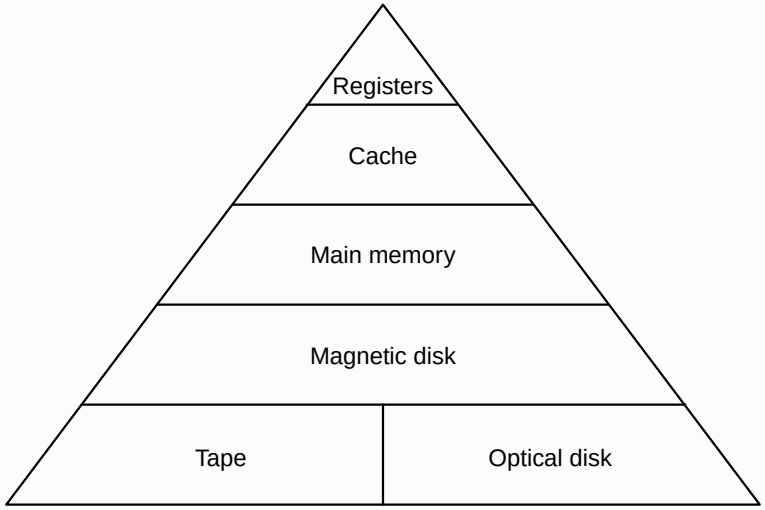
Connessioni tra livelli di memoria



Desiderata: memoria capiente e veloce

Metodo:

- i dati in uso risiedono nelle memorie veloce
- le memorie capienti contengono i dati di uso futuro

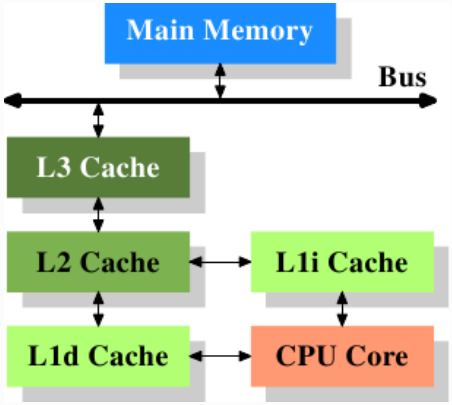
Lo spostamento dei dati:

- **esplicito**: caricamento di un programma, scrittura di un registro;
- **trasparente** non visibile ai livelli alti: gestito dal hardware (e dal sistema operativo)

Tecniche impiegate

- **memoria cache**: sposta i dati tra memoria cache e principale;
- **memoria virtuale**: sposta i dati tra memoria principale e di massa.

Memoria cache, struttura



Ad ogni accesso alla memoria

controllo se la linee di cache contiente il dato è presente in memoria:

- se la linea è presente (cache hit)accedo al dato
- se la linea non è presente (cache miss):
  - scarico una linea dalla memoria cache,
  - inserisco la linee contente il dato,
  - accedo alla dato.

Rapporto hit/miss

Il meccanismo della memoria cache funziona bene se i cache miss sono poco frequenti.

Posto:
h probabilità di cache hit;
t_c tempo di accesso alla cache;
t_p tempo di accesso alla memoria principale.
Il tempo medio di accesso alla memoria t_M è:

$$t_M = t_c + (1 - h) \times t_p$$

La memoria cache è conveniente sole se $t_M < t_p$ ossia se:

$$h > \frac{t_c}{t_p}$$

Quali dati in memoria cache

- Obiettivo mantenere nelle cache i dati che saranno usati nel prossimo futuro.
- Metodo: si sfruttano due principi statistici:
  - **località temporale** i dati usati recentemente hanno buona probabilità di essere usati in futuro
  - **località spaziale** dati contigui a dati usati recentemente hanno buona probabilità di essere usati in futuro

Memoria cache in dettaglio

Si tengono in cache i dati utilizzati di recente e i dati contigui a questi.

- La memoria viene divisa in linee di cache (dim: 32-64 byte).
- Mantengo in memoria cache un insieme di linee: quelle utilizzate più di recente.

Tecniche di implementazione

Come funziona, come si cercano i dati in memoria cache:

- **cache ad accesso diretto**
- **cache (n-way) set-associative**
- ma anche: cache associative, cache 2-way skewed associative . . . .

