**GESTIONE DELLA MEMORIA**

**MEMORIA PARTIZIONATA**

**Come fanno i sistemi operativi a gestire la memoria? (Memoria Partizionata-Paginazione-Segmentazione)**

Una dei metodi più semplici per l’allocazione della memoria consiste nel suddividere la stessa in partizioni di dimensione fissa. Ogni partizione deve contenere un processo quindi il grado di multiprogrammazione è limitato al numero di partizioni.

Nello schema a partizione fissa il sistema operativo conserva una tabella, in cui sono indicate le partizioni libere e quelle occupate.

Quando entrano nel sistema i processi vengono inseriti in una coda d’ingresso, il sistema operativo può ordinare quest’ultima secondo un algoritmo di scheduling.

Per determinare a quali processi si debba assegnare la memoria, il sistema operativo deve tener conto delle richieste di ogni processo e della quantità di memoria disponibile.

Quando ad un processo si assegna uno spazio in memoria, il processo stesso viene caricato in memoria e quindi può competere per il controllo della CPU, al termine rilascia la memoria assegnata e il sistema operativo può impiegare questo spazio di memoria liberato per un altro processo presente nella coda d’ingresso.

In ogni istante è sempre disponibile una lista delle dimensioni dei blocchi liberi e una lista della coda d’ingresso.

La memoria viene assegnata ad un processo della coda d’ingresso finché esiste un blocco di memoria, sufficientemente grande da accogliere quel processo.

Nel caso in cui nessun blocco disponibile è in grado d’accogliere il processo, il sistema operativo ha due possibilità:

1. Attende che sia disponibile un blocco sufficientemente grande d’accogliere quel processo.
2. Scorre la lista della coda d’ingresso per verificare se sia possibile soddisfare le richieste di memoria più limitate di qualche altro processo.

I criteri per scegliere uno dei buchi disponibili sono:

**First fit**: Si assegna il primo buco sufficientemente grande d’accogliere quel processo

**Best fit**: Si assegna il buco più piccolo che possa contenere quel processo

**Worst fit**: Si assegna il buco più grande che possa contenere quel processo

La gestione della memoria fisica mediante partizioni ha lo svantaggio di produrre la frammentazione della memoria.

**FRAMMENTZIONE**

**Che cos’è la frammentazione e come si può evitare?**

La frammentazione è un fenomeno che si verifica quando si caricano e si rimuovono dei processi dalla memoria.

Esistono due tipologie di frammentazione, frammentazione esterna e interna.

**Frammentazione esterna** si verifica quando lo spazio di memoria totale è sufficiente per soddisfare una richiesta, ma non è contiguo, la memoria è frammentata in tanti piccoli buchi.

**Frammentazione interna** si verifica quando abbiamo dello spazio di memoria inutilizzato all’interno di una partizione, questo può verificarsi perché i blocchi di memoria sono di dimensione fissa, quindi la memoria assegnata può essere leggermente maggiore della memoria richiesta.

Una soluzione al problema della frammentazione esterna è data dalla compattazione.

Lo scopo è quello di riordinare il contenuto della memoria per riunire la memoria libera in un unico grosso blocco.

Il più semplice algoritmo di compattazione consiste nel spostare tutti i processi verso un’estremità della memoria, mentre tutti i buchi vengono spostati nell’altra direzione formando un grosso buco di memoria, però questo metodo può essere assai oneroso.

Un’altra possibile soluzione è data dal consentire la non contiguità dello spazio degli indirizzi logici di un processo, permettendo così di assegnare la memoria fisica ai processi dovunque essa sia disponibile.

Due tecniche consentono questo la paginazione e la segmentazione.

**MEMORIA PAGINATA**

**Perché nasce la memoria a pagine? Che cos’è una pagina? Differenza tra frame e pagina?**

**Come avviene la traduzione degli indirizzi?**

La memoria a pagine è un metodo di gestione della memoria che permette che lo spazio degli indirizzi non sia contiguo.

Elimina il gravoso problema della sistemazione dei blocchi di memoria di diverse dimensioni in memoria ausiliaria.

La paginazione consiste nel suddividere la memoria fisica in blocchi di dimensione costante detti frame e nel suddividere la memoria logica in blocchi di pari dimensioni dette pagine.

