Sistemi monoprogrammati

• Esegue un solo processo alla volta
• La memoria disponibile è ripartita solo tra quel processo e il S/O
• L'unica scelta progettuale rilevante in questo caso è decidere dove allocare la memoria (dati e programmi) del S/O
• La parte di S/O ospitata in RAM è però solo quella che contiene l'ultimo comando invocato dall'utente

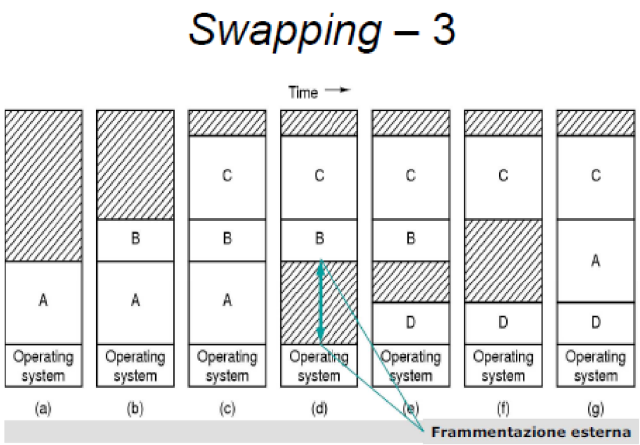
Sistemi multiprogrammati

La forma più rudimentale di gestione della memoria crea una partizione per ogni processo
• A ogni nuovo processo (o lavoro) viene assegnata la partizione di dimensione più appropriata
– Una coda di processi per partizione
– Scarsa efficacia nell'uso della memoria disponibile
• Assegnazione opportunistica
– Una sola coda per tutte le partizioni
• Quando si libera una partizione questa viene assegnata al processo a essa più adatto e più avanti nella coda
• Oppure assegnata al "miglior" processo scandendo l'intera coda
– I processi più "piccoli" sono discriminati quando invece meriterebbero di essere privilegiati in quanto più interattivi

Valutazione probabilistica di quanti processi debbano eseguire in parallelo per massimizzare l'utilizzazione della CPU

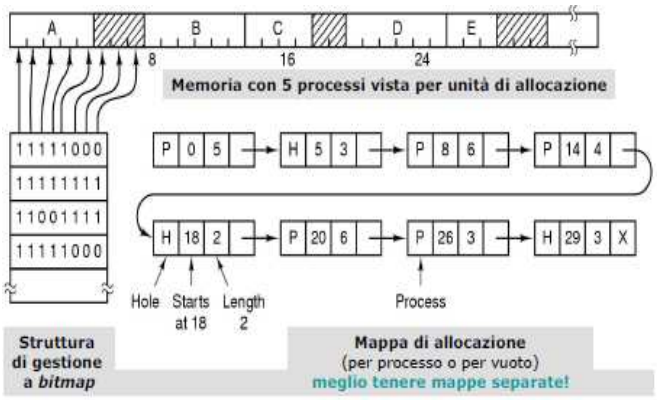
Swapping tecnica che permette di alternare processi in memoria senza garantire allocazione fissa.Trasferisce processi interi e assegna partizioni diverse nel tempo.Il processo rimosso viene salvato su una memoria secondaria 8solo le parti modificate ovviamente)

E' chiaro che processi diversi richiedono partizioni diverse assegnate ad hoc (rischio di frammentazione esterna,occorre ricompattare periodicamente la memoria principale pagando un costo temporale importante)Inoltre le dimension di memoria di un processo possono variare nel tempo!Quindi meglio assegnarle con margine



Strutture di gestione della memoria RAM

-*Mappe di bit* (memoria vista come un insieme di unità di allocazione(1bit per unità)),struttura di gestione molto grande



-*Liste collegate*: memoria vista come un insieme di segmenti (= processo|spazio libero tra processi).Ogni elemento della lista rappresenta un segmento e ne specifica punto d'inizio,ampiezza e successore.Le liste sono quindi ordinate per indirizzo di base.

Esistono varie strategie di allocazione:
First Fit: il primo segmento libero ampio abbastanza a contenere il processo verrà usato
Next Fit: come First Fit ma cercando sempre avanti
Best Fit: il segmento libero più adatto
Worst Fit:utilizza sempre il segmento libero piu ampio
Quick Fit:liste diverse di ricerca per ampiezze tipiche

Memoria Virtuale

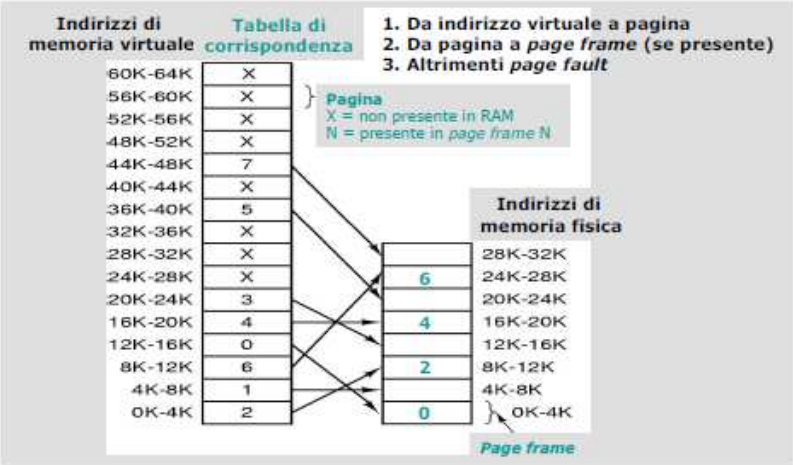
Il principio cardine è che ogni processo può avere ampiezza superiore della ram disponibile (basta caricarne in ram solo la parte strettamente necessaria lasciando il resto sul disco,senza intervento del programmatore).Oni processo ha un prorpio spazio di memoria virtuale.

