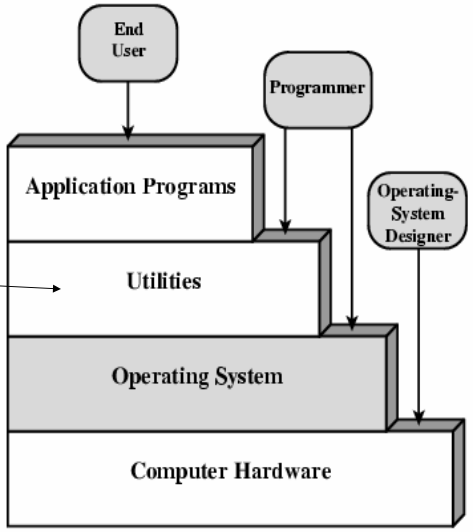
Il supporto al Sistema Operativo

Obiettivi e funzioni del S.O.

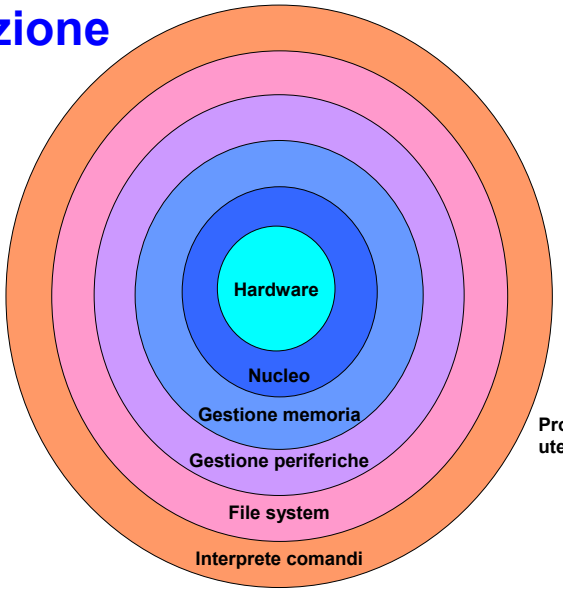
- Il Sistema Operativo è il software che controlla l'esecuzione dei programmi e amministra le risorse del sistema.
- Ha due obiettivi principali:
  - —Convenienza
    - – Facilitare l'uso del sistema di elaborazione
  - —Efficienza
    - – Ottimizzare l'uso delle risorse
- E' necessario un supporto hardware alle attività del S.O.

Strati e viste in un calcolatore

Compilatori,
linker,
debugger,

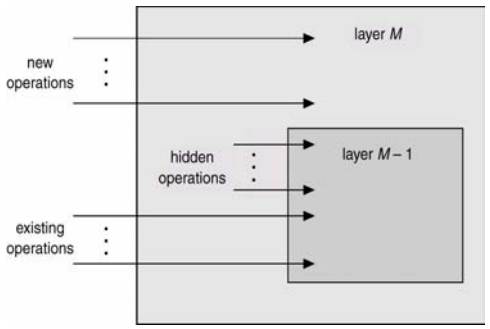


Organizzazione a livelli

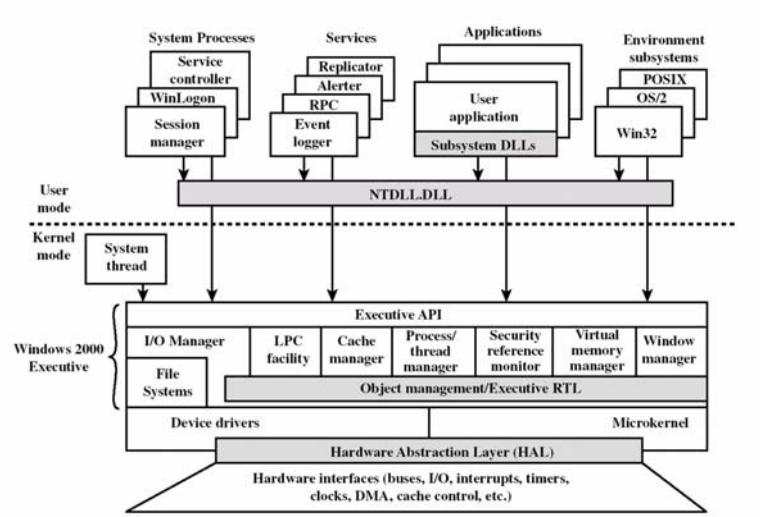


Funzioni dei livelli

- **Ogni livello realizza un certo sottoinsieme di funzioni**
- **Ogni livello realizza una *macchina virtuale*, che nasconde i meccanismi implementativi e offre un insieme ben definito di funzionalità ai livelli superiori**
- **In questo modo, ogni livello può essere modificato senza intervenire sugli altri livelli**



Struttura di Windows 2000/NT



Servizi forniti dal S.O.

