Assembly x86

Traccia

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.

- **0x00001141 <+8>: mov EAX,0x20 / 0x00001141 <+8>**: mov EAX,32 = assegna il valore 32 al registro EAX
- **0x00001148 <+15>: mov EDX,0x38 / 0x00001148 <+15>**: mov EDX,56 = assegna il valore 56 al registro EDX
- **0x00001155 <+28>: add EAX,EDX** = somma i valori dei registri EAX e EDX e salva il risultato nel registro EAX
- **0x00001157 <+30>: mov EBP, EAX** = assegna il valore del registro EAX al registro EBP
- **0x0000115a <+33>: cmp EBP,0xa / 0x0000115a <+33>: cmp EBP,10** = confronta il valore nel registro EBP con il valore di 10, poi se EBP è uguale a 10 modifica la ZF (Zero Flag) a 1 e la CF (Carry Flag) a 0, se EBP è inferiore a 10 imposta la ZF a 0 e la CF a 1, se EBP è superiore a 10 imposta la ZF a 0 e la CF a 0
- **0x0000115e <+37>: jge 0x1176 <main+61>** = è un salto condizionale che prende in considerazione il risultato dell'operazione precedente e se esso è uguale o maggiore a zero salta all'indirizzo 0x1176 <main+61> dove main+61 indica una locazione di memoria situata 61 byte dopo l'inizio della funzione main
- **0x0000116a <+49>: mov EAX,0x0 / 0x0000116a <+49>: mov EAX,0** = assegna il valore 0 al registro EAX
- **0x0000116f <+54>: call 0x1030 <printf@plt>** = chiama la funzione printf situata all'indirizzo 0x1030 nella PLT (Procedure Linkage Table) e dopo averla eseguita prosegue dall'istruzione successiva al call