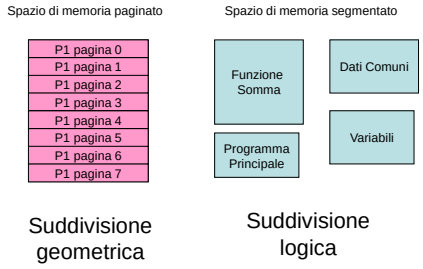
Gestore della memoria
II parte

Segmentazione

- La segmentazione come la paginazione prevede di dividere lo spazio di indirizzamento logico del processo(ovvero la memoria necessaria al processo) in più parti
- Quello che cambia è il modo in cui viene effettuata tale suddivisione

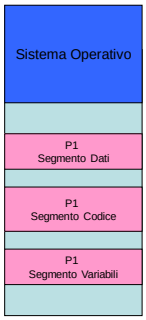
Confronto
Paginazione - Segmentazione



Suddivisione geometrica

Suddivisione logica

Segmentazione



Segmentazione

- La suddivisione di un processo in segmenti è effettuata sulla base di criteri logici che riflettono l'organizzazione del software(il programma principale e le funzioni, i dati e le variabili sono in segmenti diversi)
- I segmenti sono blocchi di memoria contigui (come le partizioni prima viste)
- La memoria fisica in questo caso non è suddivisa in frames bensì come nel caso delle partizioni forma un'unico blocco da riempire in maniera del tutto analoga a quanto già visto per le partizioni

Segmentazione: Vantaggi

- Vantaggi: + semplice la:
  - Protezione
  - Condivisione
- Esempio: se il codice di una funzione è accessibile in sola lettura e si trova in un segmento separato, invece di aggiungere una serie di bit di attributo ad ognuna delle migliaia di pagine che lo compongono utilizzo degli attributi per l'intero segmento
- In maniera analoga posso proteggere il codice di un processo dagli altri processi in maniera analoga a quanto fatto per le partizioni

Segmentazione

- Inoltre se ad un certo punto il processo ha bisogno di altro spazio, posso semplicemente aumentare la dimensione del segmento senza toccare il resto
- Nella paginazione invece è più complicato inserire nuove pagine
- Ciò è particolarmente utile nel caso di strutture dati dinamiche: esempio vettori e file la cui dimensione è stabilita non in fase di compilazione bensì di esecuzione

Tabella dei segmenti

- Abbiamo visto che per molti versi **la segmentazione assomiglia alla partizione dinamica**
- In questo caso però ad **un processo** corrispondono **più "partizioni" ovvero più segmenti**
- Come nella paginazione esiste una ***tabella dei segmenti*** che contiene gli **attributi** dei vari segmenti di un processo oltre naturalmente alle informazioni necessarie per descriverne la **collocazione nella memoria fisica**

Collocazione nella memoria fisica specificata con due valori per ogni segmento: **Inizio**(*Base*) e **dimensione**(*Limit*)

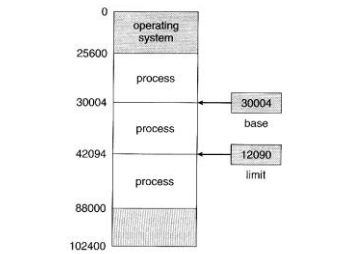


Figure 8.1 A base and a limit register define a logical address space.

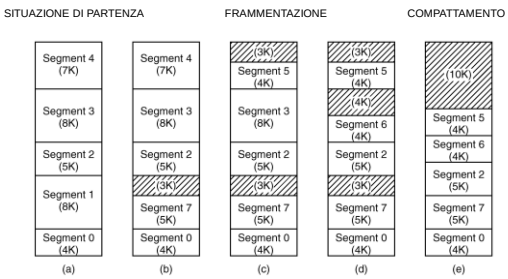
Tabella dei segmenti

Segmento	Attributi	Dimensione	Indirizzo di base
0	RW	12090	30004
1	R	1000	50032
2	RWE	32400	60440

Segmentazione : Svantaggi

- Come le partizioni dinamiche questa strategia produce frammentazione esterna
- Stessi rimedi, stesse strategie di allocazione

Frammentazione esterna



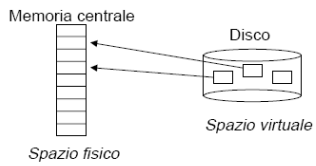
Memoria Virtuale

- I processi vengono caricati solo in parte in RAM

Memoria virtuale: principi di funzionamento

- I programmi indirizzano lo ***spazio virtuale***
- L'intero spazio virtuale è allocato in **memoria secondaria** (sull'hard-disk)
- Si mantiene in memoria centrale **solo un sottoinsieme** dello *spazio virtuale*
- Si trasferiscono man mano solo le parti dello spazio virtuale alle quali il programma fa riferimento

