



Politecnico di Bari
Facoltà di Ingegneria informatica

Corso di Sistemi Operativi
Prof. G.Piscitelli
Anno Accademico 2001-2002

**Realizzazione di un simulatore per la
gestione della memoria tramite
partizionamento dinamico**

Arcieri Francesco 512155Y
Summo Valerio 511920Y

1 Descrizione della Tecnica di gestione della memoria tramite partizionamento

La tecnica di gestione della memoria tramite partizionamento dinamico si colloca all'interno delle tecniche di multiprogrammazione, ovvero in una serie di tecniche che permettono di caricare nella memoria RAM di un calcolatore più programmi contemporaneamente.

Nella tecnica di partizionamento dinamico l'ampiezza delle partizioni di memoria in cui verranno caricati i programmi viene definita dinamicamente. In questo modo si ha un maggior grado di libertà nel caricamento dei programmi rispetto alla tecnica del partizionamento statico in cui le dimensioni e il numero delle partizioni vengono fissate in modo definitivo al momento del bootstrap del calcolatore.

La tecnica del partizionamento dinamico viene realizzata dal sistema operativo tramite due tabelle: la *tabella delle partizioni* e la *tabella delle aree libere*.

Tabella delle partizioni

# partizione	Dimensione	Locazione	Stato

Nella tabella delle partizioni il primo campo rappresenta l'identificativo della partizione, il secondo la dimensione, il terzo la locazione della memoria a partire dalla quale verrà caricato il programma e il quarto infine lo stato della partizione (Allocata(1) / Non usata(0)).

NOTA: Il numero massimo di righe, con bit di Stato pari a 1, che la tabella delle partizioni può contenere rappresenta il massimo livello di multiprogrammazione, ovvero il numero massimo di programmi che si possono caricare in memoria. Questo numero corrisponde al numero massimo di processi che si possono trovare nel ciclo di esecuzione (Ready/Run/Blocked), di cui però uno solo si troverà nello stato di Run.

Tabella delle aree libere

# area	Dimensione	Locazione	Stato

Nella tabella delle aree libere il primo campo rappresenta il numero dell' area libera, il secondo la dimensione, il terzo la locazione della memoria a partire dalla quale è presente l'area libera e il quarto infine lo stato dell'area (Disponibile (1) / Non usata (0)).

1.1 Allocazione

Vediamo quali sono i passi per l'allocazione di un nuovo programma in memoria:

1. Si verifica che il numero di partizioni nella tabella delle partizioni con bit di Stato pari a 1 sia minore del massimo livello di multiprogrammazione. Se la condizione è verificata si può procedere con i passi successivi, altrimenti bisognerà segnalare all'utente l'impossibilità di allocare il nuovo programma.
2. Si verifica che ci sia una partizione libera abbastanza grande per ospitare il nuovo programma. Nella tabella delle aree libere si analizzano le dimensioni delle aree libere con bit di stato = 1 (Disponibile). Se tale area è presente si può creare una nuova partizione. Se è impossibile caricare il programma in memoria per mancanza di spazio bisognerà allora segnalare all'utente l'impossibilità di allocare il nuovo programma.
3. Si aggiornano le due tabelle modificando lo stato della partizione o creando una nuova riga nella tabella delle partizioni e modificando i valori di locazione e dimensione nella tabella delle aree libere.

1.2 Deallocazione

Vediamo quali sono i passi per la deallocazione di un programma dalla memoria:

1. Si cambia il bit di stato della partizione che contiene il programma da deallocare. Si pone cioè stato=0 (Non usata) nella riga corrispondente all'interno della tabella delle partizioni.
2. Si modifica la tabella delle aree libere inserendo una nuova riga con locazione e dimensione pari alla partizione liberata.
3. Se l'area liberata è contigua ad un'altra area libera si fa il merge delle due aree.

