Gestione della memoria

Moduli del Sistema Operativo

Scopi del gestore della memoria

I compiti del gestore della memoria sono:

- Mantenere traccia delle zone della memoria libere e delle zone occupate.
- Assegnare i segmenti di memoria ai processi che ne fanno richiesta.
- Recuperare lo spazio di memoria rilasciata da un processo che termina.
- Gestire le operazioni di trasferimento tra memoria di massa e memoria centrale: swap-in e swap-out

Descrittore di segmento

I segmenti allocati nella memoria ai vari processi possono essere

- **globali**, accessibili a tutti i processi,
- **privati**, accessibili solo al processo che li possiede.

i segmenti sono caratterizzati da alcuni attributi:

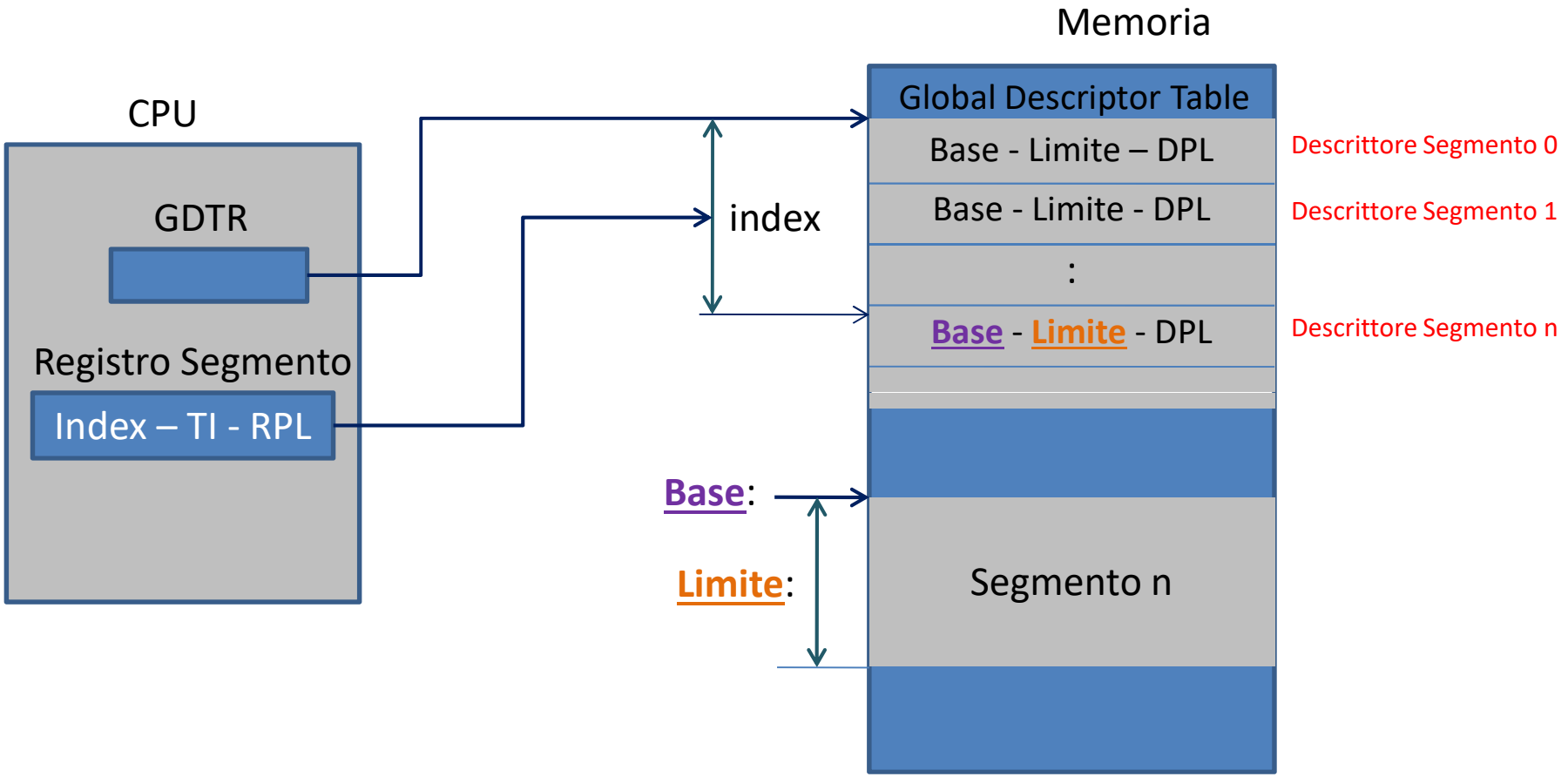
- indirizzo base,
- dimensione,
- protezione contro modifiche,
- ecc...,

tutti riepilogati in un **descrittore di segmento**

Tabelle dei descrittori

- Se il segmento è accessibile a più processi, il suo descrittore si trova nella tabella dei descrittori globali (GDT), se invece il segmento è riservato a un singolo processo il descrittore viene inserito nella tabella dei descrittori locali (LDT). Il sistema operativo, in fase di inizializzazione, decide la posizione in memoria della GDT e, tramite un'istruzione privilegiata, cioè non eseguibile da un programma utente, annota nel registro GDTR della CPU l'indirizzo di inizio di tale tabella.

Accesso in memoria



Index: Indice del descrittore nella tabella
TI: Table Indicator
GDT: Global Descriptor Table (Tabella dei descrittori Globali)
GDTR: Global Descriptor Table Register (contiene il puntatore alla tabella dei descrittori globali)

Tabelle dei descrittori

- Il GDTR è un registro interno alla CPU che contiene l'indirizzo base della GDT in memoria.
- Ogni elemento della GDT è un descrittore di segmento che contiene tutte le informazioni per accedere al segmento.
- La GDT è unica e conserverà sempre la sua posizione in memoria, quindi il GDTR non verrà più modificato.
- Una LDT invece è associata all'esistenza di un processo, quindi occuperà un segmento di memoria libero nel momento in cui nasce il processo, e di conseguenza, per ogni processo esiste una LDT a cui il sistema operativo assocerà un descrittore di segmento e lo collocherà nella GDT.

Selettore di segmento

Per formare un indirizzo in memoria, la CPU usa un **selettore**. Un registro segmento non contiene più un indirizzo base di segmento ma un selettore, un record di 16 bit così organizzato:

Index	Table Indicator	Request Privilege Level
13 bit	1 bit	2 bit

Il campo Index è un campo di 13 bit, ma viene trattato dalla CPU come un valore a 16 bit avente i 3 bit inferiori a livello 0; in questo modo il campo index rappresenta sempre un numero multiplo di 8 interpretato come l'indice di un descrittore all'interno di una delle due tabelle LDT o GDT.

Il campo TI (Table Indicator) se contiene 0 specifica che il campo Index si riferisce alla GDT, se contiene 1 specifica che il campo Index si riferisce alla LDT.

Il campo RPL (Request Privilege Level) stabilisce il livello di autorizzazione minimo richiesto per accedere al segmento.

Quando viene inserito un selettore in un registro segmento, la CPU copia il descrittore dalla tabella in memoria in un registro *cache* affiancato al registro segmento .

