



# FONDAMENTI DI INFORMATICA

## Lezione n. 14

- **GESTIONE DELLA GERARCHIA**
- **CONCETTO DI MEMORIA VIRTUALE**
- **RAGIONI PER L'USO DELLA MEMORIA VIRTUALE**
- **MECCANISMO DI GENERAZIONE DEGLI  
INDIRIZZI FISICI**
- **SEGMENTI, PAGINE**
- **ALGORITMI DI ALLOCAZIONE E POLITICHE DI  
SOSTITUZIONE**
- **ESEMPI DI MEMORIA VIRTUALE**
- **ORGANIZZAZIONE DI UNA MEMORIA CACHE**
- **MAPPING DEGLI INDIRIZZI**
- **AGGIORNAMENTO DELLA MEMORIA**



## USO DELLA GERARCHIA

**Per un uso efficiente della gerarchia è necessario stabilire:**

- **Quando effettuare la sostituzione dei blocchi.**
- **La dimensione del blocco da trasferire.**
- **Quale politica di allocazione utilizzare.**

**Queste caratteristiche devono essere specificate per ogni tipo di gerarchia.**

**Non esiste alcuna soluzione ottime per qualunque classe di applicazioni o per qualunque programma.**

**Per valutare le alternative occorre fare riferimento a un insieme di applicazioni (*benchmarking*) che siano significative a rappresentare il maggior numero di applicazioni.**



# POLITICHE DI SOSTITUZIONE

**La sostituzione può essere effettuata:**

- **A RICHIESTA** - in caso di miss si richiamano nuovi dati dal secondo livello.
- **in modo PREVENTIVO** - (molto difficile) es. prefetching.

**Il meccanismo di trasferimento a richiesta è l'unico utilizzato in questi tipi di gerarchia.**

**Un uso implicito del meccanismo di tipo preventivo consiste nel fatto che non si trasferisce mai un solo dato ma un intero blocco, e quindi si fa l'ipotesi (località spaziale) che tutti i dati così trasferiti saranno di utilità.**



## DIMENSIONE DEL BLOCCO

**La dimensione del blocco da trasferire deve tenere conto:**

- **rapporto fra tempo di trasferimento effettivo e overhead.**
- **trasferimento di dati inutili.**

**E' necessario un compromesso fra la necessità di avere blocchi grandi per diminuire gli *overhead* sul tempo di trasferimento e quella di avere blocchi piccoli per ridurre la probabilità di trasferire dati che non verranno utilizzati.**

**Per dimensionare l'ottimo si ricorre alla sperimentazione. I valori attualmente in uso variano da 512 bytes a qualche Kbytes.**



## POLITICA DI ALLOCAZIONE

- **STATICA** - Ogni blocco è posto in posizione predeterminata.
- **DINAMICA** - Si assegna la quantità di memoria necessaria nella zona momentaneamente libera.
- **SOSTITUZIONE FORZATA** - Si espelle da  $M_1$  una area scelta per ragioni opportune (preemption).
- **SENZA ALLOCAZIONE FORZATA** - Si alloca la nuova area solo se si individua uno spazio libero di dimensioni sufficienti.

**Il meccanismo con sostituzione forzata è quello maggiormente impiegato.**



## THRASHING

**Quando la politica di gestione della gerarchia non è efficiente si verificano troppi insuccessi (o miss), si ha il fenomeno di:**

***Thrashing***

**La CPU spende il proprio tempo ad allocare e disallocare aree di memoria senza più svolgere lavoro utile.**

**Questo fenomeno può inficiare le prestazioni di un sistema ed è dovuto a un cattivo dimensionamento della memoria rispetto ai dati utilizzati dal programma che opera sul sistema.**



# MEMORIA VIRTUALE

**Il meccanismo di gestione della gerarchia formata da:**

- **Memoria principale**
- **Memoria secondaria**

**Prende il nome di *MEMORIA VIRTUALE*.**

- **Ogni attività o processo eseguito dal processore vede un proprio spazio di indirizzi virtuale o logico.**
- **La memoria è indirizzata tramite indirizzi fisici.**
- **Lo spazio degli indirizzi logici è maggiore dello spazio degli indirizzi fisici.**



## MEMORIA VIRTUALE

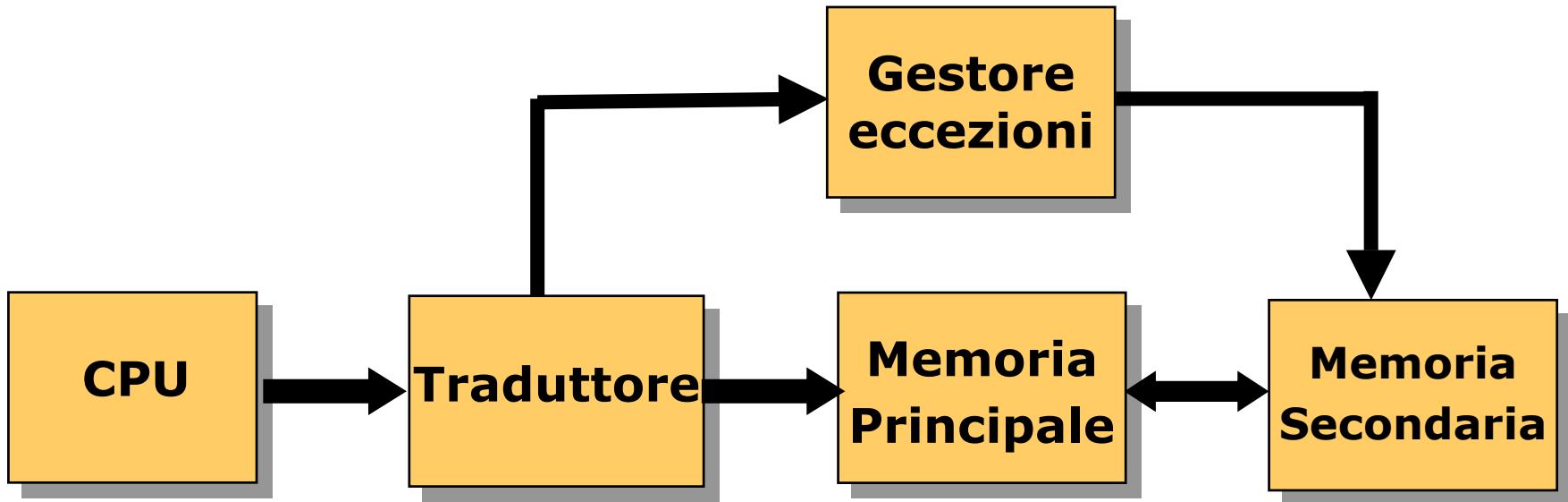
- **Esiste una corrispondenza (se necessaria) tra indirizzo logico e indirizzo fisico.**
- **La corrispondenza prende il nome di **allocazione**.**
- **L'allocazione è una funzione  $f$  che:**

$$\text{indir. fisico} = f(\text{indir. logico})$$

- **Libera il programmatore dalla necessità di occuparsi direttamente dello spazio in memoria fisica.**
- **Facilita la gestione comune della memoria tra più utenti.**

# MEMORIA VIRTUALE

- **Rende il programma indipendente dalle dimensioni di memoria della singola macchina.**
- **Sfrutta efficacemente la gerarchia.**





