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

Sistemi di memoria

Nino Cauli

UNIVERSITÀ
degli STUDI
di CATANIA

Dipartimento di Matematica e Informatica

Memoria

- 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**

Memoria primaria

- 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)

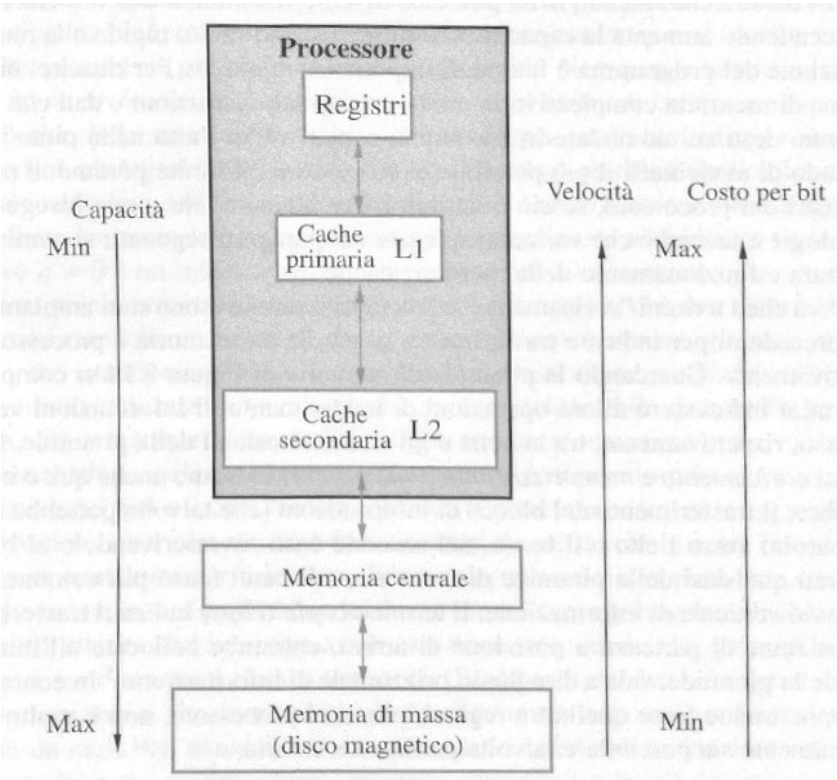
- **CACHE**: livello più alto (molto veloce, integrata nel processore)

Memoria secondaria

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

Gerarchia di memoria

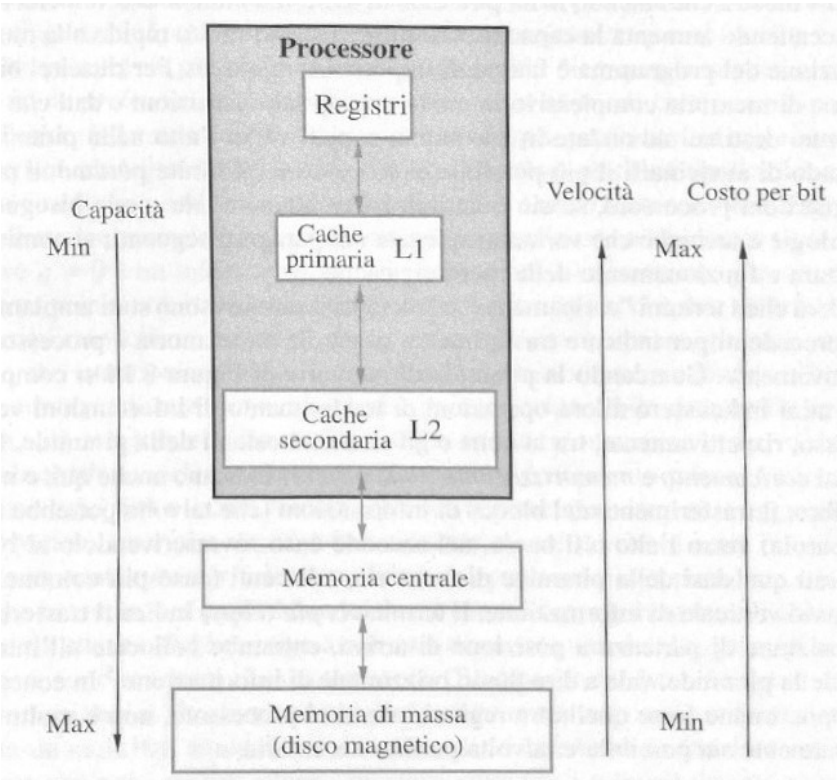
- La gerarchia di memoria è organizzata a piramide
- Tecnologia di memorizzazione più performante = maggior costo
- Livelli di memoria in cima alla piramide più piccoli e veloci
- I programmi sono immagazzinati nella memoria di massa e solo le loro porzioni attive vengono caricate nei livelli più alti