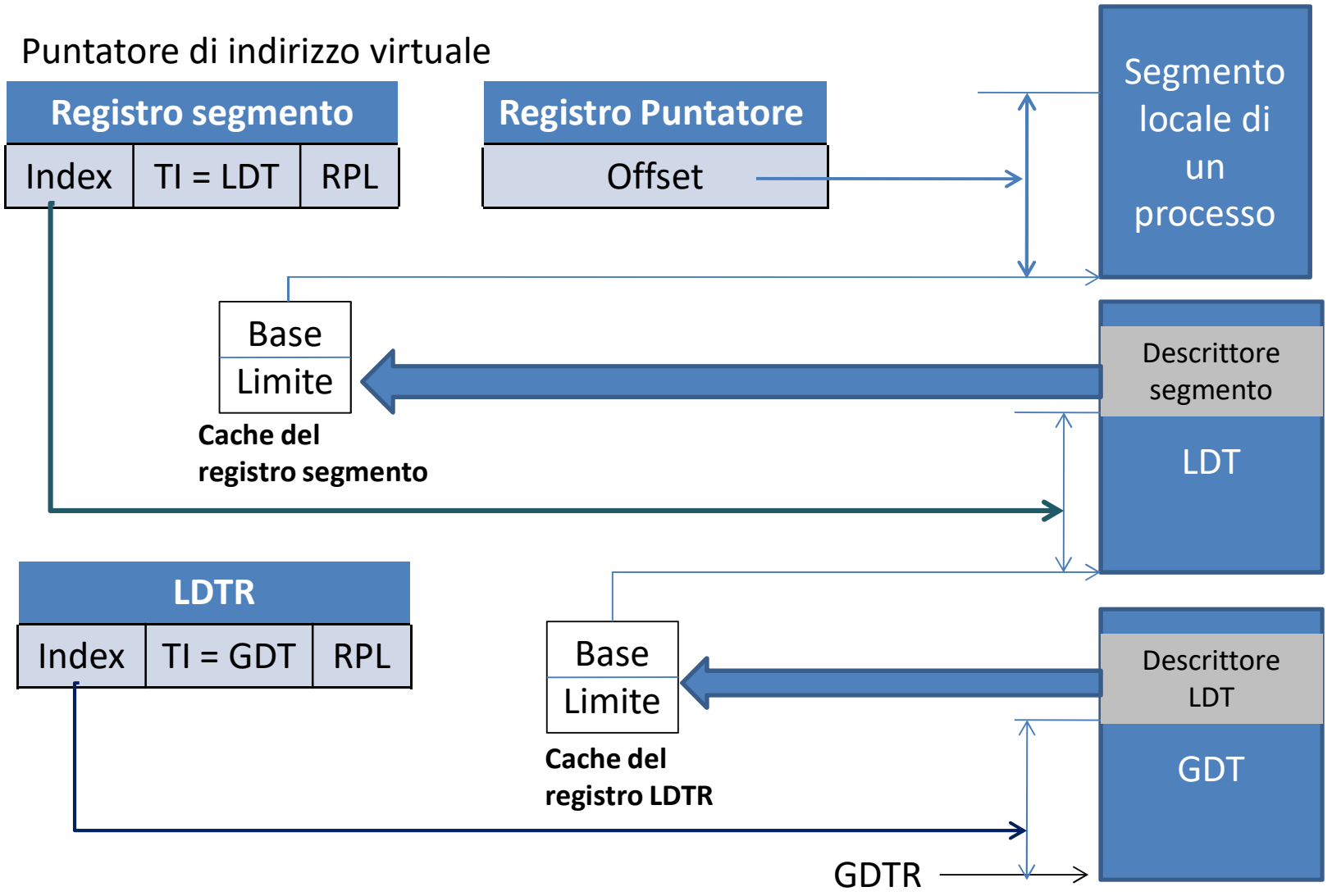
Descrittore di segmento

Il descrittore di segmento è un record di 8 byte (64 bit), e contiene i seguenti campi:

Indirizzo Base			32 bit		
Limite			20 bit		
Diritti di accesso	DPL	R/W	U/D	P	A

- Base: 32 bit contiene l'indirizzo base del segmento, quindi consente di specificare 4 GB di memoria;
- Limite: lungo 20 bit contiene la dimensione del segmento, max 1 MB;
- Byte di accesso è un record così organizzato:
  - DPL: (Descriptor Privilege Level) è un numero di due bit che la CPU confronta con il corrispondente RPL del selettore per concedere o meno a un processo l'autorizzazione di accesso al segmento;
  - R/W: (Leggibile o scrivibile)
  - U/D: (Up/Down) direzione in cui cresce il segmento, (lo stack si espande verso indirizzi bassi);
  - P: (Presente) indica se il segmento riferito è presente o meno nella memoria centrale;
  - A: (Accesso) viene posto a 1 dopo ogni operazione sul segmento e viene azzerato dal sistema operativo, che periodicamente aggiorna una tabella che mantiene il tempo trascorso dall'ultimo riferimento al segmento