# ALLOCAZIONE

**Chi effettua la traduzione da ind. logico a ind. fisico ?**

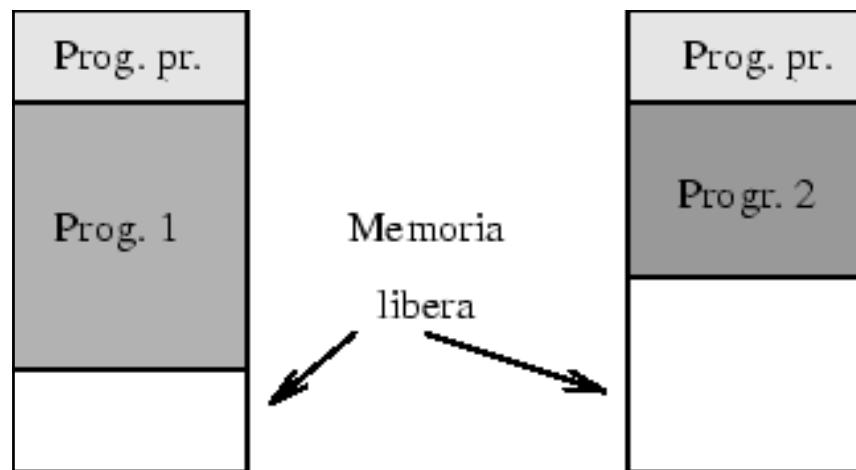
- **Il programmatore.**
- **Il compilatore in fase di compilazione.**
- **Il loader in fase di caricamento.**
- **Il sistema operativo in esecuzione.**

**In un sistema monoutente o monoapplicazione:**

- **Il programmatore (o il loader) specifica in modo diretto gli indirizzi fisici.**
- **Se la memoria necessaria all'applicazione supera la memoria fisica si ricorre all'OVERLAY.**

# OVERLAY

**Gestione diretta del programma dello spostamento  
dei dati dalla memoria secondaria alla memoria  
principale.**





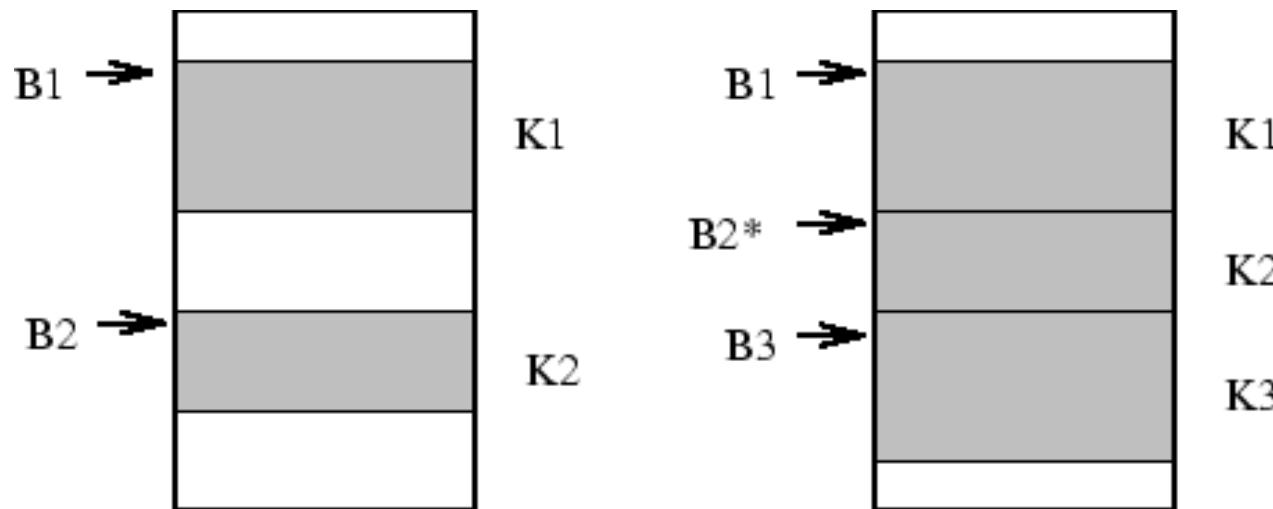
# ALLOCAZIONE

**In un sistema multiutente o multiapplicazione:**

- **Il programmatore o il compilatore specificano gli indirizzi relativi all'interno di singoli blocchi di programma.**
- **Il sistema operativo (S.O.), definisce, in fase di esecuzione, le locazioni fisiche nelle quali il singolo blocco viene posto.**

## ALLOCAZIONE DINAMICA

- **In ambiente di multiprogrammazione la collocazione fisica dello stesso codice varia.**
- **I blocchi sono rilocati in memoria alterando l'indirizzo di base.**





# ALLOCAZIONE DINAMICA

- **Il S.O. ha il controllo sulla allocazione.**
- **Una tabella in memoria *Memory address table* o *Memory map* o un insieme di registri di base o di rilocazione contengono le informazioni necessarie.**



## INDIRIZZO VIRTUALE

- **Indirizzo di blocco virtuale - da cui si estraе l'indirizzo fisico di blocco.**
- **Spiazzamento (offset) - usato direttamente come offset all'interno del blocco fisico.**

