Gestione memoria: LA SEGMENTAZIONE

La segmentazione è una tecnica *ortogonale* alla paginazione per la gestione dello spazio di indirizzamento dei processi. A differenza della paginazione, *la segmentazione non è trasparente al programmatore*: infatti gli indirizzi di memoria (nelle istruzioni del linguaggio macchina) sono espressi sotto forma di coppie:

(numero segmento, offset all'interno del segmento)

quindi lo spazio di indirizzamento di un programma diventa bidimensionale (anzichè lineare).

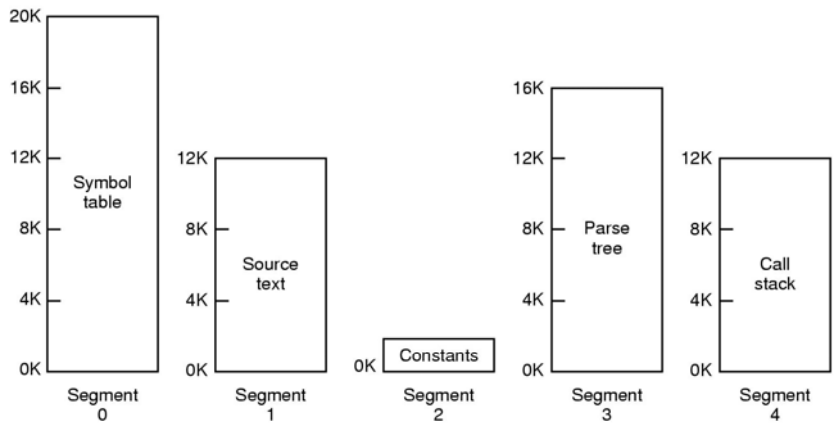
Ogni programma potrà avere uno o più segmenti di codice, uno o più segmenti dati. Per ogni segmento è possibile definire quali operazioni sono ammesse sulle informazioni in esso contenute: read, write, read&write, read&execute.

La gestione della memoria in un sistema che permetta la segmentazione dovrà occuparsi di allocare i segmenti (tutti o alcuni) di ciascun processo in memoria. Esisterà pertanto una tabella dei segmenti per ciascun processo che conterrà le informazioni sull'allocazione di ciascun segmento in memoria (oltre a informazioni sulla protezione di ciascun segmento).

1

Torniamo a sudolvisione logica dello spazio di indirizzamento, con meccanismi di protezione legati al segmento specifico.

Segmentazione



Ciascun segmento può crescere o ridursi indipendentemente

2

Tabella dei segmenti: lì ho anche info di protezione per es.
Parte più significativa è NUMERO DI SEGMENTO, che dovrà essere tradotto in un indirizzo base. Capisco se segmento è caricato in memoria, se sì a che indirizzo base.
Riproposo problema di frammentazione e allocazione in indirizzi consecutivi.

PAGINAZIONE E SEGMENTAZIONE

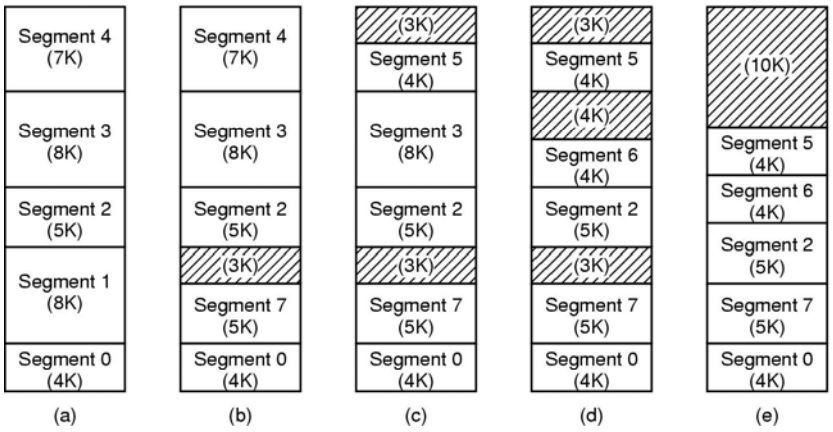
Consideration	Paging	Segmentation
Need the programmer be aware that this technique is being used?	No	Yes
How many linear address spaces are there?	1	Many
Can the total address space exceed the size of physical memory?	Yes	Yes
Can procedures and data be distinguished and separately protected?	No	Yes
Can tables whose size fluctuates be accommodated easily?	No	Yes
Is sharing of procedures between users facilitated?	No	Yes
Why was this technique invented?	To get a large linear address space without having to buy more physical memory	To allow programs and data to be broken up into logically independent address spaces and to aid sharing and protection

Vantaggio di divisione logica: prima tutto era nello stesso blocco: se lascio 3 pagine fisiche libere, non posso usarne di più. Se separo stack in un segmento a parte, la crescita dello stack è indipendente dagli altri segmenti. Non stanno uno sopra l'altro.

