S10-L3

Traccia:

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).

```
0×00001141 <+8>:     mov    EAX,0×20
0×00001148 <+15>:    mov    EDX,0×38
0×00001155 <+28>:    add    EAX,EDX
0×00001157 <+30>:    mov    EBP, EAX
0×0000115a <+33>:    cmp    EBP,0xa
0×0000115e <+37>:    jge     0×1176 <main+61>
0×0000116a <+49>:    mov    eax,0×0
0×0000116f <+54>:    call   0×1030 <printf@plt>
```

Codice e Spiegazione

1. **0x00001141 <+8>: mov EAX,0x20**
   - Quest'istruzione carica il valore esadecimale `0x20` (32 in decimale) nel registro `EAX`.
   - **Calcolo**: 2×16^1+0×16^0 = 32
   - Istruzione: `mov` è un'istruzione che copia dati.
     In questo caso, copia il valore immediato
     `0x20` nel registro `EAX`.

     Operazione: `EAX` ← 32
   - **0x00001141 :** Cella di memoria
   - **<+8> :** A 8 posizioni dall'inizio del codice

2. **0x00001148 <+15>: mov EDX,0x38**
   - **Descrizione**: Carica il valore esadecimale `0x38` (56 in decimale) nel registro `EDX`.
   - **Calcolo**: 3×161+8×160=48+8=56
   - **Spiegazione**: Questa istruzione simile alla precedente carica un valore immediato, `0x38`, nel registro `EDX`.
   - **Operazione**: `EDX` ← 56

3. **0x00001155 <+28>: add EAX,EDX**
   - **Descrizione**: Somma il valore contenuto nel registro `EDX` al valore nel registro `EAX`.
   - **Spiegazione**: Il risultato della somma viene immagazzinato in `EAX`. Quindi, `EAX` ora contiene `32 + 56 = 88`.
   - **Operazione**: `EAX` ← `EAX` + `EDX` (32 + 56 = 88)

4. **0x00001157 <+30>: mov EBP,EAX**
   - **Descrizione**: Copia il valore nel registro `EAX` nel registro `EBP`.
   - **Spiegazione**: `EBP` ora contiene il valore `88`, che è stato precedentemente calcolato in `EAX`.
   - **Operazione**: `EBP` ← `EAX` (88)

5. **0x0000115a <+33>: cmp EBP,0xa**
   - **Descrizione**: Confronta il valore nel registro `EBP` con il valore esadecimale `0xa` (10 in decimale).
   - **Calcolo**: 10×160=10
   - **Spiegazione**: Questa istruzione serve a impostare i flag del processore in base alla differenza tra `EBP` e `0xa`. Non modifica il contenuto dei registri.
   - **Operazione**: Confronta `EBP` (88) con 10.

6. **0x0000115e <+37>: jge 0x1176 <main+61>**
   - **Descrizione**: Salta all'indirizzo `0x1176` se il risultato della precedente comparazione è maggiore o uguale a zero.
   - **Spiegazione**: `jge` significa "jump if greater or equal".
     Se
     `EBP` è maggiore o uguale a `0xa`, il flusso del programma salta all'indirizzo `0x1176`.
   - **Operazione**: Se `EBP` è maggiore o uguale a 10, salta all'indirizzo `0x1176`.
   - **Dettaglio**: Controlla il flag della condizione impostato da `cmp`.
     Se
     `EBP` ≥ 10, il controllo del programma salta all'indirizzo specificato.

7. **0x0000116a <+49>: mov eax,0x0**
   - **Descrizione**: Imposta il registro `eax` a `0`.
   - **Spiegazione**: Questa operazione di reset del registro `eax` è spesso utilizzata per preparare un valore di ritorno, indicando successo o un altro codice di stato.
   - **Operazione**: `eax` ← 0
   - **Dettaglio**: Imposta il registro `eax` a 0. Potrebbe essere usato per indicare un valore di ritorno o stato finale.

8. **0x0000116f <+54>: call 0x1030 printf@plt**
   - **Descrizione**: Chiama la funzione `printf` (che probabilmente si trova all'indirizzo `0x1030`).
   - **Spiegazione**: Questa istruzione provoca un salto alla subroutine `printf` passando il controllo a essa. `printf` è usato per stampare una stringa, ed è probabile che venga utilizzato per visualizzare l'output del programma.
   - **Operazione**: Chiama la funzione `printf` per stampare un output.

Descrizione e Utilizzo dei Registri

1. **EAX (Extended Accumulator Register)**
   - **Uso**: Questo registro è spesso utilizzato per eseguire operazioni aritmetiche e logiche.
     In questo programma, viene utilizzato per accumulare il risultato della somma di due numeri (
     `0x20` e `0x38`).
2. **EDX (Extended Data Register)**
   - **Uso**: `EDX` viene utilizzato per immagazzinare il valore `0x38` (56 in decimale) prima di essere aggiunto a `EAX`.
     È anche comunemente usato come un registro di dati generico.

3. **EBP (Extended Base Pointer)**
    - **Uso**: In questo codice, `EBP` viene utilizzato come una variabile temporanea per immagazzinare il risultato della somma presente in `EAX`.
      Tradizionalmente,
      `EBP` è usato come puntatore al frame dello stack, ma in questo caso, viene utilizzato come registro generico.