



# Le memorie Cache n-associative

Prof. Alberto Borghese  
Dipartimento di Scienze dell'Informazione  
[alberto.borghese@unimi.it](mailto:alberto.borghese@unimi.it)

Università degli Studi di Milano

Riferimento Patterson: 5.2, 5.3



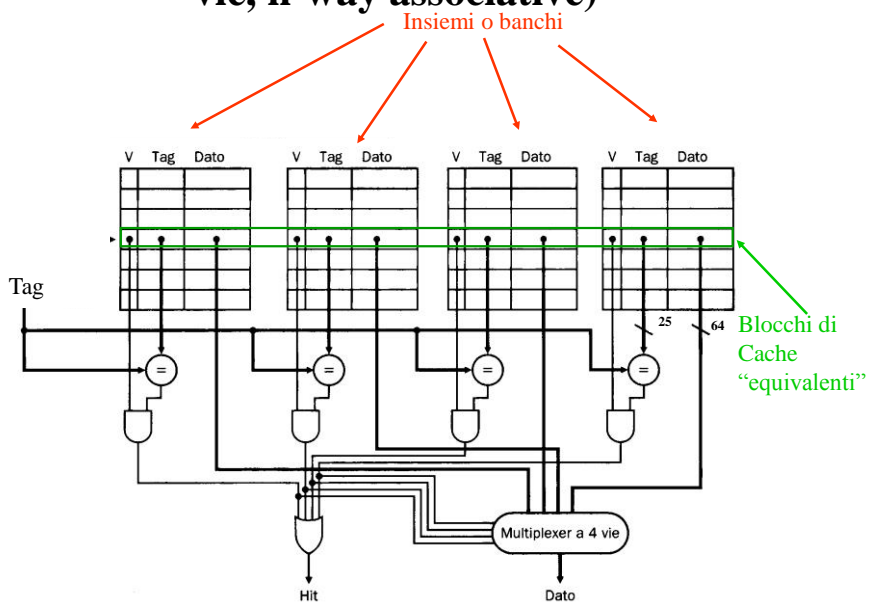
## Sommario

Memorie n-associative

SRAM



## Memorie n-associative (o associative a n-vie, n-way associative)



## 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.

**Blocco (linea di cache):** #parole (byte) lette/scritte contemporaneamente in cache, “parola” della cache.

**Insieme (banco):** cache elementare.

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

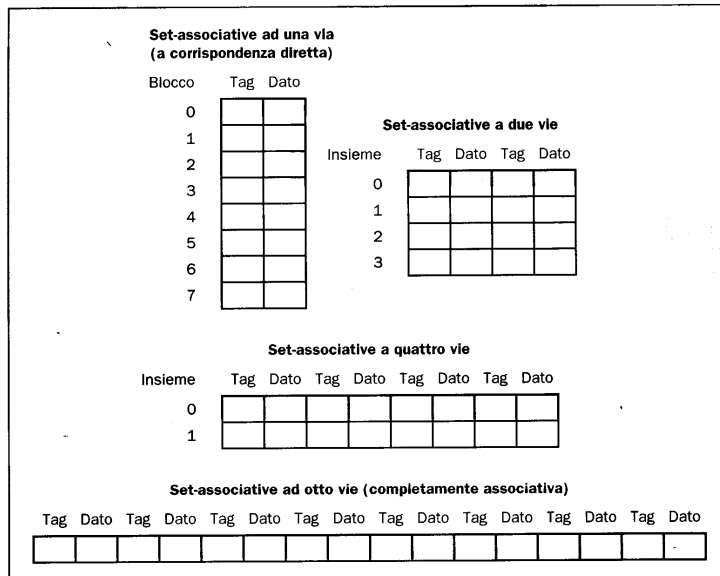
**Capacità della cache:** #parole = #Insiemi \* (#blocchi / insieme) \* (#parole / blocco).

