

Corso di Architettura degli Elaboratori e Laboratorio (M-Z)

Sistemi di memoria

Nino Cauli



UNIVERSITÀ
degli STUDI
di CATANIA

Dipartimento di Matematica e Informatica

- Le unità memoria sono usate per immagazzinare informazione necessaria per eseguire i programmi
- Sono circuiti elettronici in grado di preservare l'informazione che può essere costituita da:
 - **ISTRUZIONI**, eseguite dalla CPU
 - **DATI**, utilizzati dalle istruzioni eseguite
- La memoria si può dividere in **MEMORIA PRIMARIA** e **MEMORIA SECONDARIA**

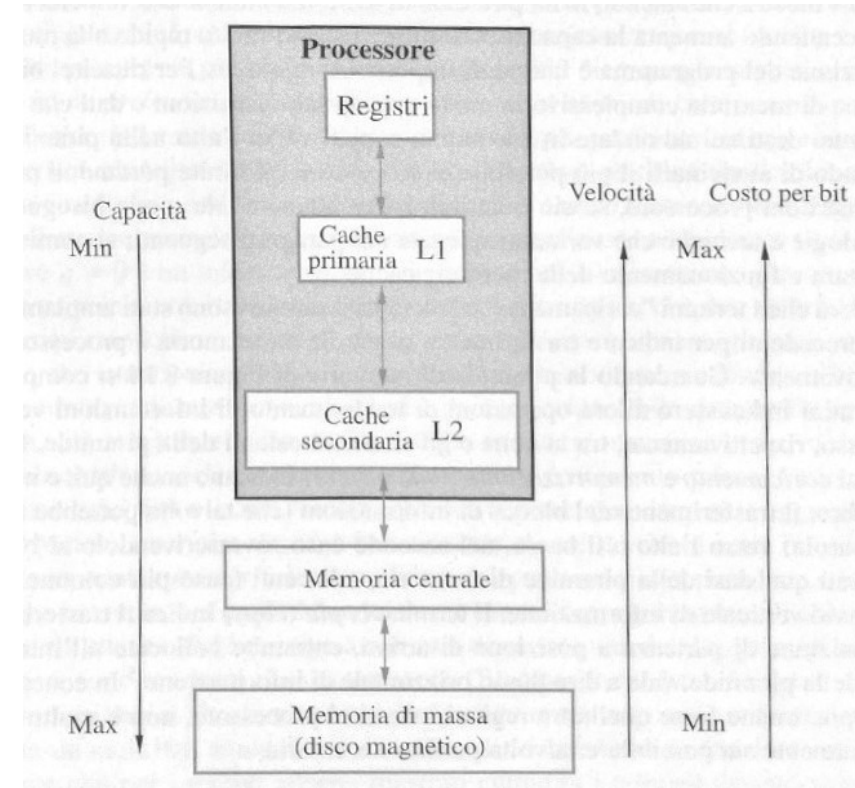
- La memoria primaria è **VELOCE**, con **CAPACITÀ LIMITATA** e **VOLATILE**
- La tecnologia usata si chiama **MEMORIA AD ACCESSO CASUALE (RAM)**
- Organizzata su **LIVELLI** (alti + veloci e – capienti, bassi – veloci e + capienti)
- **CACHE**: livello più alto (molto veloce, integrata nel processore)



- La memoria secondaria è **LENTA**, con **CAPACITÀ ELEVATA** e **NON VOLATILE**
- Viene usata per immagazzinare **GROSSE QUANTITÀ** di dati in modo **PERMANENTE** o per **LUNGI PERIODI**
- Varie tecnologie disponibili: **DISCHI MAGNETICI**, **DISCHI OTTICI** (CD e DVD), **MEMORIE FLASH**, etc.

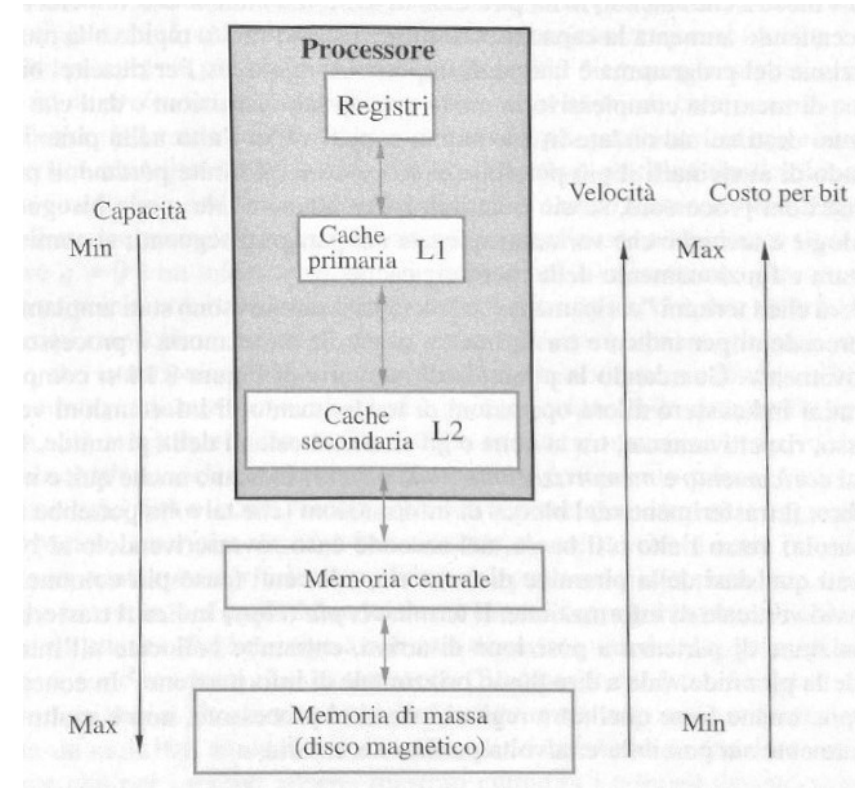


- La gerarchia di memoria è organizzata a piramide
- Tecnologia di memorizzazione più performante = maggior costo
- Livelli di memoria in cima alla piramide più piccoli e veloci
- I programmi sono immagazzinati nella memoria di massa e solo le loro porzioni attive vengono caricate nei livelli più alti



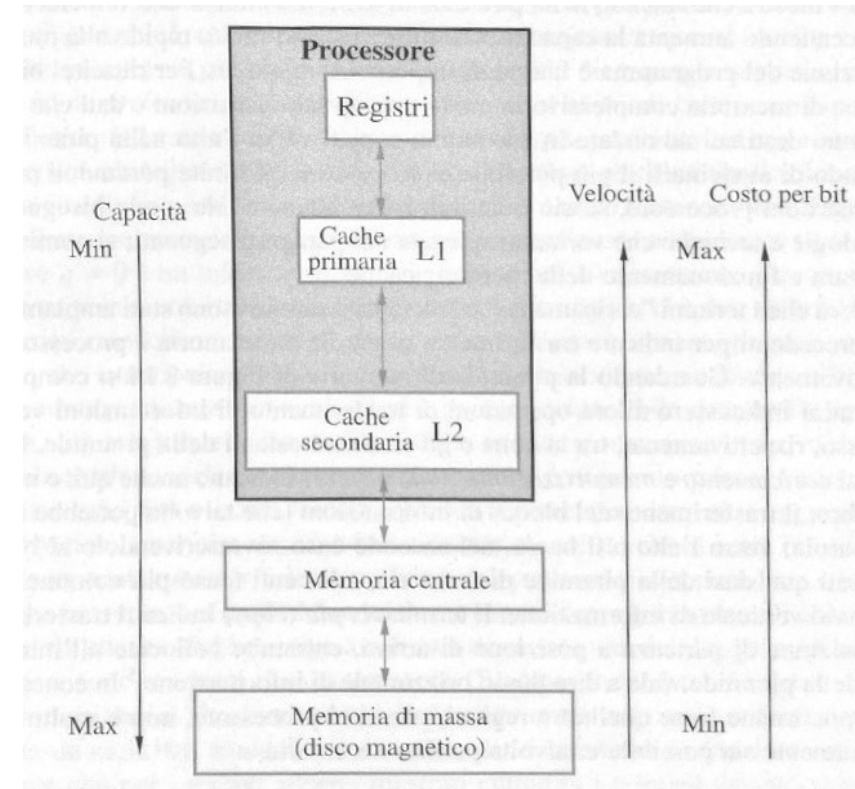
I livelli di memoria sono i seguenti:

- **Registri:** velocissimi, capacità molto ridotta, integrati nel processore, flip-flops
- **Livelli di Cache (L1, L2):** molto veloci, capacità ridotta (decine di KB / qualche MB), integrati nel processore, SRAM
- **Memoria centrale:** veloce, capacità media (qualche GB), DRAM
- **Memoria di massa:** lenta, capacità elevata (qualche TB), dischi magnetici – memorie flash

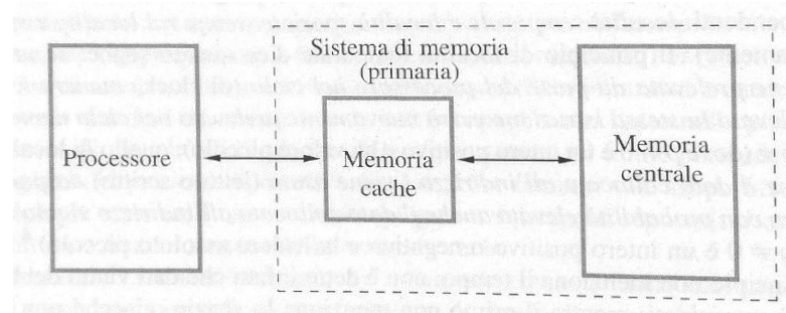


Operazioni tra livelli di memoria:

