

PAGNNG > 9.3

È un metodo alternativo che consente degli spazi non contigui.

METODO DI BASE 9.3.1

Si definiscono come strutture:

- frame: Singolo elemento della memoria.
Ogni frame occupa la stessa dimensione.
- pages: Singolo segmento di memoria logica.
Ogni page ha stessa dimensione dei frame.

L'esecuzione prevederà quindi di caricare le pages nei frame disponibili. (A caso per sicurezza)

Eventualmente si può usare la mem. secondaria, che è organizzata in blocchi di pari dimensione.

TRANSLATION SCHEME

Ogni indirizzo logico è composto da:

- page number
- offset

Nella traduzione

1. Si determina la pagina da usare dalla page table dell'S.O. usando il page number.

2. Trovato l'ind. di base, si usa l'offset per avere l'indirizzo fisico

L'S.O. sarà responsabile per definire la page size

ESEMPI

Dato $n = 2^m$ dim. page
 $m = 2^m$ spazio logico degli ind.

I. L.	P.num	P.off	F.num	Res
-------	-------	-------	-------	-----

"a"	0	0	S	
"d"	0	3	S	

$$n=2$$

$$m=4$$

$$0 + (S \cdot 2^2) = 20 = "a"$$

$$[3] + (S \cdot 2^2) = 23 = "d"$$

Def. da

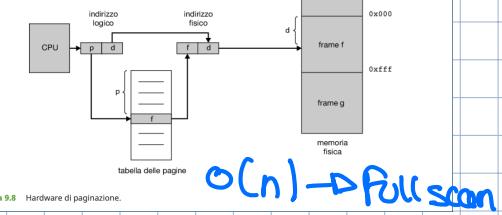
HW

page number	page offset
p	d

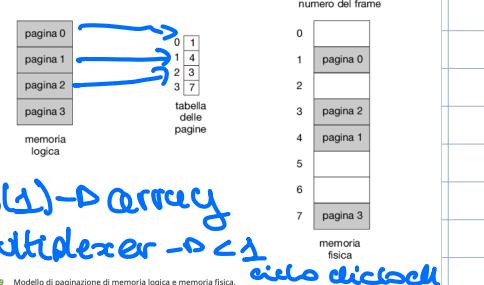
Ind: n^m ?

Ind. logico

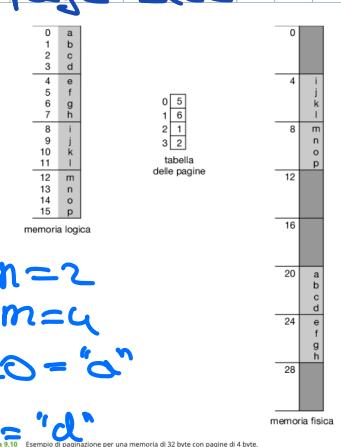
se page 4kb $\rightarrow 2^{12}$ bit



$O(n) \rightarrow$ full scan



$O(1) \rightarrow$ array
 multiplexer $\rightarrow O(1)$
 solo di accessi



"e" 1 0 6 $0 + (6 \cdot 2^2) = 24 = "e"$

FRAZIONARZIONE C'è un pre-allottrato

Dati piccoli
pre-allottrato

La paginazione consente di ridurre la paginazione esterna ma è vulnerabile alla paginazione interna.

Questo perché ogni programma ha la sua dimensione

- caso peggiore: N pagine + 1 byte → occupa 1 page in più
- caso medio: mezza pagina sprecata.

FREE FRAMES

Al loading di un programma in memoria,

si determinano le pages e se ci sono

abbastanza frame, si allocano i frame nei primi posti liberi.

Le informazioni sui frame (se libero o quale page ospita) sono disponibili nella frame table.

SUPPORTO HARDWARE 9.3.2

Ogni processo ha la sua Page Table. Essa viene salvata nel PCB e ripristinata (salvata nel context switch).

Nel PCB viene salvato il P.T.B.R., ovvero il puntatore alla memoria allocata nella tabella.

T.L.B > 9.3.2.1

Siccome la Page Table è salvata in memoria, ogni translation è raddoppiata (1 req per accedere a S.T. + 1 per avere l'indirizzo).

Per risolvere si usano le Transaction Look-aside Buffer, delle cache hardware per tenere in memoria veloce il frame.