Composizione dell'indirizzo

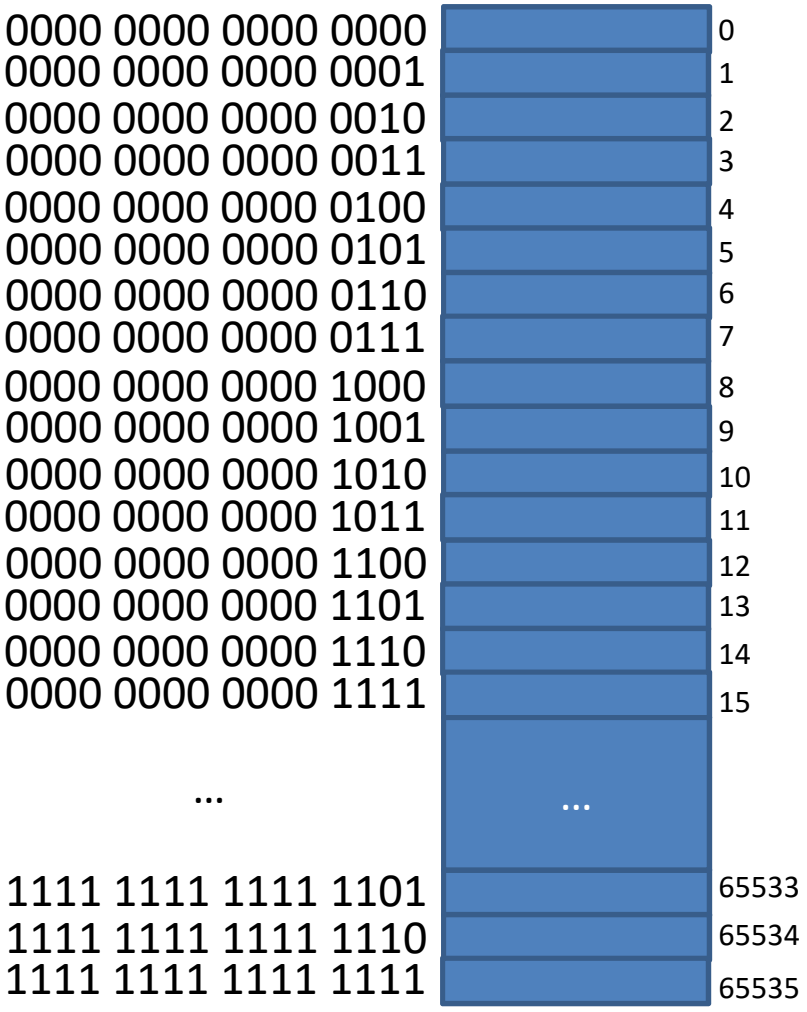


La paginazione

Gestione della memoria

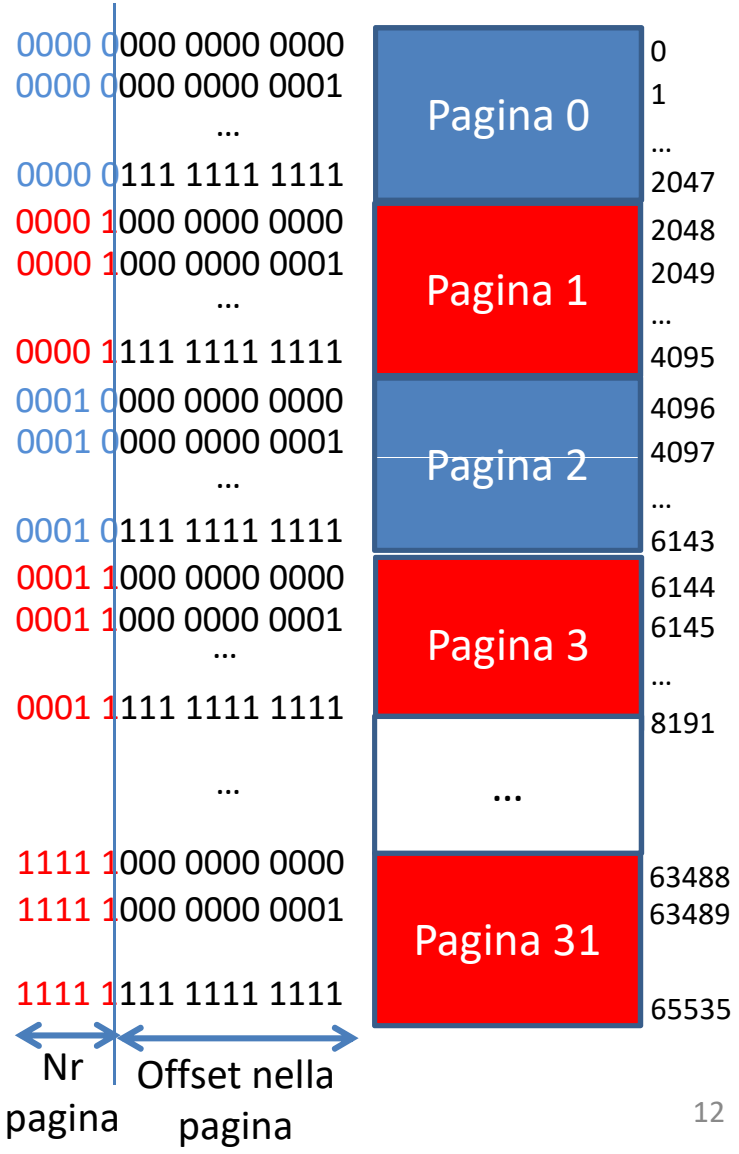
Indirizzamento lineare

- Le locazioni di una memoria di N Byte sono numerate da 0 a N-1.
- Se N=65536, occorrono 16 bit per indirizzare una locazione:



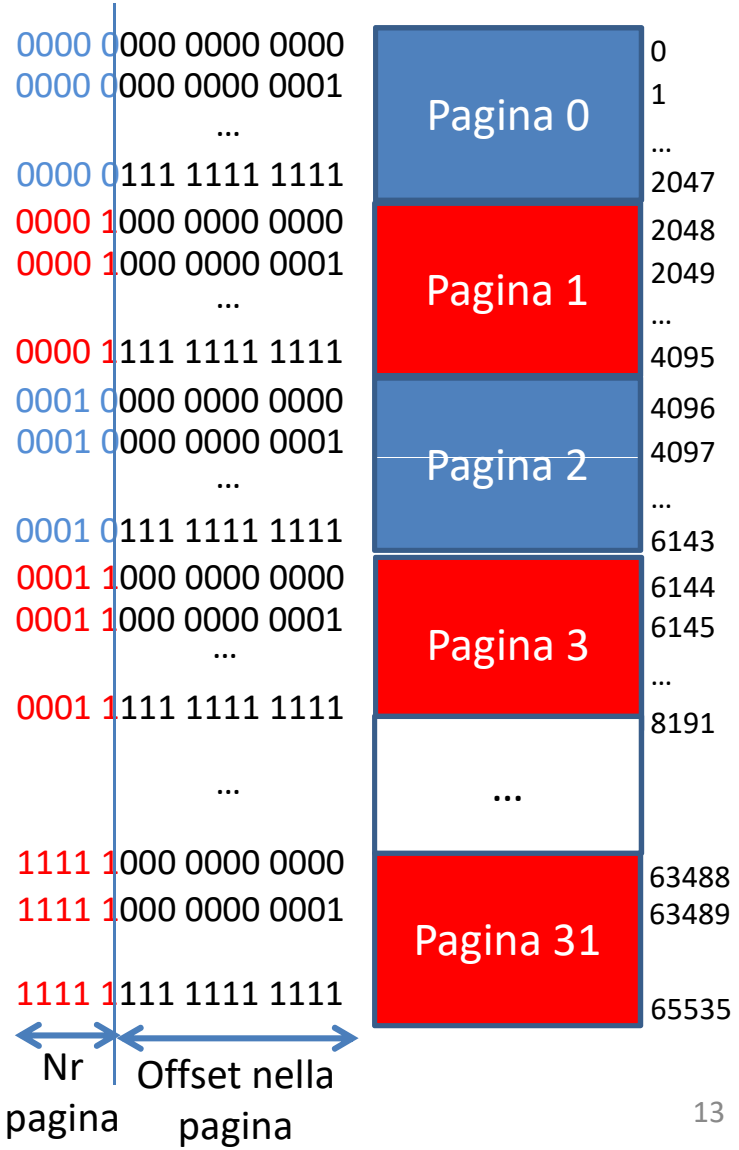
Indirizzamento con base e offset

- La memoria viene divisa in blocchi di uguale lunghezza.
- Ad esempio, una memoria di 64KB viene divisa in blocchi di 2KB.
- Quindi ci sono 32 blocchi.
- Per indirizzare una delle 2048 locazioni interne ad un blocco occorrono 11 bit
- Per indirizzare una delle 32 pagine occorrono 5 bit.



Indirizzamento con base e offset

- Una locazione della memoria può essere specificata in due modi differenti: con indirizzamento lineare o con base e offset.
- Ad esempio la locazione 6145 può essere indirizzata, in modo alternativo, come offset 1 nella pagina 3.



La paginazione

La memoria viene divisa in blocchi di lunghezza $\mathbf{L_b}$.

I programmi sono suddivisi in pagine della stessa lunghezza.

Il sistema operativo assegna i blocchi alle pagine dei programmi consultando la tabella di occupazione della memoria. Mentre il programma è caricato in memoria, il sistema operativo costruisce la tabella di traduzione degli indirizzi.

La tabella di occupazione della memoria tiene conto dei blocchi liberi e la tabella di traduzione degli indirizzi associa la pagina del programma al blocco di memoria in cui è caricata.

Dato un indirizzo I

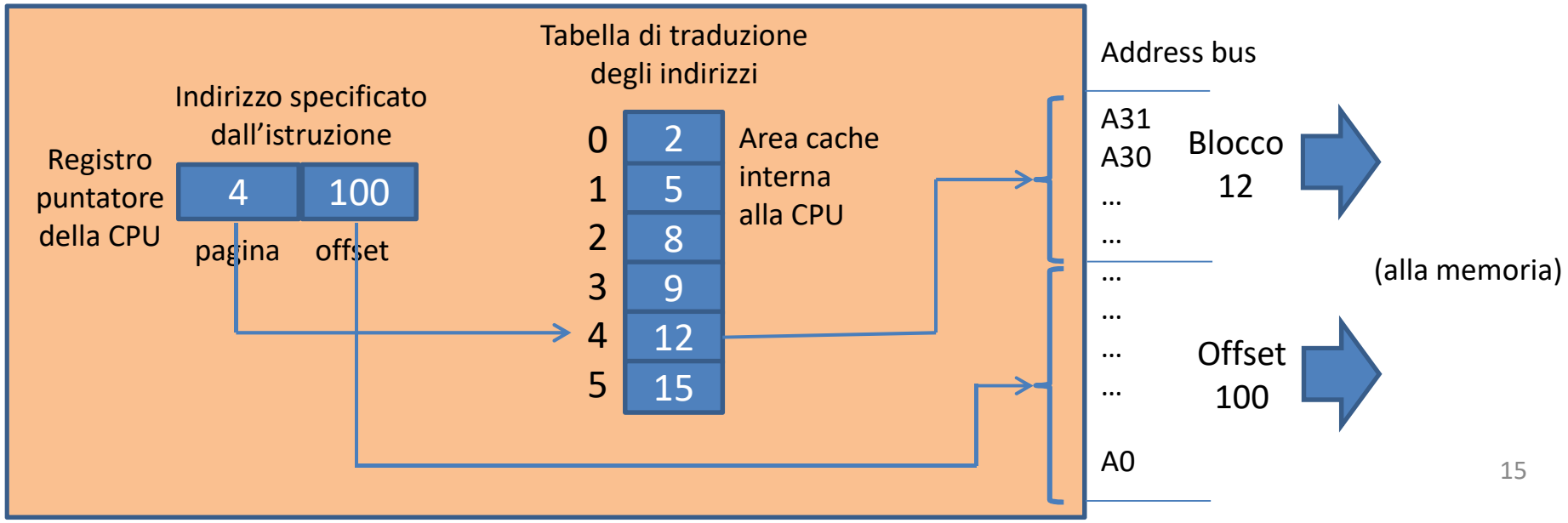
- il numero di pagina a cui appartiene è: P = quoziente(I / L_b)
- mentre l'*offset* corrispondente è: offset = resto(I / L_b)

