Esempio utilizzo dello Stack

```
r0x@ubuntu:~/lezioni/sicII/slides/code$ cat stack_ex1.c


int function_B(int a, int b)
{
        int x, y;
        x = a * a;
        y = b * b;
        return (x + y);
}

int function_A(int p, int q)
{
        int c;
        c = function_B(p,q);
        return c;
}

int main(int argc, char** argv, char** envp)
{
        int ret;
        ret = function_A(1,2);
        return ret;
}
```

Installate https://github.com/longld/peda

breakpoint on 0x08048424. digitare ‘run’ per iniziare l’esecuzione

```
(gdb) set disassembly-flavor intel
(gdb) disass function_A
Dump of assembler code for function function_A:
   0x08048411 <+0>:     push   ebp
   0x08048412 <+1>:     mov    ebp,esp
   0x08048414 <+3>:     sub    esp,0x18
   0x08048417 <+6>:     mov    eax,DWORD PTR [ebp+0xc]
   0x0804841a <+9>:     mov    DWORD PTR [esp+0x4],eax
   0x0804841e <+13>:    mov    eax,DWORD PTR [ebp+0x8]
   0x08048421 <+16>:    mov    DWORD PTR [esp],eax
   0x08048424 <+19>:    call   0x80483ed <function_B>
   0x08048429 <+24>:    mov    DWORD PTR [ebp-0x4],eax
   0x0804842c <+27>:    mov    eax,DWORD PTR [ebp-0x4]
   0x0804842f <+30>:    leave
   0x08048430 <+31>:    ret
End of assembler dump.
(gdb) b *0x08048424
Breakpoint 1 at 0x8048424: file stack_ex1.c, line 14.
(gdb) info b
Num     Type           Disp Enb Address    What
1       breakpoint     keep y   0x08048424 in function_A at stack_ex1.c:14
```

gli argomenti di function_B sono stati caricati nello stack

```
[----------------------------------code----------------------------------]
   0x804841a <function_A+9>:    mov    DWORD PTR [esp+0x4],eax
   0x804841e <function_A+13>:   mov    eax,DWORD PTR [ebp+0x8]
   0x8048421 <function_A+16>:   mov    DWORD PTR [esp],eax
=> 0x8048424 <function_A+19>:   call   0x80483ed <function_B>
   0x8048429 <function_A+24>:   mov    DWORD PTR [ebp-0x4],eax
   0x804842c <function_A+27>:   mov    eax,DWORD PTR [ebp-0x4]
   0x804842f <function_A+30>:   leave
   0x8048430 <function_A+31>:   ret
Guessed arguments:
arg[0]: 0x1
arg[1]: 0x2
[----------------------------------stack-----------------------------------]
0000| 0xffffd040 --> 0x1
0004| 0xffffd044 --> 0x2
0008| 0xffffd048 --> 0x804a000 --> 0x8049f14 --> 0x1
0012| 0xffffd04c --> 0x80484b2 (<__libc_csu_init+82>:   add    edi,0x1)
0016| 0xffffd050 --> 0x1
0020| 0xffffd054 --> 0xffffd114 --> 0xffffd2d0 ("/home/r0x/lezioni/sicII/slides/code/stack_ex1")
0024| 0xffffd058 --> 0xffffd078 --> 0x0
0028| 0xffffd05c --> 0x804844b (<main+26>:      mov    DWORD PTR [ebp-0x4],eax)
[--------------------------------------------------------------------------]
Legend: code, data, rodata, value
```

prima istruzione di function_B

L'istruzione *call* precedente ha caricato sullo stack l'indirizzo di ritorno

```
[----------------------------------code----------------------------------]
   0x80483e2 <frame_dummy+34>:  jmp    0x8048360 <register_tm_clones>
   0x80483e7 <frame_dummy+39>:  nop
   0x80483e8 <frame_dummy+40>:  jmp    0x8048360 <register_tm_clones>
=> 0x80483ed <function_B>:      push   ebp
   0x80483ee <function_B+1>:    mov    ebp,esp
   0x80483f0 <function_B+3>:    sub    esp,0x10
   0x80483f3 <function_B+6>:    mov    eax,DWORD PTR [ebp+0x8]
   0x80483f6 <function_B+9>:    imul   eax,DWORD PTR [ebp+0x8]
[----------------------------------stack-----------------------------------]
0000| 0xffffd03c --> 0x8048429 (<function_A+24>:       mov    DWORD PTR [ebp-0x4],eax)
0004| 0xffffd040 --> 0x1
0008| 0xffffd044 --> 0x2
```