Quando si deve eseguire un processo si caricano le sue pagine nei frame disponibili, prendendole dalla memoria ausiliaria (divisa in blocchi di dimensione fissa uguale a quella dei frame).

Ogni indirizzo generato dalla CPU è diviso in due parti: numero di pagina e uno scostamento di pagina.

Il numero di pagina serve come indice per la tabella delle pagine, la tabella delle pagine contiene l’indirizzo base in memoria fisica di ogni pagina. Questo indirizzo di base si combina con lo scostamento di pagina per definire l’indirizzo della memoria fisica, che s’invia all’unità di memoria.

**TLB**

**Che cos’è il TLB? Quanto è grande il TLB?**

La TLB è una memoria associativa ad alta velocità in cui ogni suo elemento consiste di due parti chiave e valore.

Quando si presenta un elemento la memoria associativa lo confronta contemporaneamente con tutte le chiavi e se trova una corrispondenza ne riporta il valore.

Una TLB può contenere dai 64 a 1024 elementi.

La TLB si usa insieme alla tabella delle pagine, la TLB contiene una piccola parte degli elementi della tabella delle pagine; quando la CPU genera un indirizzo logico, si presenta il numero di pagine al TLB, se tale numero è presente il corrispondente numero del frame è immediatamente disponibile e si usa per accedere alla memoria.

Se nella TLB non è presente il numero di pagina abbiamo un TLB miss e si deve consultare la tabella delle pagine. Il numero di pagina e del frame vengono caricati nella TLB in modo che al riferimento successivo non si causi nuovamente una TLB miss e la ricerca sarà molto più rapida.

Se la TLB è già piena di elementi il sistema operativo deve scegliere un elemento da sostituire.

I criteri di sostituzione variamo dalla scelta dell’elemento usato meno recentemente ad una scelta causale.

Inoltre alcune TLB consentono che certi elementi siano vincolati cioè non si possono rimuovere dalla TLB.

In genere si vincolano gli elementi per il codice del kernel.

**TABELLA DELLE PAGINE GERARCHICA**

La tabella delle pagine gerarchica è una semplice soluzione ad un problema che si presenta nei moderni calcolatori siccome abbiamo un spazio di indirizzi logici molto grande la tabella delle pagine finirebbe per diventare eccessivamente grande, quindi si adotta questa tecnica della tabella delle pagine gerarchiche che consiste nel suddividere la tabella delle pagine in parti più piccole.

Ho delle tabelle pagine enormi perché tendo ad utilizzare la parte alta e la parte bassa dello spazio degli indirizzi del processo il che significa che devo considerare sempre la tabella delle pagine intera.

Un metodo consiste nell’adottare un algoritmo di paginazione a due livelli, in cui la tabella stessa è paginata.

In questo algoritmo anziché suddividere l’indirizzo generato dalla CPU in numero di pagine ed offset ho una prima parte che mi individua un sotto indice della tabella delle pagine, dopodiché lì dentro entro con la seconda parte e prendo la pagina che mi serve.

Quindi la traduzione degli indirizzi si svolge dalla tabella esterna delle pagine verso l’interno, questo metodo è anche noto come tabella delle pagine ad associazione diretta.

Con questo sistema non si risparmia spazio di memoria.

Il vantaggio è che ho tante tabelle piccole che posso gestire meglio in memoria, quindi si ha una suddivisione della tabella delle pagine in tanti frammenti che sono più gestibili, alcuni dei quali possono rimanere permanentemente sul disco perché non mi servono.

Svantaggio Questa tecnica prevede l’utilizzo del TLB, (il TLB non è organizzato a due livelli ma c’è il numero di pagina e di frame) quindi se nel TLB trovo quello che mi serve non ho bisogno della tabella delle pagine ma se non trovo quello che mi serve nel TLB, in questo caso ho una grande perdita di tempo poiché per fare un accesso in memoria, devo fare un primo accesso a una tabella, un secondo accesso ad un'altra tabella e poi finalmente vado in memoria.

Questa tecnica è adotta oggi perché abbiamo degli HIT RATION del 99%.

Lo schema di paginazione a due livelli non è adatto a sistemi a 64 bit perché avremo una tabella delle pagine esterna troppo grande.

**TABELLA DELLE PAGINE DI TIPO HASH**

La tabella delle pagine di tipo hash viene utilizzata spesso per la gestione dello spazio degli indirizzi in architetture oltre i 32.