Gerarchia di memoria

I livelli di memoria sono i seguenti:

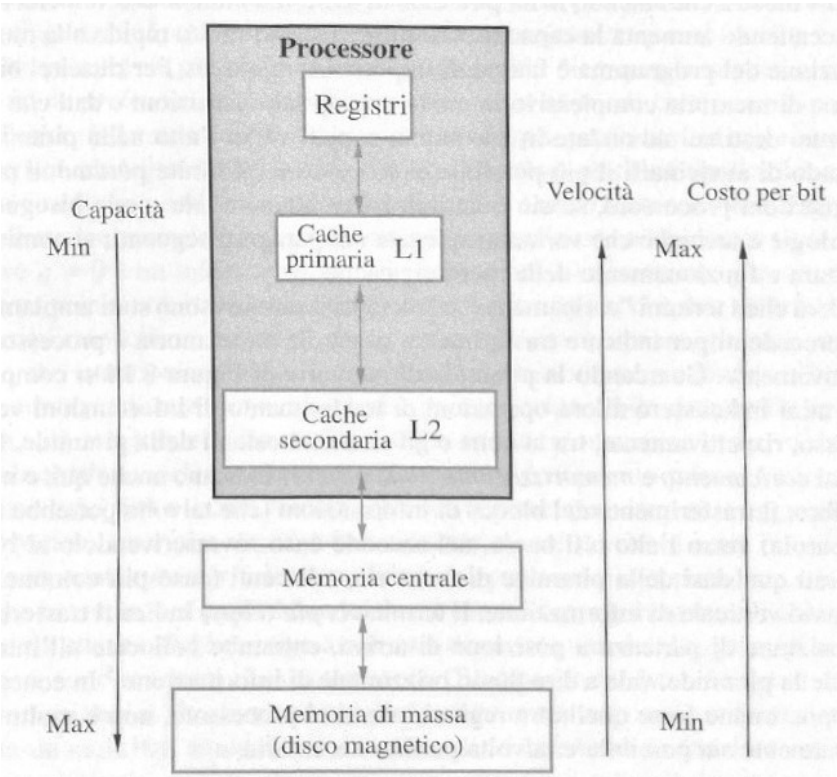
- **Registri:** velocissimi, capacità molto ridotta, integrati nel processore, flip-flops
- **Livelli di Cache (L1, L2):** molto veloci, capacità ridotta (decine di KB / qualche MB), integrati nel processore, SRAM
- **Memoria centrale:** veloce, capacità media (qualche GB), DRAM
- **Memoria di massa:**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



Gerarchia di memoria

Operazioni tra livelli di memoria:

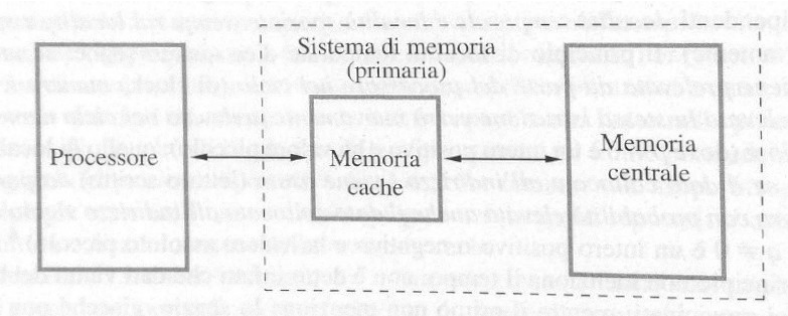
- **LOAD (caricamento):** trasferimento dei dati verso l'alto
- **STORE (memorizzazione):** trasferimento dei dati verso il basso, sovrascrivendo il dato originario
- **COPY (copia):** trasferimento "orizzontale" dei dati all'interno dello stesso livello di memoria



Memoria cache

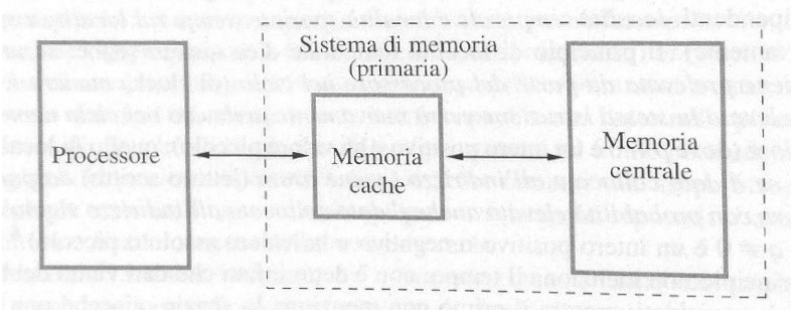
- La memoria Cache è una memoria piccola e veloce interposta tra memoria centrale e processore, che contiene copie di istruzioni e dati della memoria centrale da usare al momento