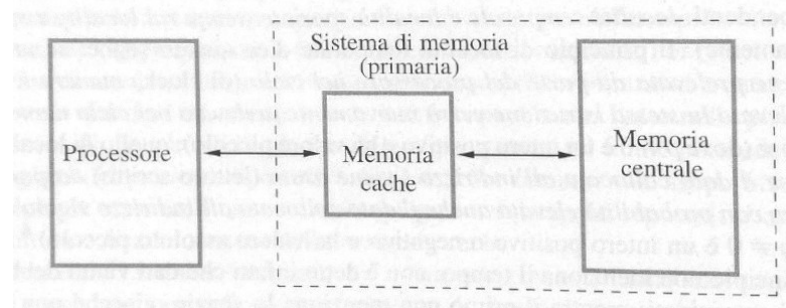
- **LOAD (caricamento):** trasferimento dei dati verso l'alto
- **STORE (memorizzazione):** trasferimento dei dati verso il basso, sovrascrivendo il dato originario
- **COPY (copia):** trasferimento “orizzontale” dei dati all'interno dello stesso livello di memoria



- La memoria Cache è una memoria piccola e veloce interposta tra memoria centrale e processore, che contiene copie di istruzioni e dati della memoria centrale da usare al momento
- Si basa sui principi di località di dati e istruzioni:
 - **Località temporale (istruzioni):** se un'istruzione è prelevata nel ciclo i , con probabilità elevata verrà prelevata nuovamente nel ciclo $i + p$ (con p piccolo intero positivo)
 - **Località spaziale (dati):** se un dato collocato all'indirizzo i viene usato dal processore, con probabilità elevata verrà usato anche il dato collocato all'indirizzo $i \pm q$ (con q piccolo intero positivo)
 - **Località spaziale-temporale (dati e istruzioni):** se un blocco di parole va in uso da parte del processore, con probabilità elevata entro breve tempo e per più volte esso verrà usato nuovamente



- La memoria centrale è organizzata in **blocchi di parole**
- Quando la CPU accede alla memoria, il blocco contenente la parola interessata viene caricato nella cache
- La cache è divisa in spazi grandi quanto i blocchi di memoria detti **linee di cache (cache lines)**
- Possono verificarsi due casi:
 - **Cache hit:** il blocco interessato è già presente in cache
 - **Cache miss:** il blocco interessato deve essere caricato dalla memoria centrale



- Se la CPU accede ad una parola di memoria contenuta in un blocco già presente in cache si dice che avviene un **cache hit**
- **Cache hit in lettura:**
 - il processore legge la parola dalla cache
- **Cache hit in scrittura:**
 - **Write through (scrittura immediata):** si aggiornano assieme sia la copia della parola in cache che quella in memoria centrale
 - **Write back (scrittura differita):** si aggiorna solo la copia della parola in cache e si marca la posizione come modificata (**dirty bit o modified bit**). La parola in memoria verrà aggiornata quando si libera la posizione corrispondente in cache

- Se la CPU accede ad una parola di memoria contenuta in un blocco non presente in cache si dice che avviene un **cache miss**
- **Cache miss in lettura:**
 - **Read back (lettura differita):** il processore attende che il blocco sia caricato sulla cache e poi procede con la lettura
 - **Load through (lettura immediata):** il processore legge la parola appena essa viene caricata in cache senza aspettare che tutto il blocco venga caricato
- **Cache miss in scrittura:**
 - **Write through (scrittura immediata):** la parola viene modificata direttamente sulla memoria centrale senza attesa del processore
 - **Write back (scrittura differita):** il processore attende che il blocco sia caricato sulla cache e poi la parola viene modificata in cache marcando la posizione come modificata

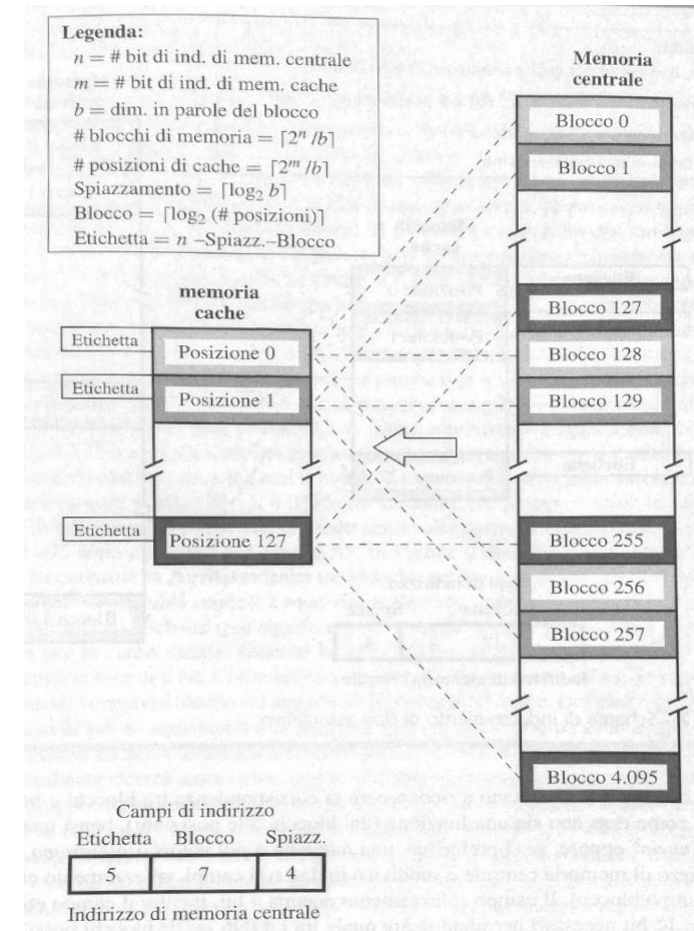
- Il numero di posizioni in cache è molto inferiore al numero di blocchi in memoria
- Lo **schema di indirizzamento** è la funzione di associazione tra blocchi di memoria e posizioni in cache
- Esistono 3 tipi di schemi di indirizzamento:
 - **Indirizzamento diretto:** ciascun blocco è caricabile in una sola posizione in cache
 - **Indirizzamento associativo:** ciascun blocco è caricabile in qualsiasi posizione in cache
 - **Indirizzamento associativo a gruppi:** ciascun blocco è caricabile in un gruppo di posizioni in cache (caso generico)

Esempio:

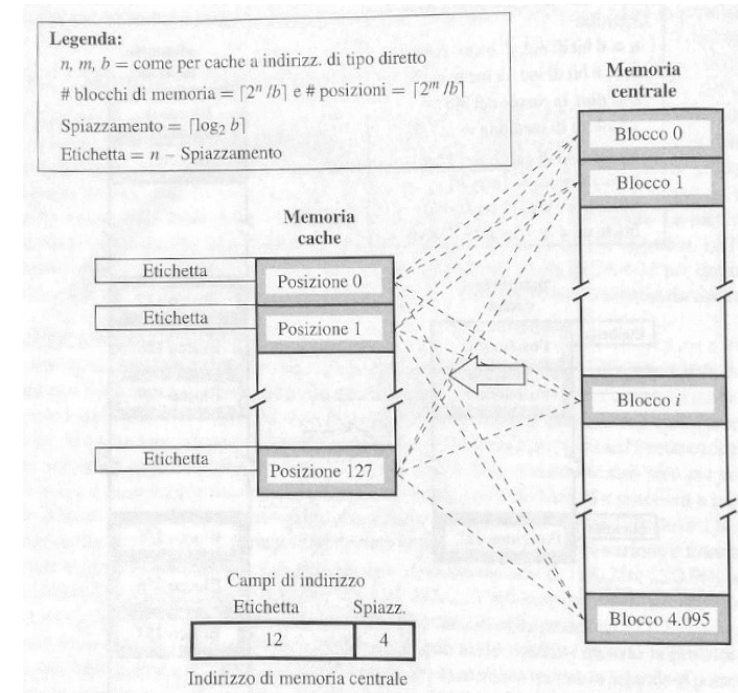
Memoria centrale: indirizzi da 16 bit (64K parole) pari a 4K blocchi da 16 parole

Cache: indirizzi da 11 bit (2K parole) pari a 128 posizioni da 16 parole

- Ogni blocco di memoria centrale è caricabile in una sola posizione di cache
- Numerando i blocchi di memoria in ordine a partire da 0, il blocco numero **i** è caricabile nella posizione di cache **i mod 128**
- Indirizzo di memoria divisibile in 3 campi:
 - **Spiazzamento [b0, b3]:** posizione della parola all'interno del blocco
 - **Blocco [b4, b10]:** posizione del blocco all'interno di un insieme di 128 blocchi
 - **Etichetta [b11, b15]:** posizione dell'insieme di blocchi all'interno della memoria
- A ciascuna posizione viene associata un'etichetta per riconoscere il blocco caricato