SOSTITUZIONE E ASID

Se un elemento è già presente, vengono usate policy (round robin, meno recente, ecc..) per sostituirlo. Alcune pages (riservate al kernel) non possono essere rimosse (wired down).

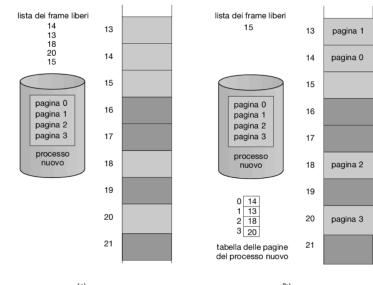


Figura 9.11 Frame liberi: (a) prima e (b) dopo l'allocazione.

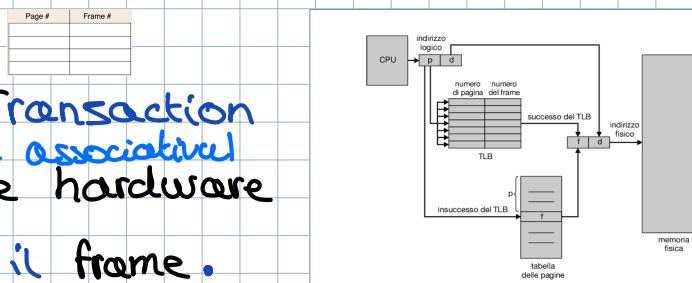


Figura 9.12 Hardware di paginazione con TLB.

ASID (Address Space Identifier)

Inoltre, alcuni TLB memorizzano gli identificatori dello spazio di indirizzamento (asid). Ogni page dovrà avere un suo asid. Questo consente alla Page Table di

- Avere elem. di spazi diversi
- Avere elem. dello stesso spazio → Necessario flush.

EFFECTIVE ACCESS TIME (EAT)

Per calcolare il tempo di accesso si definiscono:

- hit ratio: % di volte in cui il TLB contiene l'elemento
- tempo d'accesso: del singolo elemento.

Il calcolo prevede la somma di:

(1) Calcolare l'EAT nel caso buono (tlb hit)

(2) Calcolare l'EAT nel caso cattivo $[(\text{hit ratio} - 100) \cdot t_{\text{acc}}]$

ESEMPIO: Dato time unit $\alpha = 10 \text{ n.s.}$ e hit ratio: 80%

(1) Calcolo caso buono: $0.8 \cdot 10 = 8 \text{ n.s.}$ TPIZ accessi a mem

(2) Calcolo caso cattivo: $0.2 \cdot (10 \cdot 2) = 4 \text{ n.s.}$

(3) Somma: $(8 + 4) \text{ ns} = 12 \text{ ns} \rightarrow$ rallenta 20%

PROTECTION > Q.3.3

È possibile inserire dei bit di protezione nella Page Table.

I bit gestiscono i

- permessi (rwx) dei frame.
- validità del frame (impostato da S.O.)

Non tutte le pages vengono allocate come

registers perché di solito la memoria usata è molto minore.

Alcune architetture usano anche il Page Table Length Register come limite dello spazio di indirizzamento.

SHARED PAGES > Q.3.4

Se più processi devono eseguire lo stesso codice al posto di avere 2 copie, se il codice è rientrante, si possono condividere le pagine in comune.

0	1	2	3	4	5	6	7	8	9	...	pagina n
12.287											
10.468	pagina 5	bit di validità/ non validità									
	0	2	v								
	1	3	v								
	2	4	v								
	3	7	v								
	4	8	v								
	5	9	v								
	6	0	i								
	7	1	i								
00000	pagina 0	tabella delle pagine									

Figura 9.13 Bit di validità (v) o non validità (i) in una tabella delle pagine.

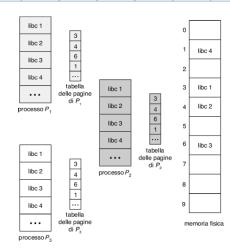


Figura 9.14 Condivisione delle librerie standard del C in un ambiente paginato.

La mem. condivisa si può implementare con shared pages.

STRUTTURA DELLA PAGE TABLE ➤ Q.4

È possibile strutturare la Page Table come:

PAGINAZIONE GERARICA Q.4.1

Siccome le memorie attuali sono molti grandi (2^{32} - 2^{64}), questo implica che le Page Table saranno molto grandi a loro volta. Quindi, bisogna fare in modo che la Page Table non sia contigua in memoria:

PAGINAZIONE A 2 LIVELLI (forward mapped PT)

1° metodo che consiste di paginare la Page Table con una Page Table esterna.