- 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 ± 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



Memoria cache

- 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



Chache hit

- 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

Chache miss

- 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 chi il blocco sia caricato sulla cache e poi la parola viene modificata in cache marcando la posizione come modificata

Schemi di indirizzamento

- 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)

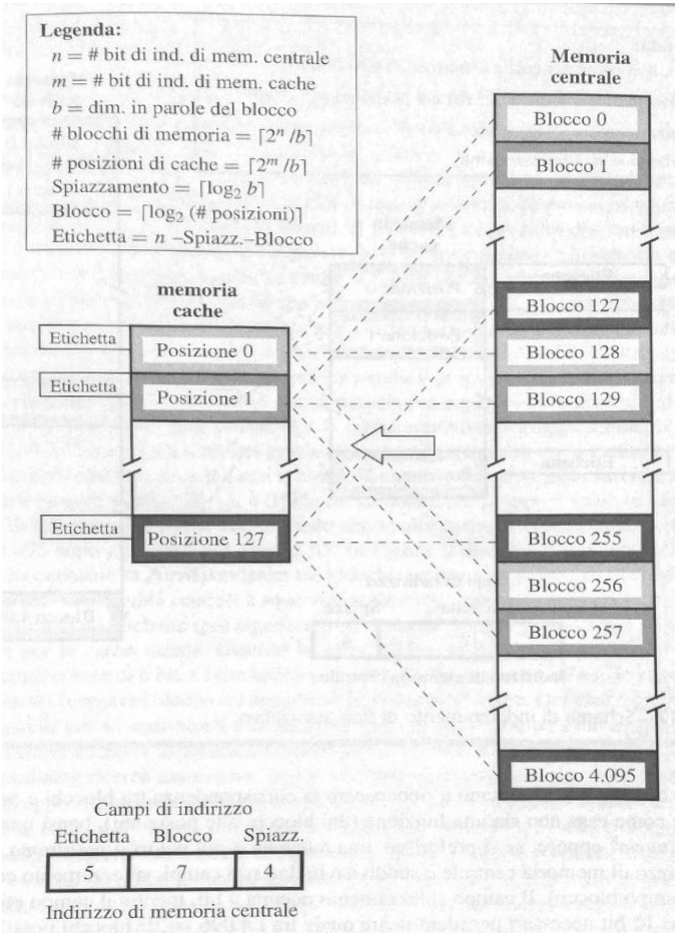
Esempio:

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

Indirizzamento diretto

- 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



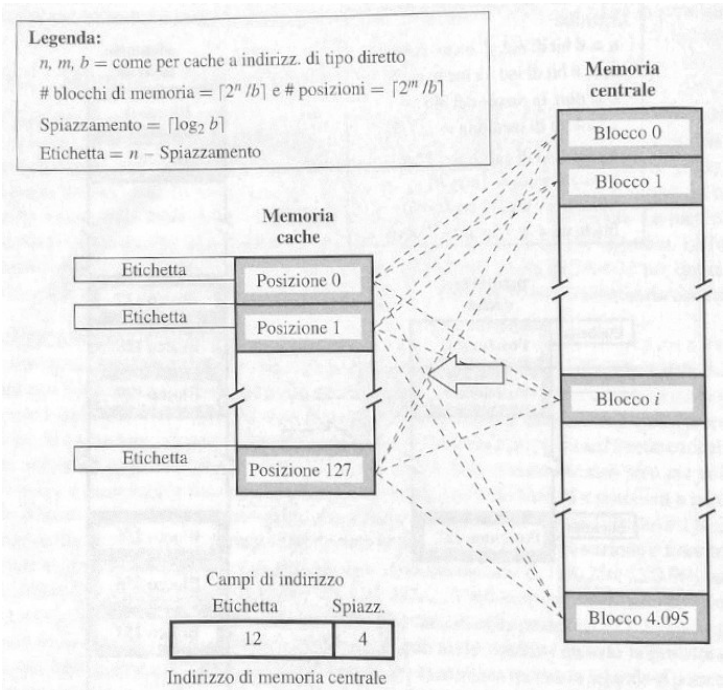
Indirizzamento associativo

- 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

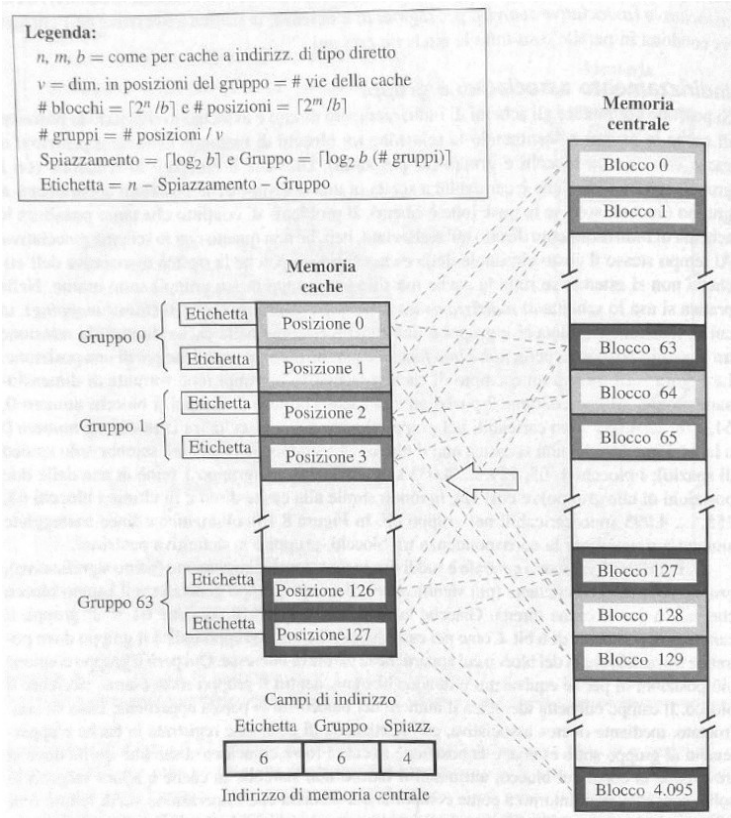
- A ciascuna posizione viene associata un'etichetta per riconoscere il blocco caricato

- 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 **mod (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



Algoritmo di sostituzione

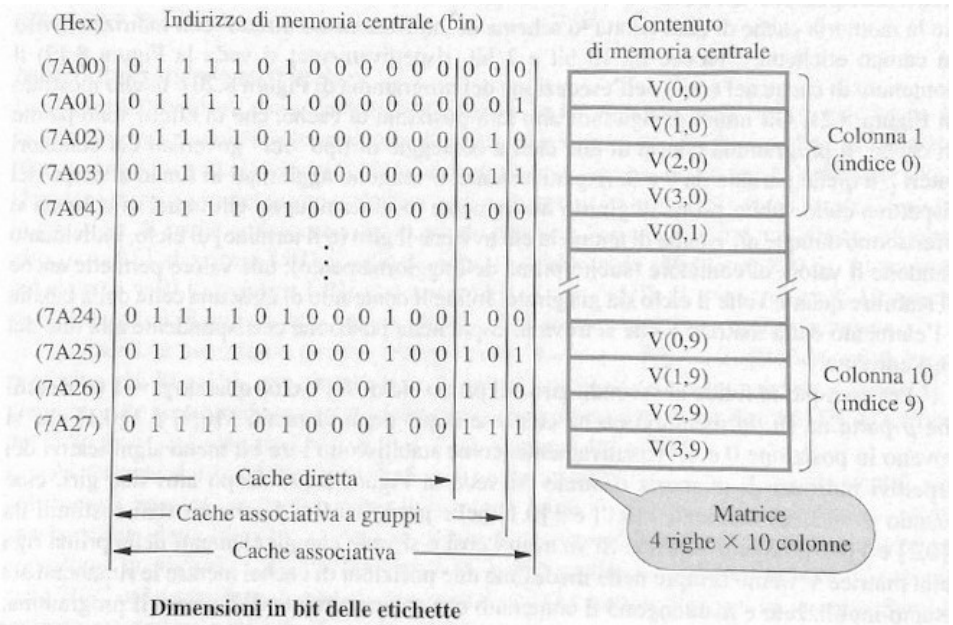
- Negli indirizzamenti associativi bisogna scegliere quale posizione di cache liberare nel caso tutte le posizioni siano occupate

- 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 azzera 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**

