Gerarchie di memoria
Memorie a componenti dinamici 1

- Denominate **Dynamic RAM**
- Memorizzazione su componenti capacitivi
  - 1 transistor per bit (4-6 per Static RAM), dunque maggiore densità
- Richiede un ciclo di *refresh* ogni 8-10 ms.
  - Non richiesto per Static RAM

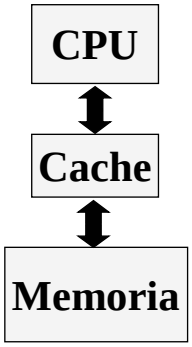
Gerarchie di memoria
Memorie a componenti dinamici 2

- Organizzazione interna a matrice bidimensionale con indirizzamento in due fasi sullo *stesso* canale di accesso
  - Per riga (*Row Access Strobe*)
  - Per colonna (*Column Access Strobe*)
- Maggiore capacita' ma tempo di accesso maggiore delle Static RAM
  - Dovuto alla diversa modalità di indirizzamento

Gerarchie di memoria
Memorie a componenti dinamici 2

Anno	Capacità	Ciclo di lettura
1980	64 KB	250 ns.
1983	256 KB	220 ns.
1986	1 MB	190 ns.
1989	4 MB	165 ns.
1992	16 MB	145 ns.
1995	64 MB	120 ns.

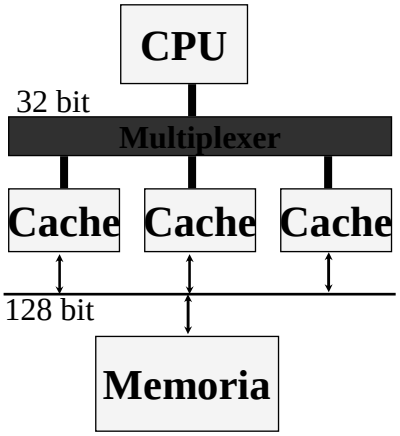
Gerarchie di memoria
Memoria principale 1



Organizzazione diretta

Il bus tra CPU e cache ha la stessa dimensione di quello tra cache e memoria principale

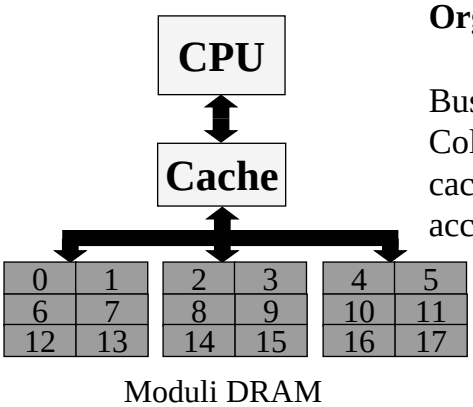
Gerarchie di memoria
Memoria principale 2



Organizzazione estesa

Un instradatore tra CPU ed N cache.
Bus più ampio tra le cache e la memoria principale

Gerarchie di memoria
Memoria principale 3

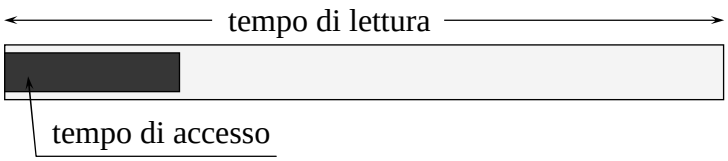


Organizzazione interallacciata

Bus semplice tra CPU e cache.
Collegamento interallacciato tra cache e moduli DRAM con accessi simultanei

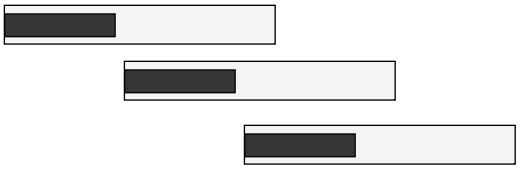
Gerarchie di memoria
Tecniche d'uso di DRAM 1

- Tempo di accesso: tempo necessario alla memoria per accedere alla cella indirizzata
- Tempo di lettura: tempo necessario alla CPU per acquisire un dato dal DB, dall'emissione dell'indirizzo in AB