Spazio fisico e virtuale



Memoria virtuale: motivazioni

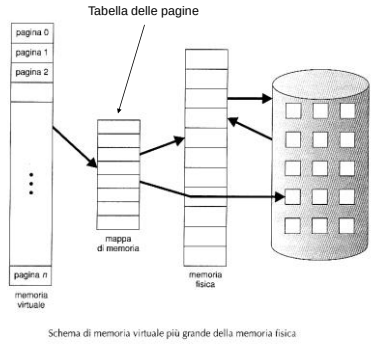
- La memoria virtuale si basa sul modo in cui i programmi funzionano
- La sequenza di **riferimenti** ovvero degli **indirizzi delle locazioni di memoria** a cui un processo accede **non è casuale**!!!
- Tutti i programmi hanno una **forte località** nei riferimenti(alla memoria)…
- …insomma **i salti sono poco frequenti !**

Località dei riferimenti

- **Località Spaziale**
  - *Riferimenti successivi hanno forte probabilità di accedere a indirizzi contigui*
- **Perché :**
  - *Le istruzioni vengono eseguite in sequenza,*
  - *Anche le strutture dati(vettore, file,…) vengono lette in sequenza (in genere)*

Località dei riferimenti

- **Località Temporale**
  - *C'è una forte probabilità di avere **più** accessi allo **stesso** indirizzo entro un **breve** intervallo di tempo*
- **Perché:**
  - *Nel codice vi sono i **cicli** che comportano numerose ripetizioni delle stesse istruzioni a breve*
  - *All'interno di una funzione è molto probabile che modifichi sempre le **stesse** variabili, lo **stesso** file, lo **stesso** vettore, etc… più volte in tempi successivi*



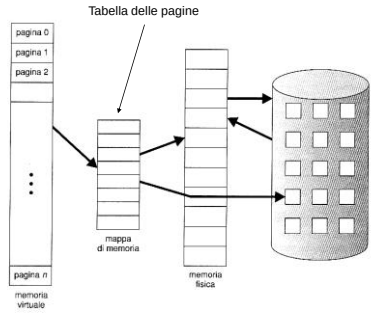
Schema di memoria virtuale più grande della memoria fisica

Memoria Virtuale + Paginazione

- Se la **pagina indirizzata NON si trova in RAM** si verifica un **page fault**
  - processo *interrotto* → *in attesa del caricamento*
  - pagina caricata in RAM dal disco rigido
  - processo → coda pronti
- Devo cercare di **evitare i page fault** perché comportano l'interruzione del processo per caricare la pagina **rallentando l'esecuzione** dell'applicazione(vedere più avanti il thrashing)

Memoria Virtuale + Paginazione

- Alcune pagine vengono mappate su frame di memoria fisica altre sul disco
- per distinguere le pagine in memoria da quelle su disco si ricorre a un bit di validità:
  - bit **valid** = 1: la pagina è in memoria e accessibile
  - bit **valid** = 0: la pagina è su disco
- in caso di *page fault* viene lanciato un interrupt, e la pagina richiesta viene caricata in un frame libero, con conseguente aggiornamento della tabella delle pagine



Schema di memoria virtuale più grande della memoria fisica

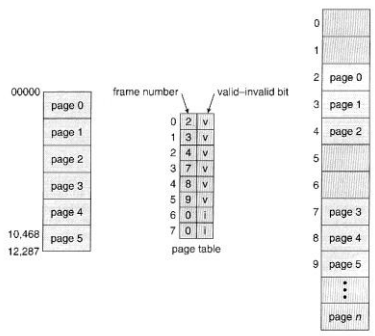


Figure 8.12 Valid (v) or invalid (i) bit in a page table.

Algoritmi di paginazione

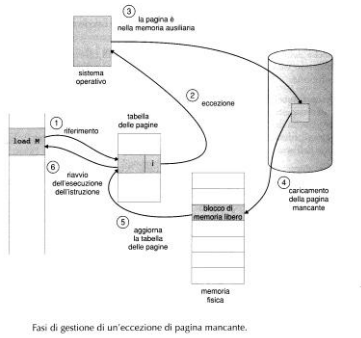
– Paginazione su richiesta (Demand paging)
  - si caricano le pagine **solo** in seguito a un *page fault*

– Avvicendamento (Swapping)
  - **tutte** le pagine che un processo usa vengono spostate tra memoria centrale e secondaria in blocco