## INDIRIZZO Fisico ( o virtuale)

**Indirizzo di blocco**

E	2	C	0	0
---	---	---	---	---

 +

**spiazzamento**

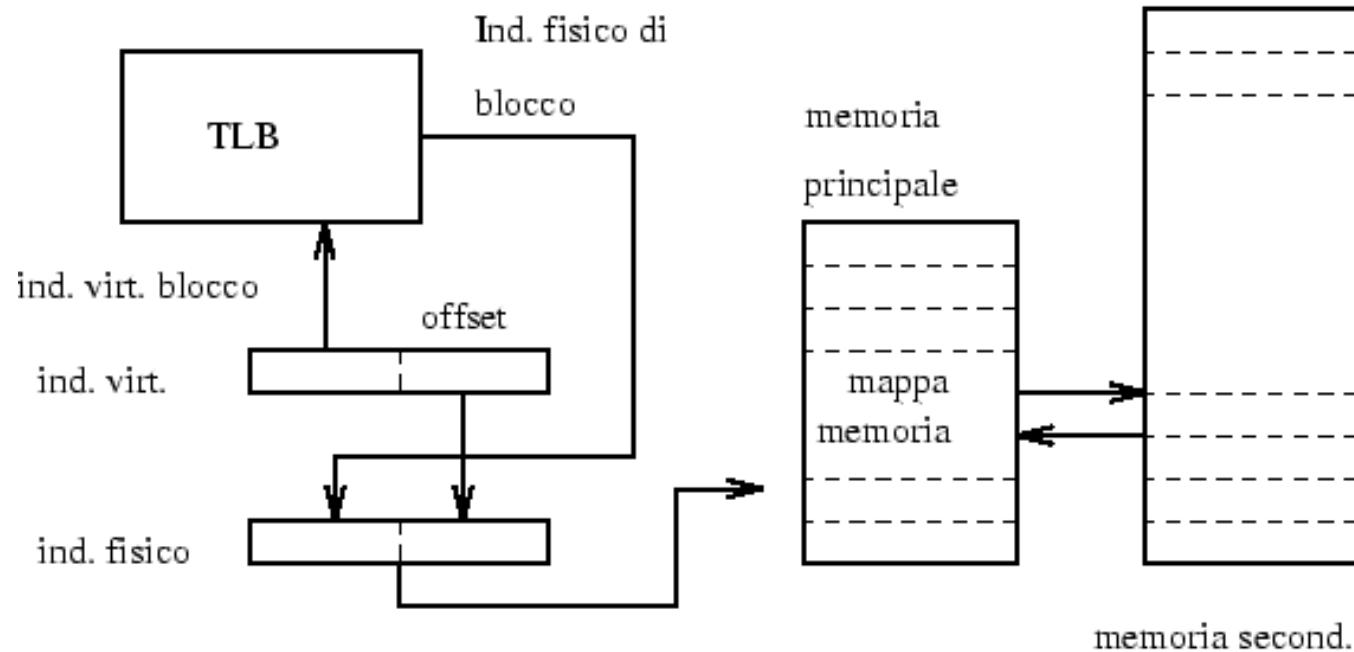
1	8	3
---	---	---

 =

**indirizzo fisico (o  
virtuale)**

E	2	D	8	3
---	---	---	---	---

# MEMORIA VIRTUALE



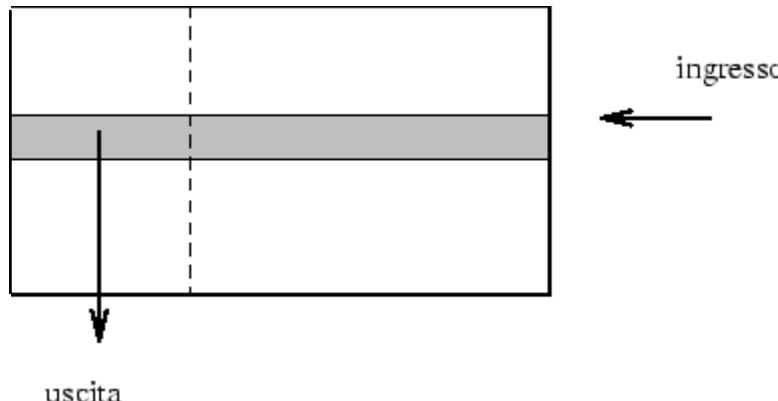
**Il TLB è una memoria temporanea che consente di effettuare velocemente la localizzazione nella memoria fisica dei dati individuati dall'indirizzo virtuale.**



## STRUTTURA

- **Parte della mappa di memoria è riportata nel TLB o Translation Lookaside Buffer.**
- **Il TLB viene gestito dinamicamente.**
- **La traduzione degli indirizzi nel TLB può avvenire con tecniche associative.**

# MEMORIA ASSOCIAUTIVA



- **Memoria tradizionale:** (in lettura) restituisce un dato conoscendone la posizione (indirizzo).
- **Memoria associativa:** restituisce una informazione fornendo un dato.

**La memoria associativa consente di verificare la presenza di un dato al suo interno e di inviare all'esterno o la posizione del dato o una parte del dato stesso.**

**Il dato presente nella memoria è formato dall'indirizzo virtuale cui è associato l'indirizzo fisico. Quando la memoria associativa individua al suo interno il dato (indirizzo virtuale) in ingresso, invia in uscita l'indirizzo fisico associato.**



## SEGMENTI

- **I programmi ad alto livello sono organizzati a moduli o blocchi.**
- **I moduli sono trasformati dal compilatore in segmenti.**
- **SEGMENTO: insieme di bytes (dati o programmi) contigui e posti in relazione logica.**
- **Il segmento è il blocco elementare che viene trasferito fra le gerarchie.**
- **I segmenti hanno dimensioni diverse.**
- **La CPU ha accesso a una tabella di descrittori di segmenti.**
- **Se il programma fa riferimento ad un segmento che non è presente in memoria viene generata una procedura di caricamento.**



# SEGMENTI

## VANTAGGI

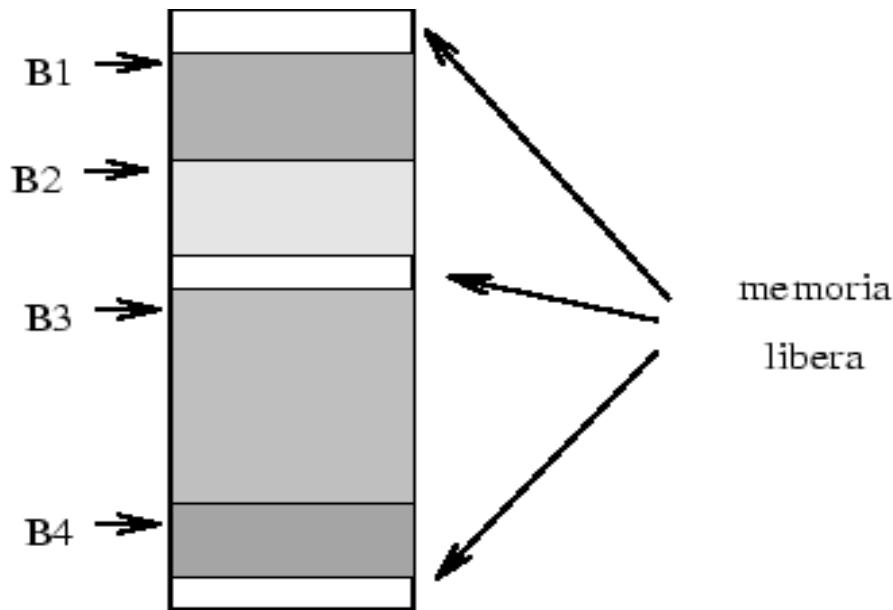
- **La struttura del segmento corrisponde alla naturale suddivisione dei dati e dei programmi.**
- **E' agevole gestire meccanismi di separazione e di protezione per i dati dei singoli utenti.**

## SVANTAGGI

**I segmenti hanno dimensioni fisiche diverse, quindi:**

- **allocazione complessa**
- **frammentazione (esterna)**
- **necessità di ricompattazione della memoria.**

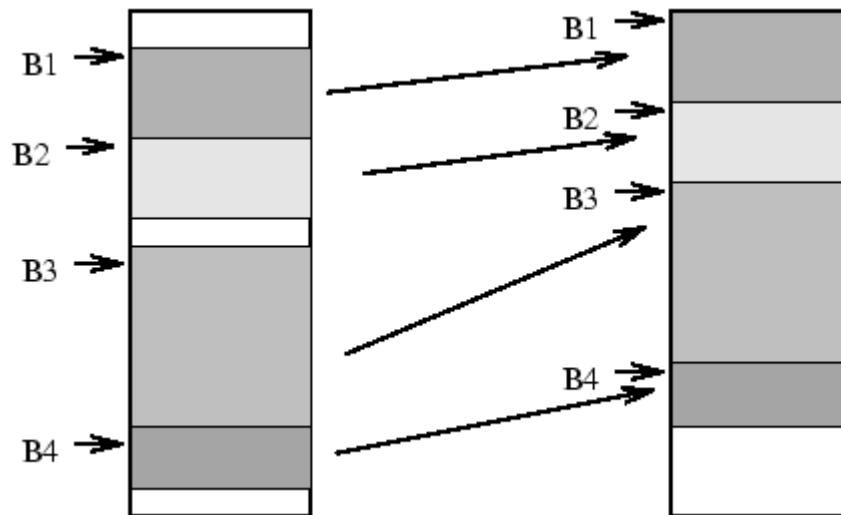
## FRAMMENTAZIONE ESTERNA



**La frammentazione esterna riduce l'utilizzazione della memoria quando risulta necessario disporre i dati dei segmenti in celle di memoria contigue. Lo stesso problema si verificava sui sistemi di memoria secondaria (dischi) quando i ogni file doveva essere memorizzato in segmenti adiacenti.**

# RICOMPATTAZIONE

- **La memoria non può essere sfruttata completamente.**
- **Tra le aree occupate rimangono zone di piccola dimensione nelle quali non possono essere allocati segmenti.**
- **Per migliorare l'utilizzazione della memoria: ricompattazione.**





# ALLOCAZIONE

**Il processo di allocazione gestisce la disponibilità della memoria fisica e cerca di ottimizzare:**

- **tasso di successo ( $H_A$ ),**
- **tempo di accesso ( $T_A$ ),**
- **utilizzo dello spazio (u).**

## ***MANCATO UTILIZZO***

- **Regioni vuote. Frammenti di memoria non utilizzati.**
- **Regioni occupate dal sistema di gestione.**
- **Regioni occupate da informazioni che non vengono utilizzate.**



## ALLOCAZIONE DEI SEGMENTI

- **Senza PREEMPTION : se non vi è spazio non si alloca.**
- **Con PREEMPTION : se non vi è spazio si ricoprono zone di memoria già occupate.**

**In questo ultimo caso occorre scegliere quale zona di memoria utilizzare: *Politica di sostituzione***



# ALLOCAZIONE SENZA PREEMPTION

**Per allocare un blocco di dimensioni L:**

- **FIRST-FIT** - Si scandisce la memoria e lo si pone nel primo posto libero che lo contiene.
- **BEST-FIT** - Si esaminano tutti gli spazi liberi (lunghezza  $n_i$ ) e si sceglie l'area libera per la quale  $n_i - L$  è minimo.