Gerarchie di memoria
Tecniche d'uso di DRAM 2

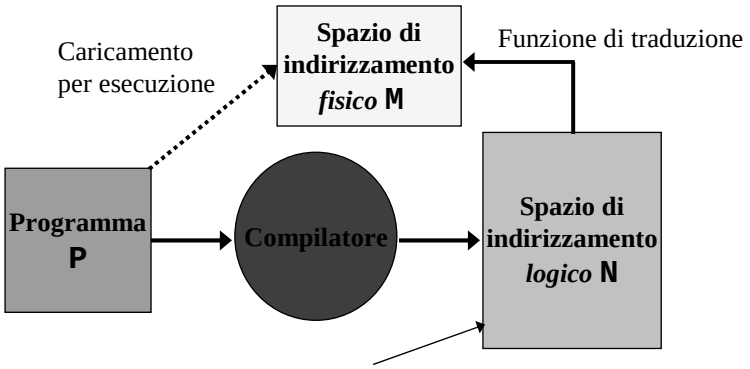
- Memorie interallacciate (*interleaving*): Accesso parallelo in memoria *senza* attendere la conclusione della lettura precedente



Memoria virtuale
Generalità

- Il concetto di *memoria virtuale* si propone di incrementare l'efficienza d'uso di un sistema, ampliandone, a costi contenuti, la capacità di memoria
  - Permettendo la presenza in memoria di **più** programmi simultaneamente
  - Permettendo l'esecuzione di programmi di dimensioni **maggiori** dell'ampiezza della memoria principale
  - Consentendo un utilizzo più avanzato dei vari livelli di memoria disponibili

Memoria virtuale
Spazi di indirizzamento - 1



L'insieme di tutti gli indirizzi, noti a tempo di compilazione, che P potrà generare a tempo di esecuzione

Memoria virtuale
Spazi di indirizzamento - 2

- A tempo di esecuzione di **P**, il processore genera indirizzi **logici** appartenenti allo spazio **N** assegnato dal compilatore al programma
- **N** è ampio al più 2^p indirizzi, dove p è l'ampiezza in bit dell'indirizzo **logico**
- La memoria principale offre uno spazio di indirizzamento **fisico** **M** al programma caricato per l'esecuzione
- L'ampiezza di **M** **non coincide** necessariamente con quella di **N**

Memoria virtuale
Spazi di indirizzamento - 3

- Nel caso di allocazione **statica**, **N** ed **M** coincidono
  - Una traduzione statica può essere necessaria se **N** ed **M** usano una **base diversa**
  - Tale traduzione avviene nella fase di caricamento (*linking loader*)
- Nel caso di allocazione **dinamica**, **M** varia sia in ampiezza che in posizionamento
  - Ciò richiede l'uso di una **funzione di rilocazione** $f : \mathbf{N} \rightarrow \mathbf{M}$ di cui si occupa il gestore della memoria

Memoria virtuale
Allocazione dinamica - 1

- La funzione di rilocazione f può utilizzare
  - **Paginazione**
    decomponendo sia **N** che **M** in blocchi di ampiezza **fissa** e contenenti informazioni **contigue** (**pagine**)
  - **Segmentazione**
    decomponendo **N** ed **M** in aree di ampiezza **variabile**, contenenti informazioni **contigue** e corrispondenti a **specifiche** entità del programma (**segmenti**)
- La funzione f viene valutata per ogni indirizzo logico generato dal processore per il programma in esecuzione

Memoria virtuale
Allocazione dinamica - 2

- L'allocazione statica o dinamica dello spazio *fisico* del programma è **altra cosa** rispetto all'allocazione delle informazioni nello spazio *logico* del programma
  - Il programmatore può operare sulla seconda mediante costrutti appositi del linguaggio
- Entrambe possono richiedere supporto di sistema operativo

Memoria virtuale
Gestione

- La gestione della memoria virtuale è assai simile a quella della cache
  - Memoria **principale** (livello inferiore) più veloce e costosa; memoria **secondaria** (livello superiore) molto piu estesa, lenta ed economica
  - Suddivisione in blocchi (pagine o segmenti)
  - *Fault di memoria* se il dato richiesto non è presente in memoria principale
  - Prelievo del blocco contenente il dato dal livello superiore ed allocazione (con rimpiazzo) al livello inferiore

