

PROBLEMI DI PROGETTAZIONE E IMPLEMENTAZIONE PER SISTEMI DI PAGING

Danilo Croce

Dicembre 2024



GESTIONE DELLA MEMORIA: OUTLINE

- Memory Abstraction
- Virtual Memory
- Algoritmi di sostituzione delle pagine
- **Problemi di Progettazione per Sistemi di Paging**



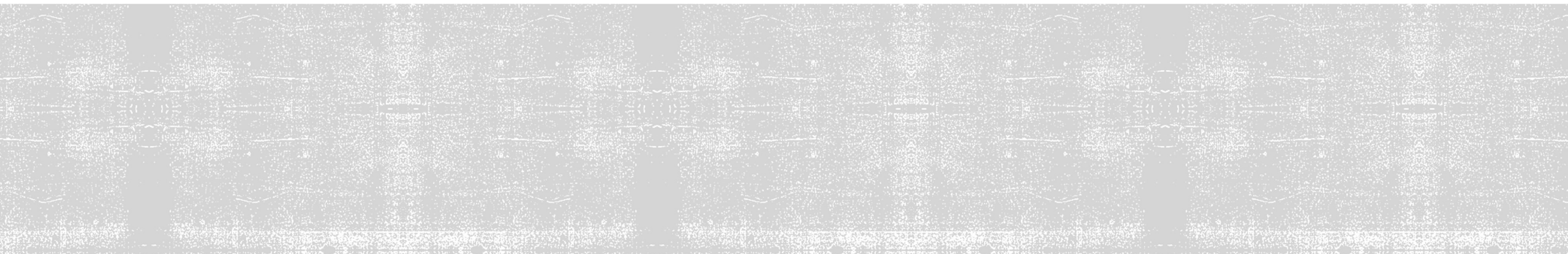
CONSIDERAZIONI NELLA PROGETTAZIONE DI SISTEMI DI PAGINAZIONE

- La paginazione è un processo complesso che richiede una comprensione approfondita di molteplici aspetti per una progettazione efficace.
- **Altri Aspetti Cruciali**
 - **Allocazione della Memoria**
 - Scelta tra **allocazione globale VS locale** oppure **allocazione equa vs proporzionale** e come questa influisce sulla gestione delle risorse e sulle prestazioni del sistema.
 - **Gestione dei Page Fault:**
 - **Monitoraggio** della frequenza dei page fault **per ottimizzare l'uso e allocazione della memoria** e ridurre i tempi di attesa.
 - **Ottimizzazione delle Prestazioni:** Valutare le prestazioni del sistema di paginazione per massimizzare l'efficienza
 - Esempio: *«quando deve essere grande una pagina?», «come limitare l'uso della memoria per i processi?»*
 - **Decisioni di Progettazione:** Considerare fattori come la dimensione del set di lavoro, il comportamento dei processi, e la località dei riferimenti alla memoria per scegliere l'algoritmo più adatto.





«PROBLEMI» DI PROGETTAZIONE



I PROBLEMI DI PROGETTAZIONE PIÙ COMUNI

- Allocazione Globale VS Locale
- Allocazione Equa VS Proporzionale
- Dinamica di Allocazione delle pagine
- Policy di pulizia
- Dimensione delle pagine
- Istruzioni separate e spazi dei dati
- Pagine e Librerie condivise
- File mappati in memoria



ALLOCAZIONE DI MEMORIA IN SISTEMI DI PAGINAZIONE: GLOBALE VS LOCALE

- **Allocazione Locale**

- Ogni processo riceve una porzione fissa della memoria.
- Semplice da implementare, ma può portare a inefficienze se il set di lavoro del processo.

- **Allocazione Globale**

- Distribuzione dinamica della memoria tra i processi.
- Più efficace per adattarsi alle esigenze variabili dei processi, ma richiede una gestione più complessa.

- **Esempio Pratico**

- In Figura la differenza tra sostituzione locale (solo pagine del processo A) e globale (pagine di tutti i processi).

	Age
A0	10
A1	7
A2	5
A3	4
A4	6
A5	3
B0	9
B1	4
B2	6
B3	2
B4	5
B5	6
B6	12
C1	3
C2	5
C3	6

(a)

Il processo A ha bisogno di allocare una pagina per A6

A0
A1
A2
A3
A4
A6
B0
B1
B2
B3
B4
B5
B6
C1
C2
C3

(b)

Allocazione Locale: è possibile rimuovere solo pagine del processo A

A0
A1
A2
A3
A4
A5
B0
B1
B2
A6
B4
B5
B6
C1
C2
C3

(c)

Allocazione Globale: è possibile rimuovere pagine di qualsiasi processo



VANTAGGI DELL'ALLOCAZIONE GLOBALE DELLA MEMORIA

- **Adattabilità degli Algoritmi Globali:**

- Gli algoritmi globali di sostituzione delle pagine si adattano meglio alle esigenze variabili dei processi.
- Aumentano l'efficienza quando la dimensione del set di lavoro varia nel tempo.

- **Limiti degli Algoritmi Locali:**

- Il **thrashing** può verificarsi con algoritmi locali se il set di lavoro di un processo cresce oltre la memoria allocata.
- La memoria può essere sprecata quando il set di lavoro di un processo si riduce e la memoria non viene riassegnata.

- **Gestione Dinamica della Memoria:**

- Con l'allocazione globale, il sistema operativo deve dinamicamente assegnare e riassegnare frame ai processi.
- E' possibile utilizzare i bit di aging per monitorare la frequenza di accesso delle pagine, anche se questo potrebbe non essere sufficiente per prevenire il thrashing.

- **Sfide del Monitoraggio del Set di Lavoro:**

- I bit di aging forniscono una stima approssimativa, che potrebbe non riflettere cambiamenti rapidi nel set di lavoro.
- È fondamentale che il sistema di paginazione possa reagire in modo agile ai cambiamenti delle esigenze di memoria.



STRATEGIE DI ALLOCAZIONE DELLA MEMORIA NEI SISTEMI DI PAGINAZIONE

- **Allocazione Equa vs Proporzionale:**

- **Allocazione Equa:**

- Distribuzione uniforme dei frame tra i processi.
 - Esempio: 12 . 416 frame divisi equamente tra 10 processi risultano in 1 . 241 frame per processo.
 - **Svantaggi:** Non tiene conto delle diverse esigenze di memoria tra processi di dimensioni varie.

- **Allocazione Proporzionale:**

- Assegnazione di frame in base alla dimensione del processo.
 - Rispecchia meglio le necessità di memoria, evitando allocazioni inadeguate.

- **Importanza del Limite Minimo di Pagine:**

- Assicurare che **ogni processo abbia abbastanza pagine** per eseguire le operazioni fondamentali.
 - **MA prevenire situazioni** in cui processi con istruzioni che **attraversano i limiti** delle pagine non possano eseguire.



DINAMICA DI ALLOCAZIONE E ALGORITMO PAGE FAULT FREQUENCY (PFF)

- **Gestione Dinamica dei Frame:**
 - Inizio con un'allocazione proporzionale alla dimensione del processo.
 - Aggiornamento dinamico dell'allocazione in base all'evoluzione delle esigenze durante l'esecuzione.
- **Page Fault Frequency (PFF):**
 - **Monitoraggio della frequenza dei page fault per regolare l'allocazione di memoria** di un processo.
 - Aumenta i frame se i page fault sono troppo frequenti, diminuisce se sono rari.
 - Non specifica quale pagina rimuovere, focalizzandosi sulla dimensione dell'allocazione.



