~/DCcurity/intro_reversing

...

Un intento de taller en la pandemia

~/DCcurity/intro_reversing

$ whoami

- Octavio Gianatiempo
- Biólogo
- Estudiante de Compu
- Organizador de DCcurity
- Tw: @ogianatiempo

--Más--

~/DCcurity/intro_reversing

Reverse Engineering (a.k.a. reversing)

El proceso de deconstruir un objeto creado por el hombre para revelar su funcionamiento y diseño.

En el caso de un programa ejecutable para deconstruirlo tenemos que desensamblarlo.

--Más--

Desensamblado

Los procesadores ejecutan código de máquina, lo que se conoce como binario.

Para que los humanos podamos entender ese código existe el lenguaje ensamblador.

Las instrucciones de este lenguaje se corresponden uno a uno con el código de máquina.

Un ensamblador traduce código en lenguaje ensamblador a código de máquina.

El proceso inverso es el desensamblado.

--Más--

Decompilado

Cuando programamos, generalmente escribimos código en lenguajes de alto nivel.

Para generar un archivo ejecutable, un compilador traduce este código a lenguaje ensamblador o directamente a código de máquina.

El proceso inverso es el decompilado.

Decompilar es mucho más difícil que desensamblar porque en la compilación se pierde información y porque depende mucho del lenguaje usado.

--Más--

~/DCcurity/intro_reversing

```c
int main(int argc, char** argv) {
      long int debug = 0;
      char buf[8];

      gets(buf);

      if (debug) {
        print_flag();
      }
}
```

```
main:
 push rbp
 mov rbp,rsp
 sub rsp,0x20
 mov [rbp-0x14],edi
 mov [rbp-0x20],rsi
 mov [rbp-0x8],0x0
 lea rax,[rbp-0x10]
 mov rdi,rax
 call 400450 <gets@plt>
 cmp [rbp-0x8],0x0
 je 400583 <main+0x2f>
 call 400544 <print_flag>
 mov eax,0x0
 leave
 ret
```

```
55
48 89 e5
48 83 ec 20
89 7d ec
48 89 75 e0
48 c7 45 f8 00 00 00 00
48 8d 45 f0
48 89 c7
e8 d9 fe ff ff
48 83 7d f8 00
74 05
e8 c1 ff ff ff
b8 00 00 00 00
c9
c3
```

Fetch, decode, execute

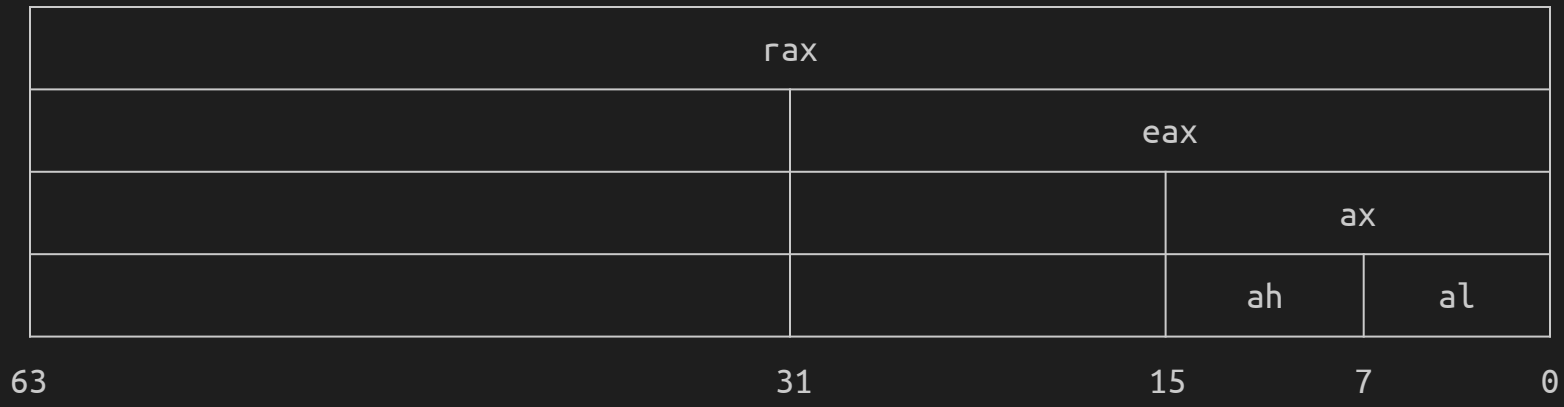
- Leer una instrucción de la memoria
- Interpretarla
- Ejecutarla
- Repetir

```
  main:
-> push rbp
   mov rbp,rsp
   sub rsp,0x20
   mov DWORD PTR [rbp-0x14],edi
   mov QWORD PTR [rbp-0x20],rsi
   mov QWORD PTR [rbp-0x8],0x0
   lea rax,[rbp-0x10]
   mov rdi,rax
   call 400450 <gets@plt>
   cmp QWORD PTR [rbp-0x8],0x0
   je 400583 <main+0x2f>
   call 400544 <print_flag>
   mov eax,0x0
   leave
   ret
```

