Lezione 27

La struttura gerarchica delle memorie

Proff. A. Borghese, F. Pedersini

Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

Tassonomia

- Tipologie di accesso a memoria:
  - **Sequenziale** (e.g. Nastri)
  - **Diretto** (posizionamento + attesa, e.g. Dischi).
  - **Random Access** (circuito di lettura / scrittura HW,tempo indipendente dalla posizione e dalla storia, e.g. Cache e Memoria principale).
  - **Associativa** (Random Access, il contenuto viene recuperato a partire da un sottoinsieme incompleto dello stesso).

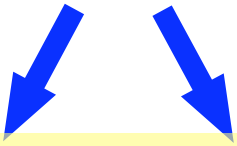
- Dispositivi di memoria, "sparsi" nell'elaboratore…
  - Processore (**registri**)
  - Memoria interna alla CPU (**cache interna**)
  - Memoria esterna (**cache esterna + memoria principale**)
  - Memoria di massa (**hard disk**)

… a "distanze" differenti dalla CPU:

Progettazione di una memoria

- Parametri di progetto di una memoria:
  - Quanta memoria?
  - Quanto veloce?
    - Maggiore è la capacità, maggiore è il tempo di accesso.
  - Quanto deve costare?
    - Maggiore è la velocità di accesso, maggiore il costo per bit.
    - Maggiore è la capacità, minore il costo per bit.

conviene progettare:



Memorie **piccole** e **veloci** | Memorie **grandi** e **lente**

Principi di località

- **Principio di località**:
  i programmi riutilizzano dati e istruzioni usati di recente.

  *Un programma spende circa il **90%** del suo tempo di esecuzione per il **10%** del suo codice*

- Basandosi sul passato recente del programma, è possibile predire con ragionevole accuratezza quali dati e istruzioni userà nel prossimo futuro.
  - **Località temporale:** elementi ai quali si è fatto riferimento di recente saranno utilizzati ancora nel prossimo futuro.
  - **Località spaziale:** elementi i cui indirizzi sono vicini, tendono ad essere referenziati in tempi molto ravvicinati.

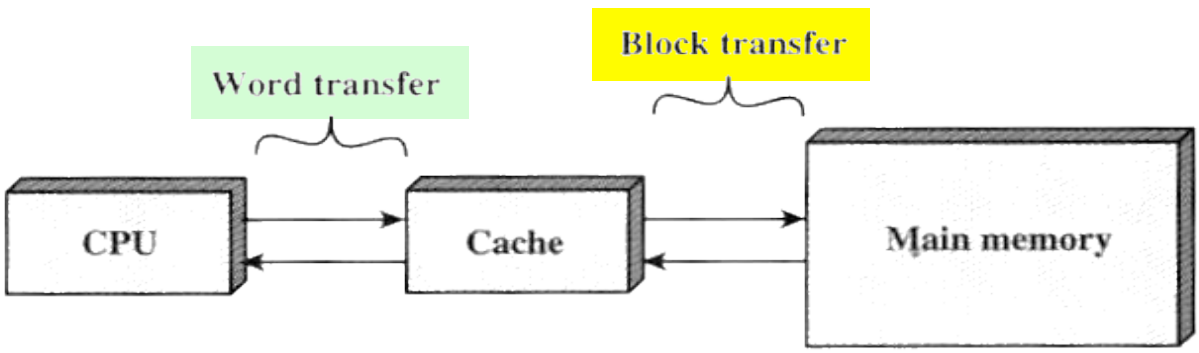
- Si possono organizzare programmi e dati in modo da sfruttare al massimo il principio di località
  - Es: scrittura di blocchi di dati nei dischi, salti locali…

Sommario

- Principio di funzionamento di una memoria cache
  - Cache a mappatura diretta
  - Il campo tag di una cache

Principio di funzionamento di una cache

- **Scopo:** fornire alla CPU…
  - una velocità di trasferimento pari a quella della memoria più veloce
  - una capacità pari a quella della memoria più grande
- Gerarchia: registri CPU – cache – memoria principale (RAM/ROM)
  - La cache contiene **copia di parte** del contenuto della memoria principale
  - "disaccoppia" i dati utilizzati dal processore da quelli memorizzati nella memoria principale



