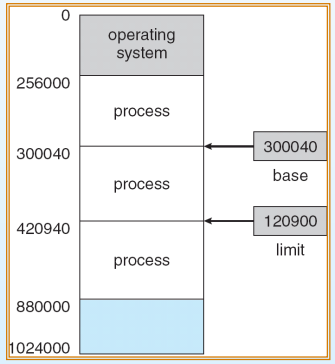
Gestione della Memoria

Il contesto

- Per essere eseguiti, i programmi devono essere trasferiti (dal disco) in memoria e tradotti in processi.
- Obiettivi: accesso rapido e protezione.
- Accesso rapido
  - La CPU ha accesso diretto solo alla memoria (RAM o una sua cache) e ai registri
  - L'accesso ai registri richiede in genere al massimo un ciclo di CPU clock.
  - L'accesso alla memoria può durare diversi cicli di CPU
  - **Cache** tra memoria centrale e registri della CPU
- Protezione
  - Bisogna assicurare che i processi non interferiscano provocando strani effetti.

Reg. di Base e reg. Limite

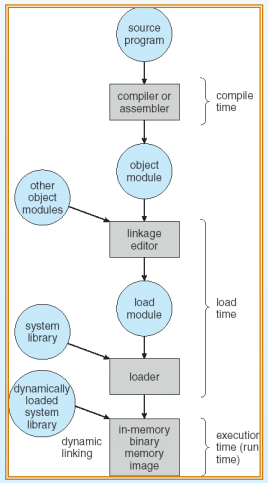
- Una coppia di registri, **base** e **limite**, definisce lo spazio degli indirizzi logici di un processo, per limitarne l'accesso.



Associazione tra dati/istruzioni e memoria

- L'associazione (binding) di istruzioni e dati agli appropriati indirizzi di memoria può avvenire a:
  - **Compile time**: se gli indirizzi di memoria sono noti a priori, viene generato **codice assoluto**; I programmi vanno ricompilati se devono essere eseguiti in un'altra posizione.
  - **Load time**: Si genera **codice rilocabile** se la posizione in memoria non è nota al tempo di compilazione
  - **Execution time**: L'associazione è ritardata fino al tempo in cui il programma viene effettivamente eseguito. Il processo può essere spostato durante la sua esecuzione. Necessita in genere di dispositivi hardware di supporto per la creazione degli indirizzi.

Programmi Utente



Caricamento Dinamico

- Es. Subroutine caricate al tempo della prima chiamata
- Migliore utilizzo della memoria; routine che non vengono usate non vengono mai caricate in memoria
- Utile quando parti cospicue del codice sono necessarie per gestire situazioni che occorrono raramente
- Implementato dal programmatore

Collegamento Dinamico

- Linking fatto durante l'esecuzione
- Piccole parti di codice, *stub (maniglie)*, usate per identificare l'approriata libreria di procedure residente in memoria
- Stub viene sostituta a tempo di esecuzione dall'indirizzo della procedura opportuna
- Il SO controlla che la procedura appartenga allo spazio degli indirizzi del processo
- Il linking dinamico è particolarmente utile per gestire librerie di procedure/funzioni
  - **shared libraries (librerie condivise)**

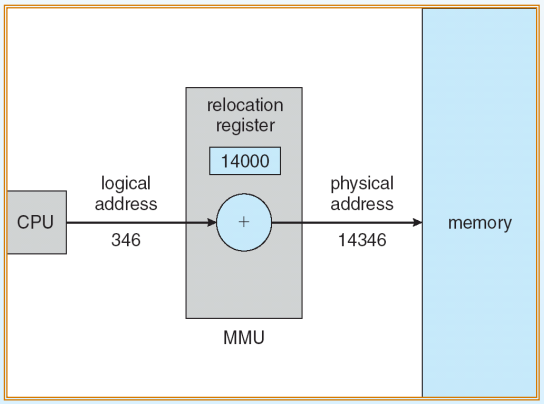
Spazio degli indirizzi: Logico e Fisico

- **Spazio degli Indirizzi Logici** – insieme degli indirizzi logici generati dalla CPU; anche detti **indirizzi virtuali**
- **Spazio degli Indirizzi Fisici** – insieme degli indirizzi fisici usati dal processo, e quindi usati dall'unità di memoria.
- Indirizzi fisici e logici coincidono nei casi di associazione a compile-time o load-time;
- Indirizzi fisici e logici differiscono nel caso di binding o associazione ad execution-time

