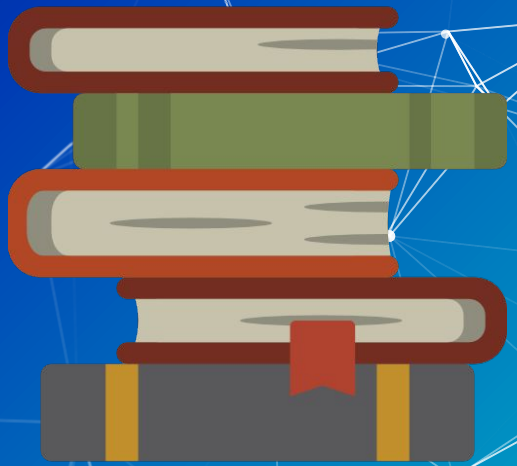
Stack Call

In x86 arch

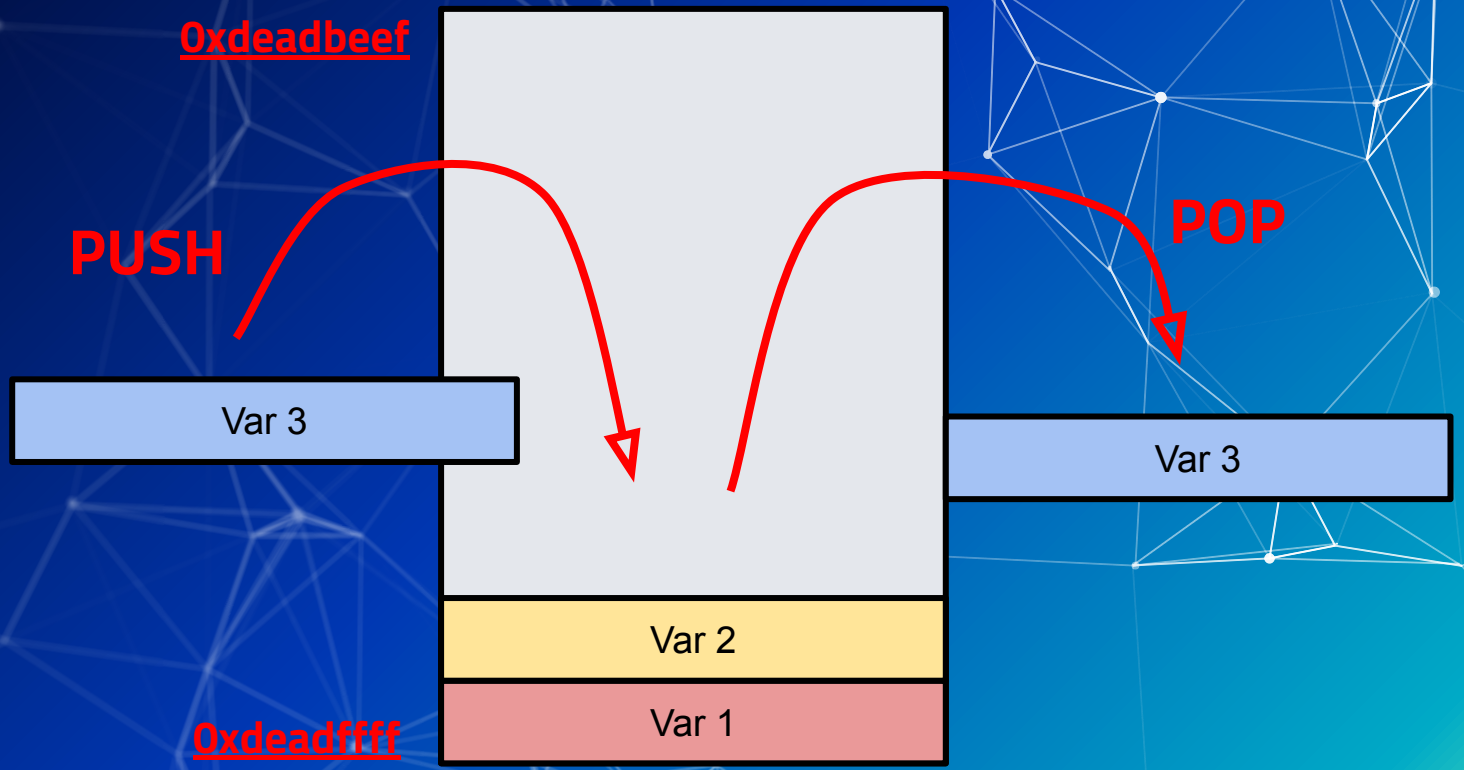


By @domysh

Function asm code

```
0040115c 55              PUSH    RBP
0040115d 48 89 e5        MOV     RBP,RSP
00401160 48 81 ec        SUB     RSP,0xd0
         d0 00 00 00
00401167 48 8d 85        LEA     RAX=>local_d8,[RBP + -0xd0]
         30 ff ff ff
0040116e 48 89 c6        MOV     RSI,RAX
00401171 48 8d 05        LEA     RAX,[DAT_00402004]
         8c 0e 00 00
00401178 48 89 c7        MOV     RDI=>DAT_00402004,RAX
0040117b b8 00 00        MOV     EAX,0x0
         00 00
00401180 e8 cb fe        CALL    <EXTERNAL>::__isoc99_scanf
         ff ff
00401185 90              NOP
00401186 c9              LEAVE
00401187 c3              RET
```

Stack



CALL instruction

```
1  call func_addr # asm function
2
3  # EQUALS TO
4
5  push $ip # push the instruction pointer
6  jmp func_addr
```

RET instruction

```
1  ret # asm function
2
3  # EQUALS TO
4
5  pop $ip # => jmp $ip
```