Instrucciones

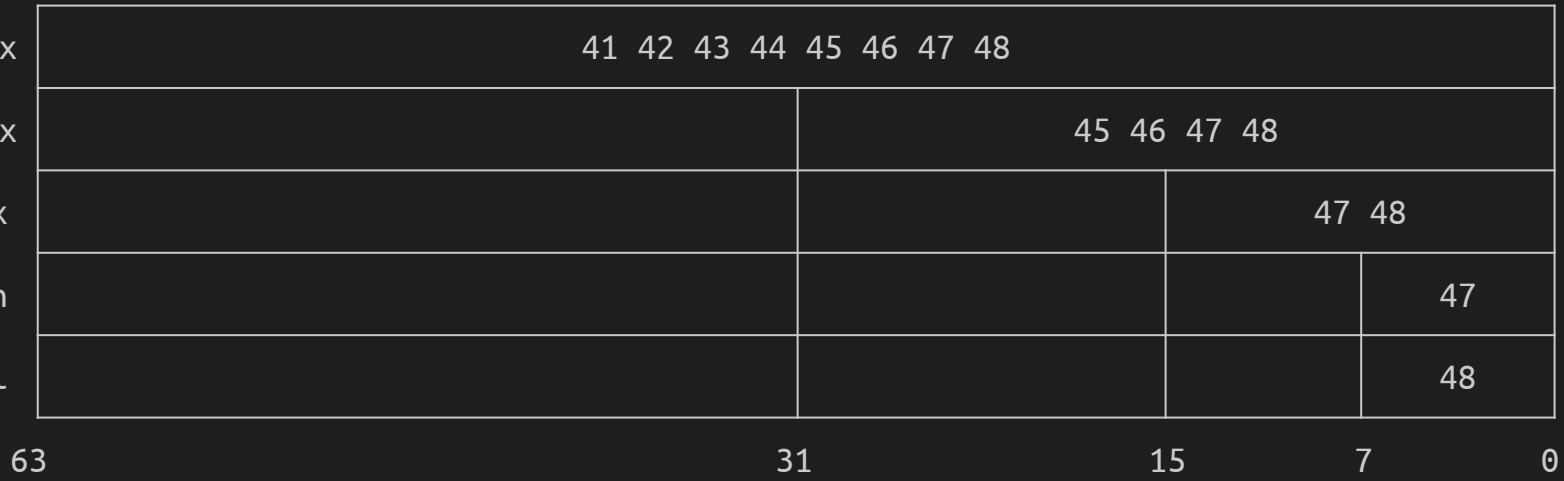
~/DCcurity/intro_reversing

Registros



~/DCcurity/intro_reversing

Registros



~/DCcurity/intro_reversing

```
# Operaciones con registros

ASM:
-> mov rax, 0x1
   mov rcx, 0x2
   mov rbx, 0x3
   sub rbx, 0x1
   mov rdx, rax
   mov rax, 0x4fa35
   mov rcx, [rax]
   mov [rax], rbx
   nop
```

~/DCcurity/intro_reversing

```
# Operaciones con registros

ASM:
   mov rax, 0x1
-> mov rcx, 0x2
   mov rbx, 0x3
   sub rbx, 0x1
   mov rdx, rax
   mov rax, 0x4fa35
   mov rcx, [rax]
   mov [rax], rbx
   nop
```

~/DCcurity/intro_reversing

```
# Operaciones con registros

ASM:
   mov rax, 0x1
   mov rcx, 0x2
-> mov rbx, 0x3
   sub rbx, 0x1
   mov rdx, rax
   mov rax, 0x4fa35
   mov rcx, [rax]
   mov [rax], rbx
   nop
```

~/DCcurity/intro_reversing

```
# Operaciones con registros

ASM:
   mov rax, 0x1
   mov rcx, 0x2
   mov rbx, 0x3
-> sub rbx, 0x1
   mov rdx, rax
   mov rax, 0x4fa35
   mov rcx, [rax]
   mov [rax], rbx
   nop
```

~/DCcurity/intro_reversing

```
# Operaciones con registros

ASM:
   mov rax, 0x1
   mov rcx, 0x2
   mov rbx, 0x3
   sub rbx, 0x1
-> mov rdx, rax
   mov rax, 0x4fa35
   mov rcx, [rax]
   mov [rax], rbx
   nop
```

