



Corso di Laurea Triennale in Informatica
Università Degli Studi della Basilicata

Sistemi Operativi

Docente:
ugo.erra@unibas.it

9° Lezione

Memoria Centrale – II° parte

Sommario della lezione



- Paginazione
 - ▣ Metodo di base
- Architettura di paginazione
 - ▣ Protezione
 - ▣ Pagine condivise
- Struttura della tabella delle pagine
- Segmentazione
 - ▣ Metodo di base
- Architettura di segmentazione
- Il caso del Pentium

Paginazione



- L'allocazione della memoria presenta diversi problemi per via della frammentazione interna ed esterna
- Il problema nasce dall'allocazione contigua della memoria che necessita di trovare buchi in cui allocare il processo
- La **paginazione** è un metodo che permette di allocare ai processi uno spazio di memoria non necessariamente contiguo

Metodo di base

- La memoria fisica è divisa in blocchi di dimensione fissa chiamati **frame** o **pagine fisiche**
- La memoria logica utilizzata dal processo è divisa in blocchi della stessa dimensione chiamati **pagine**
- Il processo è sempre allocato all'interno della memoria logica in uno spazio contiguo
- Per eseguire un processo di n pagine si determinano n frame liberi sufficienti a contenere le pagine del processo
- I frame non devono essere necessariamente contigui

Schema di traduzione degli indirizzi

- L'indirizzo generato dalla CPU è divisa in due parti:
 - ▣ **Numero di pagina (p):** è utilizzato per accedere alla tabella delle pagine che contiene l'indirizzo di base del frame o della pagina fisica
 - ▣ **Scostamento (*offset*) di pagina:** è combinato con l'indirizzo di pagina fisica per ottenere l'indirizzo effettivo della memoria fisica

Numero di pagina	scostamento
p	d

Architettura di paginazione

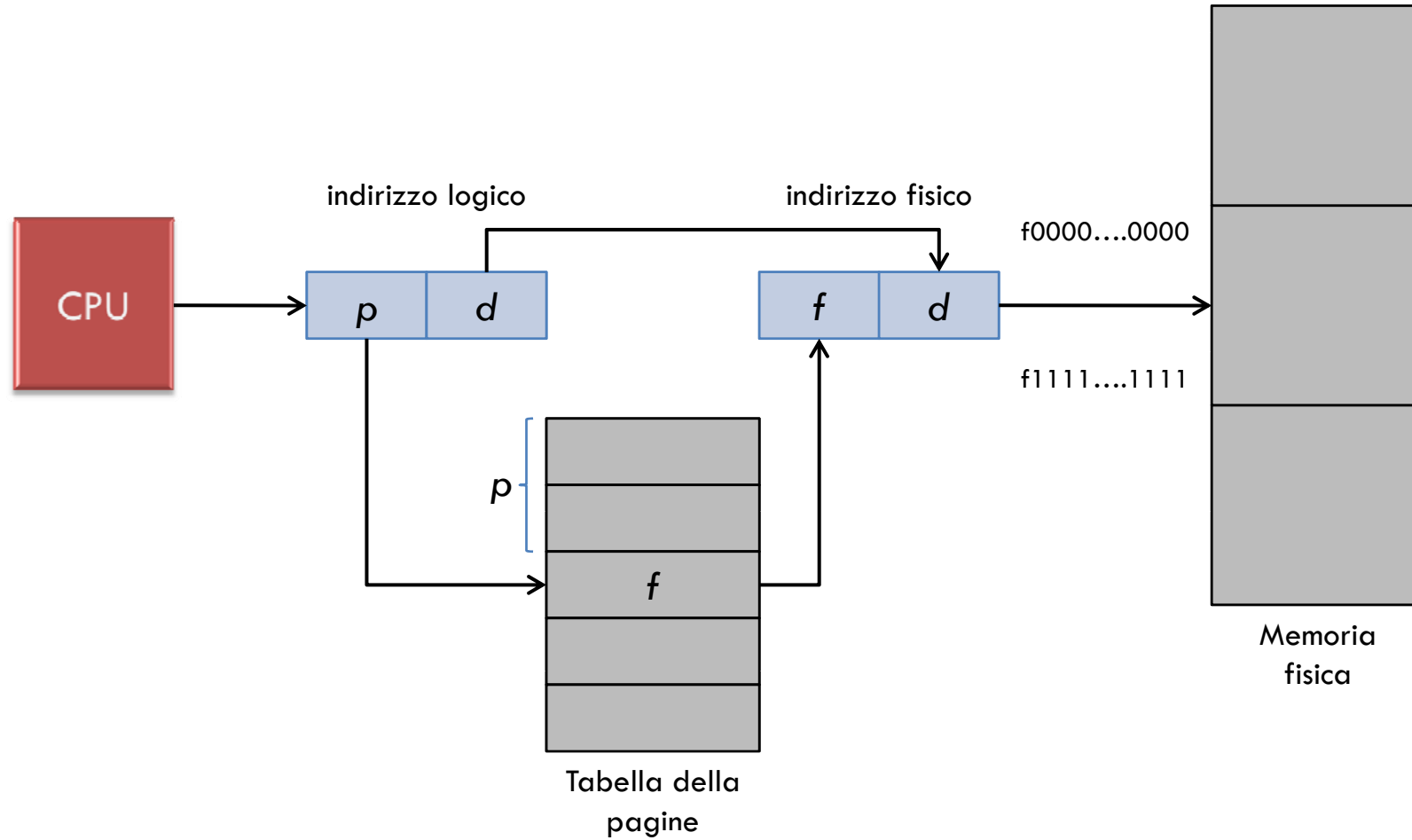
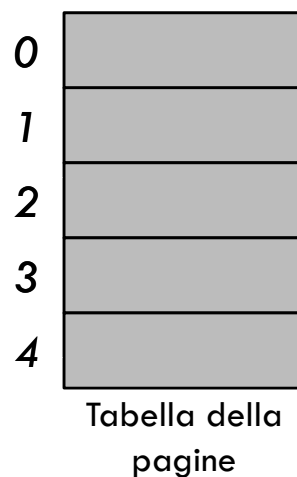
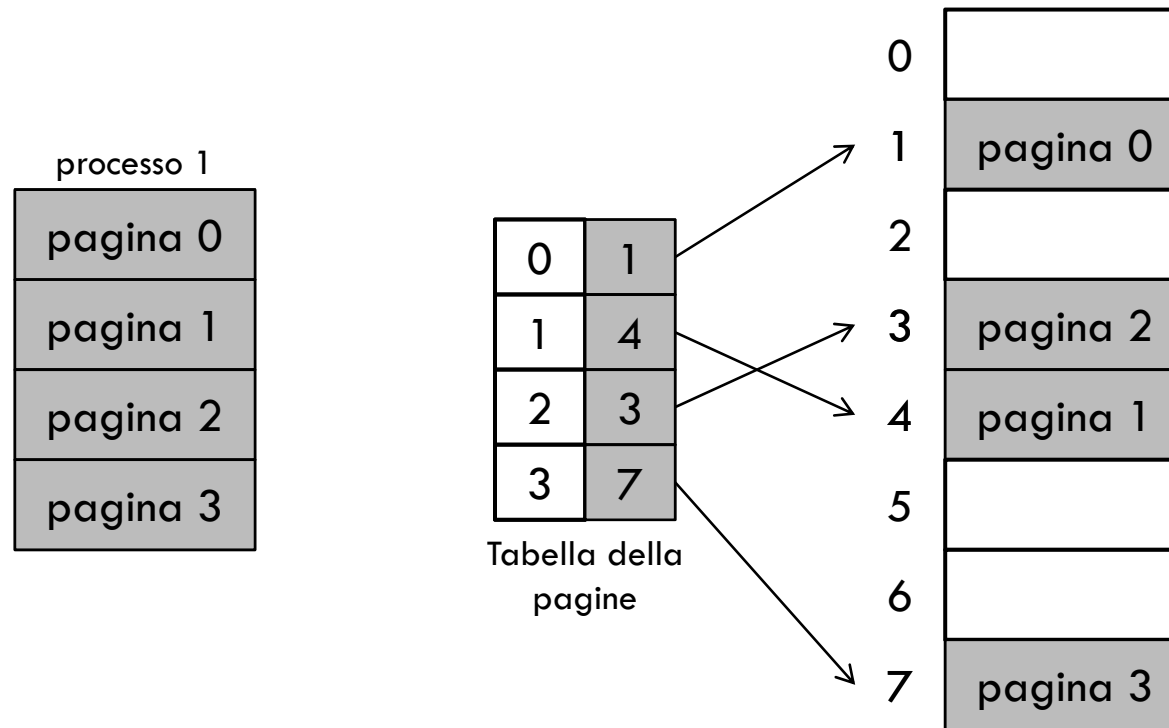