Viceversa, dato un indirizzo nella forma (P, offset) per calcolare il corrispondente indirizzo fisico si deve prelevare, dalla tabella di traduzione degli indirizzi, il numero di blocco *b* in cui è collocata la pagina p e moltiplicarlo per L_b, quindi sommare l'offset:

$$I = P \cdot L_b + \text{offset}.$$

Traduzione di un indirizzo

- Un programma specifica un indirizzo in forma relativa, ovvero riferito a una locazione di riferimento, ad esempio l'inizio del programma.
- Il sistema operativo estrae il numero di pagina e l'offset dall'indirizzo specificato dall'istruzione.
- Usa il numero di pagina specificato dall'istruzione per accedere alla tabella di traduzione degli indirizzi e determinare il blocco di memoria occupato dalla pagina.



La Memoria Virtuale

Gestione della memoria

La memoria Virtuale

La tecnica della memoria virtuale consiste nel rendere disponibile, ai programmi in esecuzione, tutto lo spazio degli *indirizzi logici* che la CPU può riferire, ad esempio:

Se il bus indirizzi della CPU è composto da 32 linee, i programmi possono indirizzare fino a 4 GB, anche se la memoria fisica è di dimensione minore.

Tutti i dispositivi di memoria secondaria (generalmente i dischi, ma non sono esclusi anche dispositivi lenti come i nastri magnetici) vengono considerati un'estensione della memoria centrale. I programmi non vengono trasferiti completamente in memoria, ma nel corso dell'esecuzione vengono prelevate le parti necessarie e vengono eliminate le parti che non si useranno più.

La memoria Virtuale

La memoria centrale è suddivisa in blocchi la cui lunghezza è fissata in un multiplo di un settore di disco.

Ad ogni programma viene assegnato uno *spazio di lavoro*, formato da un certo numero di blocchi, la cui dimensione ottimale scaturisce da un compromesso tra due fattori:

- blocchi lunghi richiedono molto tempo per trasferire i dati tra disco e memoria,
- blocchi corti provocano frequenti accessi al disco per prelevare i dati che non sono presenti in memoria.

La tecnica della memoria virtuale è conveniente se i processi hanno a disposizione uno *spazio di lavoro* sufficientemente ampio da minimizzare la frequenza di prelievi di *pagine assenti*. Il criterio per dimensionare lo *spazio di lavoro* è che il tempo per prelevare una *pagina* dal disco sia trascurabile rispetto al tempo durante il quale il programma si intrattiene sulle *pagine* presenti in memoria.

Tabella di occupazione della memoria

Lo spazio fisico di 16 MB è diviso in 4096 segmenti di 4 KB e il sistema operativo gestisce una tabella di 4096 descrittori che contengono i seguenti campi relativi allo stato di ciascun segmento:

	Descrittore di blocco				
Nr	Tipo pagina	Indirizzo su disco	Ultimo riferimento	Processo	permanente

tipo di pagina: indica se il segmento è libero, è occupato, contiene una tabella di traduzione degli indirizzi o un elenco di tabelle,
indirizzo su disco: contiene la posizione della pagina sulla memoria secondaria,
ultimo riferimento: è gestito dal sistema operativo quando deve scegliere la pagina meno recentemente riferita, per eseguire l'operazione di swapping,
nome processo: identifica il processo a cui appartiene la pagina.

La posizione (Nr) del record nella tabella è in corrispondenza biunivoca con il numero fisico del blocco in memoria, pertanto è superfluo memorizzare l'indicazione del blocco a cui si riferisce il record.
La tabella di occupazione della memoria serve al sistema operativo per trovare i segmenti liberi o per applicare l'algoritmo LRU e procedere all'operazione di swapping.

Tabella di Traduzione degli indirizzi

Le istruzioni usano un puntatore di indirizzo virtuale per riferire una locazione nello spazio di 4 GB degli indirizzi logici, e il sistema operativo compie l'associazione con un indirizzo effettivo nello spazio fisico di 16 MB tramite una **tabella di traduzione degli indirizzi**. Un elemento di tale tabella è suddiviso nei seguenti campi:

Campo	Significato
Presente	Se vale 1 la pagina è presente in memoria, altrimenti deve essere prelevata dal disco, in questo caso si verifica l'eccezione di pagina assente, che corrisponde all'attivazione dell'operazione di swapping, al termine della quale il bit è posto a 1 e il processo prosegue l'esecuzione.
Accesso	é posto a livello 1 dopo ogni accesso alla pagina, ed è azzerato dal sistema operativo quando calcola l'intervallo di tempo trascorso dall'ultimo riferimento alla pagina.
Modificata	Dopo un'operazione di scrittura nella pagina, questo bit è posto a 1, per decidere, in fase di swapping se la pagina deve essere copiata su disco
Indirizzo	il contenuto di questo campo è considerato la parte alta di un numero a 32 bit, in cui i 12 bit meno significativi sono tutti 0. In tal modo i numeri rappresentabili sono tutti multipli di 4096, cioè sono l'indirizzo iniziale di una pagina.

Tabella di Traduzione degli indirizzi

La tabella contiene 1024 record associati ad altrettante pagine in cui può essere suddiviso il programma. Di conseguenza a un processo è assegnato uno spazio complessivo di 4 MB di memoria virtuale.

Come esempio supponiamo che il sistema operativo abbia assegnato i segmenti 100, 101 e 102 alle pagine 0, 1, 2 di un programma A.

Nella tabella di occupazione della memoria, il sistema operativo annoterà le informazioni necessarie:

Tabella di occupazione della memoria

Nr. blocco	Stato pagina	Posizione su disco	Ultimo Riferimento	Nome processo
…	…	…	…	…
100	occupata	N° settore	8	A
101	occupata	N° settore	5	A
102	occupata	N° settore	14	A
…	…	…	…	…