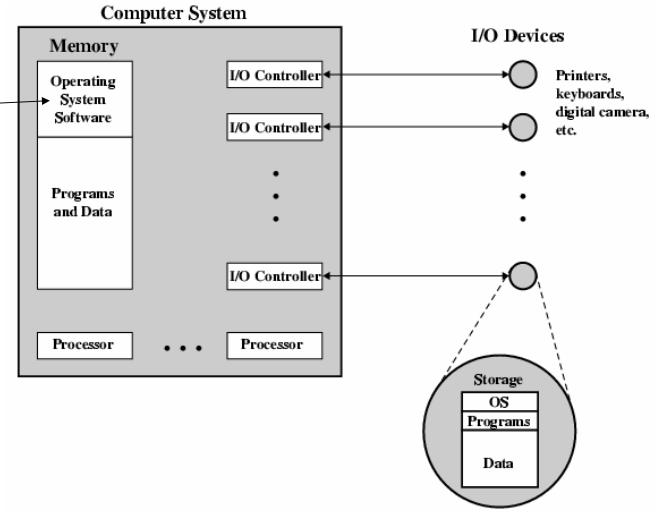
- Realizzazione di programmi
  - Programmi di utilità: editor, compilatori, debugger. Non fanno parte del S.O., ma vi interagiscono fortemente
- Esecuzione di programmi
  - Assegnazione di risorse; caricamento in M.C. del codice; inizializzazione dei devices e dei files; ecc.
- Accesso ai device di I/O
  - Virtualizzazione dei devices
- Accesso controllato ai files
  - Gestione dell'accesso; controllo dei permessi; gestione di conflitti in accesso; ecc.
- Accesso al sistema
  - Protezione da accessi non autorizzati; gestione delle contese sulle risorse; ecc.
- Individuazione e correzione di errori
- Accounting

S.O. come gestore di risorse

- Un compito fondamentale del S.O. è la gestione delle risorse del sistema:
  - Memoria centrale, dev. I/O, **CPU**
- Le risorse vanno distribuite tra i vari programmi che ne fanno richiesta
- **Paradosso**: il S.O. è esso stesso formato da un insieme di programmi
- Come fare a riacquisire la risorsa CPU, una volta che è stata ceduta ad altri ?
- Necessità di una soluzione hardware

S.O. come gestore di risorse

Kernel e altre porzioni del S.O. attive



Tipologie di S.O.

- Interattivi
  - —L'utente interagisce con il sistema
- Batch
  - —Blocchi di programmi che non richiedono interazione con l'utente
- A programma singolo (Monoprogrammati)
- Multiprogrammati (Multi-tasking)

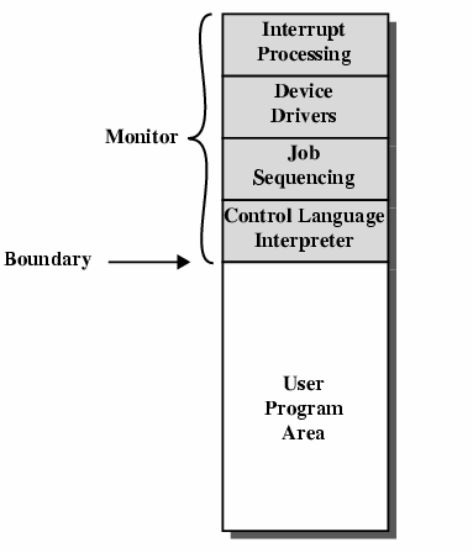
I primi sistemi

- Fine anni '40 – metà anni '50
- Assenza di S.O.
- I programmi interagiscono direttamente con l'hardware
- Due principali problemi:
  - —Scheduling
  - —Setup time

Semplici sistemi batch

- Programma "Monitor" residente in memoria
- Gli utenti forniscono dei programmi da compilare ed eseguire (job) ad un tecnico
- Il tecnico organizza i blocchi di jobs
- Il monitor controlla la sequenza di eventi (errori, fine ops. di I/O,…) per elaborare i batch
- Quando un job termina, il controllo ritorna al monitor che avvia il job successivo
- Il monitor gestisce la pianificazione (scheduling) dell'esecuzione dei job

