Codice C relazione con Assembly

martedì 14 marzo 2023 22:27

Codice C verso codice macchina

Ogni funzione del codice C, quando viene compilato, diventa una routine macchina.

compile with “gcc –c –fomit-frame-pointer”
inspect with “objdump”

```
int f(int x){
        if (x == 1) return 1;
        return 0;
}
```

I valori di ritorno tipicamente sono consegnati all’interno dei registri, se ovviamente il registro lo può ospitare. Altrimenti si usa la memoria.

```
0000000000000000 <f>:
   0:   89 7c 24 fc           mov    %edi,-0x4(%rsp)
   4:   83 7c 24 fc 01        cmpl   $0x1,-0x4(%rsp)
   9:   75 07                 jne    12 <f+0x12>
   b:   b8 01 00 00 00        mov    $0x1,%eax
  10:   eb 05                 jmp    17 <f+0x17>
  12:   b8 00 00 00 00        mov    $0x0,%eax
  17:   c3                    retq
```

AT&T syntax

Nella destinazione noi stiamo usando un altro registro, che è RSP.
Su x86 RSP è uno “Stack Pointer”.
E se noi utilizziamo un registro della CPU all’interno delle parentesi, stiamo utilizzando il valore di quel registro come “pointer” verso la memoria.

Il registro di processore che stiamo usando in x86 è “edi”, e quello che stiamo facendo nella mov è prendere i dati da questa sorgente e scriverli in una destinazione.

In quel caso bisogna saltare nel punto di memoria targato come 12, e saltiamo esattamente a quest’altra istruzione di mov.

Ora in base al risultato della comparazione si fanno cose diverse. “Jump not Equal” (JNE) ci dice che, l’ultima istruzione di compare che abbiamo eseguito ha lasciato traccia all’interno della CPU sui bit di stato, che non c’è un’uguaglianza.

Lo stack pointer ci sta dicendo “dove” in memoria noi dobbiamo andare a scaricare l’informazione che è *presente all’interno del registro* %edi.

Attenzione: non stiamo scaricando l’informazione dove sta puntando “esattamente” lo Stack Pointer perché stiamo applicando un “offset” di 0x4 negativo esadecimale.

Ora andiamo a testare if(x==1).
Il valore di x lo abbiamo portato esattamente nell’area di stack segnata, e ora chiamiamo un’istruzione di compare (cmp), che sta cercando di *comparare il valore 1 con ciò che è presente all’interno della memoria segnata*.

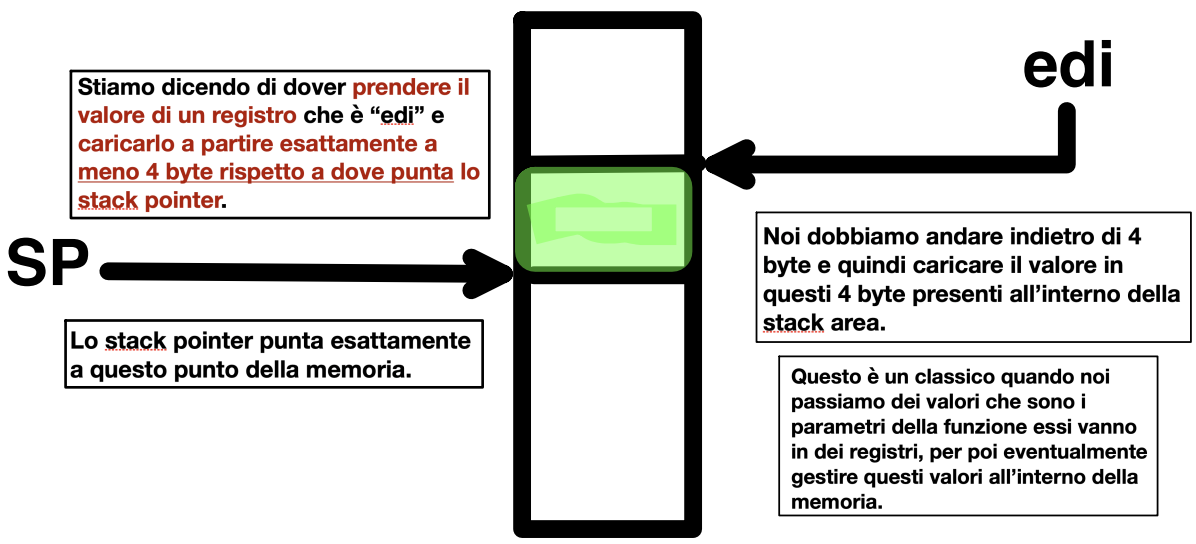
Nella 12 eseguiamo una mov del valore zero nel registro eax.

Infine con retq ritorniamo il controllo al chiamante. Pulito

I parametri che noi passiamo, tipicamente all’interno del codice macchina che viene generato, si vede che questi parametri li vado a prendere all’interno dei registri di processore.

Quindi quando noi andiamo a chiamare una funzione C e generiamo poi la compilazione di ciò che stiamo facendo, il chiamante scrive queste informazioni in registri di processore.

I PARAMETRI A FUNZIONE TIPICAMENTE IL CODICE MACCHINA CHE VIENE GENERATO LI VA A PRENDERE DIRETTAMENTE DAI REGISTRI DI PROCESSORE.



Stiamo dicendo di dover prendere il valore di un registro che è “edi” e caricarlo a partire esattamente a meno 4 byte rispetto a dove punta lo stack pointer.

Noi dobbiamo andare indietro di 4 byte e quindi caricare il valore in questi 4 byte presenti all’interno della stack area.

Lo stack pointer punta esattamente a questo punto della memoria.

Questo è un classico quando noi passiamo dei valori che sono i parametri della funzione essi vanno in dei registri, per poi eventualmente gestire questi valori all’interno della memoria.