RELAZIONE TRA ALLOCAZIONE DI MEMORIA E PAGE FAULT

▪ Relazione tra Frame Assegnati e Page Fault:

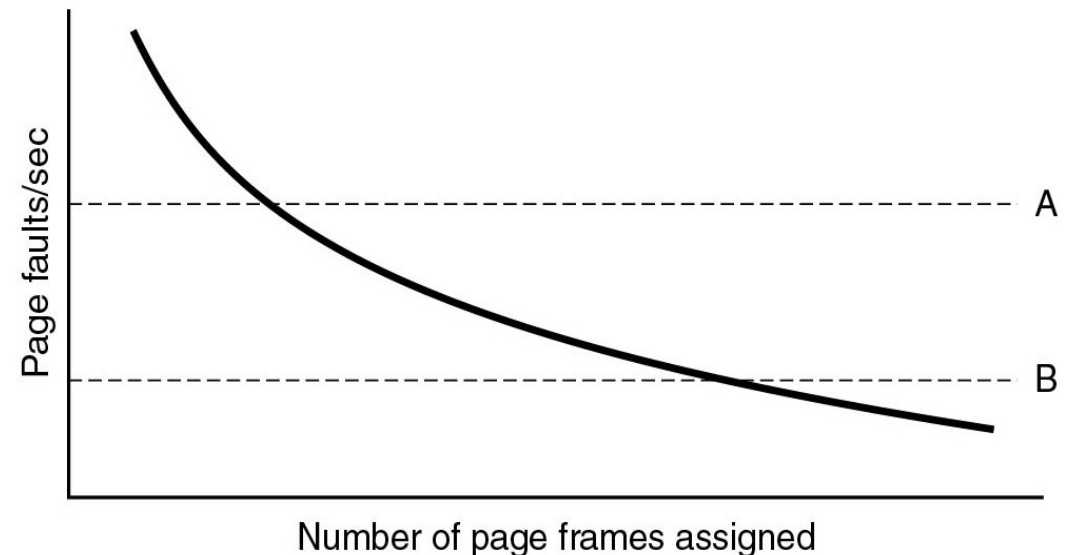
- Secondo algoritmi come LRU, più pagine vengono assegnate a un processo, meno frequenti saranno i page fault.
- La frequenza di page fault diminuisce man mano che aumenta il numero di frame assegnati.

• Monitoraggio della Frequenza dei Page Fault:

- Si contano i page fault per secondo e si utilizza una media mobile per tenere traccia delle fluttuazioni.

A. Alta frequenza di page fault indica necessità di più frame.

B. Bassa frequenza di page fault suggerisce che il processo ha più memoria del necessario.



GESTIONE DEL THRASHING E CONTROLLO DEL CARICO DI MEMORIA

- **Thrashing in Presenza di Allocazione Ottimale:**
 - Anche con il miglior algoritmo, **il thrashing purtroppo può sempre verificarsi** se i set di lavoro di tutti i processi eccedono la memoria disponibile.
 - Il PFF può segnalare una richiesta collettiva di più memoria senza che nessun processo possa cedere frame.
- **Strategie di Mitigazione:**
 - **Out Of Memory Killer (OOM):**
 - Processo di sistema che seleziona e termina i processi in base a un punteggio di "cattiveria" per liberare memoria.
 - Processi con elevato utilizzo di memoria o minor importanza sono tipicamente selezionati.
 - **Swapping (Scambio):**
 - Meno drastico dell'OOM Killer, sposta i processi su memoria non volatile, liberando le loro pagine per altri processi.
 - Può ridurre la richiesta di memoria senza interrompere l'esecuzione dei processi.



SCHEDULING A DUE LIVELLI E TECNICHE DI RIDUZIONE DI MEMORIA

- **Scheduling a Due Livelli:**
 - **alcuni processi sono in memoria non volatile** e solo una parte è schedulata attivamente
 - aiuta a **gestire meglio il carico** di memoria.
 - utile per ridurre occupazione di memoria di **processi in background in sistemi interattivi**
- **Gestione della Multiprogrammazione:**
 - La selezione dei processi da spostare considera anche caratteristiche:
 - Sono processi CPU bound o I/O bound
 - Qual è la dimensione e/o frequenza di paginazione dei processi
- **Altre Tecniche:**
 - Oltre a «uccidere» o spostare processi, si possono usare **compattamento, compressione e deduplicazione** (same page merging).



POLICY DI PULIZIA E PAGING DAEMON

- **Contesto:** La policy di **pulizia** è **un aspetto critico** nella gestione della memoria.
- **Aging e Frame Liberi:** L'aging è più efficace con molti frame di pagina liberi disponibili.
 - Se i frame sono tutti occupati e modificati, occorre scrivere le vecchie pagine in memoria non volatile prima di caricarne di nuove.
 - È **preferibile mantenere un buon numero di frame di pagina liberi** piuttosto che occupare tutta la memoria e cercare frame liberi solo al bisogno.
- **Paging Daemon:** Un processo in background usato dai sistemi di paginazione
 - **Inattivo per la maggior parte del tempo**, si attiva periodicamente per controllare lo stato della memoria.
 - Quando i frame liberi scarseggiano, inizia a selezionare pagine da rimpiazzare utilizzando un algoritmo di sostituzione delle pagine.



POLICY DI PULIZIA E PAGING DAEMON (2)

- **Scrittura in Memoria Non Volatile:** Se le pagine sono state modificate, vengono scritte in memoria non volatile.
 - I contenuti precedenti delle pagine vengono conservati, permettendo un eventuale rapido ripristino.
- **Implementazione con «Clock a Due Lancette»**
 - La **lancetta anteriore** (gestita dal paging daemon) **avanza scrivendo le pagine sporche in memoria non volatile** e procede senza azioni ulteriori sulle pagine pulite.
 - La lancetta posteriore si occupa della sostituzione delle pagine
 - maggiore probabilità di trovare pagine pulite (grazie al lavoro del paging daemon).



DIMENSIONE DELLE PAGINE E BILANCIO DEI FATTORI

- **Selezione Dimensione Pagine:** I sistemi operativi possono selezionare la dimensione delle pagine
 - Esempio: unendo due pagine da 4096 byte per formarne una da 8 KB.
- **Fattori a Favore di Pagine Piccole:** Riducono la frammentazione interna (spazio sprecato nelle pagine parzialmente vuote) e l'utilizzo di memoria
 - un programma potrebbe richiedere meno memoria con pagine più piccole.
- **Svantaggi delle Pagine Piccole:** Richiedono **tabelle delle pagine più grandi** (più voci) e possono aumentare il tempo e lo spazio necessario per il trasferimento di dati e la gestione della memoria.



DIMENSIONE OTTIMALE DELLE PAGINE E THP

- **Dimensione Ottimale:** Determinata equilibrando frammentazione interna (favorevole a pagine più grandi) e overhead della tabella delle pagine (favorevole a pagine più piccole).
 - Vedi slide successiva
- **Pagine di Diverse Dimensioni:** Alcuni sistemi operativi utilizzano pagine di diverse dimensioni per parti diverse del sistema (ad es., pagine grandi per il kernel).
- **Transparent Huge Pages (THP):** Tecnica per utilizzare pagine di grandi dimensioni ottimizzando l'uso della memoria, spostando la memoria del processo per creare intervalli contigui.



CALCOLO DELLA DIMENSIONE OTTIMALE DELLE PAGINE

- **Parametri Considerati:**

- Dimensione media del processo: s byte (esempio 1MB).
- Dimensione della pagina: p byte (da calcolare).
- Dimensione di ogni voce nella tabella delle pagine: e byte (esempio 4 o 8 byte).

- **Calcolo Overhead:**

- Numero di pagine per processo: $\approx s/p$.
- Spazio occupato nella tabella delle pagine: $s \cdot e / p$ byte.
- Memoria sprecata per frammentazione interna nell'ultima pagina: $p/2$.
- **Fenomeno dell'Ultima Pagina:** Per qualsiasi processo, l'ultima pagina di memoria allocata potrebbe non essere completamente riempita. Esempio: un processo richiede 10.5 KB di memoria la pagina è di 4 KB, il sistema dovrà allocare 3 pagine lasciando 1.5 KB di spazio inutilizzato nell'ultima pagina.