~/DCcurity/intro_reversing

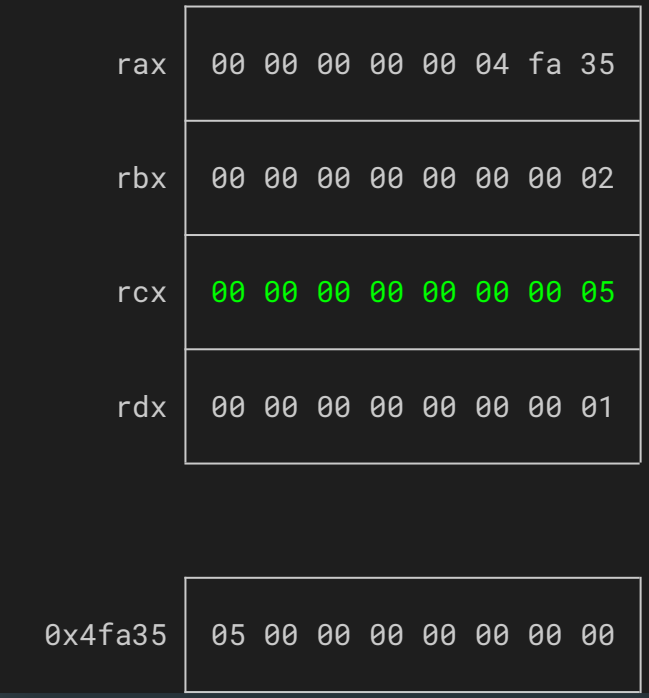
```
# Operaciones con registros

ASM:
   mov rax, 0x1
   mov rcx, 0x2
   mov rbx, 0x3
   sub rbx, 0x1
   mov rdx, rax
-> mov rax, 0x4fa35
   mov rcx, [rax]
   mov [rax], rbx
   nop
```

~/DCcurity/intro_reversing

```
# Operaciones con registros

ASM:
   mov rax, 0x1
   mov rcx, 0x2
   mov rbx, 0x3
   sub rbx, 0x1
   mov rdx, rax
   mov rax, 0x4fa35
-> mov rcx, [rax]
   mov [rax], rbx
   nop
```

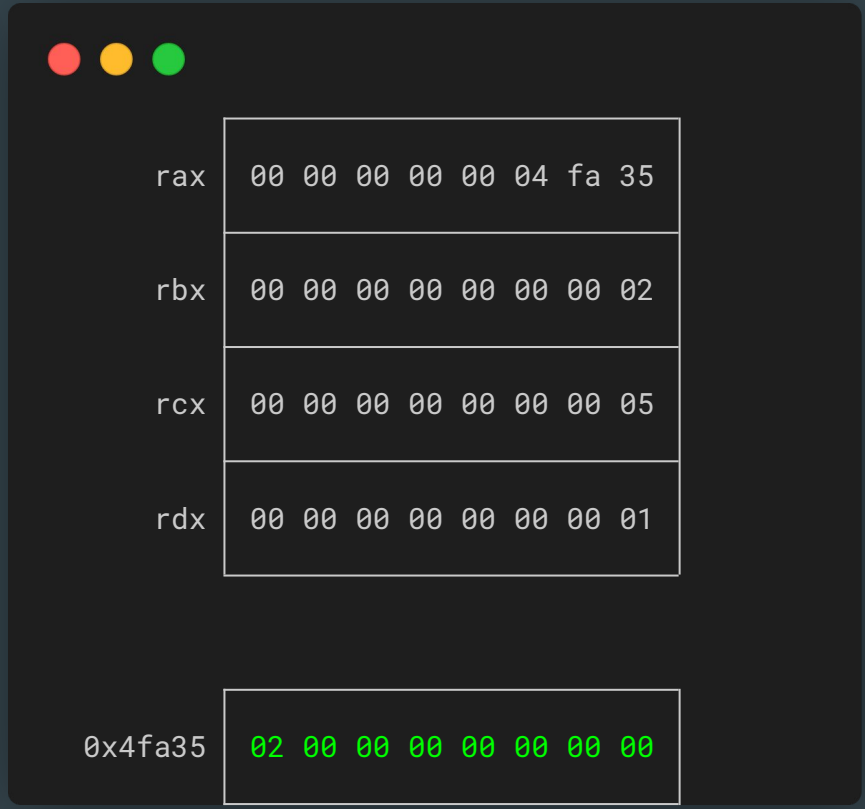
~/DCcurity/intro_reversing

```
# Operaciones con registros

ASM:
   mov rax, 0x1
   mov rcx, 0x2
   mov rbx, 0x3
   sub rbx, 0x1
   mov rdx, rax
   mov rax, 0x4fa35
   mov rcx, [rax]
-> mov [rax], rbx
   nop
```



~/DCcurity/intro_reversing

```
# Operaciones con registros

ASM:
   mov rax, 0x1
   mov rcx, 0x2
   mov rbx, 0x3
   sub rbx, 0x1
   mov rdx, rax
   mov rax, 0x4fa35
   mov rcx, [rax]
   mov [rax], rbx
-> nop
```



