L10/L3

ASSEMBLY X86

GIULIA FIACCHI

TRACCIA

Nella lezione teorica del mattino, abbiamo visto i fondamenti del linguaggio Assembly.
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).

1. 0x00001141 <+8>: mov EAX,0x20
2. 0x00001148 <+15>: mov EDX,0x38
3. 0x00001155 <+28>: add EAX,EDX
4. 0x00001157 <+30>: mov EBP, EAX
5. 0x0000115a <+33>: cmp EBP,0xa
6. 0x0000115e <+37>: jge 0x1176 <main+61>
7. 0x0000116a <+49>: mov eax,0x0
8. 0x0000116f <+54>: call 0x1030 <printf@plt>

ESERCIZI

ISTRUZIONE 1

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

mov = move; non si tratta di uno spostamento fisico, ma di una copia. Copia il valore dal secondo operando al primo operando.

EAX = Questo è uno dei registri generali a 32 bit della CPU x86. I registri sono usati per immagazzinare dati temporaneamente durante l'esecuzione del programma.

0x20 = in decimale equivale a 32

Carica il valore 0x20 (32 in decimale) nel registro EAX

ISTRUZIONE 2

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

mov = move; non si tratta di uno spostamento fisico, ma di una copia. Copia il valore dal secondo operando al primo operando.

EDX = Questo è uno dei registri generali a 56 bit della CPU x86. I registri sono usati per immagazzinare dati temporaneamente durante l'esecuzione del programma.

0x36 = in decimale equivale a 56

Carica il valore 0x38 (56 in decimale) nel registro EDX

ESERCIZI

ISTRUZIONE 3

0x00001155 <+28>: add EAX,EDX

add = somma il valore di EDX a EAX e immagazzina il risultato in EAX

EAX = Questo è uno dei registri generali a 32 bit della CPU x86. I registri sono usati per immagazzinare dati temporaneamente durante l'esecuzione del programma.

EDX = Questo è uno dei registri generali a 56 bit della CPU x86. I registri sono usati per immagazzinare dati temporaneamente durante l'esecuzione del programma.

EAX = 32 + 56 = 88

ISTRUZIONE 4

0x00001157 <+30>: mov EBP, EAX

mov = move; non si tratta di uno spostamento fisico, ma di una copia. Copia il valore dal secondo operando al primo operando.

EBP = registro a 32 bit utilizzato principalmente per la gestione dello stack frame durante le chiamate di funzione

EAX = Questo è uno dei registri generali a 32 bit della CPU x86. I registri sono usati per immagazzinare dati temporaneamente durante l'esecuzione del programma.

Memorizza il risultato della somma (88) nel registro EBP

ESERCIZI

ISTRUZIONE 5

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

cmp = modifica i flag ZF (zero flag) e CF (carry flag, che si utilizza per gestire eventuali riporti in un'operazione aritmetica)

EBP = registro a 32 bit utilizzato principalmente per la gestione dello stack frame durante le chiamate di funzione

Oxa = in valore decimale equivale a 10

Confronta il valore 88 con 10 per determinare il flusso di esecuzione successivo

ISTRUZIONE 6

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

jge = Salta all'indirizzo 0x1176 se EBP è maggiore o uguale a 10.

0x1176 = indica l'indirizzo di memoria a cui il flusso di esecuzione deve saltare se la condizione specificata è vera

<main+61> = ndica una posizione relativa all'inizio della funzione main.

Poiché 88 è maggiore di 10, il salto avverrà e il controllo verrà trasferito all'istruzione all'indirizzo 0x1176.

ESERCIZI

ISTRUZIONE 7

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

mov = move; non si tratta di uno spostamento fisico, ma di una copia. Copia il valore dal secondo operando al primo operando.

eax = Questo è uno dei registri generali a 32 bit della CPU x86. I registri sono usati per immagazzinare dati temporaneamente durante l'esecuzione del programma.

0x0= in decimale equivale a 0

Inizializza EAX con 0. Questa istruzione sarà eseguita solo se il salto (jge) non viene effettuato, cioè se EBP fosse stato minore di 10

ISTRUZIONE 8

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

call = utilizzata per chiamare una funzione o un subroutine
0x1030 = indirizzo di memoria che rappresenta la posizione di una funzione o di un'istruzione in memoria che viene chiamata
<printf@plt> = utilizzata nei programmi compilati per gestire le chiamate a funzioni esterne

Chiama la funzione printf, presumibilmente per stampare un messaggio. Questa istruzione sarà eseguita solo se il salto (jge) non viene effettuato