- **Overhead totale:** $se/p + p/2$:

- Il primo termine (tabella delle pagine) aumenta con pagine più piccole.
- Il secondo termine (frammentazione interna) aumenta con pagine più grandi.
- **L'ottimo si trova bilanciando questi due fattori.**



CALCOLO DELLA DIMENSIONE OTTIMALE DELLE PAGINE (2)

Overhead totale: $se/p + p/2$

- **Formula per la Dimensione Ottimale delle Pagine:**

- Derivata della funzione di overhead rispetto a p uguagliata a zero: $-se/p^2 + 1/2 = 0$.
- Dimensione ottimale delle pagine: $p = \sqrt{2se}$.
- Esempio: Per $s = 1$ MB e $e = 8$ byte, p ottimale è 4 KB.

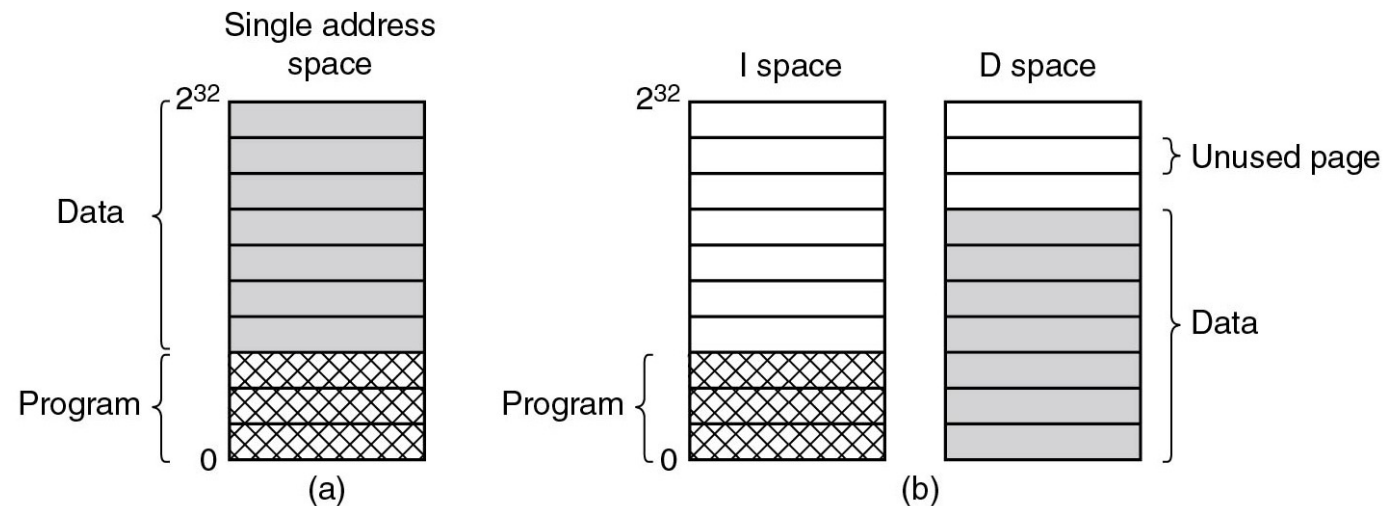
- **Variazione nelle Dimensioni delle Pagine:**

- Gamma tipica in computer commerciali: da 512 byte a 64 KB.
- Dimensione comune attuale: 4 KB



PROBLEMI DI PROGETTAZIONE: SPAZI SEPARATI PER ISTRUZIONI E DATI

- La maggior parte dei computer ha un unico spazio di indirizzamento condiviso da programma e dati. In passato alcuni sistemi avevano uno spazio di indirizzamento separato per istruzioni e dati.



- Al giorno d'oggi si vedono ancora spazi Istruzioni e Dati separati nelle cache, nei TLB, Cache L1
 - Dove lo spazio è poco, si tende a separare istruzioni (più importanti) dai dati.



CONDIVISIONE DELLE PAGINE NEI SISTEMI MULTIPROGRAMMATI

- **Motivazione della Condivisione**

- E' comune che **molti utenti eseguano lo stesso programma o utilizzino le stesse librerie.**
- **Condividere pagine di memoria** tra questi processi è più **efficiente** che mantenerne copie separate.

- **Tipi di Pagine Condivisibili**

- Le **pagine di sola lettura**, come il testo dei programmi: **SI**
- Le pagine dei dati: generalmente **NO**

- Per facilitare la condivisione è meglio separare spazi di indirizzo in:

- **I-space:** Istruzioni
- **D-space:** Dati

- **Processi diversi** possono utilizzare la **stessa tabella delle pagine per l'I-space** ma tabelle diverse per il D-space.

- **Implementazione e Scheduling:** con ciascun processo ha puntatori sia all'I-space che al D-space.
- Lo scheduler utilizza questi puntatori per impostare l'MMU.



NON E' TUTTO ORO: GESTIONE DELLE PAGINE CONDIVISE E COPY ON WRITE

- **Problemi con Pagine Condivise**

- La **rimozione di un processo** da memoria può causare **numerosi page fault in un altro processo** che condivide le stesse pagine.
- È cruciale sapere se le pagine sono ancora in uso per evitare la loro liberazione accidentale.

- **Condivisione dei Dati:** Più complessa rispetto alla condivisione del codice!

- Ad esempio, in UNIX, dopo una `fork`, genitore e figlio condividono sia il testo che i dati, inizialmente come sola lettura.

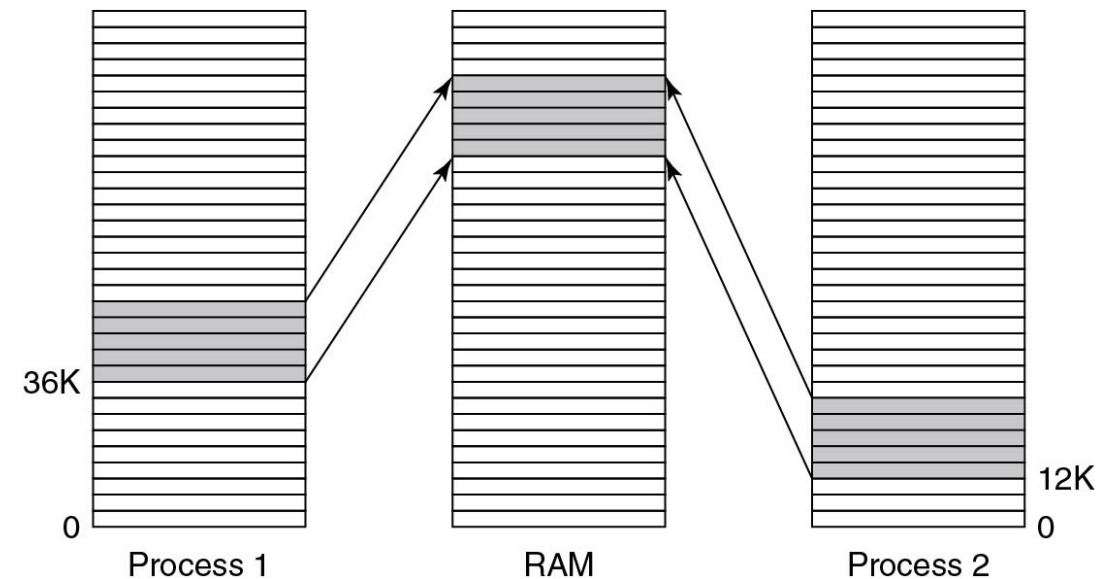
- **Copy on Write (Copia in Caso di Scrittura):** Se un processo modifica i dati, si genera una trap, e viene creata una copia della pagina modificata.