Esempio di indirizzamento

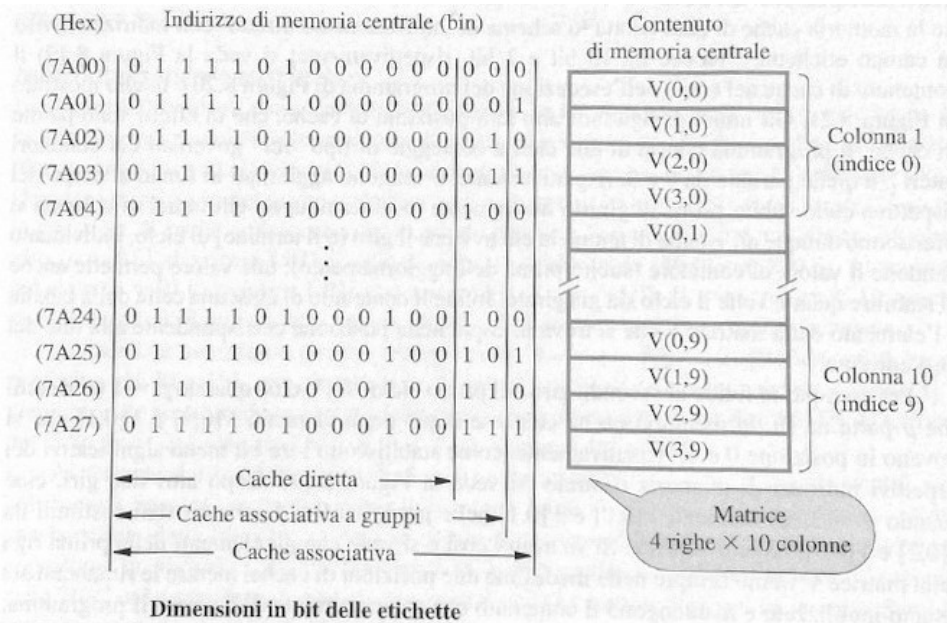
- **Problema:**
  - Si prenda una matrice 4x10 **V** di interi
  - La matrice si trova in memoria centrale in posizioni contigue disposta per colonne
  - Eseguire il programma che aggiorni il contenuto della prima riga dividendone gli elementi per la loro media
  - Mostrare il contenuto della cache di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

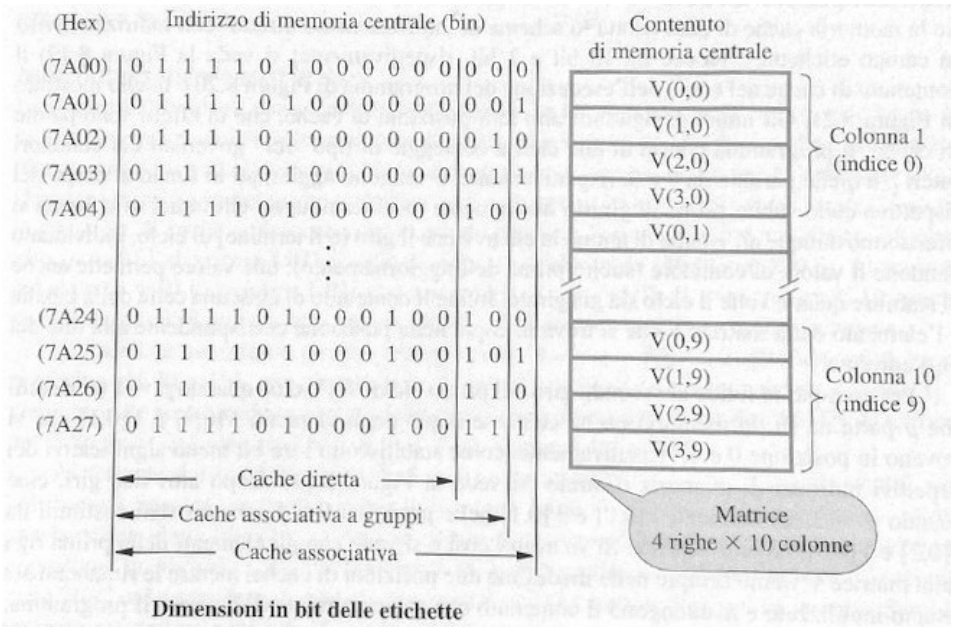


Indice di conteggio di ciclo "for"

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 = []
- Blocco = []
- 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