3

1. Tanti spazi di indirizzamento lineari separati tra loro.
2. Pagine prima erano raggruppate solo pagina per pagina. Qui faccio distinzione tra segmenti
3. Segmenti in cui posso mettere strutture dati, può darmi un ampio margine di flessibilità. Nelle pagine stanno in mezzo allo spaz. di indirizzamento.
4. Facile condivisione di info tra processi diversi. Qui ho segmento visibile da processi diversi. Non creo eseguibile con quelle funzioni più volte.

Implementazione della segmentazione pura



(a)-(d) Fenomeno della frammentazione esterna

(e) Compattazione

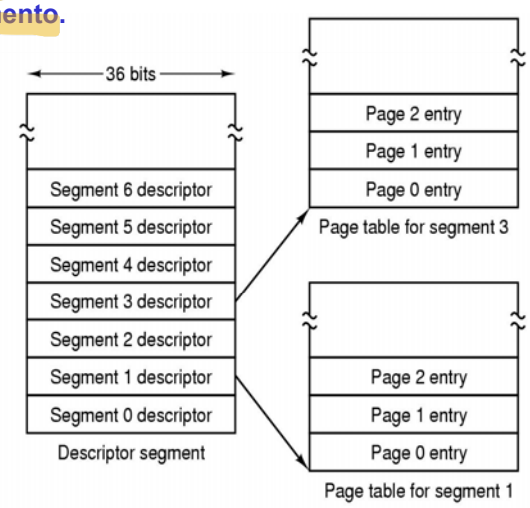
4

SVANTAGGIO: Frammentazione. Alloco segmenti di dimensione diversa. Tabella dei segmenti, con bit protezione, validità, indirizzo base. Ma tabella piccola rispetto con tante righe quanto i segmenti.

SOLUZIONE IBRIDA PER FRAMMENTAZIONE:

LA SEGMENTAZIONE PAGINATA

È possibile combinare segmentazione e paginazione: la segmentazione paginata consiste nel suddividere ciascun segmento in pagine di dimensione fissa, e gestire una tabella delle pagine per ciascun segmento.



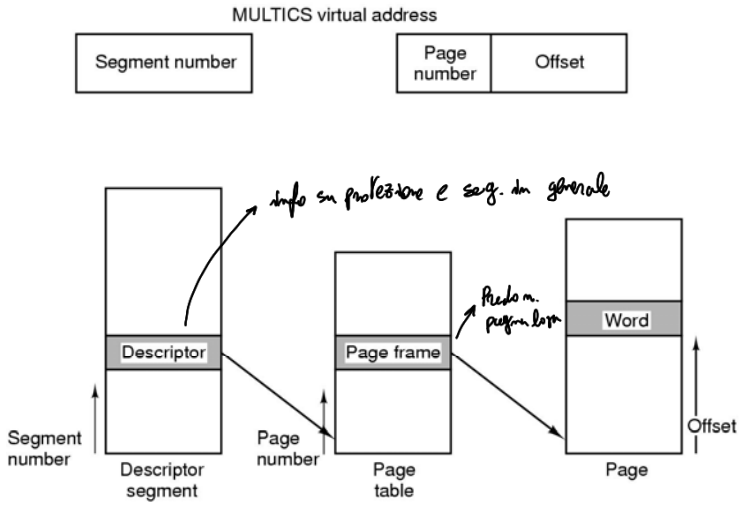
5

Allocazione deve essere di pagine di segmenti diversi. Questo mi complica la traduzione. Avevamo indirizzo logico n. pag. logica + offset, poi n. segmento + offset, qui mi servo 3 parti: n. segmento (primi bit di ind.) + bit di n. di pagina in quel seg. + offset in pagina.

Indirizzo logico è una terna. ⇒ Parto da bit più significativi. Prendo numero e accedo a tabella dei segmenti, che ha info sul seg. nella sua interezza (bit di protezione ecc.) + puntatore a tabella delle pagine del segmento.

La tabella delle pagine mi dice info su bit di validità, numero di frame fisico, bit R, M, ecc.

