Esercizio: Comprendere il Binding degli Indirizzi

Immagina di avere un programma chiamato `programma.c` con due variabili globali e una sola funzione `main` che utilizza queste variabili:

```
#include <stdio.h>

int a = 5;   // Variabile globale
int b = 10;  // Variabile globale

int main() {
    printf("Valore di a: %d\n", a);
    printf("Valore di b: %d\n", b);
    return 0;
}
```

In un programma, sia le variabili (`a` e `b`) che la funzione `main` hanno un indirizzo di memoria. Durante l'esecuzione di un programma, ogni variabile e funzione occupa una posizione specifica in memoria, a cui viene assegnato un indirizzo. Gli indirizzi di memoria per le variabili e per le funzioni possono essere assegnati in modi diversi, a seconda del tipo di *binding* utilizzato:

- Le **variabili globali** `a` e `b` sono allocate nel segmento di dati del programma, dove ogni variabile occupa una posizione specifica con un indirizzo proprio.
- La **funzione** **`main`** e le altre eventuali funzioni del programma sono allocate nel segmento di codice del programma. Anche la funzione `main`, quindi, ha un indirizzo di memoria, che serve al programma per sapere dove iniziare l'esecuzione delle istruzioni.

Quando `main` viene chiamata, il programma inizia a eseguire le istruzioni a partire dall'indirizzo assegnato a `main`. Se ci fossero altre funzioni nel programma, anche queste avrebbero un proprio indirizzo nel segmento di codice.

Domande

1. **Binding Statico a Tempo di Compilazione**
   - Supponiamo che il sistema operativo e il compilatore utilizzino un *binding* statico a tempo di compilazione. Tutti gli indirizzi di memoria per le variabili e per le funzioni vengono risolti durante la compilazione.
   - **Domanda:** Quali indirizzi fisici potrebbero essere assegnati a `a`, `b` e alla funzione `main`? Perché questi indirizzi restano invariati durante ogni esecuzione?
   - **Domanda: 2** Il segmento .text (contenente codice eseguibile) viene caricato in memoria al caricamento del programma. Quali indirizzi fisici verranno assegnati alle istruzioni del programma?
2. **Binding Dinamico a Tempo di Caricamento**
   - Supponiamo ora che il sistema utilizzi un *binding* dinamico a tempo di caricamento. Gli indirizzi relativi per le variabili e le funzioni vengono determinati durante la compilazione, ma il sistema operativo risolve gli indirizzi fisici solo quando il programma viene caricato in memoria.
   - **Domanda:** Se il programma viene caricato in una posizione di memoria diversa nelle esecuzioni successive (ad esempio, 2000 in una esecuzione e 3000 in un'altra), come cambieranno gli indirizzi di `a`, `b`, `main`, e `stampa_somma`?
   - **Domanda: 2** Il segmento .text (contenente codice eseguibile) viene caricato in memoria al caricamento del programma. Quali indirizzi fisici verranno assegnati alle istruzioni del programma?
3. **Binding Dinamico a Tempo di Esecuzione**
   - Infine, supponi che il sistema utilizzi un *binding* dinamico a tempo di esecuzione. Gli indirizzi delle variabili e delle funzioni vengono risolti quando le istruzioni vengono effettivamente eseguite, permettendo di spostare le variabili e le funzioni in memoria durante l'esecuzione.
   - **Domanda: 1** Se il programma viene caricato in una posizione di memoria diversa nelle esecuzioni successive (ad esempio, 2000 in una esecuzione e 3000 in un'altra), come cambieranno gli indirizzi di `a`, `b`, `main`, e

`stampa_somma`?

- **Domanda: 2** Il segmento .text (contenente codice eseguibile) viene caricato in memoria al caricamento del programma. Come vengono determinati gli indirizzi fisici delle istruzioni? Come interviene la MMU in questo processo durante l’esecuzione?