Gestione della memoria - requisiti di base

Index

Requisiti per la gestione della memoria

I requisiti per la gestione della memoria sono:

- **Rilocazione** → richiede che ci sia aiuto hardware (il sistema operativo viene aiutato da opportune funzioni macchina di basso livello)
- **Protezione** → richiede che ci sia aiuto hardware
- **Condivisione**
- **Organizzazione logica**

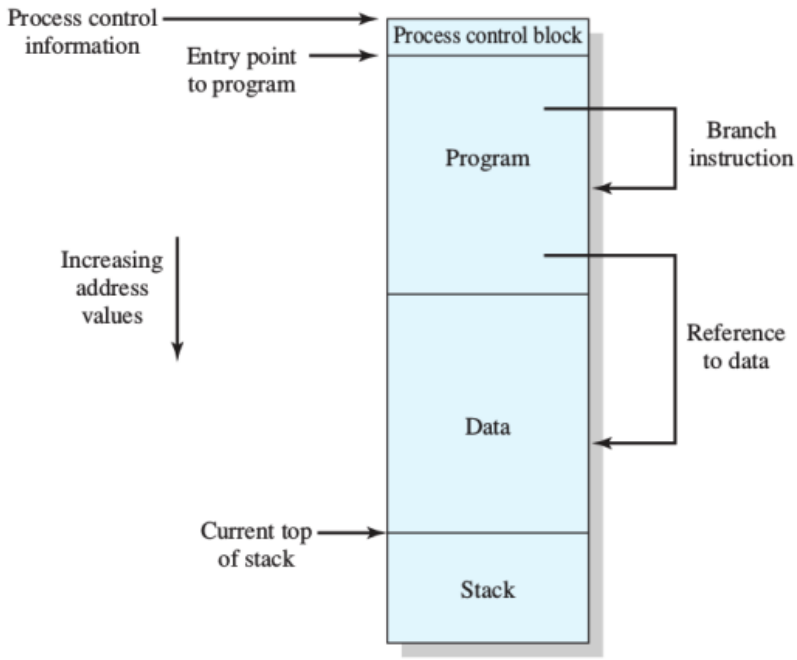
Rilocazione

Il programmatore (assembler o compilatore) non sa e non deve sapere in quale zona della memoria il programma verrà caricato. Questo atteggiamento è chiamato **rilocazione**, il programma deve essere in grado di **essere eseguito indipendentemente da dove si trovi in memoria**.
Può accadere infatti che:

- potrebbe essere swappato su disco, e al ritorno in memoria principale potrebbe essere in un'altra posizione

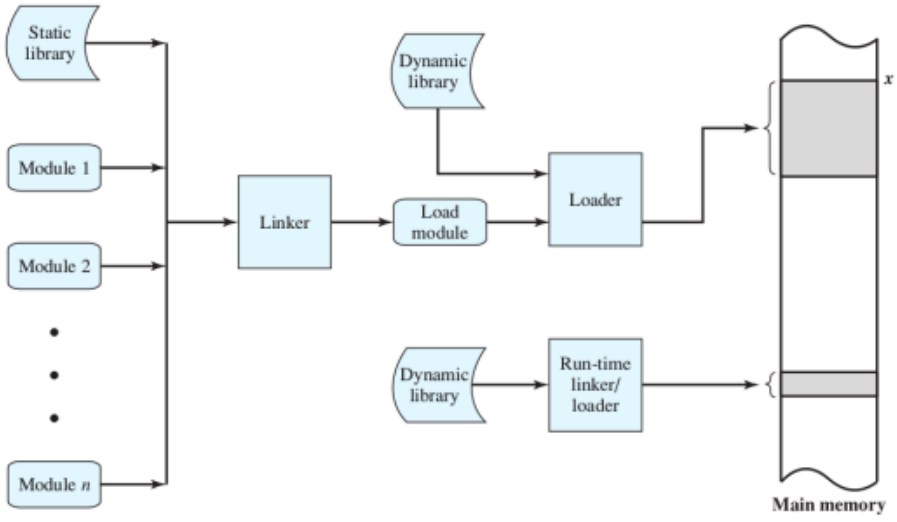
- potrebbe anche non essere contiguo, oppure con alcune pagine in RAM e altre su disco

