Esercizio: Analisi del Binding degli Indirizzi con Istruzioni Assembly

Un programma viene compilato e tradotto in codice assembly. Durante la compilazione e l'esecuzione del programma, gli indirizzi di memoria relativi alle variabili e alle funzioni devono essere determinati.

Quando il programma viene caricato in memoria, la **sezione** `.text` contiene il codice eseguibile, mentre le variabili globali come `a`, `b`, e la funzione `main` occupano altre posizioni nella memoria RAM. Gli indirizzi di questi elementi possono essere determinati in modo diverso a seconda del tipo di **binding** utilizzato dal sistema operativo e dal compilatore.

Immagina che la seguente istruzione in C

```
int a = 10;
```

venga tradotta nel seguente codice assembly semplificato:

```
mov rax, 10        ; Carica il valore 10 nel registro rax
mov [a], rax       ; Memorizza il valore di rax (10) nella variabile 'a'
```

dove l'istruzione assembly MOV permette di **spostare** (o copiare) dati da una sorgente a una destinazione. La sintassi di base è:

```
MOV destinazione, sorgente
```

dove **Destinazione** è il registro o la locazione di memoria dove i dati devono essere memorizzati, mentre **Sorgente**: è il valore o l'indirizzo che contiene i dati da trasferire.

Nel nostro esempio, `[a]` è un riferimento all'indirizzo di memoria RAM di `a`. L'indirizzo di `a` può variare a seconda del tipo di *binding* utilizzato dal sistema operativo o dal compilatore.
Come cambia l'indirizzo di `[a]` in base al tipo di **binding**? Ovvero, come cambia l'istruzione caricata in memoria RAM `mov [a], rax` nella sezione `.text` del programma in base al tipo di **binding**? Specifica quando si tratta di binding statico o dinamico.