x86 stack memory

```
#include <stdlib.h>
#include <iostream>

void switch_(int *num1, int *num2)
{
        *num1 = *num1 ^ *num2;
        *num2 = *num1 ^ *num2;
        *num1 = *num1 ^ *num2;
}

int main()
{
        int *num1 = (int*) malloc(sizeof(int)),
                *num2 = (int*) malloc(sizeof(int));
        *num1 = 8;
        *num2 = 16;

        std::cout << "num1 addr: " << num1 << " ";
        std::cout << "num2 addr: " << num2 << std::endl;
        std::cout << "num1: " << *num1 << " ";
        std::cout << "num2: " << *num2 << std::endl;

        switch_(num1, num2);

        std::cout << "num1: " << *num1 << " ";
        std::cout << "num2: " << *num2 << std::endl;

        return 0;
}
```

1. When breaking at switch_(num1, num2), the address of num1 is 0x804b008 and the address of num2 is 0x804b018.

```
(gdb) r
Starting program: /home/txuriurdin22/Desktop/a.out
num1 addr: 0x804b008 num2 addr: 0x804b018
num1: 8 num2: 16

Breakpoint 1, 0x08048750 in switch_(int*, int*) ()
```

2. ESP(stack pointer) is at 0xb0ffff0b8 and the stack memory shows in *Fig. 1*.

```
(gdb) info register esp
esp            0xbffff0b8       0xbffff0b8
(gdb) x/4xw 0xbffff0b8
0xbffff0b8:     0xbffff0e8      0x080488ab      0x0804b008      0x0804b018
```

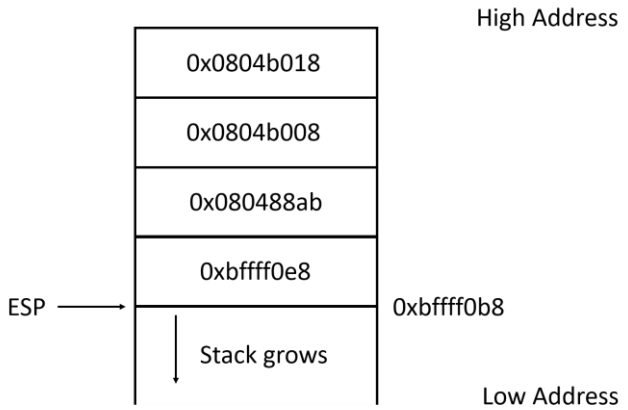


Fig. 1

3. In assembly, the program stops at 0x08048750, before this instruction, the program runs “push %esp”. When program just jump into switch_(before “push %esp” executed), the stack memory shows in *Fig. 2*.

```
0x804874d <_Z7switch_PiS_>      push   %ebp
0x804874e <_Z7switch_PiS_+1>    mov    %esp,%ebp
0x8048750 <_Z7switch_PiS_+3>    mov    0x8(%ebp),%eax
0x8048753 <_Z7switch_PiS_+6>    mov    (%eax),%edx
0x8048755 <_Z7switch_PiS_+8>    mov    0xc(%ebp),%eax
0x8048758 <_Z7switch_PiS_+11>   mov    (%eax),%eax
0x804875a <_Z7switch_PiS_+13>   xor    %eax,%edx
0x804875c <_Z7switch_PiS_+15>   mov    0x8(%ebp),%eax
0x804875f <_Z7switch_PiS_+18>   mov    %edx,(%eax)
0x8048761 <_Z7switch_PiS_+20>   mov    0x8(%ebp),%eax
0x8048764 <_Z7switch_PiS_+23>   mov    (%eax),%edx
0x8048766 <_Z7switch_PiS_+25>   mov    0xc(%ebp),%eax
0x8048769 <_Z7switch_PiS_+28>   mov    (%eax),%eax
0x804876b <_Z7switch_PiS_+30>   xor    %eax,%edx
0x804876d <_Z7switch_PiS_+32>   mov    0xc(%ebp),%eax
0x8048770 <_Z7switch_PiS_+35>   mov    %edx,(%eax)
0x8048772 <_Z7switch_PiS_+37>   mov    0x8(%ebp),%eax
0x8048775 <_Z7switch_PiS_+40>   mov    (%eax),%edx
0x8048777 <_Z7switch_PiS_+42>   mov    0xc(%ebp),%eax
0x804877a <_Z7switch_PiS_+45>   mov    (%eax),%eax
0x804877c <_Z7switch_PiS_+47>   xor    %eax,%edx
0x804877e <_Z7switch_PiS_+49>   mov    0x8(%ebp),%eax
0x8048781 <_Z7switch_PiS_+52>   mov    %edx,(%eax)
0x8048783 <_Z7switch_PiS_+54>   pop    %ebp
0x8048784 <_Z7switch_PiS_+55>   ret
```

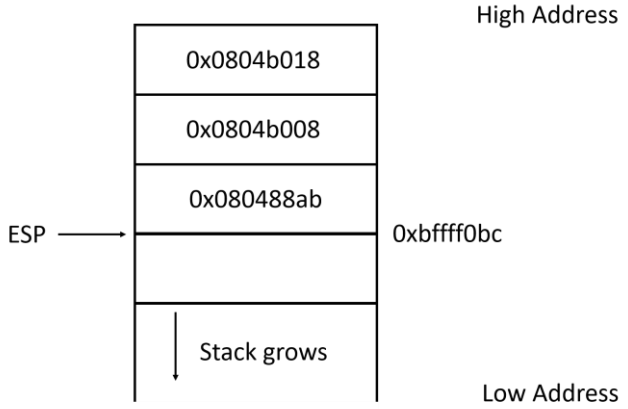
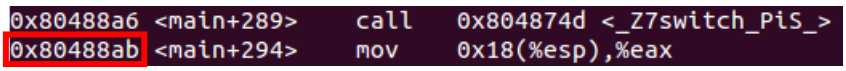


Fig. 2

4. 0x804b008 is the address of num1 and 0x804b018 is the address of num2.
5. 0x08044ab is the return address.



6. Before switch_ ends, the program pops %ebp out, so the stack memory shows in *Fig. 2* and the return address is where ESP points at.