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. 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).

|  |  |
| --- | --- |
| CODICE IN ASSEMBLY | DESCRIZIONE |
| 0x00001141 <+8>: mov EAX,0x20 | il codice muove il valore decimale 32 nel registro EAX(registro accumulatore, usato per operazioni aritmetiche e logiche.) |
| 0x00001148 <+15>: mov EDX,0x38 | Il codice muove il valore decimale 56 nel registro EDX(registro di dati, spesso utilizzato con il registro EAX per operazioni di moltiplicazione e divisione, viene utilizzato anche per operazioni di input/output. ) |
| 0x00001155 <+28>: add EAX,EDX | Il codice fa la somma dei registri EAX e EDX e aggiorna il registro EAX con il risultato: 32 + 56 = 88 ora il valore di EAX è 88 |
| 0x00001157 <+30>: mov EBP, EAX | Il codice muove il valore di EAX(88) del registro EBP(utilizzato principalmente per la gestione dello stack, in particolare per il controllo dei frame dello stack durante le chiamate di funzione. Lo stack è una struttura di dati LIFO (Last In, First Out), dove l'ultimo elemento inserito è il primo a essere rimosso come a una pila di piatti dove si puoi solo aggiungere un piatto in cima e rimuovere solo il piatto in cima.) |
| 0x0000115a <+33>: cmp EBP,0xa | Compara il valore in EBP(88) con il valore 0xa (valore decimale 10). |
| 0x0000115e <+37>: jge 0x1176 <main+61> | Il programma fa un salto incondizionato se la condizione di prima è maggiore o uguale: 88>10 quindi il salto viene effettuato e il programma continuerà dal’ indirizzo 0x1176 |
| 0x0000116a <+49>: mov eax,0x0 | Il programma muove il valore decimale 0 nel registro EAX. |
| 0x0000116f <+54>: call 0x1030 <printf@plt> | Il programma chiama la funzione printf nell’ indirizzo 0x1030 |