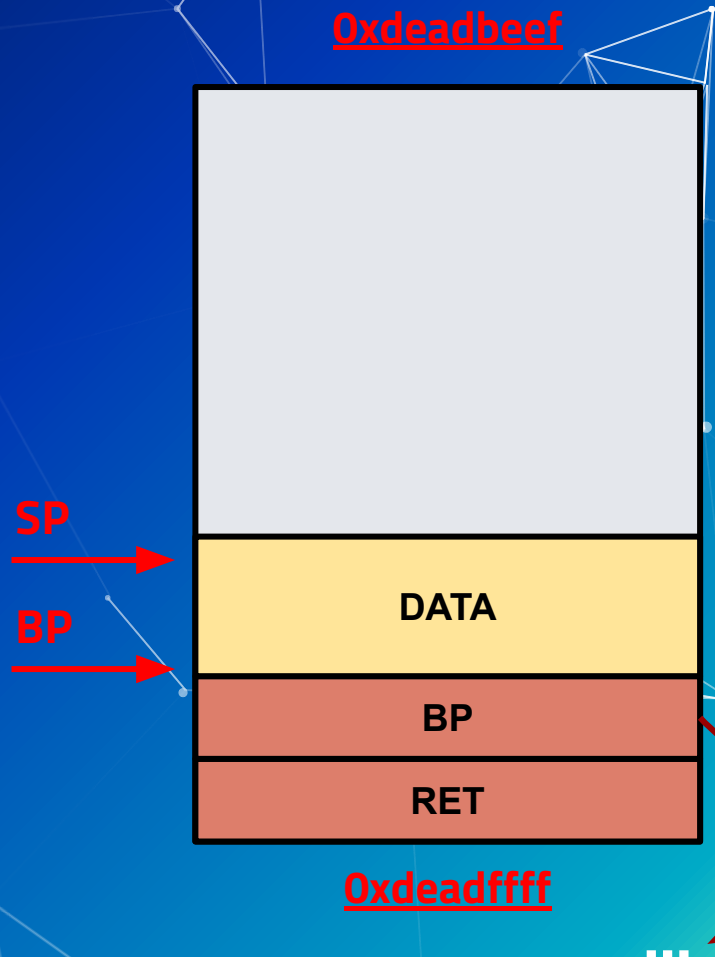
LEAVE instruction

```
1  leave # asm instruction
2
3  # EQUALS TO
4
5  mov $sp, $bp # sp = bp
6  pop $bp # restore the old bp
```

EXECUTION

Let’s see how a stack call works

```
PUSH        RBP
MOV         RBP,RSP
SUB         RSP,0xd0

LEA         RAX=>local_d8,[RBP + -0xd0]

MOV         RSI,RAX
LEA         RAX,[DAT_00402004]

MOV         RDI=>DAT_00402004,RAX
MOV         EAX,0x0

CALL        <EXTERNAL>::__isoc99_scanf

NOP
LEAVE
RET
```