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

Per cercare un dato non devo più analizzare tutte le linee di una cache, ma un’unica linea per ogni banco.



## Dalle cache a mappatura diretta alle cache associative



## Accesso a cache ad n-vie



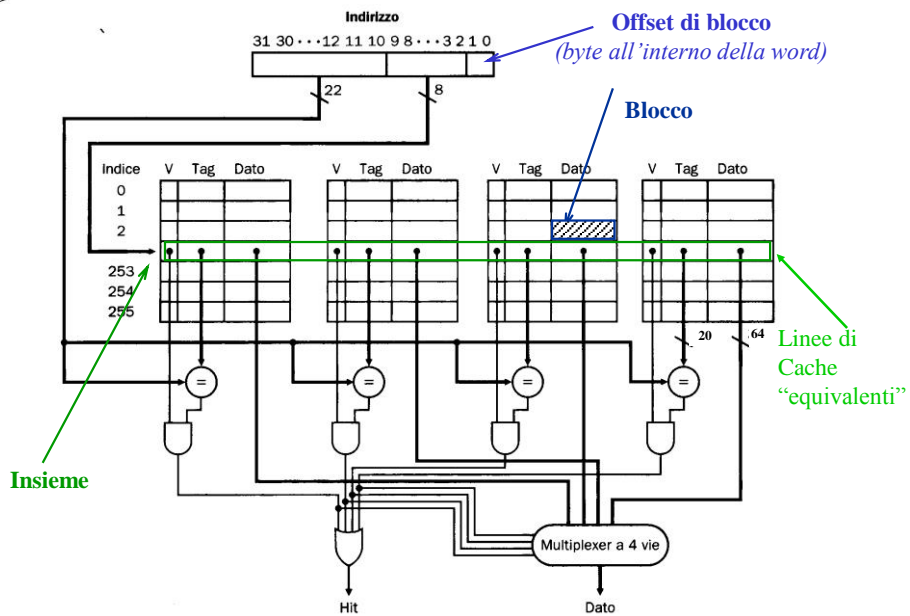
**INDICE.** Se la parola richiesta è memorizzata in cache, si trova in una particolare linea di uno dei banchi. Questa linea è individuata dall'indice. L'indice è costituito da  $k$  bit, dove  $k = \log_2(\#linee)$ . E' analogo al numero di linea nelle cache a mappatura diretta.

**TAG** – contiene il blocco della RAM a cui appartiene il dato. Cerca il tag di Memoria Principale all'interno dei TAG associati alla linea individuata in ciascun banco.

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



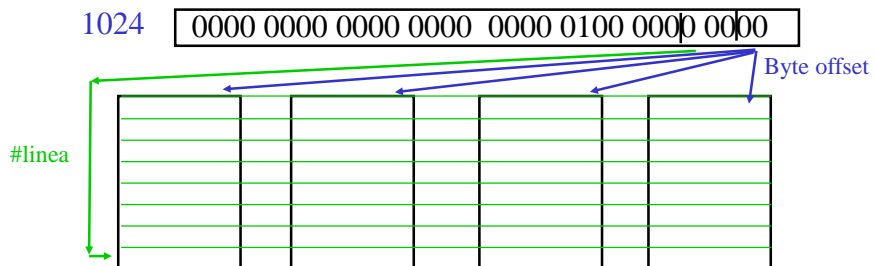
## Memorie set-associative



## Memorie n-associative con blocchi di 1 parola

Esempio: cache di 4 banche, ciascuno di 8 linee. Parola di cache = 1 word, non c'è offset nel blocco.

