



S10/L3

Maurizio Marcantoni

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>
```

Svolgimento

Il codice fornito è un frammento di un programma scritto in linguaggio Assembly per un processore x86. L'assembly è un linguaggio di programmazione di basso livello che si avvicina molto al linguaggio macchina, cioè il linguaggio nativo del processore. È utilizzato per scrivere programmi che necessitano di un controllo molto preciso dell'hardware, come i driver di dispositivo, i sistemi operativi e i programmi che richiedono ottimizzazioni particolari per la velocità e l'efficienza. Di seguito fornisco l'analisi di ogni istruzione:

0x00001141<+8>:mov EAX,0x20 : Questa istruzione sposta il valore esadecimale 0x20(32 in base decimale) nel registro EAX. Il registro EAX è comunemente usato per operazioni aritmetiche e per restituire valori dalle funzioni.

0x00001148<+15>:mov EDX,0x38: Simile alla prima , questa istruzione sposta il valore esadecimale 0x38(56 in base decimale) nel registro EDX. EDX è spesso usato insieme a EAX per operazioni che richiedono risultati a 64-bit o per divisioni e moltiplicazioni

0x00001155<+28>:add EAX , EDX: questa istruzione somma il contenuto dei registri EAX e EDX e pone il risultato in EAX. In questo caso , somma 32 e 56 per ottenere 88

0x00001157<+30>:mov EBP , EAX : qui si muove il valore contenuto in EAX al registro EBP. EBP è spesso usato come base pointer per l'accesso ai parametri di funzione e alle variabili locali nello stack frame.

Svolgimento

0x0000115a<+33>:cmp EBP,0xa: questa istruzione confronta il valore nel registro EBP con il valore esadecimale 0xA(10 in base decimale). Il risultato di questo confronto è utilizzato per le istruzioni di salto condizionato.

0x0000115e<+37>:jge 0x1176<main+61>: questa istruzione di salto condizionato salta all'indirizzo 0x1176 se il risultato del precedente confronto(cmp) è maggiore o uguale (jge sta per "jump if greater or equal"). Quindi , se EBP è maggiore o uguale a 10 , il controllo del programma salta a quella posizione nel codice.

0x0000116a<+49>:mov eax ,0x0: sposta 0 nel registro EAX. Questo è spesso usato per impostare lo stato di uscita di una funzione, con 0 tipicamente indicante "nessun errore"

0x0000116f<+54>:call 0x1030<printf@plt> : questa istruzione chiama la funzione printf(presumibilmente per stampare qualcosa a schermo). La parte @plt indica che si sta utilizzando la procedure linkage table, una caratteristica dei sistemi operativi Unix-like che aiuta nella risoluzione degli indirizzi delle funzioni di librerie condivise durante l'esecuzione.