Struttura gerarchica della memoria

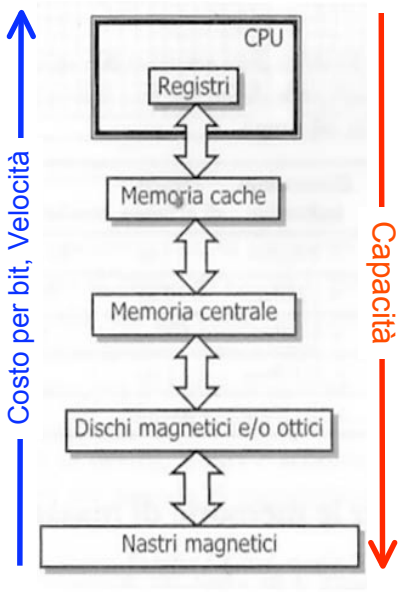
Struttura gerarchica:

Ciascun livello vede il livello inferiore

- a cui "attinge" quando necessario

- A livello superiore troviamo un sottoinsieme dei dati del livello inferiore.

Livello	Capacità	Velocità di trasferimento (Mbyte/s)
Registri	< 1 kB	400.000 (4 byte in parallelo)
Cache Primaria (P4, 3Ghz)	8–12 kB	192.000 (32 byte in parallelo)
Cache Secondaria (P4, 3Ghz)	256–512 kB	96.000 (32 byte in parallelo)
Memoria centrale (RAM, DRAM)	0,5–4 GB	1600 – 3000 (DDSRAM – doppia lettura)
Bus PCI / PCI-64	–	1060 (8 byte) / 6400 (64 byte)
Dischi	> 250 GByte	800
Nastri	> 1 TByte	10



Sottosistema di memoria

- CACHE + MEMORIA PRINCIPALE: Sottosistema di memoria
  - circuiteria dedicata alla gestione del trasferimento dati tra CPU, cache e memoria principale
- Funzione:
  - porta nella cache primaria i dati richiesti mentre il binomio processore-memoria sta lavorando
- Funzionamento:
  - Ad ogni richiesta di CPU di lettura/scrittura, controlla se la parola di memoria richiesta è già in cache:

HIT: la parola è già in cache

➔ la leggo/scrivo direttamente in cache

MISS: la parola non si trova in cache

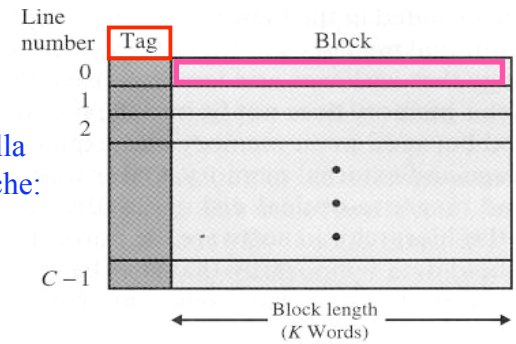
➔ copio la parola (e quelle vicine) dalla memoria di livello inferiore (mem. Principale) alla cache

Tassonomia del funzionamento

- **HIT**
  - Successo nel tentativo di accesso ad un dato: è presente al livello superiore della gerarchia.
- **MISS**
  - Fallimento del tentativo di accesso al livello superiore della gerarchia → l'indirizzo deve essere cercato al livello inferiore
- **HIT_RATE**
  - Percentuale dei tentativi di accesso ai livelli superiori della gerarchia che hanno avuto successo.
    - **HIT_RATE = Numero_successi / Numero_accessi_memoria**
- **MISS_RATE**
  - Percentuale dei tentativi di accesso ai livelli superiori della gerarchia che sono falliti
    - **MISS_RATE = Numero_fallimenti / Numero_accessi_memoria**

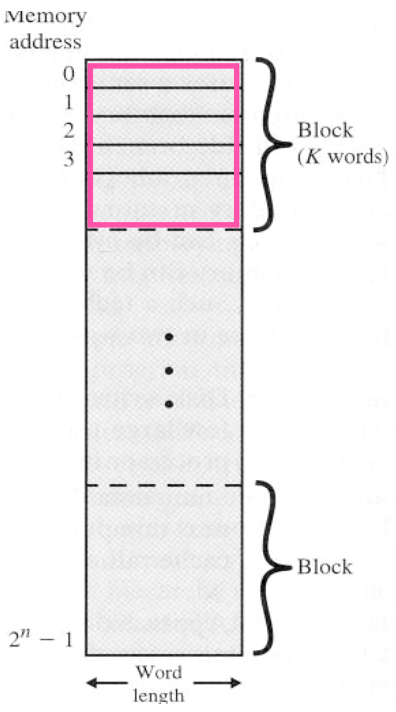
HIT_RATE + MISS_RATE = 1 (100%)