Memoria virtuale
Parallelo cache - memoria virtuale

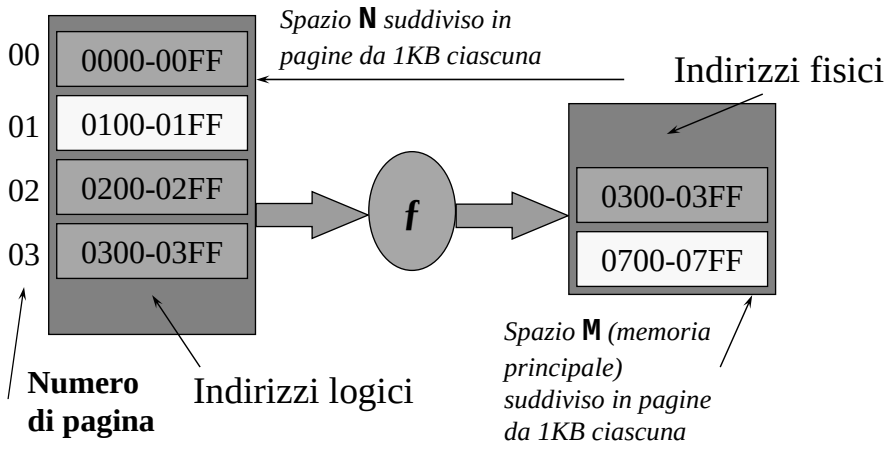
La cache *differisce* dalla memoria virtuale per:

- La *dimensione dei blocchi*, molto inferiore per la cache
- La *gestione*, che, per la cache, viene risolta interamente a **livello hardware**, mentre per la memoria virtuale richiede l'intervento del **livello sistema operativo**

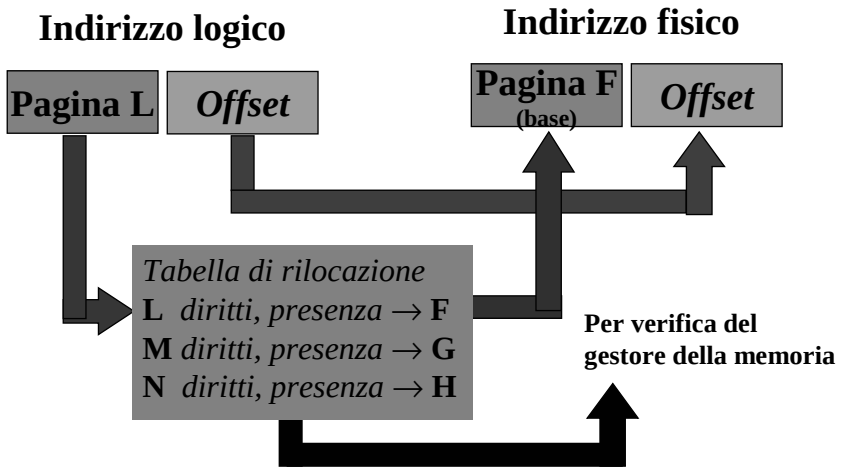
Memoria virtuale
Paginazione - 1

- **N** ed **M** suddivise in blocchi della medesima dimensione fissa e predefinita
- Lo stesso avviene per memoria principale e memoria secondaria
- La funzione *f* mappa **N** su **M**, generando un indirizzo (di pagina) in memoria principale
- La pagina richiesta può però trovarsi *solo* in memoria secondaria

