Assembly

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>
```

ASSEMBLY COMANDI BASE

MOV (Move): Questo comando copia i dati da una posizione all'altra. Viene utilizzato per assegnare valori a registri, variabili o locazioni di memoria. Ad esempio, MOV AX, BX copia il valore del registro BX nel registro AX.

ADD (Addizione): Questo comando somma due valori. Può essere utilizzato per sommare valori tra registri, variabili o costanti. Ad esempio, ADD AX, BX somma il valore del registro BX al valore del registro AX.

SUB (Sottrazione): Questo comando sottrae un valore da un altro. Può essere utilizzato per sottrarre valori tra registri, variabili o costanti. Ad esempio, SUB AX, BX sottrae il valore del registro BX dal valore del registro AX.

PUSH (Inserimento nello stack): Questo comando inserisce un dato nello stack. Può essere utilizzato per salvare registri, parametri o variabili locali nello stack. Ad esempio, PUSH AX inserisce il valore del registro AX nello stack.

POP (Estrazione dallo stack): Questo comando estrae un dato dallo stack. Viene utilizzato per ripristinare valori da uno stack precedentemente salvati con il comando PUSH. Ad esempio, POP AX estrae un valore dallo stack e lo memorizza nel registro AX.

JMP (Salto): Questo comando esegue un salto incondizionato a un'altra posizione del programma. Viene utilizzato per modificare il flusso di esecuzione. Ad esempio, JMP label salta all'etichetta label nel programma.

INT (Interrupt): Questo comando genera un'interruzione nel programma. Può essere utilizzato per chiamare funzioni o servizi di sistema. Ad esempio, INT 21h esegue un'interruzione per accedere ai servizi del DOS.

Esercizio

0x00001141 <+8>: *mov* EAX, 0x20
(*move*=sposta/copia dati tra memoria e registri)
0x20 (32 decimale) viene copiato nel registro EAX

Spiegazione
0x00001141 → Indirizzo di memoria
<+8> → è l’offset ovvero la distanza dall’inizio della funzione corrente all'istruzione corrispondente
0x20 → corrisponde a 32 in decimale
(Dalla Memoria è stato preso 32 ed è stato copiato (caricato/sovrascritto) nel registro EAX della CPU)

Il registro della CPU può prelevare dati da vari tipi di memoria:
Memoria Cache
RAM
Memoria virtuale (RAM fisica e porzione di archiviazione su disco rigido)
ROM
Altro (VRAM, PROM, EPROM)

In generale, i registri della CPU possono prelevare dati e istruzioni da diverse fonti di memoria, ma il tipo specifico di memoria dipenderà dall'architettura del sistema e dalle modalità di accesso alla memoria definite dal processore.

Durante l'avvio di un programma, la CPU carica i registri con i dati necessari per l'esecuzione delle istruzioni. Successivamente, il sistema operativo si occupa di gestire il caricamento dei dati e delle istruzioni nella memoria cache, RAM, memoria virtuale, ROM e altre memorie specializzate, se necessario.

/\

0x00001148 <+15>: *mov* EDX, 0x38
(***move***=sposta/copia dati tra memoria e registri)
0x38 (56 decimale) viene caricato nel registro EDX

0x00001155 <+28>: *add* EAX, EDX
(***add***=addizione, salva il risultato nel registro di destinazione)
EDX viene aggiunto a EAX e il risultato viene memorizzato in EAX (32+56=88)

0x00001157 <+30>: *mov* EBP, EAX
(***move***=sposta/copia dati tra memoria e registri)
EAX (88) viene copiato nel registro EBP

0x0000115a <+33>: *cmp* EBP, 0xa
(***compare***=confronto, imposta i flag in base al risultato)
Confronta il valore di EBP con 0xa (10 decimale)

0x0000115e <+37>: *jge* 0x1176 <main+61>
(***jump if greater than or equal***=salto condizionato all'indirizzo, se il confronto precedente ha dato risultato >=)
Se EBP >= 10, salta all'indirizzo 0x1176 della funzione main (secondo offset 61 byte dall'inizio di "main")

0x0000116a <+49>: *mov* EAX, 0x0
(***move***=sposta/copia dati tra memoria e registri)
Il valore di EAX viene impostato a 0

0x0000116f <+54>: *call* 0x1030 <print@plt>
(***call***=chiamata alla funzione)
Chiamata alla funzione print all'indirizzo 0x1030