Architettura degli Elaboratori I - B

Le Memorie Cache

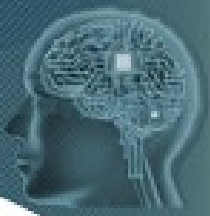
Cash e Memoria Virtuale

Daniel Riccio/Alberto Aloisio
Università di Napoli, Federico II

5 aprile 2017

Fonti: lucidi degli anni precedenti

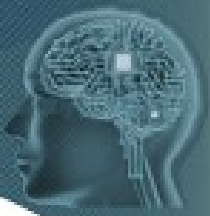
Frazionamento della memoria in moduli



E utile che i trasferimenti da/verso le unità più veloci possano essere effettuati alla stessa frequenza dell'unità più veloce.
Ciò non può avvenire se si accede nello stesso modo all'unità più lenta e a quella più veloce.
Tale risultato può invece essere raggiunto se si sfrutta il parallelismo nell'organizzazione dell'unità più lenta.
Un modo efficiente per introdurre il parallelismo nella gestione della memoria principale consiste nell'impiego di un'organizzazione ***interallacciata***.

Se la memoria principale di un calcolatore è strutturata come una collezione di moduli fisicamente separati, ognuno con il proprio registro degli indirizzi (Address Register, AR) e registro dei dati (Data Register, DR), le operazioni di accesso alla memoria possono procedere contemporaneamente in più di un modulo.
Così, si può incrementare la frequenza aggregata di trasmissione delle parole da o verso il sistema di memoria principale.

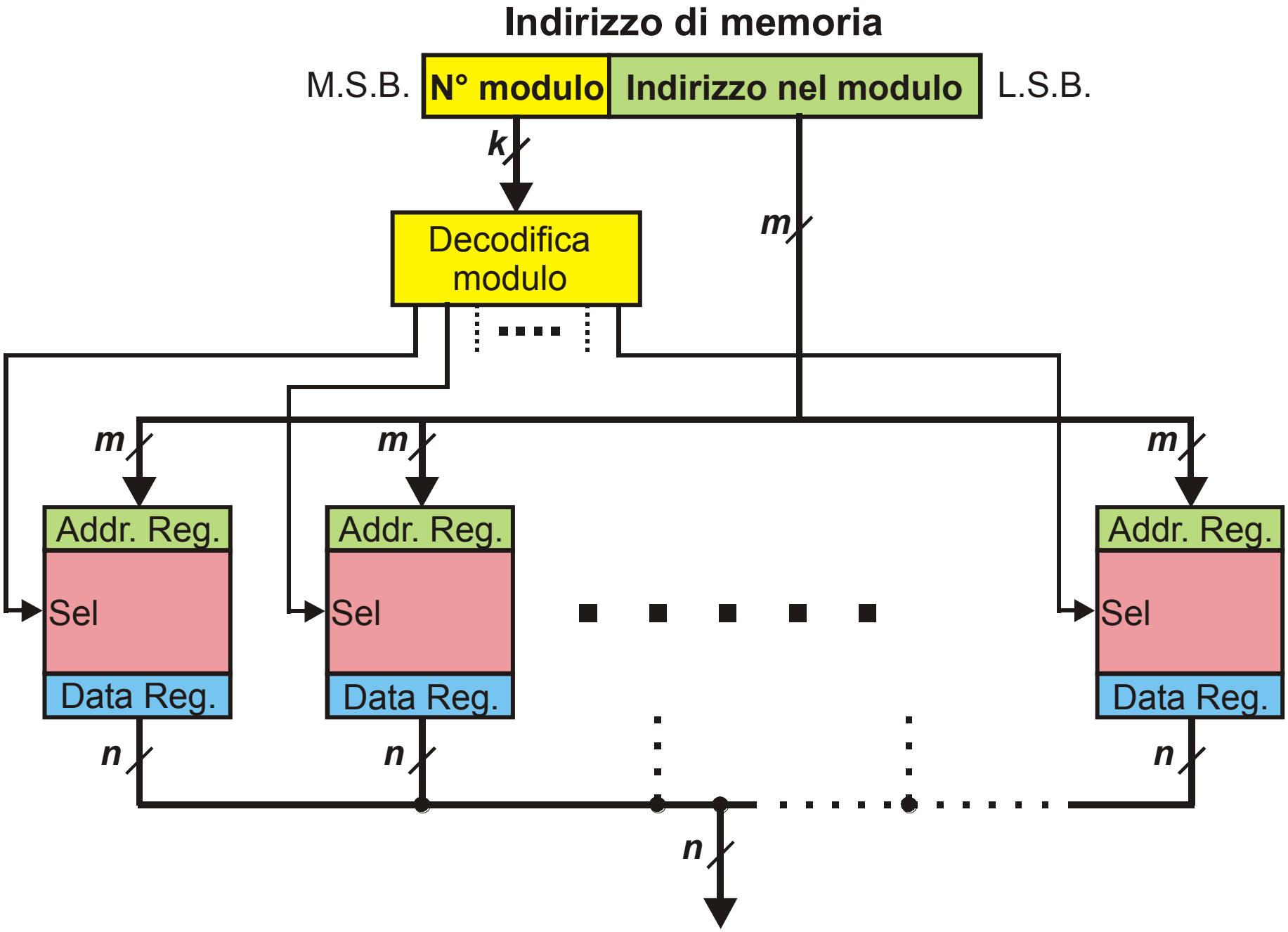
Distribuzione degli indirizzi nei moduli (1 / 4)



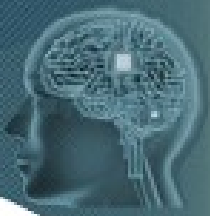
Il modo in cui gli indirizzi individuali sono distribuiti attraverso i moduli è importante per stabilire il numero medio di moduli che può essere tenuto occupato mentre le elaborazioni procedono.

Esistono due metodi di distribuzione degli indirizzi.

Nel primo, l'indirizzo di memoria generato dalla CPU viene decodificato nel seguente modo:
I ***k*** bit più significativi indicano uno degli ***n*** moduli, e gli ***m*** bit meno significativi indicano una particolare parola all'interno del modulo. Quando si accede a locazioni consecutive, come accade quando si copia un blocco di dati nella cache, viene coinvolto soltanto un modulo. Nello stesso tempo, tuttavia, dispositivi con possibilità di accesso diretto alla memoria (DMA) possono aver accesso alle informazioni contenute in altri moduli della memoria.