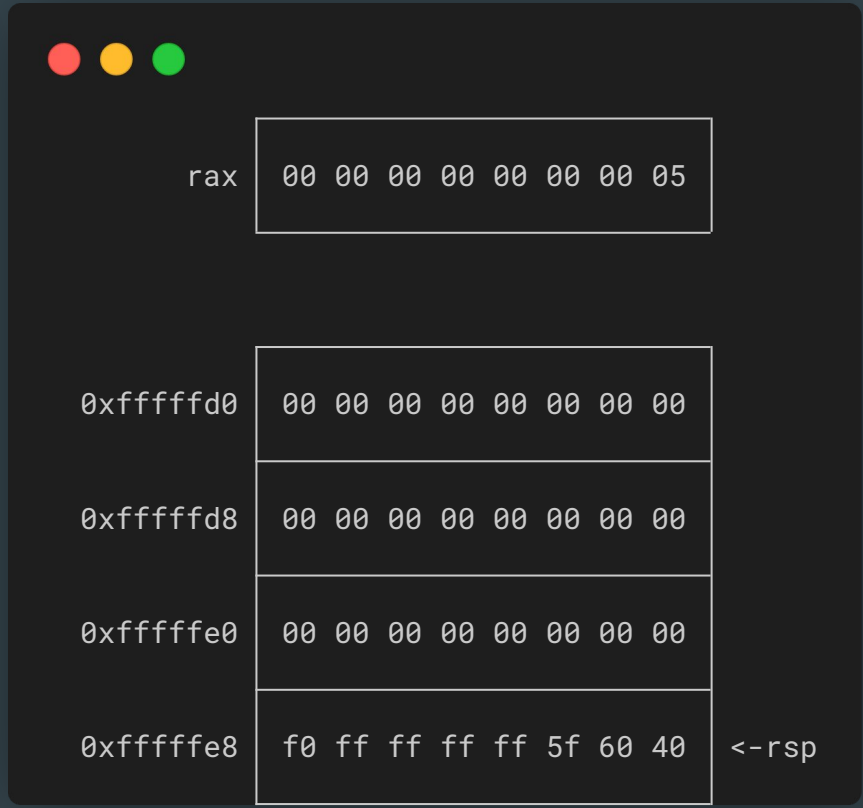
~/DCcurity/intro_reversing

Stack

- Es una estructura de almacenamiento temporario
- Guarda variables locales
- Guarda argumentos en 32bit
- Crece hacia direcciones de memoria más bajas (pila lifo)
- Dos registros apuntan al stack: rbp y rsp

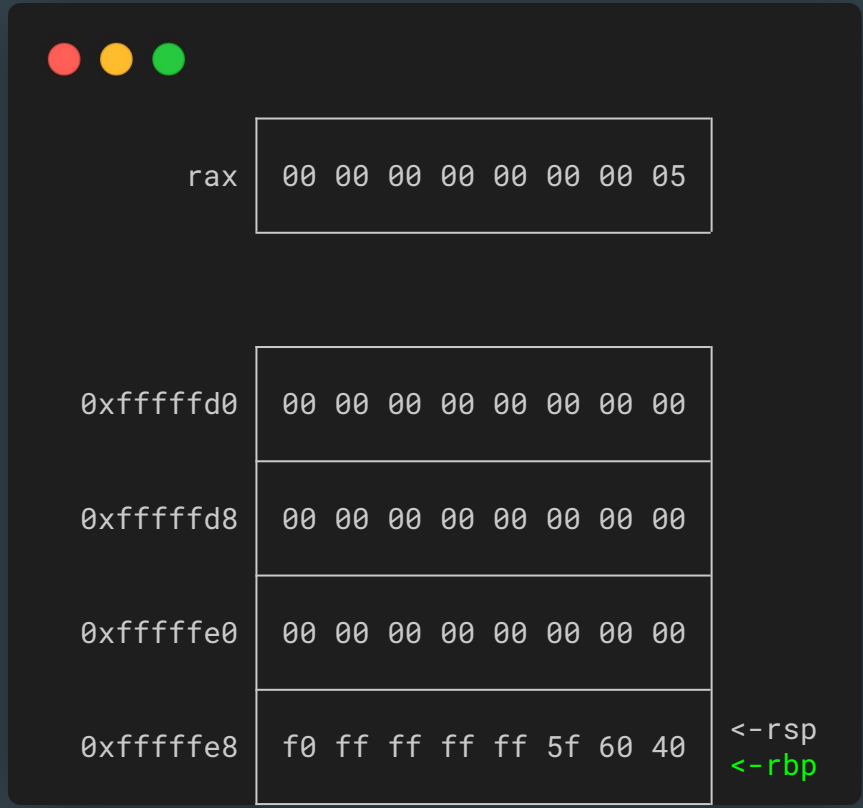
~/DCcurity/intro_reversing

```
# Operaciones con stack

ASM:
-> mov rbp,rsp
   sub rsp,0x10                 ; reserva espacio
   mov [rbp-0x08], rax          ; variable local
   mov [rbp-0x10], 0x2          ; variable local
   push 0x1
   pop rax
   nop
```



