1. RAM Virtuale e Rilocazione degli Indirizzi

La RAM virtuale è un meccanismo che crea l'illusione di uno spazio di memoria più grande di quello fisicamente disponibile. Questo permette di:

- Eseguire programmi più grandi della RAM fisica disponibile
- Permettere a più programmi di condividere la RAM
- Proteggere i programmi l'uno dall'altro

Problema degli Indirizzi

Quando un programma viene compilato, gli indirizzi assegnati alle variabili e alle istruzioni sono **relativi** (offset) rispetto a un punto di riferimento, solitamente l'indirizzo 0. Quando il programma viene caricato in memoria per l'esecuzione, questi indirizzi devono essere convertiti in indirizzi fisici reali dove il programma è stato effettivamente caricato.

Questo processo di conversione è chiamato **rilocazione degli indirizzi**.

2. Rilocazione Statica e Dinamica

Rilocazione Statica

Nella rilocazione statica, gli indirizzi sono convertiti dagli indirizzi logici agli indirizzi fisici al momento del caricamento del programma in memoria.

- **Funzionamento**: Un valore base (indirizzo di partenza in memoria fisica) viene aggiunto a tutti gli indirizzi nel programma quando questo viene caricato
- **Problema**: Una volta caricato, il programma deve rimanere in quella specifica area di memoria; non può essere spostato

```
Indirizzo fisico = indirizzo logico + base di rilocazione
```

Rilocazione Dinamica

Nella rilocazione dinamica, la conversione avviene durante l'esecuzione del programma, ogni volta che viene fatto riferimento a un indirizzo.

- **Funzionamento**: Viene utilizzato un registro hardware speciale, detto **registro base**, che contiene l'indirizzo di partenza del programma in memoria fisica
- **Vantaggio**: Il programma può essere spostato in memoria durante l'esecuzione, aggiornando semplicemente il valore del registro base

```
Indirizzo fisico = indirizzo logico + registro base
```

3. Gestione della Memoria

Partitioning

Partitioning Fisso

- La memoria è divisa in partizioni di dimensione fissa
- Ogni partizione può contenere un solo programma
- Problemi: frammentazione interna (spazio inutilizzato all'interno delle partizioni)

Partitioning Dinamico

- Le partizioni hanno dimensioni variabili, in base alle necessità dei programmi
- Problemi: frammentazione esterna (spazio inutilizzato tra le partizioni)

Strategie di Allocazione

- **First Fit**: assegna la prima partizione abbastanza grande
- **Best Fit**: assegna la partizione più piccola ma sufficiente
- **Worst Fit**: assegna la partizione più grande disponibile

Compattazione della Memoria

Processo che sposta i programmi in memoria per eliminare la frammentazione esterna.

4. Paginazione

La paginazione è una tecnica in cui la memoria fisica e lo spazio degli indirizzi logici sono divisi in blocchi di dimensione fissa.

- **Pagina**: unità di memoria logica (spazio di indirizzamento del processo)
- **Frame**: unità di memoria fisica (RAM reale)
- **Page Table**: tabella che mappa le pagine logiche ai frame fisici

Struttura di un Indirizzo Logico con Paginazione

Un indirizzo logico in un sistema paginato è costituito da due parti:

- **Numero di pagina (p)**: usato come indice nella page table
- **Offset nella pagina (d)**: spiazzamento all'interno della pagina

```
Indirizzo logico = <p, d>
```

Mapping degli Indirizzi

Il mapping da indirizzo logico a indirizzo fisico avviene come segue:

```
Indirizzo fisico = (f × dimensione pagina) + d
```

dove:

- f è il numero del frame ottenuto dalla page table per la pagina p
- d è l'offset nella pagina

Vantaggi della Paginazione

- Elimina la frammentazione esterna
- Supporta la memoria virtuale
- Facilita la condivisione di memoria tra processi

Page Table

La page table è una struttura dati che memorizza la mappatura tra pagine logiche e frame fisici.

Implementazioni della Page Table

1. **Page Table a Singolo Livello**
   - Semplice tabella con un'entrata per ogni pagina
   - Problema: può diventare molto grande
2. **Page Table Gerarchica**
   - L'indirizzo è diviso in più parti
   - Più livelli di tabelle
3. **Page Table Invertita**
   - Una singola tabella per tutto il sistema
   - Indicizzata per frame invece che per pagina

Translation Lookaside Buffer (TLB)

- Cache hardware che memorizza le mappature pagina-frame più recentemente usate
- Riduce drasticamente il tempo di accesso alla memoria

5. Memoria Virtuale

La memoria virtuale estende il concetto di paginazione permettendo l'esecuzione di programmi più grandi della memoria fisica disponibile.

Memoria Virtuale a Pagine

- Solo le pagine attivamente utilizzate vengono mantenute in memoria fisica
- Le pagine non utilizzate vengono memorizzate su disco

Page Fault

Quando si tenta di accedere a una pagina non presente in memoria:

1. Si verifica un page fault (eccezione)
2. Il sistema operativo interrompe l'esecuzione del processo
3. Trova un frame libero in memoria fisica
4. Carica la pagina richiesta dal disco
5. Aggiorna la page table
6. Riprende l'esecuzione dell'istruzione che ha causato il page fault

Algoritmi di Sostituzione delle Pagine

Quando la memoria è piena e si verifica un page fault, il sistema deve decidere quale pagina rimuovere:

- **LRU (Least Recently Used)**: rimuove la pagina non utilizzata da più tempo
- **FIFO (First-In, First-Out)**: rimuove la pagina caricata da più tempo
- **LFU (Least Frequently Used)**: rimuove la pagina utilizzata meno frequentemente
- **Algoritmo della Seconda Opportunità**: estensione di FIFO con un bit di riferimento

Memoria Virtuale a Segmenti

- Lo spazio degli indirizzi è diviso in segmenti di dimensione variabile (es. codice, dati, stack)
- Ogni segmento ha il proprio identificatore
- Si usa una tabella dei segmenti per la mappatura

Struttura dell'Indirizzo Logico con Segmentazione

Un indirizzo logico è composto da:

- **Numero di segmento (s)**: indice nella tabella dei segmenti
- **Offset nel segmento (d)**: spiazzamento all'interno del segmento

```
Indirizzo logico = <s, d>
```

Tabella dei Segmenti

Ogni entrata nella tabella dei segmenti contiene:

- Base: indirizzo fisico di inizio del segmento
- Limite: lunghezza del segmento
- Bit di protezione: permessi di accesso (lettura, scrittura, esecuzione)

Verifica degli Accessi

Prima di ogni accesso alla memoria, il sistema verifica che:

1. L'offset sia minore del limite del segmento
2. L'operazione rispetti i permessi di accesso

Segmentazione con Paginazione

I sistemi moderni spesso combinano segmentazione e paginazione:

- Ogni segmento è diviso in pagine
- Si mantiene una page table per ciascun segmento

Questo approccio combina i vantaggi di entrambe le tecniche:

- Supporto per spazi di indirizzi non contigui (segmenti)
- Efficienza nell'allocazione della memoria fisica (pagine)
- Riduzione della frammentazione esterna
- Possibilità di implementare facilmente la protezione