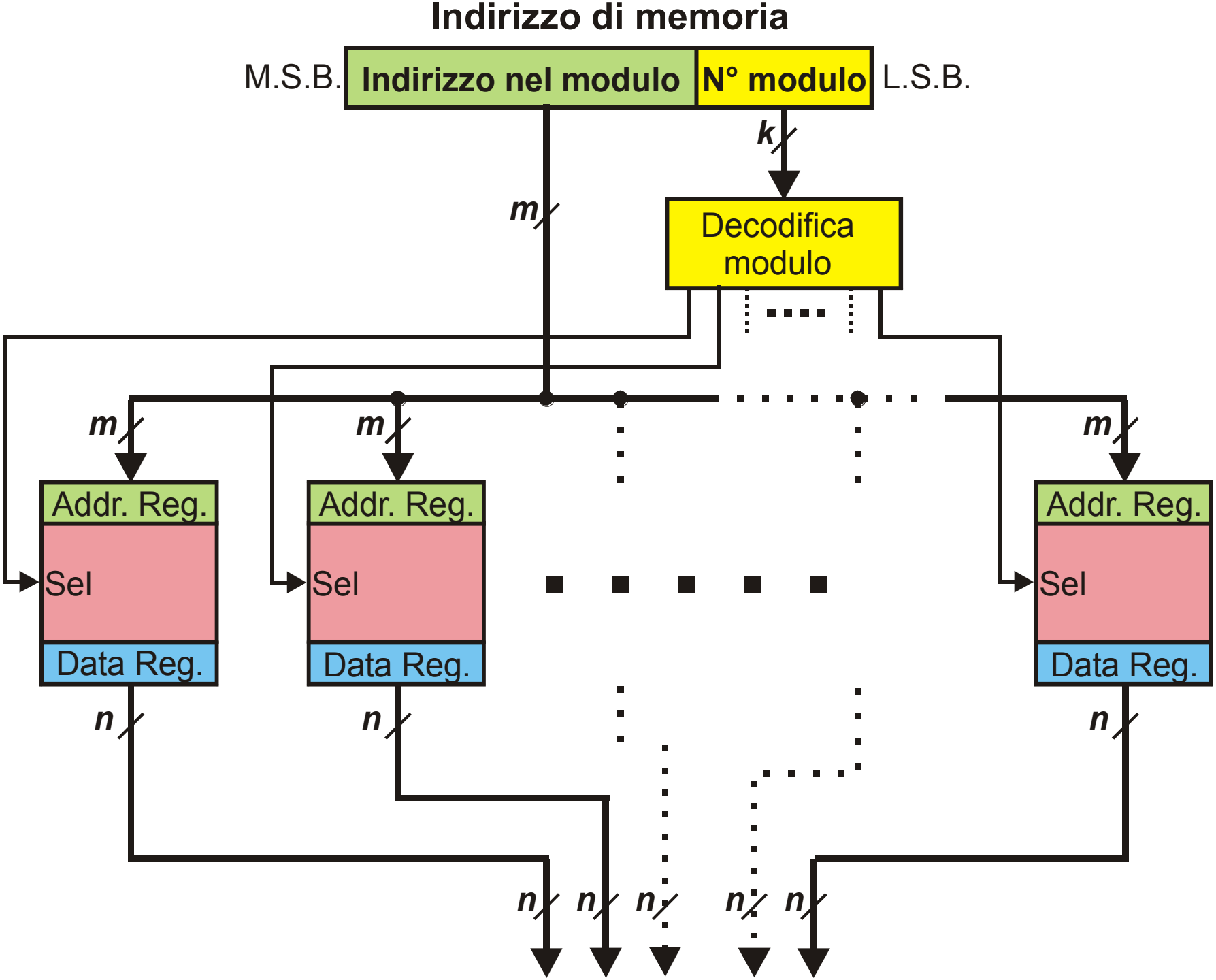

Indirizzo di memoria
M.S.B.
N° modulo
Indirizzo nel modulo
L.S.B.
k
m
Decodifica modulo
m
m
m
Addr. Reg.
Addr. Reg.
Addr. Reg.
Sel
Sel
Sel
Data Reg.
Data Reg.
Data Reg.
n
n
n
n

Distribuzione degli indirizzi nei moduli (3 / 4)

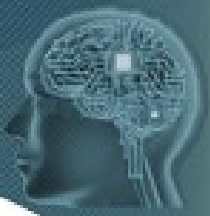


Il secondo e più efficiente modo di indirizzare i moduli è chiamato ***interallacciamento della memoria.***
I ***k*** bit meno significativi dell'indirizzo di memoria selezionano un modulo, mentre gli ***m*** bit più significativi indicano una locazione all'interno del modulo. In questo modo, indirizzi consecutivi sono posizionati in moduli successivi; così facendo, ogni componente del sistema che faccia richiesta di accesso a locazioni di memoria consecutive può tenere occupati vari moduli in un qualsiasi istante. Ciò consente di ottenere sia accessi a un blocco di dati più veloci, sia una più elevata media complessiva di utilizzo della memoria.

Per realizzare la struttura interallacciata, devono esserci *2k* moduli.
In caso contrario, ci sono degli stati vuoti associati a locazioni inesistenti nello spazio di indirizzamento della memoria.


Indirizzo di memoria
M.S.B.
Indirizzo nel modulo
N° modulo
L.S.B.
k
m
Decodifica modulo
m
m
m
Addr. Reg.
Addr. Reg.
Addr. Reg.
Sel
Sel
Sel
Data Reg.
Data Reg.
Data Reg.
n
n
n
n
n
n
n

Effetti della memoria interallacciata (1/3)



Analizziamo il tempo necessario per trasferire un blocco di dati dalla memoria principale alla cache nel caso in cui un'operazione di lettura fallisca. Supponiamo di avere una cache con **blocchi di 4 parole**.

Nel caso di un fallimento in fase di lettura, il blocco che contiene la parola desiderata deve essere trasferito nella memoria cache.

Si supponga che sia necessario un ciclo di clock per inviare un indirizzo alla memoria principale.

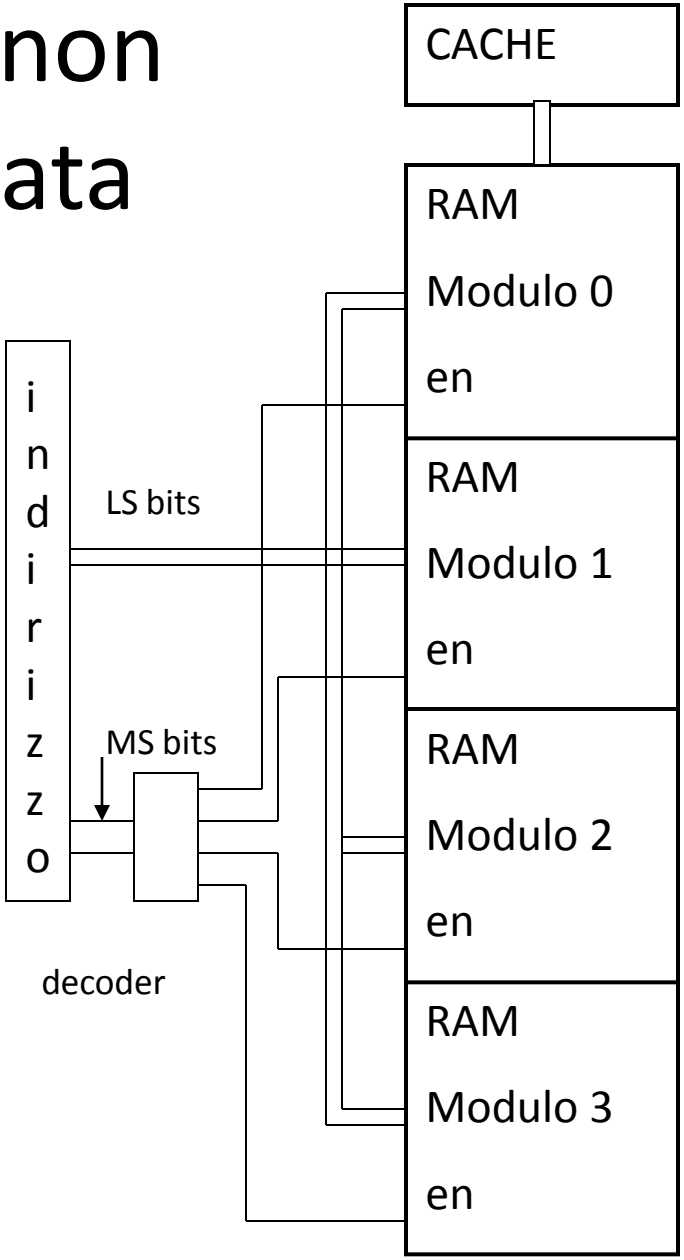
Se la memoria è composta da un unico modulo, il tempo necessario per caricare il blocco desiderato nella cache, detto penalità di fallimento è

