ASSEMBLY X86

S10/L3

ELEONORA VIOLA

TABLE OF CONTENT

03. INTRODUCTION

04. PROCEDURE

05. CONCLUSION

INTRODUCTION

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).

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

PROCEDURE

Indirizzo	Istruzione	Descrizione
0x00001141 <+8>	`mov EAX, 0x20`	Copia il valore esadecimale 0x20 (32 decimale) nel registro EAX
0x00001148 <+15>	`mov EDX, 0x38`	Copia il valore esadecimale 0x38 (56 decimale) nel registro EDX
0x00001155 <+28>	`add EAX, EDX`	Somma il valore del registro EDX (56) al valore del registro EAX (32), risultato in EAX (88)
0x00001157 <+30>	`mov EBP, EAX`	Copia il valore del registro EAX (88) nel registro EBP
0x0000115a <+33>	`cmp EBP, 0xa`	Confronta il valore del registro EBP (88) con il valore esadecimale 0xa (10 decimale)
0x0000115e <+37>	`jge 0x1176 <main+61>`	Salta all'indirizzo 0x1176 se il valore nel registro EBP (88) è maggiore o uguale a 10
0x0000116a <+49>	`mov EAX, 0x0`	Copia il valore 0 nel registro EAX
0x0000116f <+54>	`call 0x1030 <printf@plt>`	Chiama la funzione `printf`

CONCLUSION

Il codice Assembly analizzato ha il seguente flusso:

1. Carica i valori esadecimali 0x20 (32 decimale) e 0x38 (56 decimale) nei registri EAX e EDX.
2. Somma questi valori, memorizzando il risultato (88 decimale) nel registro EAX.
3. Copia il valore del registro EAX nel registro EBP.
4. Confronta il valore di EBP con il numero esadecimale 0xa (10 decimale).
5. Se il valore in EBP è maggiore o uguale a 10 (cosa che è sempre vera dato che EBP contiene 88), esegue un salto all'indirizzo 0x1176, bypassando le istruzioni successive.
6. Le istruzioni mov EAX, 0x0 e call printf non verranno mai eseguite poiché il confronto cmp EBP, 0xa sarà sempre soddisfatto, facendo eseguire il salto condizionato jge.

Quindi, con i valori forniti nel codice, le istruzioni che seguono il confronto e il salto condizionato (mov EAX, 0x0 e call printf) risultano non eseguite. Il codice essenzialmente carica, somma e confronta i valori, determinando un salto che evita l'esecuzione delle istruzioni finali.