- Entrambe le copie diventano poi modificabili (READ/WRITE).
- **Questo metodo evita la copia di pagine che non vengono mai modificate.**
- Estremamente efficiente per evitare la proliferazione di pagine



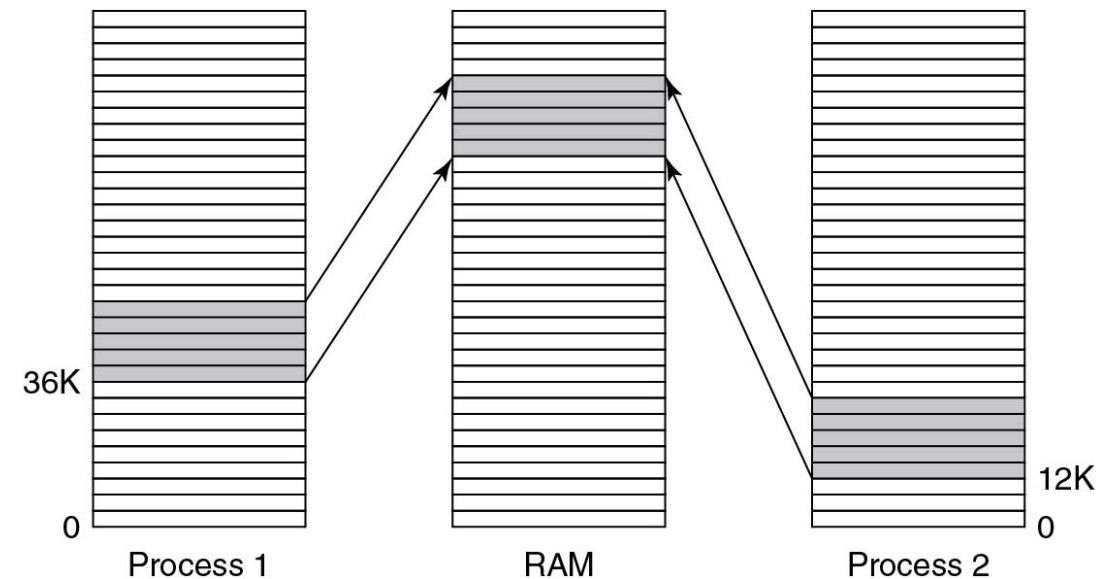
LIBRERIE CONDIVISE - PRINCIPI E FUNZIONAMENTO

- **Condivisione su Ampia Scala:**
 - I SO condividono automaticamente tutte le pagine di testo di un programma avviato più volte.
 - Per evitare problemi, meglio pagine in sola lettura
- **Copy on Write per Dati:** Se un processo modifica una pagina di dati condivisa, occorre applicare "*copy on write*".
- **Librerie condivise - Dynamic Link Libraries (DLLs):** Usate per ridurre l'ingombro di grandi librerie comuni.
 - **Vantaggi:** Risparmio di spazio



LIBRERIE CONDIVISE - PRINCIPI E FUNZIONAMENTO

- **Problema di Indirizzamento:** Le librerie condivise possono essere posizionate a indirizzi diversi nei vari processi.
 - Questo **impedisce l'uso di indirizzi assoluti** nelle istruzioni.
- **Soluzione Compilativa:** Le librerie condivise vengono **compilate con indirizzi relativi** anziché assoluti.
 - le istruzioni usano offset relativi piuttosto che puntare a indirizzi specifici.



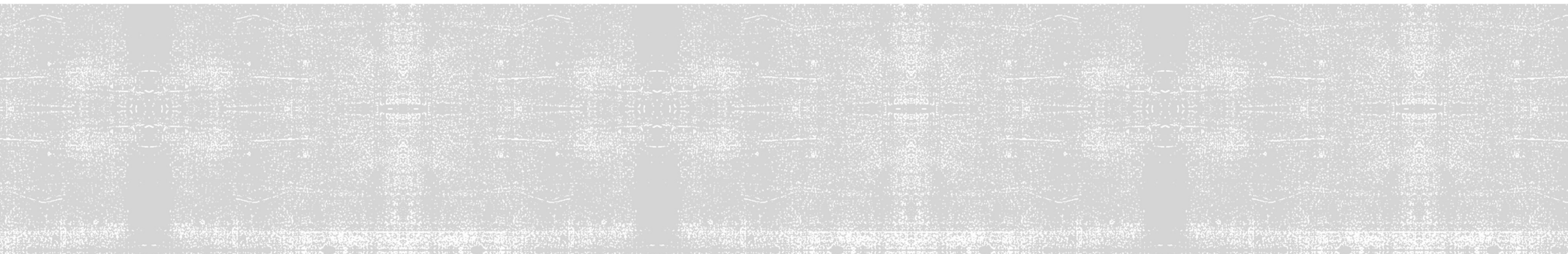
FILE «MAPPATI» IN MEMORIA E IL LORO IMPIEGO

- **Concetto:** I file mappati consentono a un processo di mappare un file all'interno del proprio spazio di indirizzi virtuali.
 - **Funzionamento:** Alla mappatura, **nessuna pagina viene caricata immediatamente**.
 - Sono paginate su richiesta dalla memoria non volatile, man mano che vengono "toccate"
 - **Scrittura su File:** Quando il **processo termina** o la mappatura è eliminata, **tutte le pagine** modificate vengono **riscritte sul file**.
- **Modello I/O Alternativo:** Offre un modo diverso di eseguire I/O, permettendo di accedere al file come se fosse un grande array di caratteri in memoria.
- **Comunicazione tra Processi:** Se più processi mappano lo stesso file contemporaneamente, possono comunicare attraverso questa memoria condivisa.
 - Le modifiche apportate da un processo sono immediatamente visibili agli altri.





DETTAGLI IMPLEMENTATIVI



PROBLEMI DI IMPLEMENTAZIONE DELLA MEMORIA VIRTUALE

- **Sfide nell'Implementazione:**

- Scelta tra algoritmi teorici (es. Seconda Chance, Aging) e pratiche operative (allocazione locale/globale, paginazione a richiesta/prepaginazione).
- Gestione di problemi pratici di implementazione della memoria virtuale.

- **Attività del Sistema Operativo nella Paginazione:**

- **Creazione del Processo:**

- Determinare le dimensioni iniziali del programma e dei dati.
- Creare e inizializzare la tabella delle pagine.
- Allocare spazio nella memoria non volatile per lo scambio.
- Inizializzare l'area di scambio e registrare informazioni nella tabella dei processi.

- **Esecuzione del Processo:**

- Azzerrare la MMU e, se necessario, svuotare il TLB.
- Rendere attiva la tabella delle pagine del processo.
- **Pre-paginazione:** Facoltativamente, caricare alcune pagine in memoria per ridurre i page fault iniziali.



GESTIONE DEI PAGE FAULT E CHIUSURA DEL PROCESSO

- **Gestione dei Page Fault:**
 - **Determinare l'indirizzo virtuale** che ha causato il **fault**.
 - **Trovare la pagina necessaria** nella memoria non volatile.
 - **Scegliere un frame disponibile**, eventualmente sfrattando pagine vecchie.
 - **Caricare la pagina nel frame** e ripristinare il contatore del programma.
- **Chiusura del Processo:**
 - **Rilasciare la tabella delle pagine**, le pagine in memoria e lo spazio su disco/SSD.
 - **Gestire le pagine condivise con altri processi**, rilasciandole solo dopo l'ultimo utilizzo.



PAGE FAULT IN 10 PASSI:

A. INIZIO DELLA SEQUENZA

1. **Trap nel Kernel da Parte dell'Hardware:**

- L'hardware esegue una trap nel kernel, salvando il contatore del programma nello stack.
- Informazioni sull'istruzione corrente salvate nei registri speciali della CPU.

2. **Avvio Routine di Servizio Interrupt:**

- Viene eseguita una routine in assembly per salvare i registri e altre informazioni volatili.
- Invocazione del gestore dei page fault.

3. **Identificazione della Pagina Virtuale Necessaria:**

- Il sistema operativo determina quale pagina virtuale manca.
- Se non disponibile dai registri hardware, recupero e analisi dell'istruzione dal contatore di programma.



PAGE FAULT IN 10 PASSI

B. GESTIONE E RISOLUZIONE

4. **Verifica Validità Indirizzo e Protezione:**

- Controllo della validità dell'indirizzo e coerenza della protezione con l'accesso.
- Se invalide, invio di un segnale di errore o terminazione del processo.

5. **Rilascio di un Frame Libero:**

- Se non ci sono frame liberi, esecuzione di un algoritmo di sostituzione delle pagine.
- Se la pagina è "sporca", viene schedulata per la scrittura in memoria non volatile e il processo è sospeso.