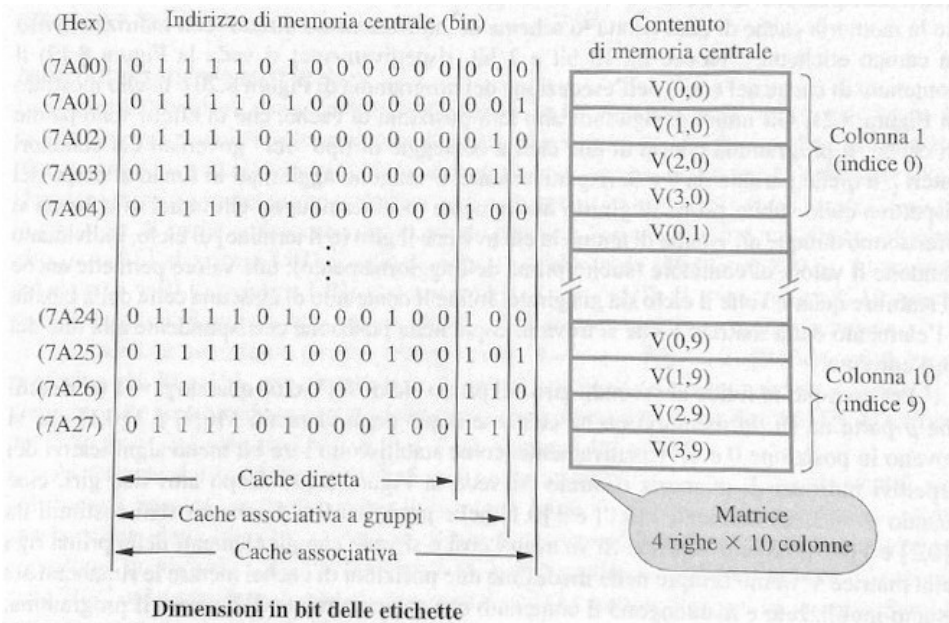


Indice di conteggio di ciclo

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 = []
- 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

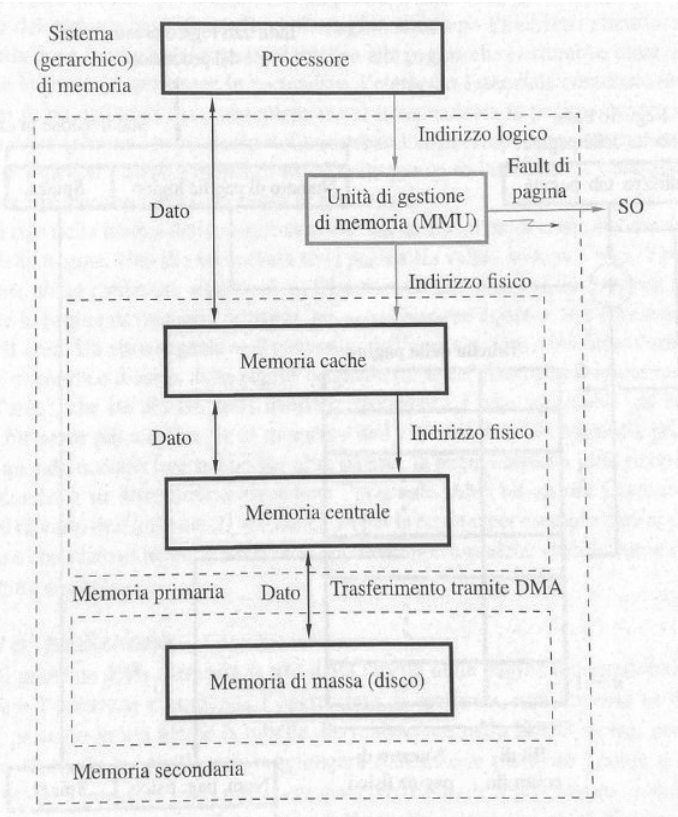


Indice di conteggio di ciclo "for"

