MEMORIA VIRTUALE > 10

La mem. virtuale è un meccanismo che consente di eseguire processi che possono anche non essere (completam.) in memoria. Questo consente di

- separare spazi logici e fisici
- permettere ind. > mem. fisica
- compilare i programmi in maniera indipendente.

- Facilita I/O, shared mem
- creazione processi.
- e multiprog.

Si può implementare con Demand Paging o Demand Segment.

DEMAND PAGING > INTRO 10.1

Si sfrutta il Paging per realizzare la mem. virtuale
In questo caso l'indirizzamento può eccedere per

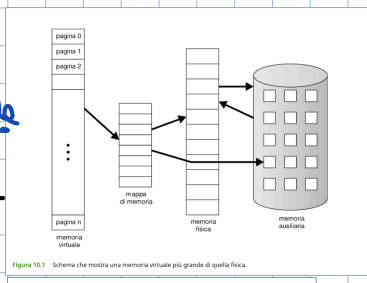

Figura 10.1 Schema che mostra una memoria virtuale più grande di quella fisica.

- swap: Parte delle pagine sono in mem. secondaria
- demand: Le pagine non necessarie come il "buco" fra stack e heap non viene allocato (solo se crescono).

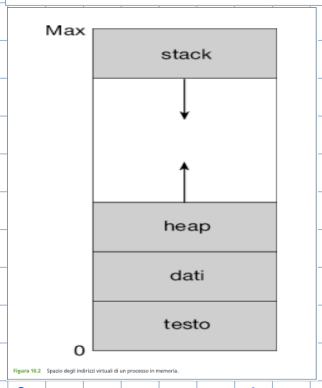

Figura 10.2 Spazio degli indirizzi virtuali di un processo in memoria.

Assegnati da SO più di quanti servono (WIN)

I vantaggi del demand paging sono:

- sharing facilitato di librerie (Shared Pages)
- sharing facilitato di mem. condivisa. Ogni processo pensa sia sua
- speedup di creazione dei processi, copiando virtualmente i dati.

DEMAND PAGING > NEL DETTAGLIO 10.2

Il concetto principale è di caricare la pagina solo se serve in mem. primaria. Tutte le altre pagine saranno in mem. secondaria.

IMPLEMENTAZIONE (BIT VAL.)

Si sfrutta il bit di validità:

- valido: Pagina è valida e in mem.
- invalido: Pagina invalida o non in mem.

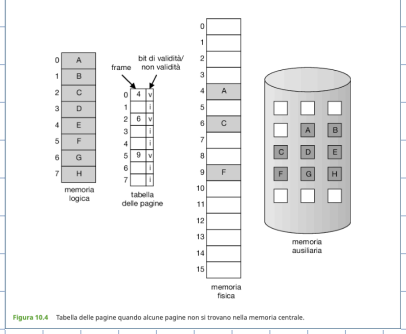

Figura 10.4 Tabella delle pagine quando alcune pagine non si trovano nella memoria centrale.

PAGE FAULT

Quando si tenta di accedere ad una page non in mem:

1. ? La Page era nel corretto spazio d'ind.?
   1. T vai avanti
   2. F segmentation fault
2. L'H.W. non trova la page nella PageT.
   e lancia una trap (come interrupt) (Page Fault)
3. L'S.O. riceve l'interrupt → effettua swap in della pagina.
   e aggiorna la Page Table
4. ri-eseguo l'istruzione.

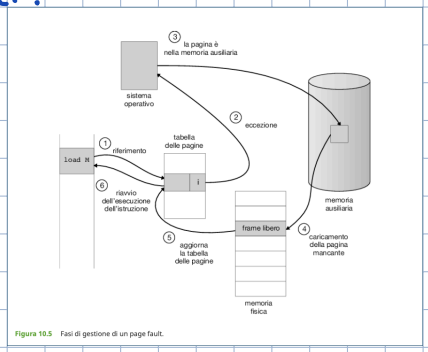

Figura 10.5 Fasi di gestione di un page fault.

È importante che l'istruzione da eseguire sia senza effetti collaterali (es: modifiche a P.C.) in modo da poter ri-eseguire.

PAGE FAULT ESTREMO (Pure Demand Page)

In casi estremi è possibile avviare il processo senza pagine e lasciare page fault multiple.

LISTA FRAME LIBERI 10.2.2

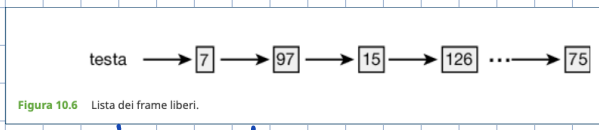

Figura 10.6 Lista dei frame liberi.

L'S.O. mantiene una lista di frame ai quali associare le Pages. Ogni frame sarà azzerato prima dell'uso.

PRESTAZIONI 10.2.3

Per calcolare il tempo di accesso effettivo si usa la formula:

$t.a.e = (1-p) \cdot ma + p \cdot t.PF$ dove

- p è la probabilità di page fault
- ma = Temp. accesso memoria
- t.PF è il tempo di gestione del Page fault.

Esempio: Dato $t.PF = 8\,ms$ e $ma = 200\,ns$

$EAT = (1-p) \cdot 200_{ns} + p \cdot 8ms = (1-p) \cdot 200 + p \cdot 8000000$

$= 200 - 200p + p \cdot 8'000'000 = 200 + 7'999'800p$

Se $p = \frac{1}{1000}$ ⇒ $200 + 7'999'800 \cdot \frac{1}{1000} = 200 + 8000$

$= 8200\,ns$

Se degr. < 10% ($\overset{10\%}{220} > 200 + 7'999'800p$)

$20 > 7'999'800p$

$0.0000025 > p$ (1 P.F ogni $\cong 400'000$)

Infine si può supportare l'ottimizzazione con

- Uso di area di swap
- Page in + free al posto di Page out per file binari.