S10/L3

L’esercizio di oggi prevede di identificare ogni singola istruzione e affiancargli una spiegazione.

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>

Le istruzioni utilizzate nell’esercizio sono:

MOV
L’operazione più comune che si incontra quando si approccia il linguaggio assembly è «mov». Essa consente di spostare una variabile o un dato da una locazione ad un’altra. In altre parole, l’operazione «mov» viene utilizzata per leggere e scrivere in memoria.
La sintassi di «mov» è: mov destination, sorgente.
Dove destinazione e sorgente sono gli operandi che come abbiamo precedentemente detto possono essere locazioni di memoria, registri oppure valori immediati.

ADD
Add somma il valore contenuto in 2 registri (a) e (b) e salva il contenuto in b.

CMP
Cmp compara 2 valori ed in base al risultato modifica i flag ZF e CF.
L’istruzione «cmp» è simile all’istruzione «sub», ma a differenza di «sub» non modifica gli operandi.

La sintassi di cmp è **“cmp destinazione, sorgente”**
JGE

Salta alla locazione specificata se la destinazione è maggiore o uguale della sorgente nell’istruzione «cmp».

CALL
L’istruzione call passa l’esecuzione del programma alla funzione chiamata per la quale verrà creato un nuovo stack.

Adesso svolgiamo i passaggi uno di seguito all’altro

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

Viene spostato il valore decimale 32 nel registro EAX

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

Viene spostato il valore decimale 56 nel registro EDX

0x00001155 <+28>: add EAX,EDX

Viene sommato il registro EAX e EDX e allocato in EAX

0x00001157 <+30>: mov EBP, EAX

Viene spostato il contenuto del registro EAX (**88**) nel registro EBP

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

Compara il contenuto del registro EBP e il valore 10, quindi porta entrambi i flag a 0

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

Salta alla locazione specificata nell’indirizzo di memoria 1176 <main+61> se la destinazione è maggiore o uguale della sorgente nell’istruzione «cmp»

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

Sposto il valore 0 nel registro EAX

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

Con l’istruzione “call” la funzione chiamante passa l’esecuzione alla funzione chiamata che è all'indirizzo 0x1030 <print@plt> per la quale viene generato un nuovo stack