Memoria virtuale
Paginazione - 2

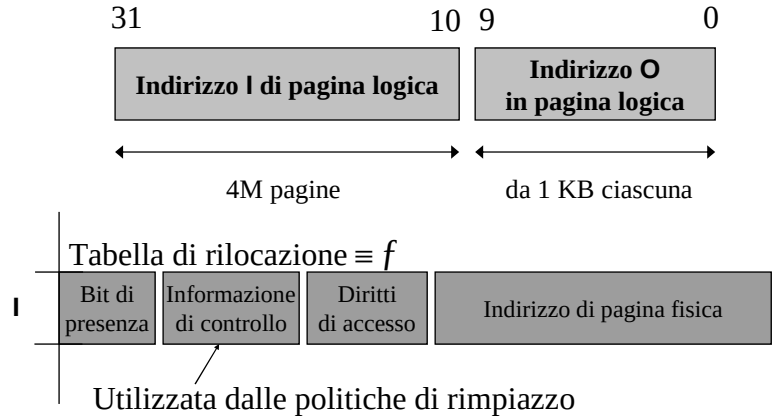


Memoria virtuale
Paginazione - 3



Memoria virtuale
Paginazione - 4

Indirizzo logico ∈ **N**



Memoria virtuale
Paginazione - 5

- *Page Fault*, quando la pagina indirizzata non è presente in memoria principale
- *Protection Fault*, quando il richiedente non ha diritti sufficienti per la richiesta
- Il *controllore degli accessi* è tipicamente un componente di sistema operativo (talvolta un componente fisico dedicato)

Memoria virtuale
Paginazione - 6

- La tabella di rilocazione può risiedere in memoria principale ed essere paginata
  - Il suo beneficio d'uso però scema con l'aumentare della frequenza d'accesso
- L'alternativa è l'uso di un dispositivo detto *Memory Management Unit* (**MMU**) contenente (parte del)la tabella
  - I programmi tendono a concentrare i riferimenti su una *piccola* quantità di pagine

Memoria virtuale
Paginazione - 7

- Le prime MMU erano realizzate in h/w, con intervento del S/O solo per la gestione degli eventi *page fault*
- Attualmente, la gestione delle pagine avviene prevalentemente a livello di S/O
- Per tabelle di rilocazione larghe abbastanza (p.es. 64 ingressi **I**), con bassa frequenza di miss, la gestione s/w è generalmente adeguata

Memoria virtuale
Frammentazione interna - 1

- Non vi è dimensione *ottimale* di pagina, ma un ottimo *locale* per applicazione
- Una dimensione *ridotta* di pagina è preferibile per ridurre la **frammentazione interna**
  - Dati *correlati* del programma vengono posti nella medesima pagina
  - La dimensione dei dati spesso non corrisponde ad un multiplo *esatto* di pagine
  - Il grado di frammentazione interna denota lo spazio di pagina inutilizzato

Memoria virtuale
Frammentazione interna - 2

- Se compattiamo i dati del programma sulle pagine disponibili, l'ultima pagina resterà parzialmente inutilizzata
  - In media, *metà* dell'ultima pagina
- Per un programma di dimensione s byte, p byte per pagina, ed e byte per ingresso in tabella di rilocazione, il costo medio C è

  $C = (s/p)e + p/2$ il cui minimo per p è $p = \sqrt{2se}$
  - Per s = 1MB ed e = 8B si ha p = 4KB (valore attuale)

Memoria virtuale
Politiche di rimpiazzo delle pagine – 1
(Tanenbaum)

- **Least Recently Used** (LRU)
  - Si rimpiazza la pagina usata *meno recentemente*
  - Di difficile implementazione
    - Occorre una lista ordinata, aggiornata ad ogni accesso, dove ogni pagina acceduta viene posta in cima → LRU = fondo della lista
- **Not Recently Used** (NRU)
  - Si rimpiazza una pagina *non utilizzata di recente*
  - Bit di utilizzo, periodicamente azzerato

Not Recently Used (NRU)

- Due bit associati ad ogni pagina (in tabella di rilocazione):
  - bit R settato a 1 quando la pagina è riferita (lettura e/o scrittura)
  - bit M, settato a 1 quando la pagina è scritta (cioè modificata in contenuto)
- Inizialmente i bit R ed M di tutte le pagine di un processo sono settati a 0
- Periodicamente, il bit R è resettato a 0
- Quando avviene un page fault e non c'è più spazio in memoria fisica, il s.o. classifica le pagine nelle seguenti 4 categorie
  - Classe 0: R==0, M==0
  - Classe 1: R==0, M==1 (può succedere perché R è periodicamente resettato)
  - Classe 2: R==1, M==0
  - Classe 3: R==1, M==1

  e si rimuove una pagina selezionata a caso fra quelle di classe (non vuota) con numero inferiore.