Come viene elaborato l'indirizzo: `lw 0($s0)`    `$s0 = 1024`



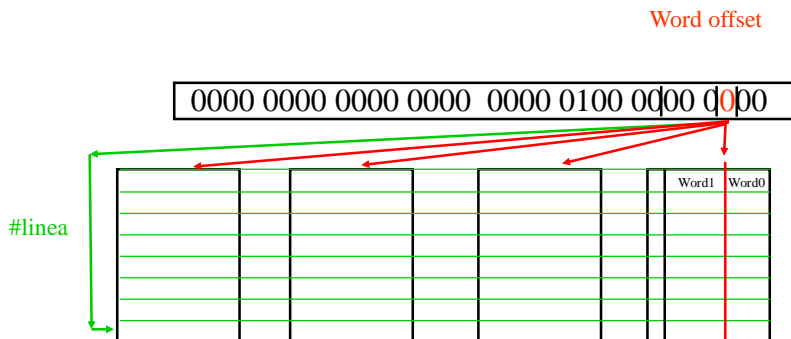


## Memorie n-associative con blocchi di 2 parole

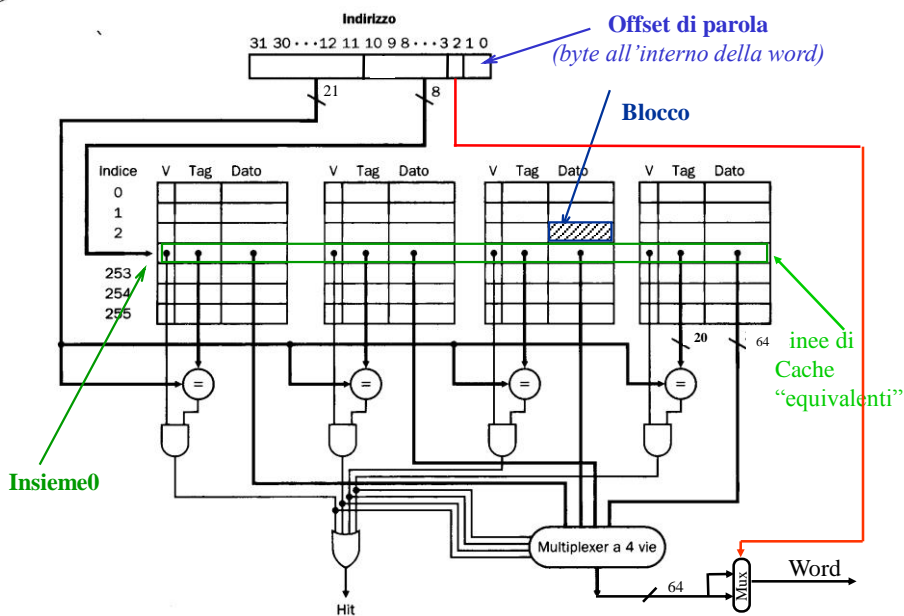


Esempio: cache di 4 banchi, ciascuno di 8 linee. Parola di cache = 2 word.

Come viene elaborato l'indirizzo: lw 0(\$s0)?    \$s0 = 1024



## Memorie set-associative





## Criteri di sostituzione di un blocco



### Dove inserisco il blocco letto dalla RAM?

Soluzione hardware, algoritmo semplice.

LRU – Least recently Used. Viene associato ad ogni blocco un bit di USE. Periodicamente viene azzerato (cf. Memoria virtuale)

FIFO – Implementazione tramite buffer circolare.

LFU – Least frequently Used. Associa un contatore ad ogni blocco di cache. Efficiente per memorie a 2 vie.

RANDOM – Non funziona molto peggio!!



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



Corrispondenza diretta: in un'unica posizione.

Memoria ad 1 via.  
#posizioni = #linee.

Completamente associative: in n posizioni (n banchi).

Ciascun banco è costituito da 1 linea.  
n insiemi o banchi.

N-associative: in m posizioni (m grado di associatività).

Ho m insiemi (banchi)  
Ciascun insieme è costituito da n linee.



## Come si trova un blocco di RAM in cache?



Corrispondenza diretta: indicizzazione.

Controllo del tag del blocco (1 comparazione).

Associativa: ricerca in tutti gli elementi della cache.