Memory-Management Unit (MMU)

- Dispositivo hardware che traduce indirizzi logici in indirizzi fisici
- In un semplice schema di traduzione mediante MMU, ad ogni indirizzo logico viene sommato il valore presente in un registro di rilocazione prima che l'indirizzo venga inviato alla memoria
- Il programma utente vede solo indirizzi *logici*; non si occupa degli indirizzi *reali (fisici)*

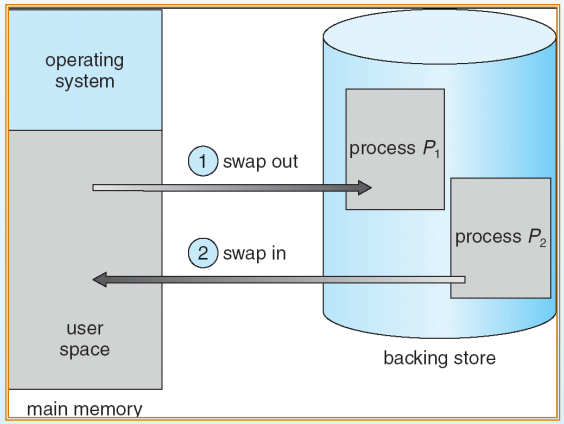
Rilocazione dinamica: registro di rilocazione



Swapping

- Un processo può essere rimosso (swapped out) dalla memoria temporaneamente e poi riportato in memoria.
- **Backing store** – tipicamente, unità a dischi sufficiente a contenere l'immagine di memoria dei processi utente; deve fornire accesso diretto
- Tempo di swap: tempo di trasferimento, direttamente proporzionale alla dimensione dell'immagine trasferita
- Il sistema mantiene una coda dei processi pronti all'esecuzione la cui immagine di memoria è su disco

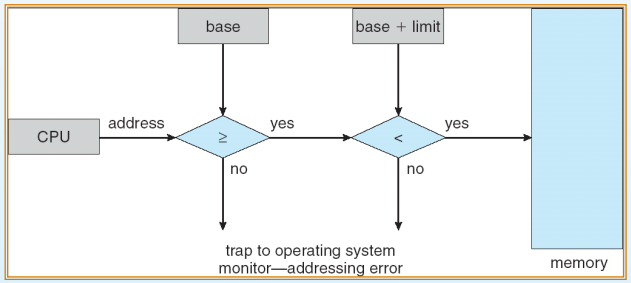
Schema di Swapping



Allocazione Contigua

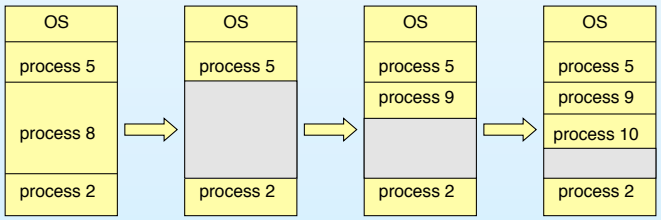
- Memoria principale divisa in:
  - SO residente, generalmente negli indirizzi bassi insieme al vettore degli interrupt
  - Processi utente mantenuti in memoria alta
- Registri di rilocazione usati per ottenere protezione dello spazio degli indirizzi dei processi utente e del SO
  - Registro base: valore del più piccolo indirizzo
  - Registro limite: taglia dello spazio degli indirizzi – ogni indirizzo logico deve essere minore del valore in tale registro
  - MMU traduce indirizzi logici dinamicamente

Protezione HW mediante registri base e limite



Allocazione Contigua (Cont.)