Tabella delle pagine

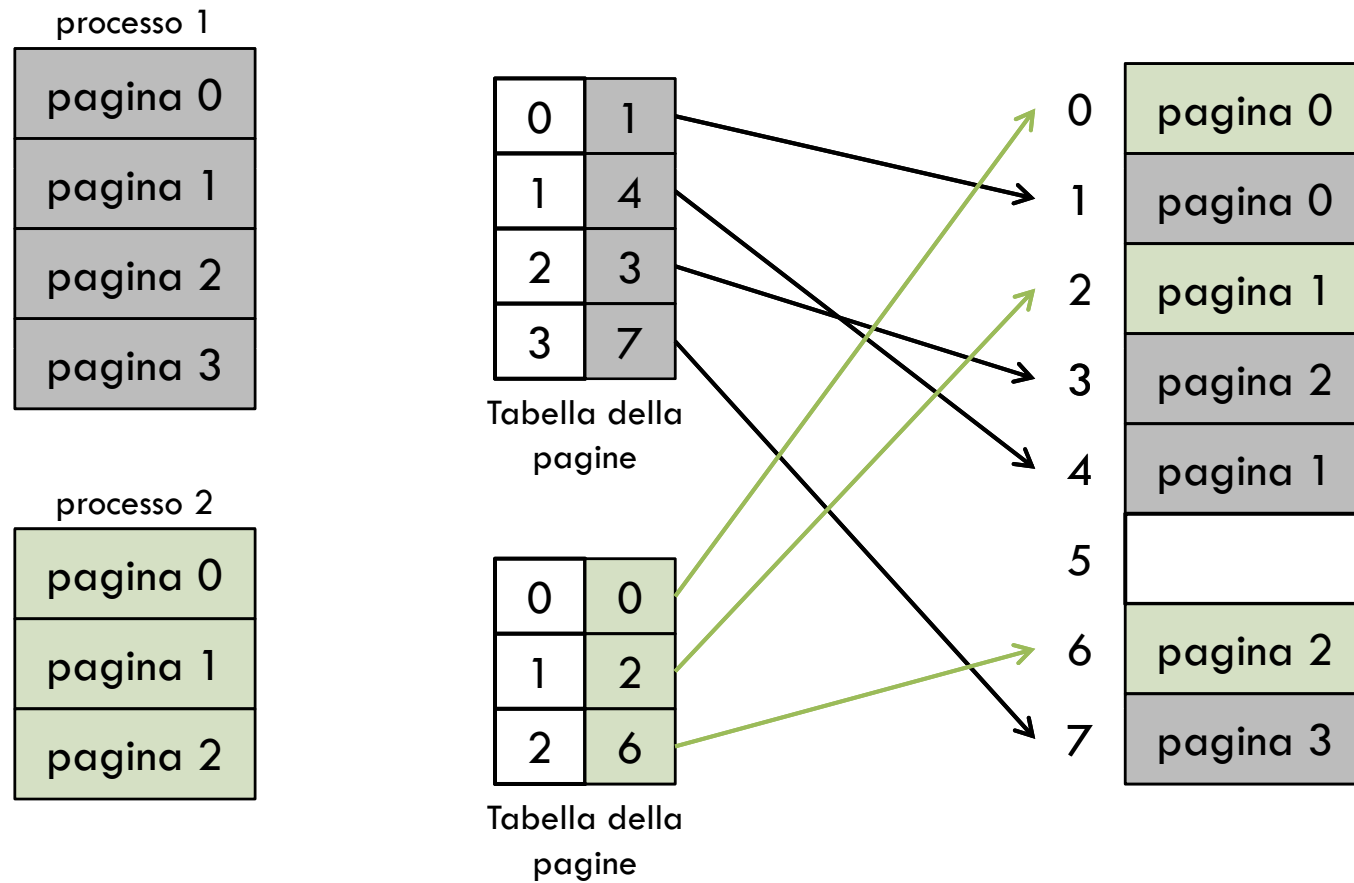
- Ad ogni processo è associata una **tabella delle pagine** (o **dei frame**) (PT) contenente i numeri dei frame usati per memorizzare le pagine del processo
 - ▣ In effetti la tabella delle pagine è un array lineare
- Per determinare il numero del frame associato ad una pagina è necessario un solo accesso



Es. di paginazione di memoria logica e memoria fisica



Es. di paginazione di memoria logica e memoria fisica



Dimensione delle pagine

- La dimensione della pagine è definita dall'architettura del sistema ed è in genere una potenza di 2
 - ▣ La dimensione di una pagina può variare dai 512 byte ai 16MB
- La scelta di una potenza di due semplifica notevolmente la traduzione di un indirizzo logico in un indirizzo fisico
- Se lo spazio logico di indirizzamento è 2^m allora abbiamo:
 - ▣ $m-n$ bit per indicizzare le pagine
 - ▣ n bit per la grandezza di ogni pagina ovvero 2^n

Numero di pagina	scostamento
p	d
$m-n$ bit	n bit

Esempio

- Supponiamo di avere le seguenti condizioni
 - ▣ Dimensione della pagine 4 byte
 - ▣ Dimensione totale della memoria fisica 32 byte
- L'indirizzo logico 3 corrisponde all'indirizzo fisico $(5 \times 4) + 3 = 23$

0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k
11	l
12	m
13	n
14	o
15	p

0	5
1	6
2	1
3	2

Tabella della pagine

0	
4	i j k l
8	m n o p
12	
16	
20	a b c d
24	e f g h
32	

Allocazione dei frame liberi

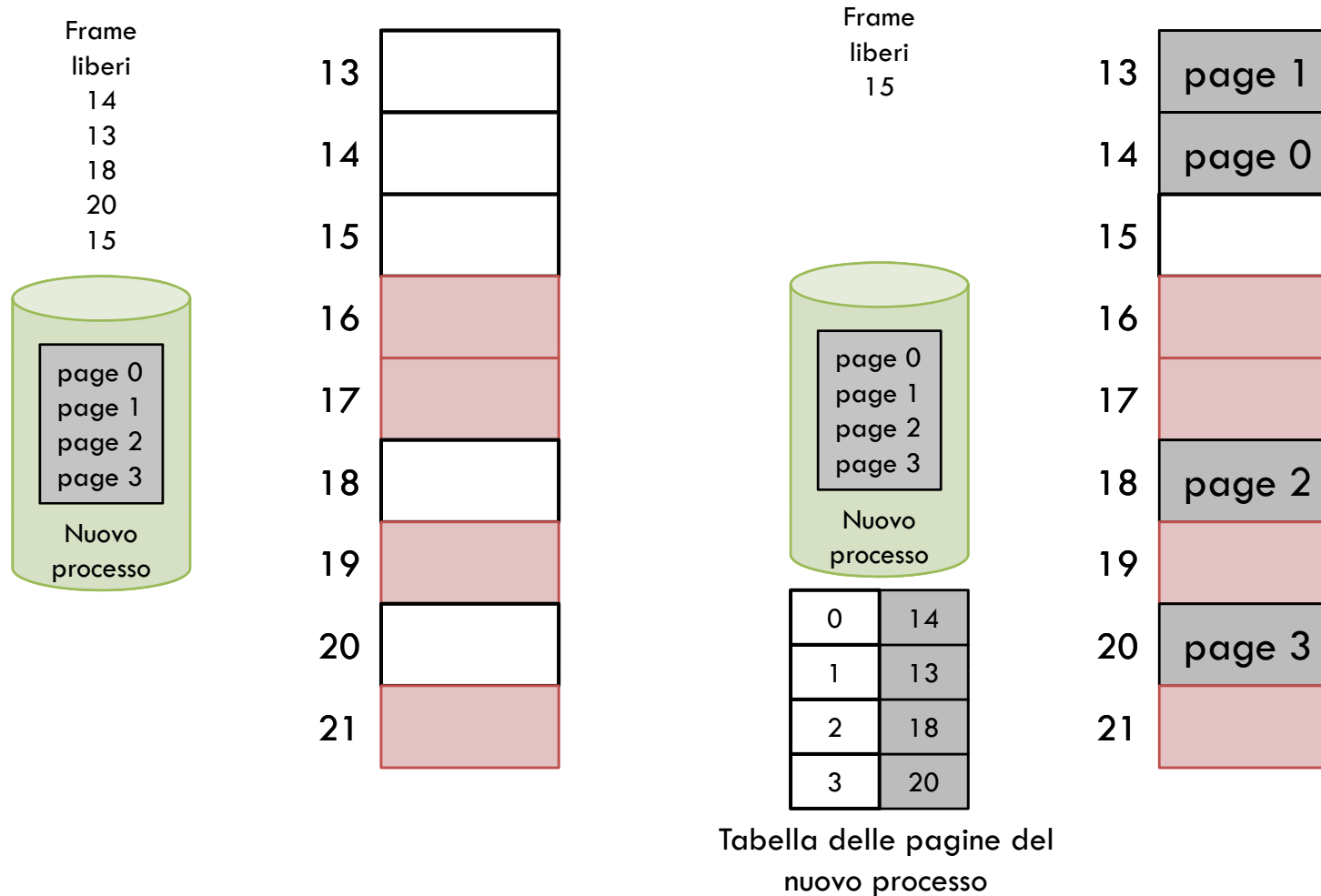


Tabella dei frame

- La paginazione permette di separare nettamente lo spazio di indirizzamento logico da quello fisico.
- Anche se il programma è “sparpagliato” nella memoria fisica il programma vede la memoria come uno spazio di indirizzamento contiguo che incomincia sempre all’indirizzo logico 0
- L’architettura di traduzione degli indirizzi permette questa trasformazione non visibile agli utenti e controllata solo dal sistema operativo