6. **Caricamento della Pagina Richiesta:**

- Una volta liberato (o scritto in memoria non volatile), il frame viene usato per caricare la pagina necessaria da disco o SSD.
- Durante il caricamento della pagina, il processo in page fault è ancora sospeso e viene eseguito, se disponibile, un altro processo utente.



PAGE FAULT IN 10 PASSI

C. CONCLUSIONE E RIPRESA

7. **Aggiornamento delle Tabelle delle Pagine:**

- Al completamento del trasferimento dal supporto non volatile, le tabelle delle pagine vengono aggiornate per riflettere la nuova posizione della pagina.
- Il frame viene contrassegnato come disponibile.

8. **Ripristino dell'Istruzione in Errore:**

- L'istruzione in errore è riportata allo stato che aveva all'inizio
- Il contatore di programma è ripristinato in modo da puntare a quell'istruzione.

9. **Ripresa del Processo in Errore:**

- Il processo precedentemente in errore viene schedato per l'esecuzione.
- Ritorno alla routine in assembly che lo aveva interrotto.

10. **Ricarica dei Registri e Ritorno allo Spazio Utente:**

- La routine di servizio ricarica i registri e le informazioni di stato.
- Il controllo ritorna allo spazio utente per continuare l'esecuzione da dove era stata interrotta.



BLOCCARE LE PAGINE IN MEMORIA DURANTE L'I/O

Scenario:

- Un processo invia una richiesta di lettura da un file o dispositivo in un buffer nel suo spazio di indirizzi.
- Mentre attende il completamento dell'I/O, può essere sospeso per permettere l'esecuzione di un altro processo.

Problema con Page Fault

- Se il secondo processo genera un page fault, esiste il rischio che la pagina contenente il buffer di I/O venga selezionata per essere rimossa
- Se avviene un trasferimento DMA (Direct Memory Access) su quella pagina, la rimozione potrebbe causare scritture errate nei dati.

Soluzione: *Pinning delle Pagine*

- Le pagine utilizzate per l'I/O vengono "bloccate" o "pinned" (fissate) in memoria, prevenendo la loro rimozione.
- Questo approccio assicura che le operazioni di I/O possano procedere senza interruzioni.

Alternativa: Gestione I/O nei Buffer del Kernel

- Un'altra strategia è gestire l'I/O nei buffer del kernel e poi copiare i dati nelle pagine utente.
- Questo metodo richiede una copia aggiuntiva dei dati, potenzialmente rallentando il processo.



MEMORIA SECONDARIA E GESTIONE DELLO SCAMBIO

Ma dove viene messa una pagina quando viene spostata nella memoria non volatile dopo essere stata «paginata fuori» dalla memoria?

- **Gestione dello Spazio di Scambio (*file o partizione di swap*)**
 - Il sistema operativo prevede **una partizione speciale o dispositivo separato per lo scambio**, come nei sistemi UNIX.
 - **Un'area del disco/SSD strutturata in maniera differente dal file system** usato per memorizzare file e cartelle (vedi lezioni successive)
 - Partizione di scambio **con file system semplificato**, utilizzando numeri di blocchi relativi
- **Allocazione in Memoria di Scambio:**
 - All'avvio, allocazione di spazio in partizione di scambio pari alla dimensione del processo.
 - Gestione come lista di parti libere (anche se esistono miglioramenti, vedi Capitolo 10 del libro).
- **Associazione Processo-Area di Scambio:**
 - Ogni **processo ha un'area di scambio in memoria non volatile**.
 - L'indirizzo in cui scrivere una pagina è calcolato sommando l'offset della pagina al suo spazio virtuale all'inizio dell'area di scambio.



STRATEGIE DI PAGINAZIONE E OTTIMIZZAZIONI

- **Gestione di Crescita dei Processi:**

- Riserva di aree separate per testo, dati e stack, per gestire l'espansione dei processi.

- **Alternativa di Allocazione Dinamica:**

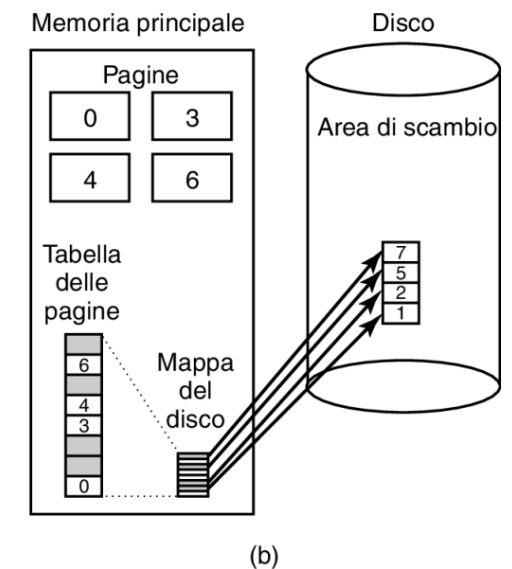
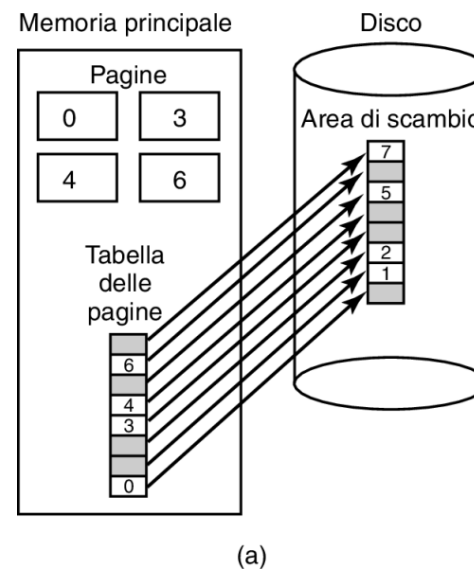
- Allocazione dello spazio su disco/SSD al momento dello scambio di ogni pagina.
- Tavola per ogni processo che indica la posizione di ogni pagina in memoria non volatile.

- **Esempi di Gestione Paginazione:**

- a) Paginazione in area di scambio statica. Ogni pagina ha una posizione fissa su disco.
- b) Salvataggio dinamico delle pagine. Indirizzo su disco scelto al momento dello scambio.

- **Ottimizzazioni su File System:**

- Uso di file pre-allocati in file system normale (es. Windows).