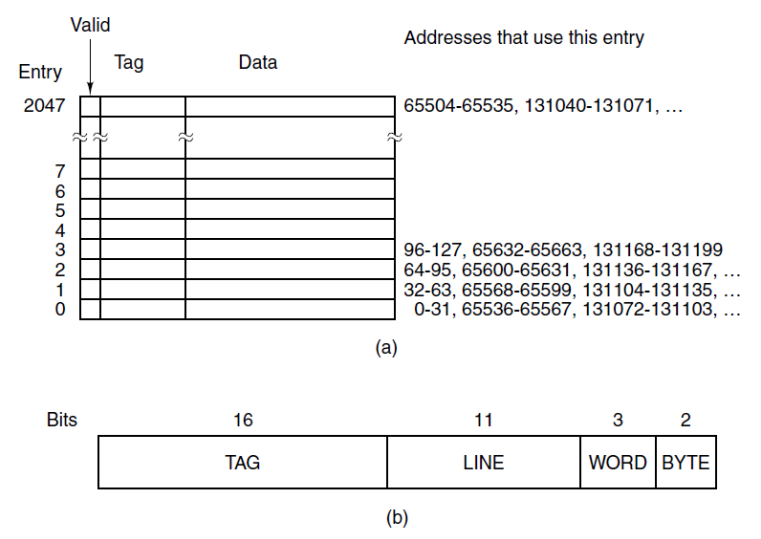
Cache ad accesso diretto

Ogni linea di cache può risiedere in un'unica posizione della memoria cache.
Posizione determinata univocamente dall'indirizzo (dei dati nella linea).
L'indirizzo viene diviso in diverse parti:

- TAG+Line: individuano la linea di cache;
- Word+Byte: individuano il byte all'interno della linea di cache.

Line: posizione della linea in memoria cache.
TAG: differenzia linee su stessa posizione.

Cache ad accesso diretto

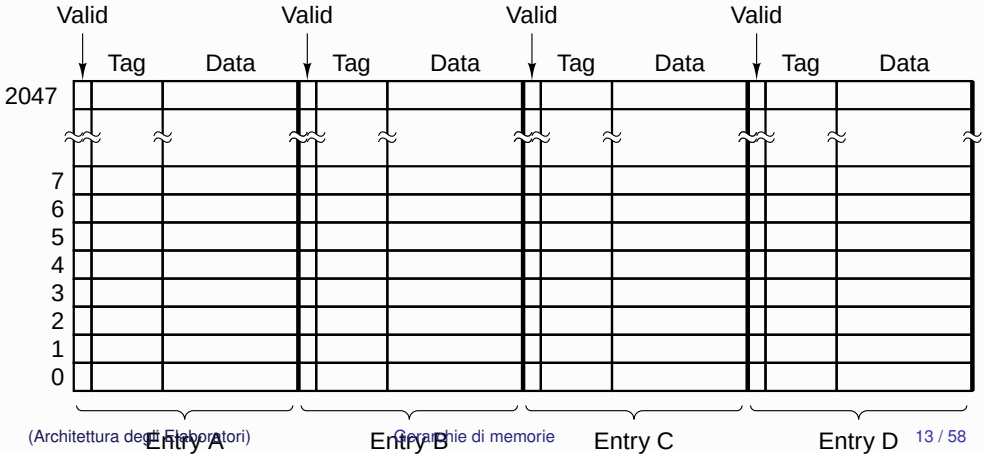


Vantaggi - svantaggi

- **vantaggi** semplice da implementare
- **svantaggi** meccanismo rigido,
  non è possibile scegliere la linea da scartare, (in base alla località temporale)
  in alcuni casi genera un elevato numero di cache miss.
  Esempio di caso critico:
  un programma accede continuamente due locazioni di memoria
  contenute in due linee di cache distinte
  ma mappate nella stessa posizione di cache:
  con identico campo Line.