Paginazione: Vantaggi - 1



- La paginazione implementa automaticamente una forma di protezione dello spazio di indirizzamento.
- Un programma può indirizzare solo i frame contenuti nella sua tabella delle pagine
- Quei frame contengono le pagine che appartengono al programma stesso

Paginazione: Vantaggi - 2

- Con la paginazione è possibile evitare la frammentazione esterna
 - ▣ Ogni frame libero si può assegnare ad un processo che ne abbia bisogno
- L'ultimo frame assegnato potrebbe non essere completamente occupato causando quindi frammentazione interna
 - ▣ Il caso peggiore si verifica quando un processo è un multiplo della dimensione della pagina + 1 byte. In questo caso una pagina è completamente sprecata
- La dimensione delle pagine è aumentata col tempo, attualmente si usano pagine di 4096, 8192, 16384 byte, fino a 4 Mbyte
 - ▣ Pagine più grandi producono maggiore frammentazione interna, ma permettono di avere tabelle delle pagine più piccole (meno occupazione di memoria)

Implementazione della tabella delle pagine

- Il Sistema Operativo mantiene anche una struttura dati globale chiamata **tabella delle pagine** per tenere traccia dell'allocazione della memoria fisica
- Per ogni frame c'è un elemento all'interno della frame table indicante
 - ▣ Se il frame è libero
 - ▣ Se è occupato a quale processo è assegnato
- In fase di contex switch il Sistema Operativo attiva la tabella delle pagine per il processo a cui è stata assegnata la CPU
- Il tempo per la fase di contex switch con l'introduzione della paginazione può quindi aumentare in maniera considerevole

Architettura di paginazione - 1

- Poiché ogni indirizzo deve passare attraverso il sistema di paginazione il meccanismo di traduzione degli indirizzi deve essere molto efficiente
 - ▣ Anche tutti gli indirizzi generati dall'applicazione per accedere a dei dati devono essere tradotti
- Il collo di bottiglia potrebbe essere l'accesso alla tabella della pagine (PT)
- Ogni indirizzo logico deve essere tradotto in un indirizzo fisico con un accesso alla PT

Architettura di paginazione - 2

- Nel caso in cui il numero di pagine per ogni processo è piccolo, la PT può essere memorizzata all'interno di appositi registri della CPU
 - ▣ Ad esempio, il PDP-11 usava 8 registri per la PT. Gli indirizzi fisici erano a 16 bit, per cui la memoria fisica aveva una dimensione di 64 Kbyte (divisi in 8 frame da 8 Kbyte)
- Negli attuali sistemi la tabella della pagine può avere un numero di elementi troppo grande per essere contenuta all'interno dei registri della CPU

Supporto hardware



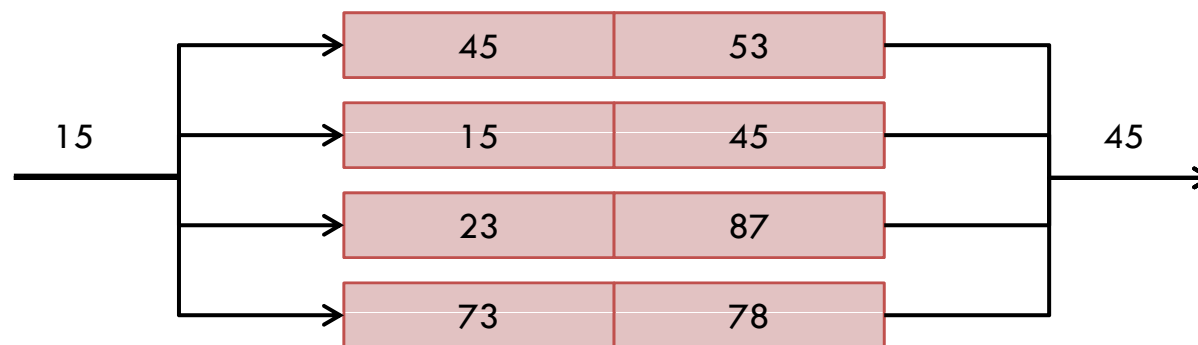
- Due sono le possibili soluzioni:
 - ▣ **Registro di base della tabella delle pagine** (*page-table base register, PTBR*)
 - ▣ **Memoria associativa** (*Translation look-aside buffer, TLB*)

Registro di base della tabella delle pagine

- In questa soluzione un registro di base punta alla tabella delle pagine in memoria
- In fase di context switch è necessario solo aggiornare questo registro riducendo il cambio di contesto
- Poiché la dimensione delle pagine può crescere notevolmente è necessario mantenerla in memoria occupando quindi spazio
- Il problema di questa soluzione è che per accedere alla RAM è necessario prima accedere alla RAM per recuperare l'indirizzo fisico e poi accedere effettivamente alla RAM per recuperare il dato
- In definitiva il numero di accessi alla memoria fisica aumenta di un fattore 2

Memoria associativa - 1

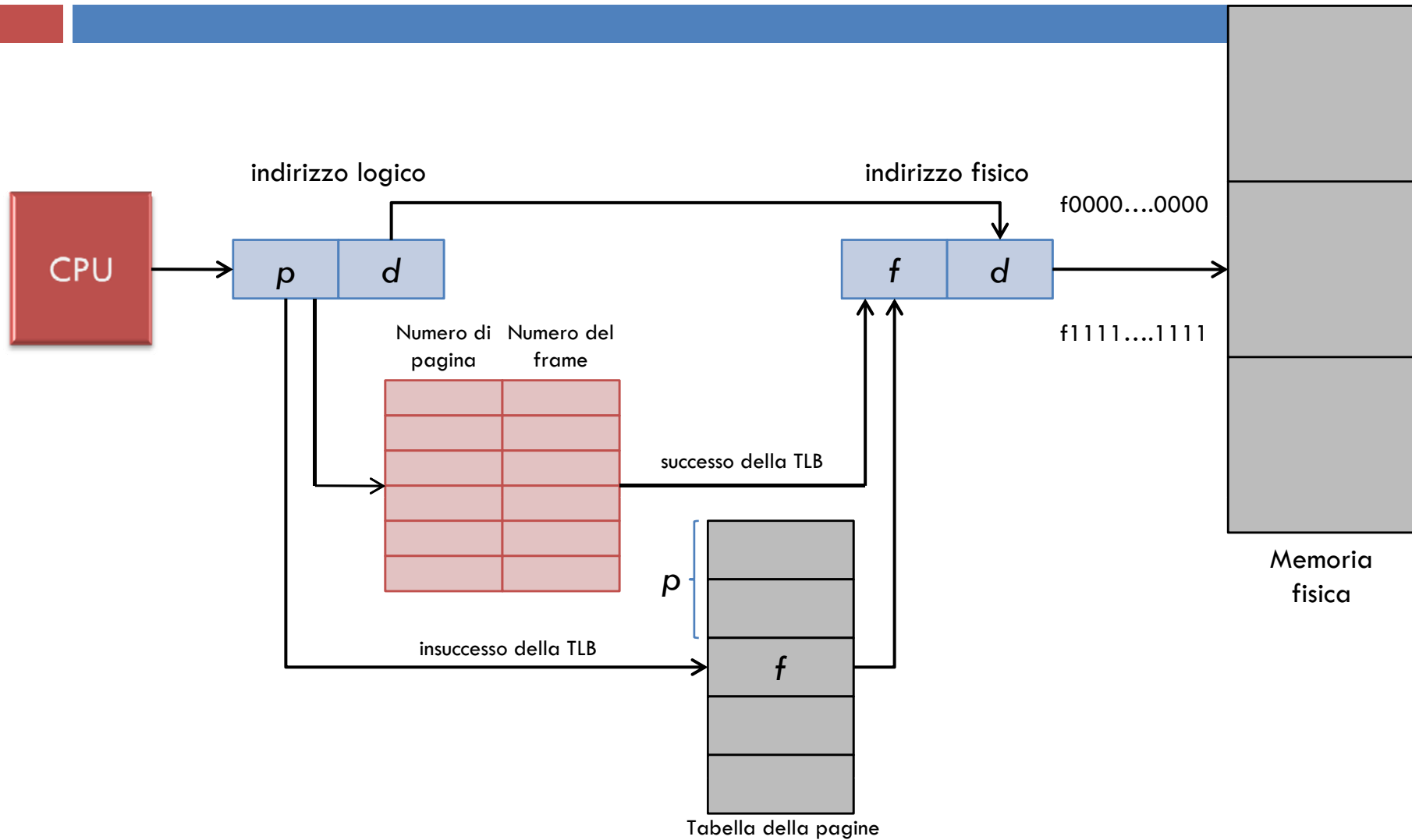
- Una **memoria associativa** (*translation look-aside buffer, TLB*) è una cache particolarmente veloce
- All'interno della memoria associativa abbiamo una coppia del tipo (*chiave, valore*)
- Per ogni chiave passata in input alla memoria associativa questa viene confrontata contemporaneamente con tutte le chiavi presenti per restituire il valore in output
- Il dispositivo è molto veloce e costoso, e le dimensioni di una TLB sono ridotte (dai 64 ai 1024 elementi)