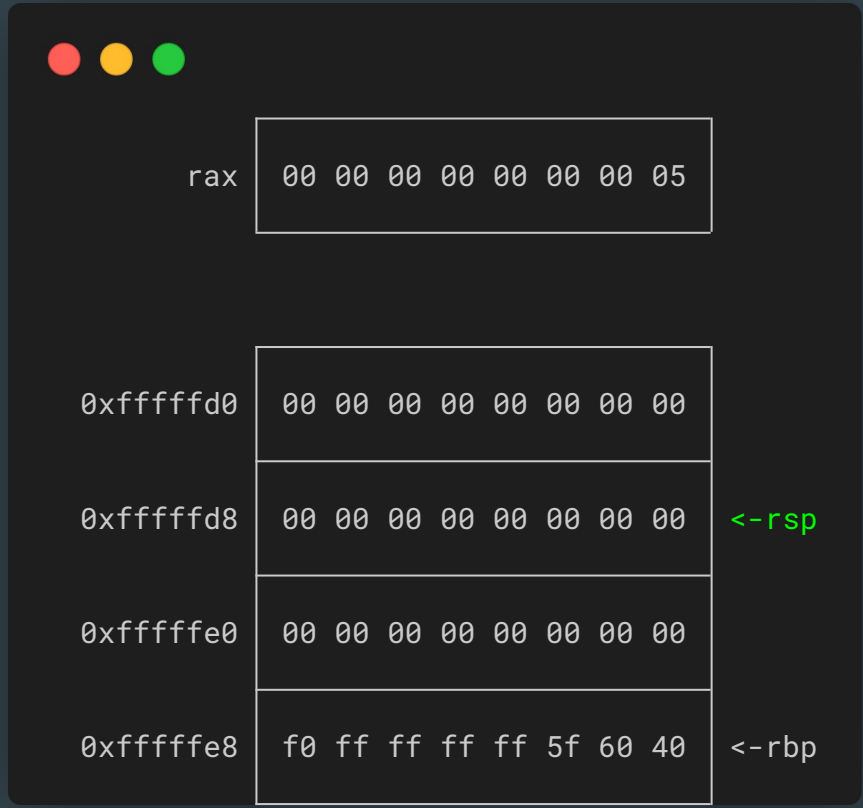
~/DCcurity/intro_reversing

```
# Operaciones con stack

ASM:
   mov rbp,rsp
-> sub rsp,0x10                 ; reserva espacio
   mov [rbp-0x08], rax          ; variable local
   mov [rbp-0x10], 0x2          ; variable local
   push 0x1
   pop rax
   nop
```



~/DCcurity/intro_reversing

```
# Operaciones con stack

ASM:
   mov rbp,rsp
   sub rsp,0x10                ; reserva espacio
-> mov [rbp-0x08], rax         ; variable local
   mov [rbp-0x10], 0x2         ; variable local
   push 0x1
   pop rax
   nop
```



~/DCcurity/intro_reversing

```
# Operaciones con stack

ASM:
   mov rbp,rsp
   sub rsp,0x10                 ; reserva espacio
   mov [rbp-0x08], rax          ; variable local
-> mov [rbp-0x10], 0x2          ; variable local
   push 0x1
   pop rax
   nop
```

~/DCcurity/intro_reversing

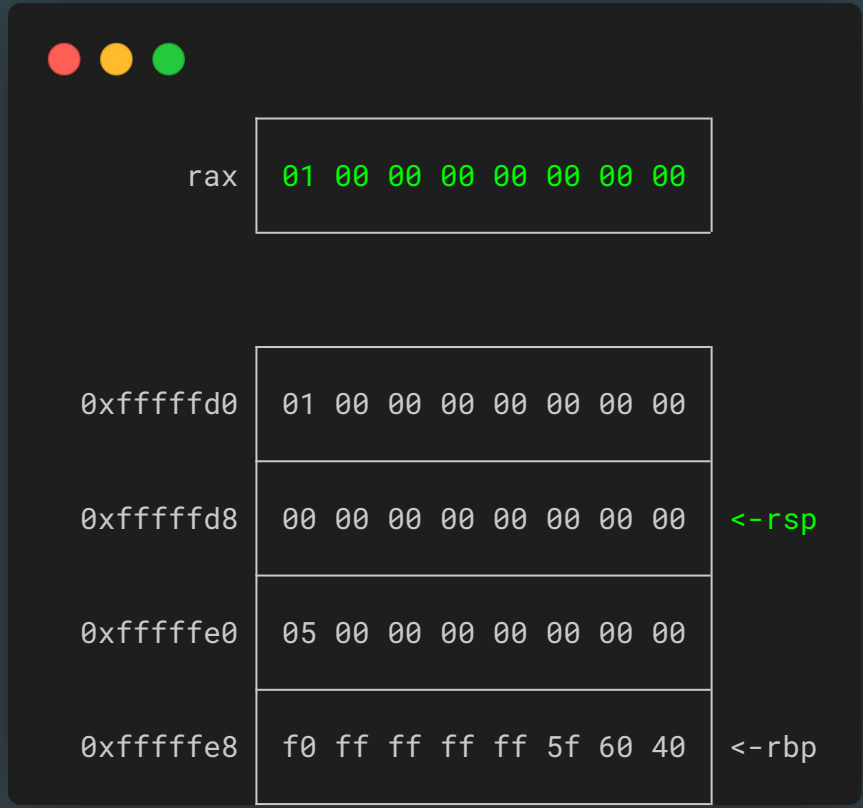
```
# Operaciones con stack

ASM:
   mov rbp,rsp
   sub rsp,0x10                 ; reserva espacio
   mov [rbp-0x08], rax          ; variable local
   mov [rbp-0x10], 0x2          ; variable local
-> push 0x1
   pop rax
   nop
```

