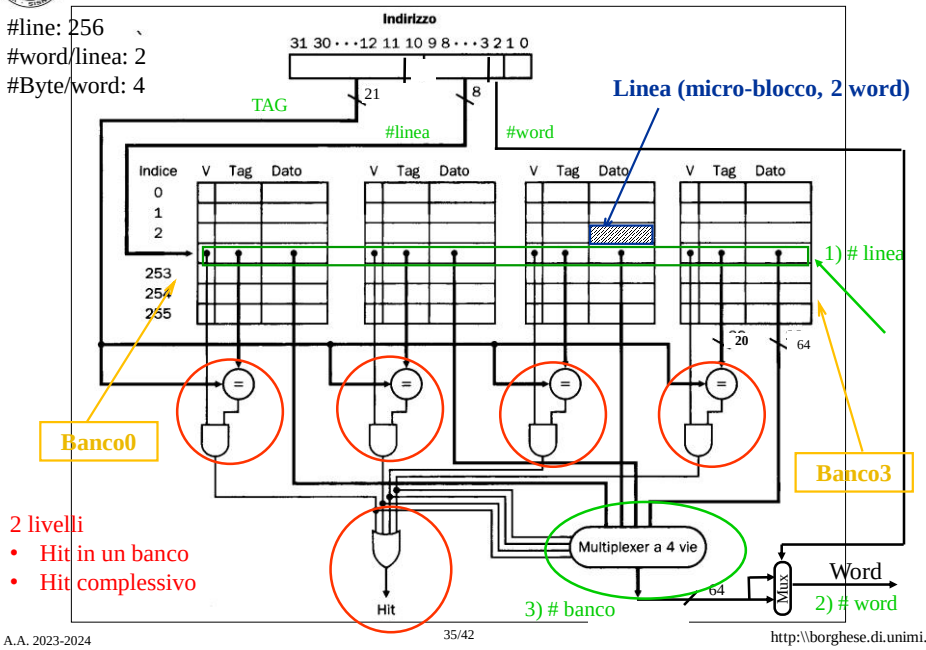
Indice (numero di linea) e colonna (numero di parola) sono equivalenti alla mappatura diretta.

Qui si introduce un terzo livello di indirizzamento attraverso **l'associatività tra banchi** (accesso per chiave).

L'insieme dei segnali di HIT pilotano anche il «MUX» che trasferisce in uscita il contenuto del banco opportuno della cache.

# Memorie set-associative

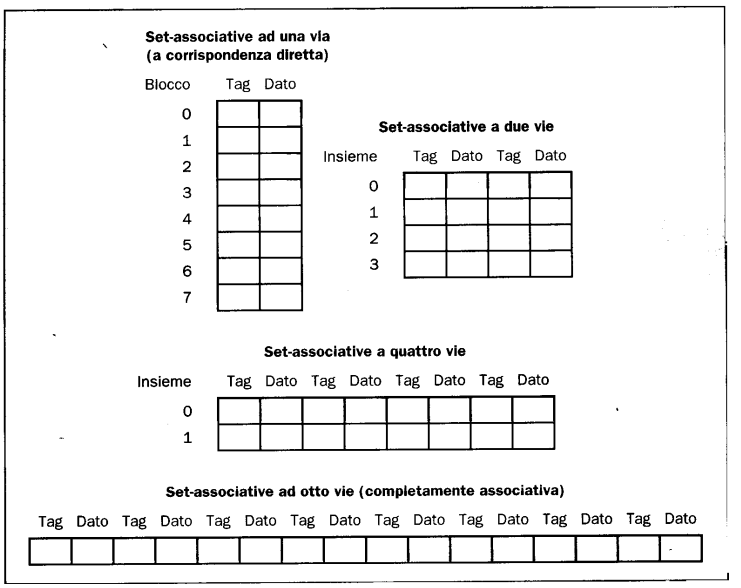
#line: 256
#word/linea: 2
#Byte/word: 4




2 livelli
- Hit in un banco
- Hit complessivo



# Dalle cache a mappatura diretta alle cache associative



# Sommario

Principio di funzionamento di una cache

Circuito di lettura / scrittura di una cache a mappatura diretta

Cache associative

Politiche di sostituzione



# Dove si può posizionare un blocco di MM in cache?

Corrispondenza diretta: in un'unica posizione.
- Memoria a 1 via. Un unico banco.
- n linee (posizione individuata dall'indice).