Memoria associativa - 2

- Nella memoria associativa viene caricata una porzione della tabella della pagine (anche tutta se è possibile)
- Ad ogni chiave corrisponde un numero di pagina a cui corrisponde un frame
- Per ogni indirizzo logico, il numero di frame viene dato in input alla memoria associativa per ottenere il relativo frame
- Se la ricerca ha successo (*hit*) si ha solo una penalizzazione del 10% del tempo di accesso alla memoria senza paginazione
- Se la ricerca fallisce (*TLB miss*) allora occorre usare la tabella delle pagine nella RAM

Architettura di paginazione con TLB



Hit ratio

- Definiamo il **tasso di successo** (*hit ratio*) la percentuale di volte che la ricerca nella TLB ha successo
 - ▣ Con un hit ratio dell'80% il numero della pagine è trovata nella TLB l'80% delle volte
- Supponiamo che un accesso in memoria impiega 100 nanosecondi
- Se la ricerca nella TLB impiega 20 nanosecondi allora un accesso in memoria impiega:
$$20 + 100 = 120 \text{ nanosecondi}$$
- Se il numero di pagina non è contenuto nella TLB allora è necessario accedere alla memoria fisica per determinarlo, quindi un accesso in memoria impiega:
$$20 + 100 + 100 = 220 \text{ nanosecondi}$$

Tempo effettivo d'accesso alla memoria

- Nel caso in cui la percentuali di hit ratio è dell'80% e di insuccessi è del 20% il tempo medio di accesso alla memoria fisica è:

$$0.80 \times 120 + 0.20 \times 220 = 140 \text{ nanosecondi}$$

le prestazioni degradano del 40%

- Nel caso in cui la percentuali di hit ratio è dell'95% e di insuccessi è del 2% il tempo medio di accesso alla memoria fisica è:

$$0.95 \times 120 + 0.02 \times 220 = 122 \text{ nanosecondi}$$

le prestazioni degradano del 22%

Gestione della TLB - 1

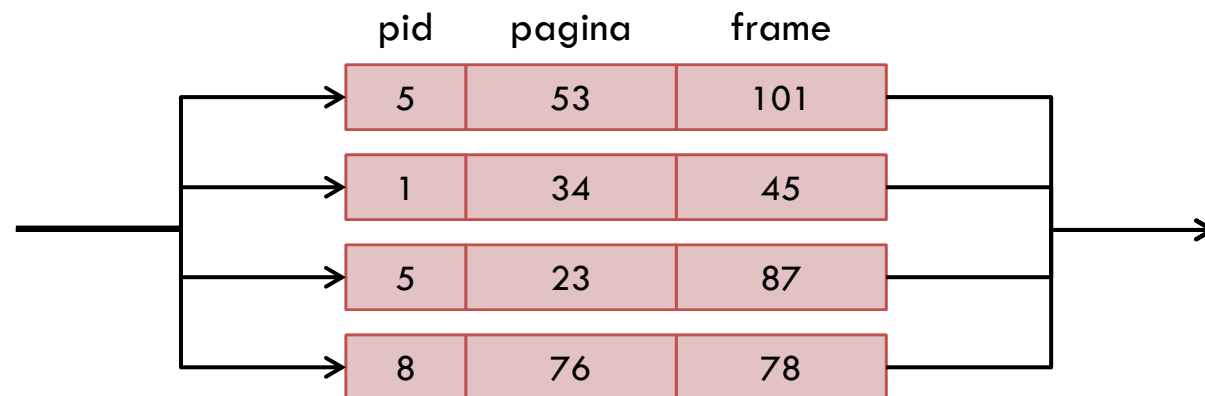
- Nella memoria associativa sono sempre presenti delle coppie (pagina, frame)
- Cosa accade quando si verifica un miss ed una pagina non è presente nella TLB?
 - ▣ Se la pagina non è presente viene recuperata in memoria ed inserita nella TLB per velocizzare gli accessi successivi
- Ma se la TLB è piena con quale elemento viene rimpiazzato?
 - ▣ L'elemento usato meno di recente
 - ▣ Un elemento scelto a caso

Gestione della TLB - 2

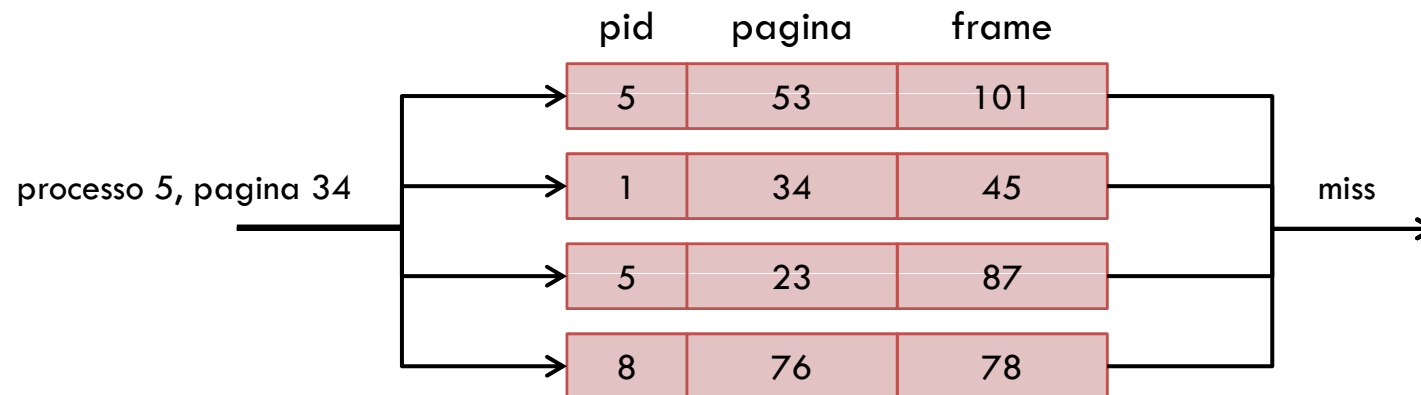
- Cosa accade durante un contex switch?
 - ▣ Tutte le entry all'interno della TLB dovrebbero essere invalidate perché si riferivano al processo in esecuzione precedentemente
- Durante il contex switch si dovrebbe svuotare la TLB, questo comporterebbe un enorme numero di miss
- Durante il contex switch il Sistema Operativo potrebbe caricare la TLB del processo salvata precedentemente ma questo renderebbe il contex switch molto più pesante

TLB con id del processo

- Una soluzione a questo problema consiste nell'avere una TLB con un elemento per ogni entry del tipo (*id*, *pagina*, *frame*)
- In ogni istante la TLB può contenere elementi che appartengono a diversi processi
- Quando viene cercato un elemento, la TLB verifica che l'identificativo del processo corrisponda a quello presente nella TLB altrimenti genera un miss



Esempio di TLB con id del processo



TLB con id del processo

- Il vantaggio nell'uso di una TLB con l'identificativo del processo sta nell'evitare di svuotare la TLB quando un nuovo processo acquisisce l'uso della CPU
- Se si verifica un miss bisogna decidere chi “scartare” per far posto al nuovo elemento
- Alcune entry all'interno della TLB potrebbero anche essere bloccate in modo da non doverle mai sostituire
 - ▣ Ad esempio quelle che appartengono allo spazio di indirizzi del SO stesso