$n$  comparazioni: controllo di tutti i tag.

La memoria virtuale è di questo tipo (tramite la *Page Table*).

N-associativa: ricerca negli  $m$  insiemi,

$m$  comparazioni.



## Sommario



Memorie n-associative

**SRAM**



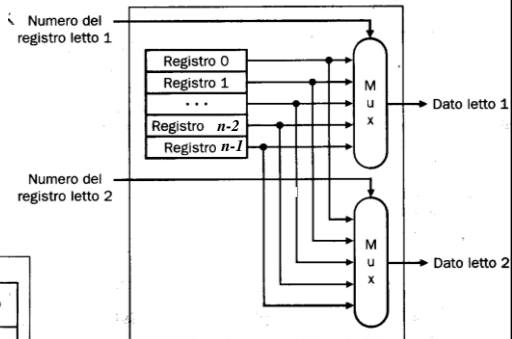
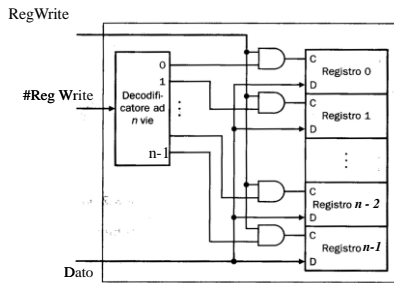
# Register file



Il tempo di lettura dipende dal cammino critico dei Mux.

Il tempo di scrittura dipende dal cammino critico del Decoder.

Numero\_registro = selettore.

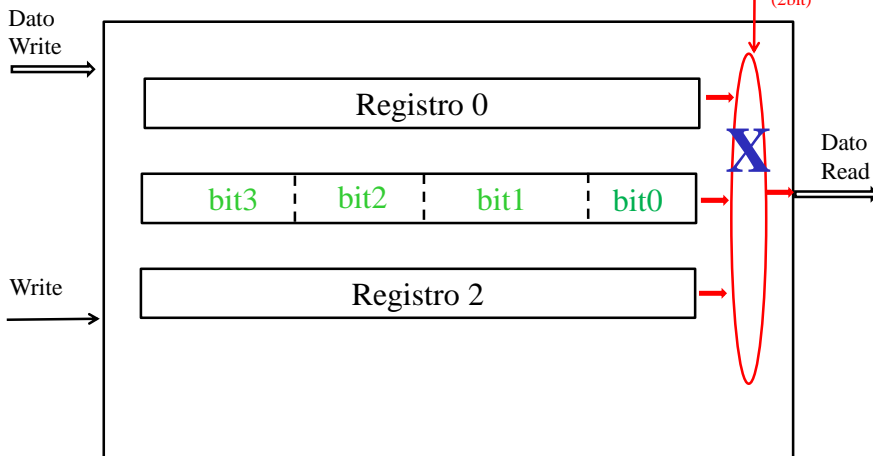


Selezione - #registro

Letture - sempre disponibile in uscita (dopo tempo di commutazione del MUX)  
Scrittura - segnale esplicito (in AND con il clock in caso di cella sincrona).

