Epicode Unit 3 Week 2

Assembly Language

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).

```
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>
```

Anzitutto si procede, per maggior chiarezza, con la conversione dei valori in oggetto con la loro forma decimale, ottenendo:

- **0x20 = 32**
- **0x38 = 56**
- **0xa = 10**
- **0x1176 = 4470**
- **0x0 = 0**
- **0x1030 = 4144**

Una volta fatto ciò, si procede con l'analisi del codice riga per riga.

- **Nella prima, si copia il valore 0x20(32) nel registro EAX;**
- **Nella seconda, si copia un secondo valore 0x38(56) in un secondo registro EDX**
- **Nella riga tre, si somma il valore contenuto nel registro EDX nel registro di destinazione EAX(88);**
- **Nella quarta riga il risultato contenuto in EAX(88 nella sua forma esadecimale) viene copiato nell'EXTENDED BASE POINTER, dunque il puntatore assumerà il valore di EAX;**
- **Nella quinta riga con il comando cmp("compare") verrà comparato il valore 0xa (10) con il valore contenuto nell'EBP, eseguendo un'operazione di sottrazione senza modificare i flag(78 decimale);**

- **In questa riga si esegue un "conditional jump" utilizzando i valori della riga precedente: nello specifico, il comando chiede di passare ad una data sezione di memoria (probabilmente dove è allocata la funzione "main" del programma) nel caso in cui l'EBP sia maggiore del valore sorgente della comparazione, ovvero 0xa(10). Dunque, essendo 88>10 si eseguirà il "salto" verso la porzione di memoria sopra indicata.**
- **Nella settima riga del codice da analizzare si copia il valore 0x0(0) nel registro EAX;**
- **Infine con il comando "Call" si esegue una chiamata alla funzione contenuta nella porzione di memoria 0x1030(4144). Nello specifico, molto probabilmente una funzione di "stampa" (printf).**