set breakpoint sull'ultima istruzione di function_B

```
gdb-peda$ disass
Dump of assembler code for function function_B:
=> 0x080483ed <+0>:     push   ebp
   0x080483ee <+1>:     mov    ebp,esp
   0x080483f0 <+3>:     sub    esp,0x10
   0x080483f3 <+6>:     mov    eax,DWORD PTR [ebp+0x8]
   0x080483f6 <+9>:     imul   eax,DWORD PTR [ebp+0x8]
   0x080483fa <+13>:    mov    DWORD PTR [ebp-0x4],eax
   0x080483fd <+16>:    mov    eax,DWORD PTR [ebp+0xc]
   0x08048400 <+19>:    imul   eax,DWORD PTR [ebp+0xc]
   0x08048404 <+23>:    mov    DWORD PTR [ebp-0x8],eax
   0x08048407 <+26>:    mov    eax,DWORD PTR [ebp-0x8]
   0x0804840a <+29>:    mov    edx,DWORD PTR [ebp-0x4]
   0x0804840d <+32>:    add    eax,edx
   0x0804840f <+34>:    leave
   0x08048410 <+35>:    ret
End of assembler dump.
gdb-peda$ b *0x08048410
Breakpoint 3 at 0x8048410: file stack_ex1.c, line 9.
```

ultima istruzione di function_B

l'istruzione *ret* carica nel registro *EIP* il valore presente in cima allo stack.

L'esecuzione continuerà da tale indirizzo.

```
[----------------------------------code----------------------------------]
   0x804840a <function_B+29>:   mov    edx,DWORD PTR [ebp-0x4]
   0x804840d <function_B+32>:   add    eax,edx
   0x804840f <function_B+34>:   leave
=> 0x8048410 <function_B+35>:   ret
   0x8048411 <function_A>:      push   ebp
   0x8048412 <function_A+1>:    mov    ebp,esp
   0x8048414 <function_A+3>:    sub    esp,0x18
   0x8048417 <function_A+6>:    mov    eax,DWORD PTR [ebp+0xc]
[----------------------------------stack-----------------------------------]
0000| 0xffffd03c --> 0x8048429 (<function_A+24>:        mov    DWORD PTR [ebp-0x4],eax)
0004| 0xffffd040 --> 0x1
0008| 0xffffd044 --> 0x2
0012| 0xffffd048 --> 0x804a000 --> 0x8049f14 --> 0x1
0016| 0xffffd04c --> 0x80484b2 (<__libc_csu_init+82>:   add    edi,0x1)
0020| 0xffffd050 --> 0x1
0024| 0xffffd054 --> 0xffffd114 --> 0xffffd2d0 ("/home/r0x/lezioni/sicII/slides/code/stack_ex1")
0028| 0xffffd058 --> 0xffffd078 --> 0x0
[--------------------------------------------------------------------------]
Legend: code, data, rodata, value
```

Prima istruzione dopo la chiamata di function_B

salva (*mov*) il valore contenuto in *EAX* nella variabile locale di function_A, puntata da ebp-0x4

```
   0x804841e <function_A+13>:   mov    eax,DWORD PTR [ebp+0x8]
   0x8048421 <function_A+16>:   mov    DWORD PTR [esp],eax
   0x8048424 <function_A+19>:   call   0x80483ed <function_B>
=> 0x8048429 <function_A+24>:   mov    DWORD PTR [ebp-0x4],eax
   0x804842c <function_A+27>:   mov    eax,DWORD PTR [ebp-0x4]
   0x804842f <function_A+30>:   leave
   0x8048430 <function_A+31>:   ret
   0x8048431 <main>:    push   ebp
[------------------------------------stack-------------------------------------]
0000| 0xffffd040 --> 0x1
0004| 0xffffd044 --> 0x2
0008| 0xffffd048 --> 0x804a000 --> 0x8049f14 --> 0x1
0012| 0xffffd04c --> 0x80484b2 (<__libc_csu_init+82>:   add    edi,0x1)
0016| 0xffffd050 --> 0x1
0020| 0xffffd054 --> 0xffffd114 --> 0xffffd2d0 ("/home/r0x/lezioni/sicII/slides/code/stack_ex1")
0024| 0xffffd058 --> 0xffffd078 --> 0x0
0028| 0xffffd05c --> 0x804844b (<main+26>:      mov    DWORD PTR [ebp-0x4],eax)
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 2, 0x08048429 in function_A (p=0x1, q=0x2) at stack_ex1.c:14
14              c = function_B(p,q);
gdb-peda$ p/x $eax
$3 = 0x5
```