I riferimenti alla memoria devono tradotti nell'indirizzo fisico "vero"; può essere fatto tramite preprocessing o **runtime** (ogni volta che viene eseguita un'istruzione, se quell'istruzione contiene un indirizzo occorre fare la sostituzione), in quest'ultimo caso è necessario avere un supporto hardware (a livello software ci sarebbe un overhead troppo grande)

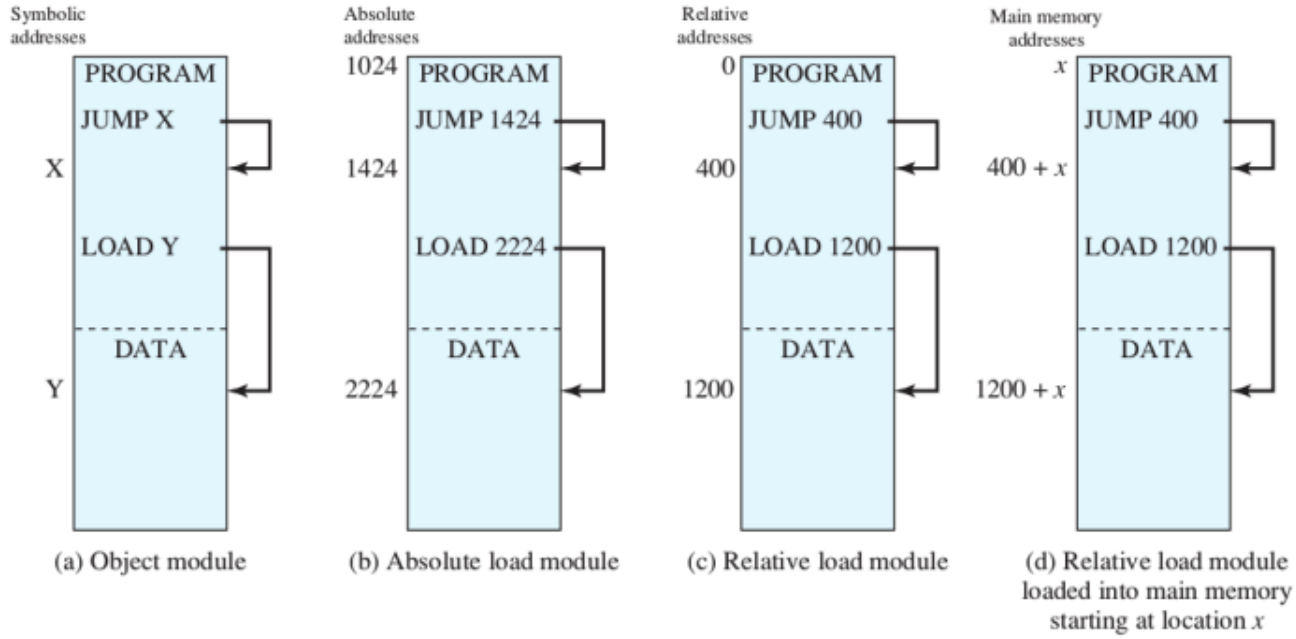


Indirizzi nei programmi

Il *linker* ha il compito di unire quelli che sono i moduli (ovvero tutti i file di un programma) e le librerie statiche (librerie esterne) e crea come output un *load module* che può essere preso e caricato in memoria RAM, passaggio che avviene tramite il *loader* che unisce eventuali librerie dinamiche



Si parte dal caso (a) in cui si hanno dei link simbolici alle parti del programma
Nel caso (b) si hanno degli indirizzi assoluti ma questo è possibile solo nel caso in cui si sa da che indirizzo si parte, deve quindi avvenire in preprocessing
Nel caso (c) si salta ad un indirizzo relativo rispetto all'inizio del programma
Nel caso (d) si salta ad un indirizzo relativo rispetto alla posizione del programma in memoria