4M+4C

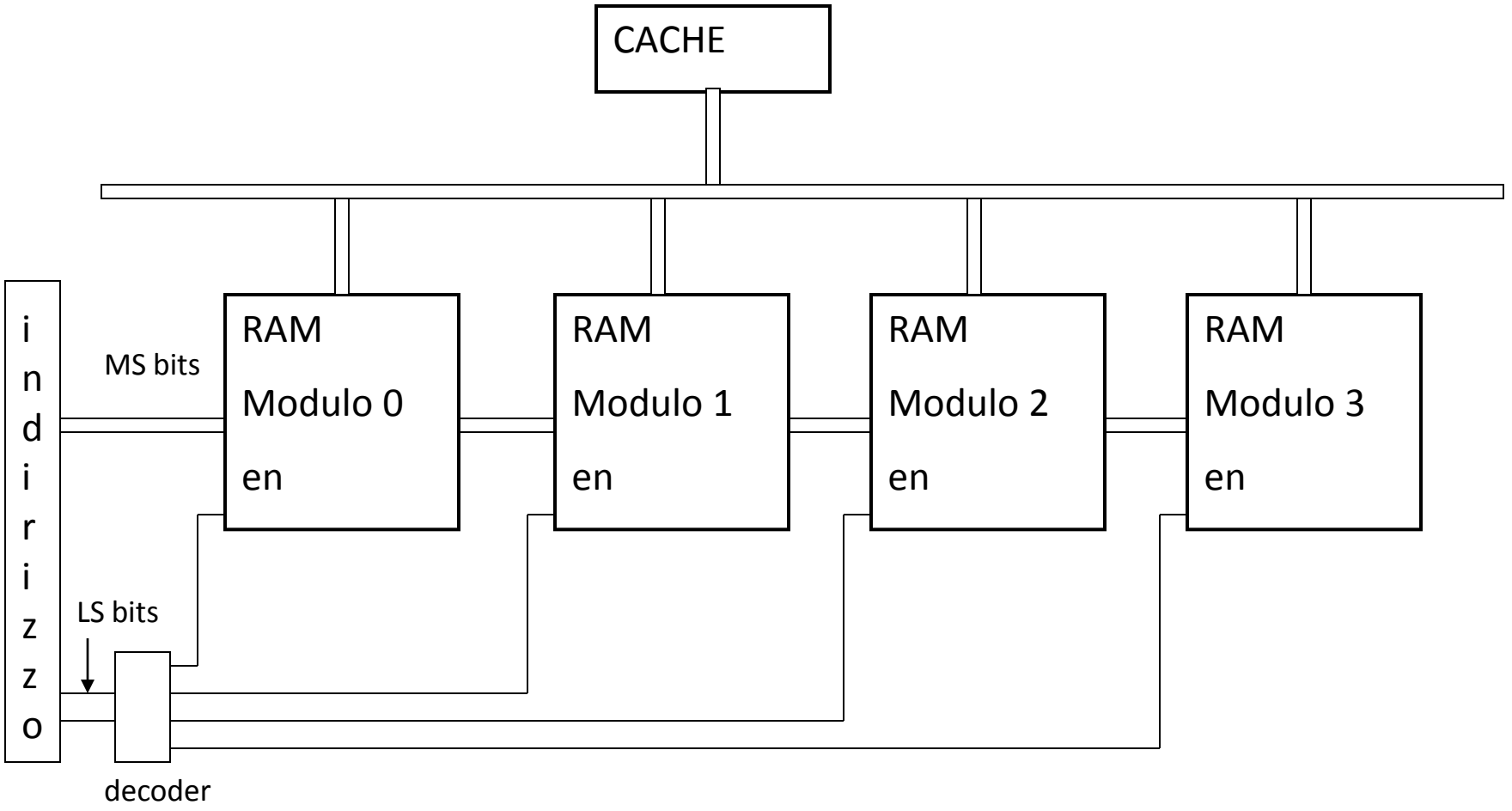
dove M e C sono i tempi di accesso alla memoria principale ed alla cache.

Nel caso di una organizzazione della memoria per moduli interallacciati, ad es. 4, si osserva che il tempo si riduce a **M+4C** in quanto i 4 moduli di memoria possono lavorare in parallelo.

Memoria non interlacciata

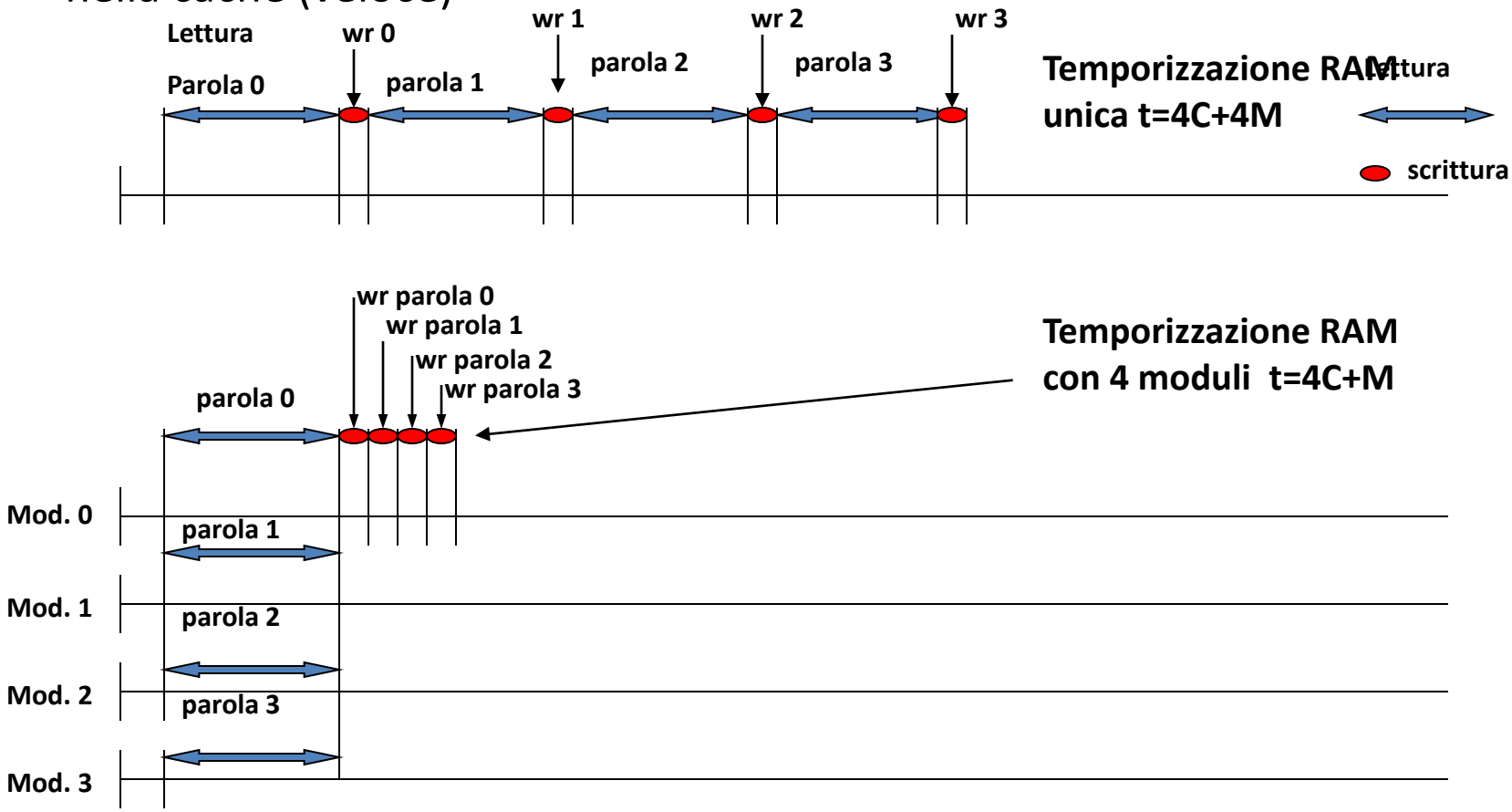


Memoria interlacciata: uno schema applicativo



Effetti della memoria *interallacciata* (3/3 grafico)

Il diagramma temporale che segue si riferisce ad un trasferimento dalla memoria centrale verso la Cache quindi lettura in RAM (lenta) e scrittura nella cache (veloce)



Frequenza di successo

Un indicatore eccellente dell'efficacia della particolare realizzazione adottata nella gerarchia di memoria è **la probabilità di successo** ottenuta nell'accesso alle informazioni ai vari livelli della gerarchia che raccorda la CPU al disco rigido, passando per la cache, e la memoria principale.

Il numero di successi indicati come **frazione di tutti gli accessi provati** viene chiamato ***frequenza di successo***, mentre la ***frequenza di fallimento*** rappresenta **il numero di fallimenti in rapporto al numero totale di accessi effettuati**.
I due valori sono tra loro complementari (la somma deve essere 1).

L’importanza della frequenza di successo

Teoricamente, se la ***frequenza di successo*** fosse 1 a tutti i livelli, l'intera gerarchia di memoria, dal punto di vista della CPU, apparirebbe come una singola unità, con il tempo di accesso di una cache sul chip del processore ma con le dimensioni di un disco magnetico.

- Quanto ci si avvicini a questo ideale dipende in misura notevole dalla frequenza di successo che si ottiene ai diversi livelli della gerarchia.
- Frequenze di successo elevate, ben oltre il 90%, sono essenziali per calcolatori a prestazioni elevate.