Memoria virtuale
Politiche di rimpiazzo delle pagine - 2

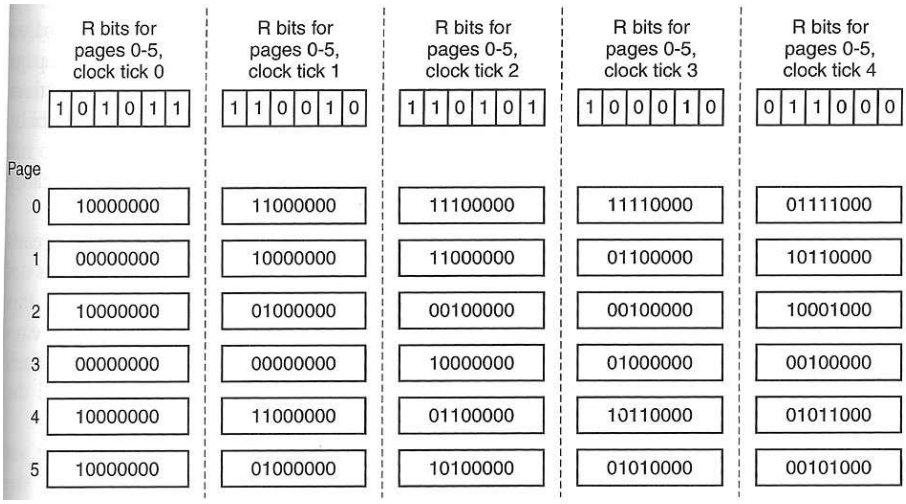
- **First-in, First-out** (FIFO)
  - Lista ordinata di tutte le pagine logiche caricate in memoria principale
  - la pagina caricata per prima è in cima alla lista, quella caricata per ultima è in fondo alla lista
  - In caso di page fault e memoria fisica piena, si rimuove la pagina in cima alla lista (che però potrebbe essere molto utilizzata!!)
- **Second Chance**
  - Come FIFO, però usa un bit di riferimento R (stesso di NRU): se la pagina in cima alla lista ha R==0 allora è rimossa, altrimenti R è resettato a 0 e la pagina messa in fondo alla lista e si torna ad esaminare la cima della lista con la stessa regola.
  - Se tutte le pagine in lista hanno R a 1, allora si ha un comportamento identico alla FIFO (dopo aver azzerato tutti gli R bit delle pagine)

Memoria virtuale

Politiche di rimpiazzo delle pagine - 3

- L'implementazione di LRU richiede supporto h/w spesso non disponibile
- Il livello S/O ne implementa una variante detta **Not Frequently Used** (NFU)
  - NFU usa la nozione approssimata di invecchiamento dall'ultimo accesso ('*aging*') mediante contatore di n bit
  - Con periodo t, il gestore trasla a destra di 1 il contatore di età e pone il bit di riferimento in MSb
  - Il contatore minore indica la pagina riferita meno frequentemente nell'intervallo di osservazione (t ×n), **non necessariamente la LRU !!**

NFU



Memoria virtuale
Politiche di richiesta di pagina

- **A richiesta** (*on demand*)
  - Si richiede la pagina mancante quando si è verificato un *page fault*
- **Con pre-caricamento** (*prefetching*)
  - Si *anticipa* la richiesta delle pagine adiacenti a quello in uso (giovandosi della proprietà di località spaziale)
  - Può causare ulteriori *page fault*

Memoria virtuale
Congruenza delle pagine

- Il gestore della memoria virtuale deve assicurare congruenza tra memoria principale e secondaria a seguito di scrittura dei dati
- L'alto costo di accesso alla memoria secondaria comporta l'uso della tecnica **write back** vista per la cache

Memoria virtuale
Segmentazione - 1

- La **paginazione** offre uno spazio di indirizzamento *uni-dimensionale*
  - Le pagine sono entità *dissociate* dalle esigenze e dalle strutture del programma (spazio di indirizzamento continuo)
- La **segmentazione** offre uno spazio di indirizzamento *multi-dimensionale*
  - I segmenti hanno dimensione variabile sia nello spazio che nel tempo
  - Ogni segmento costituisce uno spazio di indirizzamento a se stante
  - I segmenti sono entità *strettamente legate* alle strutture del programma

