Interfacciamento tra linguaggio C e linguaggio Assembly

Fabrizio Angiulli

Abstract

Il presente documento fornisce una breve descrizione delle convenzioni di chiamata delle funzioni in linguaggio C (*C Calling Conventions*) utilizzate dal compilatore GNU, sia con riferimento all'architettura x86-32 che all'architettura x86-64. La conoscenza di tali convenzioni consente di interfacciare programmi scritti in linguaggio C con programmi scritti in linguaggio Assembly, mediante l'invocazione di funzioni Assembly da funzioni C e viceversa.

1 C Calling Conventions a 32 bit

I programmi in linguaggio C su architettura x86-32 utilizzano le seguenti convenzioni per il passaggio dei parametri tra funzioni:

- I parametri passati alle funzioni vengono inseriti (*push*) nello stack in ordine inverso. Quindi, data la chiamata di funzione `func(a, b, c)`, il valore del parametro `c` viene posto nello stack per primo, quello di `b` per secondo e infine quello di `a` per terzo;
- La funzione chiamante assume che il contenuto dei registri ESP, EBP, EBX, ESI e EDI non venga alterato dalla funzione chiamata. Questo non significa che questi registri non possono essere utilizzati dalla funzione chiamata, ma piuttosto che il loro valore al termine dell'esecuzione della funzione chiamata deve coincidere con il valore all'inzio dell'esecuzione. Tutti gli altri registri possono essere modificati senza nessun accorgimento dalla funzione chiamata;
- La funzione restituisce il valore di ritorno nel registro EAX se la dimensione del tipo di ritorno è minore o uguale a 32 bit. I puntatori (a qualsiasi tipo) sono da assimilarsi ad interi a 32 bit. Nel caso il valore di ritorno sia a 64 bit, viene restituito nella coppia di registri EAX (i 32 bit meno significativi) e EDX (i 32 bit più significativi). I valori in virgola mobile (ovvero `float` oppure `double`) vengono posti in cima allo stack dei registri x87 (che non abbiamo trattato). Le strutture ed ogni altro tipo avente dimensione maggiore di 32 bit devono essere restituite per riferimento, ovvero la funzione ne restituirà l'indirizzo di partenza nel registro EAX;
- La funzione chiamata non rimuove i parametri dallo stack. La rimozione dei parametri dallo stack è a carico della funzione chiamante e può essere effettuata sommando allo stack pointer ESP la dimensione dei parametri passati.

Ad esempio, si consideri il seguente frammento di codice C, in cui il `main` chiama una funzione esterna di nome `asmfunc` scritta in linguaggio assembly x86-32:

```
extern int asmfunc(float* A, int m, int n);

int main() {
    int m = 10, n = 2;
    float* A = calloc(m*n,sizeof(float));
    int y = asmfunc(A,m,n);
    ...
}
```

Di seguito si riporta la struttura generale della funzione `asmfunc`:

```
global asmfunc       ; rende la funzione visibile all'esterno

; Posizione dei parametri nel Recordi di Attivazione della funzione
; (i primi 8 bytes sono occupati dall'indirizzo di ritorno e da EBP)
;
A          equ       8                    ; puntatore a float, occupa 32 bit (4 bytes)
m          equ       12                   ; intero a 32 bit
n          equ       16                   ; intero a 32 bit

asmfunc:
           ;
           ; sequenza di ingresso nella funzione
           ;
           push      ebp                  ; salva il Base Pointer
           mov       ebp, esp             ; il Base Pointer punta al Record di Attivazione corrente
           push      ebx                  ; salva i registri da preservare
           push      esi
           push      edi
           ;
           ; lettura dei parametri dal Recordi di Attivazione
           ;
           mov       ..., [ebp+A]         ; legge A
           mov       ..., [ebp+m]         ; legge m
           mov       ..., [ebp+n]         ; legge n
           ;
           ; corpo della funzione
           ;
           ...
           mov       eax, <valore-di-ritorno>; asmfunc restituisce un numero intero
           ;
           ; sequenza di uscita dalla funzione
           ;
           pop       edi                  ; ripristina i registri da preservare
           pop       esi
           pop       ebx
           mov       esp, ebp             ; ripristina lo Stack Pointer
           pop       ebp                  ; ripristina il Base Pointer
           ret                            ; ritorna alla funzione chiamante
```

Per completezza, di seguito si mostra come il compilatore `gcc` traduce in assembly la chiamata della funzione `asmfunc`:

```
main:
         ...
         push     <valore-di-n>       ; posiziona i parametri nello stack
         push     <valore-di-m>
         push     <indirizzo-di-A>
         call     asmfunc             ; chiama la funzione asmfunc
         add      esp, 12             ; rimuove i parametri dallo stack
         ...
```

2 C Calling Conventions a 64 bit

I programmi in linguaggio C su architettura x86-64 utilizzano le seguenti convenzioni per il passaggio dei parametri tra funzioni:

- I primi sei parametri interi (scorrendo l'elenco dei parametri da sinistra verso destra) vengono passati, rispettivamente, nei registri RDI, RSI, RDX, RCX, R8 ed R9. Ulteriori parametri interi vengono passati sullo stack. I registri di cui sopra, insieme ai registri RAX, R10 ed R11 possono essere modificati dalla funzione chiamata senza la necessità di ripristinarli ai loro valori originari;
- I valori di ritorno interi vengono restituiti nei registri RAX e RDX;
- Gli argomenti floating-point vengono passati nei registri da XMM0 a XMM7. I valori di ritorno floating point vengono restituiti nei registri XMM0 ed XMM1;
- Tutti i registri SSE ed x87 possono essere alterati dalla funzione chiamata.

Ad esempio, nel caso della chiamata della funzione

`void func(long a, double b, int c)`,

il parametro `a` viene passato in RDI, il parametro `b` viene passato in XMM0, mentre il parametro `c` viene passato in RSI.