**L'allocazione non forzata presenta notevoli inconvenienti quando il sistema di elaborazione deve operare in un ambiente nel quale più utilizzatori condividono la stessa risorsa di calcolo.**

**In tal caso infatti occorre che le attività che utilizzano la memoria giungano a termine prima di poter iniziare nuove attività.**



## PAGINAZIONE

**Per trasferire dati tra memoria principale e secondaria  
non si usano blocchi logici (segmenti) ma blocchi di  
dimensione fissa o PAGINE.**

**La Tabella Pagine contiene informazioni relative a:**

- **Indirizzo fisico.**
- **Presenza in memoria.**
- **Se modificato da quando in memoria.**
- **Diritti di accesso.**

**L'uso delle pagine al posto dei segmenti come blocco  
elementare per il trasferimento dei dati nella  
gerarchia di memoria formata dalle memorie primaria  
e secondaria consente di semplificare le procedure di  
trasferimento.**



# VANTAGGI E SVANTAGGI

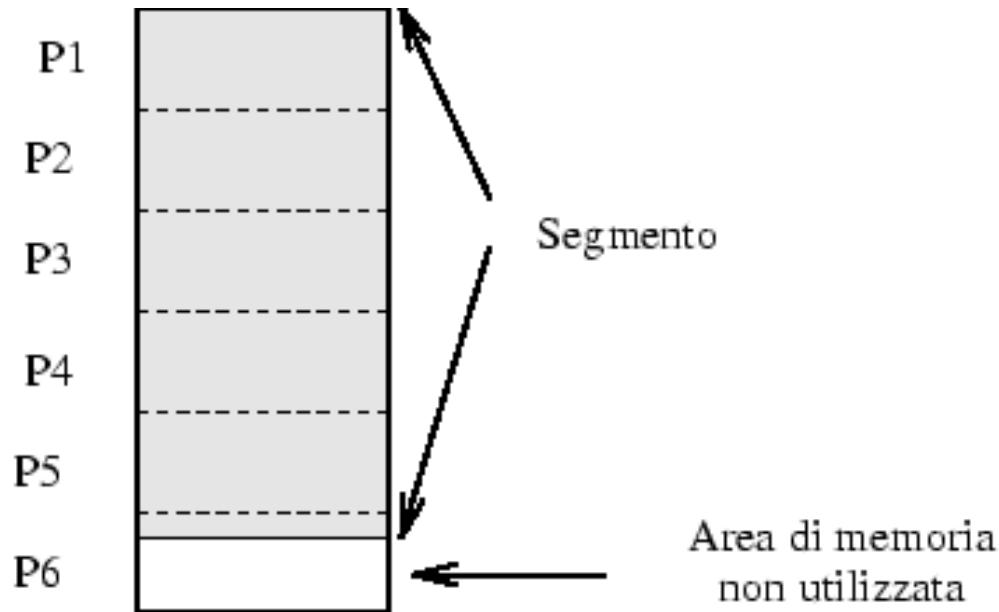
## **VANTAGGI**

- **Allocazione più semplice (dimensione fissa del blocco).**
- **Assenza di frammentazione esterna.**

## **SVANTAGGI**

- **Maggiore difficoltà di gestione delle proprietà.**
- **Frammentazione interna - La dimensione dei dati da trasferire non è un multiplo intero delle pagine.**

# FRAMMENTAZIONE INTERNA





## SEGMENTI PAGINATI

**Per cercare di ottenere i vantaggi di entrambe le soluzioni:**

- **Ogni segmento è suddiviso in pagine.**
- **I segmenti mantengono le loro proprietà.**
- **Non è più necessario allocare i segmenti in modo contiguo in memoria.**
- **Tabella segmento + Tabella pagine per ogni segmento.**



## SEGMENTI PAGINATI

**Problema: dimensione ottima della pagina per utilizzare al meglio la memoria.**

- $S_p$  dimensione pagina.
- $S_s$  dimensione segmento.

**Overhead per segmento:**

**$S = \text{mezza pag.} + 1 \text{ parola per pag.} =$**

$$S = \frac{S_p}{2} + \frac{S_s}{S_p}$$



# OTTIMIZZAZIONE

**Minimo di  $S$  quando:**

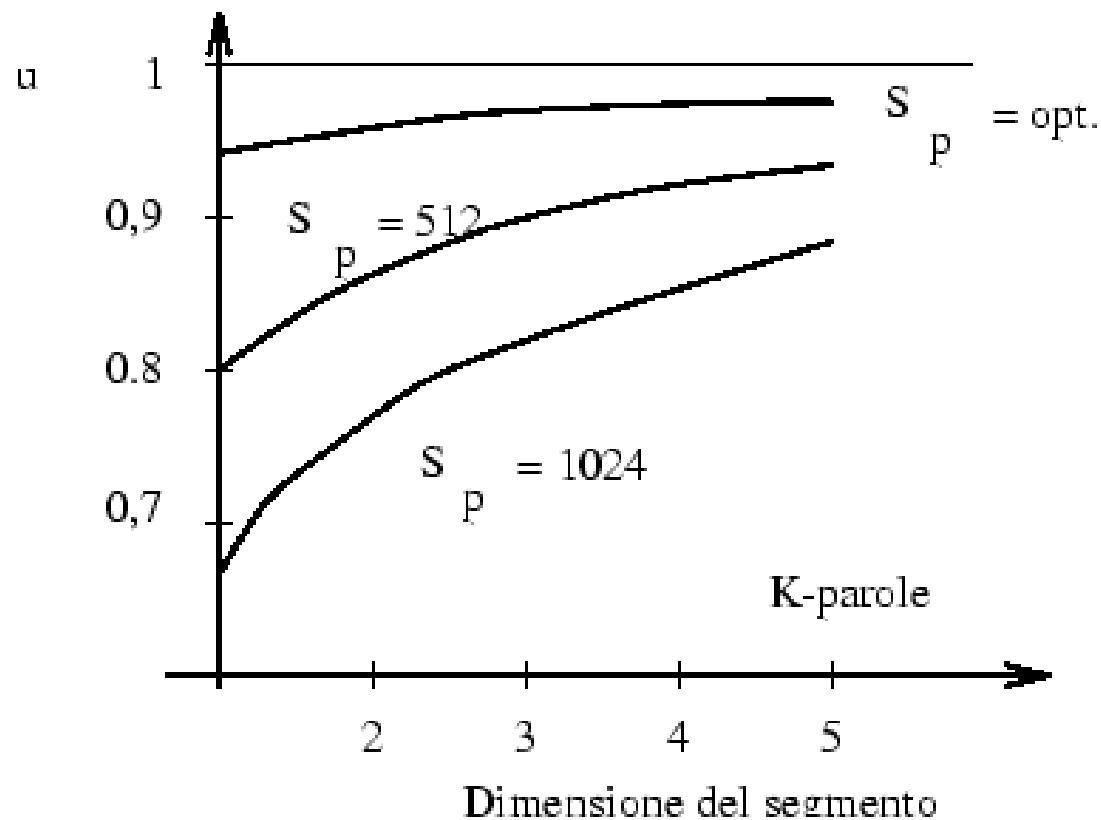
$$\frac{dS}{dS_p} = \frac{1}{2} - \frac{S_s}{S_p^2} = 0$$

$$S_p^{ott.} = \sqrt{2S_s}$$

**In questo caso l'utilizzazione diventa:**  $u = \frac{S_s}{S_s + S}$

$$u^{ott} = \frac{1}{1 + \sqrt{2/S_s}}$$

# OTTIMIZZAZIONE





# POLITICHE DI SOSTITUZIONE

**OBIETTIVO: minimizzare il numero di insuccessi  
(memory fault).**

**IPOTESI: il tasso di successo (H) assume valore massimo se si rende massimo l'intervallo tra due fault successivi .**

**Strategia di sostituzione ottima:**

**All'istante  $t_i$  determinare, per ogni pagina  $K_j$ , quando (istante  $t_j$ ) verrà richiesta.  
Scegliere  $j$  per cui  $t_j - t_i$  risulti massimo.**



