Istruzioni Assembly

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

Istruzione nell'area di memoria 0x00001141 con offset di 8 byte <+8> dall'istruzione precedente che inserisce il valore 32 (20 in esadecimale) nel registro EAX.

Risultato:

Registro	Esadecimale	Decimale
EAX	20	32

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

Istruzione nell'area di memoria 0x00001148 con offset di 15 byte <+15> dall'istruzione precedente che inserisce il valore 56 (38 in esadecimale) nel registro EDX.

Risultato:

Registro	Esadecimale	Decimale
EAX	20	32
EDX	38	56

0x00001155 <+28>: add EAX,EDX

Istruzione nell'area di memoria 0x00001155 con offset di 28 byte <+28> dall'istruzione precedente che somma i valori presenti nei registri EAX e EDX e aggiorna il registro EAX con il risultato dell'addizione.

Risultato:

Registro	Esadecimale	Decimale
EAX	**58**	**88**
EDX	38	56

0x00001157 <+30>: mov EBP, EAX

Istruzione nell'area di memoria 0x00001157 con offset di 30 byte <+30> dall'istruzione precedente che copia il contenuto del registro EAX nel registro EBP. Il registro EBP è il registro che punta alla base dello stack, ovvero dell'area della RAM utilizzata per le variabili locali, parametri di funzioni e per supportare il flusso del programma.

Risultato:

Registro	Esadecimale	Decimale
EAX	58	88
EDX	38	56
EBP	**58**	**88**

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

Istruzione nell'area di memoria `0x0000115a` con offset di 33 byte `<+33>` dall'istruzione precedente che compara il valore 10 (`a` in esadecimale) al valore contenuto nel registro `EBP`, ma non modifica i dati di quest'ultimo, bensì modifica il registro **Status Flags (EFLAGS)**, più precisamente i flag **ZF** (Zero Flag) e **CF** (Carry Flag, che si utilizza per gestire eventuali riporti in un'operazione aritmetica).

- Se `EBP = 10` allora visto che 10 – 10 = 0: lo ZF viene impostato a 1 e il CF a 0
- Se `EBP > 10` allora la sottrazione dà un risultato positivo: lo ZF viene impostato a 0 e il CF a 0
- Se `EBP < 10` allora la sottrazione dà un risultato negativo: lo ZF viene impostato a 0 e il CF a 1

Risultato:

Registro	Esadecimale	Decimale
EAX	58	88
EDX	38	56
EBP	58	88
EFLAGS (CF)	**0**	**0**
EFLAGS (ZF)	**0**	**0**

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

Istruzione nell'area di memoria `0x0000115e` con offset di 37 byte `<+37>` dall'istruzione precedente che salta all'indirizzo `0x1176` all'interno della funzione `main` a un'offset di 61 byte dal suo inizio se il valore nel registro `EAX` è maggiore o uguale a zero (`jge = Jump if Greater than or Equal`), ovvero se il risultato dell'istruzione `cmp` ha settato il Carry Flag a 0.

Risultato:

Salta all'indirizzo `0x1176 <main+61>`

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

Istruzione nell'area di memoria `0x0000116a` con offset di 49 byte `<+49>` dall'istruzione precedente che copia il valore 0 (0 in esadecimale) nel registro `EAX`.

Registro	Esadecimale	Decimale
EAX	**0**	**0**
EDX	38	56
EBP	58	88
EFLAGS (CF)	0	0
EFLAGS (ZF)	0	0

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

Istruzione nell'area di memoria `0x0000116f` con offset di 54 byte `<+54>` dall'istruzione precedente che chiama (`call`) la funzione situata all'indirizzo di memoria `0x1030`. Questo indirizzo è associato alla funzione `printf`. `printf@plt` fa riferimento alla **Procedure Linkage Table (PLT)** utilizzata in molti sistemi Unix-like per gestire le chiamate a funzioni dinamiche.