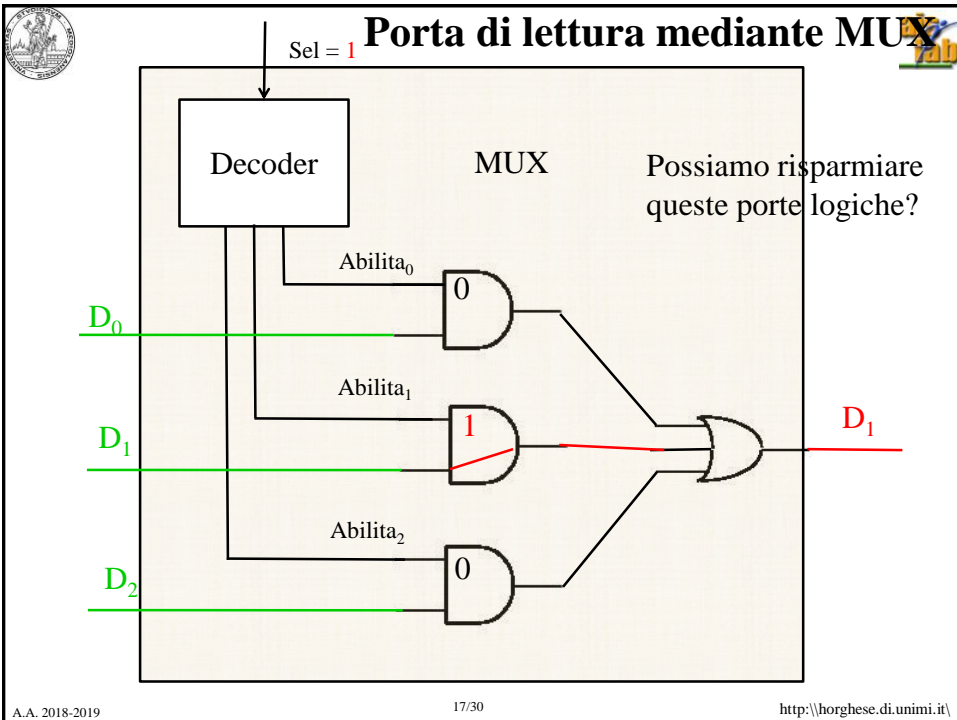


# Letture da un banco di registri



Selezione uno dei registri = porto in uscita l'uscita Q di tutti i bit del registro selezionato  
Avrò tanti Mux quanti sono i bit che costituiscono il registro (**Voglio eliminare questo mux**)





## Porte three-state

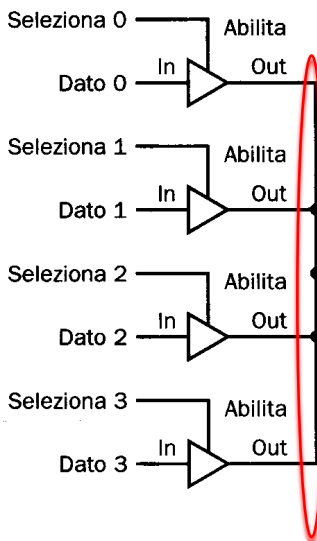
- Stato 0 – spento – connesso a massa (0V)
- Stato 1 – acceso – connesso all'alimentazione (+Vcc)
- Stato disabilitato – disconnesso – non connesso a nulla, tensione floating.

Per le memorie si può introdurre l'abilitazione in lettura / scrittura oppure in accesso in modo che internamente l'informazione venga conservata.