Protezione

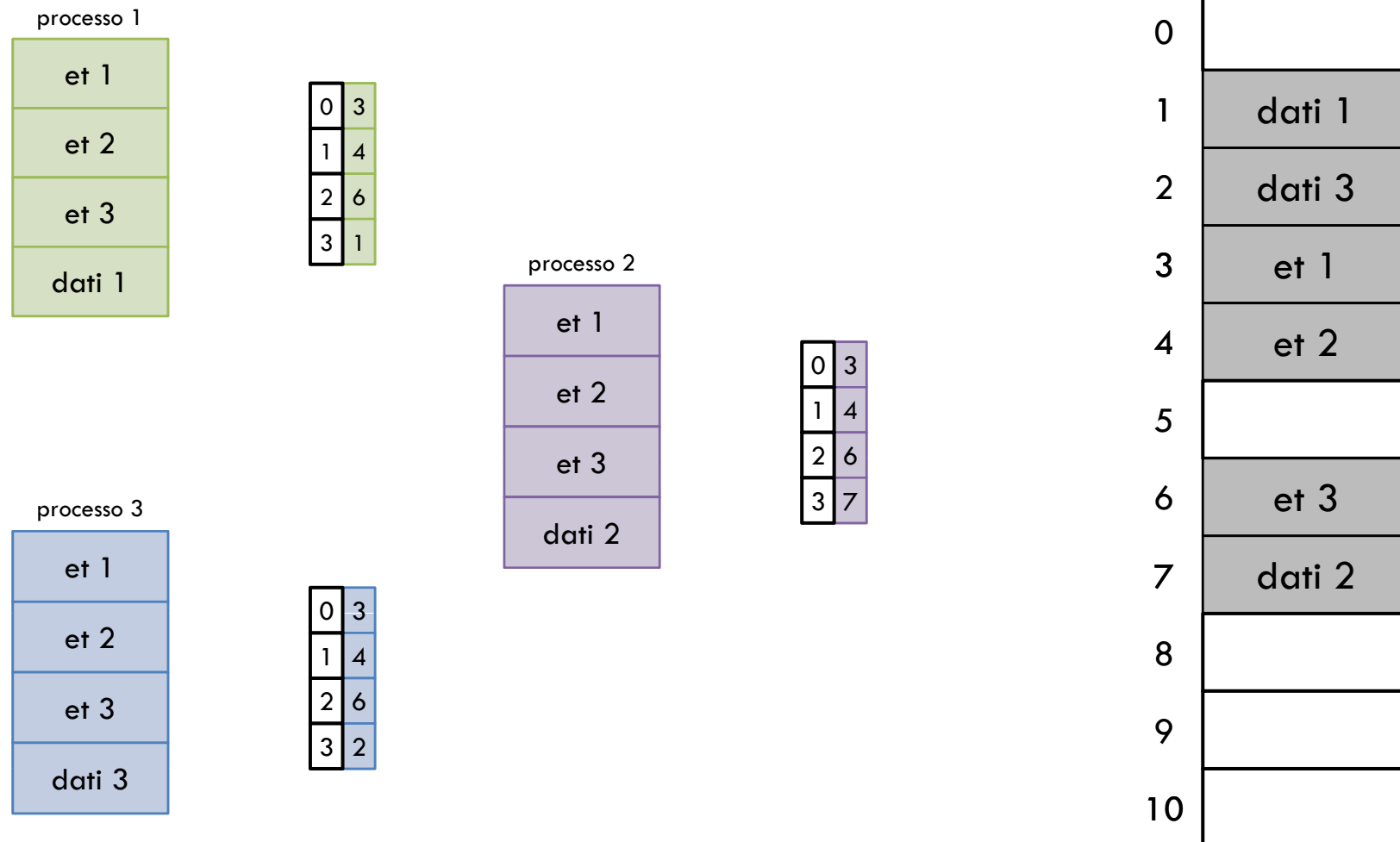


- Le pagine possono essere protette da opportuni bit associati ad ogni entry della tabella delle pagine, per proteggerle in lettura, scrittura, esecuzione
- Un caso particolare è il **bit di validità della pagina**, che indica se la pagina è effettivamente contenuta nel frame corrispondente
- L'uso di questo bit è di fondamentale importanza nella gestione della memoria virtuale

Pagine condivise

- Due processi potrebbero eseguire lo stesso codice
 - ▣ Ad esempio una libreria oppure un editor di testi in un ambiente multiutente
- La paginazione permette di condividere facilmente codice tra diversi processi (**codice puro o rientrante**)
- Questa opzione è possibile perché il codice rientrante non cambia durante l'esecuzione del processo
- Ad esempio, una pagina condivisa può essere usata per contenere il codice di una libreria dinamica usata contemporaneamente da più processi

Esempio di codice condiviso



Struttura della tabella delle pagine

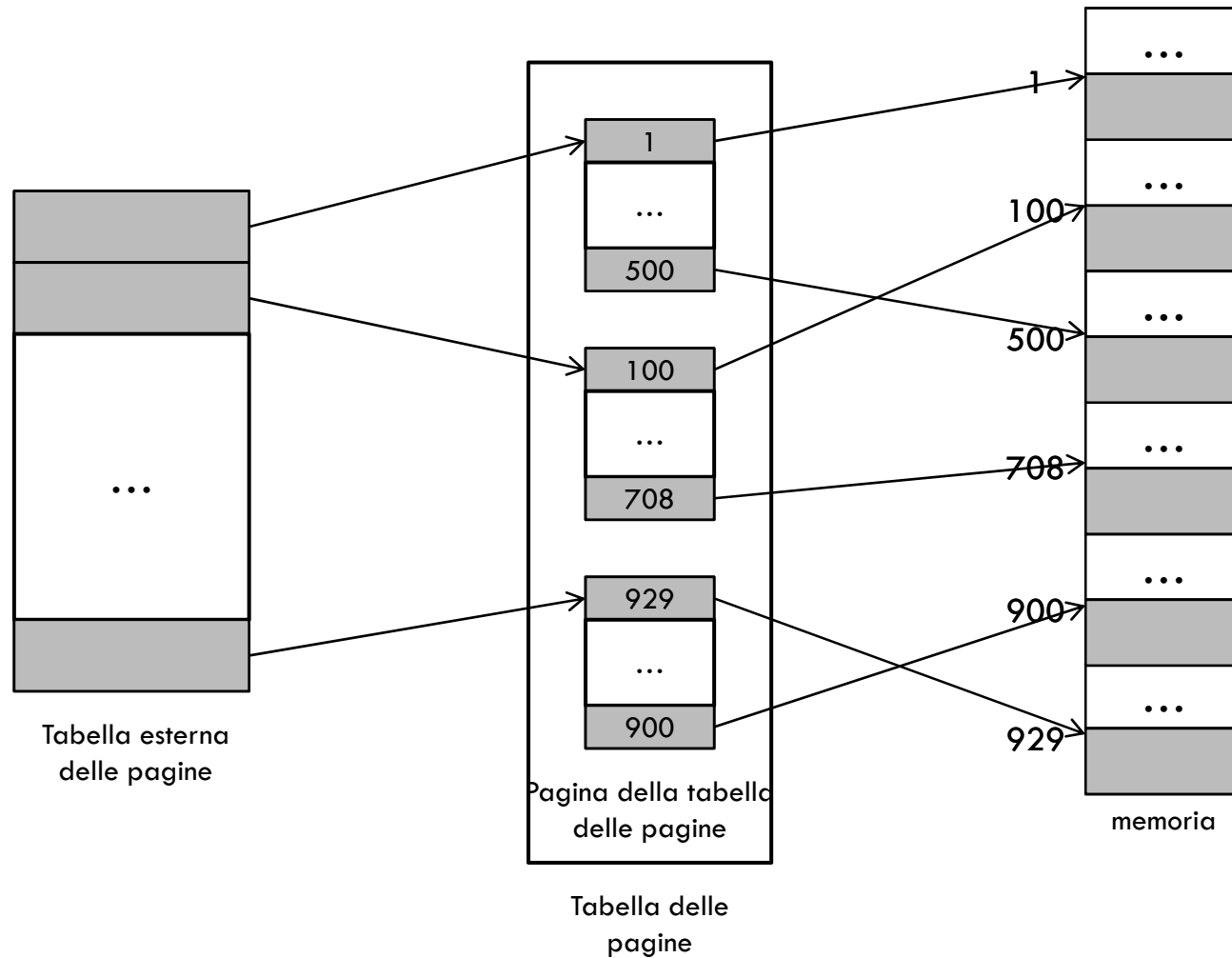
- I microprocessori moderni supportano un schema di indirizzamento di 32 o 64 bit portando lo spazio di indirizzamento logico a 2^{32} byte e 2^{64} byte
- Con uno spazio di indirizzamento logico di 2^{32} ed una dimensione delle pagine di 4KB (2^{12}) la tabella delle pagine può avere una dimensione di $2^{32}/2^{12} \approx 1$ milione di elementi
- Supponendo che ogni elemento occupi 4 byte la tabella delle pagine può arrivare a 4MB
- Alcune soluzioni possibili:
 - ▣ Paginazione gerarchica
 - ▣ Tabella delle pagine di tipo hash
 - ▣ Tabella delle pagine invertite

Paginazione gerarchica



- L'idea consiste nell'utilizzare una paginazione a due livelli in cui la tabella stessa è paginata
- Si utilizza una **tabella della pagine interna** ed una **tabella delle pagine esterna**

