

# Malware analysis ASSEMBLY X86

---

Antonio Perna  
Fabio Nobili

# S 10 L 3

---



# INTRODUZIONE

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>
```

# DESCRIZIONE ISTRUZIONI

**0x00001141 <+8>: mov EAX,0x20:** Carica il valore esadecimale 0x20 (32 in decimale) nel registro EAX.

**0x00001148 <+15>: mov EDX,0x38:** Carica il valore esadecimale 0x38 (56 in decimale) nel registro EDX.

**0x00001155 <+28>: add EAX,EDX:** Somma il valore nel registro EDX al valore nel registro EAX. Il risultato ( $32 + 56 = 88$ ) viene memorizzato nel registro EAX.

**0x00001157 <+30>: mov EBP, EAX:** Copia il valore contenuto nel registro EAX (88) nel registro EBP.

**0x0000115a <+33>: cmp EBP,0xa:** Confronta il valore nel registro EBP (88) con il valore esadecimale 0xa (10 in decimale).

**0x0000115e <+37>: jge 0x1176 <main+61>:** Se il valore nel registro EBP è maggiore o uguale a 10, salta all'indirizzo 0x1176 (main+61).

**0x0000116a <+49>: mov eax,0x0:** Carica il valore 0 nel registro EAX. Questo potrebbe essere preparatorio per una chiamata di funzione successiva.

**0x0000116f <+54>: call 0x1030 <printf@plt>:** Chiama la funzione ‘printf’, che si trova all'indirizzo 0x1030, passando il controllo del programma a questa funzione. Di solito questa chiamata stampa una stringa formattata sullo schermo.

# **Conclusioni**

---

Il motivo principale per cui queste istruzioni sono disposte in questo modo è per eseguire un'operazione aritmetica (la somma di due numeri), confrontare il risultato con un valore specifico e prendere una decisione basata su questo confronto. Il controllo condizionale (cmp e jge) è usato per decidere se saltare a un'altra parte del programma o continuare con l'esecuzione delle istruzioni successive. In sintesi, il codice confronta il risultato della somma di due valori con 10. Se il risultato è maggiore o uguale a 10, esegue un salto condizionale, altrimenti prosegue con la preparazione di una chiamata alla funzione printf.