Organizzazione in memoria per il monitor residente



Job Control Language

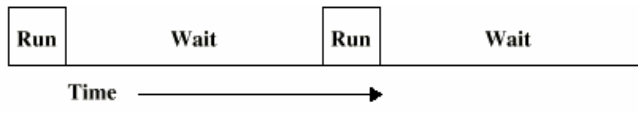
- Nel definire il job, l'utente specifica le istruzioni di setup al monitor
- Es
  - —$JOB
  - —$FTN ← Invocazione del compilatore FORTRAN
  - —... Istruzioni FORTRAN
  - —$LOAD ← Invocazione del loader
  - —$RUN
  - —... dati
  - —$END

Caratteristiche hardware necessarie

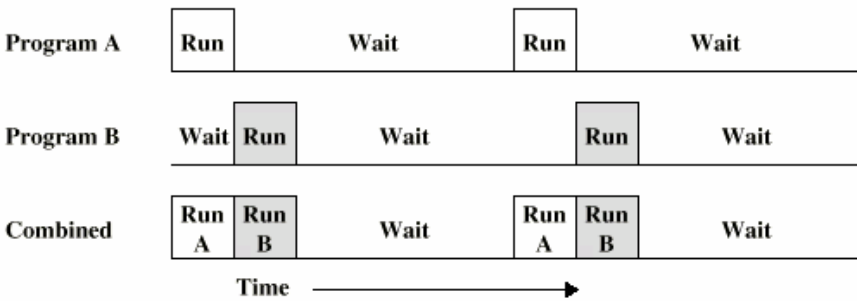
- Protezione sulla memoria
  - —Per proteggere il monitor
- Timer
  - —Per evitare che un job monopolizzi il sistema
- Istruzioni privilegiate
  - —Funzionamento del processore in due stati diversi
  - —Eseguite solo dal monitor
  - —Es. I/O
- Interruzioni
  - —Permettono di rilasciare e riguadagnare il controllo

Sistemi Batch Multiprogrammati

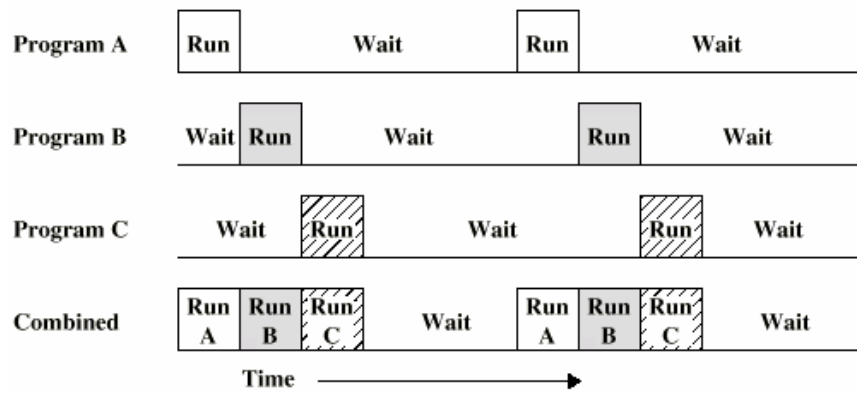
- Nel caso si esegua un solo batch alla volta (sistema monoprogrammato) ci possono essere lunghi intervalli di inattività legati all'attesa su un device di I/O
- Invece, mentre un job è in attesa di I/O, un altro potrebbe usare la CPU



Sistema Multiprogrammato (due programmi)



Sistema Multiprogrammato (tre programmi)

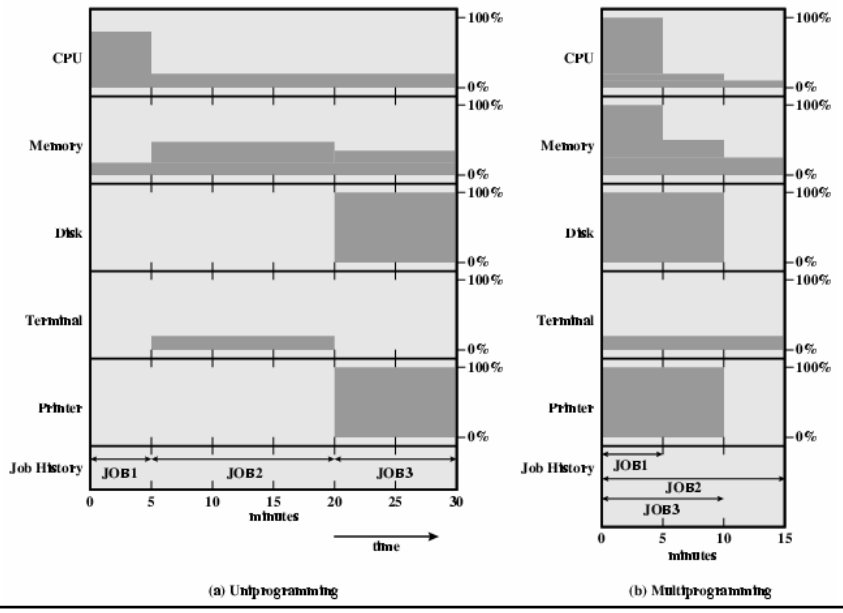


Esempio

- Supponiamo di avere 3 job con esigenze diverse:

	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 K	100 K	80 K
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes

Utilizzazione



Utilizzazione

	Uniprogramming	Multiprogramming
Processor use	22%	43%
Memory use	33%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min	15 min
Throughput rate	6 jobs/hr	12 jobs/hr
Mean response time	18 min	10 min

Sistemi time sharing

- Permettono un'interazione diretta dell'utente con il sistema
- Permettono la presenza e l'interazione di molti utenti con il sistema

	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

Scheduling

- Attività chiave per la multiprogrammazione
- Agisce sui "processi"
- Diversi tipi di scheduling:
  - —A lungo termine
  - —A medio termine
  - —A breve termine
  - — di I/O

Scheduler a lungo termine

- Determina quali programmi sono accettati per l'esecuzione
- In effetti, controlla il grado di multiprogrammazione
- Una volta accettato, un programma diventa un processo per lo scheduler a breve termine
- In alternativa, diventa un job "swapped out" per lo scheduler a medio termine

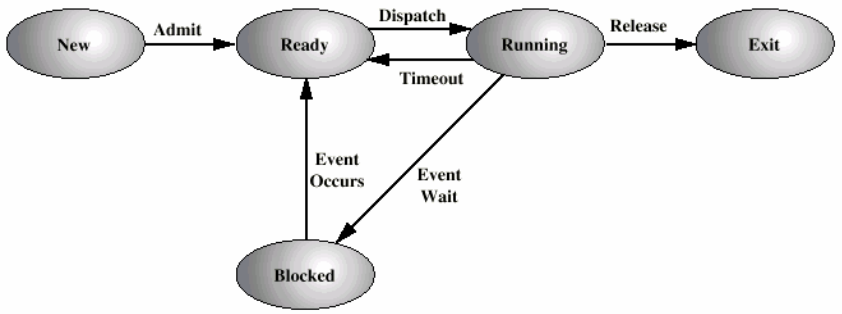
Scheduler a medio termine

- Fa parte della gestione dello "swapping"
- Fa fronte alla necessità di gestire il grado di multiprogrammazione
- Appoggiato al sistema di memoria virtuale

Scheduler a breve termine

- Noto anche come "Dispatcher"
- Decide quale processo deve essere eseguito, cioè quale processo utilizzerà la CPU nel prossimo quanto di tempo
- Nella decisione, impiega un algoritmo che garantisca un'efficienza complessiva
- Eventualmente, tiene conto delle priorità dei processi

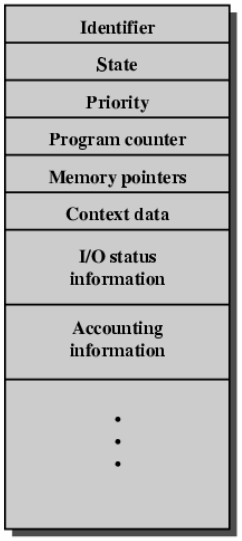
Stati di un processo

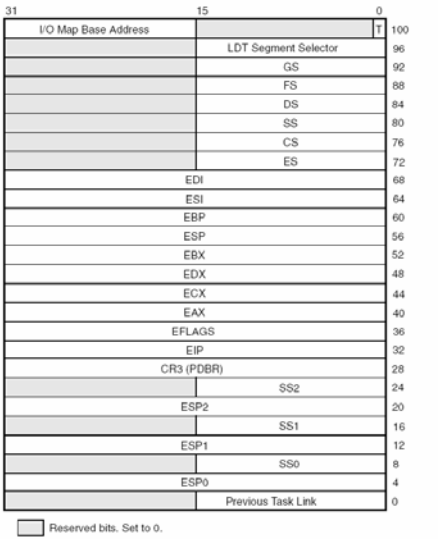


Struttura dati per la gestione dei processi (Process Control Block)

- Identificativo
- Stato
- Priorità
- Program counter
- Puntatori in memoria
  - —Indirizzi delle locazioni in memoria occupate
- Contesto
- Stato di I/O
- Informazioni di accounting

