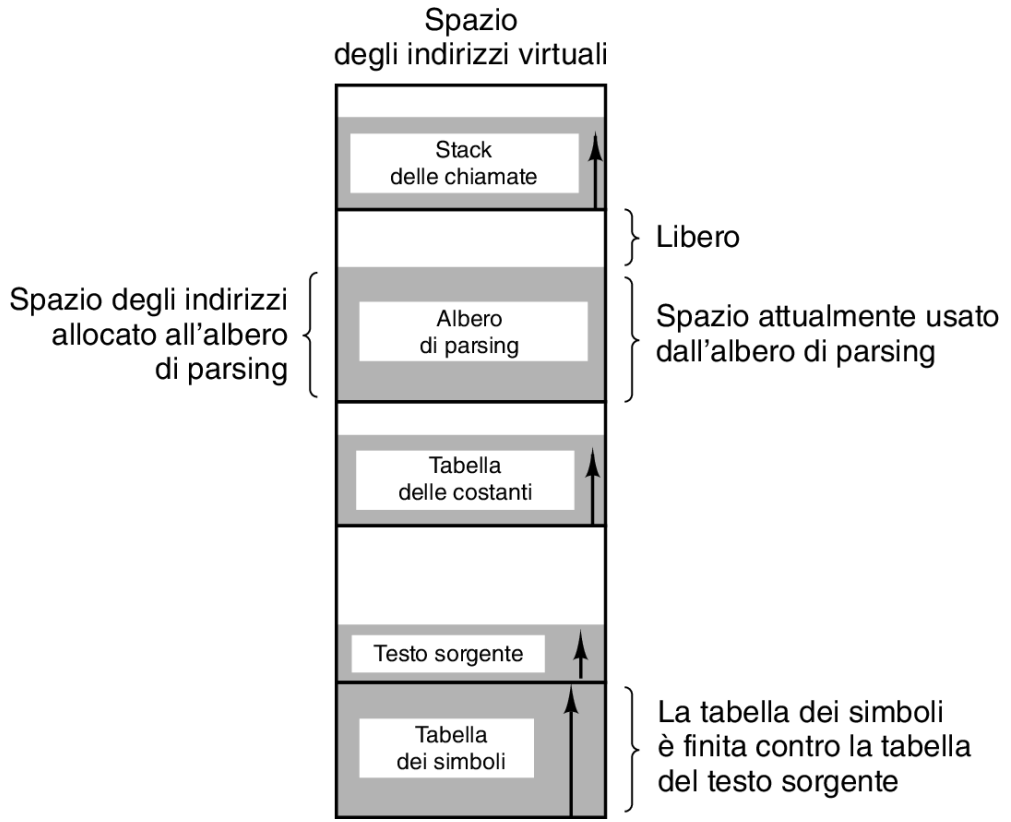
# LA SEGMENTAZIONE

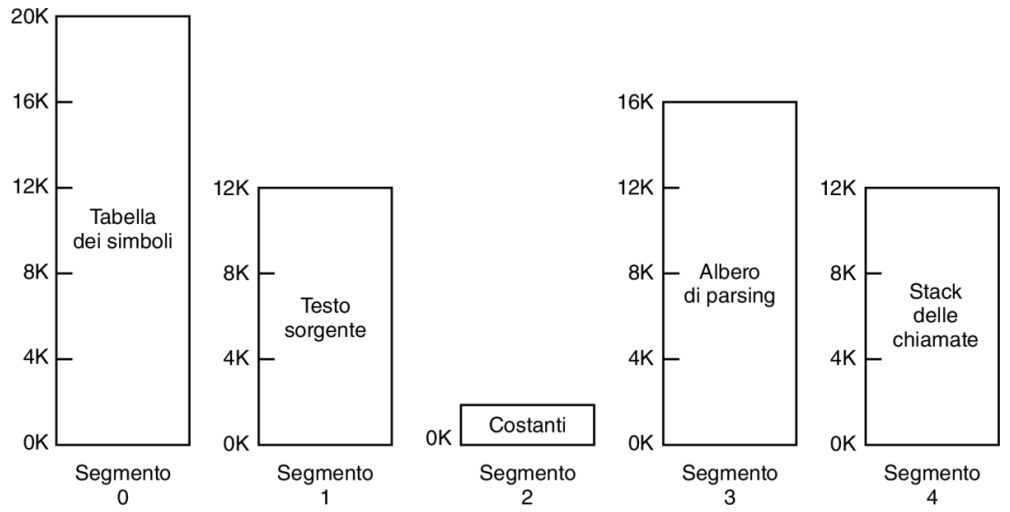
# MEMORIA MONODIMENSIONALE VS. SEGMENTAZIONE

- 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 |

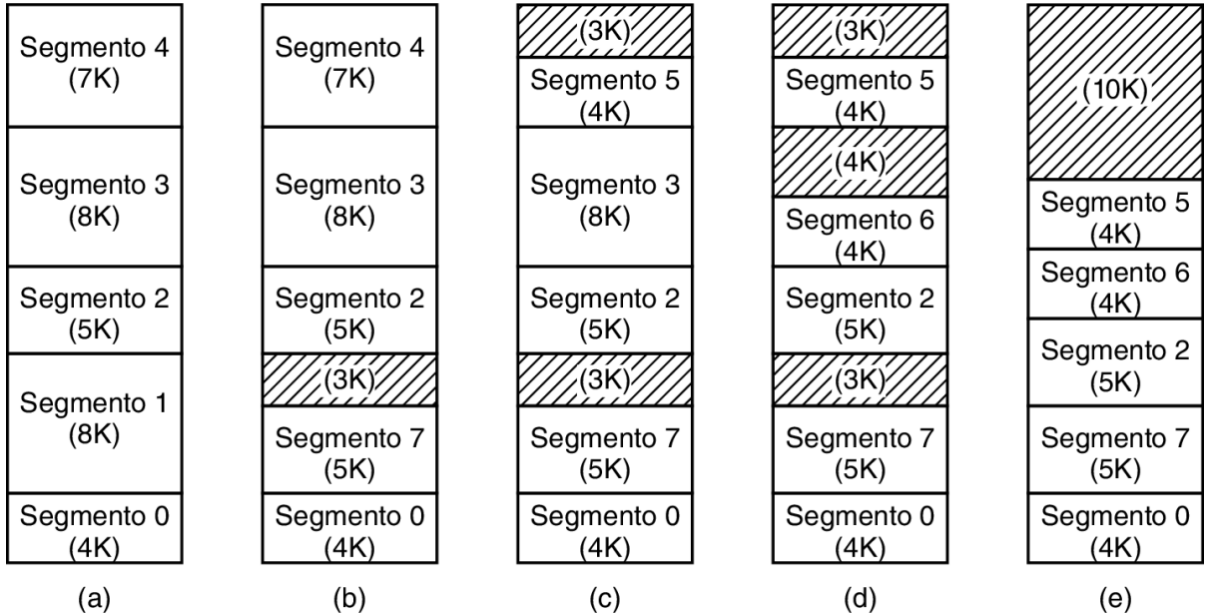


# IMPLEMENTAZIONE DELLA SEGMENTAZIONE PURA: LE SFIDE

- **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