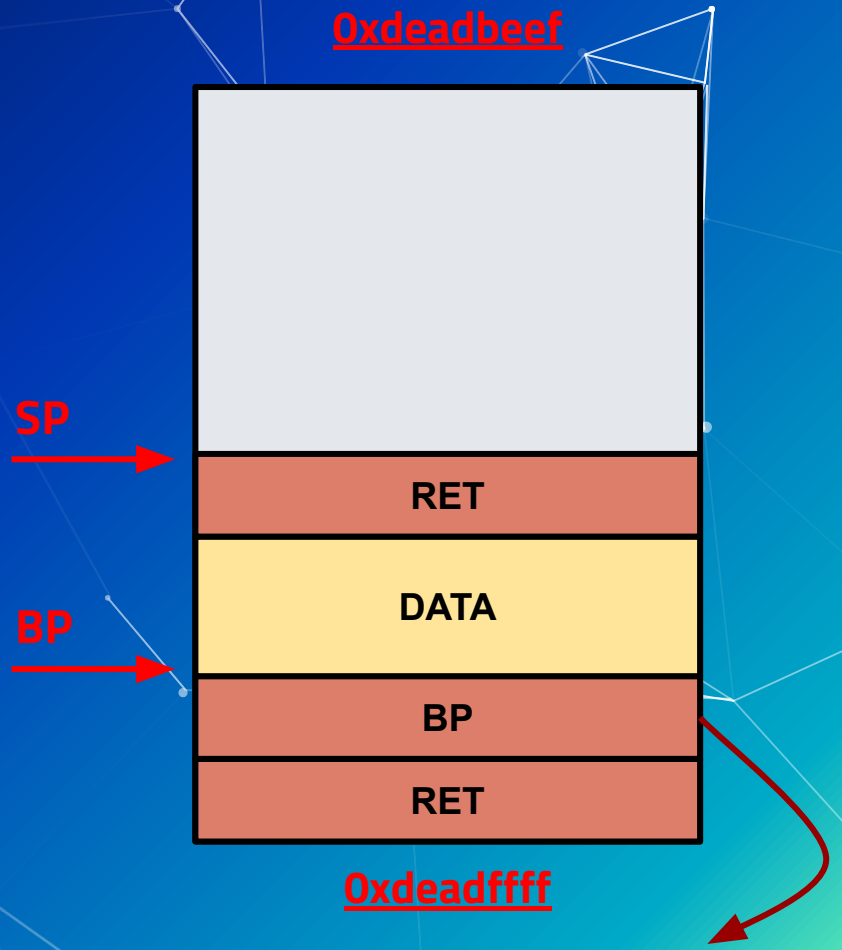


Someone call us...

```
...
call this_func
```

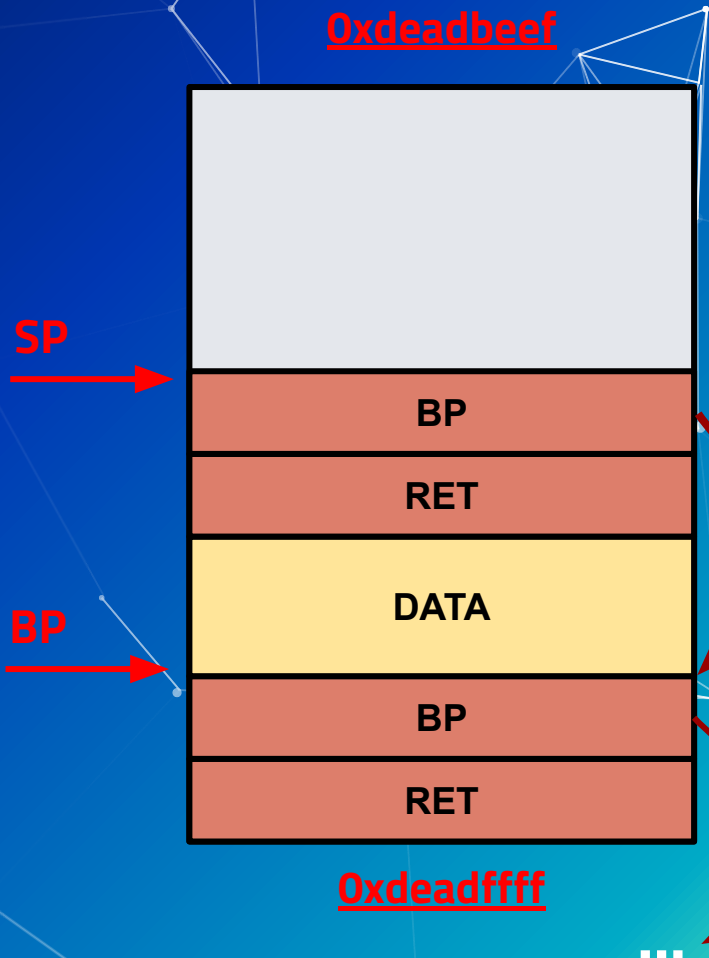
IP

```
PUSH        RBP
MOV         RBP,RSP
SUB         RSP,0xd0

LEA         RAX=>local_d8,[RBP + -0xd0]

MOV         RSI,RAX
LEA         RAX,[DAT_00402004]

MOV         RDI=>DAT_00402004,RAX
MOV         EAX,0x0

CALL        <EXTERNAL>::__isoc99_scanf

NOP
LEAVE
RET
```



IP

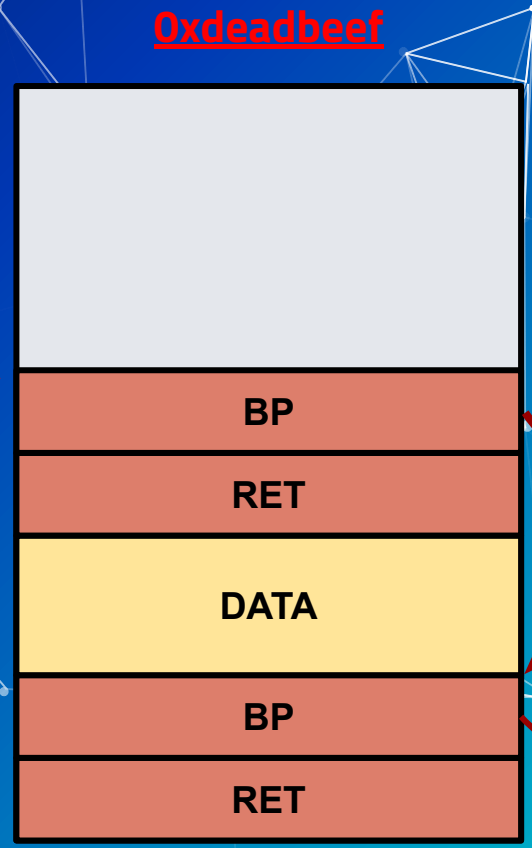
```
PUSH        RBP
MOV         RBP,RSP
SUB         RSP,0xd0

LEA         RAX=>local_d8,[RBP + -0xd0]

MOV         RSI,RAX
LEA         RAX,[DAT_00402004]

MOV         RDI=>DAT_00402004,RAX
MOV         EAX,0x0

CALL        <EXTERNAL>::__isoc99_scanf

NOP
LEAVE
RET
```



IP

```
PUSH        RBP
MOV         RBP,RSP
SUB         RSP,0xd0

LEA         RAX=>local_d8,[RBP + -0xd0]

MOV         RSI,RAX
LEA         RAX,[DAT_00402004]

MOV         RDI=>DAT_00402004,RAX
MOV         EAX,0x0

CALL        <EXTERNAL>::__isoc99_scanf

NOP
LEAVE
RET
```