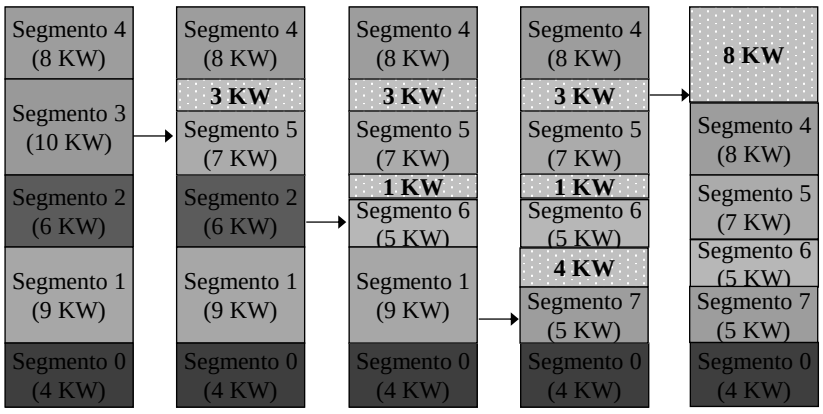
Memoria virtuale
Segmentazione - 2

- La segmentazione
  - Risponde ad esigenze *diverse da e superiori a* quelle della paginazione
    - Separazione tra dati ed istruzioni
    - Protezione di accesso più puntuale e specifica
  - Ha maggiore costo di implementazione ma anche maggiore flessibilità
    - Le dimensioni di segmento possono variare dinamicamente
  - Causa **frammentazione esterna**
  - Può essere combinata con la paginazione
    - Segmenti grandi possono essere paginati

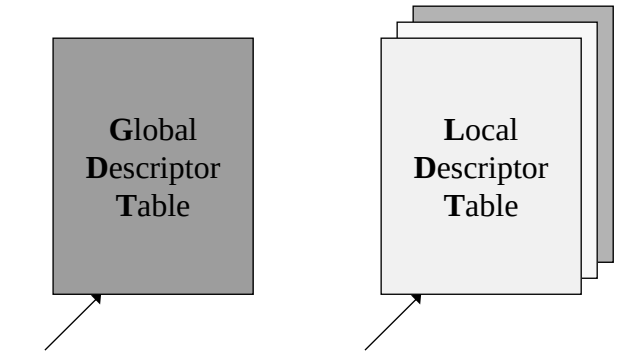
Memoria virtuale
Segmentazione - 3

- Traduzione complessa e costosa
- Delicata la scelta di dove allocare nuovi segmenti in memoria principale
  - **Best-fit** (porzione di memoria di dimensione più prossima)
  - **Worst-fit** (porzione di memoria di dimensione più grande)
  - **First-fit** (la prima porzione di memoria che può contenerlo)
- La frammentazione esterna richiede ricompattazione periodica
  - **Garbage collection**

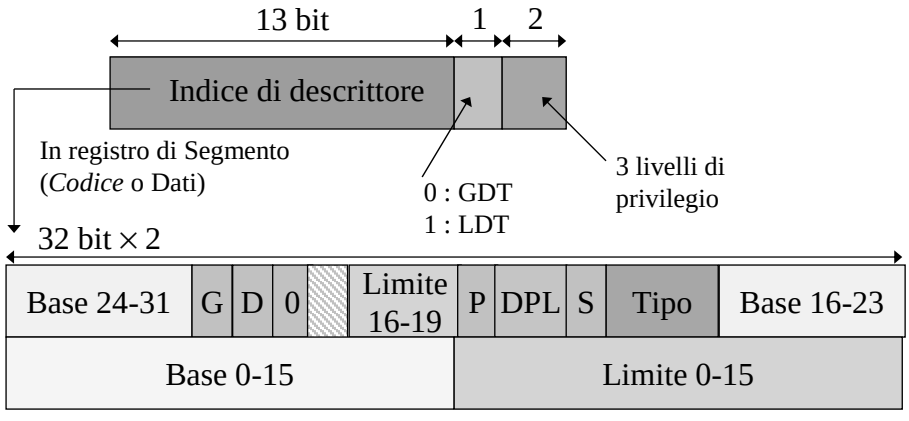
Memoria virtuale
Frammentazione e ricompattazione