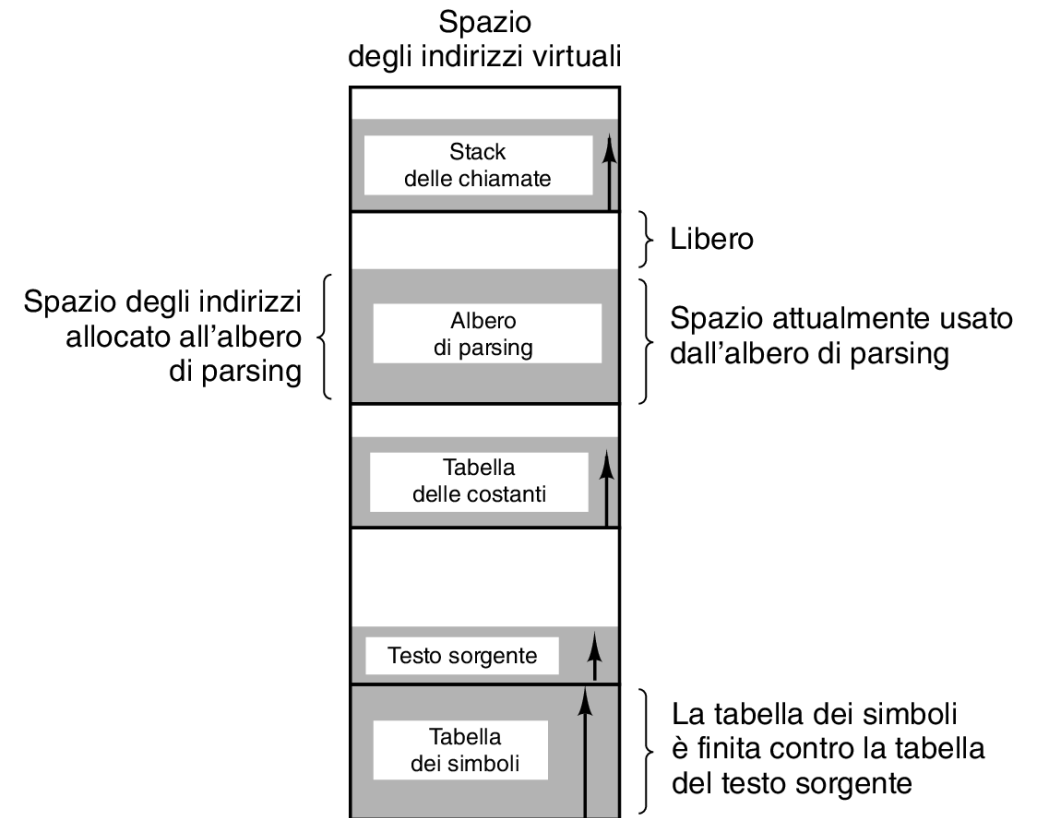




LA SEGMENTAZIONE

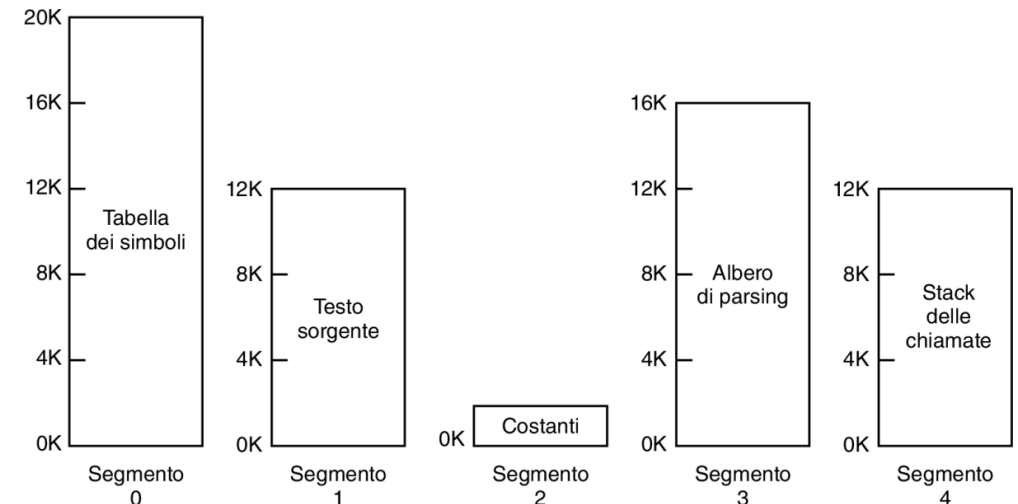
MEMORIA MONODIMENSIONALE VS. SEGMENTAZIONE

- In un sistema di **memoria monodimensionale**, gli **indirizzi virtuali**
 - vanno da 0 a un massimo
 - disposti in modo lineare e contiguo.
- Può risultare **problematica** in alcuni scenari, come nella compilazione
 - **diverse tabelle** (testo sorgente, tabella dei simboli, costanti, albero di parsing, stack) **crescono dinamicamente** e in modo imprevedibile.
- La **crescita** di una tabella **può causare sovrapposizioni con altre**, creando difficoltà nella gestione della memoria
 - richiedendo una riorganizzazione complessa.



LA SEGMENTAZIONE

- La **segmentazione** introduce l'idea di spazi di **indirizzi virtuali multipli e indipendenti**, chiamati segmenti.
 - **Ciascun segmento ha una sequenza lineare** di indirizzi, iniziando da 0 fino a un massimo variabile.
 - I segmenti **possono avere lunghezze diverse** e la loro **dimensione può cambiare** durante l'esecuzione.
- Questa struttura **consente** ai segmenti **di crescere o ridursi senza interferire** l'uno con l'altro.
- Per specificare un indirizzo in memoria segmentata, si usa un indirizzo a due parti:
 - numero di segmento
 - indirizzo nel segmento.



Una memoria segmentata consente a ogni tabella di crescere o di ridursi indipendentemente dalle altre tabelle.



VANTAGGI DELLA SEGMENTAZIONE

- **Flessibilità:** I segmenti possono crescere o ridursi in modo indipendente l'uno dall'altro.
 - Esempio, lo stack del compilatore può espandersi o contrarsi senza influenzare le altre tabelle.
 - Ciò elimina il problema di collisione presente nella memoria monodimensionale.
- **Semplificazione del Linking:** Se ogni procedura occupa un segmento separato, il **linking di procedure** diventa molto più **semplice**.
 - In caso di modifiche, non è necessario aggiornare gli indirizzi di altre procedure non correlate.
- **Condivisione e Protezione:** La segmentazione **facilita la condivisione di risorse**, come librerie condivise, tra processi diversi.
 - Offre anche la possibilità di **applicare vari livelli di protezione ai segmenti** (es. solo lettura, solo esecuzione), migliorando la sicurezza e aiutando a identificare errori.



MEMORIA SEGMENTATA VS. PAGINATA

- **Confronto:**

- La segmentazione suddivide la memoria in segmenti con indirizzi lineari.
- La paginazione divide la memoria in pagine di dimensioni fisse.

La **segmentazione offre maggiore flessibilità** e gestione delle strutture dati rispetto alla paginazione, ma può essere **più complessa da implementare**.

- **Esempi Pratici**

- **Uso nel Compilatore:** Le varie tabelle utilizzate durante la compilazione possono essere allocate in segmenti separati
 - Le tabelle possono crescere indipendentemente e di essere gestite più efficacemente.
- **Librerie Condivise:** In un sistema segmentato, una libreria grafica può essere posta in un segmento e condivisa tra più processi, risparmiando spazio e migliorando l'efficienza.



PAGINAZIONE VS SEGMENTAZIONE

Considerazione	Paginazione	Segmentazione
Il programmatore deve sapere che questa tecnica è in uso?	NO	SI
Quanti spazi di indirizzi lineari ci sono?	1	Molti
Lo spazio degli indirizzi totale può superare la dimensione della memoria fisica?	SI	SI
Le procedure e i dati possono essere distinti e protetti separatamente?	NO	SI
Le tabelle la cui dimensione varia possono essere disposte facilmente?	NO	SI
La condivisione delle procedure fra utenti è facilitata?	NO	SI
Perché fu inventata questa tecnica?	Per avere uno spazio degli indirizzi lineare grande senza dover acquistare ulteriore memoria fisica	Per consentire a programmi e dati di essere spezzati in spazi degli indirizzi logicamente indipendenti e per facilitare la condivisione e la protezione



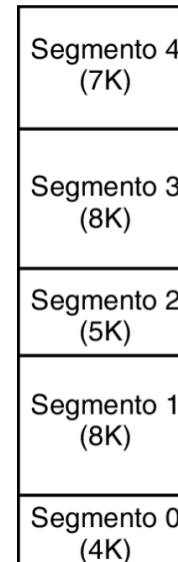
IMPLEMENTAZIONE DELLA SEGMENTAZIONE PURA: LE SFIDE

- **Paginazione VS Segmentazione:** Le pagine hanno dimensioni fisse, i segmenti no.
- **Evoluzione della Configurazione della Memoria:**
 - a) **memoria fisica** con cinque segmenti.
 - b) il segmento 1 è rimosso e il segmento 7, che è più piccolo, viene messo al suo posto
 - Fra il segmento 7 e il segmento 2 c'è dello spazio inutilizzato, cioè vuoto.
 - c) il segmento 4 è sostituito dal segmento 5
 - d) il segmento 3 è rimpiazzato dal segmento 6

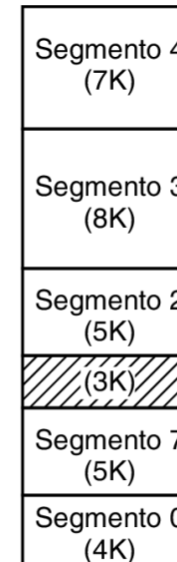
Frammentazione Esterna

("Checkerboarding"): Dopo un po', la memoria sarà suddivisa in parti, qualcuna contenente segmenti e altre spazi vuoti.