- Allocazione a partizioni multiple
  - blocchi di memoria disponibile di diversa dimensione sparsi in memoria
  - All' arrivo di un processo, gli si alloca memoria da un blocco sufficientemente grande
  - SO gestisce strutture dati per ricordare:
    a) partizioni allocate b) partitioni libere



Politiche di Allocazione dinamica

Come scegliere il blocco usare per un richiesta di dimensione *n*

- **First-fit**: Alloca il *primo* blocco sufficiente
- **Best-fit**: Alloca il più piccolo blocco di taglia non minore di n
  - Riduce la dimensione dello spazio rimanente
  - Ricerca tra i blocchi liberi
- **Worst-fit**: Alloca il blocco *più grande*;
  - Massimizza lo spazio rimanente in un blocco allocato

Statisticamente, First-fit fornisce prestazioni migliori

Frammentazione

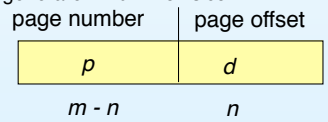
- **Frammentazione esterna** – molti blocchi liberi ma troppo piccoli
- **Frammentazione interna** – causata da allocazione di blocchi più grandi del necessario (es. sistemi a partizionamento prefissato)
- Per ovviare alla frammentazione esterna → **compattazione**
  - Gli spazi di memoria allocati vengono resi contigui in modo da riunire i blocchi liberi in un unico blocco
  - E' necessario che la rilocazione sia dinamica e fatta a tempo di esecuzione
  - Problemi possibili: I/O
    - Spostamento mentre il processo attende I/O in un fissato buffer
    - fare I/O solo usando buffer di sistema

Paginazione

- Spazio logico allocato in maniera non contigua
- Memoria fisica divisa in blocchi di dimensione fissa detti **frames** (tipicamente la taglia è una potenza di 2, tra 512 bytes e 16 Mb)
- Memoria logica divisa in blocchi della stessa dimensione detti **pagine**
- Il sistema tiene traccia dei frame liberi
- Per un programma con ***n*** pagine, bisogna trovare *n* frame liberi
- Uso di una "***tavola delle pagine***" per ricordare l'associazione frame-pagine
- Frammentazione Interna possibile
  - Solo sull'ultimo frame

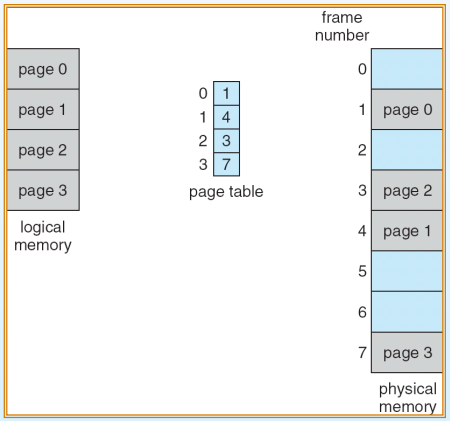
Schema di traduzione degli indirizzi

- Indirizzo generato dalla CPU viene diviso in:
  - **Numero di Pagina: *p*** – usato come valore indice in una tavola delle pagine che contiene l'idirizzo di base di ogni pagina in memoria
  - **Scostamento (Page offset): d** – viene aggiunto all'indirizzo base per generare l'indirizzo fisico

