REPORT ASSEMBLY X86

TRACCIA:
Dato il seguente codice convertire i seguenti numeri esadecimali, in decimali, attraverso calcolatrice elettronica e fornire breve spiegazione per ognuna di esse.

0x00001141 <+8>: mov EAX, 0x20
0×00001148 <+15>:mov EDX, 0×38
0x00001155 <+28>: add EAX,EDX
0×00001157 <+30>: mov EBP, EAX
0x0000115a <+33>: cmp EBP, 0xa
0×0000115e <+37>: jge 0×1176 <main+61>
0×0000116a <+49>:mov EAX, 0x0
0x0000116f <+54>: call 0×1030 <printf@plt>

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

- Tramite comando mov andiamo ad impostare come valore immediato 0x20 in esadecimale oppure 32 in decimale al resistro EAX. L'equivalente di questa stringa in C sarebbe: a=32;

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

- Tramite comando mov andiamo ad impostare come valore immediato 0X38 in esadecimale oppure 56 in decimale al registro EDX. L'equivalente di questa stringa in C sarebbe d=56;

0x00001155 <+28>: add EAX, EDX

- Tramite il comando add andiamo a sommare il registro EDX in EAX andando ad aggiornare in 88. Il suo equivalente in C sarebbe a=a+b;

0x00001157 <+30>: mov EBP, EAX

- Tramite il comando mov andiamo ad impostare il valore del registro EAX al registro EBX. L'equivalente di questa stringa in C sarebbe: b=a;

0x0000115a <+33>: cmp EBP, Oxa

- Tramite il comando cmp andiamo a fare un sub. Il quale equivale alla sottrazione, senza aggiornare il valore di EBP. Questo comando serve per verificare se il numero 0, ha avuto un riporto oppure nessuno dei due. Nel nostro caso abbiamo: EBP (88) - 10 = 78. Ergo, In questo caso, abbiamo la Zero Flag a 0 e la Carry Flag a 0;

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

- Tramite il comando jge andiamo a fare un salto nell'indirizzo di memoria 0x1176 solo nel caso in cui la destinazione sia di valore maggiore o uguale alla sorgente;

0x0000116a <+49>: mov EAX. 0x0

- Tramite comando mov andiamo ad impostare come valore immediato 0X0 in esadecimale oppure O m decimale al registio EAX. L' equivalente di questa stringa in C Sarebbe: a=0;

0x0000116f <+54>: call 0x1030 <printf@plt>
L'istruzione call viene utilizzata per richiamare una funzione. Questa istruzione segue due operazioni:

- Spinge l'indirizzo di ritorno (indirizzo successivo all'istruzione CALL) sullo Stack. Cambia 'EIP nella destinazione della chiamata. In questo modo si trasferisce il controllo alla destinazione della chiamata e si avvia l'esecuzione.
- Printf@plt è in realtà un piccolo sub, ovvero una porzione di codice utilizzata per simulare il comportamento di funzionalità software o interfaccia COM che può fungere da temporaneo sostituto di codice ancora da sviluppare, il quale richiamerà le funzione di printf rendendo più veloci le chiamate successive. La vera funzione printf è mappata in una posizione arbitraria in un dato processo, cosi come il codice che cerca di chiamarla.