S10E3 - Natalino Imbrogno

Traccia

Dato il codice in linguaggio assembly per la CPU x86 quì di seguito, identificare 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 del codice

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

- *0x00001141* è l'indirizzo di memoria della prima istruzione del programma assembly fornito. In generale, un indirizzo di memoria è un numero esadecimale che identifica una posizione di memoria all'interno di un processo. In un programma assembly, gli indirizzi di memoria vengono utilizzati dunque per fare riferimento ai dati e ai codici memorizzati nella memoria;
- *<+8>* è una notazione utilizzata per indicare l'indirizzo di memoria della prossima istruzione;
- *mov* è un'istruzione che copia i dati da una posizione ad un'altra. Può copiare dati tra registri, tra memoria e registri, o tra memoria e memoria;

- *EAX* è un registro a 32 bit;
- *0x20* è un valore espresso in base al sistema di numerazione esadecimale che si traduce in decimale così
  2 x 16^1 + 0 x 16^0 = 32
- si può pertanto concludere che questa istruzione carica il numero *2* nel registro *EAX*.

0x00001148 <+15>: mov EDX,0x38
- *EDX* è un altro registro a 32 bit;
- si può pertanto concludere che questa istruzione carica il numero *56* nel registro *EDX.*

0x00001155 <+28>: add EAX,EDX
- *add* è un'istruzione che esegue l'addizione di due operandi. Tali operandi possono essere registri, valori immediati o posizioni di memoria;
- si può pertanto concludere che questa istruzione aggiunge il valore contenuto nel registro *EDX* al valore contenuto nel registro *EAX*;

0x00001157 <+30>: mov EBP, EAX
copia il valore di *EAX* nel registro *EBP*

0x0000115e <+37>: cmp EBP,0xa
- *cmp* è un'istruzione che confronta due operandi. Gli operandi possono essere registri, valori immediati o posizioni di memoria;
- *0xa* è un valore espresso attraverso il sistema di numerazione esadecimale corrispondente al numero decimale *10* dato che la lettera *a* in quel sistema di numerazione equivale proprio a questo numero;
- si può pertanto concludere che questa istruzione confronta il valore contenuto nel registro *EBP* con il numero *10*.
-

0x0000115e <+37>: jge 0x1176 <main+61>
- *jge* è un'istruzione di salto condizionato che salta all'indirizzo specificato se una determinata condizione è soddisfatta;
- l'istruzione salta all'indirizzo *0x00001176 <main+61>* se il valore contenuto nel registro *EBP* è maggiore o uguale a *10*. In assembly,

main+61 è un'espressione che viene utilizzata per calcolare l'indirizzo di memoria della riga di codice 61 nella funzione *main()*.

0x0000116a <+49>: mov EAX,0x0
carica il valore 0 nel registro *EAX*

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

- *call* è un'istruzione che viene utilizzata per chiamare una procedura, ovvero un blocco di codice che può essere eseguito in modo indipendente;
- *0x1030 <printf@plt>* è un'etichetta che indica l'indirizzo della procedura *printf()*;
- *printf()* è una procedura, ovvero una funzione della libreria standard di C per la stampa di testo;
- si può pertanto concludere che questa istruzione chiama la suddetta funzione, che viene utilizzata per visualizzare un messaggio sullo schermo.

Considerazioni conclusive

In definitiva, il programma inizia caricando il numero *32* nel registro *EAX*. Successivamente carica il numero *56* nel registro *EDX*. Quindi, somma i due numeri e memorizza il risultato nel registro *EBP*. Infine, confronta il numero contenuto nel registro *EBP* con *10*, nel senso che se è maggiore o uguale a 10, allora il programma stampa un messaggio, altrimenti ne stampa un altro.