~/DCcurity/intro_reversing

```
# Operaciones con stack

ASM:
   mov rbp,rsp
   sub rsp,0x10                 ; reserva espacio
   mov [rbp-0x08], rax          ; variable local
   mov [rbp-0x10], 0x2          ; variable local
   push 0x1
-> pop rax
   nop
```

~/DCcurity/intro_reversing

```
# Operaciones con stack

ASM:
   mov rbp,rsp
   sub rsp,0x10                 ; reserva espacio
   mov [rbp-0x08], rax          ; variable local
   mov [rbp-0x10], 0x2          ; variable local
   push 0x1
   pop rax
-> nop
```



~/DCcurity/intro_reversing

Llamado a función (x64)

C:

```
fun(1,2,3)
```

ASM:

```
-> mov rdi, 0x1
   mov rsi, 0x2
   mov rdx, 0x3                fun:
   call 400450 <fun> -------> push rbp
   mov rdi, rdx   <----       mov rbp,rsp
                      |       .
                      |       .
                      |       leave
                      ------  ret
```

0xffffffc0	00 00 00 00 00 00 00 00
0xffffffc8	00 00 00 00 00 00 00 00
0xffffffd0	00 00 00 00 00 00 00 00
0xffffffd8	00 00 00 00 00 00 00 00
0xffffffe0	00 00 00 00 00 00 00 00
0xffffffe8	00 00 00 00 00 00 00 00

~/DCcurity/intro_reversing

```
# Llamado a función (x64)

C:
  fun(1,2,3)

ASM:
   mov rdi, 0x1
-> mov rsi, 0x2
   mov rdx, 0x3                fun:
   call 400450 <fun> -------> push rbp
   mov rdi, rdx   <----       mov rbp,rsp
                      |       .
                      |       .
                      |       leave
                      ------ ret
```

0xffffffc0	00 00 00 00 00 00 00 00
0xffffffc8	00 00 00 00 00 00 00 00
0xffffffd0	00 00 00 00 00 00 00 00
0xffffffd8	00 00 00 00 00 00 00 00
0xffffffe0	00 00 00 00 00 00 00 00
0xffffffe8	00 00 00 00 00 00 00 00

~/DCcurity/intro_reversing

```
# Llamado a función (x64)

C:
  fun(1,2,3)

ASM:
   mov rdi, 0x1
   mov rsi, 0x2
-> mov rdx, 0x3                fun:
   call 400450 <fun> --------> push rbp
   mov rdi, rdx   <----        mov rbp,rsp
                       |       .
                       |       .
                       |       leave
                       ------- ret
```

0xffffffc0	00 00 00 00 00 00 00 00
0xffffffc8	00 00 00 00 00 00 00 00
0xffffffd0	00 00 00 00 00 00 00 00
0xffffffd8	00 00 00 00 00 00 00 00
0xffffffe0	00 00 00 00 00 00 00 00
0xffffffe8	00 00 00 00 00 00 00 00