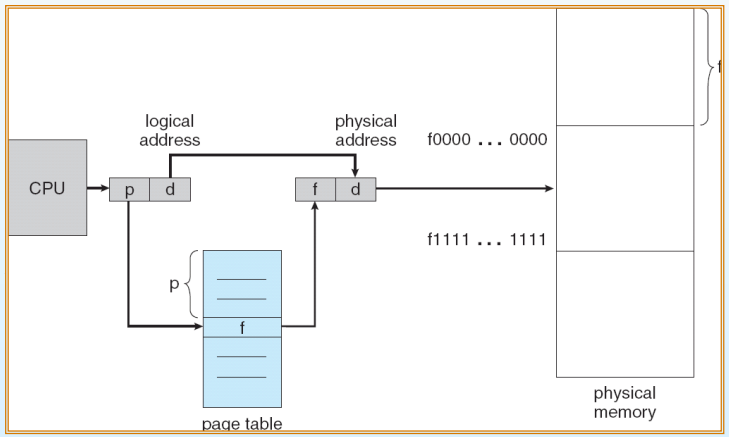


  - Spazio logico di indirizzamento 2^m e pagine di dimensione 2^n

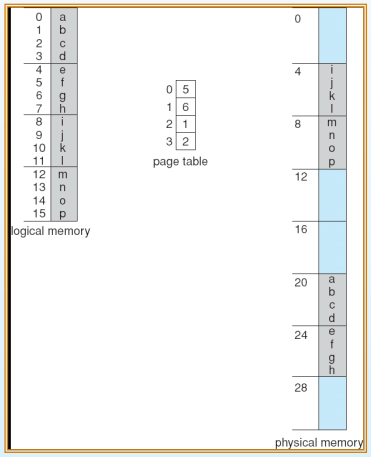
Paginazione: Indirizzi logici e fisici



Hardware per la Paginazione



Paginazione: Esempio



Memoria: 32-byte, pagina: 4-byte pages

Indirizzo logico di *j* : 1001

#pagina	scostamento
10	01
2 bit	2 bit

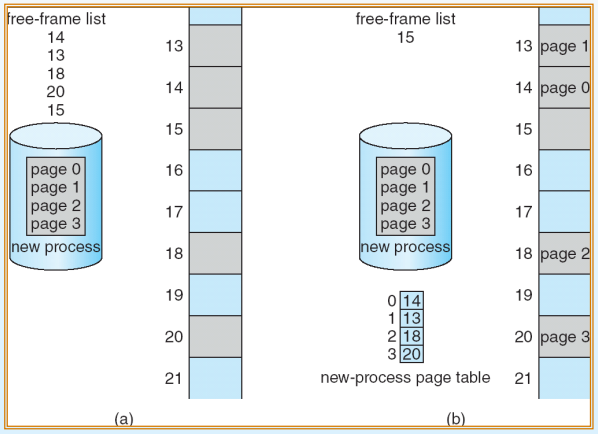
Pagina: 2, scostamento: 1

Frame: 1, scostamento: 1

#frame	scostamento
001	01
3 bit	2 bit

Indirizzo fisico di *j* : 00101

Frame Liberi



pre-allocazione

post-allocazione

Tavola delle Pagine: implementazione

- Tavola delle pagine mantenuta in memoria
- **Page-table base register (PTBR)**
- **Page-table length register (PRLR)** : taglia della tavola
- In tale schema ogni indirizzamento richiede due accessi in memoria: alla tavola e poi al dato vero e proprio
- Soluzione piu' veloce: fast-lookup hardware cache detto **associative memory** or **translation look-aside buffers (TLBs)**
  - Memorizza un **address-space identifier (ASID)** in ogni riga della TLB – per identificazione univoca dei processi e protezione dello spazio degli indirizzi

Memoria Associativa

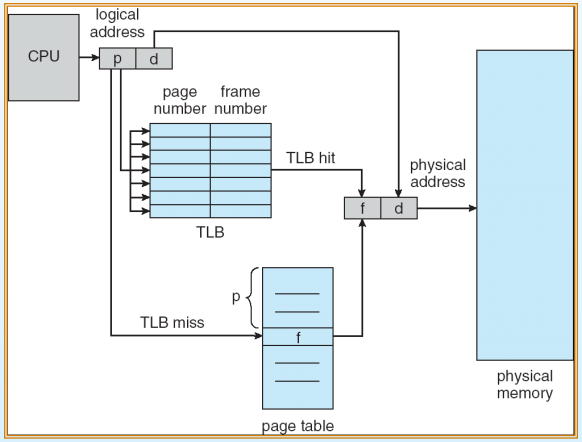
- Associative memory - TLB *(Translation look-aside buffer)*
  - ricerca in parallelo

Page #	Frame #

Traduzione degli indirizzi (p, d)

- Se p e' presente nella TLB, ottieni #frame in output
- Altrimenti il # frame va ricercato nella tavola delle pagine in memoria

Paginazione: Hardware con TLB



Tempo di Accesso Effettivo

- Associative Lookup = ε unità di tempo
- Hit ratio = α (probabilità che il frame è in TLB)
- **Tempo effettivo di accesso** (EAT)

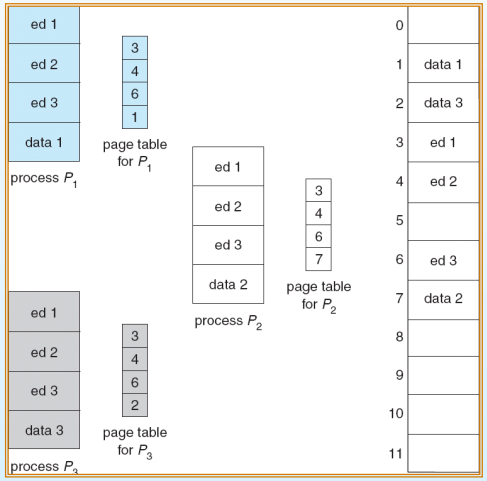
$$EAT = (1 + \varepsilon)\,\alpha + (2 + \varepsilon)(1 - \alpha)$$
$$= 2 + \varepsilon - \alpha$$