Cache (n-way) set-associative

n tabelle di cache (ad accesso diretto).
ogni dato contenuto in una qualsiasi tabella



Cache (n-way) set-associative

- Sistema più flessibile dell'accesso diretto: è possibile scegliere tra più alternative dove inserire la linee di chace, (quale linea di cache scartare).
- Vengono evitati i casi critici descritti della cache ad accesso diretto.
- Più complesso da implementare: richiede più hardware.
- Lo standard dei processori attuali:
  - cache Core i7: 8 - 4 - 16 -way associative (a seconda del livello);
  - cache ARM Cortex A15 2 - 16 - way associative.

Set-associative: rimozione linee

- Quale linea scartare per far spazio ad una nuova linea?
- In base alla località temporale: quella usata meno recentemente Least Recently Used LRU).
- Implementazione: per ogni indirizzo una lista descrive l'ordine di accesso alle varie tabelle.

Scrittura in cache

Cosa fare in caso di scrittura in un dato in memoria cache?
Due alternative:

- write through: si scrive in cache e in memoria principale. Più semplice da implementare, mantiene la coerenza ma genera più accessi alla memoria.
- write deferred – write back: si scrive solo in cache, si aggiorna la memoria principale quando la linea cache viene scaricata: più efficiente (meno accessi in memoria) ma genera inconsistenze tra cache e memoria principale.

Scrittura in memoria principale

Cosa fare in caso di scrittura in un dato non in memoria cache?
Due alternative:

- si modifica solo la memoria principale,
- write allocation: il dato viene portato in memoria cache

Nota: la scrittura non è necessariamente un operazione bloccante:
dopo un comando di scrittura, il processore non è deve attendere nessun dato,
può procedere immediatamente all'istruzione successiva.

Memoria Virtuale

Parte della memoria di massa utilizzata come memoria principale.
Trasferimento automatico dei dati tra massa e memoria principale.

Motivazioni:

- simulare una memoria principale più ampia, programmi di grosse dimensioni, molti processi in esecuzione;
- implementare meccanismi di protezione, impedire ad un processo l'accesso ai dati di un altro processo.

Cenni storici

Prime soluzioni alla carenza di memoria (anni 50):
memoria non sufficiente a contenere tutto il programma,
programmi divisi in moduli, overlay, esplicitamente caricati e scaricati dalla memoria.
Memoria disponibile (anni 50): pochi KB.

Memoria virtuale nata per realizzare una gestione automatica degli overlay

- Ideazione: anni 60 gruppo di Manchester.
- Utilizzazione: anni 70.

Idea base

Similmente alla memoria cache, i dati possono risiedere:

- in una memoria relativamente veloce: memoria principale
- in una memoria più lenta e capiente: memoria di massa (disco magnetico).

Con la memoria virtuale, si diversificano

- spazio di indirizzamento (indirizzi utilizzabili dal programma) e
- posizioni della memoria (indirizzi fisici della memoria disponibile)

Diverse realizzazioni: paginazione, segmentazione.

Paginazione

Molte similitudini con la memoria cache.

Spazio di indirizzamento diviso in parti,
tutte con la stessa dimensione (una potenza di 2)
pagine

- alcune presenti in memoria principale (e di massa),
- alcune presenti solo in memoria di massa,
- altre vuote e non esistenti.

Accesso ad una parola in memoria

Dato l'indirizzo di una parola in memoria,

- si determina la pagina contenente la parola,
- si controlla se la pagina è presente in memoria principale,
- se non presente, page fault, la pagina viene caricata in memoria (dal sistema operativo),
- si calcola la posizione della parola cercata e si accede ad essa.

Meccanismo trasparente (invisibile) al programmatore.

Memoria virtuale

Si distingue tra:

- indirizzo virtuale indirizzo utilizzato all'interno del programma
- indirizzo fisico indirizzo utilizzato dal hardware, dove effettivamente si trova il dato

e parallelamente tra:

- pagina virtuale pagina della memoria virtuale
- pagina fisica pagina residente in memoria

Mappa: indir. virtuale $\rightarrow$ indir. fisico

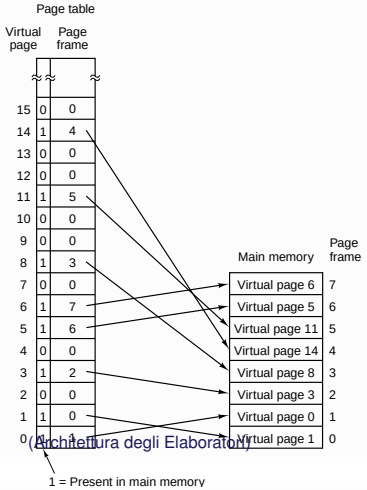
Meccanismi più sofisticati rispetto alla memoria cache: ogni pagina può risiedere in un qualsiasi slot della memoria principale.