SP, BP



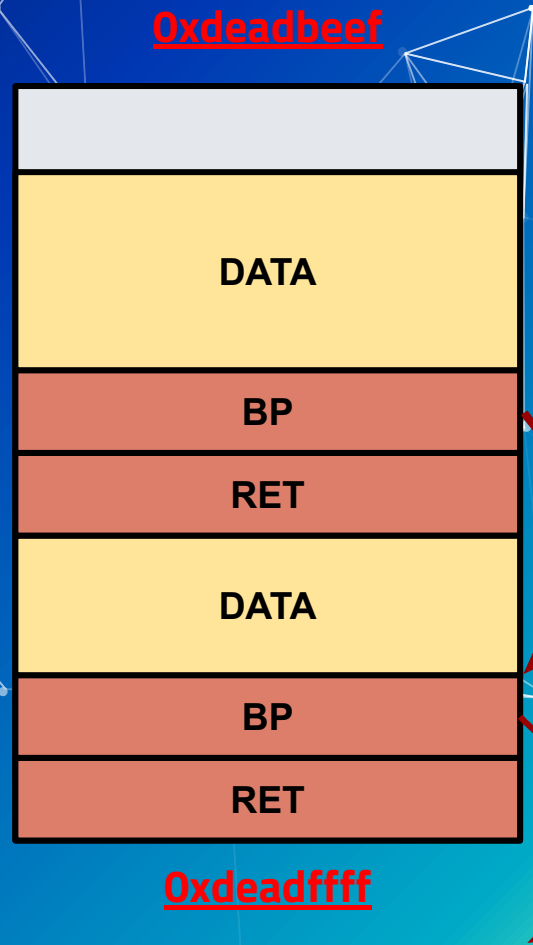
...

```
PUSH        RBP
MOV         RBP,RSP
SUB         RSP,0xd0

LEA         RAX=>local_d8,[RBP + -0xd0]

MOV         RSI,RAX
LEA         RAX,[DAT_00402004]

MOV         RDI=>DAT_00402004,RAX
MOV         EAX,0x0

CALL        <EXTERNAL>::__isoc99_scanf

NOP
LEAVE
RET
```

IP

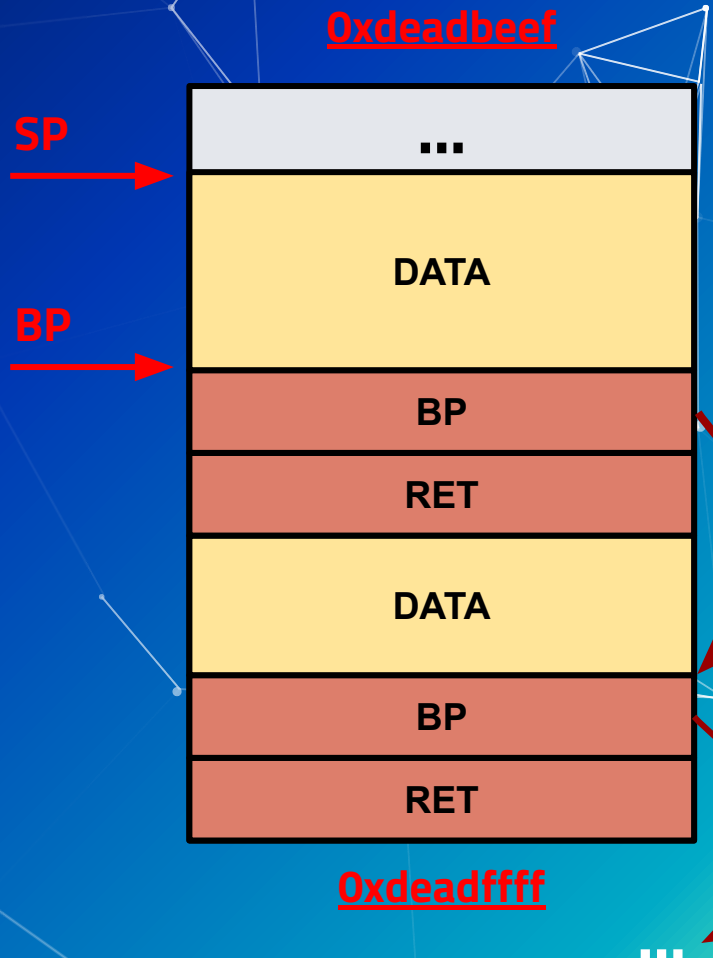
SP

BP



Function body...

```
PUSH        RBP
MOV         RBP,RSP
SUB         RSP,0xd0

LEA         RAX=>local_d8,[RBP + -0xd0]

MOV         RSI,RAX
LEA         RAX,[DAT_00402004]

MOV         RDI=>DAT_00402004,RAX
MOV         EAX,0x0

CALL        <EXTERNAL>::__isoc99_scanf

NOP
LEAVE
RET
```

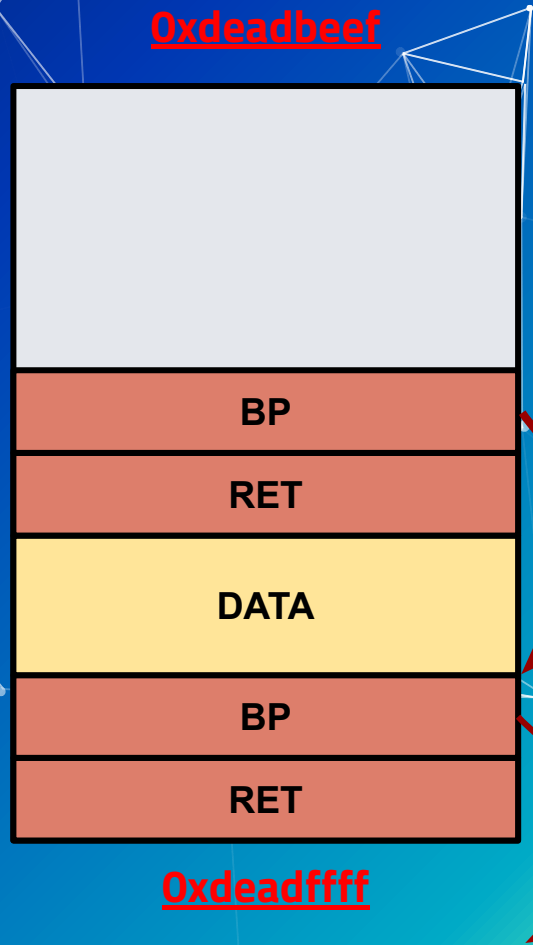


LEAVE pt.1 - MOV $SP, $BP

```
PUSH        RBP
MOV         RBP,RSP
SUB         RSP,0xd0

LEA         RAX=>local_d8,[RBP + -0xd0]

MOV         RSI,RAX
LEA         RAX,[DAT_00402004]

MOV         RDI=>DAT_00402004,RAX
MOV         EAX,0x0

CALL        <EXTERNAL>::__isoc99_scanf

NOP
LEAVE
RET
```