Contenuto della cache



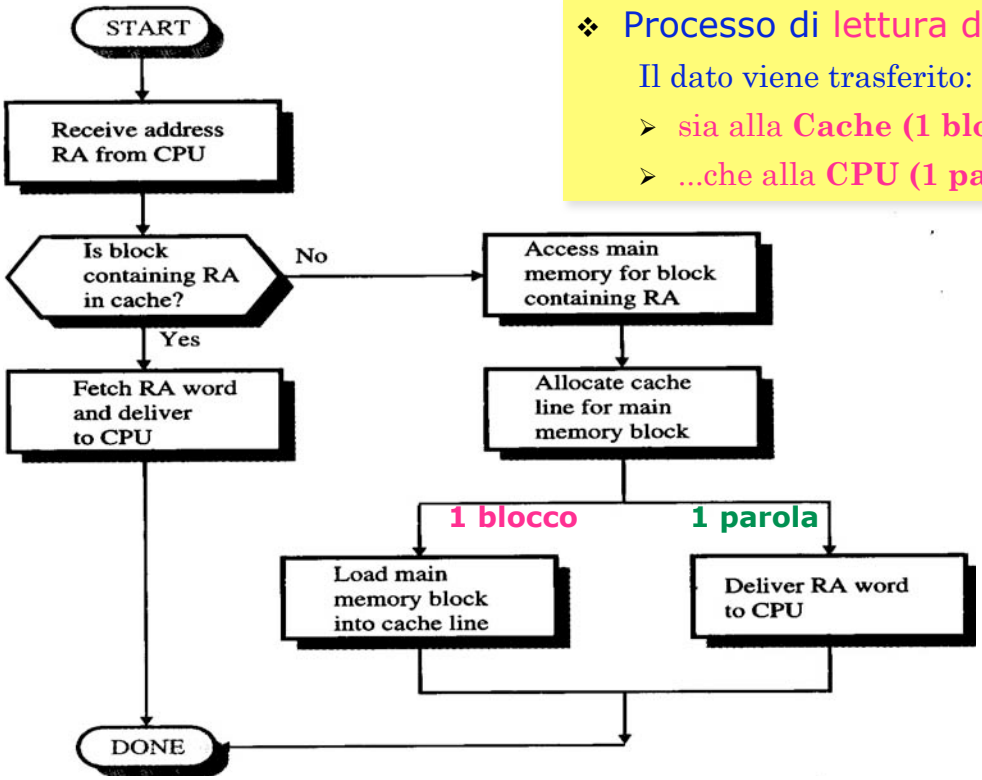
Altezza della memoria cache: # di linee

Ampiezza della memoria cache: C Parole → 4C byte in MIPS



- La cache può contenere solamente una parte della memoria principale.
- Ogni parola di cache contiene C parole della memoria principale **(blocco)**
- Il campo **Tag** indica quale blocco di RAM è scritto nella linea di cache.

Cache operation



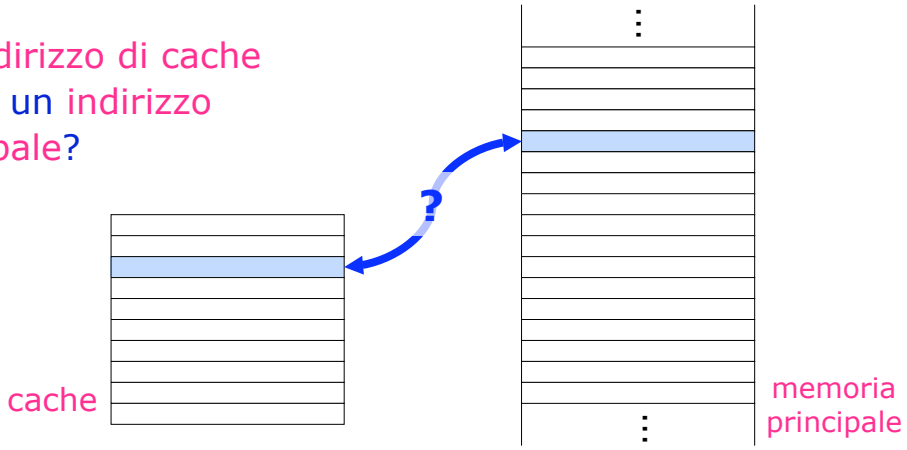
- Processo di lettura da memoria:
  Il dato viene trasferito:
  - sia alla **Cache (1 blocco)...**
  - ...che alla **CPU (1 parola)**