Paginazione a due livelli



Esempio - 1

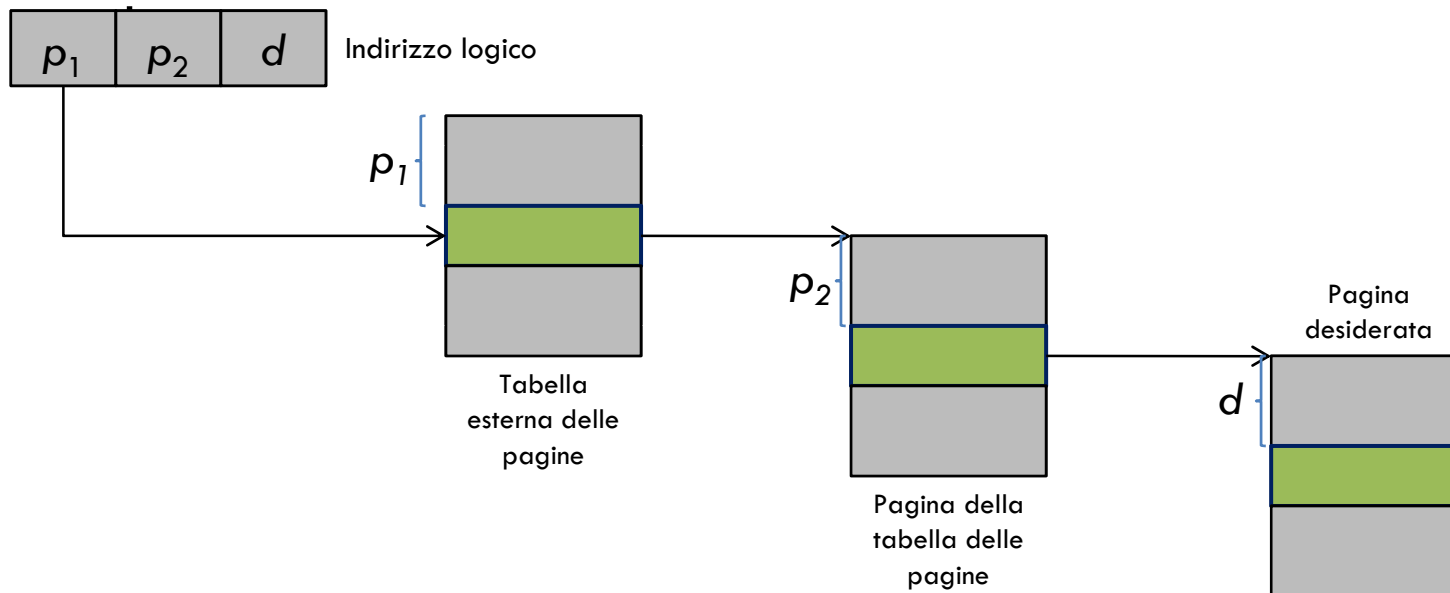
- Consideriamo una macchina a 32 bit con pagine da 4 Kbyte
- Un indirizzo logico per come lo conosciamo è definito in:
 - ▣ 20 bit più significativi = numero della pagina
 - ▣ 12 bit meno significativi = offset nella pagina
- In uno schema a due livelli anche la tabella delle pagine è paginata, quindi il numero di pagina è ulteriormente diviso in:
 - ▣ 10 bit più significativi = indice nella PT esterna che punta alla pagina *p* che contiene un pezzo della PT interna
 - ▣ 10 bit meno significativi = offset nella pagina *p*

Esempio - 2

- L'indirizzo logico è:

Numero di pagina		scostamento
p_1	p_2	d
10	10	12 bit

- La traduzione degli indirizzi avviene secondo il seguente



Paginazione a più livelli

- Un esempio di paginazione a due livelli è dato dal processore Pentium
- L'architettura VAX (Dec) usava una PT esterna di 4 elementi
- Se lo spazio logico è di 64 bit la paginazione a due livelli non è più sufficiente
 - ▣ Si dovrebbe paginare anche la tabella esterna
- Nelle architetture SPARC a 32 bit si utilizza un schema a 3 livelli
- La CPU a 32 bit Motorola 68030 ha uno schema di paginazione a 4 livelli
- Ma 4 livelli non sono ancora sufficienti per architetture a 64 bit
 - ▣ L'architettura UltraSPARC a 64 bit richiederebbe 7 livelli di paginazione
 - ▣ Se la pagina cercata non è nel TLB, la traduzione impiega troppo tempo

Tabella delle pagine di tipo hash

- Nell'utilizzo di una tabella hash si prevede una funzione hash che prende come argomento la pagina e restituisce un frame

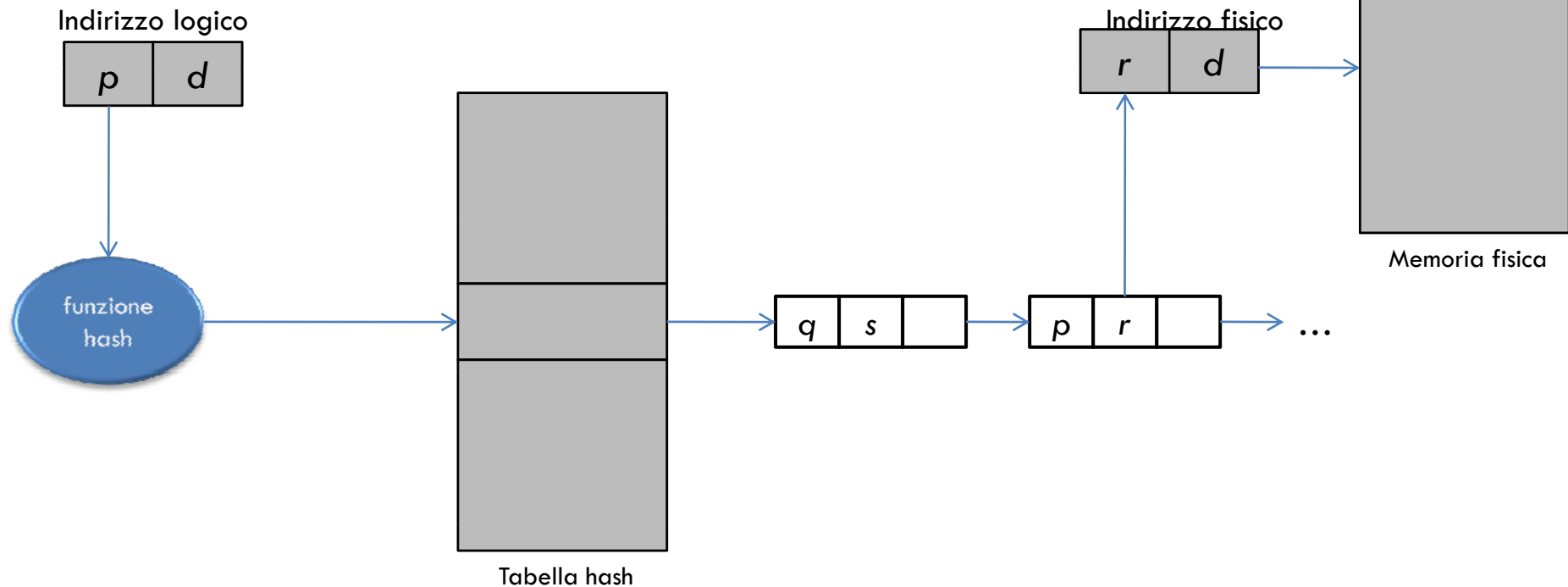


Tabella delle pagine invertite - 1

- L'occupazione dello spazio per la tabella della pagine deve essere mantenuto per ogni processo in esecuzione
- Nelle architetture a 64 bit questa tabella può crescere moltissimo
- Nei sistemi UltraSPARC a 64 bit si utilizza la tecnica della **tabella delle pagine invertite (IPT)**

Tabella delle pagine invertite - 2

- Una IPT descrive l'occupazione dei frame della memoria fisica, quindi:
 - ▣ C'è una sola IPT per tutto il sistema (anziché una PT per processo)
 - ▣ La dimensione della IPT dipende strettamente dalla dimensione della memoria primaria
 - ▣ L'indice di ogni entry della IPT corrisponde al numero del frame corrispondente di MP
- Ogni elemento della IPT contiene una coppia (*process-id*, *page-number*)
 - ▣ *process-id*: identifica il processo che possiede la pagina
 - ▣ *page-number*: indirizzo logico della pagina contenuta nel frame corrispondente a quella entry

Tabella delle pagine invertite - 3

- Ogni indirizzo logico generato dalla CPU è una tripla:
(process-id, page-number, offset)
- Per generare un indirizzo fisico si cerca nella IPT la coppia (process-id, page-number)
- Se la si trova all' i -esimo elemento, si genera l'indirizzo fisico ($i, offset$)