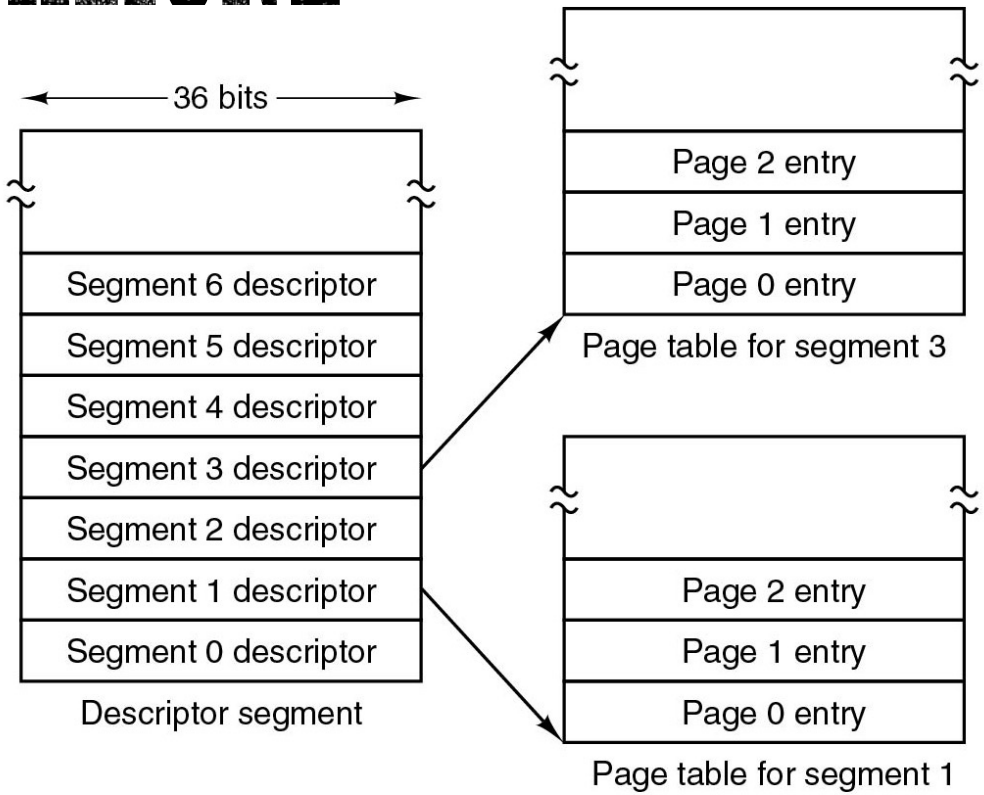
  **Frammentazione Esterna ("Checkerboarding"):** Dopo un po', la memoria sarà suddivisa in parti, qualcuna contenente segmenti e altre spazi vuoti.

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



# MULTICS: PIONIERE DELLA SEGMENTAZIONE E PAGINAZIONE

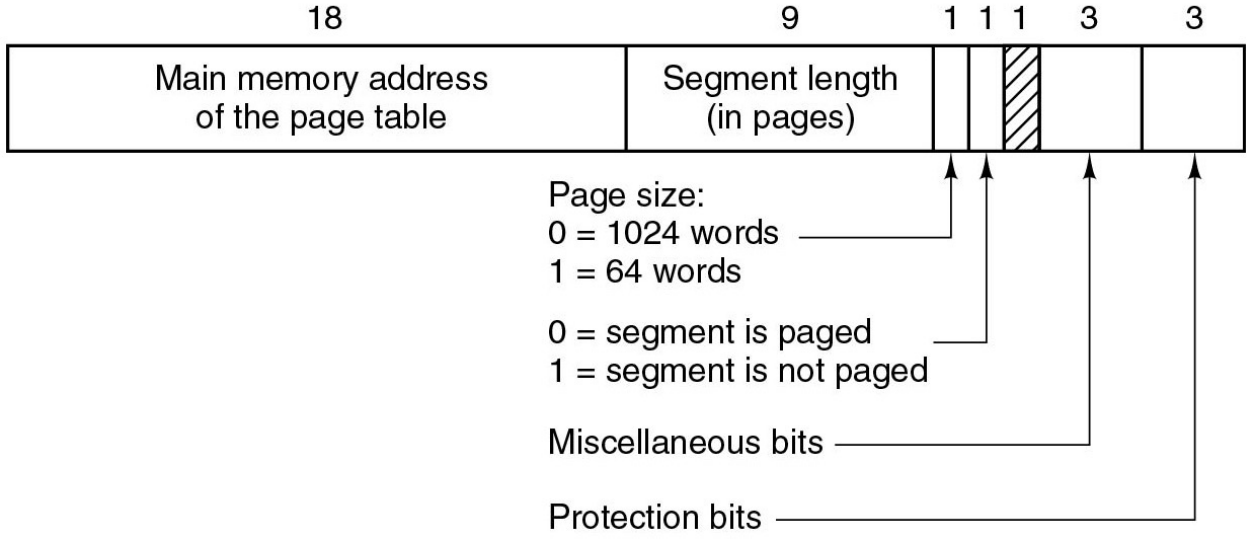
- **Breve storia di MULTICS**: Progetto di ricerca del M.I.T. diventato operativo nel 1969, influente fino al 2000.
- **Rilevanza storica**: Impatto su
  - UNIX,
  - architettura x86
  - TLB
- **Architettura della Memoria:** MULTICS forniva fino a $2^{18}$ segmenti per programma, con ogni segmento lungo 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.