A.A. 2018-2019 18/30 http://horghese.di.unimi.it/

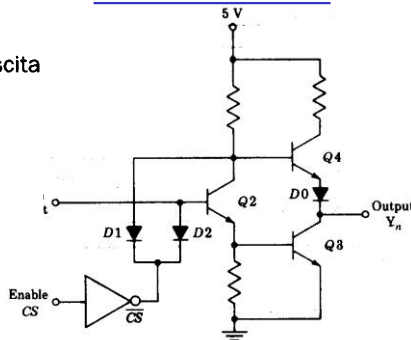


## Memoria three-state (soluzione HW)

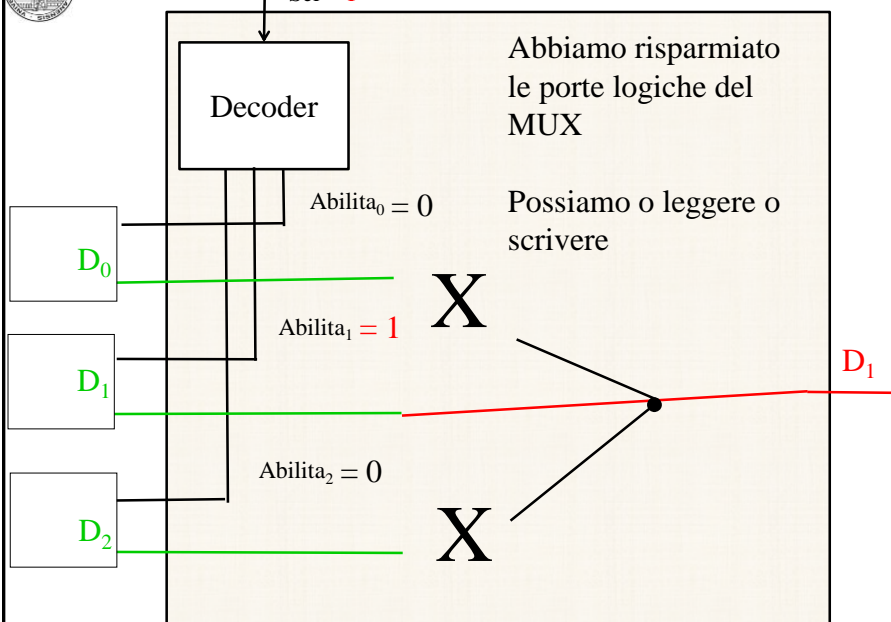


Tutte le uscite delle celle sono collegate ad un'uscita comune => E' necessario evitare conflitti fra le uscite.

Stadio di uscita "isolata" con porte di uscita three-state  
 Selezione una sola cella alla volta



## Porta di lettura three state



Abbiamo risparmiato le porte logiche del MUX

Possiamo o leggere o scrivere

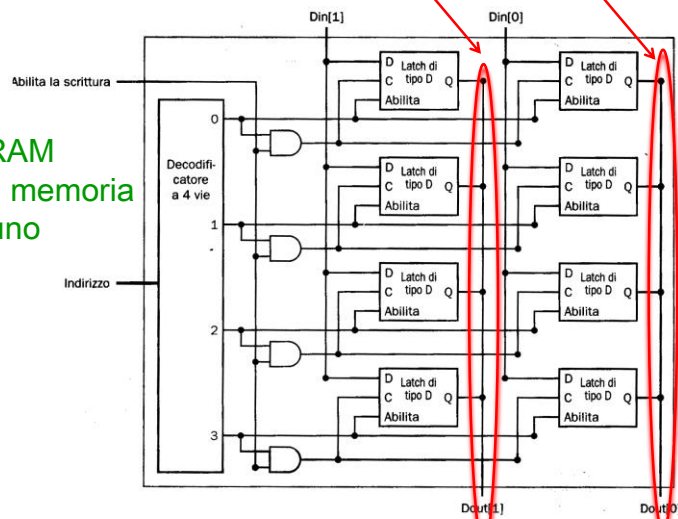


## Esempio di SRAM (cache)

Risparmio il Mux di uscita



Esempio: SRAM  
4 elementi di memoria  
x 2 bit ciascuno



Problemi con il crescere del numero di linee. Esempio: SRAM  $2M \times 16$  ( $2M$  elementi x 16 bit)  
Decodificatore a  $21$  ( $\log_2 2M$ ) bit. 16 uscite.  $2M$  linee di abilitazione e di selezione (ingresso C) dei bistabili.