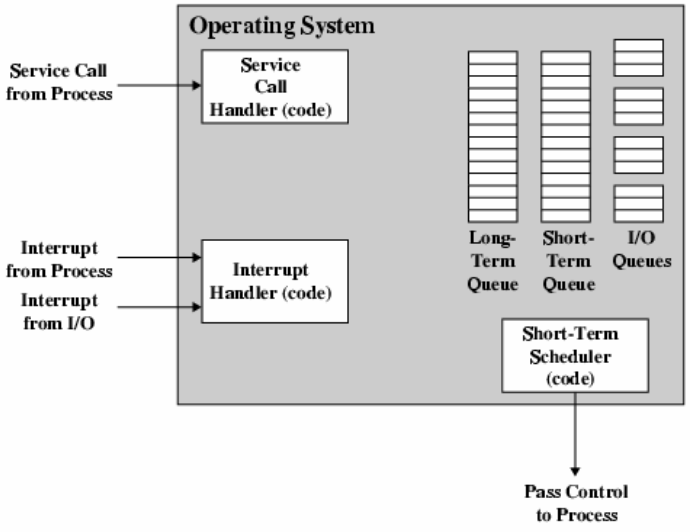
Diagramma PCB



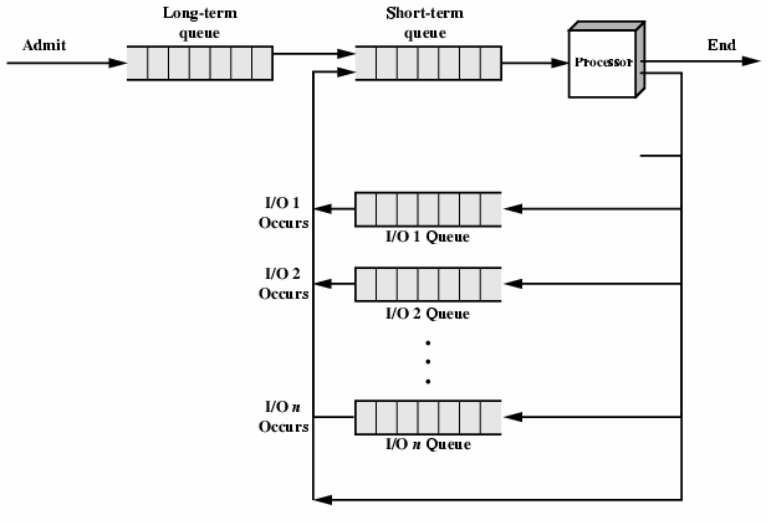


Intel Task Status Segment (TSS)

Elementi di base di un S.O. MProg.



Code gestite dallo scheduling



Gestione della memoria

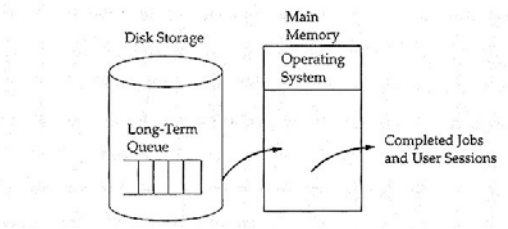
- Sistemi monoprogrammati
  - —La memoria è divisa in due:
    - Una parte per il Sistema Operativo (monitor)
    - Una parte per il programma corrente
- Sistemi multiprogrammati
  - —La parte "utente" è suddivisa tra i vari processi attivi
  - —Necessaria una gestione della memoria efficace:
    - L'obiettivo è di massimizzare il numero di processi attivi in memoria in modo da ottimizzare l'uso della CPU

Swapping

- Problema: la velocità della CPU in confronto ai dispositivi di I/O è tale che, anche nei sistemi multiprogrammati, la CPU può rimanere inoperosa per la maggior parte del tempo
- Soluzioni:
  - —Incrementare la memoria principale
    - – Costoso
    - – Inseguimento memoria/software
  - —Swapping

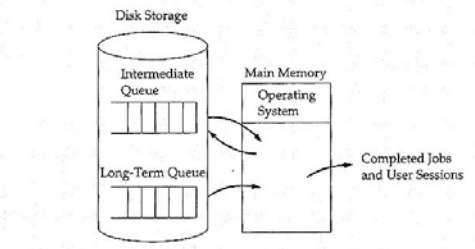
Swapping

- Nella organizzazione vista finora si sono considerate tre tipi di coda:
  - — Coda a lungo termine
  - — Coda a breve termine
  - — Coda di I/O
- La coda di lungo termine è allocata in memoria
- I processi sono trasferiti in memoria uno alla volta appena si libera spazio sufficiente
- Appena un processo termina è rimosso dalla memoria