Sommario

- Principio di funzionamento di una memoria cache
- Cache a mappatura diretta

Corrispondenza Cache – Memoria Principale

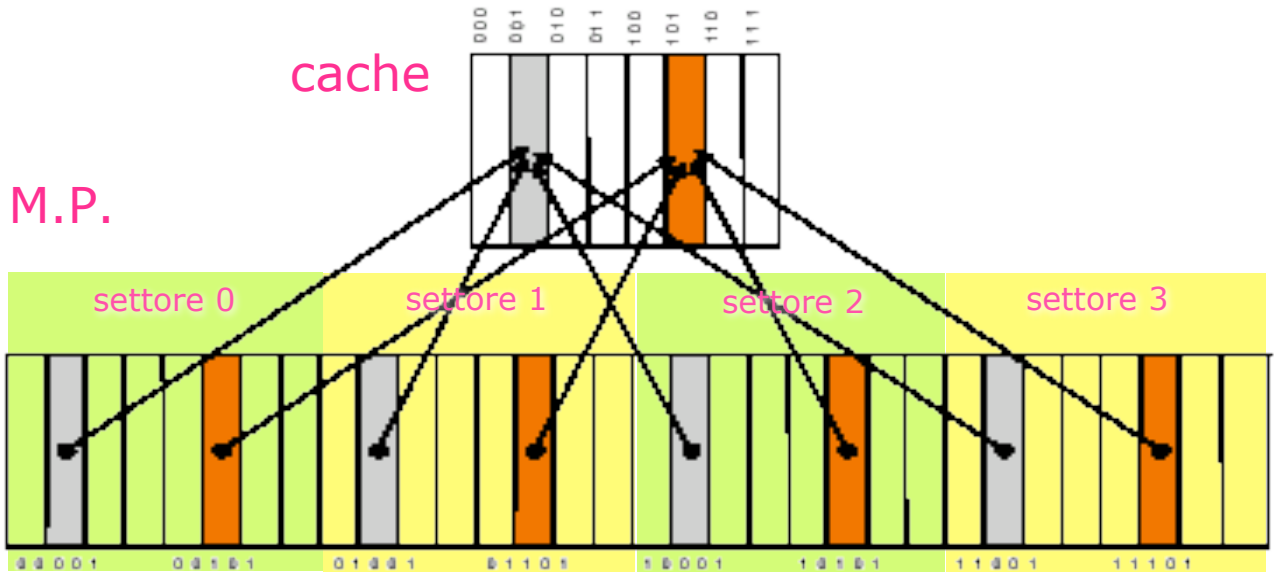
- Una linea di cache corrisponde ad un blocco di memoria principale
  - un blocco è costituito da **K parole**, ciascuna parola è costituita da **m byte** (ad esempio, MIPS: m=4)
- Devo mettere in corrispondenza:
  - un blocco di Memoria Principale (***1*** blocco: ***K*** words, ***n = K•m*** byte) con una linea di cache
- Come ottengo l'indirizzo di cache corrispondente ad un indirizzo di memoria principale?