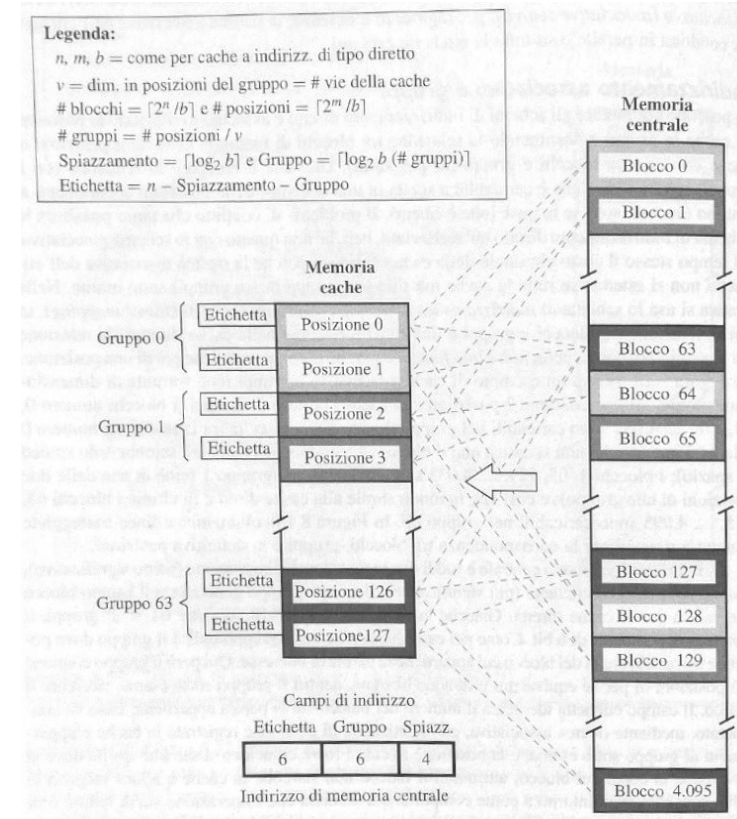


- Ogni blocco di memoria centrale è caricabile in qualsiasi posizione di cache
- Indirizzo di memoria divisibile in 3 campi:
 - **Spiazzamento [b0, b3]:** posizione della parola all'interno del blocco
 - **Etichetta [b4, b15]:** posizione del blocco all'interno in memoria
- A ciascuna posizione viene associata un'etichetta per riconoscere il blocco caricato
- È necessario un algoritmo di sostituzione per decidere quale posizione svuotare in caso di cache piena



Indirizzamento associativo a gruppi

- Ogni blocco di memoria centrale è caricabile in una sola posizione di cache
- Posizioni in cache divise in gruppi da v posizioni (cache a v vie)
- Numerando i blocchi di memoria in ordine a partire da 0, il blocco numero i è caricabile nella posizione di cache $i \bmod (128 / v)$
- Indirizzo di memoria divisibile in 3 campi come diretto:
 - **Spiazzamento** $[b0, b3]$, **Gruppo** $[b4, b9]$, **Etichetta** $[b10, b15]$:
- A ciascuna posizione viene associata un'etichetta per riconoscere il blocco caricato



- Negli indirizzamenti associativi bisogna scegliere quale posizione di cache liberare nel caso tutte le posizioni siano occupate
- Vari possibili algoritmi:
 - **LRU (least recently used):** sostituire il blocco usato meno di recente. Si assegna un contatore modulo v ad ogni posizione del gruppo
 - **Cache hit:** si azzerava il contatore della posizione interessata, si incrementano di 1 i contatori inferiori e si lasciano invariati quelli superiori
 - **Cache miss e gruppo non pieno:** si carica il blocco in una posizione vuota azzerandone il contatore e si incrementano di 1 gli altri contatori
 - **Cache miss e gruppo pieno:** si libera la posizione con contatore massimo, vi si carica il nuovo blocco azzerandone il contatore e si incrementano di 1 gli altri contatori
 - **FIFO (first in first out):** sostituire il blocco caricato meno di recente
 - **Casuale su distribuzione uniforme**

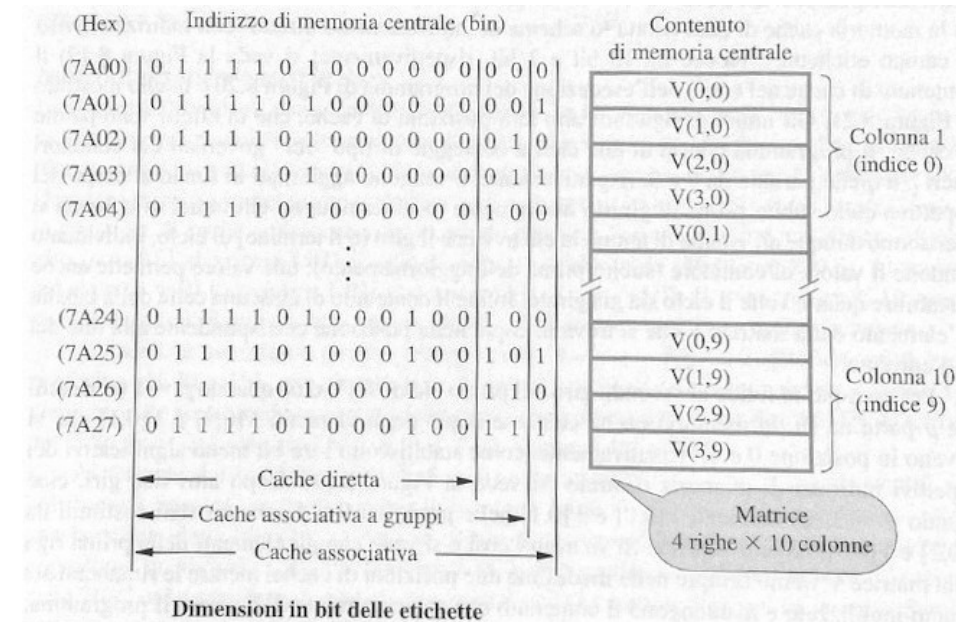
- **Problema:**

- Si prenda una matrice 4x10 **V** di interi
- La matrice si trova in memoria centrale in posizioni contigue disposta per colonne
- Eseguire il programma che aggiorni il contenuto della prima riga dividendone gli elementi per la loro media
- Mostrare il contenuto della cache di dato nel tempo

- **Dettagli sistema:**

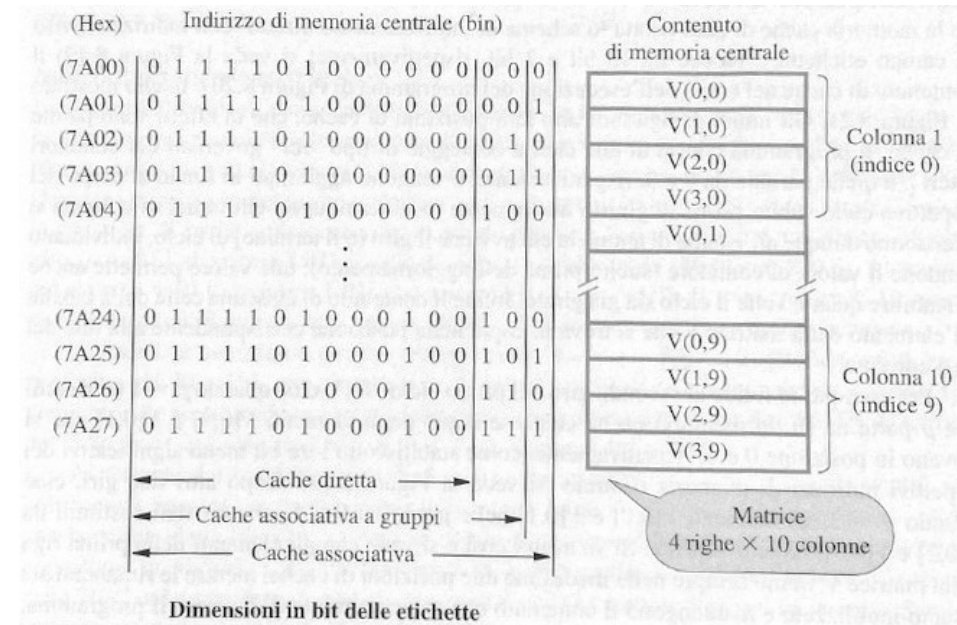
- Due cache (istruzioni e dati)
- Cache di dato con 8 posizioni
- Blocco da 1 parola da 16 bit
- Indirizzo di memoria da 16 bit
- Algoritmo di sostituzioni LRU

```
SOMMA := 0
for p := 0 to 9 do
    SOMMA := SOMMA + V(0, p)
end
MEDIA := SOMMA / 10
for q := 9 downto 0 do
    V(0, q) := V(0, q) / MEDIA
end
```