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

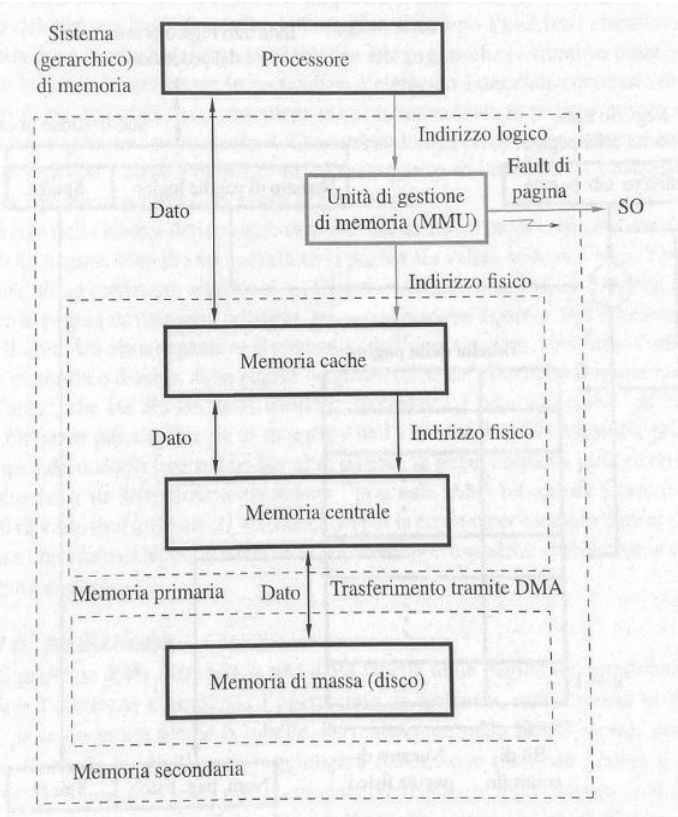
Memoria virtuale

- 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**



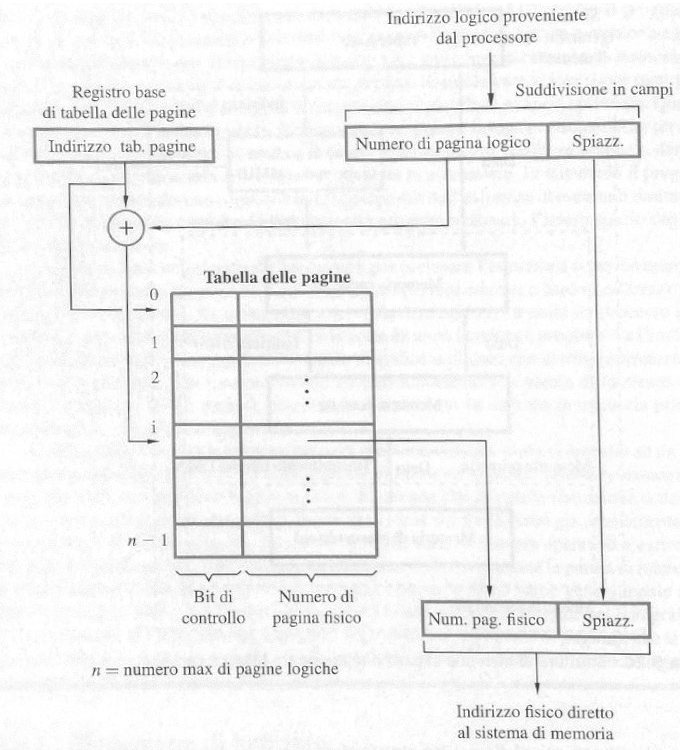
Memoria virtuale

- **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 falut di pagina è un'eccezione generata internamente al calcolatore



Traduzione di indirizzo

- 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



Traduzione di indirizzo

- 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**

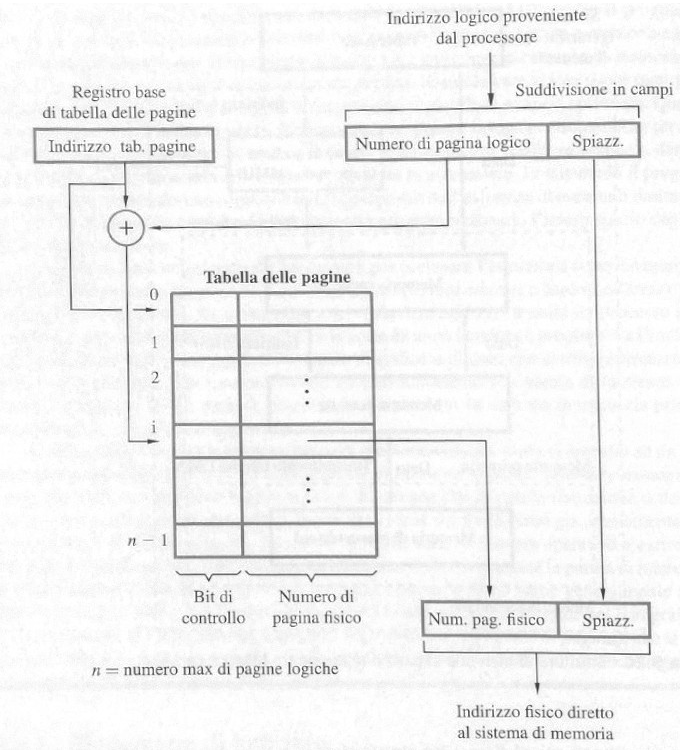
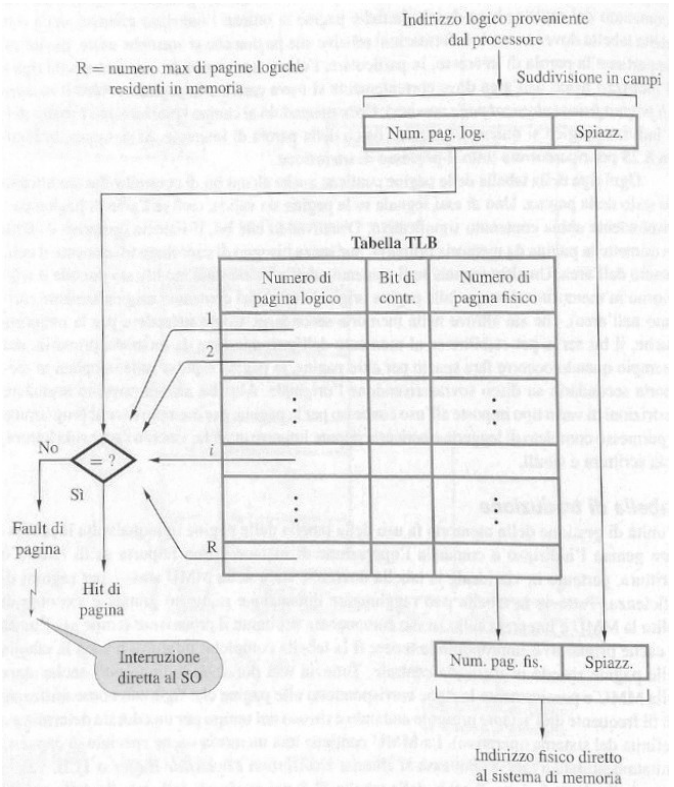