Assumendo tempo *trascurabile* per inserire in TLB la nuova pagina

Pagine Condivise

- **Shared code**
  - una copia read-only condivisa da piu' processi (es. text editors, compilatori, etc.).
  - Il codice condiviso appare nello stesso spazio logico di tutti i processi che lo condividono
- **Dati/codice esclusivi**
  - Ogni processo ha una copia privata (controllata)
  - Le pagine esclusive possono essere in posizioni qualsiasi dello spazio logico

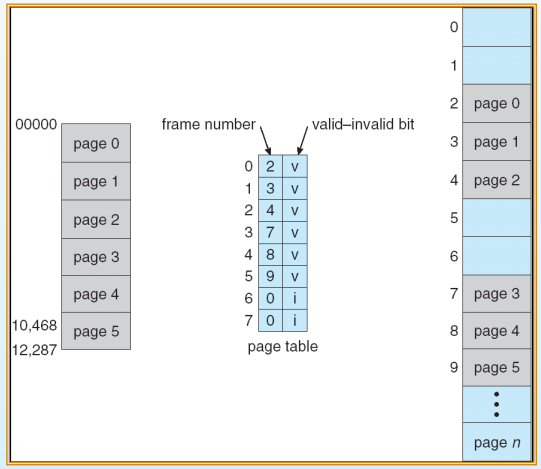
Pagine Condivise: Esempio



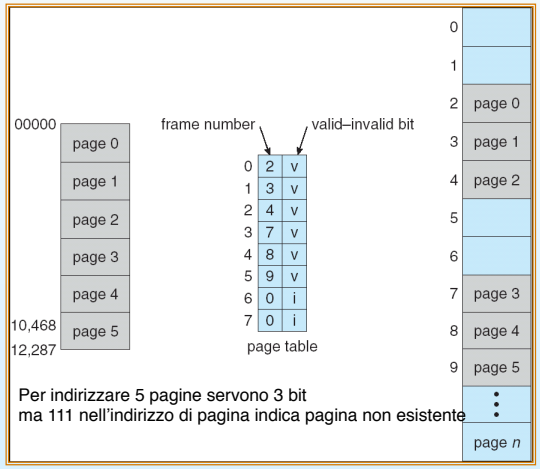
Protezione della memoria

- La protezione puo' essere implementato mediante un bit di protezione associato ad ogni frame
- **Bit di Validita'** per ogni elemento della tavola delle pagine:
  - "V" per indicare che la pagina associata e' nello spazio degli indirizzi logici del processo
  - "I" indica che la pagina non fa parte dello spazio degli indirizzi logici del processo

Bit di validita' (v/i) nella Page Table



Bit di validita' (v/i) nella Page Table



Struttura della Tabella delle Pagine

- Paginazione gerarchica
- Tabelle delle pagine con hashing
- Tabella delle pagine Invertita

Problema: Tavola delle Pagine troppo grande

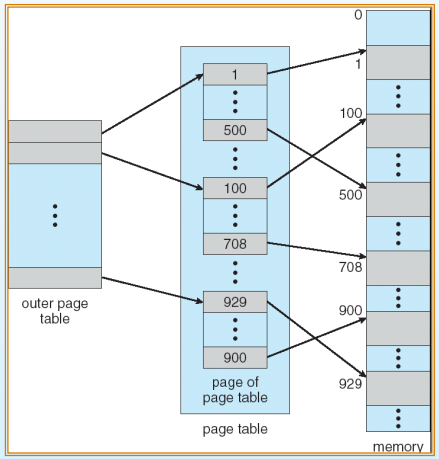
- 32 bit di indirizzamento logico
- Pagine di 4 Kb → scostamento 12 bit
- 32 – 12 = 20 bit per il numero di pagina
  - → la tavola delle pagine contiene piu' di un milione di elementi!!