Viene utilizzata una
Tabella delle pagine (Page Table), chiamata anche
Mappa della memoria, PMT Page-Map Table,
che ad ogni pagina virtuale associa:

- un bit di presenza in memoria principale,
- se presente, un indirizzo di inizio pagina (posizione in memoria).

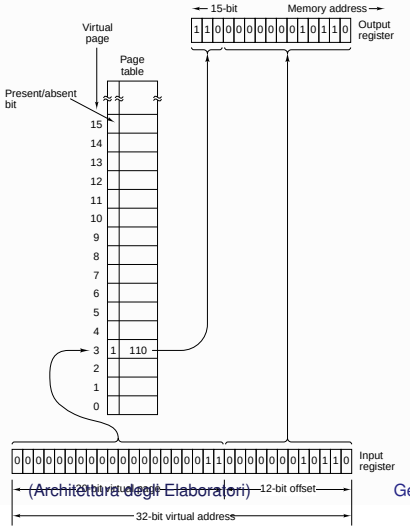
Esempio di tabella delle pagine

associa a ogni numero di pagina:
bit di presenza, indirizzo base.



Esempio di calcolo indirizzo fisico

Indirizzo fisico = inizio pagina + offset.



Accesso ad un indirizzo virtuale

- Si scompone l'indirizzo in numero pagina e offset.
- Con il numero di pagina si accede alla tabella delle pagine e si controlla se la pagina è presente in memoria principale.
- Se la pagina è presente, la tabella delle pagine fornisce l'indirizzo base della pagina (l'indirizzo della prima parola della pagina), sommando, offset e indirizzo base, si ottiene l'indirizzo fisico corrispondente.
- Se la pagine è assente (page fault), chiamata al sistema operativo, si carica la pagina, dalla

Esempio

L'esempio considera un calcolatore con:

- indirizzi di memoria di 32 bit - spazio di indirizzamento: 4GB;
- diviso 1 M di pagine di 4kB ciascuna

In questa caso, un indirizzo:

- i primi 20 bit identificano la pagina: numero pagina,
- i 12 bit rimanenti identificano la parola all'interno della pagina: offset.

Le pagine molto più ampie delle linee di cache.

Page Frame

Zona della memoria principale contente una pagina fisica.

Alcune zone di memoria sono ad accesso diretto (senza paginazione).
Es.: la zona contenenti la page table; indirizzi relativi memory mapped I/0.

Page	Virtual addresses
≈	≈
15	61440 – 65535
14	57344 – 61439
13	53248 – 57343
12	49152 – 53247
11	45056 – 49151
10	40960 – 45055
9	36864 – 40959
8	32768 – 36863
7	28672 – 32767
6	24576 – 28671
5	20480 – 24575
4	16384 – 20479
3	12288 – 16383
2	8192 – 12287
1	4096 – 8191
0	0 – 4095

Page frame	Bottom 32K of main memory Physical addresses
7	28672 – 32767
6	24576 – 28671
5	20480 – 24575
4	16384 – 20479
3	12288 – 16383
2	8192 – 12287
1	4096 – 8191
0	0 – 4095

Page fault

Ogni page fault genera un chiamata al sistema operativo che

- Cerca un page frame vuoto.
- Se non esiste, scarica una pagina da un page frame: quale pagina?
  - LRU Least Recently Used (o una sua approssimazione): si sfrutta la località temporale.
  - FIFO First In First Out: più semplice da implementare.
- Se la pagina da scaricare è stata modificala, aggiorna la copia in memoria di massa.
- Carica, in memoria principale la pagina cercata.

L'accesso al disco è costosa in termini di tempo.

Assegnazione pagine

I page frame devo essere ripartiti tra i processi, quanti frame per processo?

Working set: insieme indirizzi di uso corrente, (quelli usati nell'ultimo secondo).
Tutte le pagine del working set devono essere contenuta in memoria principale.
Altrimenti: thrashing, generazione di continui page fault.