~/DCcurity/intro_reversing

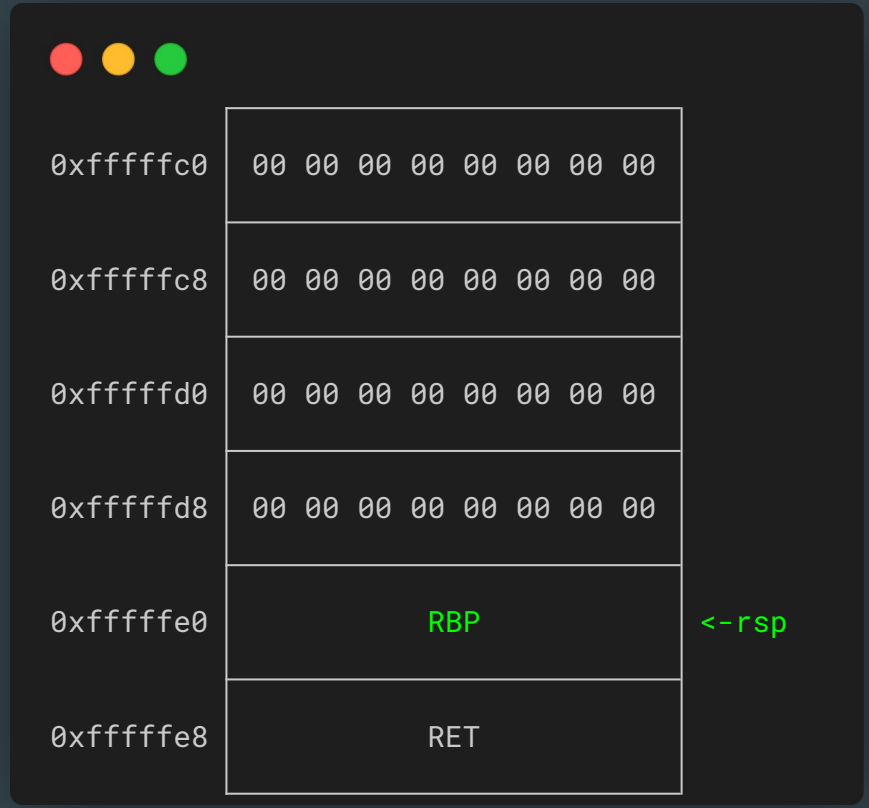
```
# Llamado a función (x64)

C:
  fun(1,2,3)

ASM:
   mov rdi, 0x1
   mov rsi, 0x2
   mov rdx, 0x3                fun:
-> call 400450 <fun> -------> push rbp
   mov rdi, rdx   <----        mov rbp,rsp
                       |       .
                       |       .
                       |       leave
                       ------- ret
```

0xffffffc0	00 00 00 00 00 00 00 00
0xffffffc8	00 00 00 00 00 00 00 00
0xffffffd0	00 00 00 00 00 00 00 00
0xffffffd8	00 00 00 00 00 00 00 00
0xffffffe0	00 00 00 00 00 00 00 00
0xffffffe8	00 00 00 00 00 00 00 00

```
# Llamado a función (x64)

C:
  fun(1,2,3)

ASM:
   mov rdi, 0x1
   mov rsi, 0x2
   mov rdx, 0x3                fun:
   call 400450 <fun> --------> push rbp    <-
   mov rdi, rdx   <----        mov rbp,rsp
                       |       .
                       |       .
                       |       leave
                       ------- ret
```

~/DCcurity/intro_reversing

```
# Llamado a función (x64)

C:
  fun(1,2,3)

ASM:
   mov rdi, 0x1
   mov rsi, 0x2
   mov rdx, 0x3                fun:
   call 400450 <fun> -------> push rbp
   mov rdi, rdx   <----       mov rbp,rsp <-
                      |       .
                      |       .
                      |       leave
                      ------  ret
```



```
# Llamado a función (x64)

C:
  fun(1,2,3)

ASM:
   mov rdi, 0x1
   mov rsi, 0x2
   mov rdx, 0x3                fun:
   call 400450 <fun> -------> push rbp
   mov rdi, rdx   <----       mov rbp,rsp
                      |       .           <-
                      |       .
                      |       leave
                      ------  ret
```

~/DCcurity/intro_reversing

```
# Llamado a función (x64)

C:
  fun(1,2,3)

ASM:
   mov rdi, 0x1
   mov rsi, 0x2
   mov rdx, 0x3                fun:
   call 400450 <fun> -------> push rbp
   mov rdi, rdx   <----       mov rbp,rsp
                      |       .
                      |       .           <-
                      |       leave
                      ------  ret
```