Esempio di indirizzamento diretto

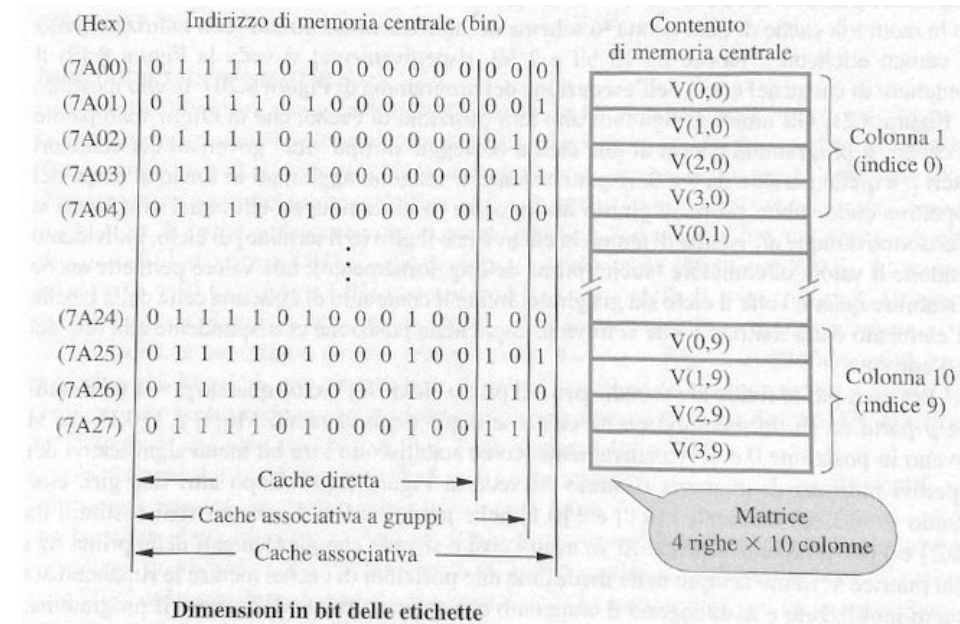
- Spiazzamento = \square
- Blocco = $[b0, b2]$
- Etichetta = $[b3, b15]$
- Gli elementi della prima riga sono associati solo alle posizioni 0 e 4
- Si ha una cache miss ogni passo del primo ciclo for e negli ultimi 8 passi del secondo
- 6 posizioni su 8 rimangono sempre vuote



Indice di conteggio di ciclo "for"									
Posizione	$p = 1$	$p = 3$	$p = 5$	$p = 7$	$p = 9$	$q = 6$	$q = 4$	$q = 2$	$q = 0$
0	V(0,0)	V(0,2)	V(0,4)	V(0,6)	V(0,8)	V(0,6)	V(0,4)	V(0,2)	V(0,0)
1									
2									
3									
4	V(0,1)	V(0,3)	V(0,5)	V(0,7)	V(0,9)	V(0,7)	V(0,5)	V(0,3)	V(0,1)
5									
6									
7									

Esempio di indirizzamento associativo

- Spiazzamento = \square
- Blocco = \square
- Etichetta = $[b0, b15]$
- Si ha una cache miss ogni passo del primo ciclo for e negli ultimi due passi del secondo
- Tutte le posizioni vengono riempite
- Si sfrutta il fatto che il secondo ciclo sia decrescente

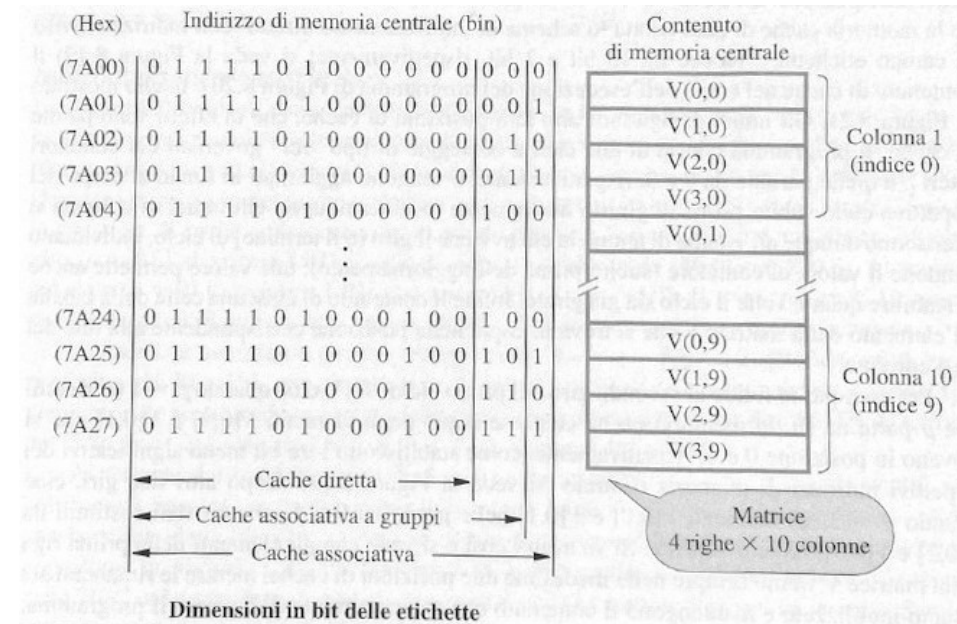


Indice di conteggio di ciclo

Posizione	$p = 7$	$p = 8$	$p = 9$	$q = 1$	$q = 0$
0	V(0,0)	V(0,8)	V(0,8)	V(0,8)	V(0,0)
1	V(0,1)	V(0,1)	V(0,9)	V(0,1)	V(0,1)
2	V(0,2)	V(0,2)	V(0,2)	V(0,2)	V(0,2)
3	V(0,3)	V(0,3)	V(0,3)	V(0,3)	V(0,3)
4	V(0,4)	V(0,4)	V(0,4)	V(0,4)	V(0,4)
5	V(0,5)	V(0,5)	V(0,5)	V(0,5)	V(0,5)
6	V(0,6)	V(0,6)	V(0,6)	V(0,6)	V(0,6)
7	V(0,7)	V(0,7)	V(0,7)	V(0,7)	V(0,7)

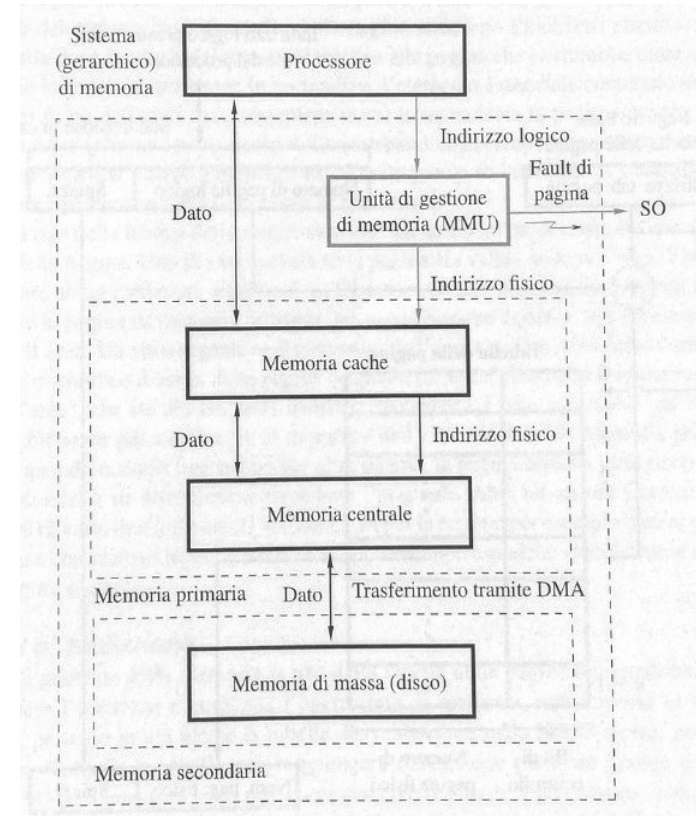
Esempio di indirizzamento associativo

- Spiazzamento = \square
- Blocco = $[b0]$
- Etichetta = $[b1, b15]$
- Cache a 4 vie
- Gli elementi della prima riga sono associati al primo gruppo
- Si ha una cache miss ogni passo del primo ciclo for e negli ultimi 6 passi del secondo
- 4 posizioni su 8 rimangono sempre vuote

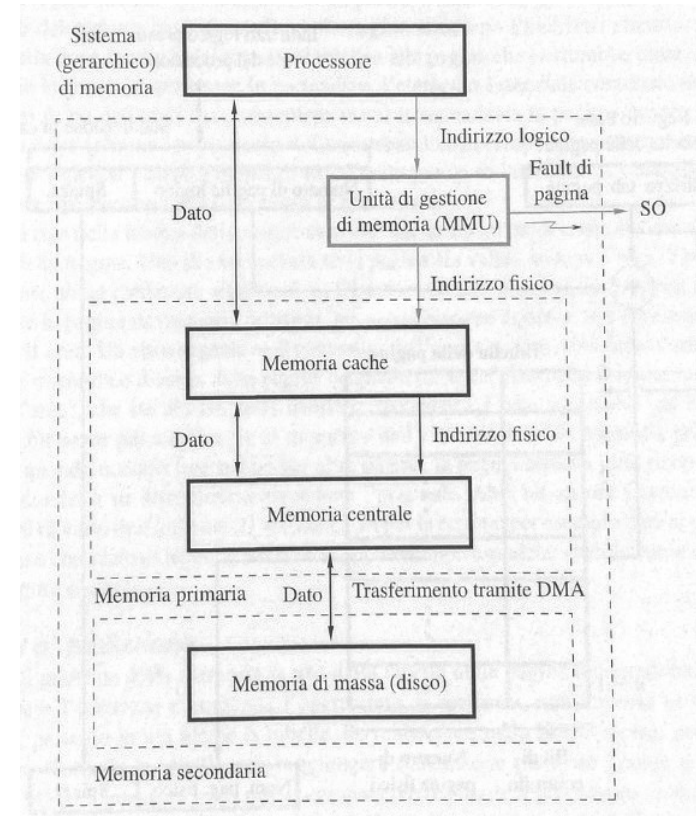


Indice di conteggio di ciclo "for"						Posizione
p = 3	p = 7	p = 9	q = 4	q = 2	q = 0	
Gruppo 0 {	V(0,0)	V(0,4)	V(0,8)	V(0,4)	V(0,0)	0
	V(0,1)	V(0,5)	V(0,9)	V(0,5)	V(0,1)	1
	V(0,2)	V(0,6)	V(0,6)	V(0,6)	V(0,2)	2
	V(0,3)	V(0,7)	V(0,7)	V(0,7)	V(0,3)	3
Gruppo 1 {						4
						5
						6
						7