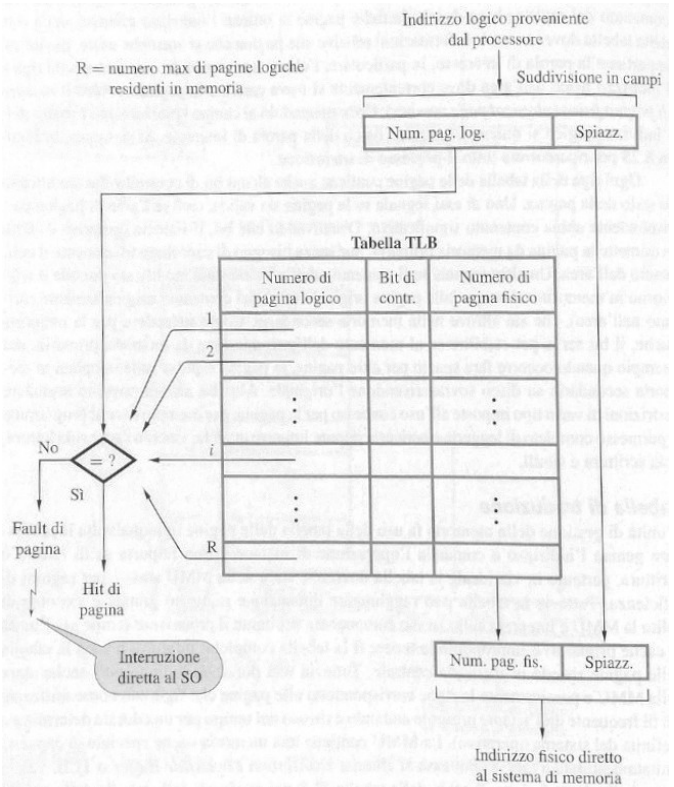
Tabella di traduzione

- 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



Fault di pagina

- **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

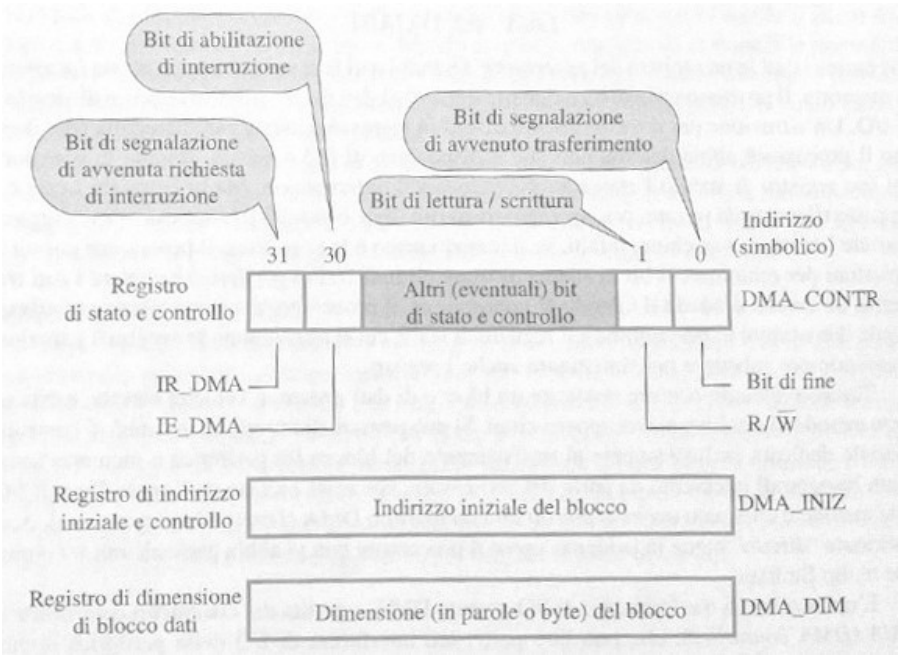


Direct Memory Access (DMA)

- **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

Registri DMA

- **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



Controllore DMA

- 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