

CONSEGNA S10/L3

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>
```

SPIEGAZIONE

Questo frammento di codice assembly sembra essere parte di una **funzione main** o di qualche altra **sezione di codice**. Ecco una spiegazione riga per riga:

mov EAX, 0x20: Questa istruzione sposta il numero esadecimale 0x20 (in decimale: 32) nel registro EAX. EAX è uno dei registri general purpose usati comunemente per le operazioni aritmetiche o come puntatore ai dati.

mov EDX, 0x38: Sposta il numero esadecimale 0x38 (in decimale: 56) nel registro EDX. Anche EDX è un registro general purpose, spesso usato insieme ad EAX per operazioni che richiedono più di 32 bit, o per scopi specifici come la divisione e la moltiplicazione.

add EAX, EDX: Aggiunge il valore contenuto in EDX al valore contenuto in EAX, e il risultato dell'operazione è memorizzato in EAX. Dopo questa operazione, EAX conterrà 0x20 + 0x38 = 0x58 (in decimale: 88).

mov EBP, EAX: Sposta il valore attualmente contenuto in EAX nel registro EBP. EBP è comunemente usato come base pointer per accedere a parametri di funzione e variabili locali nello stack.

cmp EBP, 0xa: Compara il valore contenuto in EBP con il numero esadecimale 0xA (in decimale: 10). Questa istruzione imposta i flag nel registro FLAGS basati sul risultato del confronto, che può essere usato da istruzioni condizionali di salto come JGE (Jump if Greater or Equal).

jge 0x1176 <main+61>: Effettua un salto condizionale all'indirizzo 0x1176 se il valore in EBP è maggiore o uguale (\geq) a 0xA (10 in decimale). Questo significa che se il risultato dell'addizione precedente è ≥ 10 , il controllo del programma salta all'indirizzo specificato.

mov eax, 0x0: Sposta il numero 0 nel registro EAX. Questa istruzione è spesso usata per impostare il valore di ritorno di una funzione, con 0 tipicamente indicante "successo" in molte convenzioni.

call 0x1030 <printf@plt>: Chiama la funzione printf, che è una funzione standard della libreria C per la stampa di output. L'indirizzo 0x1030 è un indirizzo all'interno della Procedure Linkage Table (PLT), che gestisce le chiamate a funzioni condivise, come quelle della libreria C. Prima della chiamata, i parametri per printf dovrebbero essere preparati sui registri o nello stack, ma qui non vediamo questa preparazione, quindi è possibile che questa parte del codice sia incompleta o che i parametri siano stati preparati precedentemente.

In sintesi, questo codice carica alcuni valori nei registri, esegue un'addizione, confronta il risultato e, basandosi sul confronto, potrebbe saltare a un altro punto nel codice. Infine, prepara un valore (probabilmente per una funzione di ritorno) e chiama la funzione printf. La preparazione specifica dei parametri per printf non è mostrata in questo frammento.