Swapping

- Se nessuno dei processi in memoria è pronto (sono tutti in attesa di I/O) la CPU rimarrebbe in attesa
- Alternativa:
  - — Un processo in attesa di I/O è trasferito al di fuori della memoria (swapped out) su una coda intermedia allocata sul disco
  - — Viene trasferito in memoria (swapped in) un processo ready o un processo new e l'esecuzione continua con il nuovo arrivato



Swapping

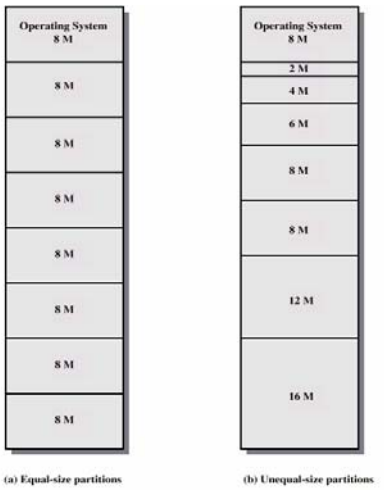
- Potenziale problema: lo swapping stesso è un'operazione di I/O
- Siccome il disco è in generale tra le periferiche più veloci del sistema, lo swapping di solito porta ad un effettivo miglioramento delle prestazioni
- Uno schema più sofisticato e più efficace richiede l'impiego della memoria virtuale (vd. oltre)
- Necessario introdurre prima i concetti di
  - Partizionamento
  - Paging

Partizionamento

- La memoria è sezionata in diverse parti, ognuna delle quali verrà occupata da un processo (compreso il Sistema Operativo)
- Il partizionamento può essere:
  - —a dimensione fissa
  - —a dimensione variabile

Partizionamento a dimensione fissa

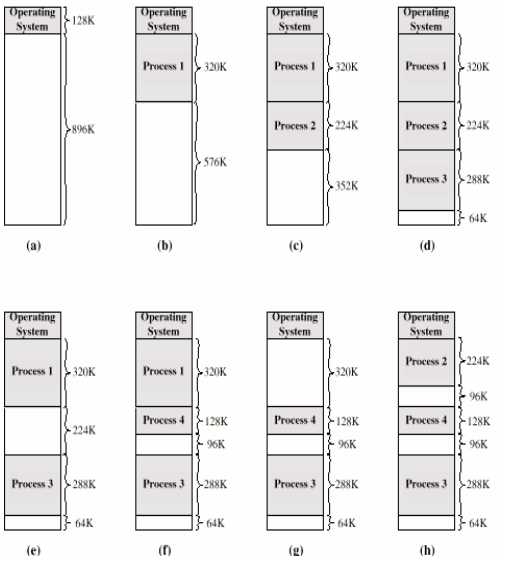
- Lo schema più semplice è dato dall'impiego di partizioni a dimensione fissa
- Le partizioni, sebbene di dimensione fissa, possono avere dimensioni diverse
- Quando un processo è trasferito in memoria, viene allocato sulla partizione più piccola in grado di contenerlo
- Possibili sprechi di memoria



Partizionamento a dimensione variabile

- Ad ogni processo trasferito in memoria viene allocato lo spazio strettamente necessario
- Dopo l'allocazione di un certo numero di processi, si creerà un "buco" nella parte terminale della memoria, troppo piccolo per essere utilizzato
- Quando tutti i processi sono bloccati, si trasferisce fuori dalla memoria un processo e se ne alloca un altro
- Il nuovo processo potrebbe essere più piccolo del processo swapped e quindi potrebbe crearsi un'altra zona di memoria libera ma inutilizzabile

Effetti del partizionamento a dimensione variabile



Effetti del partizionamento a dimensione variabile

- A regime si potrebbero avere molti buchi (frammentazione) con un conseguente utilizzo inefficiente della memoria
- Soluzioni:
  - —Buchi adiacenti vengono combinati in un unico spazio
  - —Compattazione: periodicamente il S.O. sposta i processi in memoria in modo da combinare tutti i buchi in un unico blocco di memoria utilizzabile (procedura dispendiosa in termini di tempo e di risorse di CPU)

Rilocazione dei processi

- Lo spazio di indirizzi assegnati ad un processo può essere modificato durante l'esecuzione (rilocazione)
  - —in seguito a swapping in
  - —in seguito a spostamenti per compattazione
- Le istruzioni del processo fanno riferimento a indirizzi di
  - —Dati (load, store)
  - —Istruzioni (salti)
- A causa della rilocazione, tali indirizzi possono non essere costanti durante l'esecuzione