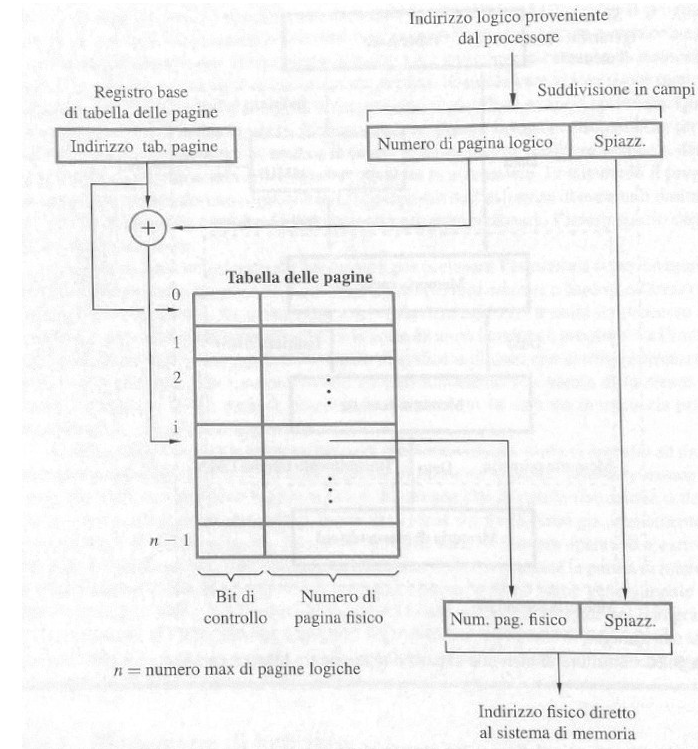
- Spesso la memoria centrale non è grande come lo spazio di indirizzamento del processore
- Solo le parti in uso del programma sono caricate in memoria centrale, mentre il resto risiede in memoria secondaria
- Il processore vede la memoria come un'entità unica (**memoria virtuale**) veloce come la cache e capiente come la memoria secondaria
- I blocchi sono trasferiti direttamente tra disco e memoria centrale tramite tecnica **DMA**



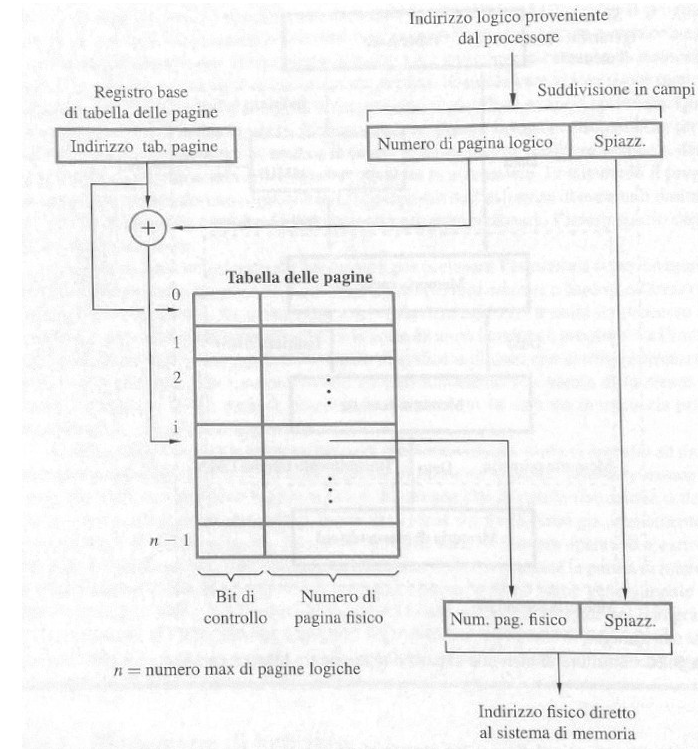
- **L'unità di gestione di memoria (MMU)** gestisce gli indirizzamenti tra processore e memoria
- L'MMU traduce gli indirizzi logici di memoria virtuale in indirizzi fisici
- Se il blocco (parola) non si trova in memoria centrale, l'MMU forza il sistema operativo a caricarla dal disco attivando il segnale **fault di pagina**
- Il fault di pagina è un'eccezione generata internamente al calcolatore



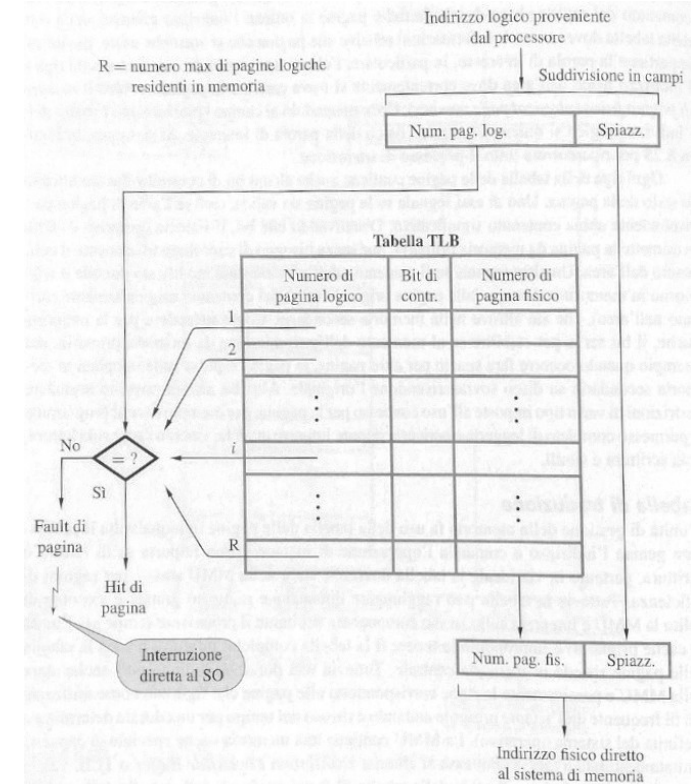
- L'unità elementare di informazione trasferibile tra memoria centrale e disco è chiamata **pagina**
- La dimensione di una pagina va da 2K a 16K parole
- La regione di memoria centrale capace di contenere una pagina è chiamata **area di pagina**
- L'indirizzo logico viene diviso in:
 - **Spiazzamento**: posizione della parola all'interno della pagina
 - **Numero di pagina logico**: posizione della pagina nella memoria virtuale



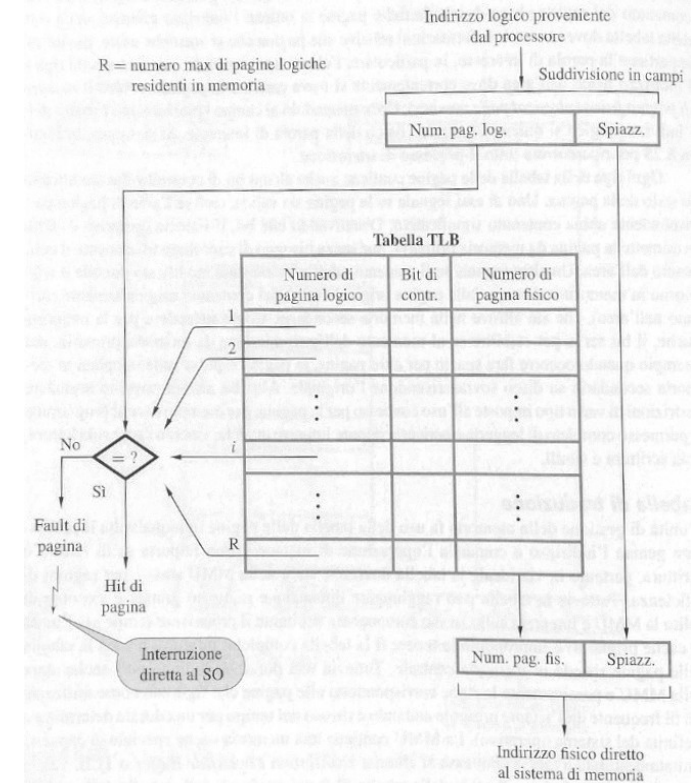
- La **Tabella delle pagine** è usata per generare l'indirizzo fisico
- Contiene un campo per ogni numero di pagina logico
- Ogni campo è formato da dei bit di controllo (bit di validità, bit di modifica, permessi di accesso, etc.) e il numero di pagina fisico
- Il **registro di base di tabella di pagina** contiene l'indirizzo al primo elemento della tabella
- L'MMU ritrova il numero di pagina fisico nel campo della tabella con **indirizzo = registro di base di tabella + numero di pagina logico**



- L'MMU è integrata nel processore
- La tabella delle pagine solitamente risiede in memoria centrale
- L'MMU possiede una cache contenente il **Translation Lookaside Buffer (TLB)**
- Il TLB contiene una copia delle righe della tabella delle pagine usate più di recente
- Ciascun campo del TLB contiene il numero di pagina logico, quello fisico e i bit di controllo
- Il SO si occupa di assegnare i bit di validità e permanenza in TLB delle pagine