Memoria virtuale
La soluzione Intel Pentium - 1



3.3 Memoria virtuale
La soluzione Intel Pentium - 2



Descrittore di segmento codice in registro di microprogramma

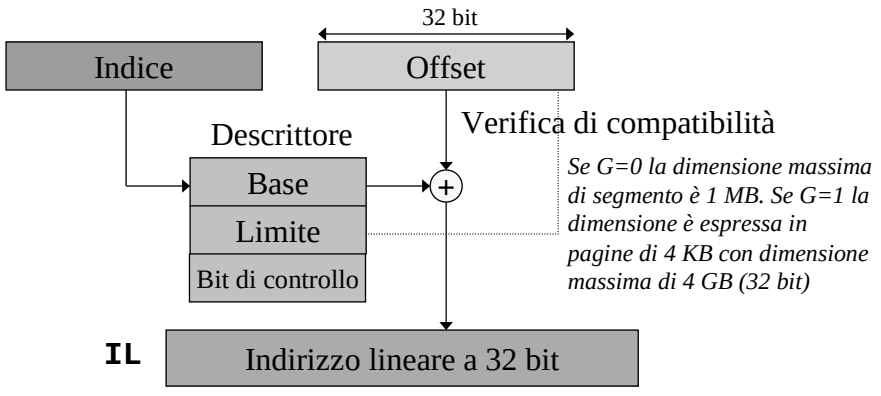
3.3 Memoria virtuale
La soluzione Intel Pentium - 3

Bit di controllo nel descrittore di segmento

G → 0 : Limite (*dimensione*) espresso in byte ; 1 : in pagine di 4kB
D → 0 : segmento a 16 bit ; 1 : a 32 bit
P → 0 : segmento assente da memoria ; 1 : presente
DPL → 3 livelli di privilegio
S → 0 : sistema ; 1 : applicazione
Tipo → tipo di segmento (Codice/Dati) e protezione

L'**indice** designa al più 8k descrittori su 16k disponibili
La **dimensione** di segmento dipende da G (*granularità*)

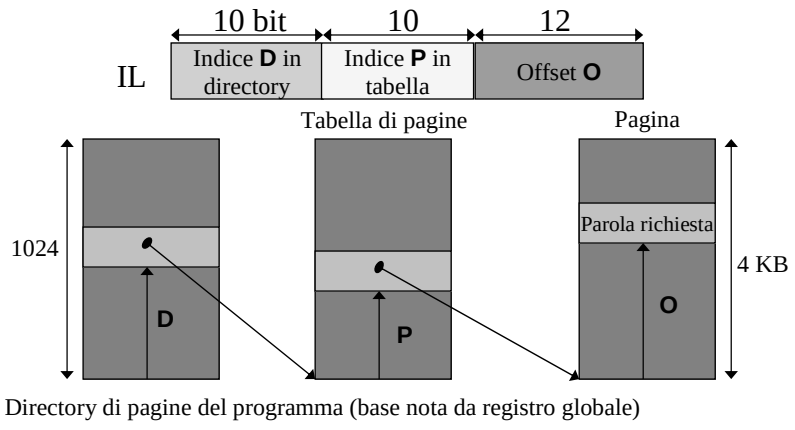
3.3 Memoria virtuale
La soluzione Intel Pentium - 4



3.3 Memoria virtuale
La soluzione Intel Pentium - 5

- Un bit in un registro globale di controllo indica se la paginazione è abilitata o meno
- Se *disabilitata*, abbiamo uno schema a segmentazione *pura*, dove IL rappresenta un indirizzo fisico
  - Segmenti diversi possono sovrapporsi
- Se *abilitata*, IL viene trattato come un indirizzo *virtuale*

3.3 Memoria virtuale
La soluzione Intel Pentium - 6



3.3 Memoria virtuale
La soluzione Intel Pentium - 7

- La gestione coinvolge *sia* il livello di macchina microprogrammata *che* il livello di macchina virtuale (S/O)
- Il S/O interviene ogniqualvolta si verifichino **eccezioni** (violazioni di diritti di accesso, offset erroneo, assenza di segmento, …)
- Tali eccezioni sono **sincrone** all'esecuzione del programma e si chiamano **trap**

3.3 Memoria virtuale
Considerazioni finali

- La memoria virtuale estende lo spazio di indirizzamento *oltre* il limite fisico della memoria principale
- La paginazione è *trasparente* al programmatore, la segmentazione *no*
- La segmentazione consente maggior protezione, anche tra processi *concorrenti*
- La tabella di rilocazione può essere complessa