Penalità di fallimento

- Le prestazioni sono influenzate negativamente dalle azioni che devono essere effettuate quando l'accesso a un dato fallisce.
- Il tempo addizionale necessario per portare le informazioni desiderate nella cache è chiamato ***penalità di fallimento***.
- **Questa penalità si riflette, in ultima analisi, sull'intervallo di tempo durante il quale la CPU rimane in una situazione di stallo a causa della mancata disponibilità delle istruzioni o dei dati necessari per l'esecuzione.**

Penalità di fallimento (continua)

- **In generale, la penalità di fallimento è il tempo richiesto per portare il blocco di dati desiderato da un'unità più lenta a una più veloce nella gerarchia di memoria.**

- Essa viene ridotta se si individuano e si adottano meccanismi efficienti per il trasferimento dei dati fra i vari livelli della gerarchia di memoria.

Abbiamo visto come una memoria interallacciata possa ridurre in modo significativo la penalità di fallimento tra cache e memoria principale.

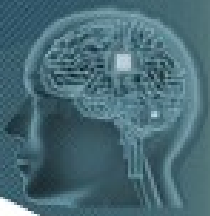
Incremento di prestazioni con la cache (1/3)

Valutiamo ora l'impatto della cache sulle prestazioni complessive del sistema di calcolo. Sia ***h*** la frequenza di successo, ***M*** la penalità di fallimento, ossia il tempo necessario per accedere alle informazioni nella memoria principale, e ***C*** il tempo per accedere alle informazioni nella memoria cache. Il tempo medio di accesso sperimentato dalla CPU è

$$t_{ave} = h\,C + (1 - h)\,M$$

Consideriamo un ***processore veloce*** senza cache e con ***memoria principale*** DRAM, per il quale sono necessari 10 cicli di clock per ogni accesso in lettura alla memoria.
Si supponga ora che allo stesso sistema sia aggiunta una memoria cache che contiene blocchi di otto parole e che la memoria principale sia interallacciata. In questo caso, supponiamo che siano necessari 17 cicli per caricare un blocco nella cache.

Incremento di prestazioni con la cache (2/3)



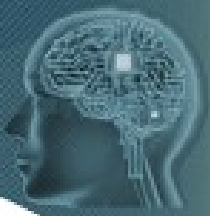
Si ipotizzi che il 30% delle istruzioni in un programma tipico effettui un'operazione di scrittura o di lettura, e che la frequenza di successo sia il 95% per le istruzioni e il 90% per i dati.

Si supponga, inoltre, che la penalità di fallimento sia la stessa per operazioni di scrittura e operazioni di lettura.

Con le ipotesi fatte, i tempi nei due casi sono:

$$\frac{Senza-cache}{Con-cache} = \frac{100\times 10}{70\left(0{,}90\times 1+0{,}05\times 17\right)+30(0{,}95\times 1+0{,}1\times 17)} = 4{,}90$$

Incremento di prestazioni con la cache (3/3)



Possiamo anche valutare quanto sia efficace la memoria cache con le prestazioni ipotizzate se confrontata con una cache ideale con una frequenza di successo pari al 100% (in tal caso, tutti gli accessi alla memoria richiedono un solo ciclo di clock).

$$\frac{70(0{,}95\times 1+0{,}05\times 17)+30(0{,}9\times 1+0{,}1\times 17)}{100}=2{,}04$$

In pratica la cache ipotizzata permette alla CPU di funzionare come se avesse una memoria principale, basata su DRAM, con tempi d'accesso che sono solo il doppio di quelli della cache.

Nell'esempio abbiamo supposto la frequenza di successo diversa per istruzioni e dati. Sebbene frequenze di successo con valori superiori a 0.9 siano ottenibili per entrambi, la frequenza di successo per le istruzioni è di solito maggiore di quella per i dati. Questi valori dipendono dal progetto della cache e dal profilo di accesso a istruzioni e dati che caratterizza il programma in esecuzione.

Una cache per le istruzioni ed una per i dati?

Per i motivi che abbiamo più volte ricordato **il posto migliore in cui collocare una cache è il chip della CPU**. Sfortunatamente, però, si trova posto solo per cache piccole.
Tutti i processori con prestazioni elevate hanno una cache “on chip”.

Alcuni produttori hanno deciso di realizzare due cache separate, una per le istruzioni e un'altra per i dati, come nei processori 68.040 e PowerPC 604. In altri casi è stata adottata una sola cache per istruzioni e dati, come nel processore PowerPC 601.

Una cache per le istruzioni ed una per i dati?

Una sola cache per istruzioni e dati in teoria dovrebbe avere una frequenza di successo superiore, poiché offre una maggiore flessibilità nella memorizzazione di nuove informazioni.

Tuttavia, se si utilizzano cache separate, è possibile accedere contemporaneamente a entrambe le memorie cache, aumentando così il parallelismo e, di conseguenza, le prestazioni della CPU.

Lo svantaggio delle cache separate è che l'incremento del grado di parallelismo è accompagnato da circuiti molto più complessi.

Una cache secondaria

Poiché le dimensioni della memoria cache sul chip della CPU sono limitate, una buona strategia consiste nell'utilizzare tale cache come ***cache primaria***.

Una *cache secondaria* esterna, costruita con chip SRAM, viene poi aggiunta per ottenere la capacità desiderata.

In questo caso la cache primaria deve essere progettata in modo tale da consentire accessi molto veloci da parte della CPU, visto che il suo tempo di accesso ha un'influenza significativa sulla frequenza di clock della CPU.

Ma non si può accedere a una memoria cache alla stessa velocità con cui si accede al banco di registri interni della CPU, perché la cache ha dimensioni maggiori ed è più complessa.

Utilità di una cache secondaria

La cache secondaria può essere anche notevolmente più lenta, ma dovrebbe essere molto più grande di quella primaria in modo tale da garantire una frequenza di successo elevata negli accessi.

La velocità di questa cache è meno critica perché influenza soltanto la penalità di fallimento della cache primaria.

Una workstation può avere una *cache primaria* con una capacità di decine di kilobyte e una *cache secondaria* di diversi megabyte.

La presenza di una cache secondaria riduce ulteriormente l'impatto della velocità della memoria principale sulle prestazioni del calcolatore.

Tempo di accesso con due livelli di cache

Il tempo medio di accesso visto dalla CPU è:

$$t_{ave} = h_1 C_1 + (1 - h_1)\, h_2\, C_2 + (1 - h_1)(1 - h_2)\, M$$

dove i parametri sono definiti come segue:

h_1 = frequenza di successo nella cache primaria;
h_2 = frequenza di successo nella cache secondaria;
C_1 = tempo di accesso alle informazioni nella cache primaria;
C_2= tempo di accesso alle informazioni nella cache secondaria;
M = tempo di accesso alle informazioni nella memoria principale.

Il numero di fallimenti di accesso nella cache secondaria, dato dal termine **$(1 - h_1)(1 - h_2)$**, dovrebbe essere basso.
Se h_1 e h_2 raggiungono entrambi valori dell'ordine del 90%, allora il numero di fallimenti sarà inferiore all'1% degli accessi alla memoria da parte della CPU. Quindi, la penalità di fallimento M sarà meno critica dal punto di vista delle prestazioni.

La Memoria Virtuale

Sommario

- Premesse storiche.
- Compiti svolti dalla memoria virtuale in un moderno sistema
- Supporti hardware della MMU: la tabella dei numeri di pagina.
- Funzioni di controllo: bit di validità e bit di modifica.
- Registro base della tabella dei numeri di pagina.
- Il buffer per la traduzione degli indirizzi (TLB).
- Terminologia del sistema di memoria virtuale

La Memoria Virtuale

Agli albori dell'informatica la memoria centrale di un computer raramente superava le 4k parole ed i sistemi operativi erano, a dir poco, essenziali.

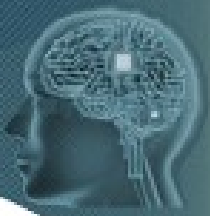
A quell'epoca, chiunque si disponesse a scrivere un programma doveva prima informarsi di quale fosse la disponibilità di spazio nella memoria centrale del sistema da adoperare, al netto delle necessità della CPU e della "parte residente" del sistema operativo.

Fatto il conteggio doveva scrivere un programma fatto di pezzi da mandare in esecuzione uno per volta (***overlay,* sovrapposizione**), registrando i risultati parziali in gruppi di memoria i cui indirizzi fossero noti anche ai successivi pezzi di programma.