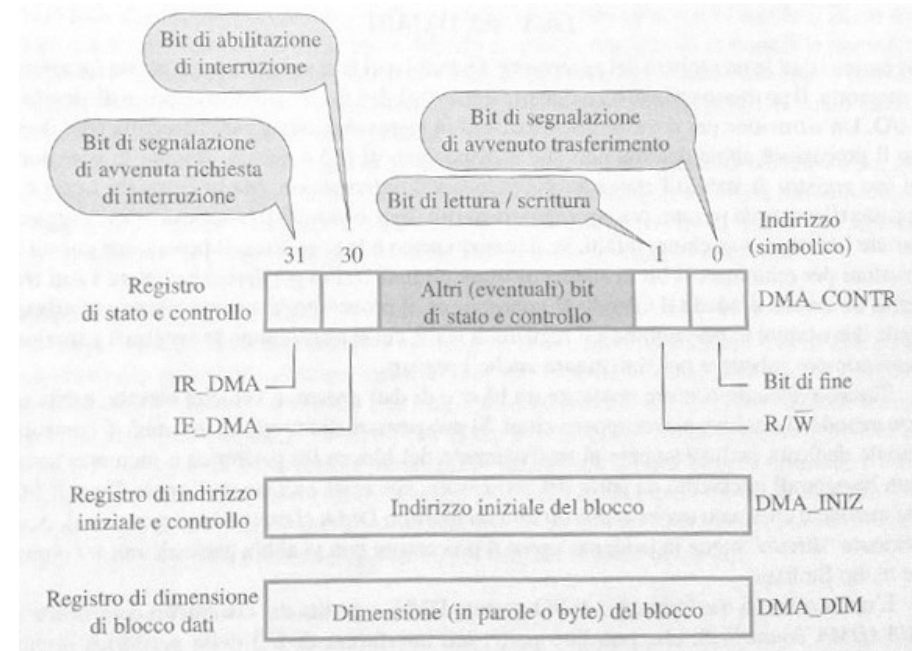


- **TLB hit:** il numero di pagina cercato si trova nel TLB e l'indirizzo fisico viene creato
- **TLB miss:** il numero di pagina non si trova nel TLB e deve essere caricato dalla tabella delle pagine
- **Fault di pagina:** la pagina non si trova in memoria centrale (bit di validità a 0) e deve essere caricata dal disco
- Il fault di pagina genera un'interruzione bloccando l'esecuzione dell'istruzione
- Nel caso la memoria centrale sia piena si usano tecniche di sostituzione simili a quelle della cache

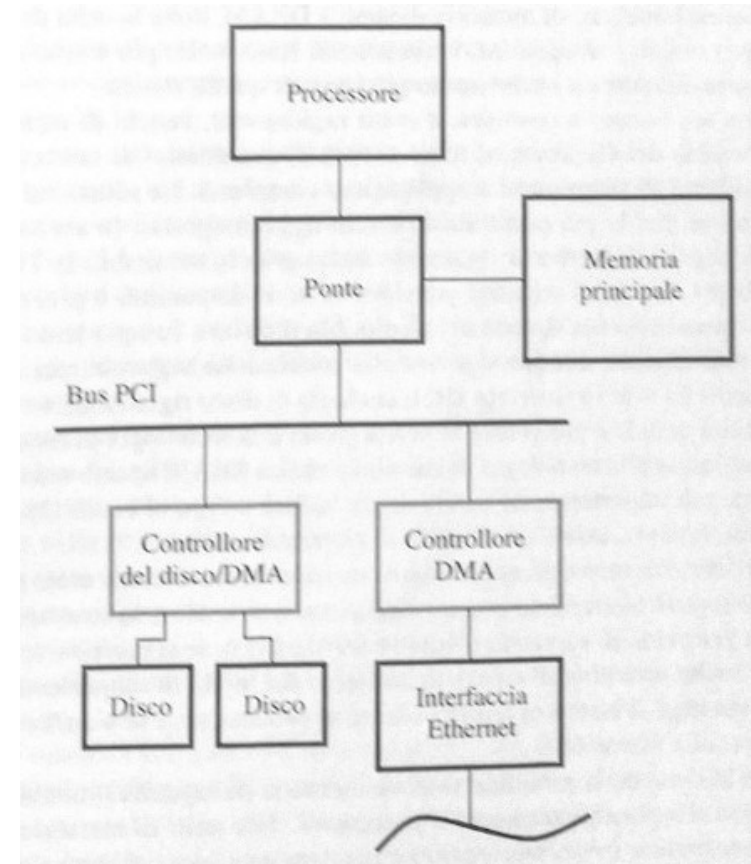


- **Direct Memory Access (DMA):** tecnica di trasferimento che permette ad un dispositivo di I/O di interagire con la memoria indipendentemente dal processore
- **Controllore DMA:** componente collegata al bus da un lato e alla periferica dall'altro (spesso parte dell'interfaccia I/O). Gestisce il trasferimento di grossi blocchi di dato tra periferiche e memoria centrale
- Ogni trasferimento è inizializzato tramite appositi registri
- Durante il trasferimento il processore non interviene (libero di eseguire istruzioni)
- Il controllore DMA può generare un'interruzione a trasferimento concluso

- **Registro di controllo:** contiene i bit di controllo del trasferimento
 - Bit di fine
 - R/W
 - Bit di abilitazione interruzioni
 - Bit di richiesta di interruzione
- **Registro di indirizzo:** contiene l'indirizzo iniziale del buffer di memoria per il blocco dati
- **Registro di dimensione:** dimensione del buffer di memoria per il blocco dati



- Esempio di controllori DMA collegati ad un bus PCI
- Ci sono 3 canali DMA (due per il controllore disco e uno per quello ethernet)
- Ogni canale possiede i suoi registri distinti
- Il controllore del disco gestisce i due dischi indipendentemente



- Le prestazioni della memoria sono dipendenti dalla frequenza delle cache hit:
 - **Tasso di hit (hit rate):** $h = \text{numero di hit} / \text{numero di accessi}$
 - **Tasso di miss (hit rate):** $(h - 1) = \text{numero di miss} / \text{numero di accessi}$
- Per valutare le prestazioni della memoria si può calcolare il **tempo medio di accesso:**

$$\Delta t_{\text{accesso}} = hH + (1 - h)M$$

- H = tempo di accesso alla cache,
- M = tempo di accesso alla memoria centrale (**penalità di miss**)

- Si prenda in considerazione un processore con le seguenti statistiche:

- Tempo di accesso alla cache: $H = \tau$
- Penalità di miss: $M = 19\tau$
- Percentuale di istruzioni di accesso alla memoria: **30%**
- Tasso di hit per le istruzioni: $h_i = 0.95$
- Tasso di hit per i dati: $h_d = 0.9$

- Il tempo medio di accesso sarà:

$$\Delta t_{\text{accesso}} = 1 \cdot (h_i H + (1 - h_i) M) + 0.3 \cdot (h_d H + (1 - h_d) M) = 1 \cdot (0.95\tau + 0.05 \cdot 19\tau) + 1 \cdot (0.9\tau + 0.1 \cdot 19\tau) = 2.74\tau$$

- Stima aumento del tempo di accesso rispetto al caso ottimo:

$$\Delta t_{\text{accesso}} / \text{tempo ideale} = 2.74\tau / 1.3\tau = 2.1$$

- Si può generalizzare il tempo medio di accesso per processori con 2 livelli di cache:

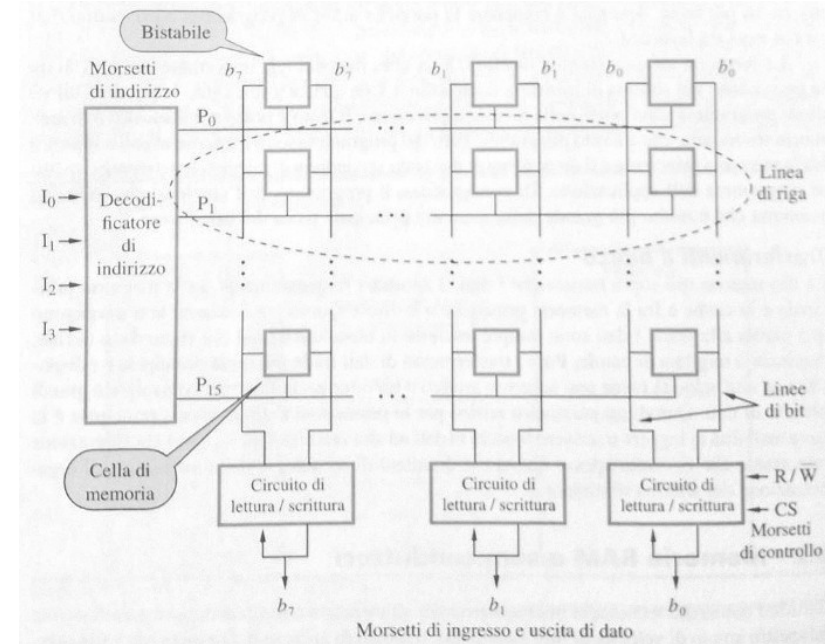
$$\Delta t_{\text{accesso}} = h_1 H_1 + (1 - h_1) \cdot (h_2 H_2 + (1 - h_2) M)$$

