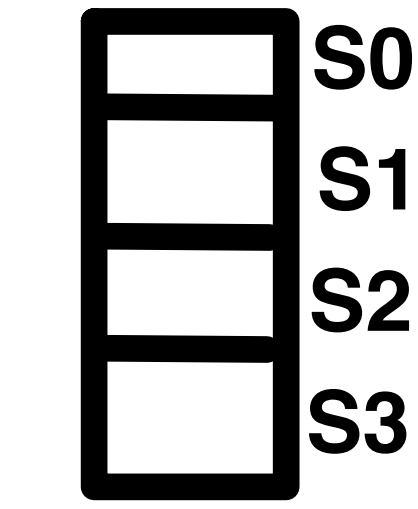
Segmentazione
sabato 13 maggio 2023 19:03

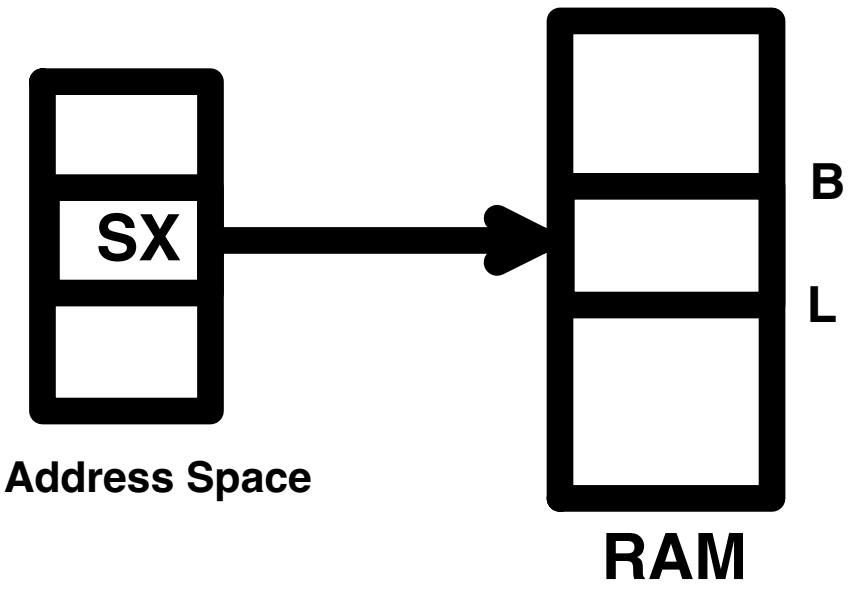
Segmentazione

È quella tecnica secondo cui, lo spazio di indirizzamento è visto come un insieme di segmenti distinti fra di loro (Esempio: .TEXT, .STACK, dati globali). Le taglie sono diverse.
Ciascuno di questi segmenti ha informazioni scorrelate da un altro segmento. All'interno di un singolo segmento ci sono informazioni omogenee.