Questi li prelevavano per usarli ed, a loro volta registravano i risultati in locazioni prefissate.

Senza questa procedura il programma non poteva essere eseguito.

La Memoria Virtuale



Tutto questo è oggi completamente ignoto agli utilizzatori di computer.

Il fatto che talvolta siano indicati dei requisiti minimi del sistema per usare un certo programma, è un semplice suggerimento per ottenere dal programma un comportamento ragionevole, senza lunghissimi tempi di attesa tra due operazioni successive.

Oggi è assolutamente normale che programmi che tra occupazione diretta ed indiretta avrebbero bisogno di 25-30 Mbyte girino su macchine con memoria centrale da 16 Mbyte, grazie alla tecnica della "***memoria virtuale***", inventata da un gruppo di ricercatori inglesi di Manchester nel 1961 ed adottata dall'IBM negli anni '70.

La tecnica si avvale di strutture hardware che è fondamentale che siano nel circuito integrato della CPU e ciò spiega perché essa è apparsa nei sistemi basati su microprocessori "single chip" relativamente di recente.

La gestione è affidata al sistema operativo

In estrema sintesi la tecnica della *memoria virtuale* prevede che il sistema operativo assegni una parte degli spazi liberi nella memoria centrale ai segmenti di un programma istallato su disco.

Il sistema operativo seguirà criteri prefissati, assegnando uno spazio compreso tra un minimo ed un massimo, in funzione della dimensione del programma.

Gli spazi potranno essere anche non continui.

Il risultato è che programmi piccoli saranno trasferiti interamente ma, naturalmente, in funzione della mappa di occupazione corrente potranno essere dislocati in maniera frammentata in qualsiasi parte della memoria.

La memoria è gestita dinamicamente

Programmi di grandi dimensioni avranno a disposizione spazi insufficienti a contenerli completamente e saranno soggetti a continue operazioni di "***swapping***", cioè di sostituzione dinamica dei loro segmenti nello spazio di memoria centrale assegnato.

L'operatore può imporre al sistema operativo di dedicare più spazio ad un programma, ma se esso usa una gran parte del campo d'indirizzamento del processore, non potrà in ogni caso essere completamente trasferito in memoria.

La tecnica di gestione degli spazi ha molti punti in comune con il funzionamento di una memoria cache.

Anche in questo caso, infatti, se gli spazi a disposizione nella memoria principale sono saturati, e il programma accede ad una locazione di memoria che non corrisponde ad un indirizzo esistente nella memoria centrale, si deve ricorrere ad un algoritmo di sostituzione.

Dimensioni della pagina di memoria

La parte di programma contenente l'indirizzo richiesto viene prelevato dal disco (trasferimento in DMA) e sostituita in memoria ad una delle parti che vi risiedevano.

Poiché il tempo di accesso al disco costituisce una penalità di fallimento molto più pesante, in termini di tempo rispetto a quella di una cache, l'entità minima da trasferire è presa molto più grande del blocco della cache ed è una *pagina di memoria* di solito compresa tra 2k e 16k; normalmente in un accesso al disco vengono scambiate molte pagine.

La Memory Management Unit (MMU)

Quello a cui fa riferimento il processore nell'elaborazione è detto indirizzo "***virtuale***" o logico ed ha come unica limitazione di essere entro il campo di indirizzamento massimo del processore.

Il compito di tradurre questo indirizzo in un indirizzo che punti ad una locazione effettivamente esistente nel sistema (***indirizzo fisico***) è demandato ad una struttura logica contenuta nel chip del processore, che si chiama **Memory Management Unit (MMU)** e che opera sotto il controllo software del sistema operativo.

La Memory Management Unit (MMU)

Page

un blocco di dati (o istruzioni) che può essere scritto (memorizzato) su un frame pagina

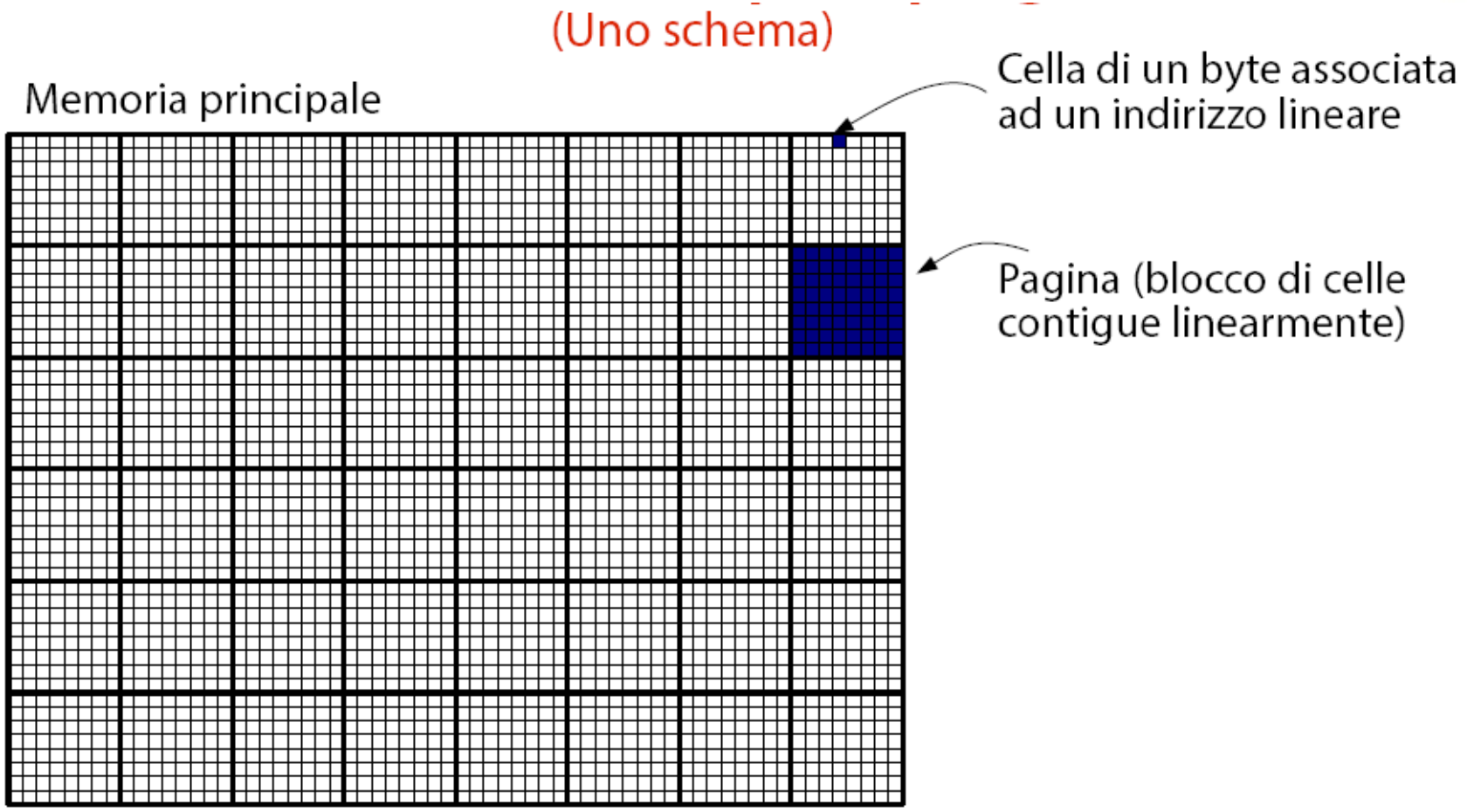
- il termine pagina si riferisce sia alla pagina di ind. lineari che alla pagina di dati memorizzati in RAM in corrispondenza
- Una frame pagina memorizza esattamente il blocco di dati (o di istruzioni) contenuti in una pagina (di indirizzi lineari)
- La paginazione mappa pagine di indirizzi lineari su frame pagine (e sui dati in essi memorizzati)
- Una pagina è in RAM o su disco