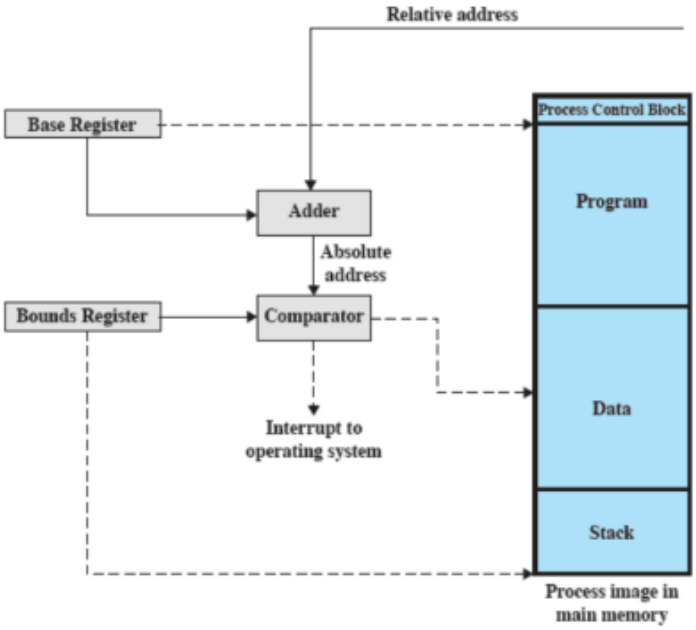


Abbiamo quindi tre tipi di indirizzi:

- **Logici** → il riferimento in memoria è indipendente dall'attuale posizionamento del programma in memoria
- **Relativi** → riferimento è espresso come uno spiazzamento (differenza) rispetto ad un qualche punto noto (caso particolare degli indirizzi logici)
- **Fisici o Assoluti** → riferimento effettivo alla memoria

Soluzioni possibili

Vecchissima soluzione: gli indirizzi assoluti vengono determinati nel momento in cui il programma viene caricato (nuovamente o per la prima volta) in memoria
Soluzione più recente: gli indirizzi assoluti vengono determinati nel momento in cui si fa un riferimento alla memoria (serve hardware dedicato).
E' necessario un hardware dedicato in quanto, se non ci fosse, ogni volta che un processo viene riportato in memoria, potrebbe essere in un posto diverso. Nel frattempo, potrebbero essere arrivati altri processi e averne preso il posto, quindi, ad ogni ricaricamento in RAM, occorre ispezionare tutto il codice sorgente del processo sostituendo man mano tutti i riferimenti agli indirizzi. Ciò risulterebbe in troppo overhead



Base register (registro base) → indirizzo di partenza del processo
Bounds register (registro limite) → indirizzo di fine del processo
I valori per questi registri vengono settati nel momento in cui il processo viene posizionato in memoria mantenuti nel PCB del processo fa parte del passo 6 per il process switch (vedere slides sui processi); non vanno semplicemente ripristinati: occorre proprio modificarli

Dunque il valore del registro base viene aggiunto al valore dell'indirizzo relativo per ottenere l'indirizzo assoluto e il risultato viene confrontato con il registro limite. Se va oltre, viene generato un interrupt per il SO (simile al segmentation fault)

Protezione

I processi non devono poter accedere a locazioni di memoria di un altro processo, a meno che non siano autorizzati. Anche questo, a causa della rilocazione, non può essere fatto a tempo di compilazione, deve quindi esser fatto a tempo di esecuzione. Per questo motivo è necessario aiuto hardware

---

Condivisione

Solitamente la maggior parte dell'immagine del processo deve essere protetta, ma ci possono essere dei casi (es. se può processi devono risolvere uno stesso problema) ci può essere una parte del processo accessibile sia in lettura sia in scrittura da un altro processo.

Dunque può avvenire sia perché il programmatore ne è cosciente, sia può essere fatto direttamente dal sistema operativo. Un caso tipico è quando più processi vengono creati eseguendo più volte lo stesso codice sorgente (fintantoché questi processi restano in esecuzione, è più efficiente che condividano il codice sorgente, visto che è lo stesso)

Organizzazione logica

A livello hardware la memoria è organizzata in modo lineare (sia RAM che disco), mentre a livello software i programmi sono tipicamente scritti in moduli, che possono essere compilati separatamente e che possono avere permessi diversi

Questo requisito fa si che il SO faccia il **bridging** (traduzione) tra quello che ad alto livello vedono i programmatori e ciò che effettivamente viene eseguito (spesso tramite segmentazione)

Gestione fisica

La gestione fisica serve a gestire il flusso di dati tra RAM (piccola, veloce e volatile) e memoria secondaria (grande, lenta permanente).

Fino a circa 40 anni fa la gestione della RAM veniva lasciata al programmatore, doveva infatti gestire esplicitamente lo swapping in base a quanta memoria gli veniva messa a disposizione dal SO. Risulta però un processo molto complesso, per questo è stato deciso di affidare questo compito esclusivamente al SO