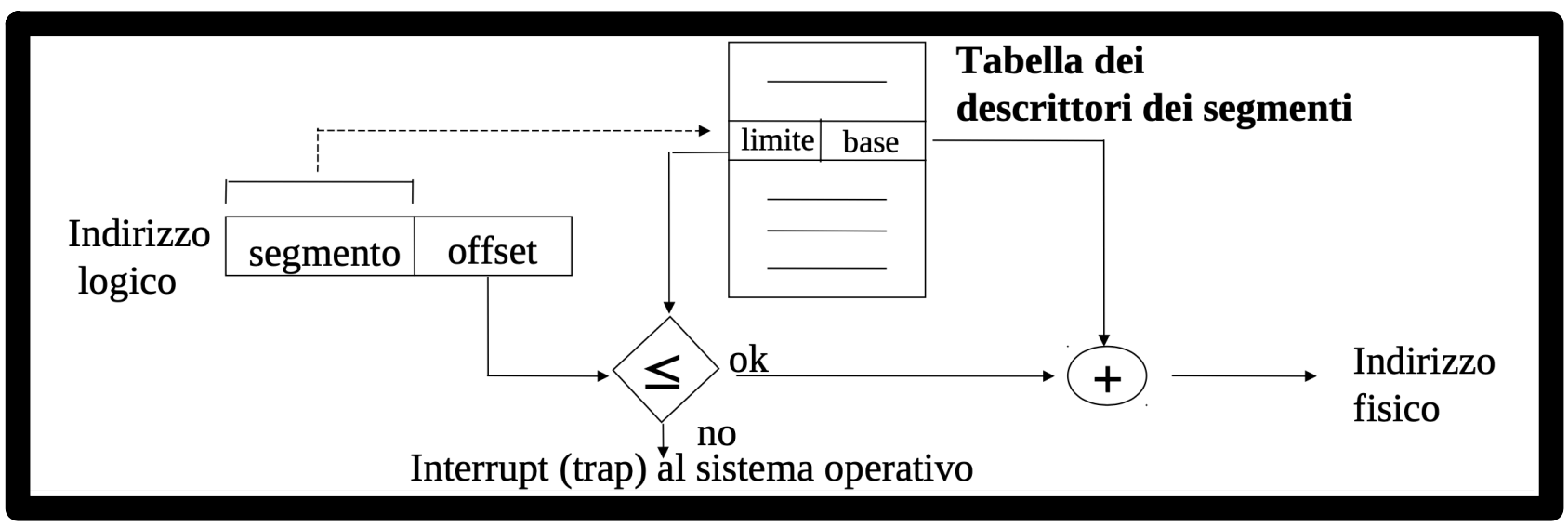


Per toccare un oggetto all'interno di questo contenitore, dobbiamo discriminare innanzitutto qual è il segmento SX a cui vogliamo accedere, e successivamente esprimere un offset valido all'interno di quel segmento.
In particolare un indirizzo logico in uno schema di segmentazione è fatto esattamente così.
Quando noi stiamo cercando di toccare questi segmenti secondo questa logica, per toccare realmente le informazioni che sono presenti all'interno di questi segmenti, in particolare per leggerle e portarle all'interno di un registro o per scriverle, dobbiamo avere un supporto per sapere dove ciascuno di questi segmenti è posizionato all'interno della memoria fisica.
Per far questo avevamo la tabella dei descrittori dei segmenti.
È un concetto quasi similare al concetto di page table, ma la segmentazione è ben diversa dalla paginazione, perché in uno schema di segmentazione, i segmenti possono avere taglie differenti.
Mentre invece in uno schema di paginazione vediamo l'address space come suddiviso in zone della stessa taglia, ossia pagine.

Nella segmentazione, quando esprimiamo un indirizzo logico che è costituito dall'identificatore del segmento logico a cui vogliamo accedere e dall'offset all'interno di quel segmento, dobbiamo andare all'interno della tabella dei descrittori ed identificare una entry associata a quell'identificatore di segmento, e quella entry ci dice dove il segmento è collocato in memoria fisica: in particolare qual è la base di quel segmento in memoria fisica e qual è il limite.
Supponendo di avere un address space e di avere un segmento SX, la tabella ci va ad indicare dove questo segmento è riportato nella memoria fisica.



Per accedere in memoria REALMENTE a questa informazione basta conoscere la BASE e applicare l'offset a cui volevamo accedere.
Una volta identificata la base, verificato che l'offset non ci porta fuori dalla RAM, questo offset viene sommato alla base per ottenere l'indirizzo fisico al quale accediamo.
Chiaramente se esprimiamo un offset che vorrebbe portarci fuori dal segmento e quindi fuori dal limite di memoria che possiamo utilizzare per questo segmento, questa cosa viene controllata dal sottosistema di gestione di questo processore in modo tale che venga generata una trap.

