LUCA GASPARI

Analisi codici Assembly

2024

Traccia esercizio S10L3

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>
```

Analisi dei codici Assembly

Descrizione delle istruzioni:

0x00001141 <+8>: mov EAX,0x20

- Questa istruzione carica il valore esadecimale 0x20 (32 in decimale) nel registro EAX.
- EAX è un registro di uso generale nella CPU x86. Questa istruzione inizializza EAX con il valore 32.

0x00001148 <+15>: mov EDX,0x38

- Questa istruzione carica il valore esadecimale 0x38 (56 in decimale) nel registro EDX.
- EDX è un altro registro di uso generale. Questa istruzione inizializza EDX con il valore 56.

0x00001155 <+28>: add EAX,EDX

- Questa istruzione somma il valore contenuto in EDX al valore contenuto in EAX, memorizzando il risultato in EAX.
- Dopo questa operazione, EAX conterrà 32 + 56 = 88.

0x00001157 <+30>: mov EBP,EAX

- Questa istruzione copia il valore di EAX nel registro EBP.
- EBP è spesso utilizzato come puntatore di base per lo stack, ma in questo caso è utilizzato come registro di uso generale, dopo questa istruzione, EBP conterrà 88.

0x0000115a <+33>: cmp EBP,0xa

- Questa istruzione confronta il valore di EBP con 0xa (10 in decimale).
- Questa istruzione non modifica i registri, ma imposta i flag della CPU in base al risultato del confronto tra EBP e 10.

0x0000115e <+37>: jge 0x1176 <main+61>

- Questa istruzione salta all'indirizzo 0x1176 se il valore di EBP è maggiore o uguale a 0xa (10).
- Poiché EBP contiene 88, che è maggiore di 10, il salto avverrà.

0x0000116a <+49>: mov eax,0x0

- Questa istruzione carica il valore 0 nel registro EAX.
- Questa istruzione non verrà eseguita poiché il programma salta all'indirizzo 0x1176 a causa della precedente istruzione jge.

0x0000116f <+54>: call 0x1030 printf@plt

- Questa istruzione chiama la funzione printf tramite l'indirizzo 0x1030.
- Questa istruzione non verrà eseguita poiché il programma salta all'indirizzo 0x1176.

Analisi degli Indirizzi di Memoria e degli Offset

Ogni istruzione è preceduta da un indirizzo di memoria e da un offset relativo:

- Indirizzi di memoria: Specificano la posizione esatta di ciascuna istruzione nella memoria del programma.
- Offset relativi: Indicano la distanza dall'inizio della funzione o del blocco di codice corrente. Sono utili per comprendere la posizione relativa delle istruzioni.

Esempi:

x00001141 <+8>: L'istruzione mov EAX, 0x20 si trova all'indirizzo 0x00001141 e ha un offset di 8 byte dall'inizio della funzione.

0x00001148 <+15>: L'istruzione mov EDX, 0x38 si trova all'indirizzo 0x00001148 e ha un offset di 15 byte dall'inizio della funzione.