    ($2^{20} > 1000000$)

Paginazione gerarchica

- Lo spazio di indirizzamento logico viene frazionato in tabelle multiple
- Caso semplice: tabella delle pagine a due livelli

Paginazione a due livelli



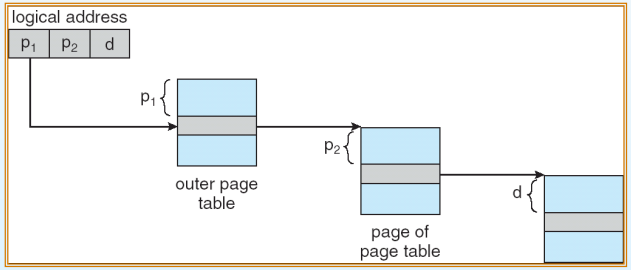
Paginazione a due livelli: Esempio

- un indirizzo logico (su una macchina a 32-bit co pagine di 4K) viene diviso in:
  - Un numero di pagina a 20 bits
  - Uno scostamento (offset) a 12 bits
- La tabella delle pagine e' a sua volta paginata
  - Il numero di pagine e' suddiviso in:
    - Un numero di pagina a 10-bit
    - Un offset a 10-bit
- Indirizzo logico:

page number		page offset
p_i	p_2	d
10	10	12

dove p_i e' l'indice nella tabella delle pagine di primo livello, e p_2 e l'offset nella pagina della tavola delle pagine di secondo livello

Schema di traduzione



Paginazione a 3 livelli

outer page	inner page	offset
p_1	p_2	d
42	10	12

2nd outer page	outer page	inner page	offset
p_1	p_2	p_3	d
32	10	10	12

Tabelle delle Pagine di tipo Hash

- Comune quando lo spazio degli indirizzi > 32 bit
- Il numero di pagina logico viene codificato mediante una funzione hash in una tavola delle pagine. Hashing concatenato
- Ricerca di match
  - L'indirizzo del frame corrispondente viene prelevato

Tabella delle Pagine di tipo Hash

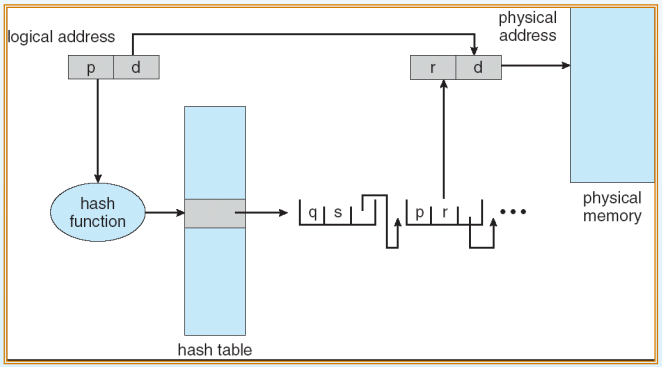
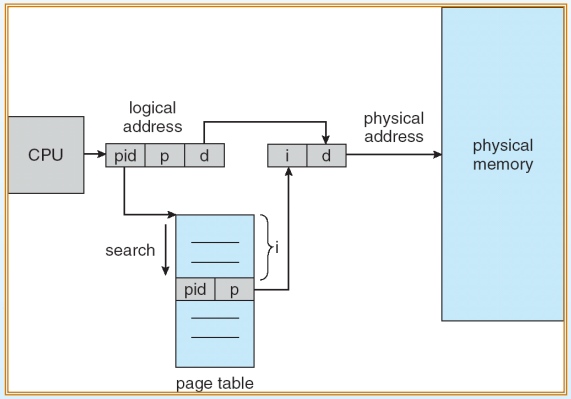


Tabella delle Pagine Invertita

- Un elemento per ogni frame
- Ogni elemento contiene l'indirizzo logico della pagina memorizzata in quel frame, ed informazioni sul processo proprietario
  - Riduce la quantita' di memoria necessaria a memorizzare la tabella delle pagine,
  - Aumenta il tempo di ricerca
- Si puo' usare hashing per ridurre il tempo di ricerca