## Migliore strutturazione



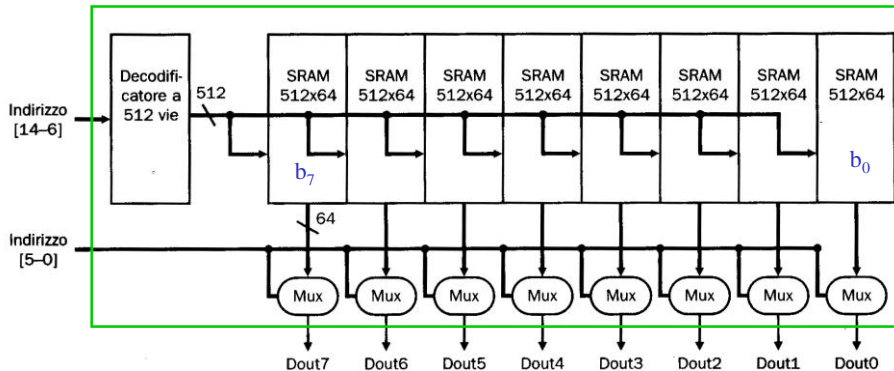
- C'è un limite elettrico al numero di linee che si possono collegare assieme.
- È più conveniente separare la lettura delle linee dalla lettura delle colonne (estrazione di una linea «lunga» dalla memoria).



## Indirizzamento SRAM a matrice



Esempio: SRAM 32K x 8 bit. Trasformo 32K linee in una matrice: 512 linee x 64 elementi. Ciascun elemento della matrice contiene 8 bit.



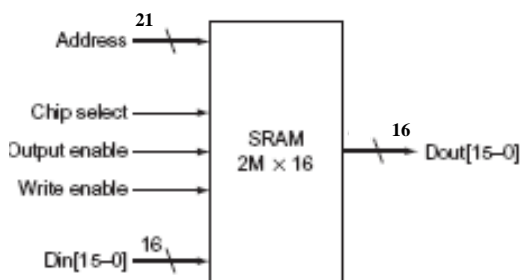
Il decodificatore sarà a 9 bit ( $\log_2 512$ ) per selezionare una delle 512 linee (cf. cache). Ciascuna linea fornisce 64 bit. Ne seleziono uno con il Mux (controllato dai 6 bit meno significativi).

Nell'approccio non a matrice avrei avuto bisogno di un decodificatore a 15 bit ( $\log_2 32K$ ). Qual è il vantaggio?

Quale gruppo di 64 bit associati al  $b_j$  bit della parola su 8 bit leggo dalle SRAM?



## Chip di SRAM (2M x 16)



Tempo di accesso:  
da Address a Dout.

Selezione – indirizzo + Chip select.

Scrittura – Write enable.

Letture – Output enable.