Il thrashing può essere causato
da una memoria principale insufficiente,
da algoritmi di assegnazioni di pagine e di rimozione di pagine inadatti.

Assegnazione delle pagine

Costruzione della page table, una per processo.

Al caricamento di un programma, due possibili scelte:

- un sottoinsieme di pagine viene caricato in memoria principale
- nessuna pagina caricata in memoria principale: demanding paging paginazione su richiesta.

L'esecuzione del programma determina le pagine caricate, in condizioni ottimali, un sovrainsieme del working set.

Implementazione – MMU

- In parte hardware: MMU (Memory Management Unit): esegue la mappatura indirizzo logico in indirizzo fisico;
  contenuta nel chip del processore,
  dal punto di vista logico, un unità indipendente,
- in parte software: la mancanza di una pagina page fault causa un eccezione, gestita dal sistema operativo.

Memorizzazione delle page table

Devono essere letta velocemente,
altrimenti ogni accesso alla memoria virtuale ha il costo di due accessi alla memoria fisica.

Soluzione:
Parte della page table, contenuta in una memoria cache dedicata
Translation Lookside Buffer (TLB),
contenuta all'interno dalla MMU.

Dimensioni:
ogni processo in esecuzione ha una sua page table,
ciascuna di circa $\sim$ 10 MB,
troppo grandi per essere contenute tutte nella cache.

Dimensioni della pagina

Pagine grandi per:

- Tempo di accesso al disco
- Località spaziale
- Page table più piccole

Pagine piccole per:

- sfruttare al meglio la memoria
- Limitare il costo dei page fault
- Frammentazione: l'ultima pagina di ogni programma utilizzata solo in parte

Dimensioni della pagina

Anni 70: 0.5 – 1 KB

Attualmente: $>$ 4 KB

Nei processori Sparc è possibile scegliere tra:
8KB, 64KB, 512KB, 4MB

Limiti della paginazione

Paginazione: un unico spazio di indirizzamento condiviso da tutti i dati.
Memoria non strutturata: un unico calderone.

Spesso è utile strutturare la memoria in parti logicamente distinte. Esempi:

- la memoria della JVM (4 parti diverse);
- in un calcolatore troviamo:
  - programmi
    - diversi programmi
    - programmi divisi in procedure
  - dati
    - divisi in strutture dati

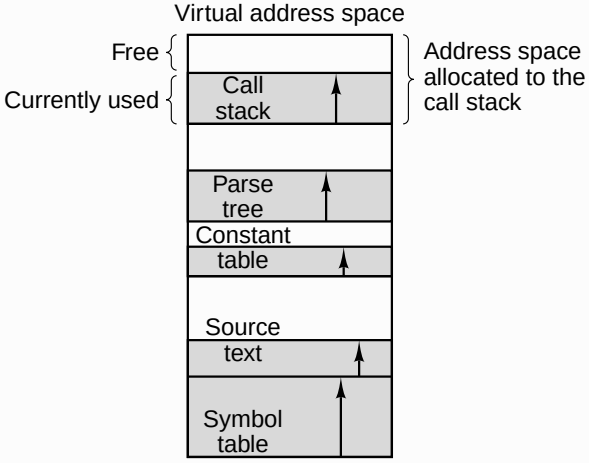
Memoria strutturata

Permette di gestire meglio:

- strutture dati di lunghezza variabile;
- l'implementazione di meccanismi di protezione: diversi tipi di dati usano restrizioni all'accesso differenti

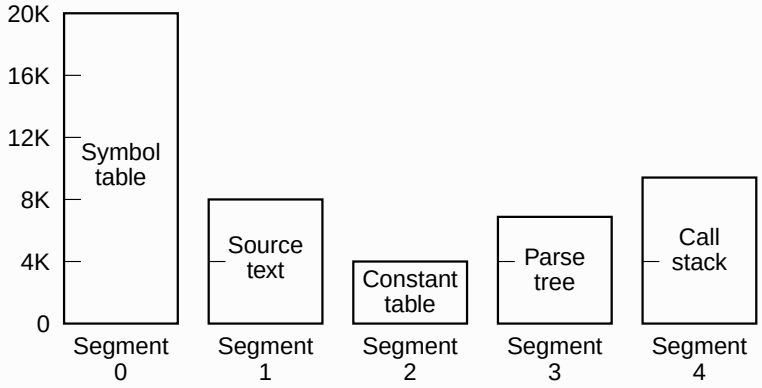
Esempio

Spazio dei dati di un compilatore.