Architettura del modello a tabella delle pagine invertita

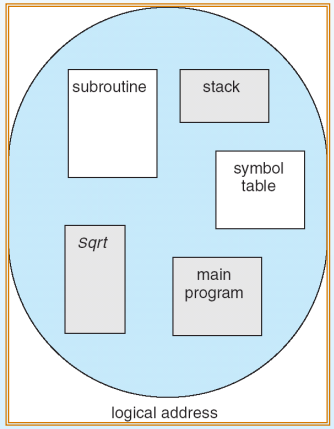


Segmentazione

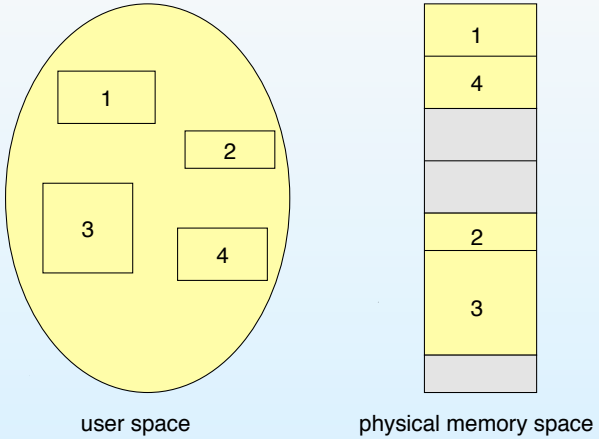
- Schema di gestione della memoria che implementa la prospettiva dell'utente sullo spazio di memoria
- Un programma è un insieme di segmenti. Ogni segmento rappresenta un'unità logica, tipo:

  main program,
  procedure,
  funzioni,
  metodi,
  oggetti,
  variabili locali e globali
  stack,
  tabella dei simboli, array

Programmi: Prospettiva dell' Utente



Schema logico della segmentazione



Architettura per Segmentazione

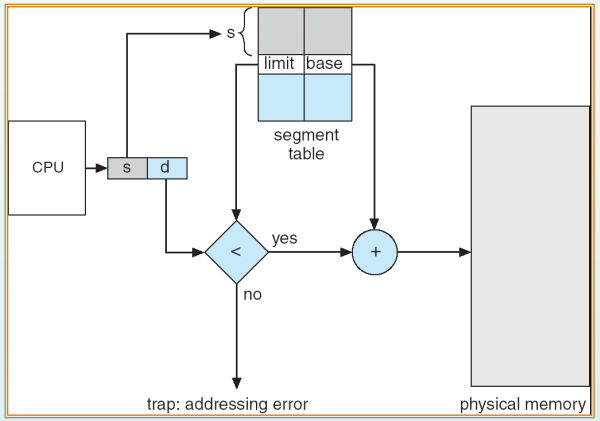
- Indirizzo logico consiste di una *coppia ordinata di valori*:
  <num-segmento, offset>,
- **Tabella dei Segmenti** – ogni elemento consiste di:
  - **base** – l' indirizzo fisico iniziale del segmento
  - **limite** – la lunghezza del segmento
- **Segment-table base register (STBR)** punta alla locazione di memoria dove risiede la tabella dei segmenti
- **Segment-table length register (STLR)** indica il numero di segmenti usati dal programma;
  num-segmento ***s*** è legale se ***s*** < **STLR**

<s, o> è legale se ***s*** < ***STLR*** **e** ***o*** **<** ***limite***

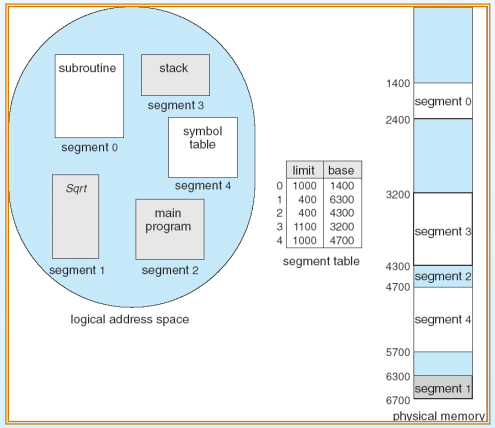
Architettura per Segmentazione

- Protezione
  - Ad ogni elemento della tabella si associa:
    - Bit di validazione = 0 ⇒ segmento illegale
    - Privilegi: read/write/execute
- Bit di protezione associato ai segmenti
  - sharing a livello di segmenti
- Segmenti di lunghezza variabile
  - Allocazione con i problemi dell'allocazione dinamica