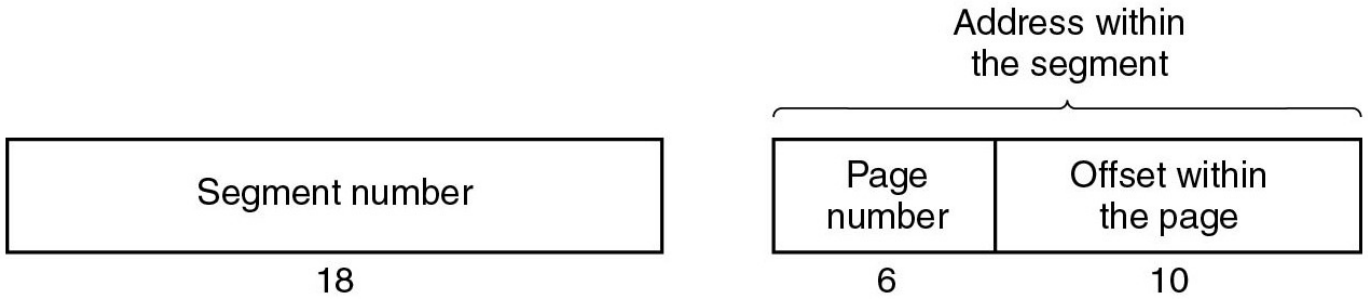



Memoria virtuale del MULTICS: Un descrittore di segmento. I numeri indicano le lunghezze dei campi.

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

# GESTIONE DEGLI INDIRIZZI

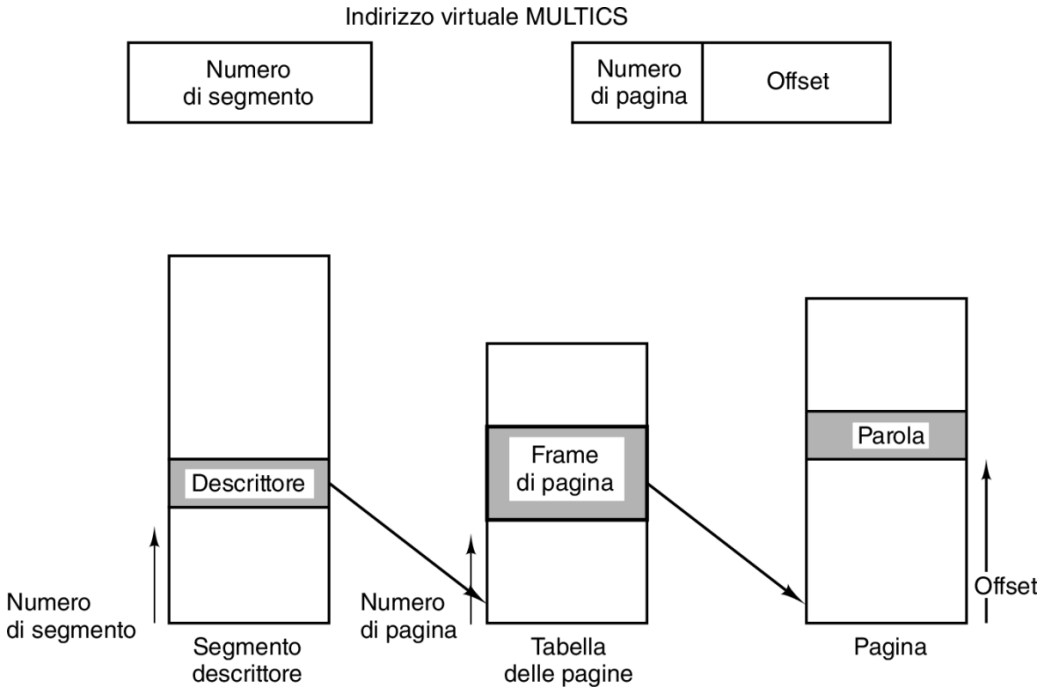
- Indirizzi costituiti da due parti - segmento e indirizzo nel segmento - con suddivisione in numero di pagina e parola nella pagina.




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 | | | | | Is this entry used? |
|---|---|---|---|---|---|
| Segment number | Virtual page | Page frame | Protection | Age | |
| 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.