REPORT DETTAGLIATO
SULL'ESERCIZIO DI ASSEMBLY

01

S10L3

MICHELE COVI

INTRODUZIONE AL LINGUAGGIO ASSEMBLY E CONVERSIONE NUMERICA

- Cos'è il Codice Assembly?

Il linguaggio Assembly è una forma di rappresentazione del codice macchina che è più facilmente leggibile dagli esseri umani. Esso è strettamente legato all'architettura del processore per cui è scritto, in questo caso l'architettura x86. Assembly fornisce un controllo diretto sull'hardware, permettendo agli sviluppatori di ottimizzare il codice per specifiche esigenze di prestazione e di funzionamento. Le istruzioni in Assembly sono composte da mnemonici, come MOV, ADD, e SUB, che rappresentano le operazioni base eseguite dalla CPU.

CONVERSIONE DA ESADECIMALE A DECIMALE

Il sistema esadecimale (base 16) è spesso usato in programmazione per una rappresentazione più compatta dei numeri binari, che sono la base del funzionamento dei computer. Per convertire un numero da esadecimale a decimale:

- Ogni cifra esadecimale viene moltiplicata per 16^n, dove n è la posizione della cifra da destra (partendo da 0).

- La somma dei prodotti fornisce il valore decimale equivalente.

Esempi di Conversione:

- 0x20 in esadecimale è 2×16^1+0×16^0= 32 in decimale.

- 0x38 in esadecimale è 3×16^1+8×16^0= 56 in decimale.

CONVERSIONE DA DECIMALE A ESADECIMALE

Il sistema esadecimale (base 16) è spesso usato in programmazione per una rappresentazione più compatta dei numeri binari, che sono la base del funzionamento dei computer. Per convertire un numero da decimale a esadecimale, si seguono questi passaggi:

- Divisione per 16: Dividi il numero decimale per 16. Il quoziente ottenuto sarà utilizzato per la prossima divisione, mentre il resto rappresenta la cifra esadecimale meno significativa.

- Ripeti il Processo: Ripeti la divisione per 16 con il quoziente ottenuto, finché il quoziente stesso non è zero.

- Il resto è ciò che rimane dopo aver sottratto il prodotto del quoziente intero per la base dal numero originale.

Esempi Pratici:

- Per convertire il numero decimale 56 in esadecimale, procediamo come segue:
  - 56 / 16 = 3.5, Tuttavia, in queste operazioni, consideriamo solo la parte intera del quoziente. Quindi, il quoziente intero è 3. Successivamente si passa al calcolo del resto, 56 - (3 x 16) = 48, il resto sarà 56-48, ovvero 8.
  - Dopo aver ottenuto il quoziente di 3 e il resto di 8:
  - La cifra esadecimale meno significativa è 8 (ottenuta dal resto).
  - La cifra successiva è 3 (ottenuta dal quoziente intero).
  - Quindi, il numero 56 in decimale si converte in 38 in esadecimale, che si scrive come 0x38.

RISOLUZIONE DELLA TRACCIA DI CODICE ASSEMBLY

01

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

- Istruzione: mov EAX, 0x20
- Azione: Carica il valore esadecimale 0x20 (decimale 32) nel registro EAX.
- Scopo: Inizializza EAX con un valore specifico, probabilmente per un uso successivo in calcoli o come parametro di una funzione.

02

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

- Istruzione: mov EDX, 0x38
- Azione: Carica il valore esadecimale 0x38 (decimale 56) nel registro EDX.
- Scopo: Prepara EDX con un valore specifico, utile per operazioni di calcolo o come parte di una procedura più complessa.

03

0x00001155 <+28>: add EAX,EDX

- Istruzione: add EAX, EDX
- Azione: Somma il valore in EDX a quello in EAX, aggiornando EAX con il risultato.
- Scopo: Realizza un calcolo aritmetico, aggiungendo due valori e salvando il risultato per usi futuri

04

0x00001157 <+30>: mov EBP,EAX

- Istruzione: mov EBP, EAX
- Azione: Trasferisce il valore di EAX (ora 88) a EBP.
- Scopo: Potrebbe essere usato per configurare EBP come puntatore base per accessi di memoria relativi a uno stack frame.

05

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

- Istruzione: cmp EBP, 0xa
- Azione: Compara EBP con 0xa (decimale 10), influenzando i flag di stato.
- Scopo: Determina come EBP si confronta con 10, per controllare il flusso del programma basandosi su questa comparazione.

06

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

- Istruzione: jge 0x1176
- Azione: Esegue un salto all'indirizzo specificato se EBP è maggiore o uguale a 10.
- Scopo: Controlla il flusso del programma basato sul risultato del confronto, saltando a differenti parti del codice a seconda delle condizioni.

07

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

- Istruzione: mov eax, 0x0
- Azione: Imposta EAX a zero.
- Scopo: Resetta EAX, tipicamente in preparazione per una nuova operazione o per impostare un valore di ritorno di una funzione.

08

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

- Istruzione: call 0x1030
- Azione: Chiama la funzione printf situata all'indirizzo 0x1030.
- Scopo: Stampa output o gestisce stringhe/formati, utilizzando printf per comunicare con l'utente o per altri scopi di output.