L’argomento della funzione hash è il numero della pagina virtuale.

Per la gestione delle collisioni, ogni elemento della tabella delle pagine contiene una lista concatenata di elementi che la funzione hash fa corrispondere alla stessa locazione.

Ciascun elemento è composto da tre campi:

1. Numero di pagina virtuale
2. Indirizzo del frame
3. Puntatore al successivo elemento della lista

Si applica la funzione hash al numero di pagina contenuto nell’indirizzo virtuale, identificando un elemento della tabella.

Si confronta il numero di pagina virtuale con il numero di pagina virtuale del primo elemento della lista concatenata.

Se i valori coincidono, si usa l’indirizzo del relativo frame per generare l’indirizzo fisico desiderato.

Altrimenti si esaminano gli elementi successivi della lista concatenata.

Le tabelle delle pagine sono particolarmente utili per gli spazi degli indirizzi sparsi, in cui i riferimenti alla memoria non sono contigui ma distribuiti per tutto lo spazio d’indirizzi.

Esiste una variante della tabella delle pagine invertite adatta agli spazi di indirizzamento a 64 bit.

Si tratta della tabella delle pagine a gruppi, simile alla tabella hash; ciascun elemento della tabella delle pagine contiene però i riferimenti alle pagine fisiche corrispondenti a un gruppo di pagine virtuali contigue.

**TABELLA DELLE PAGINE INVERTITE**

**Tabella delle pagine invertite come funziona? E quella normale?**

L’idea alla base della tabella delle pagine invertita è che visto che i frame in memoria non sono tantissimi, posso fare una cosa al rovescio “invertita” ovvero ricordarmi per ogni frame che pagina ci sta dentro.

La tabella delle pagine invertite memorizza per ogni frame qual è la pagina che ci sta dentro, siccome ogni processo ha la pagina 0 la pagina 1 la pagina 2 e così via, oltra a memorizzare il numero di pagina di se deve memorizzare anche il numero del processo.

Lo svantaggio è che non devo convertire frame in pagine ma pagine in frame quindi se questa organizzazione al rovescio mi fa risparmiare spazio mi penalizza quando vado a fare la conversione perché in pratica avrò il numero di pagina e in qualche maniera anche il PID (identificativo del processo) a questo punto dovremo cercare nella tabella delle pagine organizzata al rovescio la coppia PID-pagina e vedere il quale posizione si trova.

La ricerca non avviene sequenzialmente ma tramite tecniche di hashing (convertono la coppia pid pagina in una posizione all’interno della tabella)

Con la tecnica della paginazione invertita non c’è una tabella delle pagine per ogni processo ma c’è una tabella per sistema.

La tabella delle pagine normale memorizza per ogni pagina qual è il frame che ci sta dentro. Ed ogni processo ha una tabella delle pagine.

**SEGMENTAZIONE**

La segmentazione è uno schema di gestione della memoria ed è un meccanismo nel quale lo spazio indirizzi di un processo non mi viene suddiviso in fette tutte quante uguali ma viene suddiviso in pezzi che hanno qualcosa in comune, tipo tutto codice, tutto stack, dopodiché questi pezzi vengono allocati in memoria.

Lo svantaggio della segmentazione è che i segmenti non sono più della stessa grandezza per cui ho il problema della gestione dinamica della memoria.

La segmentazione può essere implementata in maniera simile alla paginazione però è un po’ più complicata perché ogni indirizzo deve essere fin dall’inizio suddiviso in due parti, cioè gli indirizzi logici prodotti dalla CPU devono essere sin dall’inizio suddivisi in due parti numero di segmento ed offset all’interno del segmento; dopodiché con il numero di segmento entro nella tabella dei segmenti (Ogni elemento della tabella dei segmenti è una coppia ordinata: la base di segmento e il limite di segmento) controllo se l’offset è compreso tra 0 e il limite di segmento se tale condizione è sommo l’offset alla base del segmento e ho tradotto l’indirizzo, altrimenti s’invia un segnale di eccezione al sistema operativo.

**Aggiunte Johnny**

La dimensione di una pagina, così come quella di un frame è definita dall’architettura del calcolatore, ed è in genere una potenza di 2 compresa tra 512 byte e 16MB.

Dimensione attuale delle pagine 4KB o 8KB