

Exemple stack.c

désassemblage par objdump -s -j stack

```
int add(int a, int b){
51d: 55                push    %ebp
51e: 89 e5             mov     %esp,%ebp
520: 83 ec 10          sub     $0x10,%esp
523: e8 78 00 00 00    call   5a0 <__x86.get_pc_thunk.ax>
528: 05 b0 1a 00 00    add     $0x1ab0,%eax
        int c = a+b;
52d: 8b 55 08          mov     0x8(%ebp),%edx
530: 8b 45 0c          mov     0xc(%ebp),%eax
533: 01 d0             add     %edx,%eax
535: 89 45 fc          mov     %eax,-0x4(%ebp)
        return c;
538: 8b 45 fc          mov     -0x4(%ebp),%eax
}
53b: c9                leave   %eax
53c: c3                ret
```

0000053d <main>:

```
int main(int32_t argc, char** argv){
53d: 8d 4c 24 04       lea     0x4(%esp),%ecx
541: 83 e4 f0          and     $0xfffffffff0,%esp
544: ff 71 fc          pushl   -0x4(%ecx)
547: 55                push    %ebp
548: 89 e5             mov     %esp,%ebp
54a: 53                push    %ebx
54b: 51                push    %ecx
54c: 83 ec 10          sub     $0x10,%esp
54f: e8 cc fe ff ff    call   420 <__x86.get_pc_thunk.bx>
554: 81 c3 84 1a 00 00 add     $0x1a84,%ebx
        int i = 1;
55a: c7 45 ec 01 00 00 00 movl    $0x1,-0x14(%ebp)
        int j = 3;
561: c7 45 f0 03 00 00 00 movl    $0x3,-0x10(%ebp)
        int t = add(i,j);
568: ff 75 f0          pushl   -0x10(%ebp)
56b: ff 75 ec          pushl   -0x14(%ebp)
56e: e8 aa ff ff ff    call   51d <_Z3addii>
573: 83 c4 08          add     $0x8,%esp
576: 89 45 f4          mov     %eax,-0xc(%ebp)
        printf("%d + %d = %d\n",i,j,t);
579: ff 75 f4          pushl   -0xc(%ebp)
57c: ff 75 f0          pushl   -0x10(%ebp)
57f: ff 75 ec          pushl   -0x14(%ebp)
582: 8d 83 58 e6 ff ff lea     -0x19a8(%ebx),%eax
588: 50                push    %eax
589: e8 22 fe ff ff    call   3b0 <printf@plt>
58e: 83 c4 10          add     $0x10,%esp
591: b8 00 00 00 00    mov     $0x0,%eax
596: 8d 65 f8          lea     -0x8(%ebp),%esp
599: 59                pop     %ecx
59a: 5b                pop     %ebx
59b: 5d                pop     %ebp
59c: 8d 61 fc          lea     -0x4(%ecx),%esp
59f: c3                ret
```

État de la pile juste avant le `call <add(int,int)>`

```
0x40054b <main(int32_t, char**)+14>    push    %ecx
0x40054c <main(int32_t, char**)+15>    sub     $0x10,%esp
0x40054f <main(int32_t, char**)+18>    call   0x400420 <_x86.get_pc_thunk.bx>
0x400554 <main(int32_t, char**)+23>    add     $0x1a84,%ebx
0x40055a <main(int32_t, char**)+29>    movl    $0x1,-0x14(%ebp)
0x400561 <main(int32_t, char**)+36>    movl    $0x3,-0x10(%ebp)
0x400568 <main(int32_t, char**)+43>    pushl   -0x10(%ebp)
0x40056b <main(int32_t, char**)+46>    pushl   -0x14(%ebp)
> 0x40056e <main(int32_t, char**)+49>    call    0x40051d <add(int, int)>
0x400573 <main(int32_t, char**)+54>    add     $0x8,%esp
0x400576 <main(int32_t, char**)+57>    mov     %eax,-0xc(%ebp)
0x400579 <main(int32_t, char**)+60>    pushl   -0xc(%ebp)
0x40057c <main(int32_t, char**)+63>    pushl   -0x10(%ebp)
0x40057f <main(int32_t, char**)+66>    pushl   -0x14(%ebp)

native process 10309 In: main
(gdb) x/10wx $esp
0xbffff078:  0x00000001  0x00000003  0x00000001  0x00000001
0xbffff088:  0x00000003  0x004005d1  0xbffff0b0  0x00000000
0xbffff098:  0x00000000  0xb7df6f21
(gdb) print $ebp
$1 = (void *) 0xbffff098
(gdb) print *$esp
Attempt to take contents of a non-pointer value.
(gdb) x/lwx $esp
0xbffff078:  0x00000001
(gdb) x/lwx $ebp
0xbffff098:  0x00000000
(gdb) x/8wx $esp
0xbffff078:  0x00000001  0x00000003  0x00000001  0x00000001
0xbffff088:  0x00000003  0x004005d1  0xbffff0b0  0x00000000
(gdb) |
```

