ESERCITAZIONE S10 L3

Assembly x86

Dato il codice in Assembly per la CPU x86, identificare lo scopo di ogni istruzione, inserendo una descrizione per ogni riga di codice.

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

è un'operazione di spostamento dei dati e ha lo scopo di assegnare il valore esadecimale 0x20, che corrisponde al valore 32 in decimale, al registro EAX.

Il registro EAX è uno dei registri generali nell'architettura x86 e viene spesso utilizzato per vari scopi, inclusi calcoli e manipolazioni di dati.

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

Questa istruzione è un'altra operazione di spostamento dei dati e ha lo scopo di assegnare il valore esadecimale 0x38, che corrisponde a 56 in decimale, al registro EDX. Come nel caso precedente, il registro EDX è un registro generale nell'architettura x86, utilizzato per vari scopi, come il mantenimento di valori temporanei durante l'esecuzione del programma.

0x00001155 <+28>: add EAX,EDX

Questa istruzione esegue un'operazione di addizione tra i valori contenuti nei registri EAX ed EDX e salva il risultato nel registro EAX. In altre parole, il valore corrente nel registro EAX viene sommato al valore corrente nel registro EDX, e il risultato di questa operazione di addizione viene memorizzato nel registro EAX.

0x00001157 <+30>: mov EBP, EAX

Questa istruzione muove il contenuto del registro EAX nel registro EBP.

Nella maggior parte dei casi, EBP è utilizzato come registro del puntatore di base del frame dello stack.

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

Questa istruzione esegue un'operazione di confronto tra il valore contenuto nel registro EBP e il valore 0xa (10 in decimale). La differenza tra i due valori viene calcolata, ma il risultato non viene memorizzato; in quanto lo scopo è quello di aggiornare gli indicatori di stato della CPU in base al risultato del confronto.

Successivamente, le istruzioni condizionali possono essere utilizzate per prendere decisioni basate sul risultato di questo confronto. Ad esempio, potrebbe essere seguito da un salto condizionale a una determinata posizione nel codice a seconda del risultato del confronto (se il confronto è vero o falso).

Il *salto condizionale* si riferisce a un'istruzione di salto (jump) nel flusso del programma che avviene solo se una specifica condizione è soddisfatta. In linguaggio assembly, le istruzioni di salto condizionale sono spesso utilizzate dopo un'istruzione di confronto (cmp) o dopo un'operazione aritmetica che imposta i flag della CPU

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

È un'istruzione di salto condizionale, indica "Jump if Greater Than or Equal" (salta se maggiore o uguale).
L'indirizzo specificato, 0x1176, rappresenta il punto nel codice dove il flusso del programma si dirigerà se la condizione "maggiore o uguale" è verificata.
Il salto condizionale indica che, se il risultato del confronto precedente è maggiore o uguale, il controllo del programma verrà trasferito all'indirizzo 0x1176.
Altrimenti, il flusso del programma continuerà normalmente all'istruzione successiva.

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

Questa istruzione assegna il valore immediato 0x0 al registro EAX. Ossia, sta impostando il registro EAX a 0.

Questo tipo di operazione è spesso utilizzato per azzerare un registro o inizializzare una variabile a zero.

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

L’ istruzione *call* viene utilizzata per chiamare una funzione o una procedura. Nel caso specifico, sta chiamando la funzione ***printf*** situata all'indirizzo 0x1030, presente nella Procedure Linkage Table (PLT). La funzione printf è comunemente utilizzata per stampare dati in output.

Dopo questa chiamata di funzione, il flusso del programma si sposterà all'indirizzo specificato (0x1030 printf@plt), eseguirà la logica all'interno della funzione printf, e quindi tornerà al punto successivo nell'esecuzione del programma dopo la chiamata (call).