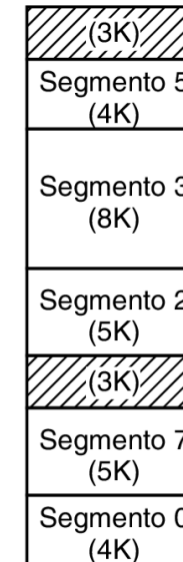
- e) Può essere risolto tramite la **compattazione**.



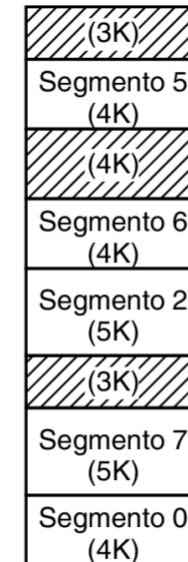
(a)



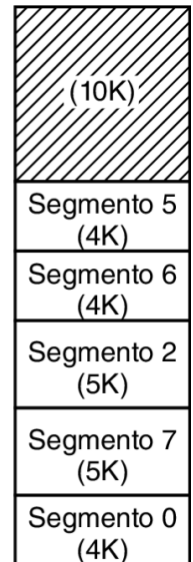
(b)



(c)



(d)

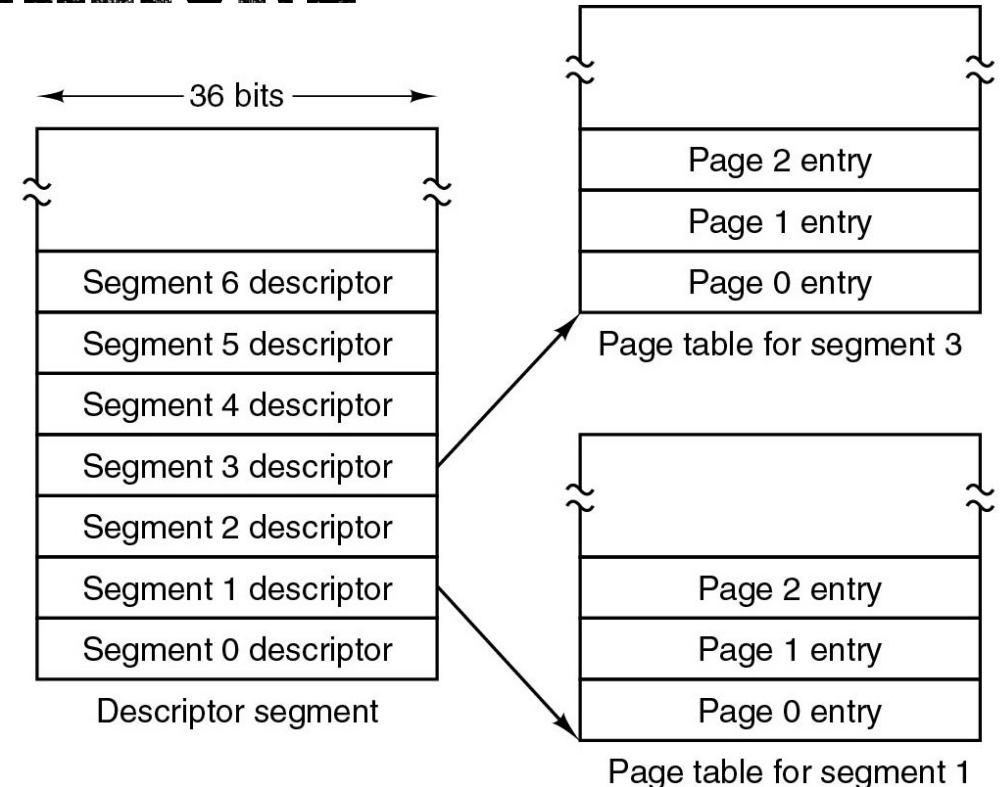


(e)



MULTICS: PIONIERE DELLA SEGMENTAZIONE E PAGINAZIONE

- **Breve storia di MULTICS:** Progetto di ricerca del M.I.T. diventato operativo nel 1969, influente fino al 2000.
- **Rilevanza storica:** Impatto su
 - UNIX,
 - architettura x86
 - TLB
- **Architettura della Memoria:** MULTICS forniva fino a 2^{18} segmenti per programma, con ogni segmento lungo fino a 65.536 parole ($= 2^{16}$).
- **Approccio alla Memoria Virtuale**
 - I **segmenti** venivano trattati **come spazi di memoria virtuale indipendenti e paginati** per gestire meglio lo spazio in memoria.

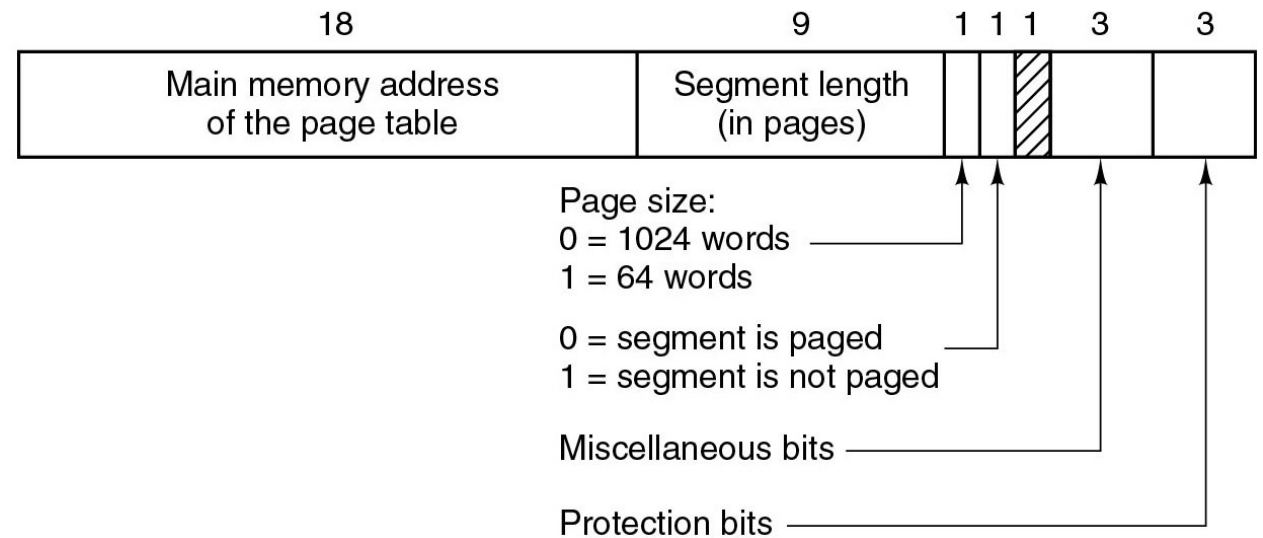


Memoria virtuale del MULTICS: Il segmento dei descrittori punta alle tabelle delle pagine.



GESTIONE DELLA MEMORIA E DEI SEGMENTI IN MULTICS

- **Struttura dei Segmenti:** Ogni segmento trattato come spazio virtuale indipendente e paginato.
- **Tabella dei Segmenti:** Descrittori per ogni segmento, indicando se sono in memoria e collegamenti alle tabelle delle pagine.
- **Funzionamento dei Descrittori:** Descrittori con puntatori, dimensione del segmento, bit di protezione, e altre informazioni.



