Report sull'Analisi del Codice Assembly per CPU x86

Data 07/02/24

Introduzione

Durante la lezione teorica del mattino, abbiamo esaminato i fondamenti del linguaggio Assembly, un linguaggio di basso livello utilizzato per scrivere codice macchina in modo più comprensibile rispetto al linguaggio macchina binario. In questo report, analizzeremo il codice Assembly per la CPU x86 fornito, identificando lo scopo di ogni istruzione e fornendo una descrizione per ogni riga di codice.

Codice Assembly

```
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>
```

La sezione a sinistra di ogni riga prima dei due punti rappresenta l'indirizzo di memoria in cui viene memorizzata l'istruzione corrispondente, scritta dopo i due punti.

Il valore tra le parentesi angolari invece rappresenta un commento nel linguaggio Assembly. In questo caso rappresenta la riga di testo in base alla prima riga di programma.

Ad esempio <+8> significa +8 righe rispetto la prima riga del programma.

Descrizione delle Istruzioni

Per prima cosa convertiamo i numeri esadecimali nei corrispondenti valori interi

0x20 = 32

0x38 = 56

0xa = 10

0x1176 = 4470

0x0 = 0

0x1030 = 4144

Riscriviamo il codice con i numeri interi :

0x00001141 <+8>: mov EAX, 32

0x00001148 <+15>: mov EDX, 56

0x00001155 <+28>: add EAX,EDX

0x00001157 <+30>: mov EBP, EAX

0x0000115a <+33>: cmp EBP, 10

0x0000115e <+37>: jge 4470 <main+61>

0x0000116a <+49>: mov EAX, 0

0x0000116f <+54>: call 4144 <printf@plt>

Mov EAX, 0x20 -→ Sta dicendo di spostare il valore 32 al registro di memoria EAX (32 bit)

Mov EDX, 0x38 →Sta dicendo di spostare il valore 56 al registro di memoria EDX (32 bit)

Add EAX, EDX →Sta dicendo di sommare i valori presenti nei due registri di memoria dichiarati

Mov EBP, EAX →Sta dicendo di spostare il valore presente in EAX nel registro EBP

Cmp EBP, 0xa →Sta dicendo di confrontare il valore 10 con il valore presente nel registro EBP

Jge 0x1176 →In base al risultato della comparazione, se è positivo il valore della comparazione salta alla cella di memoria (0x1176), se il risultato è negativo il programma salta questa istruzione e va avanti con le prossime istruzioni.

Mov EAX, 0x0 →Sposta il valore 0 al registro di memoria EAX

Call 0x1030 → Sta dicendo di chiamare l'istruzione che è presente alla cella di memoria che corrisponde al valore (0x1030)

Nozioni sui registri utilizzati dal programma

EAX

EAX, abbreviazione di "Extended Accumulator", è un registro di dati a 32 bit utilizzato nei processori x86. È uno dei registri general-purpose, il che significa che può essere utilizzato per una varietà di scopi, come memorizzare dati temporanei, operazioni aritmetiche e logiche, o indirizzi di memoria.

Nel contesto dell'assembly x86, EAX viene spesso utilizzato come registro accumulator, cioè viene spesso coinvolto nelle operazioni aritmetiche, logiche e di trasferimento dei dati. Ad esempio, molte istruzioni aritmetiche e logiche prendono uno degli operandi da EAX o scrivono il risultato in EAX.

In alcuni contesti specifici, EAX può avere un ruolo dedicato. Ad esempio, in alcune chiamate di sistema o funzioni di libreria, EAX può essere utilizzato per restituire valori di ritorno, come un codice di errore o il risultato di un'operazione.

EDX

EDX, abbreviazione di "Extended Data", è un altro registro a 32 bit nei processori x86 che svolge una funzione simile a EAX. Come EAX, anche EDX è un registro general-purpose che può essere utilizzato per una varietà di scopi, tra cui memorizzazione di dati temporanei, operazioni aritmetiche e logiche, o indirizzi di memoria.

Nel contesto dell'assembly x86, EDX viene spesso utilizzato insieme a EAX in operazioni che coinvolgono dati di dimensioni superiori a 32 bit, come moltiplicazioni e divisioni, in cui il risultato può essere più grande di quanto possa contenere un registro a 32 bit. In questi casi, EAX può contenere i bit più significativi del risultato e EDX i bit meno significativi.

Inoltre, come per EAX, in alcune chiamate di sistema o funzioni di libreria, EDX può essere utilizzato per restituire valori di ritorno o per passare parametri aggiuntivi, a seconda della convenzione di chiamata utilizzata.

EBP

EBP, abbreviazione di "Extended Base Pointer", è un altro registro utilizzato nei processori x86. Tuttavia, a differenza di EAX ed EDX che sono registri general-purpose, EBP ha un'utilità specifica nella gestione della memoria durante l'esecuzione di subroutine o funzioni.

Il registro EBP viene spesso utilizzato come "base pointer" per accedere alle variabili locali e ai parametri delle funzioni all'interno dello stack di chiamate. Durante l'esecuzione di una funzione, il valore di EBP viene solitamente salvato nello stack e utilizzato come riferimento per accedere ai dati locali della funzione stessa.

Quando una funzione viene chiamata, il suo "stack frame" viene creato, e il valore di EBP viene spesso utilizzato per indicare l'inizio di questo frame nello stack. Ciò consente alla funzione di

accedere in modo efficiente ai suoi parametri e variabili locali, indipendentemente dalla loro posizione nello stack.

Inoltre, EBP può essere utilizzato anche per l'accesso a variabili locali di funzioni annidate o per stabilire un "frame pointer chain" che consente di risalire la catena delle chiamate delle funzioni nello stack.