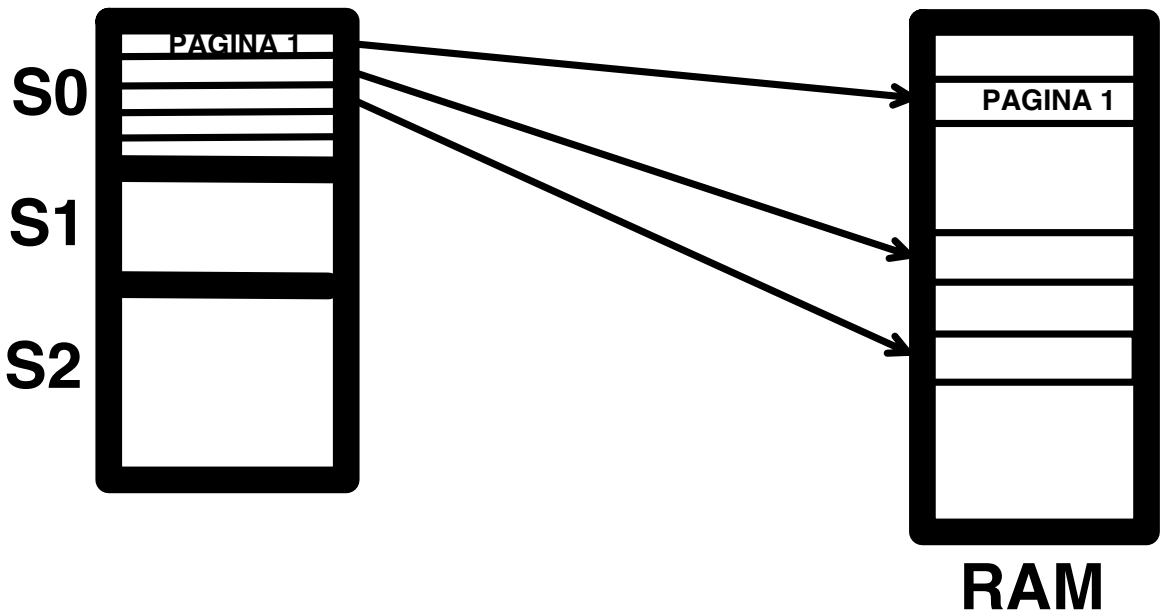


Questo è inaccettabile in qualsiasi schema di multiprogrammazione perché potremmo accedere ad informazioni che sono all'interno di un address space di altre applicazioni.
La segmentazione fu una proposta interessante non soltanto non soltanto per utilizzare la memoria fisica in maniera migliore rispetto al collocare un intero address space all'interno di una zona della memoria fisica contigua, ma anche perché in realtà noi potevamo rappresentare in maniera molto semplice informazioni di protezione dell'address space all'interno della tabella dei descrittori dei segmenti.
Se la entry di questa tabella era associata ad un segmento in cui noi avevamo inserito il testo della nostra applicazione, potevamo avere anche dei pochi bit di protezione per andare ad indicare che quel testo non era modificabile: quindi un unico bit per andare a proteggere tutto il testo dell'applicazione.
Se l'applicazione cerca di eseguire una scrittura nella zona .TEXT dell'AB, abbiamo una TRAP al S.O.
La segmentazione era importante anche per proteggere informazioni, perché queste informazioni all'interno dei segmenti (per esempio il testo) sono tutte omogenee tra di loro, quindi potevamo proteggerle in maniera omogenea utilizzando un'unica informazione di protezione indipendentemente dalla taglia di questo segmento.

La segmentazione non è stata utilizzata in maniera autonoma e basta, è stata ricombinata alla paginazione.

Segmentazione e paginazione

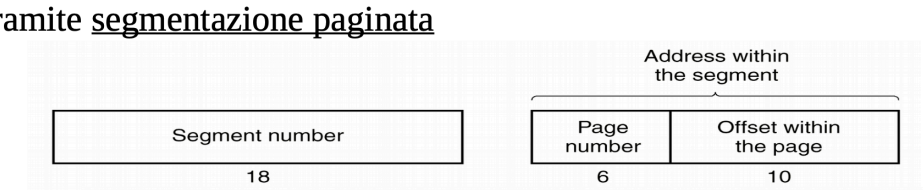
Nella segmentazione combinata alla paginazione, quello che noi abbiamo è esattamente quello che abbiamo detto prima, quindi quando noi accediamo al nostro address space esso è visto come una serie di segmenti S0 S1 S2 magari di taglie diverse, ma quando esprimiamo un accesso all'interno di uno di questi segmenti, in realtà i segmenti stessi sono visti come suddivisi in pagine.



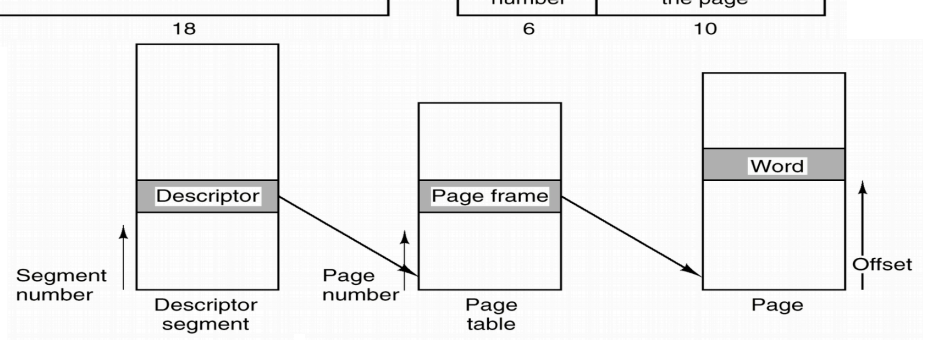
Il che implica dire che un segmento NON deve essere più caricato all'interno della memoria fisica in modo contiguo, ma poteva avere la zeresima pagina caricata in un dato frame1, la prima pagina caricata sotto in un altro frame2, e così via.
Questo avveniva in maniera SPARSA in memoria.
Paginare i vari segmenti implica dire che non abbiamo più il problema della frammentazione esterna.
Quando esprimiamo la volontà di accedere dobbiamo esprimere il codice del segmento SX a cui vogliamo accedere e l'OFFSET all'interno di quel segmento.
Per capire dove è collocata la pagina del nostro segmento all'interno della memoria fisica dobbiamo avere un'architettura di memory management complessa.
Questa architettura era composta in questo modo:
Nel momento in cui noi esprimevamo l'indirizzo tramite un codice di segmento a cui volevamo accedere, ovviamente avevamo anche l'offset all'interno di quel segmento, per capire quale fosse l'indirizzo fisico a cui accedere questa cosa era risolta passando attraverso una prima tabella di descrittori di segmento, in cui l'elemento associato al segmento a cui stiamo cercando di accedere ci va ad indicare dove è presente la tabella delle pagine per quel segmento, una volta che sappiamo qual è la tabella delle pagine utilizziamo il "Page number" per andare in questa tabella per conoscere qual è la posizione fisica della pagina a cui stiamo accedendo, per accedere realmente infine applichiamo l'offset che realmente abbiamo espresso.