1.3 Gestione della tabella delle aree libere.

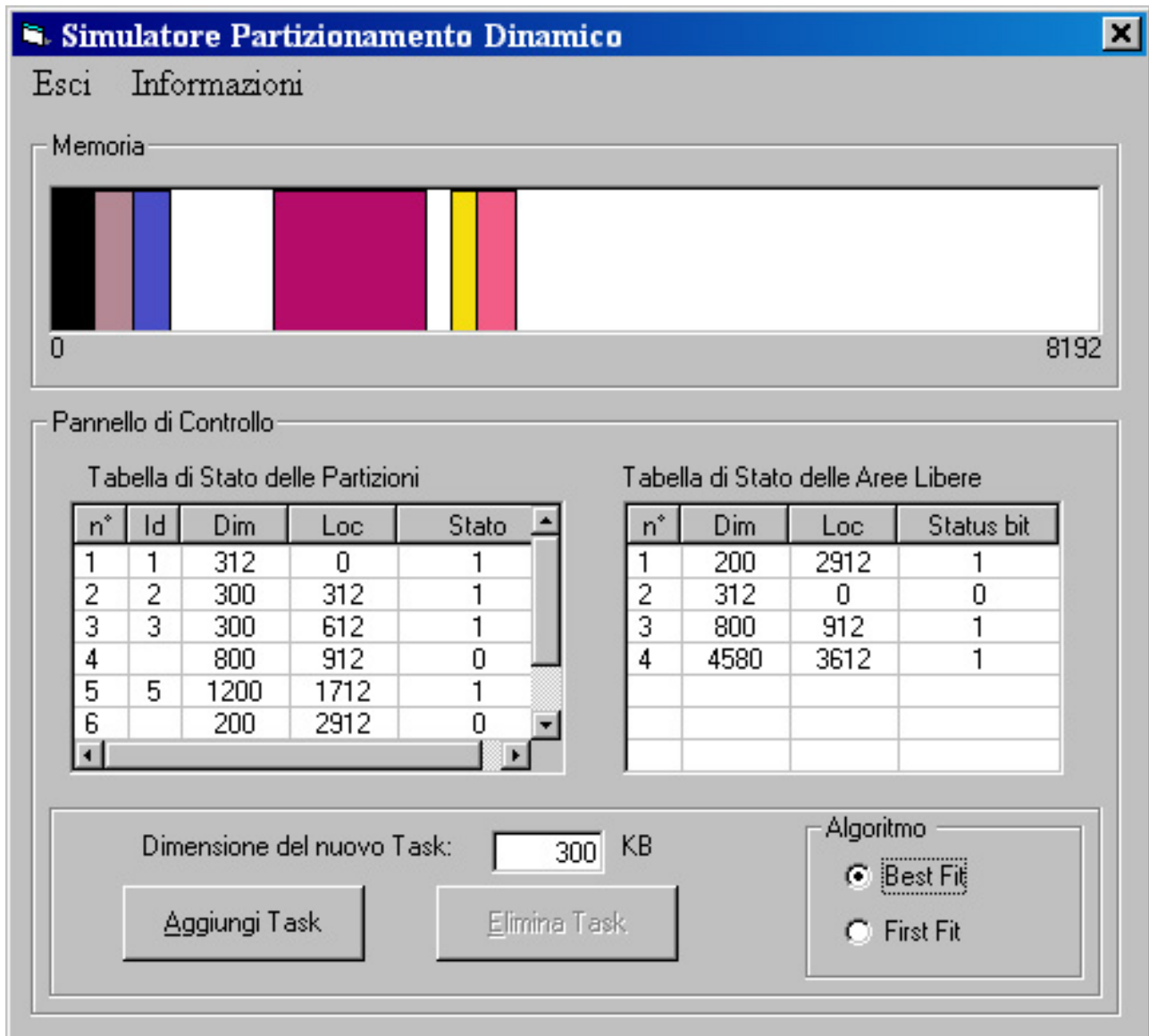
La tabella delle aree libere viene ordinata secondo uno dei due possibili algoritmi:

- **Best Fit:**
Questo algoritmo ordina le righe della tabella delle aree libere in base al contenuto del campo "Dimensione" in modo crescente.
In questo modo si assegna al programma da caricare la più piccola area disponibile di memoria abbastanza grande da contenerlo.
L'effetto negativo di questo algoritmo risiede nella creazione di piccoli frammenti di memoria difficilmente riutilizzabili.
- **First Fit:**
Questo algoritmo ordina le righe della tabella delle aree libere in base al contenuto del campo "Locazione" in modo crescente.
In questo modo si assegna al programma da caricare la prima area di memoria disponibile.
Come effetto negativo questo algoritmo non preserva le aree di memoria più grandi.
Può infatti succedere di assegnare una grossa area di memoria ad un programma di piccole dimensioni.

2 Realizzazione di SPD

Per la realizzazione di SPD (Simulatore Partizionamento Dinamico) è stato utilizzato Visual Basic come ambiente di sviluppo.