IP

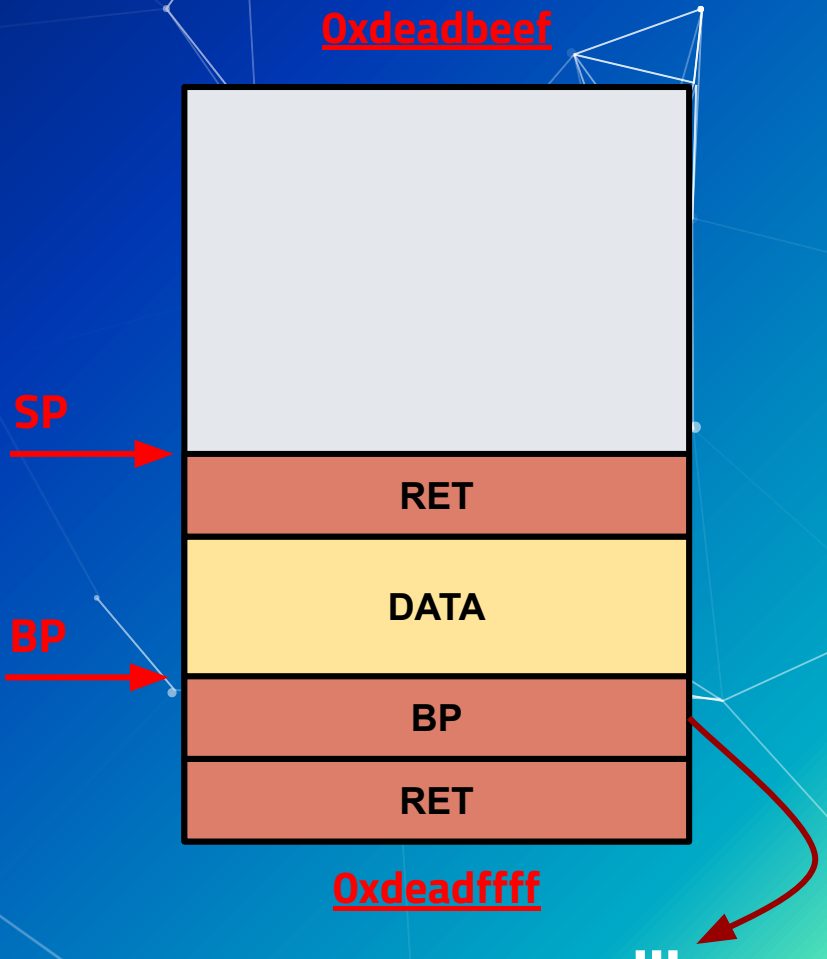
BP, SP



LEAVE pt.2 - POP $BP

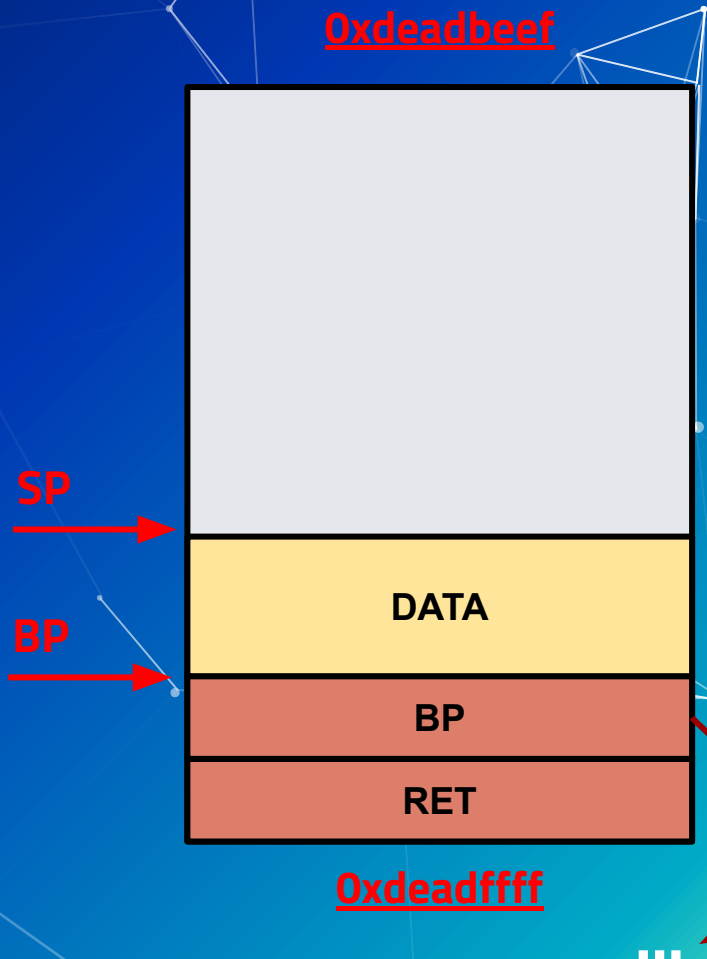
```
PUSH        RBP
MOV         RBP,RSP
SUB         RSP,0xd0

LEA         RAX=>local_d8,[RBP + -0xd0]

MOV         RSI,RAX
LEA         RAX,[DAT_00402004]

MOV         RDI=>DAT_00402004,RAX
MOV         EAX,0x0

CALL        <EXTERNAL>::__isoc99_scanf

NOP
LEAVE
RET
```

IP



IP → [...] (The next instruction after the call that required to execute us)

```
PUSH        RBP
MOV         RBP,RSP
SUB         RSP,0xd0

LEA         RAX=>local_d8,[RBP + -0xd0]

MOV         RSI,RAX
LEA         RAX,[DAT_00402004]

MOV         RDI=>DAT_00402004,RAX
MOV         EAX,0x0

CALL        <EXTERNAL>::__isoc99_scanf

NOP
LEAVE
RET
```