# **POLITICA DI SOSTITUZIONE OTTIMA**

**Richiede più passi:**

- **eseguire il programma per ricavare le richieste di accesso.**
- **calcolare le sostituzioni ottime.**
- **eseguire il programma con la sequenza ottima.**

**La strategia di sostituzione ottima non è utilizzabile perché richiede l'esecuzione preventiva del programma per identificare la strategia ottima. Questa soluzione è di difficile utilizzazione anche nel caso più favorevole in cui una sola attività operi sul sistema.**

**Nella maggior parte dei casi le attività che vengono eseguite su di un sistema sono più di una e quindi è impossibile una ottimizzazione che tenga in conto tutte le attività.**



## POLITICHE DI SOSTITUZIONE

- **FIFO - First-In First-Out**
  - **si associa al blocco un numero di sequenza.**
  - **si sostituisce il blocco introdotto meno recentemente.**
- **LRU - Least-Recently Used**
  - **si sostituisce il blocco che da più tempo non viene utilizzato.**



# POLITICA OTTIMA



# POLITICA FIFO



# POLITICA LRU

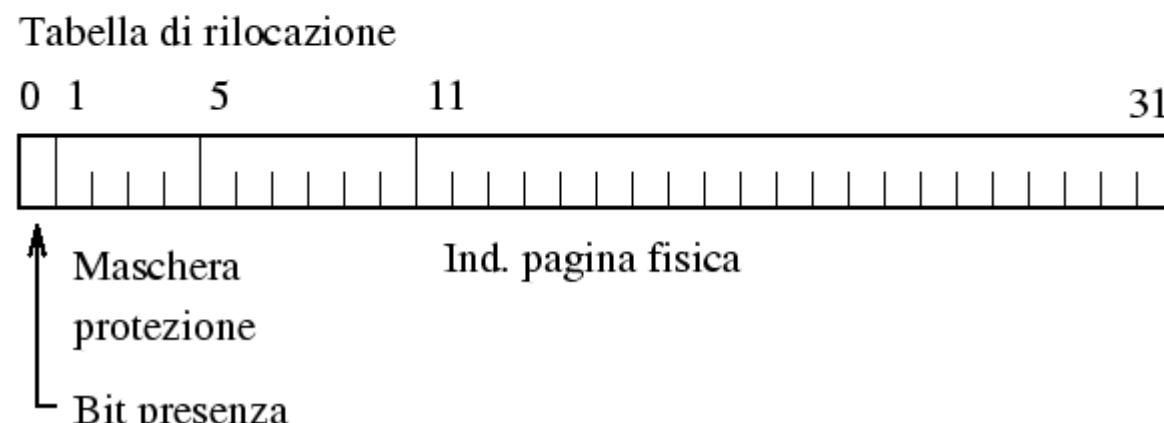
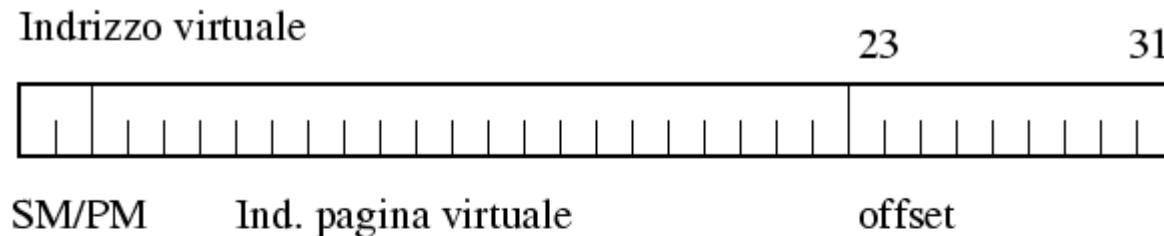


## ESEMPIO: VAX11/780

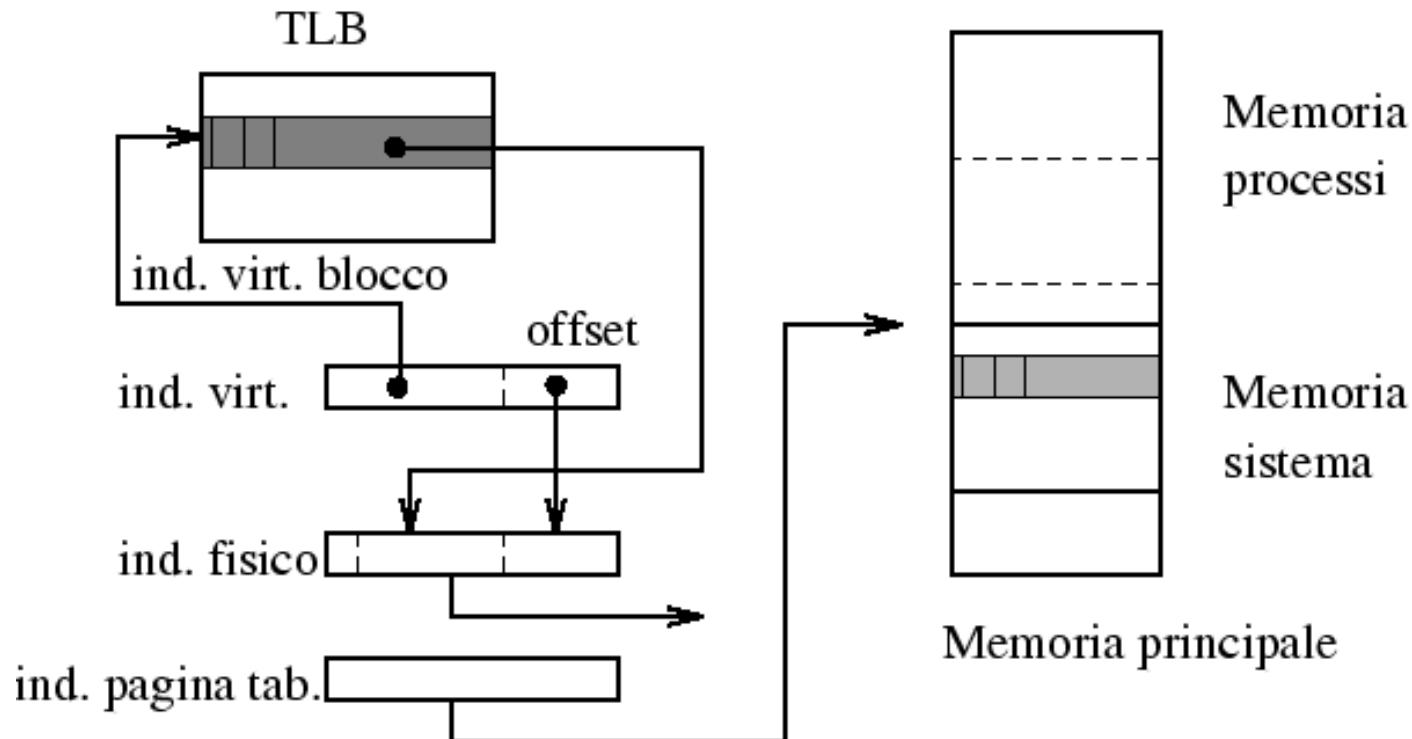
- **L'architettura VAX11 venne progettata per operare in ambiente di multiprogrammazione.**
- **Numerosi processi operano contendendosi le risorse: CPU, I/O, Memoria.**
- **Spazio di memoria virtuale  $2^{32}$  bytes suddivisi in pagine di 512 bytes.**
  