- la segmentazione, a differenza della paginazione, può aiutare il programmatore nell'organizzazione del software (modularizzazione)
- la segmentazione presenta il problema della frammentazione esterna, che può essere ridotto tramite segmentazione paginata

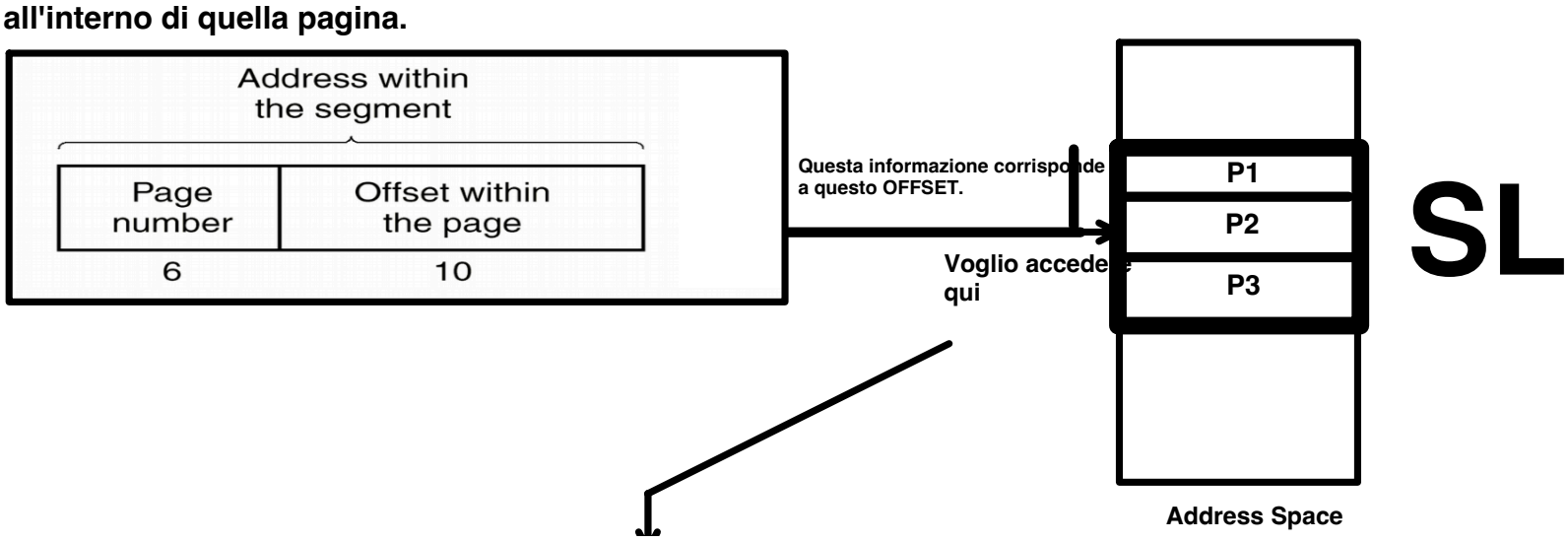
Segmento->pagina->spiazzamento



Es. MULTICS (1964 – MIT/Bell-Labs)



L'accesso ad un segmento era basato sull'idea di andare a discriminare qual è il segmento a cui vogliamo accedere SX, e indicare qual è il punto del segmento a cui siamo interessati ad accedere, quindi l'offset all'interno di questo segmento.
Questo offset ci viene rappresentato come il numero della pagina all'interno del segmento, e offset all'interno di quella pagina.



▼

**Saremo all'interno di uno specifico segmento,
all'interno di una certa pagina con un offset.
Dentro al segmento SL, nella pagina P2, con un offset
di pagina.**

Per questo tipo di indirizzamento e conoscere la posizione fisica, non c'è verso, bisogna passare all'interno della page table. Ne abbiamo una per segmento.

Il numero della pagina era fatto da 6 bit, quindi avevamo 2^6 pagine possibili, e 2^{10} indirizzi che potevamo esprimere all'interno di una singola pagina.