Page Frame

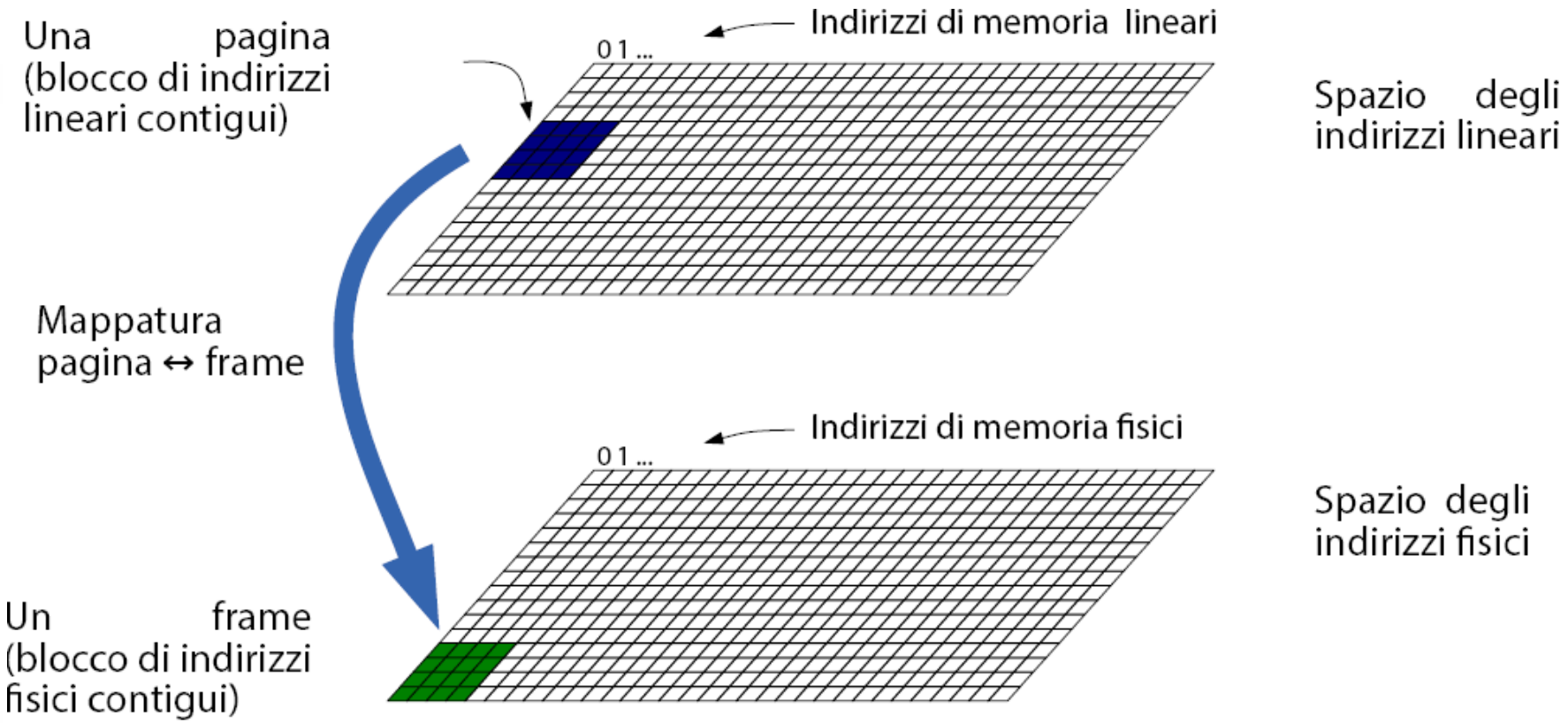
una area fisica (intervallo di locazioni contigue) di memoria RAM di dimensione fissata (su Intel x86 4KB o 4MB)

La Paginazione

(Uno schema)

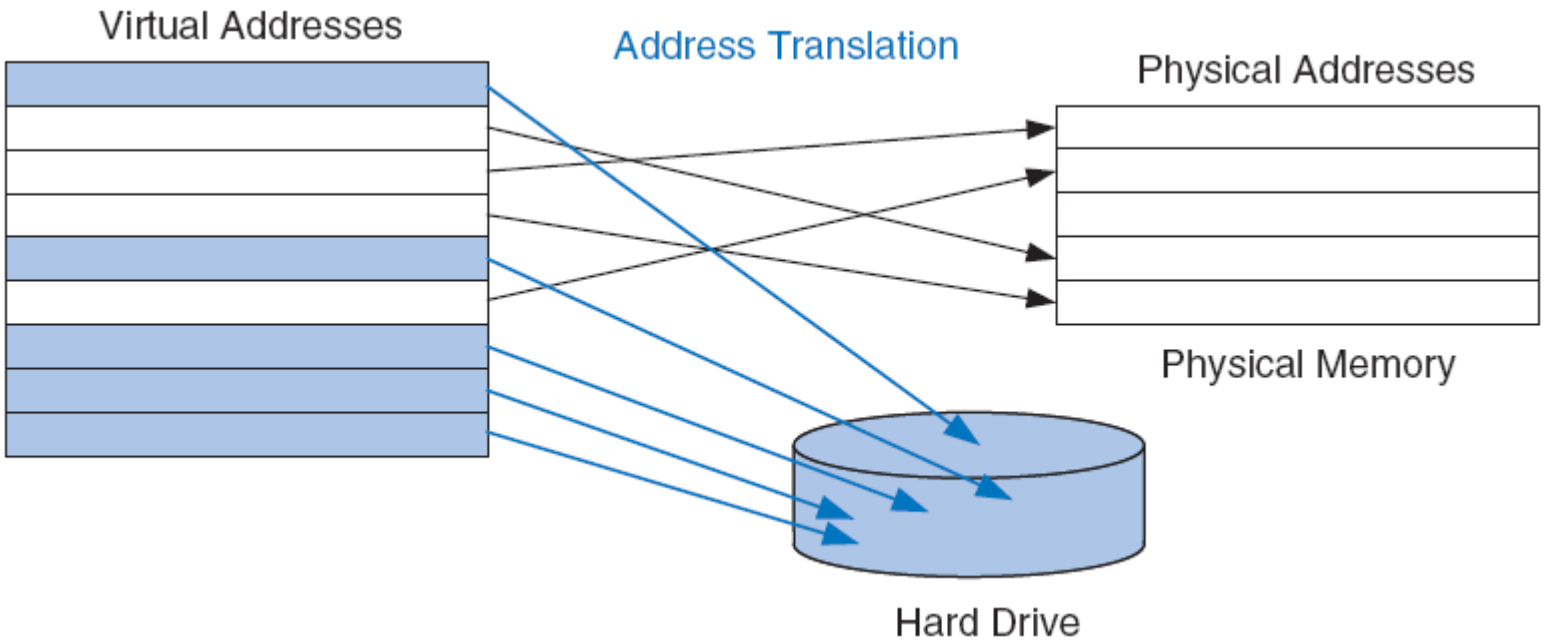


La Paginazione

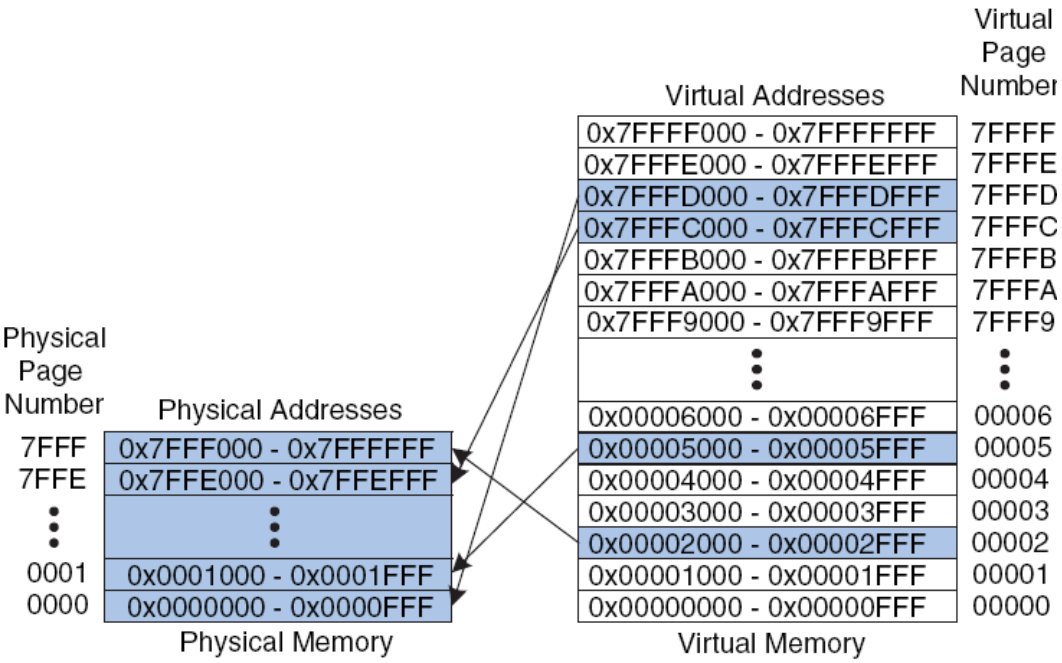


Traduzione degli indirizzi

Durante l'esecuzione di un programma alcune delle sue pagine sono residenti in memoria e altre su disco. Durante il processo di **traduzione degli indirizzi** (virtuale→fisico) una **pagina virtuale** può essere mappata su qualunque **pagina fisica**.