Segmentazione con paginazione: MULTICS

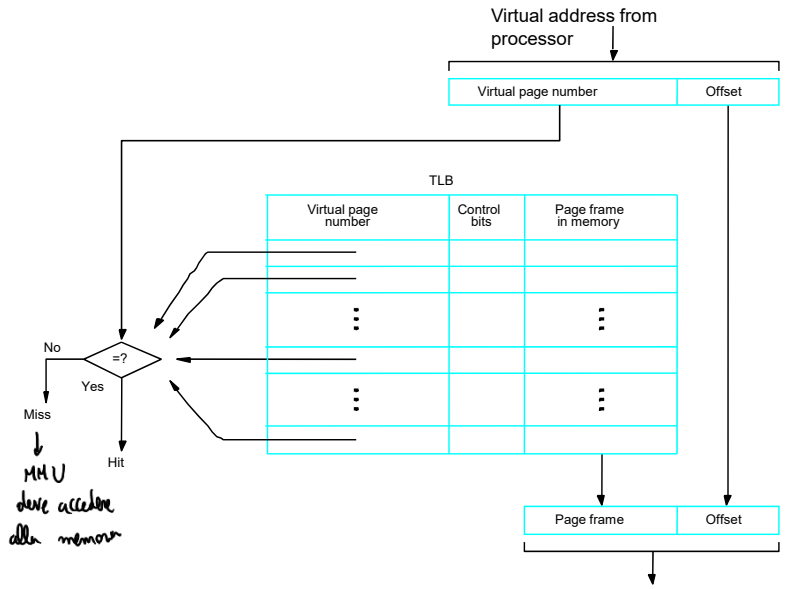


Conversione di un indirizzo virtuale MULTICS in un indirizzo fisico

6

1. Vantaggi della pag: no frammentazione ☺
2. Svantaggi: a diff. della paginazione, mi servono 2 accessi per la traduzione.
3. Vantaggi della seg: mantenimento di porzioni logiche

TLB



7

Ci avvicina TLB. Cambia un po'. ⇒ TLB, buffer di mem. a cui accedo con chiave e ottengo valore associato, ma qui comparazione in parallelo bit a bit che dà confronto bit a bit. Se stringa è presente bit produce 1 e mi esce frame fisico. Differenza: ora la chiave pag. logica non è sufficiente a identificare la pagina nel segmento. Ora la chiave diventa coppia n. segmento + n. pagina logica. Risparmio di più! Unico costo è chiave = coppia seg. + pag. logica.

Versione semplificata del TLB di MULTICS

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

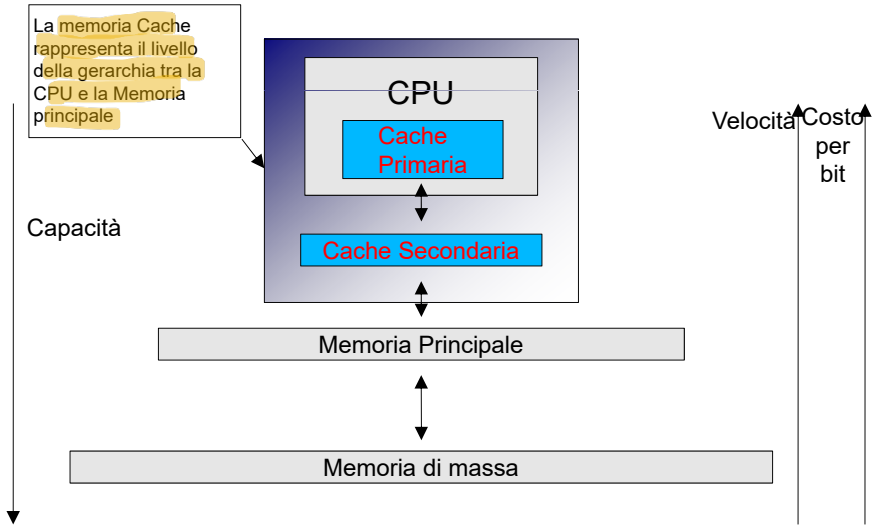
8

Dobbiamo però sempre ricordare la penalità di miss, dove qui il tempo necessario è più grande rispetto alla semplice paginazione. Dobbiamo aumentare il tasso di successo.

C'è un'interruzione non voluta tra indirizzo non tradotto e gestione nella cache?
Cache: pezzi di memoria piccoli con tempi di accesso bassi.

Gerarchia di Memorie

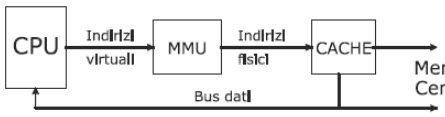
La memoria Cache rappresenta il livello della gerarchia tra la CPU e la Memoria principale



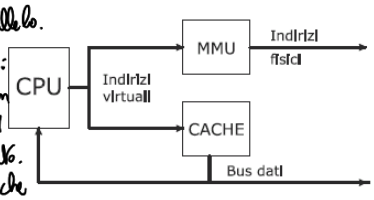
9

Accesso nella cache quando viene fatto? All'interno indirizzo fisico? Cache è copia di memoria principale.
Ma se processore cerca indirizzi virtuali?

1. Lascio cache come prima e la fa a valle della MMU post traduzione. Non cambia quasi niente della cache, ma la traduzione richiede tempo.



2ª sol: fare ricerca in cache con indirizzi virtuali o in parallelo.
Problema di questa soluzione: indirizzi virtuali possono non essere univoci. #proc. Logici 1 offset 2. Può essere duplicato.
Mi serve un reg aggiuntivo in cache per identificare processo (sorta di PID)



Problema dell'address aliasing

Figura 1.11 Posizionamento della cache. Nello schema in alto la cache è posta a valle della traduzione degli indirizzi e dunque la ricerca in cache avviene attraverso gli indirizzi fisici (in questo caso si parla di *cache fisica*); nello schema in basso la cache è posta a monte della traduzione e dunque la ricerca viene effettuata in base agli indirizzi virtuali (*cache virtuale*).

Stessa cosa per i segmenti.

Page fault gestito identicamente e potrei essere ancora più locale nella sostituzione.