Tale valore viene salvato nel registro *EAX* in quanto è il valore di ritorno della funzione

```
[-------------------------------------code-------------------------------------]
   0x8048421 <function_A+16>:   mov    DWORD PTR [esp],eax
   0x8048424 <function_A+19>:   call   0x80483ed <function_B>
   0x8048429 <function_A+24>:   mov    DWORD PTR [ebp-0x4],eax
=> 0x804842c <function_A+27>:   mov    eax,DWORD PTR [ebp-0x4]
   0x804842f <function_A+30>:   leave
   0x8048430 <function_A+31>:   ret
   0x8048431 <main>:    push   ebp
   0x8048432 <main+1>:  mov    ebp,esp
[------------------------------------stack-------------------------------------]
0000| 0xffffd040 --> 0x1
0004| 0xffffd044 --> 0x2
0008| 0xffffd048 --> 0x804a000 --> 0x8049f14 --> 0x1
0012| 0xffffd04c --> 0x80484b2 (<__libc_csu_init+82>:   add    edi,0x1)
0016| 0xffffd050 --> 0x1
0020| 0xffffd054 --> 0x5
0024| 0xffffd058 --> 0xffffd078 --> 0x0
0028| 0xffffd05c --> 0x804844b (<main+26>:      mov    DWORD PTR [ebp-0x4],eax)
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
15              return c;
```

La comprensione delle immagini a lato è lasciata al lettore come esercizio

```
[----------------------------------code-----------------------------------]
   0x8048424 <function_A+19>:   call   0x80483ed <function_B>
   0x8048429 <function_A+24>:   mov    DWORD PTR [ebp-0x4],eax
   0x804842c <function_A+27>:   mov    eax,DWORD PTR [ebp-0x4]
=> 0x804842f <function_A+30>:   leave
   0x8048430 <function_A+31>:   ret
   0x8048431 <main>:    push   ebp
   0x8048432 <main+1>:  mov    ebp,esp
   0x8048434 <main+3>:  sub    esp,0x18
[----------------------------------stack-----------------------------------]
0000| 0xffffd040 --> 0x1
0004| 0xffffd044 --> 0x2
0008| 0xffffd048 --> 0x804a000 --> 0x8049f14 --> 0x1
0012| 0xffffd04c --> 0x80484b2 (<__libc_csu_init+82>:   add    edi,0x1)
0016| 0xffffd050 --> 0x1
0020| 0xffffd054 --> 0x5
0024| 0xffffd058 --> 0xffffd078 --> 0x0
0028| 0xffffd05c --> 0x804844b (<main+26>:      mov    DWORD PTR [ebp-0x4],eax)
[--------------------------------------------------------------------------]
Legend: code, data, rodata, value
16      }
gdb-peda$ p/x $eax
$4 = 0x5
```

```
   0x8048429 <function_A+24>:   mov    DWORD PTR [ebp-0x4],eax
   0x804842c <function_A+27>:   mov    eax,DWORD PTR [ebp-0x4]
   0x804842f <function_A+30>:   leave
=> 0x8048430 <function_A+31>:   ret
   0x8048431 <main>:    push   ebp
   0x8048432 <main+1>:  mov    ebp,esp
   0x8048434 <main+3>:  sub    esp,0x18
   0x8048437 <main+6>:  mov    DWORD PTR [esp+0x4],0x2
[----------------------------------stack-----------------------------------]
0000| 0xffffd05c --> 0x804844b (<main+26>:      mov    DWORD PTR [ebp-0x4],eax)
0004| 0xffffd060 --> 0x1
0008| 0xffffd064 --> 0x2
0012| 0xffffd068 --> 0x804846b (<__libc_csu_init+11>:   add    ebx,0x1b95)
0016| 0xffffd06c --> 0xf7fbb000 --> 0x1a9da8
0020| 0xffffd070 --> 0x8048460 (<__libc_csu_init>:      push   ebp)
0024| 0xffffd074 --> 0x0
0028| 0xffffd078 --> 0x0
[--------------------------------------------------------------------------]
Legend: code, data, rodata, value
0x08048430      16      }
```