Rilocazione dei processi

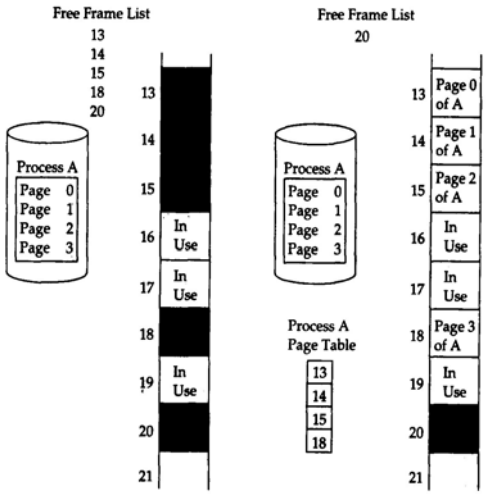
- Per risolvere il problema, vengono distinti due tipi di indirizzi
- **Indirizzo logico**
  - Locazione relativa all'inizio del programma (indirizzo prodotto dal traduttore)
- **Indirizzo fisico**
  - Locazione effettiva in memoria centrale
- Quando la CPU esegue un processo, realizza una conversione automatica tra indirizzo logico e indirizzo fisico, aggiungendo al primo l'indirizzo (fisico) corrente di inizio del programma (base address)
- Caratteristica hardware introdotta per venire incontro alle esigenze del Sistema Operativo

Paging

- Il partizionamento (sia a dimensione fissa che a dimensione variabile) non portano ad un uso efficiente della memoria
  - Motivo: il processo deve essere interamente allocato su una partizione
- Alternativa: dividere la memoria e lo spazio di memoria dei processi in blocchi relativamente piccoli, aventi una stessa dimensione fissa
- Definiamo
  - **Pagina**: un blocco di processo
  - **Frame**: un blocco di memoria
- In questo modo, l'allocazione di un processo in memoria centrale consiste nel disporre le pagine del processo su frames disponibili della memoria.
- Lo spazio di memoria inutilizzato è al più una frazione dell'ultimo frame

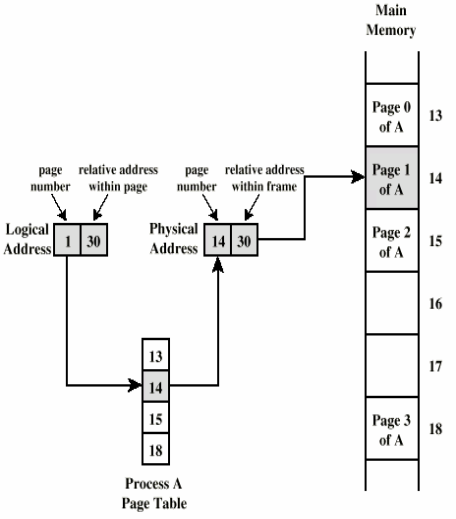
Paging

- Il S.O. mantiene una lista dei frames liberi
- Non è necessario che un processo ottenga frames contigui
- Per ogni processo, il S.O. mantiene una tabella delle pagine (page table)



Paging: Indirizzi logici e fisici

- Con il paging, l'indirizzo logico è formato dalla coppia
  (num. pagina, indir. relativo)
- La conversione ad indirizzo fisico è realizzata in hardware dalla CPU che deve poter accedere alla page table del processo
- In questo modo si risolvono i problemi di frammentazione



Paginazione a richiesta (Demand Paging)

- Un ulteriore miglioramento alla paginazione è introdotto con la paginazione a richiesta
  - Non vengono allocate in memoria tutte le pagine del processo
  - Le pagine vengono allocate su richiesta
- Motivazione: in un certo istante l'esecuzione sarà limitata a sezioni limitate di codice e di dati (principio di località)
- Il S.O. deve riservare uno spazio apposito su disco
- Si evita così di allocare pagine che non saranno utilizzate prima della sospensione del processo
- …e se la pagina richiesta non è in memoria ?