Esempio

Dal punto di vista logico: spazi indipendenti in zone separate della memoria



Segmentazione

Una diversa implementazione della memoria virtuale dove:

- il programmatore divide i dati in unità logiche: segmenti,
- la memoria virtuale
  - mappa i segmenti sulla memoria fisica,
  - li sposta, a seconda delle necessità, da memoria di massa a memoria principale,
- lo spazio occupato dai segmenti può superare quello della memoria fisica.

Paginazione - segmentazione

- Segmento: definito dal programmatore — Pagina: invisibile al programmatore
- unità logica — unità fisica,
- ampiezza variabile — di ampiezza fissa.

Nella segmentazione:

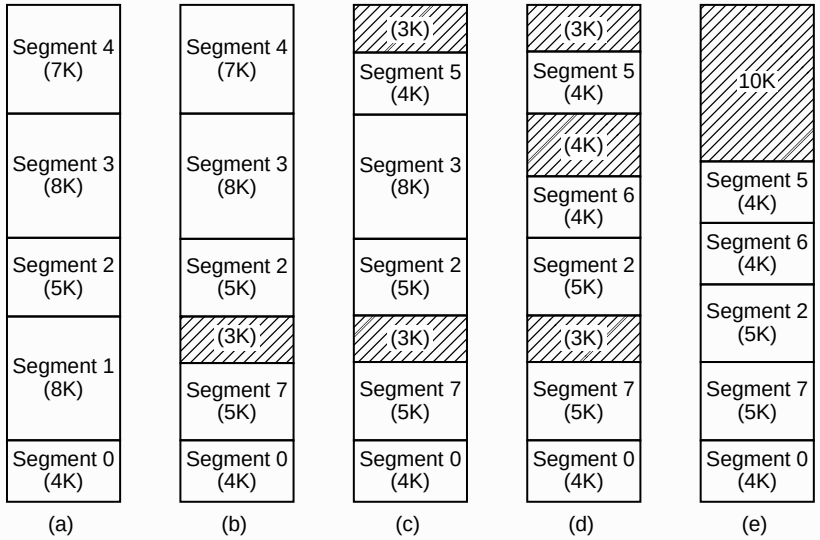
Indirizzo: numero segmento - posizione relativa

Supporto per meccanismi di protezione: si assegna sui vari segmenti differenti livelli di privilegio.

Implementazione: Swapping

- Segmenti caricati, per intero, in memoria principale in base alla necessità,
- scaricati, per far posto a nuovi segmenti, quando giudicati non necessari.
- Problemi dovuti alla dimensione variabile dei segmenti: ricerca di spazi liberi.

Esempio



Frammentazione esterna

La memoria libera viene divisa in tante zone non contigue: carenza di spazi liberi di grosse dimensioni (checkerboarding)

Alcuni rimedi:

- unione di spazi liberi contigui;
- compattazione: spostare e riunire i segmenti, richiede tempo, non utilizzabil frequentemente.

Algoritmi di selezione spazi liberi

Necessità di opportune strategie per la scelta degli spazi liberi, dove inserire un nuovo segmento: si selezione tra gli spazi utilizzabili

- best fit: quello più piccolo (complicato da implementare)
- first fit: il primo spazio travato (semplice e con buone prestazioni)