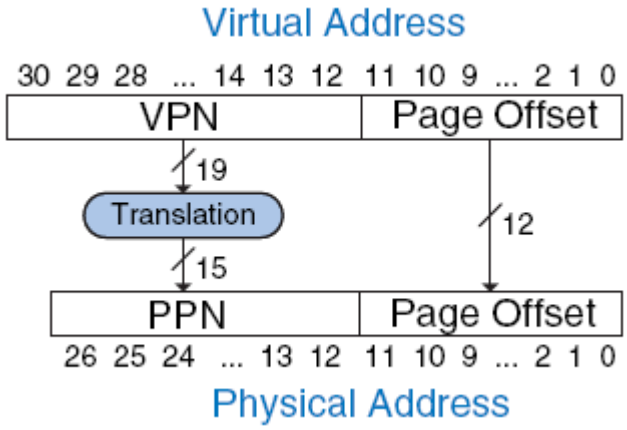
Traduzione degli indirizzi



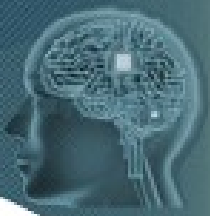
In un sistema reale con **2Gb** di **memoria virtuale** e **128 Mb** di **memoria fisica** e pagine di **4Kb**. Gli indirizzi sono tutti a **32 bit**. I bit più significativi non necessari sono sempre posti a 0. Inoltre ci sono $2^{31}/2^{12}=\mathbf{2^{19}}$ **pagine virtuali** e $2^{27}/2^{12}=\mathbf{2^{15}}$ **pagine fisiche**.

In un **indirizzo virtuale** **19 bit** individuano la pagina e **12 bit** (**offset**) la parola al suo interno.

In un **indirizzo fisico** **15 bit** individuano la pagina e **12 bit** (**offset**) la parola al suo interno.



Traduzione degli indirizzi



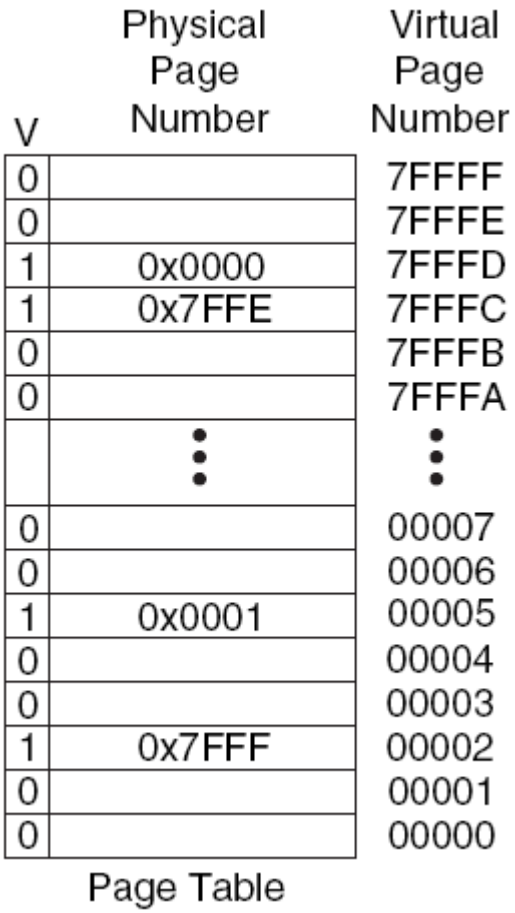
La memoria virtuale si comporta come una sorta di cache completamente associativa per il disco.

Tuttavia, se si gestisse la memoria virtuale come le cache, ossia tramite un comparatore sui bit più significativi per verificare la corrispondenza delle pagine, servirebbe un numero improponibile di comparatori, dato l'elevato numero di pagine in cui è organizzata la RAM.

In alternativa, il processo di traduzione degli indirizzi è gestito mediante **tabelle di paginazione**.

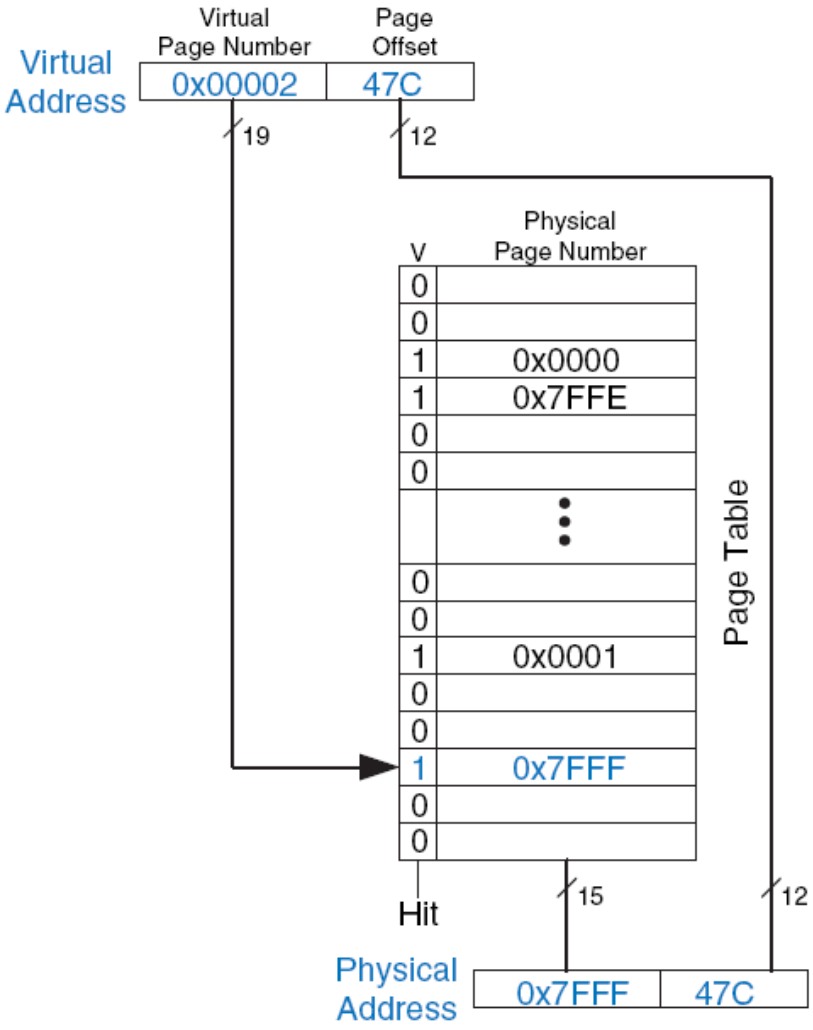
Traduzione degli indirizzi

La **tabella delle pagine** è vista come un array, in cui ciascuna entry specifica la pagina fisica corrispondente alla pagina virtuale con tale indice. Per ogni entry è, inoltre, impostato un **bit di validità**, che è **0** se la pagina non è caricata in RAM, **1** altrimenti.