- **Ogni processo dispone del diritto ad operare su una quantità di memoria fisica definita (working set).**
- **Quando un processo è attivo risiede completamente in memoria virtuale.**

## ESEMPIO: VAX11/780

- **Nella memoria fisica risiedono solo le pagine in uso.**
- **La memoria è suddivisa in Memoria di sistema (SM) e Memoria dei Processi (PM).**

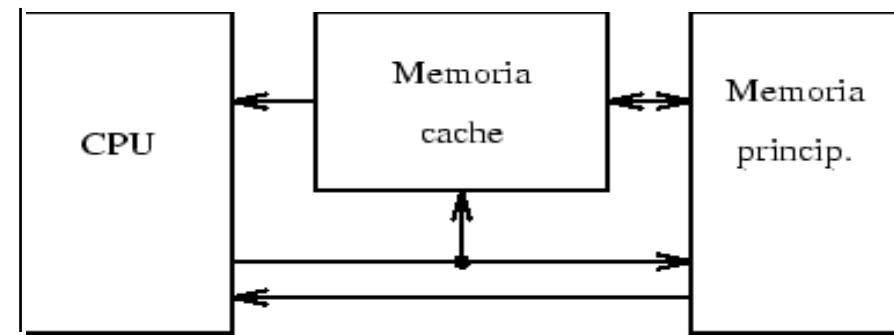


## ESEMPIO: VAX11/780



## MEMORIA CACHE

- **La cache è una memoria veloce e di piccole dimensioni posta fra la CPU e la memoria principale.**
- **La cache e la memoria principale formano una gerarchia di memoria.**



**I tre livelli di memoria:  
Cache - Mem. principale - Mem. secondaria  
sono gestiti come due gerarchie a due livelli:**

- **Mem. principale - Mem. secondaria.**
- **Cache - Memoria principale.**



## MEMORIA CACHE

**Le prestazioni della memoria cache dipendono anche dalla sua posizione rispetto alla CPU:**

- Cache su scheda.
- Cache su chip.

**La presenza di una memoria cache sullo stesso chip del processore rappresenta la soluzione che garantisce la maggiore efficienza.**