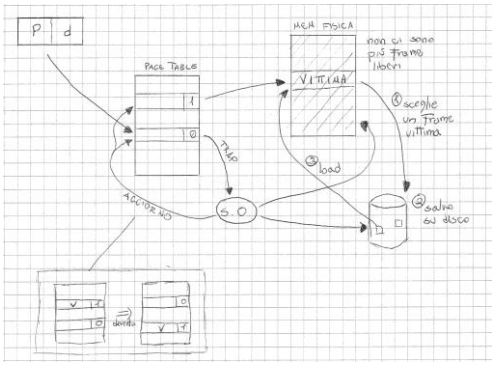
– Pre-Paginazione (Pre-paging)
  - **anticipa** il caricamento in memoria rispetto al momento dell'uso, sfruttando il principio della località spaziale

In caso di Page Fault (Memoria Virtuale + Paginazione)

- Ricerca di un frame libero
- Se non vi sono frame liberi si impiega un algoritmo di sostituzione delle pagine per scegliere una pagina vittima
- si scrive la pagina vittima sul disco (area di swap = avvicendamento); si modificano conseguentemente le tabelle delle pagine e dei blocchi di memoria
- si scrive la pagina richiesta nel blocco di memoria appena liberato; si modificano le tabella delle pagine e dei blocchi di memoria
- si riavvia il processo utente



Fasi di gestione di un'eccezione di pagina mancante.



Swapping

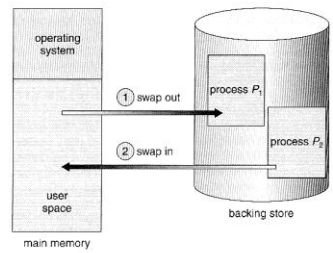


Figure 8.5 Swapping of two processes using a disk as a backing store.

Pre-paging : Working set

- Il S.O. cerca di prevedere quali saranno le pagine richieste e le carica preventivamente in RAM
- Il working set è l'insieme di pagine utilizzate più frequentemente
- Il S.O. le carica all'avvio e aggiorna l'elenco continuamente

Memoria Virtuale : vantaggi

- permette di eseguire **programmi che non sono allocati completamente in memoria**;
- Ne deriva che:
  - È necessaria meno memoria per l'esecuzione di un processo → posso caricare + processi
  - È possibile eseguire programmi potenzialmente **più grandi** della RAM

Osservazioni

- La memoria virtuale è particolarmente efficace in un contesto di multiprogrammazione
- Un processo andato in page fault viene sostituito sulla CPU da un altro

Memoria Virtuale :svantaggi

- Se non alloco abbastanza memoria fisica ad ogni processo sono costretto ad accedere troppo spesso al disco : Thrashing
- L'effetto percepito dall'utente è quello di un rallentamento se non temporaneo blocco delle applicazioni utilizzate

Thrashing

Problema: decidere quanti frames dare a ciascun processo

- Se un processo ha troppi pochi frames va in *page fault* troppo spesso
- Se il **tasso di page fault** diventa **eccessivo** il sistema rallenta
- Si dice che il sistema va in ***thrashing***

Page Fault Rate

- Il **Page Fault Rate** (PFR) dipende da:
  - Numero di frames
  - Strategia di sostituzione delle pagine (RANDOM, FIFO, LRU, NRU)
- Il S.O. varia *dinamicamente* il numero dei processi in memoria centrale ed il numero di frame loro assegnati in modo da controllare il PFR di ciascuno:

  min < PFR <max

Paginazione

abbiamo bisogno di 2 algoritmi:

- **algoritmo di assegnazione delle pagine**
- **algoritmo di sostituzione delle pagine**

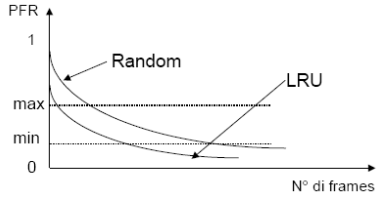
Algoritmo di sostituzione

- un algoritmo di sostituzione è tanto migliore quanto più limita i page fault (estremamente costosi per il sistema poiché costringono ad accessi al disco)
- osservazione: in caso di page fault, se nessun blocco di memoria è libero, si rendono necessari due trasferimenti di pagina (uno fuori e uno dentro la memoria) -> raddoppia il tempo di gestione dell'eccezione

Algoritmo di sostituzione

- Vari algoritmi come nel caso dell'allocazione, ad esempio:
  - RANDOM = (**casuale**) scelgo una pagina a caso
  - FIFO = la vittima è la pagina da più tempo in memoria
  - LRU = la vittima è la pagina usata meno di recente (**Least Recently Used**)
  - NRU = (**Not Recently Used**) non usata recentemente : approssimazione di LRU

Controllo del Trashing



Strategie di sostituzione

- Le strategie di sostituzioni possono essere:
  - Locali
  - Globali
- Locali se la pagina vittima (il frame da liberare) viene scelta tra le pagine del processo
- Globali se il frame viene scelto tra tutti quelli disponibili