Hardware per Segmentazione

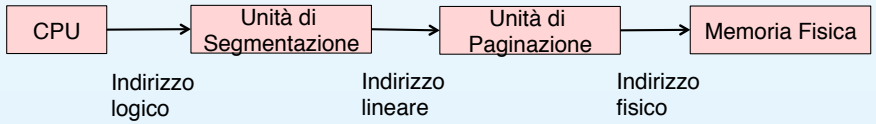


Esempio di Segmentazione



Pentium: Segmentazione e Paginazione

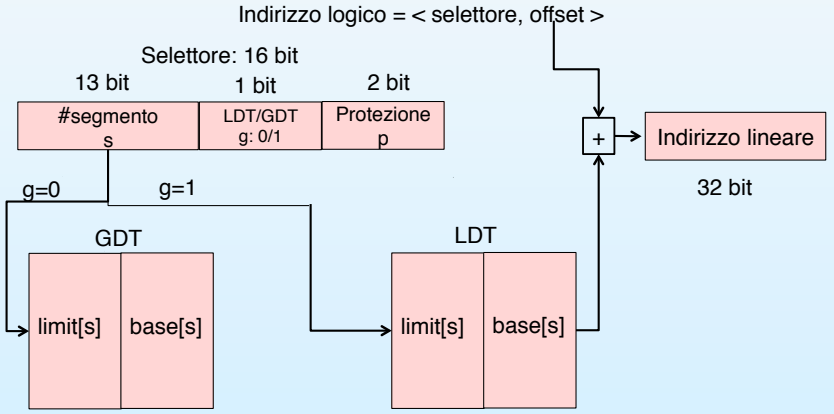
I due modelli vengono combinati

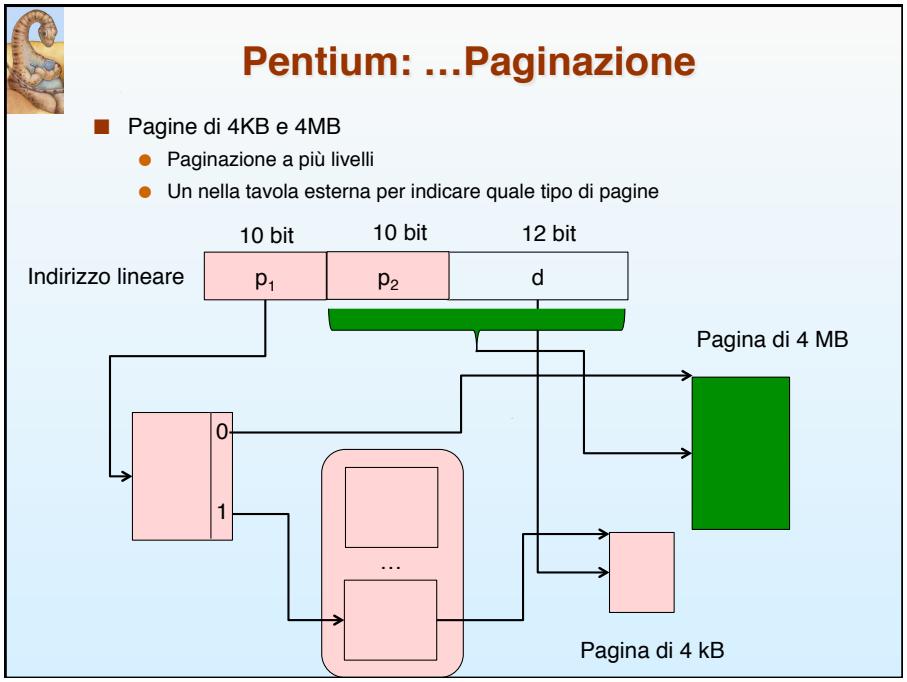


Ogni processo ha a disposizione fino a 2^{14} possibili segmenti
- 2^{13} segmenti privati
- 2^{13} segmenti condivisi

Pentium: Segmentazione …

- Un processo può usare
  - 2^{13} segmenti privati (LDT – local descriptor table)
  - 2^{13} segmenti condivisi tra tutti I processi (GDT – global descriptor table)




Pentium: …Paginazione
Pagine di 4KB e 4MB
Paginazione a più livelli
Un nella tavola esterna per indicare quale tipo di pagine
10 bit
10 bit
12 bit
Indirizzo lineare
p_1
p_2
d
Pagina di 4 MB
0
1
…
Pagina di 4 kB