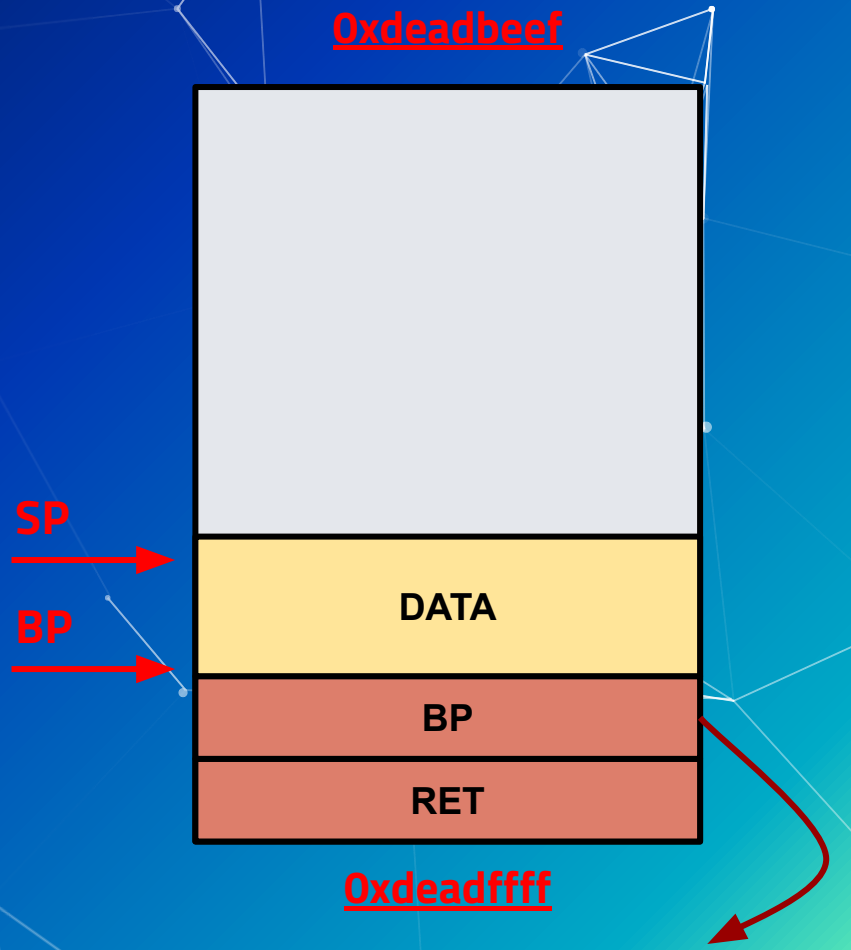
EXECUTION - NO DATA CALL

Let’s see how a stack call works for functions without stack data

Function asm code

```
00401146 55              PUSH    RBP
00401147 48 89 e5        MOV     RBP,RSP
0040114a 48 8b 05        MOV     RAX,qword ptr [presenza_ptr]
         1f 2f 00 00
00401151 48 89 c7        MOV     RDI,RAX
00401154 e8 d7 fe        CALL    <EXTERNAL>::puts
         ff ff
00401159 90              NOP
0040115a 5d              POP     RBP
0040115b c3              RET
```

```
PUSH        RBP
MOV         RBP,RSP
MOV         RAX,qword ptr [presenza_ptr]

MOV         RDI,RAX
CALL        <EXTERNAL>::puts

NOP
POP         RBP
RET
```