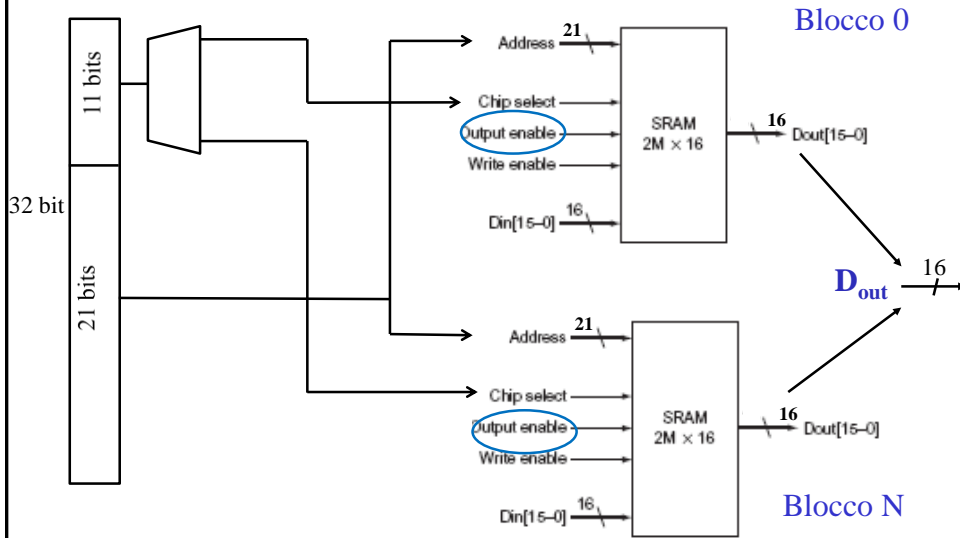
**E' una memoria veloce, ma costosa.**



## Memoria RAM come insieme di chip



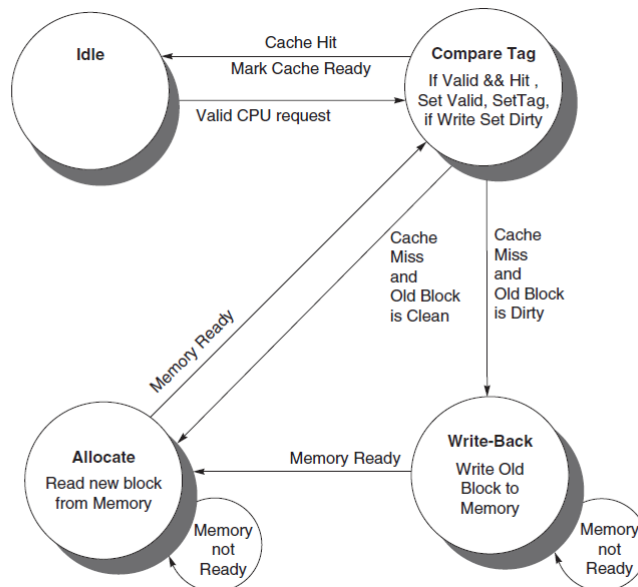
Capacità totale =  $2M * N$  Byte



Questi chip “tappizzano” la struttura logica della memoria



## Controllore di una memoria cache



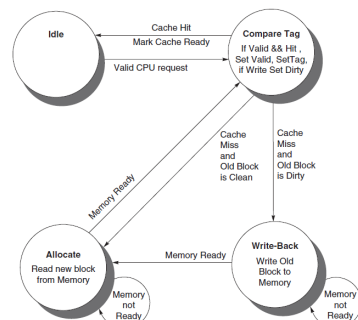


## Funzionamento del controller



### 4 STATES:

- **Idle:** waits for a valid read/write from the processor
- **Compare tags:** identify the cache line and check TAG and validity bit.
  - HIT -> tag is found and line is valid
  - MISS -> tag is not found.
    - Dirty bit is set to 1 → the line has to be copied in Main Memory before overwriting (write-back)
    - Dirty bit is set to 0 → the data can be immediately be transferred from Main Memory (write allocate)
- **Allocate:** Read a new cache line from main memory into cache.
- **Write-Back:** Write old cache line into main memory.



## Controllore di una memoria cache



Macchina a stati finiti:

Stati = {Idle, CompareTag, Write-Back, Allocate}

Input = {Valid\_CPU\_Request, Cache\_Hit, Dirty, Memory\_Ready}

Uscite = {Read\_new\_Block, Write\_old\_block, SetHit, SetMiss, SetValid, SetTag, SetDirty}

NB SetDirty imposta il bit "Dirty", ma costituisce anche un input.

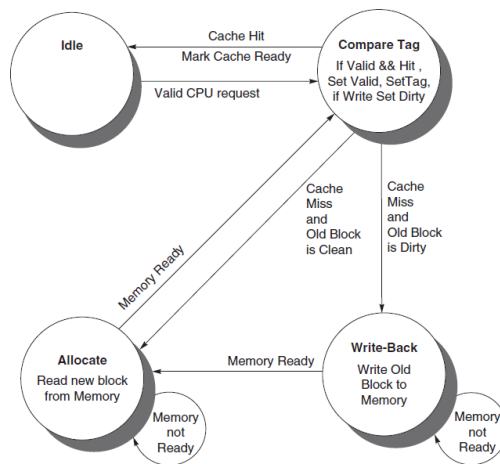
Stato iniziale: idle.

Stato prossimo:  $f(x,u)$

Uscita:  $g(x)$



## Controllore di una memoria cache



Sintetizzare come macchina di Huffman



## Sommario

Memorie n-associative

SRAM