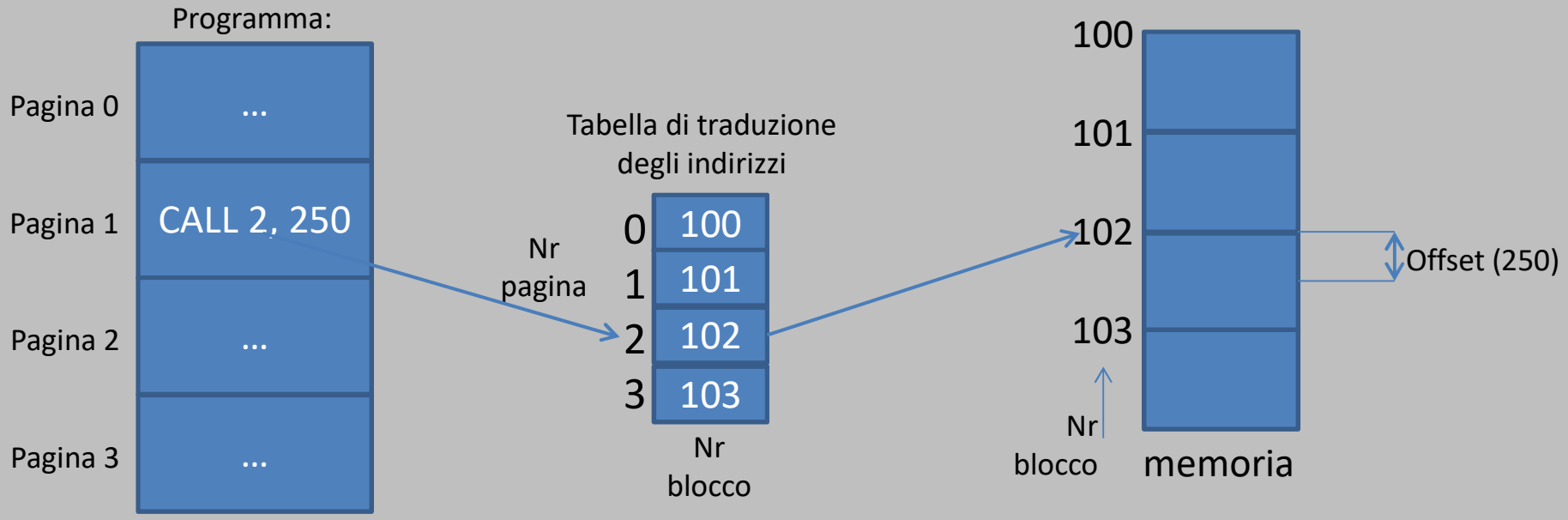
Tabella di traduzione degli indirizzi del programma A

Nr. Pagina	Presente	Accesso	Modificata	Numero di blocco
0	P	A	M	100
1	P	A	M	101
2	P	A	M	102

Nello stesso momento in cui, dalla tabella di occupazione della memoria, determina la pagina libera, nella tabella di traduzione degli indirizzi del processo A, il sistema operativo inizializza il campo *numero di blocco*

Composizione dell'indirizzo

- Un riferimento a una posizione nel programma memoria può essere interpretato come una coppia:
  - numero della pagina
  - dell'offset all'interno della pagina.
- Ad esempio, per calcolare l'indirizzo di memoria di un riferimento alla posizione 250 della pagina 2 del programma A, supponendo che la pagina 2 sia stata collocata nel blocco 102, corrisponde all'indirizzo (102x4096 + 250).



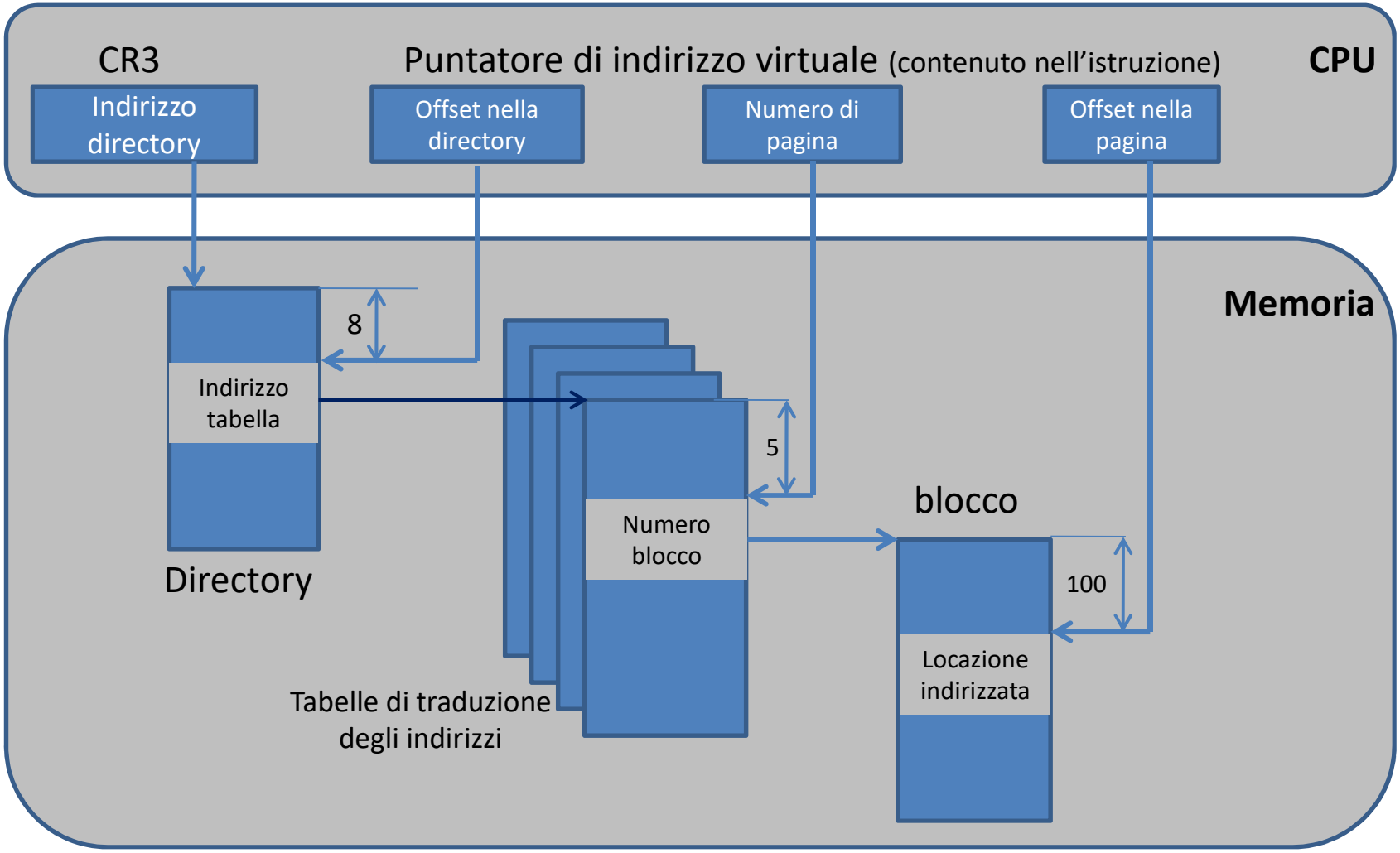
La consultazione delle tabelle per formare l'indirizzo effettivo rallenta l'elaborazione. L'accesso in memoria viene eliminato grazie alla presenza nella CPU di una memoria cache capace di contenere i 32 record della tabella di traduzione degli indirizzi, più recentemente usati.

Composizione dell'indirizzo

- Nei sistemi multitasking è necessario costruire una tabella di traduzione degli indirizzi per ogni processo. Il segmento in cui viene posta ciascuna tabella è determinato dal sistema operativo, sulla base delle indicazioni contenute nella tabella di occupazione della memoria. Pertanto quando l'uso della CPU viene tolto ad un processo per concederlo ad un altro, il sistema operativo deve poter individuare celermente la tabella di traduzione degli indirizzi associata. Per questo scopo c'è un segmento dedicato a contenere la *directory delle tabelle* di traduzione degli indirizzi, e un apposito registro, interno alla CPU, contiene l'indirizzo base della directory in memoria.
- Ciascun record della directory ha una struttura simile a quella dei descrittori della tabella di traduzione degli indirizzi.
- Un puntatore di indirizzo virtuale, cioè ad una posizione del programma, viene considerato come se fosse composto da tre parti:
  - un offset all'elemento di directory che punta alla tabella di traduzione degli indirizzi del processo in corso, corrispondente al PID (Process Identifier)
  - il numero della pagina riferita,
  - lo spiazzamento all'interno della pagina.

