Segmentazione

La memoria virtuale illustrata fino ad ora è monodimensionale -> Gli indirizzi vanno da 0 a un certo indirizzo. Per alcuni problemi è meglio avere più spazi di indirizzi virtuali.

Compilatore

Durante la compilazione, il compilatore deve gestire diverse tabelle:

- Il testo sorgente
- La tabella dei simboli, contenente nomi e attributi di variabili
- La tabella di tutte le costanti
- L'albero di parsing che contiene l'analisi sintattica del programma
- Lo stack usato per le chiamate di procedure

Tutte le tabelle crescono continuamente, mentre l'ultima cresce e si riduce in modi imprevedibili.

Memoria monodimensionale --> Tabelle allocate in parti contigue dello spazio virtuale degli indirizzi.

Situazione

Programma con molte variabili, la tabella dei simboli si riempe, mentre ci potrebbe essere spazio libero nelle altre tabelle.

Problema

La crescita di una tabella può causare sovrapposizione con altre creando difficoltà nel gestire la memoria.

Dobbiamo trovare un modo per non obbligare il programmatore a gestire l'espasione e la contrazione delle tabelle.

Soluzione generia e ditretta -> Fornire alla macchina molti spazi degli indirizzi completamente indipendenti chiamati **segmenti**.

Segmento

Sequenza lineare di indirizzi da 0 ad un certo numero di lungehzza variabile per l'intera dimensione. Inoltre la lunghezza dei segmenti può variare durante l'esecuzione.

Questa struttura consente ai segmenti di crescere o ridursi senza interferire tra di loro.
Capita raramente che i segmenti si riempiano perché molto grandi.

Per specificare un indirizzo in questa memoria **bidimensionale** il programma deve fornire un indirizzo a due parti:

- **Numero di segmento.**
- **Indirizzo di segmento.**

I vantaggi della segmentazione sono:

1. Semplifica trattamento delle strutture che crescono e diminuiscono.
2. Semplificazione del linking di procedure compilate (Con segmenti separati che cominciano da 0).

Linking

Quando compiliamo diverse parti del programma queste possono essere trattate separatamente. Il sistema può collegarle facilmente tra loro perché ogni indirizzo parte da 0 nel suo segmento, quindi è semplice sapere dove comincia ogni funzione.
Inoltre se una procedura viene modificata non sarà necessario cambiare gli indirizzi delle altre procedure, poiché ogni segmento è indipendente.

3. Facilita la condivisione di risorse (Es: librerie condivise) fra molti processi. Inoltre fornisce la possibilità di applicare varai livelli di protezione ai segmenti, come solo la scrittura o lettura, migliorando la sicurezza e aiutando ad identificare errori.

Considerazione	Paginazione	Segmentazione
Il programmatore deve sapere che questa tecnica è in uso?	No	Si
Quanti spazi di indirizzi lineari ci sono?	1	Molti

Considerazione	Paginazione	Segmentazione
Lo spazio degli indirizzi totale può superare la dimensione della memoria fisica?	No	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 è facilitata?	No	Si
Perche fu invetata questa tecnica?	Per avere uno spazio degli indirizzi lineare grande senza acquistare 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 delle segmentazione pura

La differenza principale tra pagine e segmenti e che le pagine sono di dimensione fissa, mentre i segmenti non lo sono.
Avendo dimensioni diverse, quando andiamo a inserire e togliere alcuni segmenti della memoria, potremmo ritorvarci alcuni spazi occupati dai segmenti ed alcuni vuoti.
Questo fenomeno si chiama **Checkboarding o Frammentazione esterna** e lo si può risolvere tramite la **compattazione**.

Segmentazione con paginazione: MULTICS

Se i segmenti sono troppo grandi diventa impossibile mantenerli in memoria, da cui arriva l'idea di paginarli, il primo sistema a fare ciò è stato MULTICS.

> ⓘ **Breve storia su MULTICS**
>
> Era un progetto di ricerca del M.I.T diventato operativo nel 1969, influente fino al 2000.
> Ha avuto un forte impatto sullo sviluppo di altri sistemi e tecnologie come: UNIX, architettura x86, TLB.

MULTICS forniva ad ogni programma una memoria virtuale di 2^{18} segmenti, ognuno lungo fino a 65536 parole (da 36 bit).

L'idea dei progettisti era di trattare ==ogni segmento come una memoria virtuale e paginarlo ==combinando i vantaggi di paginazione e segmentazione.

Ogni programma MULTICS aveva una **tabella dei segmenti**, con un **descrittore** per segmento, quest'ultima segmentata e paginata a sua volta.
I descrittori di segmento indicano se il segmento è in memoria o meno, se una qualunque parte del segmento era in memoria, allora era considerato in memoria e quindi anche la tabella delle pagine era in memoria.

Segmento in memoria --> Descrittore teneva un puntatore a 18 bit alla tabella delle pagine.

Il descrittore conteneva anche la dimensione del segmento, i bit di protezione e altri oggetti.

In MULTICS ogni indirizzo era costituito da due parti:

- **Segmento**
- **Indirizzo nel segmento**

L'indirizzo nel segmento era ulteriormente suddiviso in un numero di pagina e un una parola nella pagina, quando avveniva un riferimento alla memoria veniva messo in atto il seguente algoritmo:

1. Il numero del segmento veniva usato per trovare il descrittore.
2. Se la tabella delle pagine era in memoria, allora veniva localizzata, se non ci stava allora avevamo un **segment fault**, se c'era una violazione allora si verifacava un trap.
3. Veniva esaminata la voce della pagina virtuale richiesta nella tabella delle pagine. Se la pagina non era in memoria, allora page fault, altrimenti dalla voce della tabella delle pagine veniva estratto l'indirizzo della pagina nella memoria principale.
4. Si otteneva l'indirizzo nella memoria principale in cui era localizzata la parola aggiungendo l'offset all'origine della pagina.
5. Lettura o salvataggio.

Questo processo avrebbe rallentato molto la velocità dei programmi, quindi l'hardware di MULTICS usava un TLB ad alta velocità a 16 parole.
Quando veniva presentato un indirizzo si controllava se prima stava nel **TLB** (Primo TLB di sempre).

Nel TLB venivano mantenuti gli indirizzi delle 16 pagine con i riferimenti più recenti.
I programmi con un set di lavoro minore della dimensione del TLB erano eseguiti molto velocemente, perché il TLB poteva contenerne tutte le pagine contemporaneamente.

Segmentazione con paginazione: Intel x86

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

Dal x86-64 in poi la segmentazione è diventata obsoleta e non più supportata, viene mantenuta solo per compatibilità.

I motivi del cambiamento sono due:

- SO come UNIX e Windows non la usavano per questioni di portabilità.
- Intel ha eliminato la segmentazione per ottimizare lo spazio del chip nelle CPU a 64 bit.

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