Page Table

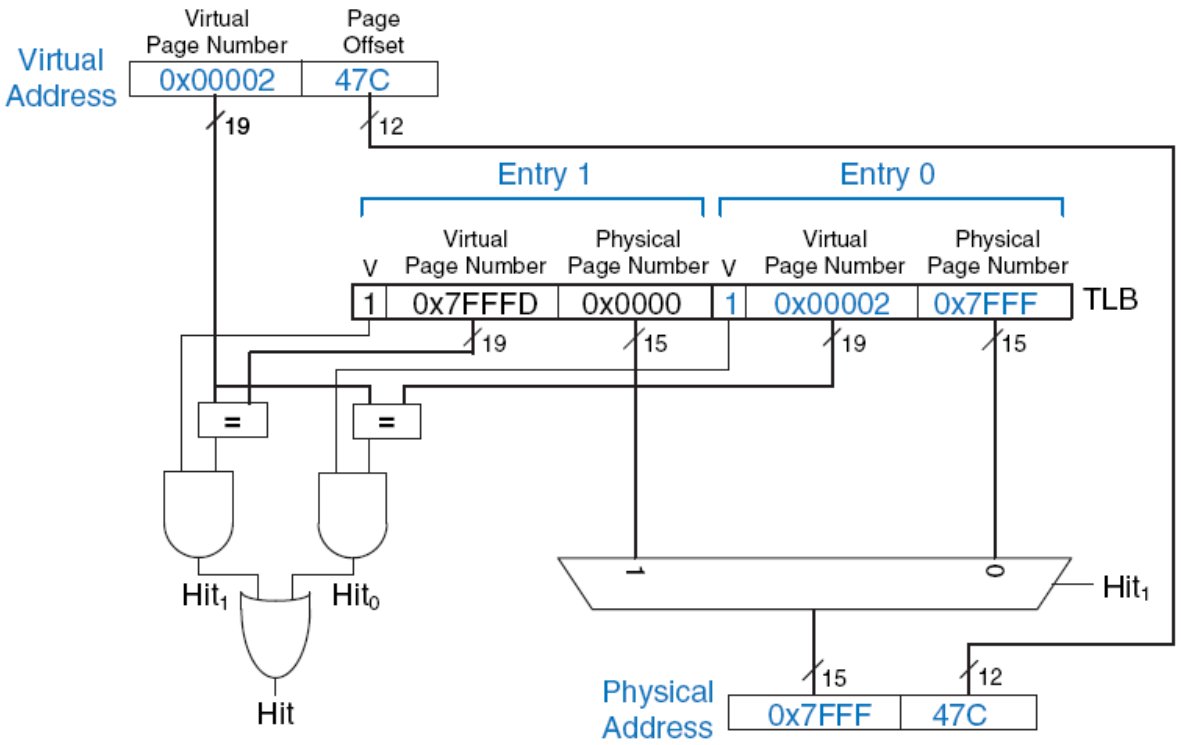
Traduzione degli indirizzi



Nel processo di traduzione, l'indirizzo logico è scomposto in due parti. I **19 bit** di **etichetta** indicano la posizione nella tabella in cui recuperare l'**indirizzo di base** della pagina fisica al quale vengono sommati i **12 bit** meno significativi (**offset**), che individuano la parola all'interno della pagina.

Il Translation Look Aside Buffer (TLB)

Poiché la tabella delle pagine risiede nella memoria fisica, accedere ad essa per la traduzione degli indirizzi produce un certo overhead. Al fine di ridurre tale overhead, gli indirizzi già tradotti sono memorizzati in una cache detta TLB.



Il TLB si comporta come una cache completamente associativa, con un numero di entry che può variare fra 16 e 512.

Tabelle delle Pagine

I sistemi moderni dispongono di uno spazio di indirizzamento logico molto grande (2^{32}, 2^{64} elementi).
La tabella delle pagine è gigantesca.

Si usa uno **schema di paginazione gerarchico**.

Ciascun elemento della tabella delle pagine punta all'inizio di una tabella più piccola.

Le tabelle più piccole puntano ai frame fisici.

Tale riorganizzazione della tabella delle pagine richiede una specifica organizzazione del formato degli indirizzi lineari.

Tabelle delle Pagine

Lo schema a due livelli consente di ridurre la quantità di memoria necessaria per allocare le tabelle di paginazione.

Ogni processo ha una sola **Global Page Directory** e da **1** a **1024** **Page Table** (allocate via via che il processo richiede memoria).

Ogni **Page Table** ha **1024** elementi (indirizza **1024** frame pagine) e Ogni processo può indirizzare: **1024** x **1024** x **4096** = $\mathbf{2^{32}}$ celle di memoria.

Tabelle di Paginazione

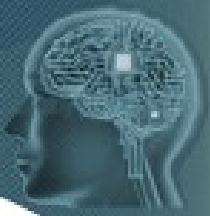


Tabelle di Paginazione

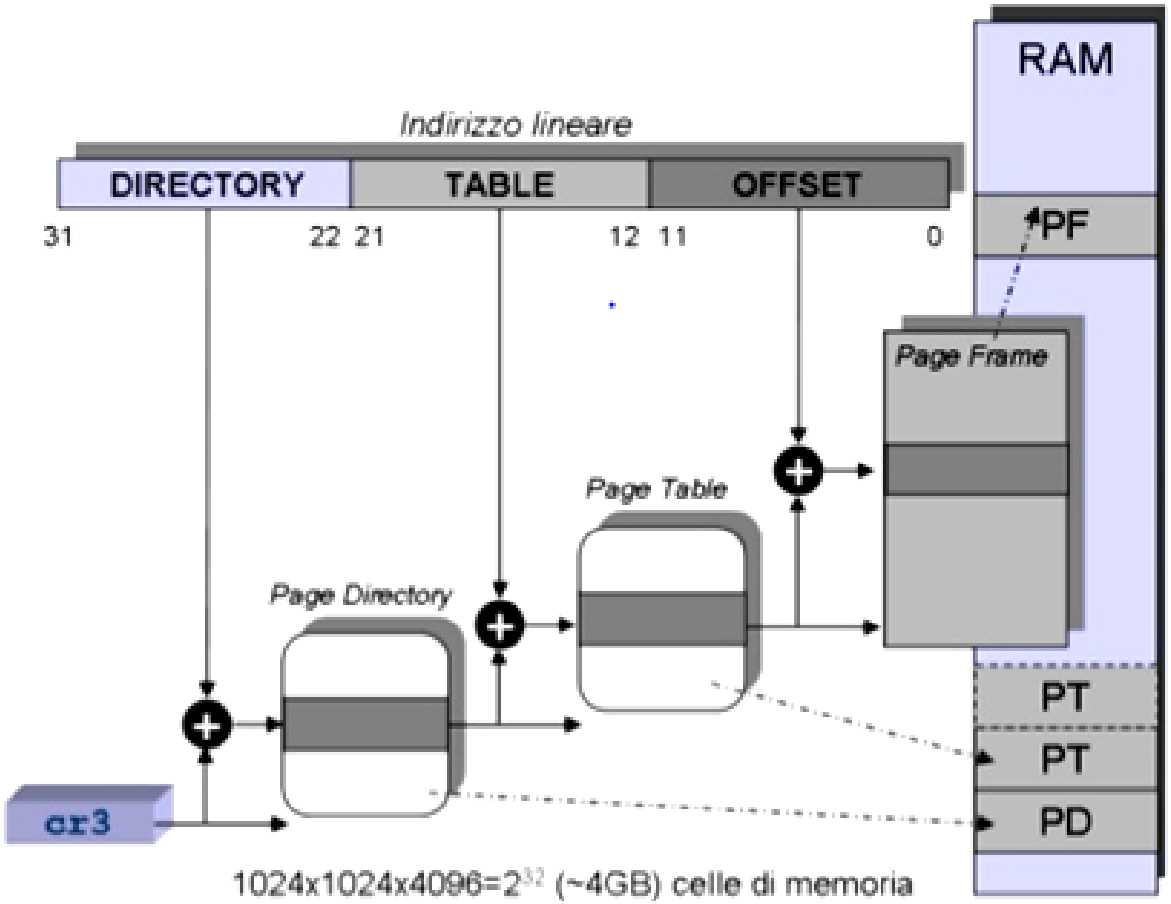
insieme di strutture dati in RAM usate per mappare indirizzi lineari in indirizzi fisici (e nelle pagine fisiche di appartenenza)

- Le tabelle di paginazione sono allocate direttamente in RAM e devono essere propriamente inizializzate/scritte in RAM dal kernel
- Ogni processo ha le proprie tabelle di paginazione
- L'indirizzo lineare è diviso in tre campi:

  DIRECTORY : 10 bit più significativi
  TABLE : 10 bit intermedi
  OFFSET : 12 bit meno significativi

Tabelle di Paginazione

Figura: Paginazione a due livelli



$1024 \times 1024 \times 4096 = 2^{32}$ (~4GB) celle di memoria