Accesso in memoria

Esempio - il processo 8 fa riferimento alla locazione 100 contenuta nella pagina 5



Accesso in memoria

- Il processo attivo è il numero 8, la CPU indirizza l'ottavo record della directory e legge il campo *indirizzo base* per conoscere la locazione in cui è collocata la tabella di traduzione degli indirizzi del processo 8. In questa tabella, la CPU, accedendo al campo *numero di blocco* del descrittore della pagina 5 estrae l'indirizzo base del blocco contenente la pagina riferita; infine, per ottenere l'indirizzo della locazione, somma l'indirizzo base del blocco in memoria con il campo *offset*.

Efficienza della memoria virtuale

I programmi registrati su disco sono suddivisi in **pagine** di dimensione N Byte, e i blocchi in memoria centrale hanno la stessa dimensione di una pagina. Il sistema operativo, nel prelevare un programma dal disco, assegna a ciascuna pagina un blocco, ma non carica tutto il programma, alcune pagine restano sul disco per consentire la presenza in memoria anche di altri programmi.

Il principio della località

La memoria virtuale ha un rendimento elevato se i programmi soddisfano il requisito della *località*. Questa proprietà in genere è rispettata perché l'esecuzione di un programma è composta da tanti cicli di calcolo durante i quali i dati e le istruzioni riferite sono contenute in un'area ristretta dello spazio di memoria, e nella maggior parte del tempo l'esecuzione procede avanzando in zone di memoria contigue. Si ammette cioè che nei programmi i riferimenti a locazioni esterne alla pagina si verificano meno frequentemente di quelli a locazioni interne.

Vantaggi della memoria virtuale

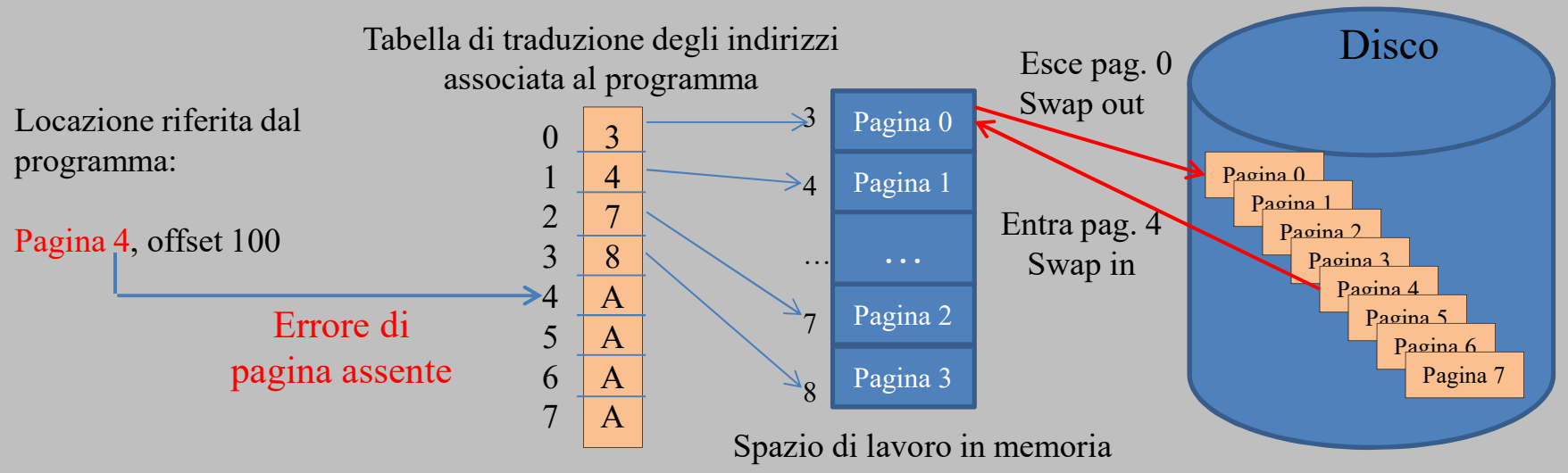
- Il programmatore scrive i propri programmi ritenendo di avere a disposizione uno spazio di memoria illimitato o, almeno, l'intero spazio di indirizzamento virtuale (determinato dalla dimensione dei registri della CPU) anche se la capacità della memoria è di dimensioni minori.
- In memoria possono risiedere ed essere eseguiti più programmi contemporaneamente, anche se nel loro insieme i programmi richiedono più spazio di quello disponibile.

Questo implica che un programma non viene caricato completamente in memoria.

Il sistema operativo si occupa di assegnare lo spazio di memoria al programma e di convertire i riferimenti a indirizzi interni al programma in riferimenti a indirizzi fisici.

Page Fault

- Quando il programma fa riferimento a una pagina non residente, si verifica un'errore di *pagina assente* che il sistema operativo deve risolvere. Se esiste un blocco libero allora vi colloca la pagina prelevata dal disco, altrimenti deve selezionare un blocco da liberare.
- Se la pagina da eliminare è stata modificata allora questa deve essere salvata sul disco al posto della sua copia precedente. Questa operazione di scambio di pagine è detta **swapping**.



Algoritmi di sostituzione

- Quando si verifica un errore di “Pagina Assente”, il sistema operativo deve applicare un algoritmo di sostituzione. Deve, cioè, decidere quale pagina si può eliminare dalla memoria per sostituirla con quella riferita.
- Per tale decisione si tengono in considerazione almeno due indicatori.
  - Se la pagina è stata modificata si deve aggiornare anche la copia su disco, ma siccome il programma non è ancora terminato, la copia originale viene lasciata inalterata e la pagina viene temporaneamente salvata in un’area di swap del disco.
  - Per non degradare le prestazioni del sistema, si deve evitare di eliminare una pagina che verrà riferita nuovamente.

Algoritmi di sostituzione

Errore di pagina assente.

- Quando, durante l'esecuzione di un programma, un'istruzione fa riferimento ad una istruzione contenuta in una pagina non presente in memoria, si verifica un errore di pagina assente. Il sistema operativo deve trasferire, da disco a memoria centrale, la pagina richiesta.
- Se il programma ha usato tutto lo spazio che gli è stato assegnato, il sistema operativo deve individuare un blocco contenente una pagina da eliminare.

Requisiti della pagina da eliminare

- Per liberare un blocco e consentire di caricare in esso la pagina richiesta, si deve scegliere una pagina che non verrà usata più, allo scopo di ridurre al minimo la frequenza degli errori di pagina assente.
- **L'algoritmo di sostituzione migliore è quello che provoca il minor numero di errori di "pagina Assente"**

Algoritmi di sostituzione

Quando si deve eliminare una pagina dalla memoria, per far posto ad una nuova pagina, si rischia di sostituire una pagina che verrà richiesta subito dopo essere stata eliminata, provocando immediatamente un nuovo page fault.

Algoritmo predittivo.

- Il programma dovrebbe dichiarare le future richieste di pagina, in modo che il sistema operativo elimini quella che non verrà più richiesta o, almeno, verrà riferita molto lontano nel tempo.
- Si supponga che il programma abbia avuto 4 blocchi e siano stati occupati con le pagine 0, 1, 2 e 3. Quando il programma richiederà un accesso alla pagina 4, si verifica un errore di pagina assente.
- Se il sistema operativo conoscesse le richieste future di accessi che farà il programma, ad esempio la sequenza di pagine:

  4, 3, 4, 2, 3, 1, 4, 0,

  preferirà eliminare dalla memoria la pagina 0.
- Questo è un algoritmo ideale. Nella pratica è irrealizzabile.