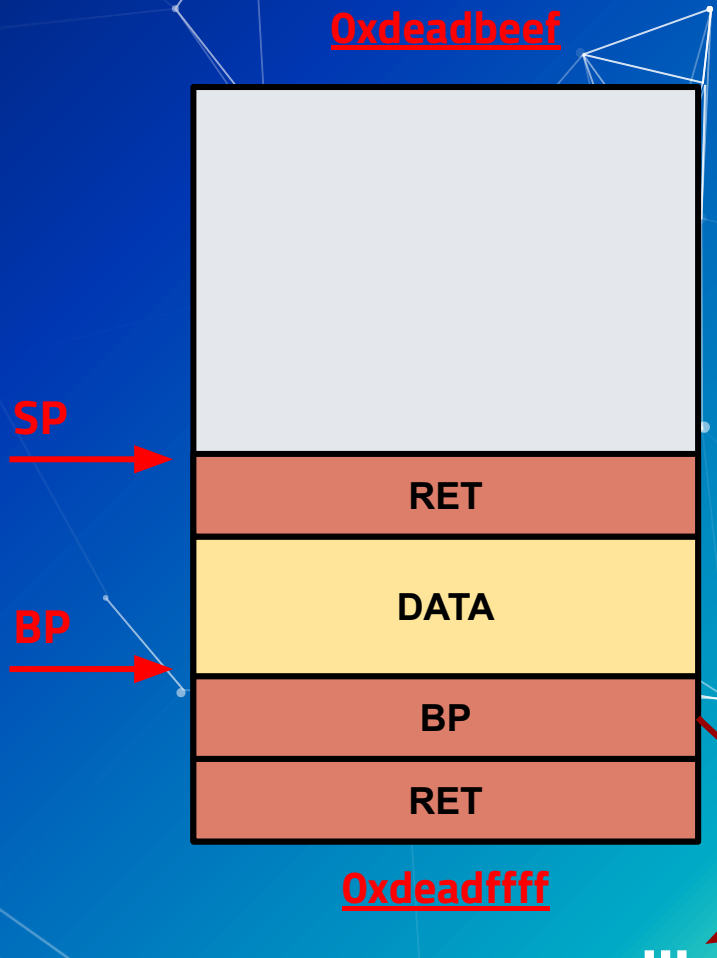


Someone call us...

```
...
call this_func
```

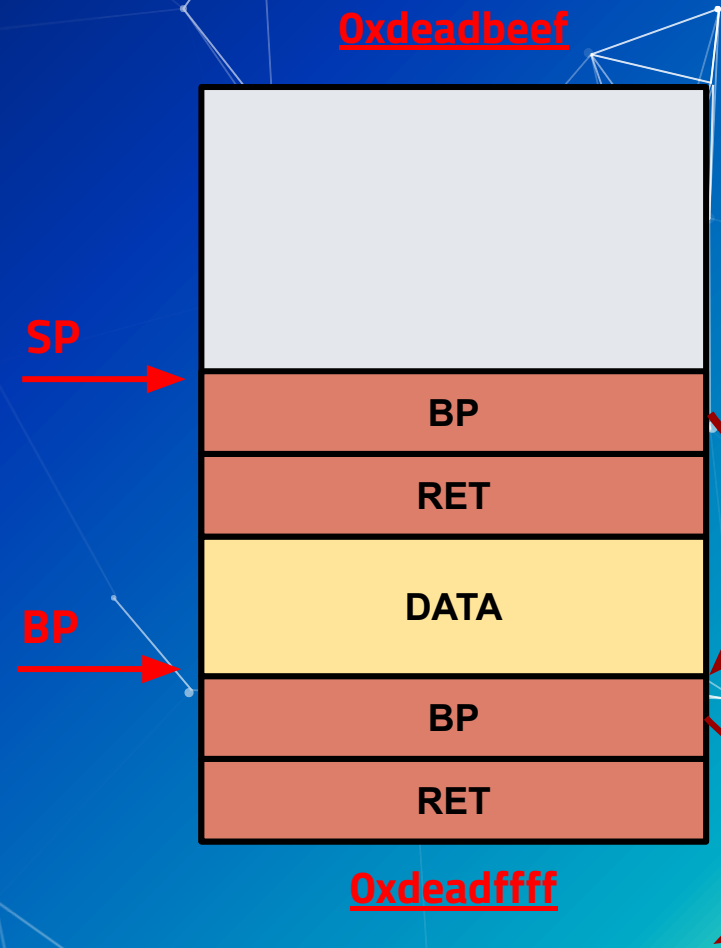
IP

```
PUSH        RBP
MOV         RBP,RSP
MOV         RAX,qword ptr [presenza_ptr]

MOV         RDI,RAX
CALL        <EXTERNAL>::puts

NOP
POP         RBP
RET
```



IP

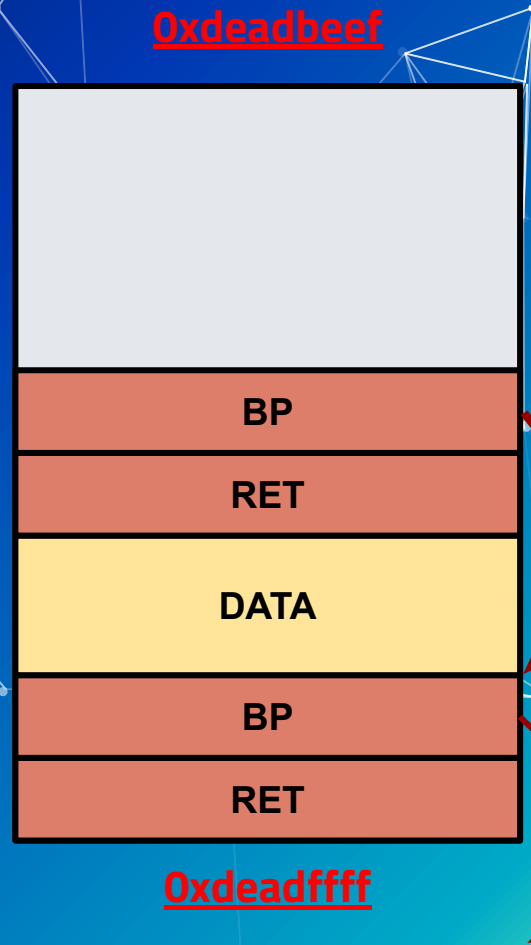
```
PUSH        RBP
MOV         RBP,RSP
MOV         RAX,qword ptr [presenza_ptr]

MOV         RDI,RAX
CALL        <EXTERNAL>::puts

NOP
POP         RBP
RET
```



```
PUSH        RBP
MOV         RBP,RSP
MOV         RAX,qword ptr [presenza_ptr]

MOV         RDI,RAX
CALL        <EXTERNAL>::puts

NOP
POP         RBP
RET
```