Non è generalmente distinta per dati ed istruzioni.

Gli indirizzi generati dal processo non denotano piu direttamente una locazione in RAM ma vengono interpretati da un unità detta MMU che li mappa verso indirizzi fisici reali.

Modello 1:Paginazione

- La memoria virtuale è suddivisa in unità a *dimensione fissa* dette pagine.
- La RAM è suddivisa in unità “cornici” *ampie come le pagine* (page frame).
- Trasferimenti da e verso disco avvengono sempre per pagine intere (inoltre di ognuna è necessario sapere se risiede in RAM oppure no attraverso un *Bit di presenza*)
- Se una pagina è assente in RAM riferita si genera un evento detto page fault gestito dal SO tramite trap.



- La traduzione da indirizzo virtuale a fisico avviene tramite tabella delle pagine (che può essere molto grande).
- La traduzione deve essere molto veloce (ogni istruzione potrebbe far riferimento piu volte alla tabella).L'indirizzo di disco dove è presente la pagina quando non è in ram non è nella tabella,e nel caso di page fault sta al SO caricarla.

- La tabella è cosi grande da dover risiedere in RAM.Nasce quindi il concetto di ***TLB*** (*translation lookaside buffer*),ossia di una piccola memoria associativa interna alla MMU che si basa sul concetto che un processo in genere usa piu frequentemente poche pagine (funziona come una cache).

-Ogni indirizzo emesso dalla MMU viene prima trattato con la TLB (se la pagina è presente in TLB si ha traduzione senza accesso alla tabella delle pagine).Se non presente si ha l'equivalente di un cache miss e le info richieste vengono prelevate dalla tabella delle pagine e caricate in TLB.

- Oggi le TLB sono principalmente SW (prestazioni accettabili,MMU più semplice e piu piccola,quindi più spazio in CPU ad esempio per maggior cache).

Problema:con le architetture 64bit le tabelle assumono dimensioni.

*Soluzione:*Tabella invertita

Non si usa più una riga della tabella per pagina ma una riga per page frame (pagina in RAM).
La traduzione da virtuale a fisico però diventa ancora piu complessa,dal momento che la pagina potrebbe risiedere in un quasiasi page frame e bisognerebbe scandire l'intera tabella

---

Politiche di rimpiazzo delle pagine

NRU (not recently used)
- Per ogni page frame vengono aggiornati
  - Bit Modified inizializzati a 0 dal SO
  - Bit Referenced,posto a 0 periodicamente dal SO per stimare la frequenza d'uso
- Le pagine nei frame sono classificate in
  - Classe 0:non riferita,non modificata
  - Classe 1:non riferita,modificata
  - Classe 2:riferita,non modificata
  - Classe 3:riferita,modificata
- questa politica *sceglie una pagina a caso nella classe non vuota a indice piu basso*

FIFO
- Rimuove la pagina di ingresso più antico in RAM (basta una lista ordinata di page frame in cui ogni inserimento avviene in coda e le rimozioni avvengono dalla testa

Second Chance corregge FIFO rimpiazzando solo le pagine con bit Referenced=0

Orologio come second chance ma i page frame sono mantenuti in una lista circolare e l'indice di ricerca si muove come una lancetta

LRU approssima l'algoritmo ottimale,ma richiede hardware dedicato e lista aggiornata ad ogni riferimento a memoria

NFU realizzabile via sw,per ogni page frame aggiorna periodicamente un contatore C che cresce di piu se il bit Referenced=1.Non perde mai memoria

Aging NFU modificato,approssima LRU (valuta solo periodicamente,ed usando N bit per C perde memoria dopo N aggiornamenti

---

Cos'è il WS?

- Il Working Set è l'insieme delle pagine che un processo ha in uso in un dato istante.Conoscerlo esatt è troppo costoso.
- Se la memoria non basta ad ospitare tutto il WS si verifica il thrashing.
- Se il WS viene caricato prima dell'esecuzione si ha il prepaging (si evita cosi page fault)

Anomalia di Belady la frequenza di pagefault non diminuisce all aumentare dell'ampiezza della RAM (l'unico modo per evitare questa cosa è usare la politica LRU,ma la sua forma pura è irrealizzabile).

Nel *rimpiazzare la pagina* si può scegliere tra politiche:
- Locali,ossia rimpiazzo nel WS del processo che ha causa page fault.In tal caso ogni processo conserva una quota fissa in RAM.Ha prestazioni inferiori rispetto alle pol globali.
- Globali,ossia quando la scelta avviene tra page frame senza distinzione di processo.L'allocazione della ram varia in modo dinamico nel tempo.E' una politica più efficace.

Modello 2:Segmentazione

- Spazi di indirizzamentocompletamente indipendenti gli uni dagli altri per dimensione e posizione in RAM (possono variare dinamicamente)