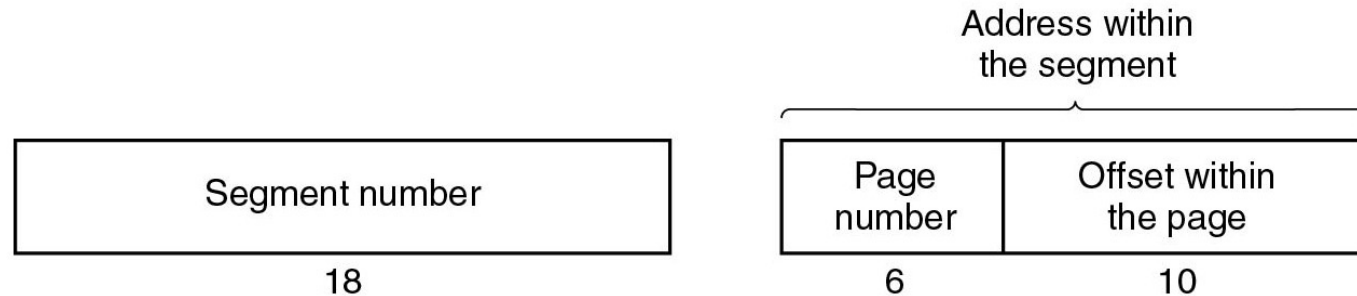
Memoria virtuale del MULTICS: Un descrittore di segmento. I numeri indicano le lunghezze dei campi.

- Nota che la somma di tutti i campi è 36 (il numero di bit nel descriptor segment, vedi slide precedente, è)



GESTIONE DEGLI INDIRIZZI

- Indirizzi costituiti da due parti - segmento e indirizzo nel segmento - con suddivisione in numero di pagina e parola nella pagina.

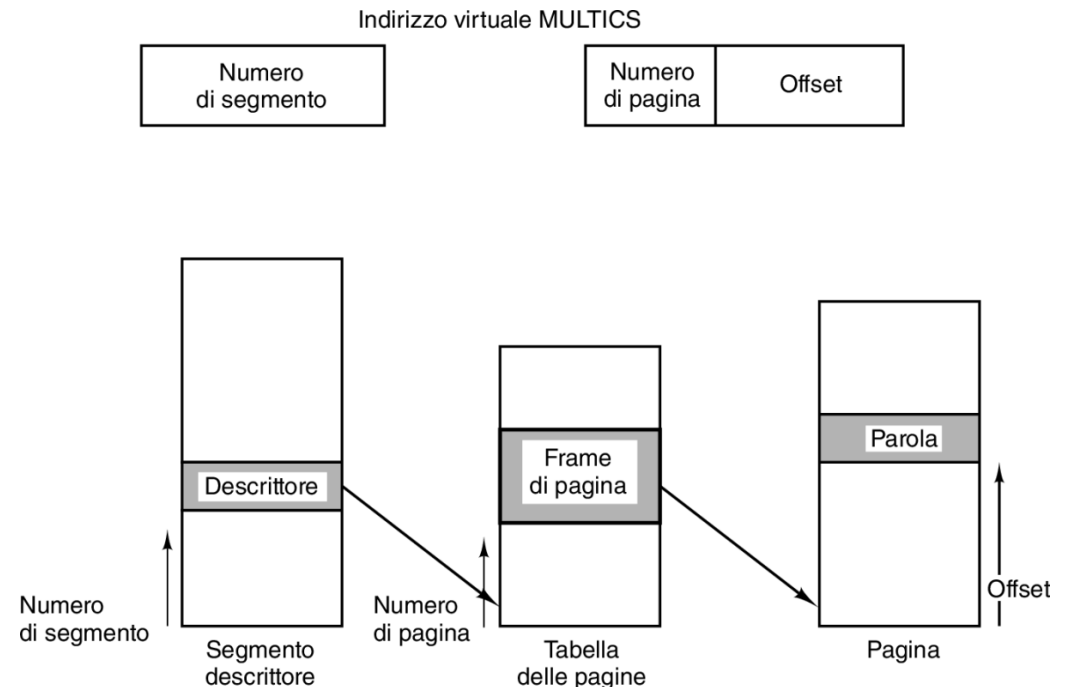


Un indirizzo virtuale MULTICS a 34 bit.



CONVERSIONE DI UN INDIRIZZO MULTICS

1. Il numero del segmento usato per **trovare il descrittore del segmento.**
2. Si **verificava se la tabella delle pagine** del segmento in memoria.
 - Prevenire eventuali errori
3. Si **esamina la voce della pagina virtuale**
 - Se la pagina non in memoria: page fault
 - Se in memoria, dalla voce della tabella delle pagine viene estratto l'indirizzo dell'inizio della pagina nella memoria principale.
4. Si **ottiene l'indirizzo nella memoria principale** in cui era localizzata la parola aggiungendo l'offset all'origine della pagina.
5. **Avviene la lettura o il salvataggio.**



OTTIMIZZAZIONE DELLE PRESTAZIONI IN MULTICS

- **Uso del TLB (Translation Lookaside Buffer):**

- Primo sistema ad utilizzare un TLB per ottimizzare l'accesso alla memoria.
- TLB con 16 parole per velocizzare la ricerca degli indirizzi.

- **Prestazioni e Set di Lavoro:**

- Programmi con set di lavoro minori del TLB raggiungono una maggiore efficienza.
- Gestione di errori del TLB per set di lavoro più grandi.

Comparison field		Page frame	Protection	Age	Is this entry used? ↓
Segment number	Virtual page				
4	1	7	Read/write	13	1
6	0	2	Read only	10	1
12	3	1	Read/write	2	1
					0
2	1	0	Execute only	7	1
2	2	12	Execute only	9	1

Una versione semplificata del TLB MULTICS. L'esistenza di due dimensioni di pagina ha reso più complicato il TLB vero e proprio.



EVOLUZIONE E DECLINO DELLA SEGMENTAZIONE IN INTEL X86

- **Eredità di MULTICS nell'x86:**

- Fino all'x86-64, Intel x86 rifletteva il modello di MULTICS, combinando segmentazione e paginazione.
- 16.000 segmenti indipendenti per x86, ognuno fino a 1 miliardo di parole a 32 bit.

- **Transizione all'x86-64:**

- Nell'x86-64, la segmentazione diventa obsoleta e viene mantenuta solo per compatibilità.

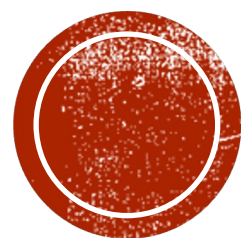
- **Ragioni del Cambiamento:**

- Sistemi operativi chiave come UNIX e Windows non adottano la segmentazione per questioni di portabilità.
- Intel elimina la segmentazione per ottimizzare lo spazio del chip nelle CPU a 64 bit.

- **Riflessioni sull'x86:**

- L'architettura x86 è lodata per il suo equilibrio tra paginazione, segmentazione e compatibilità con versioni precedenti.





IL COMANDO free

UTILIZZO DEL COMANDO `free` IN LINUX PER MONITORARE LA MEMORIA

- **Funzione del Comando `free`:** Fornisce dettagli sull'utilizzo della memoria fisica e dello swap nel sistema Linux.
- **Output del Comando:**
 - `total`: Quantità totale di memoria fisica disponibile.
 - `used`: Memoria attualmente in uso.
 - `free`: Memoria libera/non utilizzata.
 - `shared`: Memoria condivisa (obsoleta, presente solo per compatibilità).
 - `buff/cache`: Memoria utilizzata per buffer/cache/**slab**, recuperabile se necessario.
 - `available`: Stima della memoria disponibile per nuove applicazioni, considerando buffer e cache.



OPZIONI E INTERPRETAZIONE DELL'OUTPUT DI FREE

- **Formato Leggibile:** Utilizzo dell'opzione `-h` per visualizzare i dati in formato megabyte o gigabyte.
- **Specifica delle Unità di Misura:**
 - Opzioni come `-b`, `--kilo`, `--mega`, `--giga` per specificare l'unità di misura (byte, kilobyte, megabyte, gigabyte).
- **Visualizzazione dei Totali:** Opzione `-t` per mostrare il totale della memoria e dello swap.
- **Aggiornamento Continuo:** Opzione `-s` per aggiornamenti continui ogni tot secondi, simile al comando `watch`.
- **Nota:** Il comando `free` è essenziale per comprendere come la memoria viene utilizzata nel sistema, identificando potenziali spazi liberi per nuove applicazioni e monitorando l'efficienza del sistema in termini di gestione della memoria.