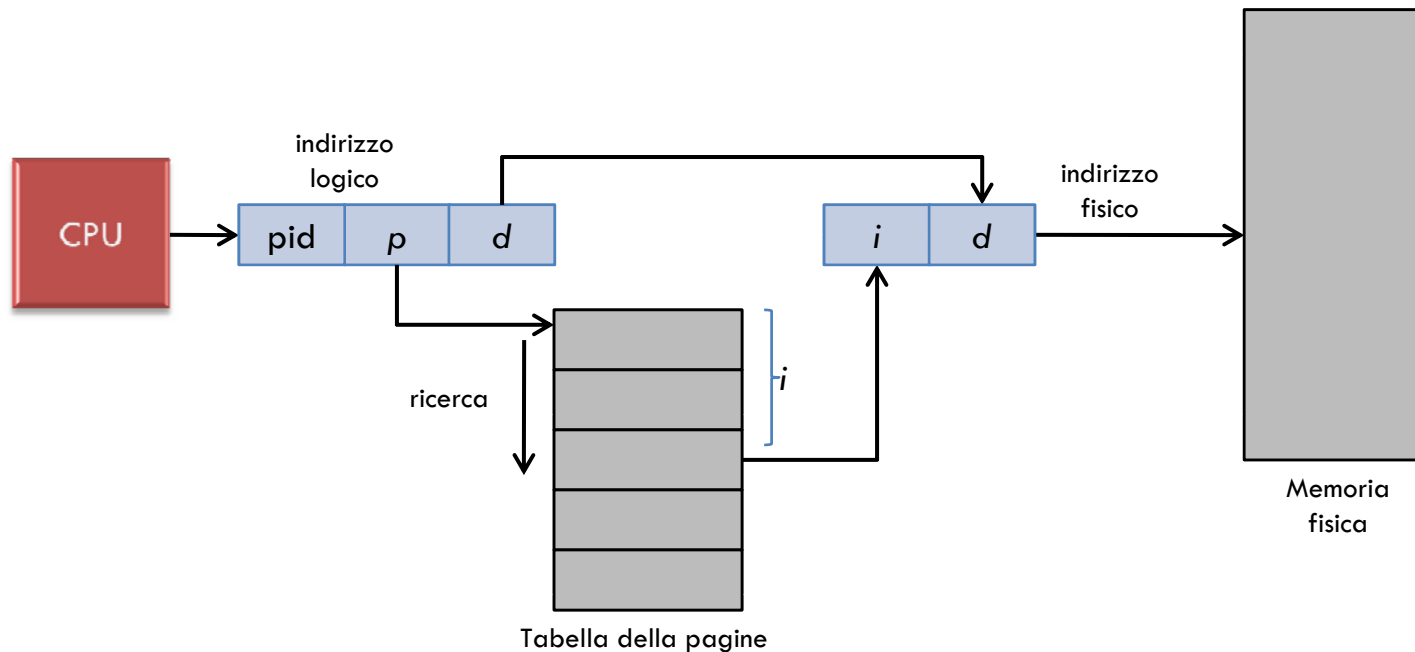


Tabella delle pagine invertite: Vantaggi e Svantaggi

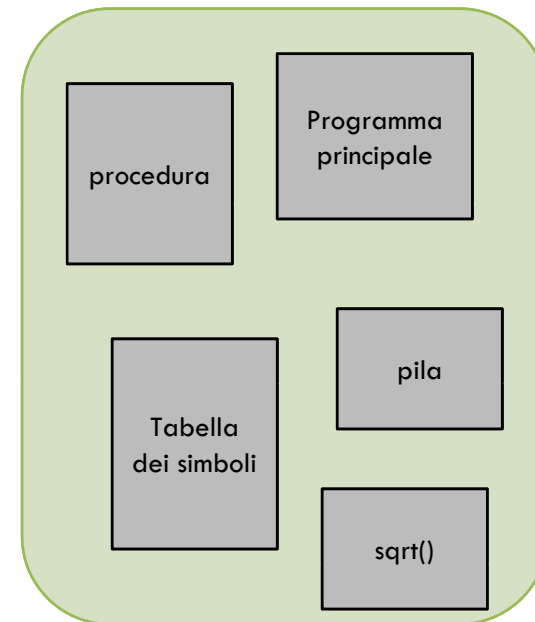
- Con l'IPT si risparmia spazio, ma si perde in efficienza per cercare nella IPT l'entry che contiene la coppia

(process-id, page-number)

- Devono essere usate delle memorie associative che contengono una porzione della IPT per velocizzare la maggior parte degli accessi

Segmentazione

- La paginazione permette una visione separata della memoria dell'utente dalla memoria fisica
 - ▣ I due spazi di indirizzamento non coincidono ma il meccanismo di traduzione permette di collegarli
- Normalmente l'utente percepisce un programma come un collezione di “segmenti” separati
 - ▣ Programma principale
 - ▣ Procedure
 - ▣ Aree dati
 - ▣ Stack
 - ▣ Tabelle dei simboli



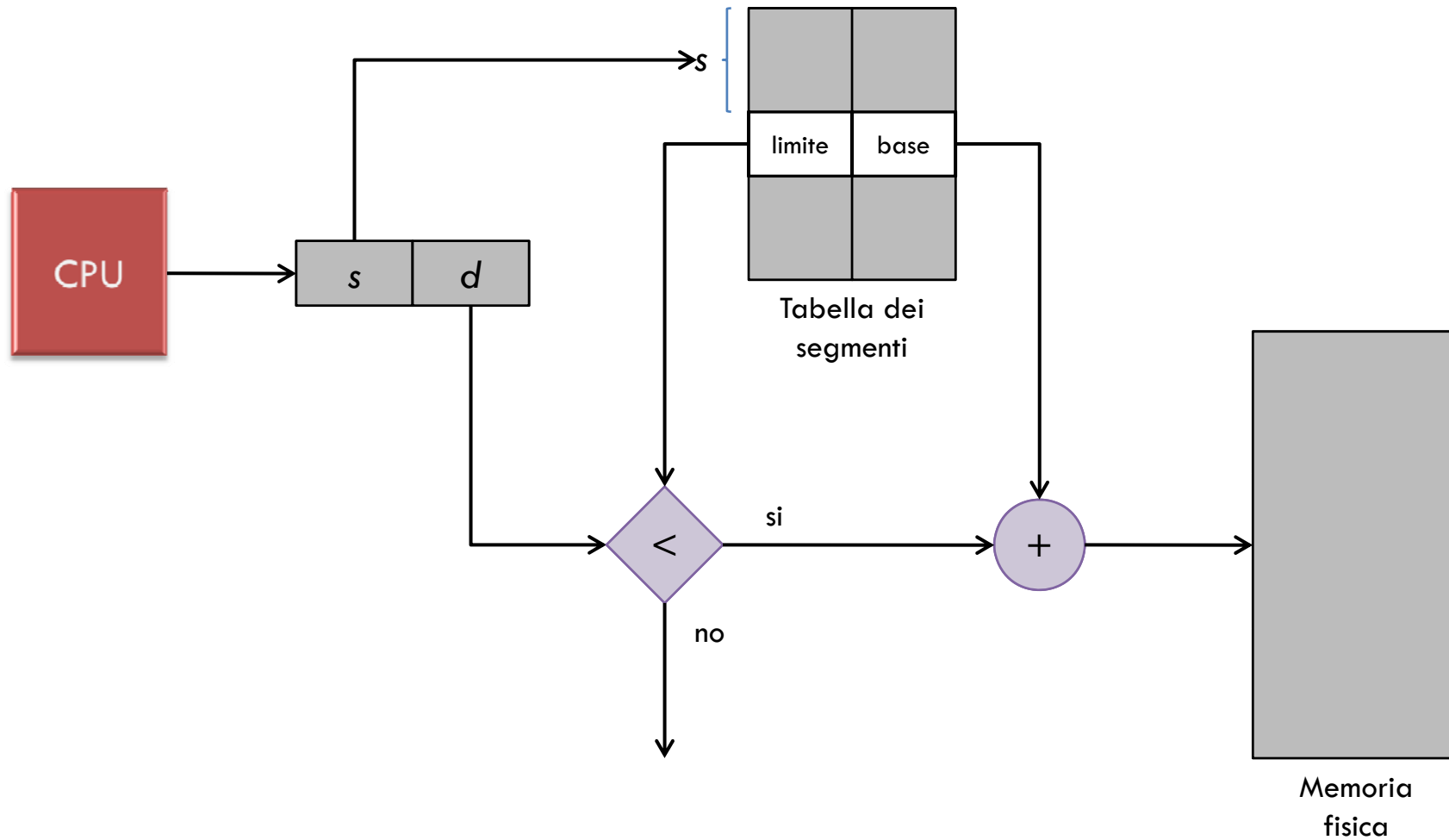
Indirizzare un elemento del segmento

- Ogni segmento è definito dalla sua dimensione
- Un elemento del segmento (codice o dati) è indicato mediante la coppia (*#segmento*, *offset*)
 - ▣ *#segmento*: numero del segmento
 - ▣ *offset*: scostamento all'interno del segmento
- Un compilatore in fase di compilazione genera automaticamente i segmenti, ad esempio un compilatore C crea i seguenti segmenti:
 - ▣ Il codice
 - ▣ Variabili globali
 - ▣ Heap, da cui si alloca la memoria
 - ▣ Variabili locali statiche di ogni funzione o procedura
 - ▣ Librerie standard del C