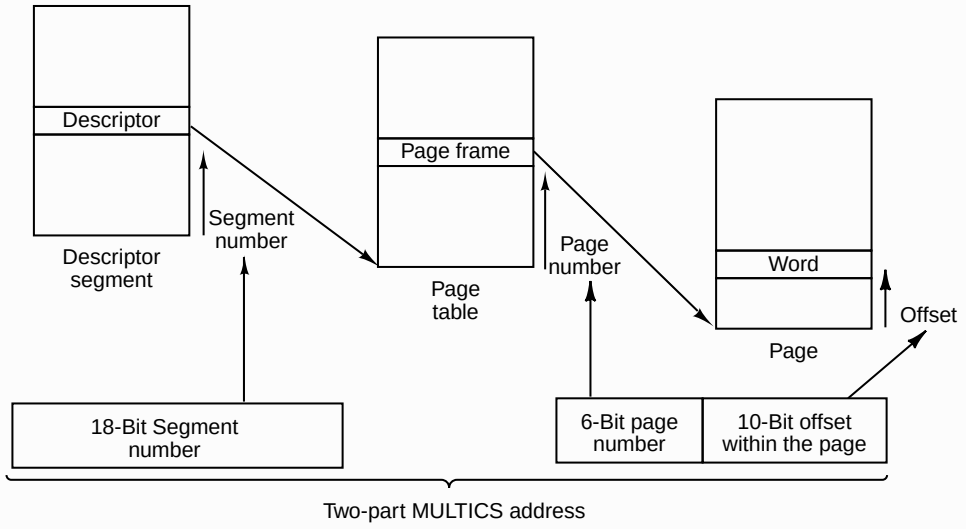
Algoritmi di rimozione segmenti, per far posto ad un nuovo segmento.
Devono tener conto della lunghezza dei segmenti e del loro uso.
Piuttosto complessi.

Paging

Segmentazione paginata: combinazione di segmentazione e paginazione:
ogni segmento diviso in pagine.

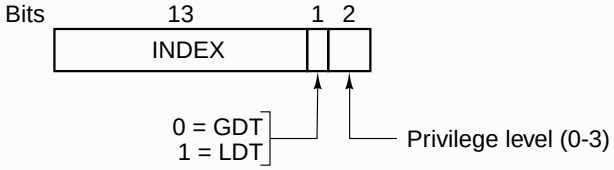
- Si risolvono i problemi di frammentazione esterna.
- Un ulteriore livello di divisione.
- Prime implementazioni: MULTIX (1965) - progenitore di UNIX.
- Supportato dai processori x86 (Core i-n).

Esempio di Paging



Memoria virtuale nei processori x86

Segmentazione paginata (ma anche pura o solo paginazione)
Selezione del segmento: implicita attraverso registri interni, due spazi di indirizzamento
CS segmento codice
DS segmento dati



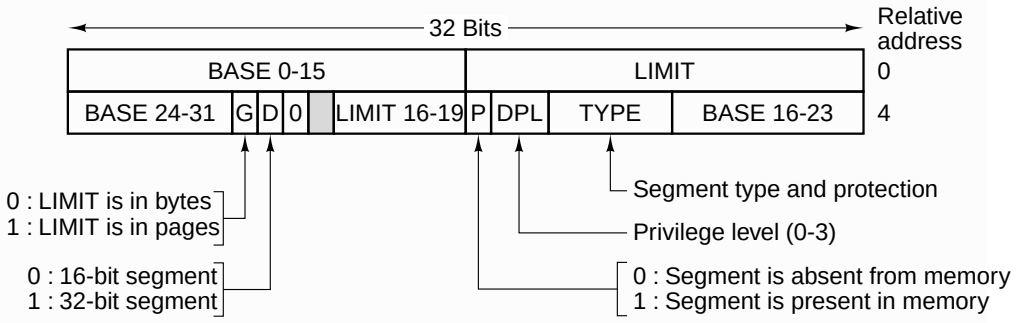
Memoria virtuale nei processori x86

Due tabelle descrittori di segmenti:

- LDT (Local Descriptor Table) Specifica del processo
- GDT (Global Descriptor Table) Comune tra tutti i processi

Descrittore di segmento

Struttura complessa dovuta a vincoli di retrocompatibilità.

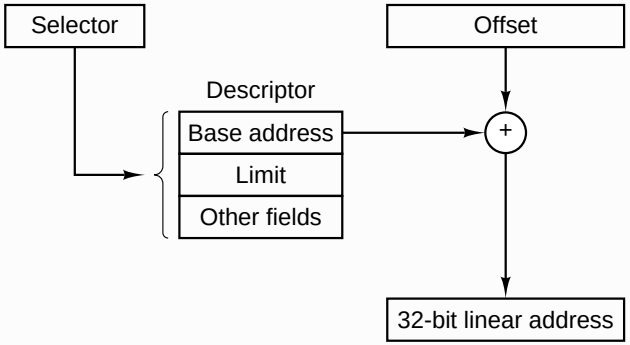


Descrittore di segmento

Campi principali:

- Base Posizione di inizio del segmento
- Limit Dimensioni del segmento
- D Dimensioni descrittore (16–32 bit)
- P Presente in memoria
- DPL Livello di privilegio.