- h_1 = tasso di hit cache L1
- h_2 = tasso di hit cache L2
- H_1 = tempo di accesso alla cache L1
- H_2 = tempo di accesso alla cache L2
- M = tempo di accesso alla memoria centrale (**penalità di miss**)

- **Buffer di scrittura (write buffer):**
 - Buffer usato per salvare temporaneamente il valore da scrivere in memoria
 - Il contenuto del buffer verrà scritto in memoria successivamente
 - La CPU non deve attendere durante la scrittura immediata
- **Caricamento anticipato (prefetch):**
 - Vengono caricati anticipatamente in cache blocchi di memoria
 - Via software tramite istruzioni apposite che simulano cache miss
 - Via hardware tramite circuiti di controllo appositi
- **Cache non bloccante (lockup-free cache):**
 - Cache che non si blocca durante il caricamento di blocchi dovuti a miss
 - Sistema di controllo più complesso per gestire miss multiple

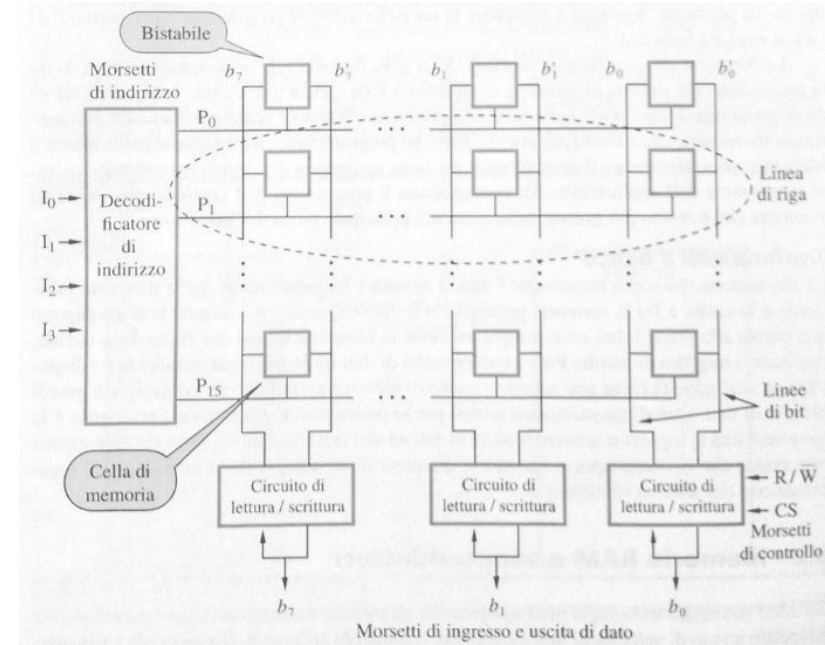
- Nelle **memorie ad accesso casuale (RAM)** si può accedere ad ogni parola nello stesso tempo costante (non importa la posizione in memoria)
- Sono realizzate con **tecnologie microelettroniche a semiconduttori**
- Il tempo di accesso va dai **100 ns** a **1 ns**
- Sono **volatili** (mantengono il contenuto fintanto che sono alimentate)
- Usate per realizzare i livelli della memoria primaria (cache e memoria centrale)
- Gruppi di celle da un bit formano il **componente integrato di memoria (memory chip)**
- Gruppi di memory chip formano il **banco di memoria (memory file o bank)**

- Nel **memory chip** le celle da un bit sono organizzate a matrice
- Le righe rappresentano le parole
- Le celle di una riga sono collegate ad una **linea di parola** comune
- Le celle di una colonna sono collegate a **due linee dati** comuni

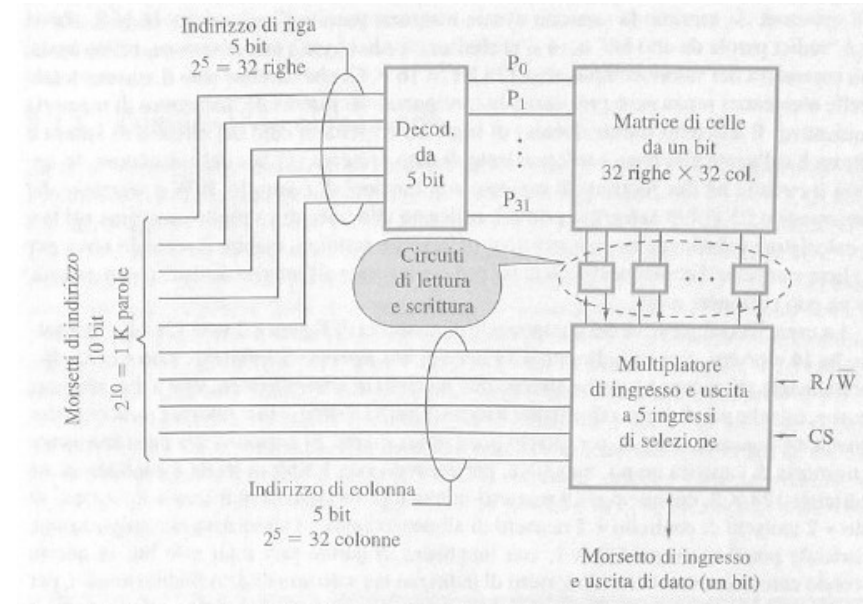


Componente integrato di memoria

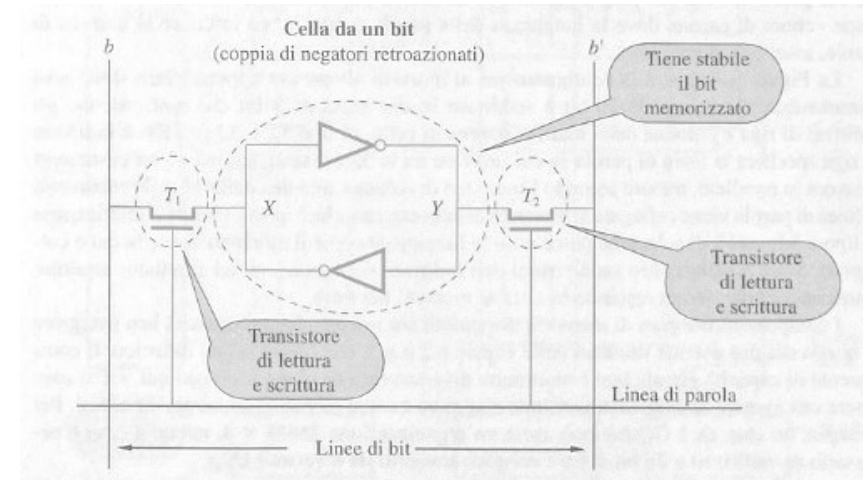
- Il **decodificatore di indirizzo** seleziona la linea di parola da attivare
- I **circuiti bidirezionali di lettura/scrittura** collegano i singoli bit ai morsetti di ingresso/uscita
- Il segnale **R/W** è usato per selezionare l'operazione di accesso
- Il segnale **CS** è usato per attivare il memory chip all'interno di un blocco di memoria
- In figura un memory chip da **sedici parole da otto bit (16x8)** con 16 morsetti (4 indirizzo, 8 di dato, 2 di controllo e 2 di alimentazione)



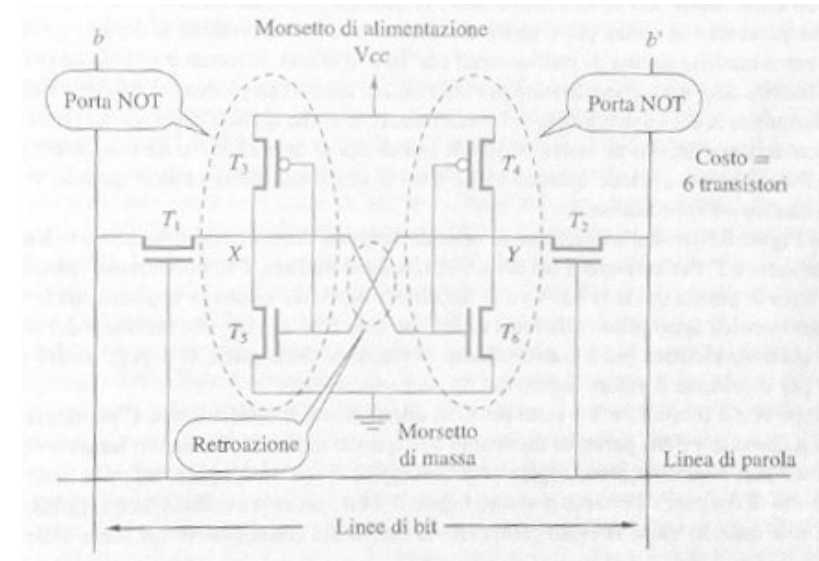
- La struttura interna del memory chip non sempre rispecchia quella dei morsetti nel circuito integrato
- In figura si ha un memory chip **1K x 1** che internamente è strutturato con una matrice **32x32**
- 10 morsetti di indirizzo usati sia per le righe che per le colonne
- **Multiplicatore di ingresso e uscita** bidirezionale che seleziona 1 dei 32 bit di parola



- La **memoria statica (SRAM)** è una memoria volatile che mantiene il suo stato fintanto che viene alimentata
- È realizzata da una **coppia di negatori retroazionati** collegati alle due linee di bit da due transistori
- La linea di parola mette in conduzione o interdizione i transistori
- **Linea di parola = 0:**
 - cella isolata mantiene il suo valore
- **Linea di parola = 1:**
 - Lettura: circuito di lettura legge le linee di dato
 - Scrittura: circuito di scrittura forza il valore sulle linee di dato