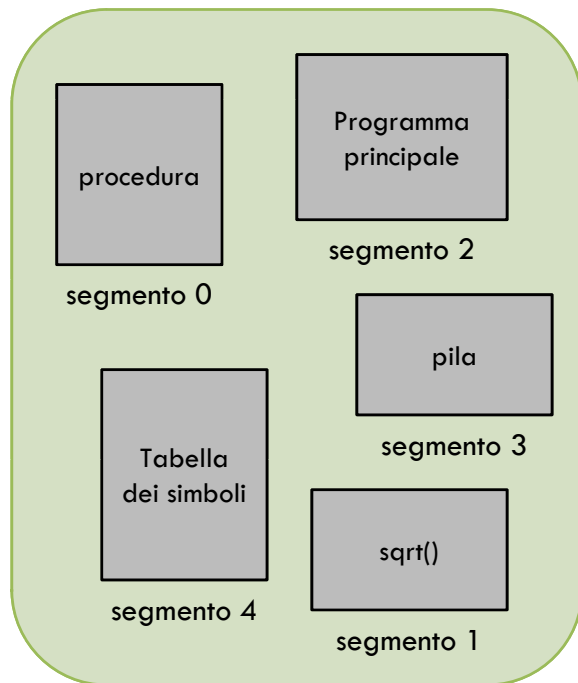
Architettura di segmentazione

- I segmenti sono allocati all'interno della memoria fisica
- Per indicizzare i segmenti abbiamo bisogno di una **tabella dei segmenti**
- Ogni elemento della tabella dei segmenti è formata da una coppia:
 - ▣ **Base del segmento:** l'indirizzo fisico di partenza del segmento in MP
 - ▣ **Limite del segmento:** la lunghezza del segmento
- Il numero di ogni segmento è l'indice all'interno della tabella contenente la coppia (*base,limite*) relativa a quel segmento

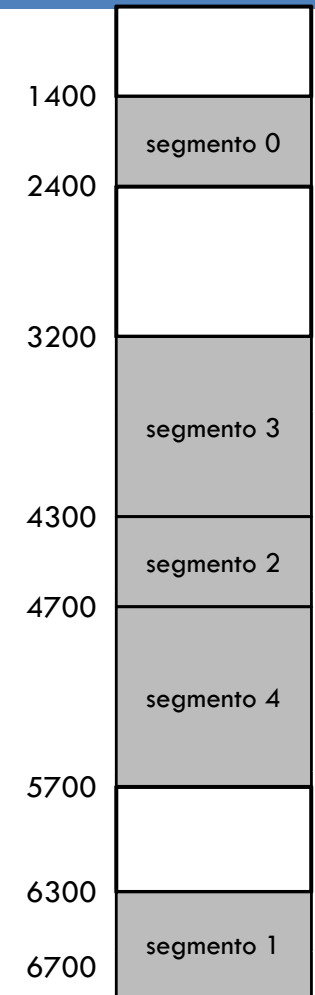
Architettura di segmentazione



Esempio di segmentazione



	limite	base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700



Memoria fisica

Architettura di segmentazione

- Se la tabella dei segmenti è piccola, può stare nei registri della macchina
- Solitamente si utilizzano due registri speciali:
 - ▣ **Segment Table Base Register (STBR)**: indirizza la ST in MP
 - ▣ **Segment Table Length Register (STLR)**: mantiene il numero di segmenti usati (ogni numero di segmento usato va confrontato con l'STLR)
- E' possibile utilizzare un meccanismo a memoria associativa per limitare l'overhead

Frammentazione

- I segmenti di un processo, come nel caso delle pagine, non devono essere contigui nella memoria fisica
- I segmenti sono di dimensione diversa, e un qualsiasi “buco” va bene, per cui:
 - ▣ E' necessario adottare strategie di allocazione best/first fit
 - ▣ Ci possono essere problemi di frammentazione esterna
 - ▣ Un segmento potrebbe non trovare spazio in ram (anche se lo spazio c'è...)
- Tuttavia, la grandezza media dei segmenti è piccola, per cui il loro uso è comunque più vantaggioso dell'allocazione contigua a partizioni variabili

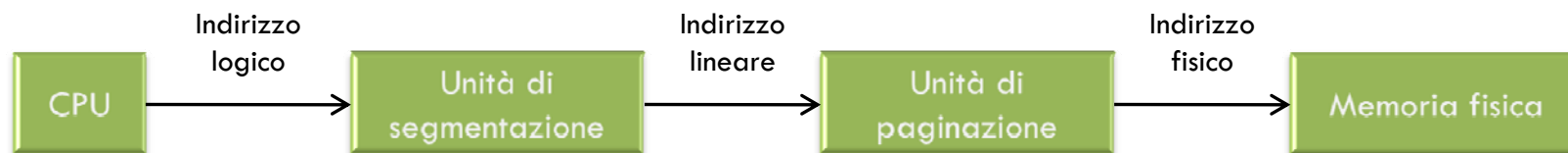
Segmentazione e Paginazione



- La segmentazione è una soluzione più “naturale” della paginazione, ma soffre degli stessi problemi (seppure mitigati) dell’allocazione contigua a partizioni variabili
- Si possono paginare i segmenti mantenendo i vantaggi della segmentazione
- Queste tecniche combinate vengono usate nella maggior parte dei sistemi operativi moderni

Il caso del Pentium - 1

- Il Pentium fornisce un supporto hardware per la paginazione e per la segmentazione paginata
- Il sistema operativo si incarica di decidere come usare il supporto hardware fornito dalla CPU
- Nel processore Pentium gli indirizzi fisici sono scritti su 32 bit, ed è quindi possibile indirizzare al massimo 4GB di memoria
- Il S.O. può scegliere di utilizzare pagine da 4 Mbyte o da 4 Kbyte
 - ▣ In quest'ultimo caso il Pentium adotta uno schema di paginazione a due livelli.
- La traduzione degli indirizzi da logici a fisici avviene seguendo lo schema



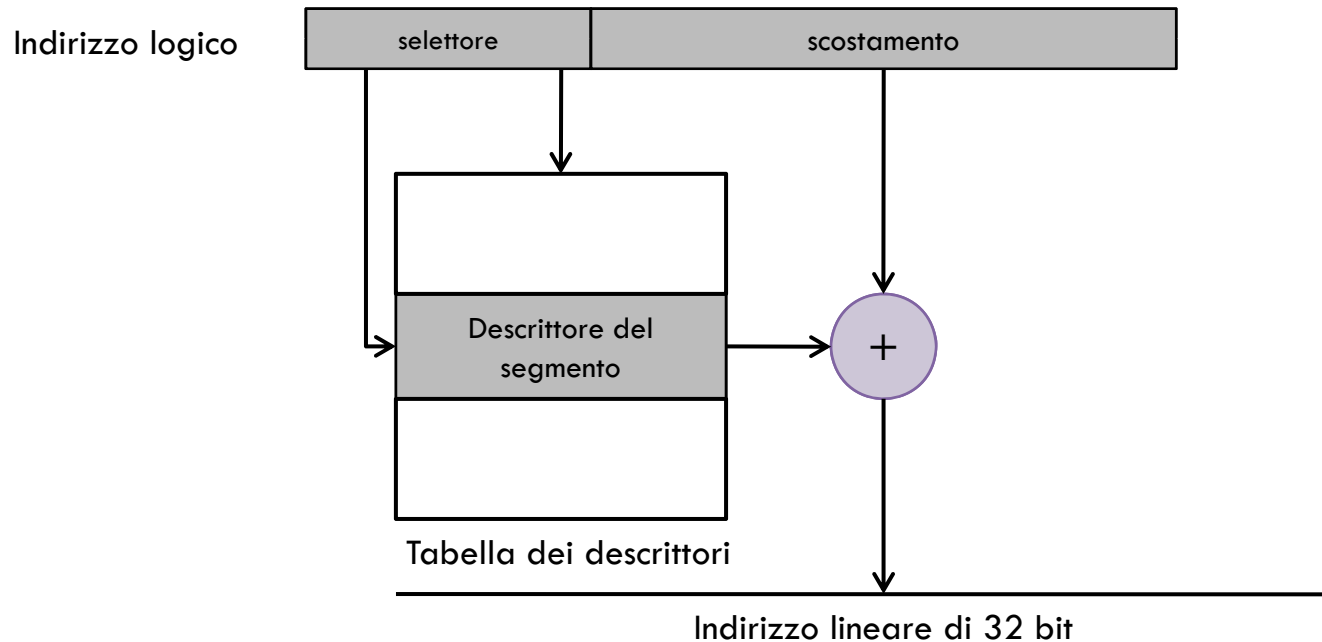
Il caso del Pentium - 2

- Il Pentium permette al Sistema Operativo di implementare anche la segmentazione paginata per lo spazio di indirizzamento dei processi
- E' possibile utilizzare un solo segmento per l'intero processo
 - ▣ In questo caso si ricade in una gestione a pura paginazione
- Oppure ogni processo può essere suddiviso in:
 - ▣ Un massimo di $2^{13} = 8192$ segmenti riservati al processo stesso
 - ▣ Ed altri 2^{13} segmenti condivisi con gli altri processi del sistema

Il caso del Pentium - 3

- Un indirizzo logico, formato dalla coppia
(*numero segmento, offset*)

viene usato per generare un indirizzo lineare a 32 bit, che verrà dato in input all'unità di paginazione



Alcune conclusioni

- Diverse sono le tecniche adoperate per la gestione della memoria alcune semplici altre complesse
- Il supporto hardware è fondamentale per diverse ragioni:
 - ▣ Determina la classe di tecniche usabili
 - ▣ Migliorare l'efficienza dei diversi approcci
- Le tecniche di gestione della MP cercano di aumentare il più possibile il livello di multiprogrammazione:
 - ▣ Permettono lo swapping e la rilocalizzazione dinamica del codice
 - ▣ Limitano la frammentazione
 - ▣ Favoriscono la condivisione del codice fra i diversi processi