# Risoluzione esercizio assembly x86

## Traccia:

Nella lezione teorica del mattino, abbiamo visto i fondamenti del linguaggio Assembly. Dato il codice in Assembly per la CPU x86 allegato qui di seguito, identificare lo scopo di ogni istruzione, inserendo una descrizione per ogni riga di codice. Ricordate che i numeri nel formato 0xYY sono numeri esadecimali. Per convertirli in numeri decimali utilizzate pure un convertitore online, oppure la calcolatrice del vostro computer (per programmatori).

- 0x00001141 <+8>: mov EAX,0x20
- 0x00001148 <+15>: mov EDX,0x38
- 0x00001155 <+28>: add EAX,EDX
- 0x00001157 <+30>: mov EBP, EAX
- 0x0000115a <+33>: cmp EBP,0xa
- 0x0000115e <+37>: jge 0x1176 <main+61>
- 0x0000116a <+49>: mov eax,0x0
- 0x0000116f <+54>: call 0x1030 <printf@plt>

Ogni istruzione svolge una funzione specifica all'interno di un programma più grande. Analizziamole una per una per capire il loro scopo:

1. **0x00001141 <+8>: mov EAX,0x20**
   - Questa istruzione carica il valore esadecimale 0x20 (32 in decimale) nel registro EAX. I registri come EAX sono usati per operazioni aritmetiche, di confronto e altre funzioni di manipolazione dei dati.
2. **0x00001148 <+15>: mov EDX,0x38**
   - Similmente alla prima istruzione, questa carica il valore esadecimale 0x38 (56 in decimale) nel registro EDX. EDX è un altro registro comunemente usato per memorizzare dati temporanei.
3. **0x00001155 <+28>: add EAX,EDX**
   - Questa istruzione somma il valore nel registro EDX al valore nel registro EAX. Dopo questa operazione, EAX conterrà la somma di 32 e 56, ovvero 88.
4. **0x00001157 <+30>: mov EBP, EAX**
   - Copia il valore attuale di EAX (che ora è 88) nel registro EBP. EBP è tipicamente usato come base pointer per accedere ai dati nella stack frame del programma.
5. **0x0000115a <+33>: cmp EBP,0xa**
   - Confronta il valore in EBP (88) con 0xa (10 in decimale). Questa istruzione prepara il processore per un'operazione condizionale successiva, impostando i flag interni in base al risultato del confronto.

6. **0x0000115e <+37>: jge 0x1176 <main+61>**
    - Questa istruzione di salto condizionale ("jump if greater or equal") fa sì che il programma salti all'indirizzo 0x1176 se il valore in EBP è maggiore o uguale a 10, che in questo caso lo è sicuramente. Questo è un esempio di controllo di flusso basato sui risultati di un confronto.
7. **0x0000116a <+49>: mov eax,0x0**
    - Imposta il registro EAX a 0. Questa istruzione potrebbe essere usata per preparare il valore di ritorno di una funzione, indicando, ad esempio, un'uscita senza errori.
8. **0x0000116f <+54>: call 0x1030 printf@plt**
    - Chiama la funzione printf, una funzione standard di C per la stampa di output, che si trova all'indirizzo 0x1030 (probabilmente una chiamata attraverso la Procedure Linkage Table, PLT, per gestire chiamate a funzioni condivise).