Assembly x86

Richiesta: Dato il seguente codice in Assembly per la CPU x86, identifica lo scopo di ogni istruzione, inserendo una descrizione per ogni riga di codice.

```
0×00001141 <+8>:    mov    EAX,0×20
0×00001148 <+15>:   mov    EDX,0×38
0×00001155 <+28>:   add    EAX,EDX
0×00001157 <+30>:   mov    EBP, EAX
0×0000115a <+33>:   cmp    EBP,0xa
0×0000115e <+37>:   jge    0×1176 <main+61>
0×0000116a <+49>:   mov    eax,0×0
0×0000116f <+54>:   call   0×1030 <printf@plt>
```

Analisi

Definiamo principalmente di che si tratta Assembly, un linguaggio di programmazione a basso livello, utilizzato per comunicare direttamente con l'hardware di un dispositivo.

Struttura codice:

"0x00001141 <+8>: mov EAX,0x20"
Questa istruzione sposta il valore esadecimale 0x20 (che equivale a 32 in decimale) nel registro EAX. Il registro EAX è comunemente usato per operazioni aritmetiche, logiche, e per i valori di ritorno delle funzioni.

"0x00001148 <+15>: mov EDX,0x38"
Questa istruzione sposta il valore esadecimale 0x38 (che equivale a 56 in decimale) nel registro EDX. Il registro EDX è spesso usato insieme a EAX per operazioni che richiedono più di 32 bit.

"0x00001155 <+28>: add EAX,EDX"
Qui, si sommano i valori contenuti nei registri EAX e EDX, e il risultato viene salvato in EAX. Dato che EAX aveva 32 e EDX aveva 56, EAX
ora conterrà 88 (0x58 in esadecimale)

"0x00001157 <+30>: mov EBP, EAX"
Questa istruzione sposta il valore che è attualmente in EAX (88) nel registro EBP. Il registro EBP è comunemente usato come base
pointer per accedere a parametri e variabili locali in una funzione.

"0x0000115a <+33>: cmp EBP,0xa"
Qui, si confronta il valore in EBP con 0xa (10 in decimale). Questo è usato per controllare i salti condizionali (come jge, jle, ecc.) basati sul risultato del confronto

“0x0000115e <+37>: jge 0x1176 <main+61>”
Questa è una istruzione di salto condizionale che dice: "Salta all'indirizzo 0x1176 se il risultato del precedente cmp è maggiore o uguale (Greater or Equal, jge)". Se EBP è maggiore o uguale a 10, l'esecuzione continuerà dall'indirizzo 0x1176.

“0x0000116a <+49>: mov eax,0x0”
Qui, si sta resettando il registro EAX a 0. È comune fare questo prima di chiamare una funzione, soprattutto se EAX viene usato per il valore di ritorno.

“0x0000116f <+54>: call 0x1030 <printf@plt>”
Infine, questa istruzione chiama la funzione printf che si trova all'indirizzo 0x1030. plt sta per Procedure Linkage Table, che è usato nelle chiamate a funzioni di librerie condivise in un ambiente Linux.