Algoritmi di sostituzione: LRU

Algoritmo LRU (Least Recently Used)

- L'algorimo **LRU** (Least Recently Used) consiste nello scegliere, tra le pagine da eliminare, quella che non è riferita da più tempo. Con molta probabilità tale pagina non verrà più usata.
- L'algoritmo predittivo è ideale, non è realizzabile, ma può essere simulato ammettendo che conoscendo il comportamento passato del programma, in base al principio della località, si possano prevedere i prossimi riferimenti in memoria.
- Cioè, le pagine usate di recente saranno nuovamente usate, con elevata probabilità.
- LRU cerca la pagina che è rimasta inutilizzata più a lungo. Per cercare la pagina da eliminare il sistema operativo può gestire una lista di pagine ordinate in base all'istante di riferimento. In pratica il sistema operativo, per ogni accesso in memoria, sposta in testa alla lista la pagina riferita.
- In alternativa si può affidare all'Hardware il compito di gestire la lista.

Algoritmi di sostituzione: LRU

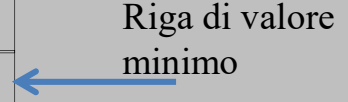
- Se la memoria di lavoro di un processo è suddivisa in N blocchi, la CPU gestisce una matrice di NxN bit. Quando un processo ottiene le sue pagine, tutti i bit della matrice vengono posti a 0.
- Quando avviene l'accesso alla pagina k, la CPU:
  - scrive 1 in tutti i bit della riga k della matrice
  - 0 in tutti i bit della colonna k. La pagina meno recentemente usata è quella che corrisponde alla riga di valore minore.

Dopo la seguente sequenza di accessi

0, 1, 0, 2, 3, 2, 0

La tabella contiene delle righe che vengono interpretate come numeri senza segno:

	0	1	2	3
0	0	1	1	1
1	0	0	0	0
2	0	1	0	1
3	0	1	0	0



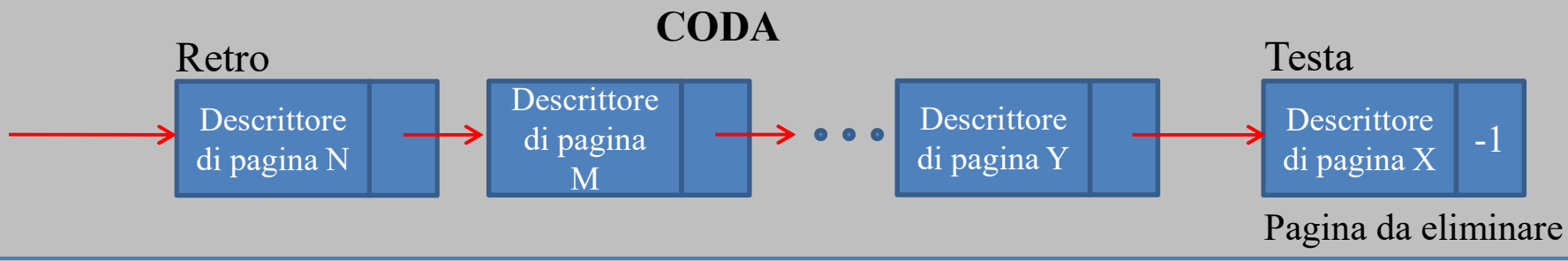
Algoritmi di sostituzione: LRU

- Un'altra tecnica per realizzare l'algoritmo di sostituzione LRU consiste nell'utilizzare alcuni bit nel descrittore di pagina, contenuto nella tabella di traduzione degli indirizzi, che riportano le informazioni sugli accessi alle pagine, Il bit di accesso e il bit di pagina Modificata.
  - il bit di *Accesso* viene posto a 1 quando la pagina viene riferita e azzerato periodicamente dal Sistema Operativo.
  - in corrispondenza dell'azzeramento periodico, il Sistema Operativo incrementa un contatore per tutte le pagine con bit di accesso a 0. In questo modo il conteggio associato ad una pagina indica l'anzianità della pagina.
  - Tra tutte le pagine con bit di accesso uguale a 0, viene sostituita quella pagina con il valore del contatore più alto.
- Inoltre, se si trovano più pagine con lo stesso valore del contatore, si preferisce sostituire la pagina che non è stata modificata, allo scopo di evitare l'operazione di swap out

Algoritmi di sostituzione: FIFO

Algoritmo FIFO.

- Il sistema operativo, mentre inserisce le pagine in memoria, mantiene una lista ordinata cronologicamente dei descrittori di tali pagine. In testa si trova il descrittore della pagina più vecchia, in coda si trova il descrittore dell'ultima pagina caricata. Quando si verifica l'errore di "pagina assente" il sistema operativo elimina la pagina di testa della lista ed inserisce in coda il descrittore della nuova pagina caricata.



Aggiornamento della coda dopo un errore di pagina assente

2) Il numero di riga del nuovo descrittore diventa il nuovo valore di **Retro**

Pagina da caricare

1) Il puntatore viene posto uguale a **Retro**

3) Il penultimo descrittore diventa l'elemento di Testa della coda

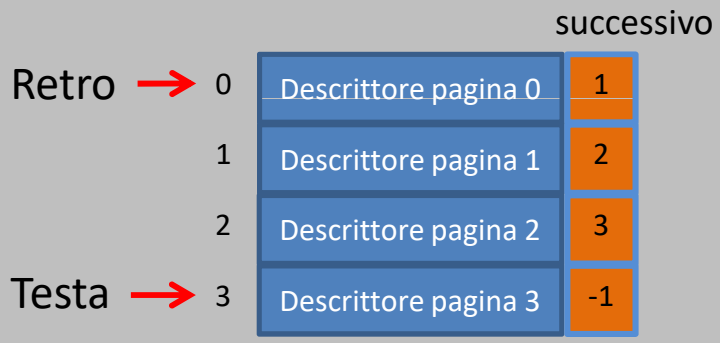
Gestione di una coda (approfondimento)

- Una coda in memoria può essere realizzata con una lista. In una lista, ogni elemento possiede un puntatore al successivo.
- Gli elementi di una lista possono essere
  - Creati durante l'esecuzione del programma,
  - Memorizzati in una tabella.

Per gestire la coda, servono due puntatori: Uno indica la posizione dell'ultimo elemento, l'altro indica la posizione del primo elemento.

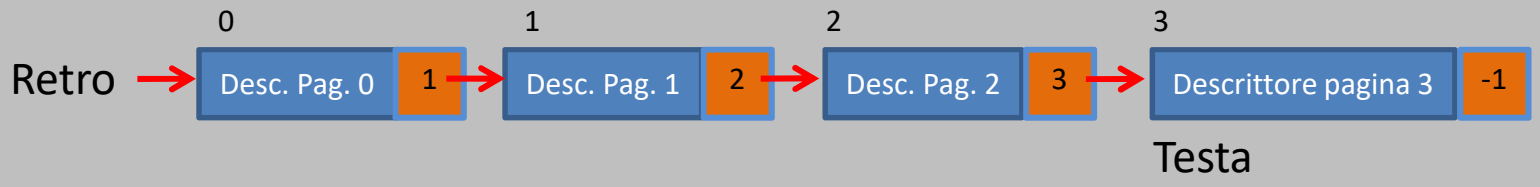
Retro=0

Testa=3



CODA realizzata in una tabella

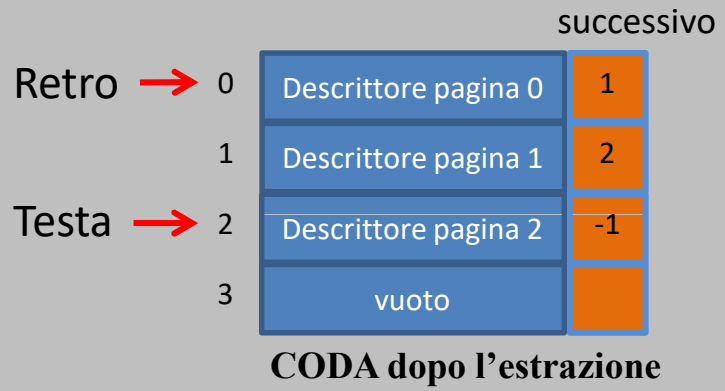
Gli elementi della tabella rappresentati in una lista lineare:



Gestione di una coda (approfondimento)

- Le operazioni su una coda sono:
  - Scansione della coda,
  - Estrazione dell'elemento di testa,
  - Inserimento di un elemento in coda.