**Nei sistemi più recenti sono presenti entrambe le soluzioni.**



# CONFRONTO

**Gerarchia**

**Rapporto tempi  
di accesso**

**Gestione**

**Dim. pagina**

**Accesso CPU**

**2° livello**

**Cache - M.Pr.**

**5/1**

**Hardware**

**4 - 128 bytes**

**Diretto**

**M.Pr. - M.Se.**

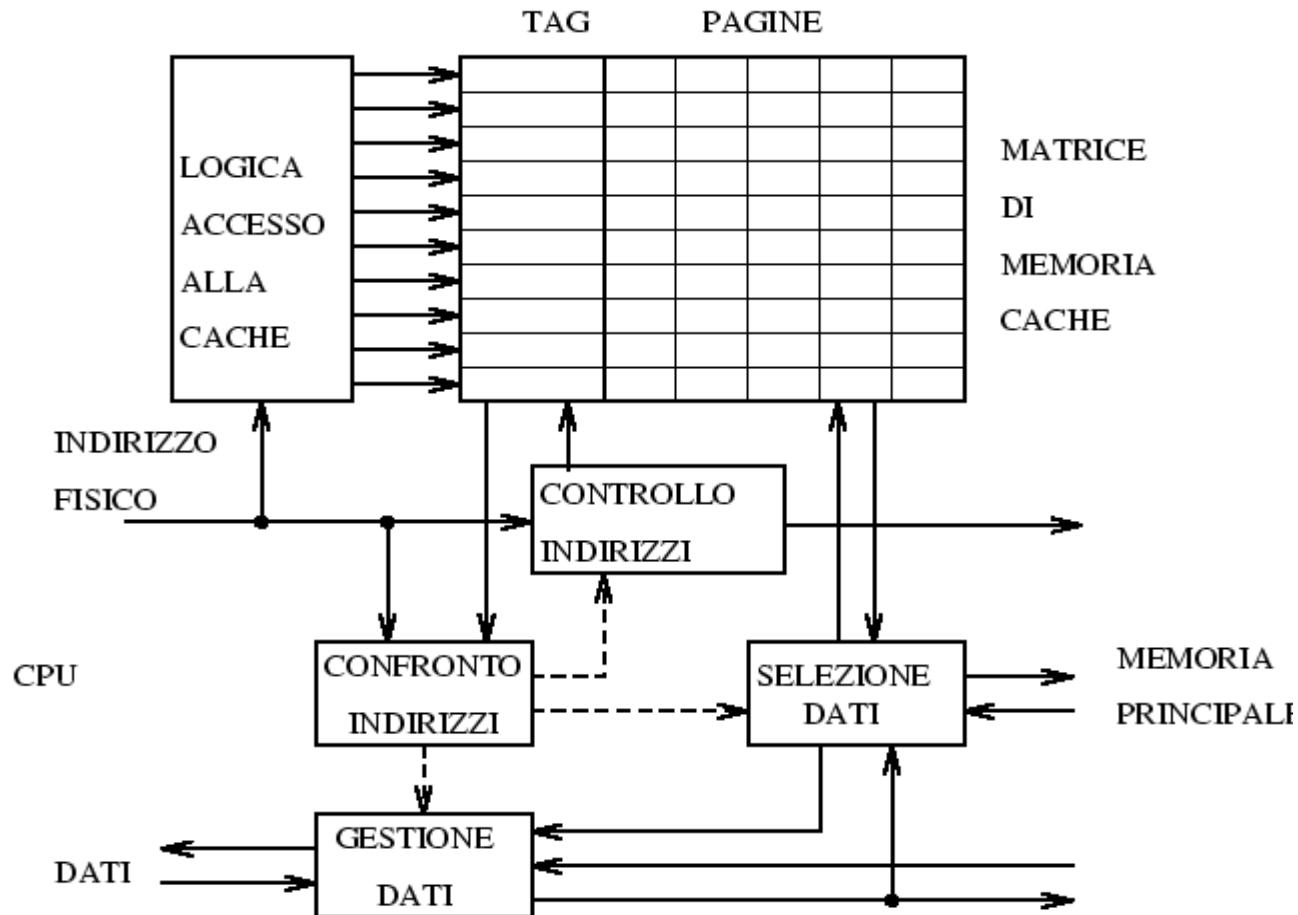
**1000/1**

**Software**

**64 -4096 bytes**

**Sempre tramite  
il primo**

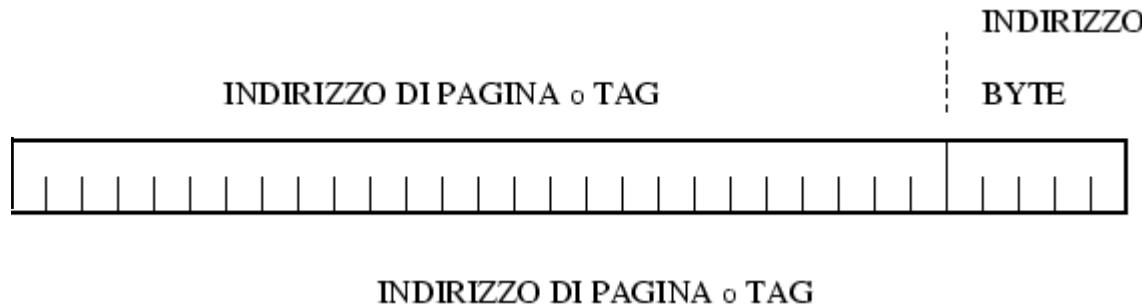
# MEMORIA CACHE





## DATI NELLA CACHE

**I dati vengono memorizzati in pagine di cache (o linee) caratterizzate da una parte dell'indirizzo o tag.**





## CICLO OPERAZIONI

**Un indirizzo fisico viene inviato dalla CPU.**

**La cache confronta la parte rilevante dell'indirizzo.  
Se vi è  $\{hit\}$  viene completato il ciclo.**

**In caso di  $\{miss\}$ :**

- La cache inizia la lettura da memoria principale di dati che comprendono quello richiesto.**
- La politica di sostituzione ha gli stessi problemi di quella preemptive della memoria virtuale.**



## PROBLEMI SPECIFICI

- **Come sono organizzati nella memoria cache i dati provenienti dalla memoria principale ?**
- **Quando la CPU modifica un dato nella cache, come viene aggiornata la memoria principale ?**

### ***ORGANIZZAZIONE CACHE***

**Il dato richiesto deve essere trovato rapidamente nella memoria cache.**

- **Tecniche associative.**
- **Accesso diretto.**
- **Tecniche miste.**



## TECNICHE ASSOCIATIVE

- **La cache è organizzata come una memoria associativa.**
- **Il tag viene utilizzato come chiave di lettura.**

**Come si è già visto in precedenza la tecnica associativa è molto costosa quando le dimensioni della memoria crescono.**



## ACCESSO DIRETTO

**La memoria cache  $M_1$  è suddivisa in  $S_1 = 2^k$  pagine (o linee).**

**Ogni pagina immagazzina n parole consecutive.**

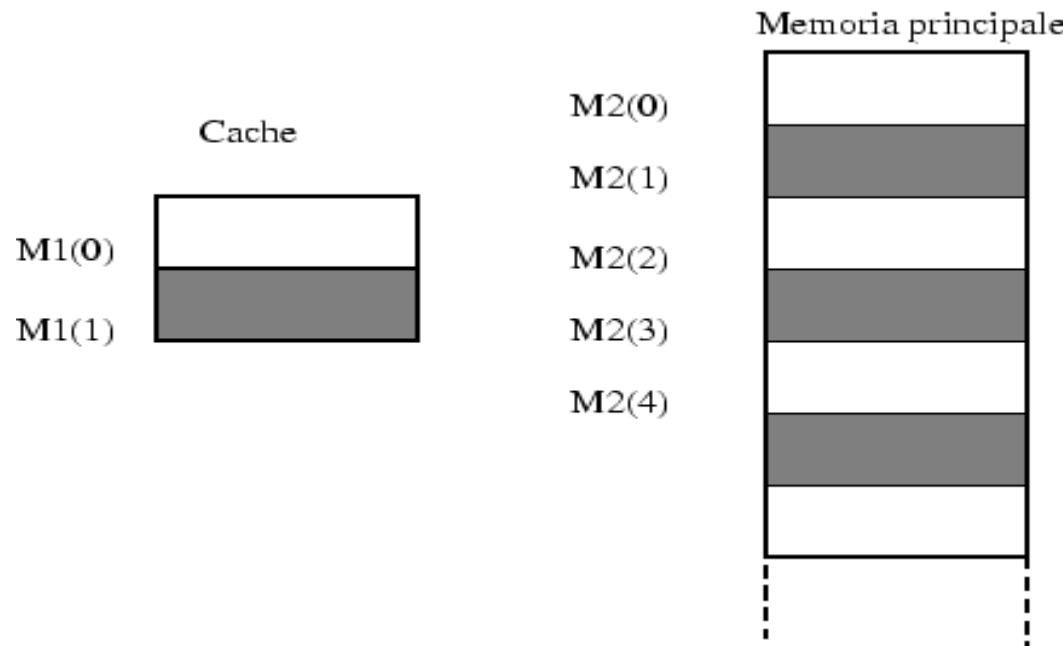
**La memoria principale  $M_2$  è suddivisa in  $S_2$  regioni di eguale dimensione.**

**$M_1(1), \dots M_1(i), \dots M_1(S_1)$ : pagine in cache.**

**$M_2(1), \dots M_2(j), \dots M_2(S_2)$ : pagine in mem. principale.**

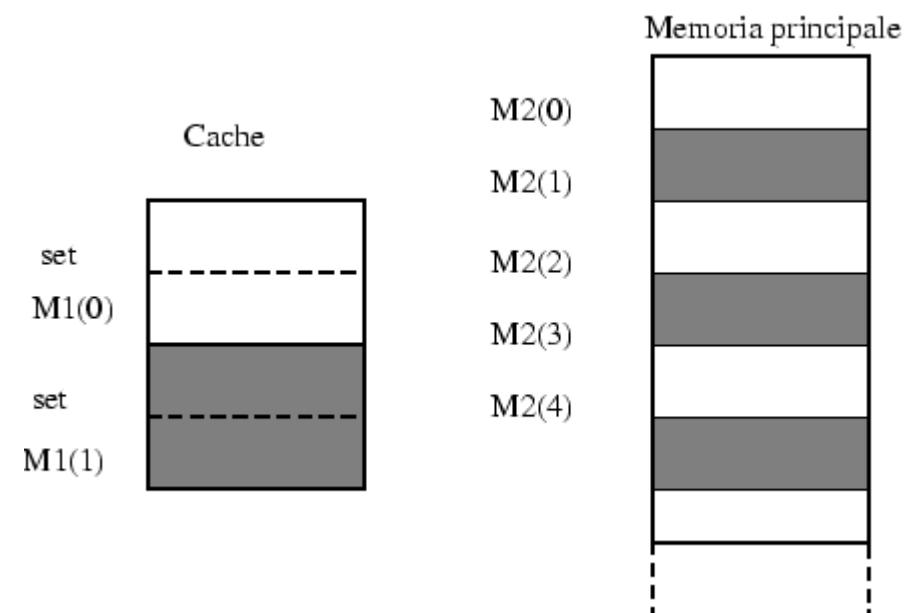
**$M_2(j)$  va in  $M_1(i)$  con  $i = j \text{ (mod. } S_1)$**

# ACCESSO DIRETTO



# TECNICA MISTA O SET ASSOCIATIVE

- **Le linee nella cache sono suddivise in gruppi.**
- **Ogni gruppo può ospitare linee della memoria principale individuate secondo la tecnica di accesso diretto.**
- **All'interno di ogni gruppo la linea viene individuata in modo associativo.**
- **L'accesso diretto avviene rapidamente.**
- **Sono necessarie più memorie associative ma di dimensioni ridotte.**





# AGGIORNAMENTO MEMORIA PRINCIPALE

**Quando la CPU scrive sulla memoria cache si genera una discrepanza fra il valore del dato nella memoria principale e nella cache.**

- ***Write-back o Copy-back.***
  - **Quando viene modificato un dato nella cache la linea corrispondente viene marcata.**
  - **Quando una linea è scaricata viene ricopiata nella memoria principale solo se marcata**
- ***Write-through o Copy-through.***
  - **In un programma il rapporto fra le operazioni di lettura e di scrittura è dell'ordine di 10.**
  - **Quando viene modificato un dato nella cache, lo stesso viene modificato anche nella memoria principale.**