Di seguito è mostrata un'immagine dell'interfaccia dell'applicazione:



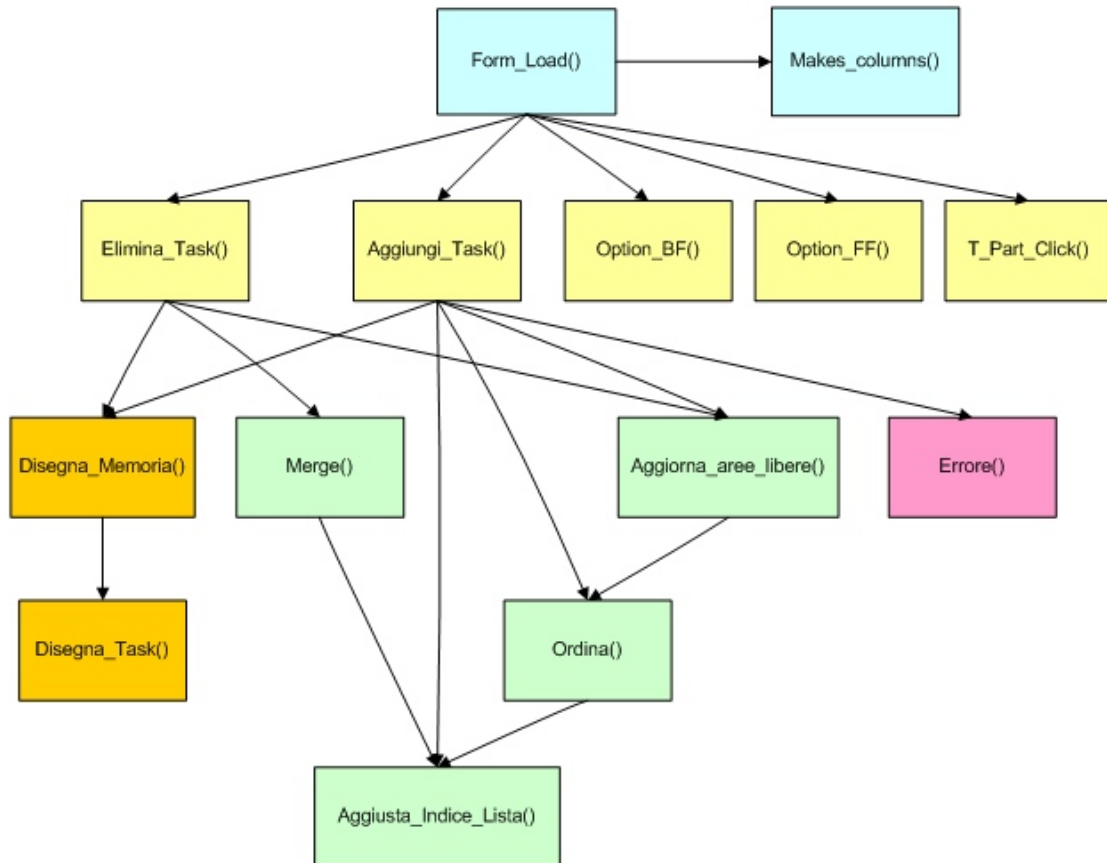
Come si può vedere è presente una rappresentazione grafica della memoria e dei task attivi.

All'interno del pannello di controllo abbiamo la possibilità di aggiungere un task specificandone la dimensione in Kb, eliminare il task selezionato e scegliere l'algoritmo di ordinamento della tabella delle aree libere (Best Fit / First Fit).

Vengono inoltre visualizzate la tabella delle partizioni e la tabella delle aree libere.

Infine è possibile uscire dal programma o visualizzare informazioni su di esso.

Di seguito è rappresentato lo schema delle relazioni presenti tra le varie procedure del programma.



- Le procedure *Form_Load()* e *Makes_columns()* sono procedure di inizializzazione.
- Le procedure *Elimina_Task()*, *Aggiungi_Task()*, *Option_BF()*, *Option_FF()* e *T_Part_click()* corrispondono ai controlli posizionati sul form e rappresentano quindi l'interfaccia del programma. *Elimina_Task()* e *Aggiungi_Task()* contengono anche gran parte della logica dell'applicazione.
- Le procedure *Disegna_Memoria()* e *Disegna_Task()* servono alla rappresentazione grafica dei task sull'interfaccia.
- La procedura *Errore()* gestisce le varie situazioni di errore che si possono verificare e i relativi messaggi di avviso all'utente.
- La procedura *Merge()* gestisce l'unione di aree libere adiacenti.
- La procedura *Aggiorna_aree_libere()* gestisce l'aggiornamento dei valori relativi alla tabella delle aree libere dopo l'allocazione o la deallocazione di un programma.
- La procedura *Ordina()* ordina la tabella delle aree libere in base all'algoritmo di ordinamento scelto (Best Fit / First Fit)
- La procedura *Aggiusta_Indice_Lista()* è una procedura di servizio per l'ordinamento degli indici all'interno delle tabelle.

3 Sommario

1	Descrizione della Tecnica di gestione della memoria tramite partizionamento	2
1.1	Allocazione	3
1.2	Deallocazione.....	3
1.3	Gestione della tabella delle aree libere.	3
2	Realizzazione di SPD.....	4
3	Sommario	6