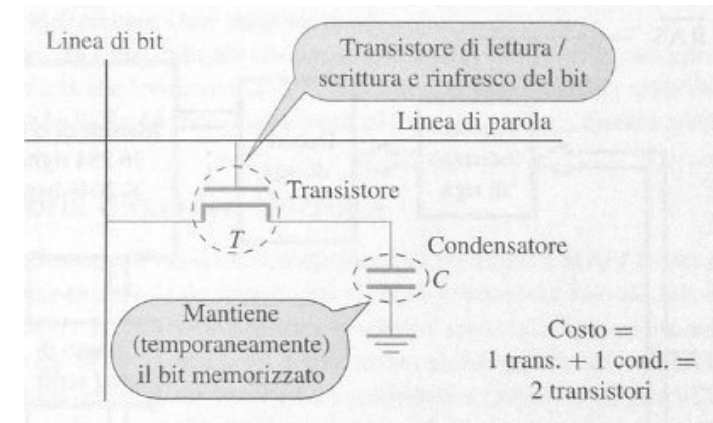


- Una cella di memoria SRAM si può realizzare con 2 **negatori CMOS**
- **Consumi molto bassi** (fluisce corrente solo durante lettura e scrittura)
- Quando la linea di parola è a zero non esiste continuità elettrica tra massa e alimentazione

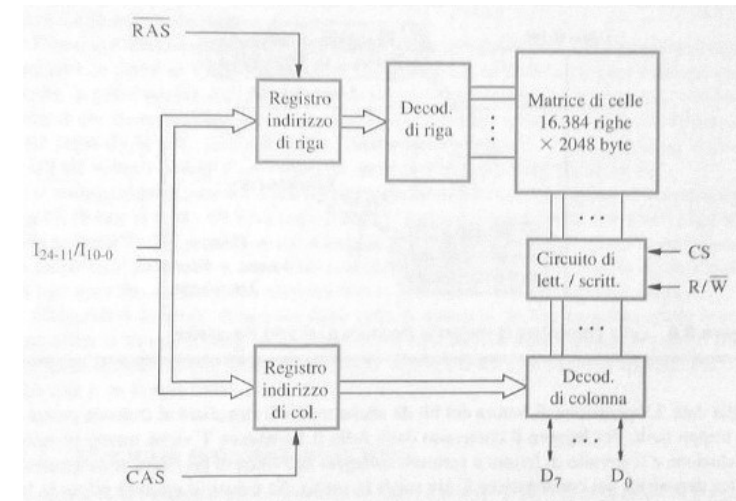


- Le SRAM sono costose (6 transistori per realizzarle)
- Le **memorie ad accesso casuale dinamiche (DRAM)** sono memorie meno costose, ma che perdono il loro stato dopo poco tempo seppure alimentate
- Lo stato di una cella è rappresentato come **carica elettrica di un condensatore**
- La carica del condensatore si dissipa dopo **10-100 ms**
- La carica va **rinfrescata (refreshed)** periodicamente

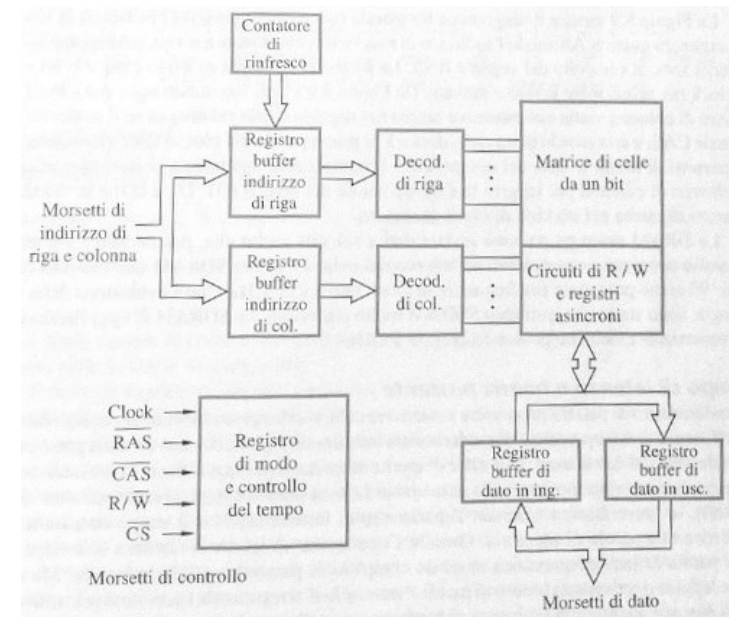
- La cella è formata da un **condensatore** collegato ad una linea di bit tramite un **transistore**
- Il transistore è controllato da la linea di parola
- Quando il condensatore è **isolato (linea di parola = 0)** mantiene la sua carica fino alla dissipazione
- Quando il condensatore è **collegato (linea di parola = 1)**:
 - Lettura: il circuito di lettura e scrittura legge e rinfresca il contenuto della linea di bit
 - Scrittura: il circuito di lettura e scrittura aggiorna il contenuto della linea di bit
- **Basso costo** (2 componenti)



- Nelle **DRAM asincrone** il controllo è completamente gestito da un **circuito di controllo esterno al chip**
- In figura un memory chip da 32M x 8 con configurazione interna di 16K x 16K
- **Due registri di ingresso per gli indirizzi** di riga e colonna attivati dai segnali **RAS (Row Address Strobe)** e **CAS (Column Address Strobe)**
- Possibilità di ridurre il numero di morsetti moltiplicando i morsetti di indirizzo
- **Fast page mode:** blocchi di dati appartenenti alla stessa riga inviati più velocemente (linea di parola attivata solo una volta)

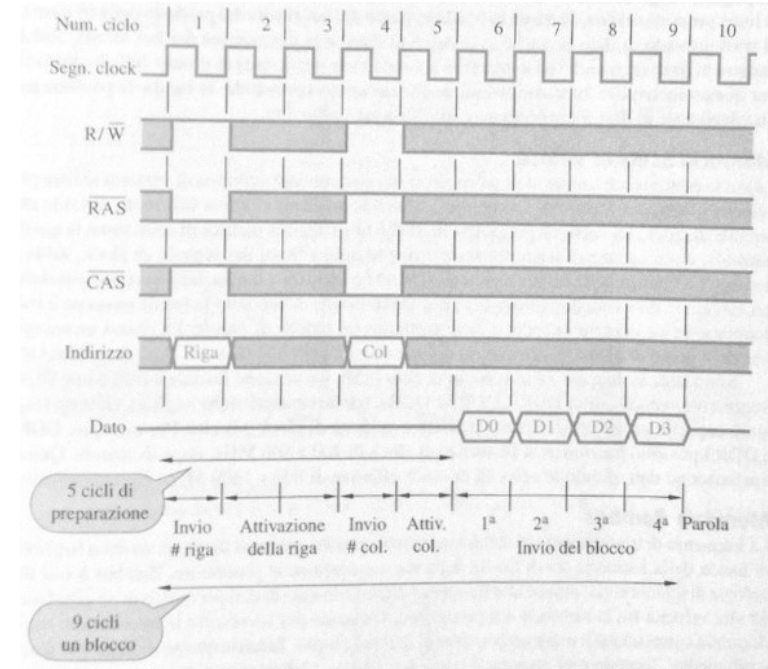


- Nelle **DRAM sincrone (SDRAM)** il controllo è scandito da un ciclo di clock ed è gestito da un **circuito di controllo integrato nel chip**
- Il rinfresco delle linee di celle è gestito da un **contatore di rinfresco**
- **Registri buffer di ingresso e uscita** sono usati per velocizzare il trasferimento
- Un **registro di modo** usato per selezionare la modalità di funzionamento (per esempio trasferimento di blocchi di una certa lunghezza)



Esempio lettura DRAM sincrona

- Esempio di lettura di un blocco da 4 parole con memoria SDRAM
- Inizialmente viene selezionata la riga **campionando l'indirizzo di riga** in risposta al segnale **RAS attivo**
- Dopo alcuni cicli per l'attivazione della riga, la prima parola viene selezionata **campionando l'indirizzo di colonna** in risposta al segnale **CAS attivo**
- Dopo 1 ciclo di clock la prima parola viene mandata in uscita seguita dalle altre nei 3 cicli di clock successivi



- Le SDRAM standard sono sincronizzate sul fronte di salita del ciclo di clock
- Le **SDRAM a frequenza doppia (DDR SDRAM)** selezionano le righe sul fronte di salita, ma **trasferiscono i dati su entrambi i fronti**
- **La latenza** (tempo di setup ed invio di un singolo dato) delle DDR SDRAM **è uguale** a quella delle SDRAM standard
- **La banda passante** (frequenza di trasferimento dati) delle DDR SDRAM **è doppia** rispetto a quella delle SDRAM standard
- Esistono nuove versioni ancora più performanti (**DDR2, DDR3 e DDR4**)

- I memory chip possono essere combinati in **banchi di memoria** più capienti
- Struttura matriciale simile a quella dei singoli chip
- La **linea di parola** viene collegata al **morsetto CS** di ciascun chip, attivandolo o disattivandolo a seconda del segnale
- I **bit più significativi** dell'indirizzo vengono decodificati ed **usati per selezionare la riga**
- In figura ogni chip fornisce 8 bit della parola da 32 bit del banco di memoria