# AGGIORNAMENTO MEMORIA PRINCIPALE

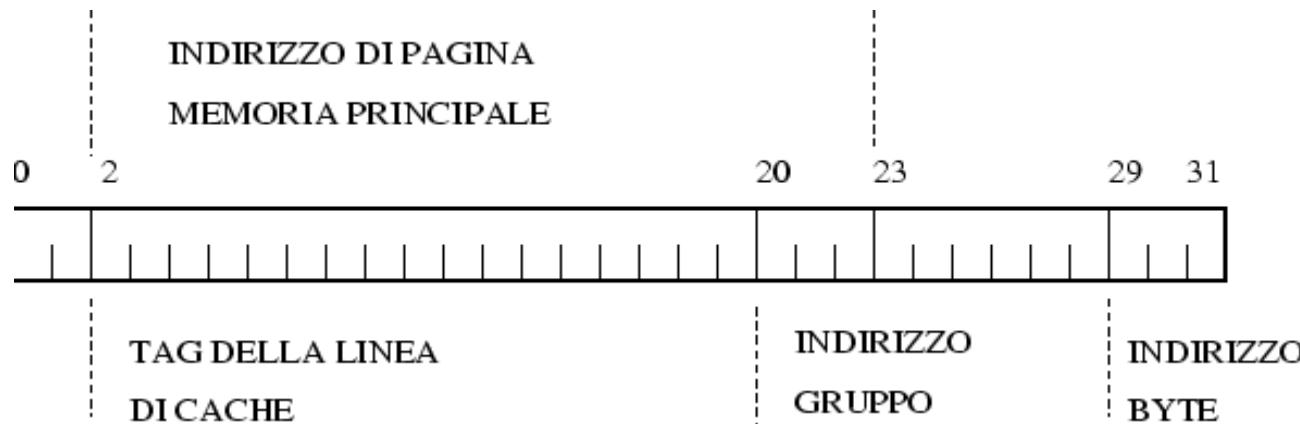
**La connessione tra la memoria principale e la memoria cache è fondamentale per le prestazioni del sistema.**

**Le tecniche che consentono di migliorare le prestazioni della connessione sono:**

- **L'uso di parallelismo nel collegamento (più byte trasferiti in parallelo).**
- **L'uso di buffer per la memorizzazione temporanea dei dati. Quando una linea è scaricata dalla cache viene dapprima memorizzata in una memoria temporanea con tempo di accesso ridotto e trasferito solo successivamente nella memoria principale.**

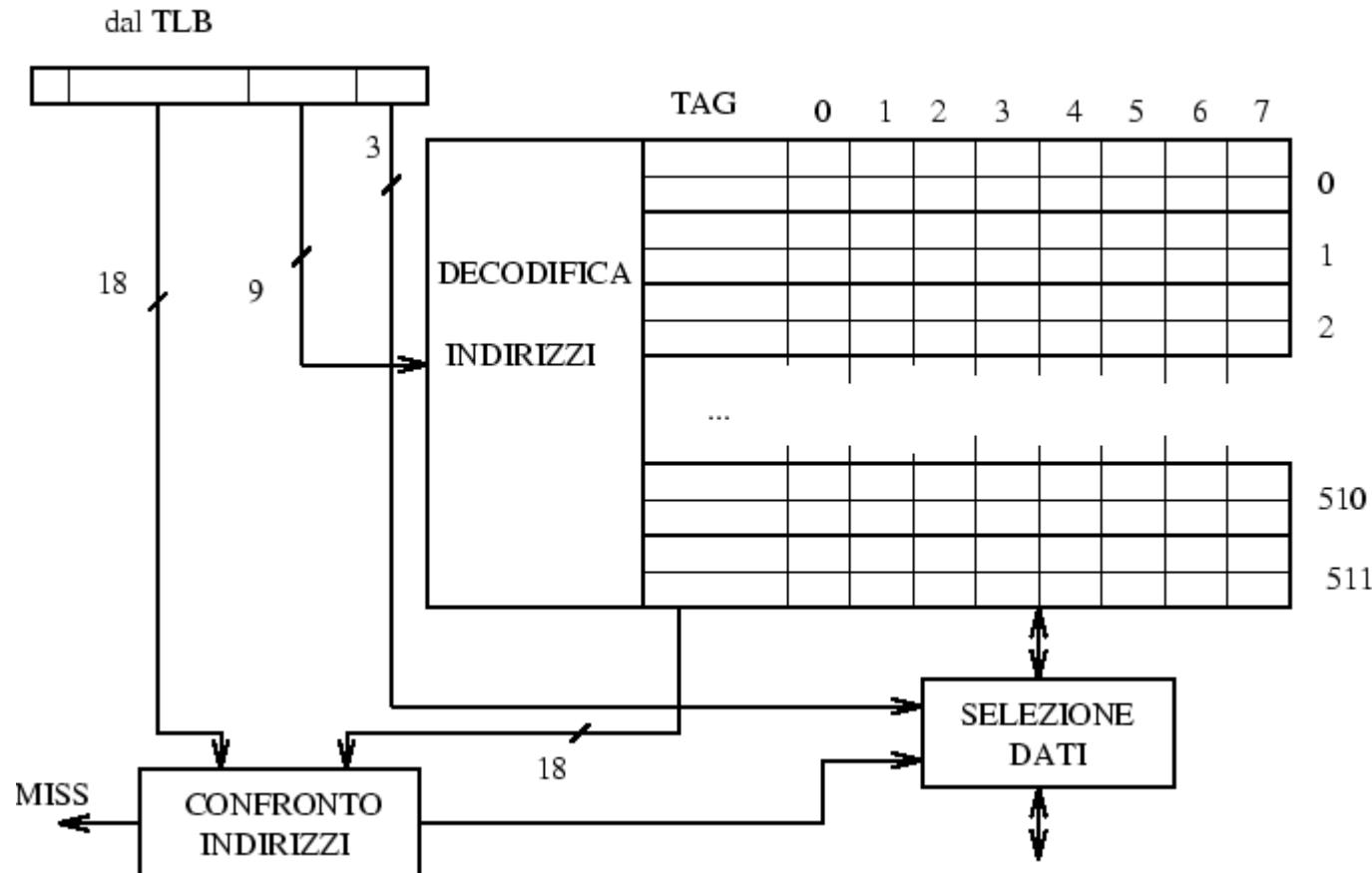
## CACHE VAX11/780

- **Memoria cache con capacità 8K bytes.**
- **Ogni linea contiene 8 bytes.**
- **Tecnica *set associative* con 512 gruppi di 2 linee.**



**Interpretazione dell'indirizzo fisico da parte della cache.**

# CACHE VAX11/780





## CACHE VAX11/780

**Come si vede nella figura precedente:**

- **i tre bit (A0-A2) meno significativi dell'indirizzo vengono utilizzati per individuare il byte all'interno della linea,**
- **i nove bit (A3-A11) immediatamente precedenti sono utilizzati per individuare il gruppo nel quale la linea può essere immagazzinata.**

**Per riconoscere la presenza del dato nella memoria cache si procede quindi alla ricerca associativa nel gruppo (composto da due elementi).**



## ALCUNI PROBLEMI

**La memoria cache può fare riferimento all'indirizzo virtuale invece che all'indirizzo fisico.**

- **La traduzione dell'indirizzo e la ricerca nella cache possono essere parzialmente sovrapposte.**
- **La gestione è molto più complessa.**

## ALCUNE ALTERNATIVE

- **Cache separate per i dati e le istruzioni.**
- **Livelli di cache multipli.**

**La utilizzazione di memorie cache separate e di livelli di cache diversi si va affermando nelle unità di elaborazione più moderne.**