Completamente associative: in n posizioni (n banchi).
- Ciascun banco è costituito da 1 linea.
- n insiemi o banchi (equivalenti a n memorie indipendenti, banco individuato mediante chiave)

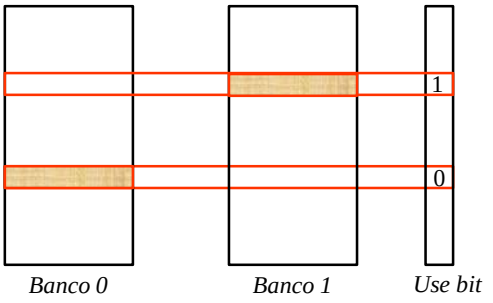
m-associative: in m posizioni (m grado di associatività).
- Ho m insiemi (banchi)
- Ciascun insieme è costituito da n linee (posizione individuata dall'indice)

# Politiche di sostituzione di una linea di cache

**Quando posso scegliere il banco, quale banco sovra-scrivo?**

LRU – Least recently Used (linea del banco utilizzato meno di recente).

Cache a 2-vie. 1 unico bit (*use bit*) di utilizzo che viene impostato a 0 o a 1 ogni volta che viene letta/scritta la linea di uno dei due banchi.

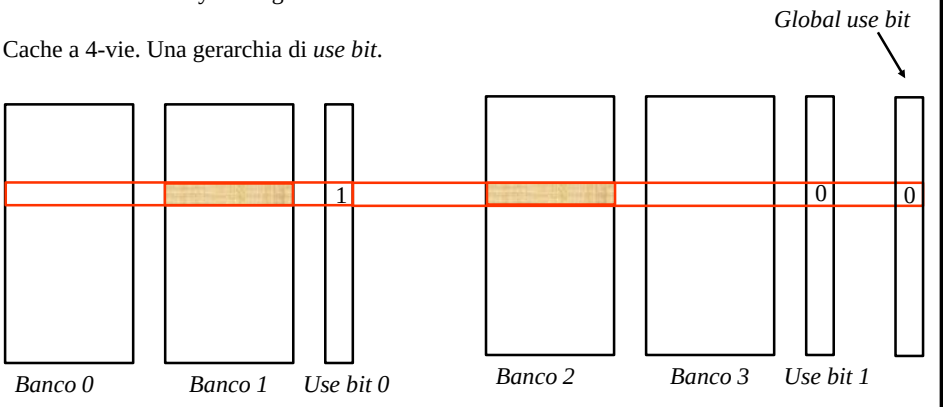





# Use bit in cache a 4-vie

LRU – Least recently Used gerarchico

Cache a 4-vie. Una gerarchia di *use bit*.




Il "Global use bit" viene impostato a 0, quando leggo/scrivo nel banco 0/1 e a 1 quando leggo/scrivo nel banco 2/3.

Identifico il banco da cui scaricare la linea percorrendo la gerarchia degli use bit:
Global use bit = 0 (coppia di banchi più vecchi è la coppia #1: banco #2 – banco #3)
Use bit 1 = 1 (il banco più vecchio è il primo della coppia: banco #2).

# Altre politiche di sostituzione

- LRU approssimato. Use bit che viene periodicamente impostato a 0 su tutte le linee (reference bit della memoria virtuale).
- LFU – Least frequently Used. Associa un contatore ad ogni linea di cache. Efficiente per memorie a 2 vie.
- FIFO – Implementazione tramite buffer circolare (cache a n-vie)
- Scelta random della linea da scaricare. Non così cattiva! Nelle cache a 2-vie, la miss rate è di circa 1.1 volte quella della politica LRU. Spesso la soluzione per cache con grado di associatività > 4.



# Sommario

Principio di funzionamento di una cache

Circuito di lettura / scrittura di una cache a mappatura diretta

Cache associative

Politiche di sostituzione