Mappatura diretta (direct map)

Cache a mappatura diretta:

- Ad ogni indirizzo di MP corrisponde un indirizzo di cache
- Indirizzi diversi MP corrispondono allo stesso indirizzo cache



Parsing dell'indirizzo

Esempio di CACHE:

Capacità: 128 byte suddivisi in: **8 linee,** contenenti **4 parole** di **4 byte**

128 indirizzi – 2^3 linee, 2^2 parole/linea, 2^2 byte/parola

Indirizzo MP di 16 bit: 0000 0000 0000 00 (00) → 0000 0000 0111 11 (11)

- **1 Blocco** (= **1 Linea**) contiene: **4** parole di **4** bytes → **16** bytes
- I 2 bit meno significativi indicano la posizione del byte all'interno della parola: **Byte offset**
- I 2 bit seguenti indicano la posizione della parola ricercata all'interno della linea della cache: **Word offset**
- I 3 bit seguenti indicano la linea della cache interessata: **INDEX**
- I bit seguenti indicano il N. del settore della memoria principale messo in corrispondenza con la cache: **TAG**

Esempio: indirizzamento di una MP

0000 0000 0000 00(00)

0000 0000 0111 11(11)

128 indirizzi diversi
(32 parole di 4 byte)

BLOCCO
4 words = 16 byte

0000 0000 0111 1100 – 31				Byte 127
0000 0000 0111 1000 – 30				
0000 0000 0111 0100 – 29				
0000 0000 0111 0000 – 28				
0000 0000 0110 1100 – 27				
0000 0000 0110 1000 – 26				
0000 0000 0110 0100 – 25				
0000 0000 0110 0000 – 24				
0000 0000 0101 1100 – 23				
0000 0000 0101 1000 – 22				
0000 0000 0101 0100 – 21				
0000 0000 0101 0000 – 20				
0000 0000 0100 1100 – 19				
0000 0000 0100 1000 – 18				
0000 0000 0100 0100 – 17				
0000 0000 0100 0000 – 16				
0000 0000 0011 1100 – 15				
0000 0000 0011 1000 – 14				
0000 0000 0011 0100 – 13				
0000 0000 0011 0000 – 12				
0000 0000 0010 1100 – 11				
0000 0000 0010 1000 – 10				
0000 0000 0010 0100 – 9				
0000 0000 0010 0000 – 8				
0000 0000 0001 1100 – 7	1 blocco			
0000 0000 0001 1000 – 6				
0000 0000 0001 0100 – 5				
0000 0000 0001 0000 – 4				
0000 0000 0000 1100 – 3				
0000 0000 0000 1000 – 2	1 word			
0000 0000 0000 0100 – 1				
0000 0000 0000 0000 – 0	Byte 0	Byte 1	Byte 2	Byte 3

Indirizzamento scartando i bit meno significativi

Indirizzo cache LINEA n.	Indirizzo decimale Memoria Principale	Indirizzo binario Memoria Principale
7 (111)	112-127, 240-255, 368-383, ...	00 0111 0000 – 00 0111 1111
6 (110)	96-111, 224-239, 352-367, …	00 0110 0000 – 00 0110 1111
5 (101)	80-95, 208-223, 336-351, ...	00 0101 0000 – 00 0101 1111
4 (100)	64-79, 192-207, 320-335, 448-461,...	00 0100 0000 – 00 0100 1111
3 (011)	48-63, 176-191, 304-319, ...	00 0011 0000 – 00 0011 1111
2 (010)	32-47, 160-175, 288-303, ...	00 0010 0000 – 00 0010 1111
1 (001)	16-31, 144-159, 272-287, ...	00 0001 0000 – 00 0001 1111
0 (000)	0-15, 128-143, 256-261, 384-399, ...	00 0000 0000 – 00 0000 1111

La capacità della cache è di: **8 * 16 byte = 128 byte = x0000 – x1111**

Esempio di parsing dell'indirizzo

0000 0000 0000 00 (00) → **0000 0000 0111 11 (11)**

128 indirizzi diversi (32 parole di 4 byte)

- Cache – 8 linee di 4 parole (di 4 byte):
  - Blocco / linea di cache ha dimensione: **n** = 4 * 4 byte = 16 byte
  - Capacità della cache: **C** = 8 * 16 byte = 128 byte

`lw $t0, 195($zero)`

- **195 div [4 * 4 * 8] = 1** → 2° settore, resto: **R1 = 195 mod 128 = 67**
  - R1 rappresenta l'offset in byte all'interno della cache
- **67 div [4 * 4] = 4** → 5ª linea della cache, resto: **R2 = 67 mod 64 = 3**
  - R2 rappresenta l'offset in byte all'interno della linea di cache
- **3 div 4 = 0** → 1ª parola della cache, resto: **R3 = 3 – 0 * 4 = 3**
  - R3 rappresenta l'offset in byte all'interno della parola

Parsing dell'indirizzo

0000 0000 0000 00 (00) → **0000 0000 0111 11 (11)**

128 indirizzi diversi (32 parole di 4 byte)

Prendiamo un indirizzo **M** di MP ed **operiamo per divisioni successive:**

Capacità cache

M / [(bytes/parola) * (parole/linea) * n.linee] = N. settore di MP: **TAG**

- Il resto **R1:** rappresenta l'offset (in byte) all'interno della cache.