État de la pile dans `call <add(int,int)>`

```
> 0x40051d <add(int, int)>            push    %ebp
0x40051e <add(int, int)+1>            mov     %esp,%ebp
0x400520 <add(int, int)+3>            sub     $0x10,%esp
0x400523 <add(int, int)+6>            call   0x4005a0 <_x86.get_pc_thunk.ax>
0x400528 <add(int, int)+11>           add     $0x1ab0,%eax
0x40052d <add(int, int)+16>           mov     0x8(%ebp),%edx
0x400530 <add(int, int)+19>           mov     0xc(%ebp),%eax
0x400533 <add(int, int)+22>           add     %edx,%eax
0x400535 <add(int, int)+24>           mov     %eax,-0x4(%ebp)
0x400538 <add(int, int)+27>           mov     -0x4(%ebp),%eax
0x40053b <add(int, int)+30>           leave
0x40053c <add(int, int)+31>           ret
B+ 0x40053d <main(int32_t, char**)>    lea     0x4(%esp),%ecx
0x400541 <main(int32_t, char**)+4>    and     $0xffffffff0,%esp

native process 10309 In: add
(gdb) x/lwx $esp
0xbffff078:  0x00000001
(gdb) x/lwx $ebp
0xbffff098:  0x00000000
(gdb) x/8wx $esp
0xbffff078:  0x00000001  0x00000003  0x00000001  0x00000001
0xbffff088:  0x00000003  0x004005d1  0xbffff0b0  0x00000000
(gdb) si
add (a=1, b=3) at tests.c:6
1: $esp = (void *) 0xbffff074
2: $ebp = (void *) 0xbffff098
(gdb) x/11wx $esp
0xbffff074:  0x00400573  0x00000001  0x00000003  0x00000001
0xbffff084:  0x00000001  0x00000003  0x004005d1  0xbffff0b0
0xbffff094:  0x00000000  0x00000000  0xb7df6f21
(gdb) |
```

État de la pile juste avant la fin du `<add(int,int)>`

```
0x40051d <add(int, int)>      push    %ebp
0x40051d <add(int, int)>      push    %ebp
0x40051e <add(int, int)+1>     mov     %esp,%ebp
0x400520 <add(int, int)+3>     sub     $0x10,%esp
0x400523 <add(int, int)+6>     call    0x4005a0 <__x86.get_pc_thunk.ax>
0x400528 <add(int, int)+11>    add     $0x1ab0,%eax
0x40052d <add(int, int)+16>    mov     0x8(%ebp),%edx
0x400530 <add(int, int)+19>    mov     0xc(%ebp),%eax
0x400533 <add(int, int)+22>    add     %edx,%eax
0x400535 <add(int, int)+24>    mov     %eax,-0x4(%ebp)
0x400538 <add(int, int)+27>    mov     -0x4(%ebp),%eax
> 0x40053b <add(int, int)+30>    leave
0x40053c <add(int, int)+31>    ret
B+ 0x40053d <main(int32_t, char**)>    lea     0x4(%esp),%ecxsp
    0x400541 <main(int32_t, char**)+4>    and     $0xffffffff,%esp
native process 10309 In: add
(gdb) xprocess 11124 In: add
2: $ebp = (void *) 0xbffff070
(gdb) ni
1: $esp = (void *) 0xbffff060
2: $ebp = (void *) 0xbffff070
(gdb) ni
1: $esp = (void *) 0xbffff060
2: $ebp = (void *) 0xbffff070
(gdb) ni
1: $esp = (void *) 0xbffff060
2: $ebp = (void *) 0xbffff070
(gdb) x/14wx $esp
0xbffff060:  0xb7fb6000  0xb7fb6000  0x00000000  0x00000004
0xbffff070:  0xbffff098  0x00400573  0x00000001  0x00000003
0xbffff080:  0x00000001  0x00000001  0x00000003  0x004005d1
0xbffff090:  0xbffff0b0  0x00000000
(gdb)
```