Calcolo dell'indirizzo

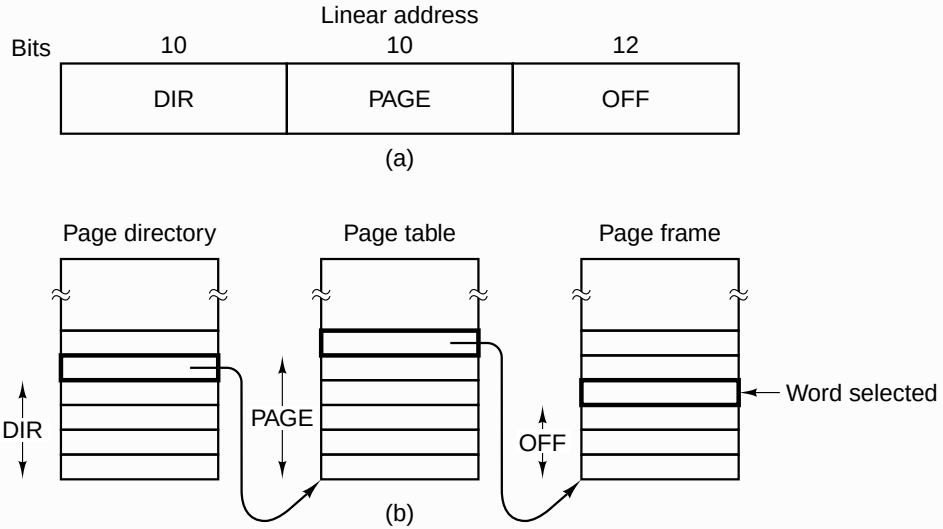


Calcolo dell'indirizzo

Due alternative (in base a bit nel registro globale di controllo)

- paginazione non attivata: accesso diretto alla memoria (segmentazione pura)
- paginazione attivata: accesso tramite paginazione (segmentazione paginata)

Struttura delle page tables



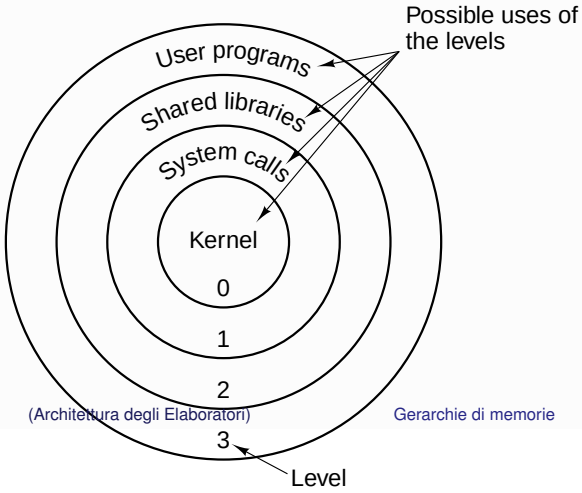
Struttura delle page tables

- Ogni segmento (programma) una page table,
- Page table divisa in più sottotabelle.
- Problemi di dimensione: parte della page table contenuta in memoria secondaria.
- Tempi di accesso: nella MMU una memoria cache per i descrittori di pagina usati più recentemente
- Utilizzo di un unico segmento $\rightarrow$ paginazione

Protezione

Si confrontano i livelli di protezione (privilegio):

- dei segmenti (descrittore di segmento),
- del processo in esecuzione (PSW program status word)



Protezione

Se il PSW assegna al processore ha livello di privilegio *n*, è impedito l'accesso ai segmenti con livelli di protezione più basso di *n*

Il livello di protezione del PSW modificabile in maniera controllata, mediante chiamate al sistema operativo, a procedure in punti di ingresso ufficiali.