Estrazione dell'elemento di testa:
Nell'esempio l'elemento di testa occupa la riga 3 e, nella lista è preceduto (*individuato dopo una scansione*) dall'elemento in posizione 2.
Per eliminarlo, basta porre **Testa=2** e scrivere -1 nel campo **successivo** dell'elemento in posizione 2



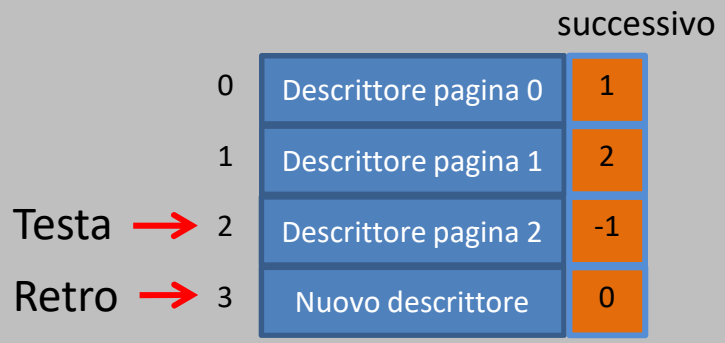
CODA dopo l'estrazione

Inserimento in coda:
Dopo aver completato il campo "Descrittore di pagina" di una riga vuota, il nuovo elemento viene inserito in coda:
Nel campo **successivo** del nuovo elemento si scrive il valore di **Retro**:

successivo[3] = Retro

e al puntatore "Retro" si assegna il numero di riga del nuovo elemento:

Retro = 3



CODA dopo l'inserimento

Algoritmi di sostituzione

Algoritmo Second Chance.

- L'algoritmo FIFO rimpiazza la pagina più vecchia in memoria.
- L'algoritmo Second Chance evita di eliminare la pagina più vecchia se il bit R (Riferita) indica che questa è stata usata.
- Il sistema operativo usa una coda per mantenere l'ordine cronologico di caricamento in memoria dei descrittori delle pagine.

Al verificarsi di un page fault

- Se la pagina di testa (la più vecchia) possiede il bit R=0 (non riferita) la pagina viene eliminata (come in FIFO), se il descrittore della pagina in testa alla lista possiede R=1, la pagina non viene sostituita, però il bit R viene posto a 0 e il descrittore della pagina viene inserito in coda alla lista, come se la pagina fosse appena entrata. Si offre una seconda possibilità alla pagina.
- il bit R viene periodicamente azzerato.

Algoritmi di sostituzione: NRU

Algoritmo NRU (Not Recently Used).

- Il sistema operativo associa, ad ogni pagina, due bit di stato:
  - il bit R (Riferita)
  - il bit M (Modificata).
- Ad ogni accesso ad una pagina, il bit R viene posto a 1. Periodicamente il sistema operativo rimette a 0 il bit R di tutte le pagine.

Il Sistema Operativo raggruppa le pagine nelle 4 classi che si possono formare con le possibili combinazioni dei valori dei bit R ed M:

- Classe 0: pagina non riferita (R=0) e non modificata (M=0)
- Classe 1: pagina non riferita (R=0), modificata (M=1)
- Classe 2: pagina riferita (R=1), non modificata (M=0)
- Classe 3: pagina riferita (R=1), modificata (M=1)

L'algoritmo NRU sostituisce una qualsiasi pagina della classe di ordine inferiore, non vuota.

Algoritmi di sostituzione

Algoritmo LFU (Least Frequently Used)

- Least Frequently Used associa un *contatore* ad ogni pagina.
- Periodicamente il sistema operativo addiziona il bit R (0 o 1) al contatore, ed azzera R. Quindi il contatore indica il numero di riferimenti di ogni pagina. Quando si verifica un errore di “pagina assente”, il sistema operativo elimina la pagina con il minimo conteggio.
- Se una pagina è stata molto riferita, il suo contatore potrebbe essere massimo, e tale resterebbe anche se la pagina non verrà più usata. Questa pagina non verrebbe mai sostituita.

Algoritmi di sostituzione

Algoritmo dell'invecchiamento.

- Per tener conto, quindi, oltre che dell'utilizzo della pagina, anche dell'istante del riferimento alla pagina, anziché contare i riferimenti, il contatore viene usato a scorrimento, il bit R viene inserito a sinistra, facendo scorrere a destra gli altri bit. Poi il bit R viene azzerato. Quando si verifica un page fault viene rimossa la pagina col contatore minimo.

Contatore	Pagina	R	Contatore
0000 0000	0	1 →	1000 0000
0000 0000	1	0 →	0000 0000
0000 0000	2	0 →	0000 0000
0000 0000	3	1 →	1000 0000
0000 0000	4	0 →	0000 0000
0000 0000	5	0 →	0000 0000

Aggiornamento
t = T

R	Contatore
1 →	1100 0000
1 →	1000 0000
0 →	0000 0000
0 →	0100 0000
1 →	1000 0000
1 →	1000 0000

Aggiornamento
t = 2T

Stima delle prestazioni di una memoria virtuale

Influenza della memoria virtuale sul tempo di elaborazione

- Una memoria virtuale non dovrebbe rallentare eccessivamente il sistema di elaborazione impegnando i processori a trasferire pagine dalla memoria secondaria alla memoria centrale per risolvere i page fault.

Influenza della memoria virtuale sullo spazio di memoria utilizzato

- Per ogni programma caricato in memoria viene riservato uno spazio di memoria per mantenere la tabella di traduzione degli indirizzi.
- La paginazione introduce uno spreco di memoria solo nell'ultima pagina di un programma.

Criteri per calcolare la dimensione ottimale dello spazio di lavoro da assegnare ai programmi, cioè quante pagine conviene caricare in memoria?

- Si devono bilanciare gli effetti di queste due cause:
  - Pagine di troppo lunghe introducono tempi di trasferimento alti, durante le fasi di swapping e riducono lo spazio per le pagine di altri processi.
  - Pagine troppo corte introducono frequenti eccezioni per "pagina assente" e il sistema trascorrerebbe troppo tempo a trasferire pagine.

Stima delle prestazioni di una memoria virtuale

- Ad ogni programma è associato uno spazio virtuale complessivo, indicato con "SpV“.
- Un programma viene diviso in pagine di lunghezza "L_p" byte, quindi la tabella di traduzione degli indirizzi deve avere una lunghezza sufficiente a contenere SpV/L_p record.
- Questo spazio non sarebbe usato in un sistema a memoria fisica limitata, quindi è da considerare come un *costo* conseguente all'uso della memoria virtuale. Analogamente, poiché non è detto che un programma copra completamente le pagine che gli vengono assegnate, si può ritenere che nell'ultima pagina vi sia uno spazio vuoto, che in media vale mezza pagina: $L_p/2$. Pertanto il *costo medio* della memoria virtuale è la somma dei due spazi di memoria che essa richiede:

$$\text{Spazio virtuale sprecato} = L_p/2 + SpV/L_p$$

In cui si suppone che un elemento della tabella di traduzione degli indirizzi sia grande 1 byte.

Lunghezza ottimale di una pagina

- Questo costo è espresso con una funzione che dipende da due variabili.
- Per un dato processo SpV ha un valore costante. Per determinare il valore di L_p che rende minimo il costo, si calcoli la derivata della funzione rispetto a L_p e la si ponga uguale a 0:
- d/dL_p(Spazio virtuale sprecato) = ½ $-SpV/(L_p)^2 = 0$
- Lo spazio sprecato è minimo quando

$$L_p = \sqrt{(2SpV)}.$$