Ogni indirizzo logico dovrà contenere:

- P_1 : Indice Page Table esterna.
- P_2 : Offset della pagina trovata da P_1 .
- d : Offset del frame.

La paginazione a 2 livelli regge fino ad architetture a 32 bit (2^{32}). Architetture più grandi come 64 bit avrebbero bisogno di 3/4 layer di paginazione, aumentando la complessità.

MASKED PAGE TABLES Q.4.2

Prevedono l'uso di una tabella Hash con liste di trabocco. In questo caso

- Key: Il # della Page
- Value: L'indirizzo del frame, al quale aggiungere un eventuale offset.

Per i sistemi a 64 bit il value si riferirà a gruppi di pagine contigue (clustered Page Table)

INVERTED PAGE TABLE Q.4.3

Prevede di strutturare la Page Table con 1 elem per frame ➤ Singola Questo per evitare di censire tutti gli indirizzi virtuali.

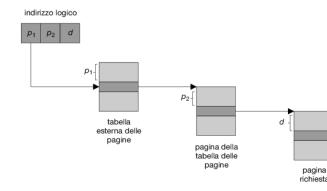


Figura 9.16 Traduzione degli indirizzi per un'architettura a 32 bit con paginazione a due livelli.

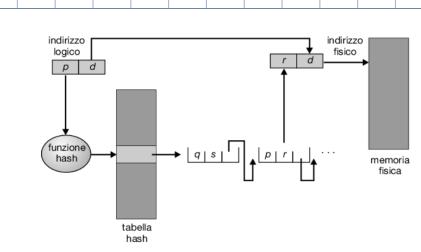
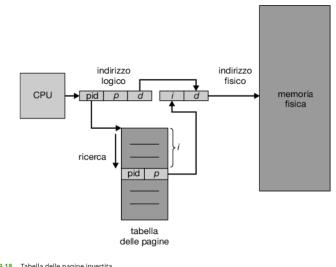


Figura 9.17 Tabella delle pagine di tipo hash.

Gli indirizzi logici saranno formati da:

- PID: Per filtrare dentro la tabella i frame con lo stesso spazio di indirizzamento.

- P : Page number
- d : Offset



Trovato p, la sua posizione i-esima nella tabella sarà il frame.

+ vantaggi: - memoria occupata per Page Table

- svantaggi: Ricerca in $O(n)$ sulla Page Table

- ↳ con hash table → tempi ridotti ma $\times 2$ accessi
- ↳ migliorabile con T.L.B.
- ↳ shared memory più dificile da implementare

ORACLE SPARC SOLARIS 9.4.4 → Skipped

SWAPPING > Q.5

È una policy che consente di spostare un intero processo o una parte all'interno della memoria secondaria (backing store):

- swap out : Spostamento verso la mem. secondaria.
- swap in : \rightleftarrows verso la mem. primaria.

In questo modo, lo spazio di ind. può effettivamente superare la mem. primaria.

STANDARD SWAP Q.5.1 (Unix)

Prevede di sovrascrivere la memoria secondaria.

PAGING SWAP Q.5.2. (Windows/ Linux)

Prevede di effettuare lo swap delle singole pagine.

PRIORITY SWAP

Prevede di usare roll out e roll in per spostare processi con meno priorità.

Swapping time è una metrica usata per il tempo di swap.

POSIZIONE PAGINA IN SWAP IN

Nello swap-in, la posizione della pagina al ritorno

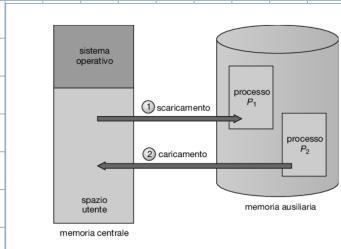


Figura 9.19 Avvicendamento standard di due processi con un disco come memoria auxiliaria.

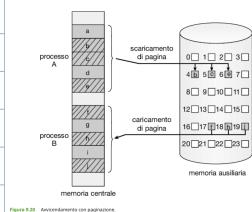


Figura 9.20 Avvicendamento con paginazione.

può essere:

- lo stesso: fare attenzione a frame in uso.
- una nuova: Necessita che ogni I/O o il bus usi gli ind. logici e translation.