Paginazione a richiesta (Demand Paging)

- Se la pagina richiesta non è in memoria si ha un **fallimento di pagina** (**page fault**)
- Conseguenze:
  - Il S.O. deve caricare in memoria la pagina richiesta
  - Potrebbe essere necessario rimuovere una pagina allocata per fare spazio
  - Necessario definire una politica per scegliere la pagina da rimuovere

Thrashing

- Una cattiva scelta potrebbe far rimuovere una pagina che sarà richiesta subito dopo. Quando questo fenomeno diventa troppo frequente si instaura una condizione di **thrashing**.
- Di conseguenza, il S.O. spende il tempo a sua disposizione nello swap in/swap out di pagine con grosso spreco di risorse
- Problema:
  - Troppi processi allocati su una memoria centrale insufficiente
- Soluzioni:
  - Migliorare gli algoritmi di sostituzione
  - Ridurre il numero di processi attivi
  - Aumentare la memoria centrale

Memoria virtuale

- Non è necessario che un processo sia interamente in memoria per poter essere eseguito
- Le pagine vengono trasferite quando richieste
- Conseguenza: è possibile eseguire processi che richiedono uno spazio di memoria più ampio della memoria disponibile sul sistema
- Gli utenti/programmatori vedono una **memoria virtuale**, molto più ampia della memoria effettiva

Gestione delle tabelle di pagina

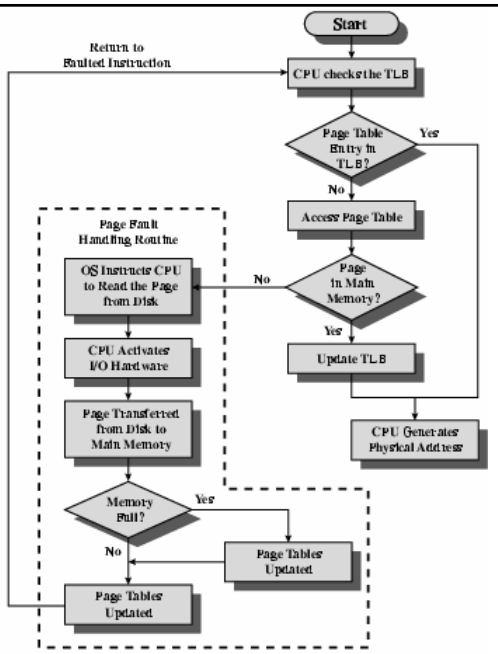
- Se lo spazio di memoria del processo è grande, la sua page table potrebbe essere molto lunga e richiedere diverse pagine per essere allocata.
- Nello stesso istante, poi, possono essere attivi molti processi
- Lo spazio di memoria richiesto dalle tabelle di pagina può essere enorme
- Soluzione: utilizzare la memoria virtuale per allocare le tabelle di pagina e mantenere in memoria solo le parti necessarie

Gestione delle tabelle di pagina

- Ogni indirizzamento in memoria richiede due accessi :
  - —Prelevare l'entry della tabella di pagina
  - —Prelevare il dato o istruzione
- Possibile un raddoppio dei tempi di accesso in memoria
- Soluzione: si impiega una cache speciale per le entry delle tabelle di pagina
  - —Translation Lookaside Buffer (TLB)

Operazioni TLB

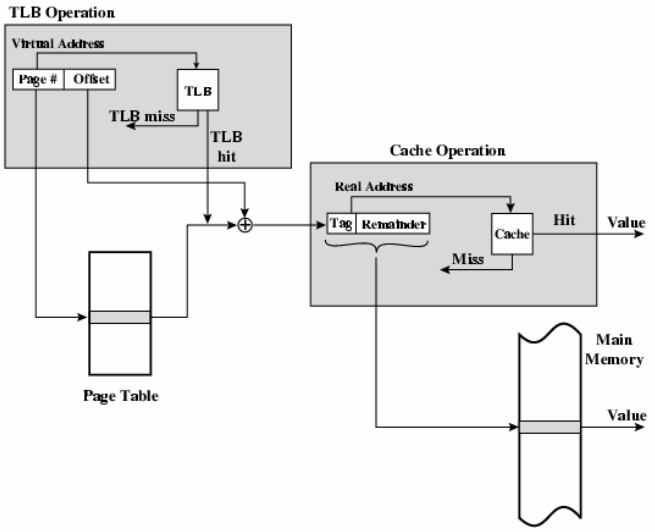
- L'adozione del TLB permette di migliorare sensibilmente le prestazioni di memoria virtuale



Interazione tra TLB e cache

- A valle della conversione dell'indirizzo logico in indirizzo fisico si inizia l'accesso alla memoria reale
- A questo punto si attraversa la gerarchia di memoria vista in precedenza

Interazione tra TLB e cache



Segmentazione

- Il paging di solito non è visibile al programmatore
- La Segmentazione è un'organizzazione dell'indirizzamento offerta dalla CPU simile al paging (indirizzo= ind.base segmento+offset) utilizzabile dal programmatore
- Di solito si impiegano segmenti diversi per
  - Dati
  - Codice
  - stack
- Vantaggi: flessibilità nella gestione dati e codice, protezione
- Svantaggi: meccanismo di indirizzamento complesso

Gestione della memoria virtuale del Pentium II