Dim. linea

R1/ [(bytes/parola) * (parole/linea)] = Numero linea di cache: **INDEX**

- Il resto, **R2**, rappresenta l'offset (in byte) all'interno della linea di cache.

Dim. parola

R2/ [(bytes/parola)] = N. parola nella linea di cache: **WORD OFFSET**

- Il resto, **R3**, rappresenta l'offset (in byte) nella parola: **BYTE OFFSET**

Come si recupera un dato dalla cache?

- Posso ottenere i 4 campi mediante operazioni di **divisione intera** (**div**) e di **resto** (**mod**):

  Byte Offset = Ind_MP [bytes] **mod** Capacità_cache [bytes]

  Word Offset = Ind_MP [parole] **mod** Capacità_cache [parole]

  INDEX = Ind_MP [linee] **mod** Capacità_cache [linee]

  TAG = Ind_MP [byte] **div** Capacità_cache [byte] =
  = Ind_MP [linee] **div** Capacità_cache [linee]

- Se le dimensioni in gioco sono potenze della base utilizzata (base 2):
- l'operazione `modulo` si ottiene scartando le cifre **più** significative
- l'operazione `div` si ottiene scartando le cifre **meno** significative

- Esempio: Memoria principale: **64 kB** (indirizzo: **16 bit**)
  Cache: capacità **128** bytes, in **8** linee di **4** words da **4** bytes

indirizzo: **0x1234 = 0001 0010 0 011 01 00**

TAG | Index | W.O. | B.O.

Come scoprire se un dato è presente in cache?

- Dato un indirizzo **M** in memoria principale, la sua posizione in cache è definita da: **INDEX, Word Offset, Byte Offset**
- Nel meccanismo di mappatura diretta, si perde l'informazione del N. settore di MP di appartenenza:

- È necessario memorizzare il N. di settore: campo **TAG**
- Non è invece necessario memorizzare gli altri campi, perché ottenuti da **M.**

- Campo **TAG**
  - Associato a ciascuna linea di cache
  - Contiene l'informazione: da quale settore di MP arriva il blocco?

- TAG è rappresentato dai bit che costituiscono la parte più significativa dell'indirizzo.
  - E' costituito da K bit $K = M - \sup(\log_2 C)$
  - Nell'esempio precedente: $K = 32 - \sup(\log_2 128) = 25$ bit

Bit di validità

- Può accadere che il dato in cache non sia significativo
  - Cache ancora vuota, valore in cache non consistente con il valore in MP

Bit di validità:

- Indica la consistenza tra cache e MP
- Quando un blocco viene copiato da MP a cache:
  - VALID ➔ 1
- Quando un blocco in cache viene modificato da CPU
  - e si perde quindi la consistenza:
  - VALID ➔ 0

- Un blocco in cache può essere utilizzato (HIT) se:
  - Il campo TAG coincide con l'indirizzo
  - VALID = 1

Struttura di una linea di cache

- Quanto è lunga una linea di cache?
  - Nell'esempio precedente avremo linee di cache di lunghezza:

    L = 25 + 1 + 4*4*8 = 134 bit
- Quanti settori ci sono in MP?
  - tanti quanti ne può indirizzare il campo: **tag**
  - nell'esempio precedente: ***K = 25 → $N_{SMP} = 2^{25}$ = 32 M Settori***

Come leggere / scrivere su cache

Meccanismo di lettura/scrittura su cache:

1. Individuare la linea della cache dalla quale leggere / scrivere
2. Confrontare il campo TAG con il blocco di MP in cui risiede il dato.
3. Controllare il bit di validità.
4. Se:

   (TAG == [parte alta dell'indirizzo]) ***AND*** *(Valid_bit = TRUE)*

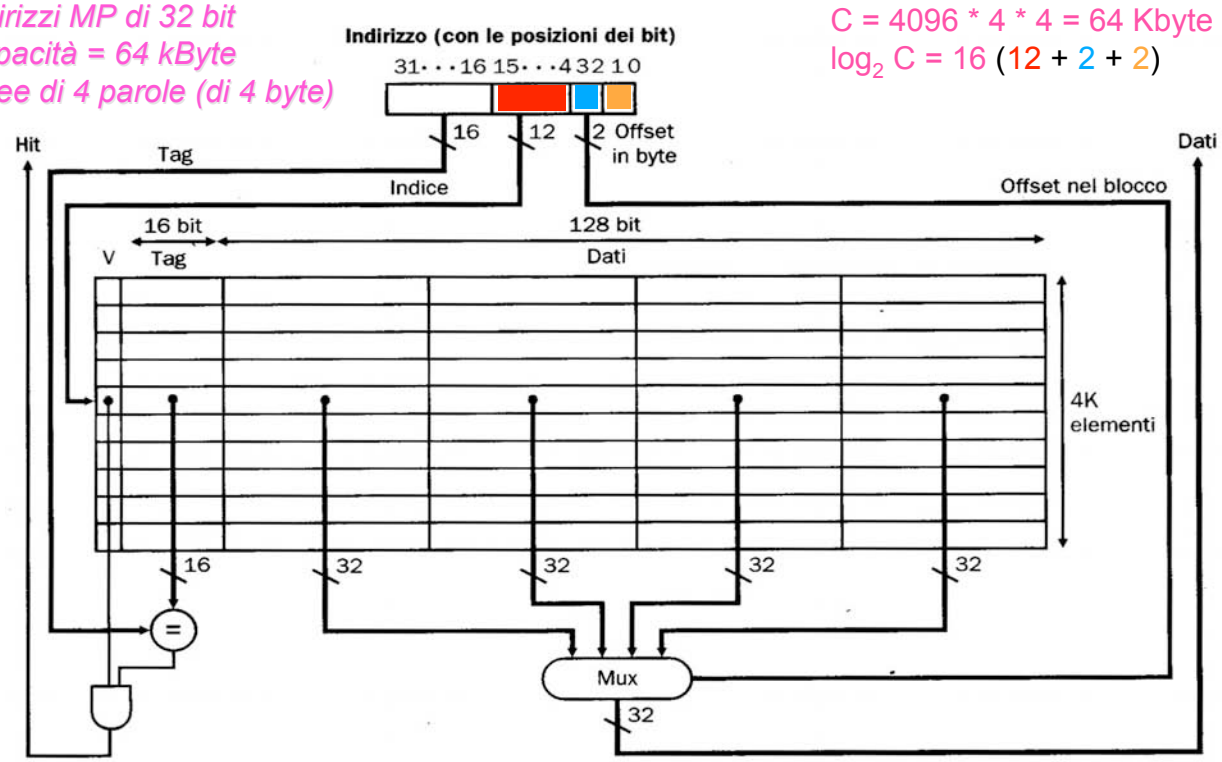
   Allora:

   Leggere (scrivere) **l'intero blocco (linea)** di appartenenza della parola indirizzata.

Schema: cache a mappatura diretta

Indirizzi MP di 32 bit
Capacità = 64 kByte
Linee di 4 parole (di 4 byte)

C = 4096 * 4 * 4 = 64 Kbyte
$\log_2$ C = 16 (12 + 2 + 2)



Esercizi

- Sia data una cache a corrispondenza diretta contenente 64Kbyte di dati e avente blocchi di 1 parola. Assumendo che gli indirizzi siano di 32 bit quale è il numero totale di bit richiesto per l'implementazione della cache?

- Supponendo che il MIPS abbia una cache di 512byte, indicare cosa succede nei campi della cache quando vengono eseguite le seguenti istruzioni:

```
lw $t1, 0x0000($t0)   $t0 = 1kbyte = 1,024 byte
lw $t1, 0x0000($t0)   $t0 = 0
lw $t1, 0x0202($t0)   $t0 = 1kbyte = 1,024 byte
lw $t1, 0x0001($t0)   $t0 = 0
lw $t1, 0x0201($t0)   $t0 = 1kbyte = 1,024 byte
```