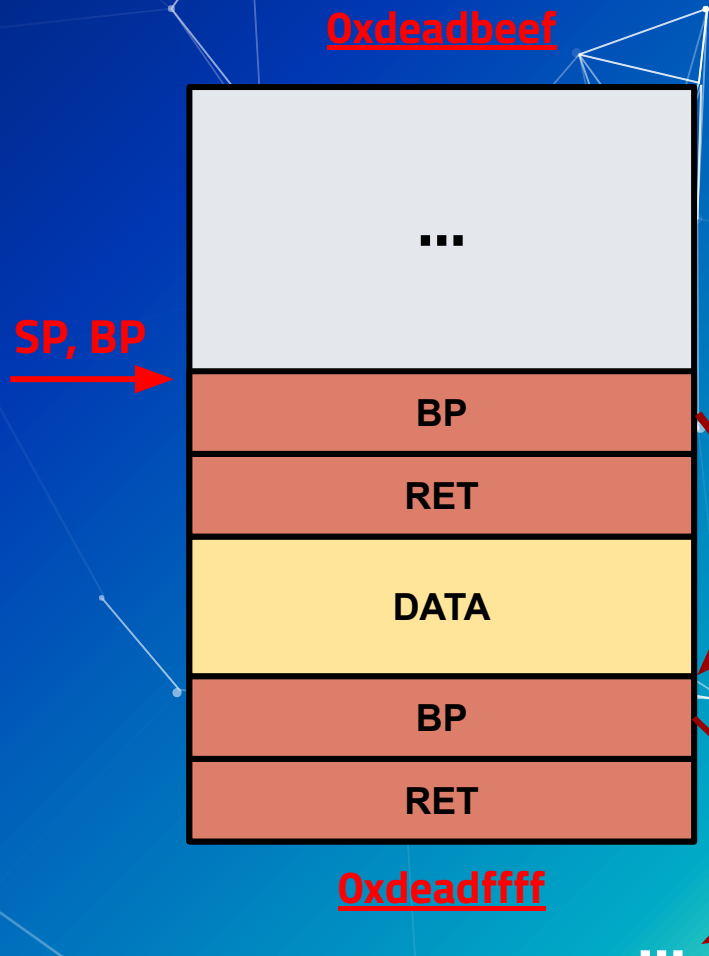
IP

SP, BP



Function body...

```
PUSH        RBP
MOV         RBP,RSP
MOV         RAX,qword ptr [presenza_ptr]

MOV         RDI,RAX
CALL        <EXTERNAL>::puts

NOP
POP         RBP
RET
```

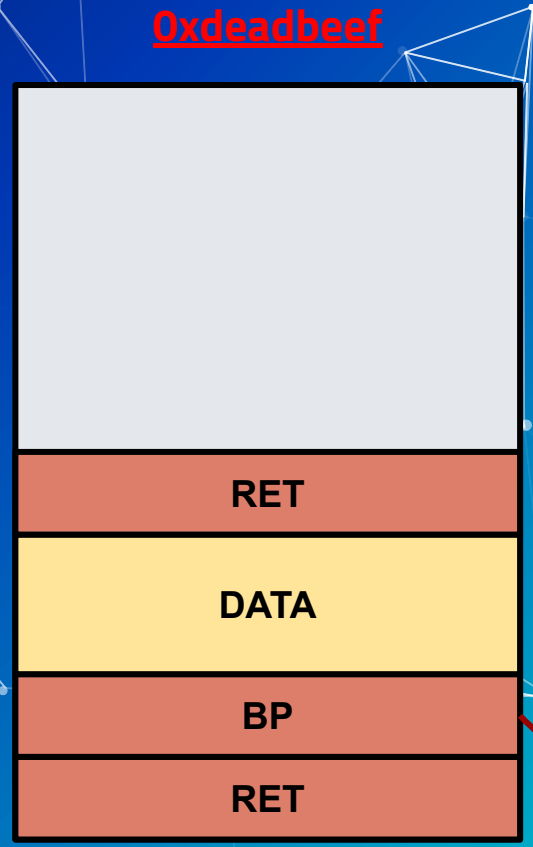


```
PUSH        RBP
MOV         RBP,RSP
MOV         RAX,qword ptr [presenza_ptr]

MOV         RDI,RAX
CALL        <EXTERNAL>::puts

NOP
POP         RBP
RET
```

IP

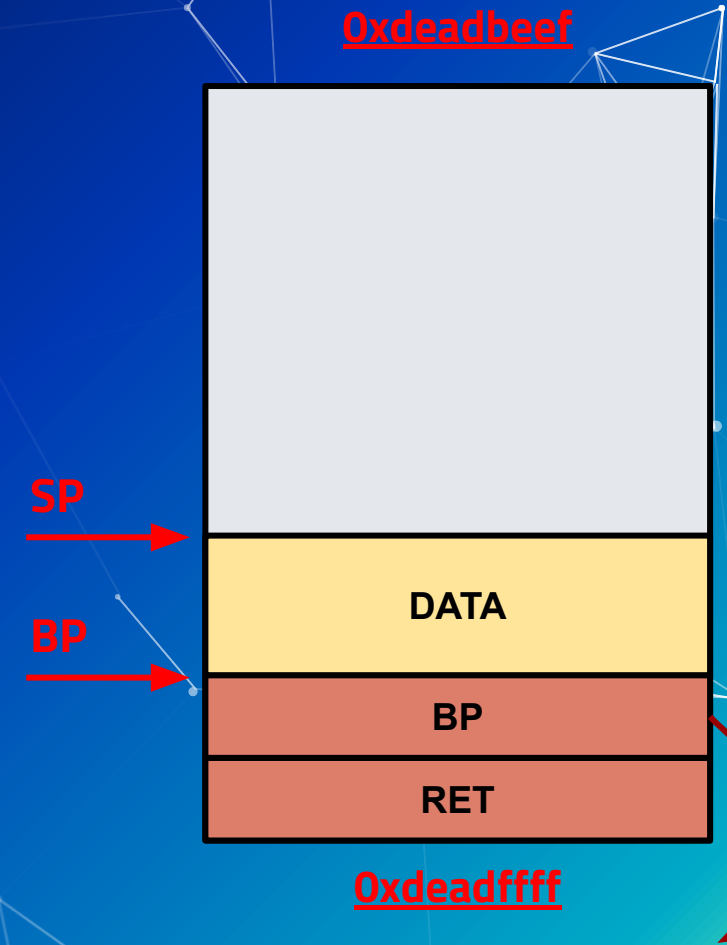
SP

BP



IP → [...] (The next instruction after the call that required to execute us)

```
PUSH        RBP
MOV         RBP,RSP
MOV         RAX,qword ptr [presenza_ptr]

MOV         RDI,RAX
CALL        <EXTERNAL>::puts

NOP
POP         RBP
RET
```



Happy pwning :)