~/DCcurity/intro_reversing

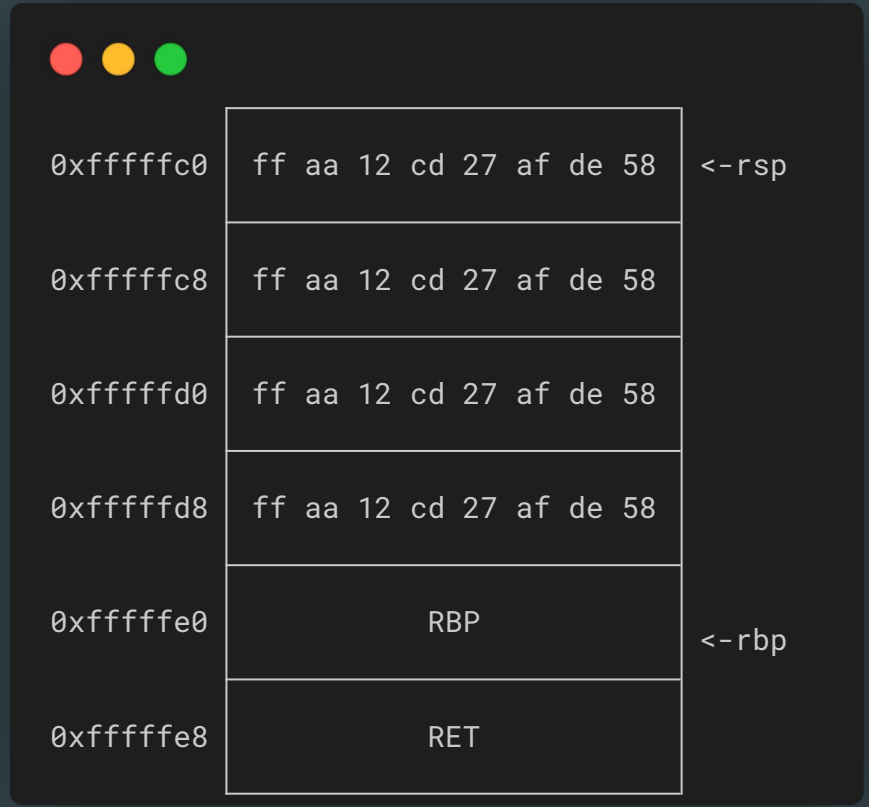
Llamado a función (x64)

C:

```
fun(1,2,3)
```

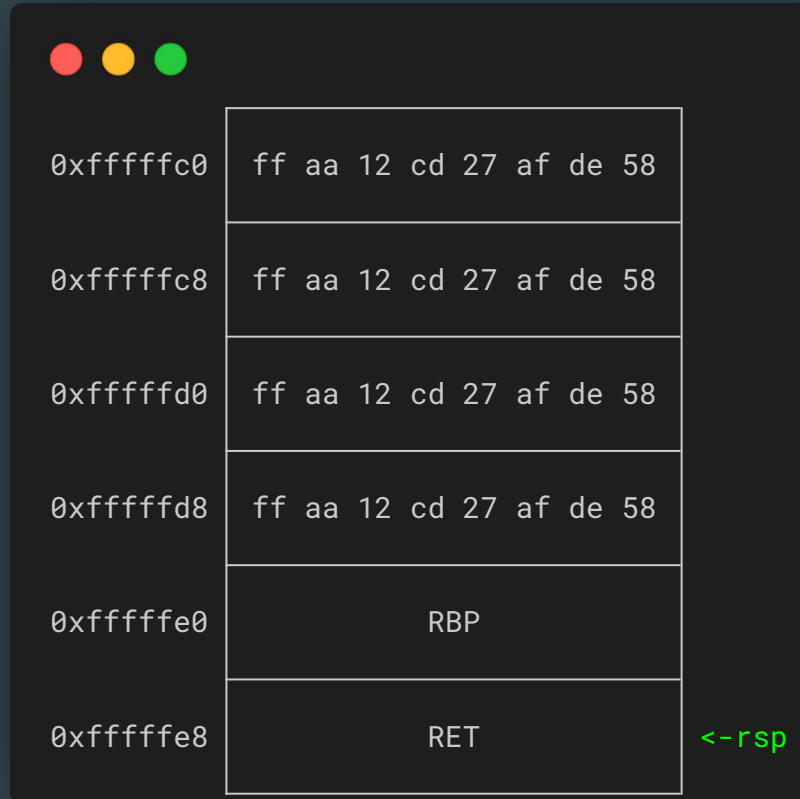
ASM:

```
   mov rdi, 0x1
   mov rsi, 0x2
   mov rdx, 0x3                fun:
   call 400450 <fun> -------> push rbp
   mov rdi, rdx   <----       mov rbp,rsp
                      |       .
                      |       .
                      |       leave       <-
                      ------ ret
```



~/DCcurity/intro_reversing

```
# Llamado a función (x64)

C:
  fun(1,2,3)

ASM:
   mov rdi, 0x1
   mov rsi, 0x2
   mov rdx, 0x3                fun:
   call 400450 <fun> --------> push rbp
   mov rdi, rdx   <----        mov rbp,rsp
                       |       .
                       |       .
                       |       leave
                       ------- ret          <-
```



~/DCcurity/intro_reversing

Llamado a función (x64)

C:

```
fun(1,2,3)
```

ASM:

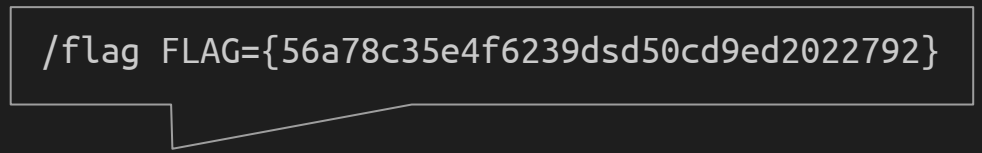
```
   mov rdi, 0x1
   mov rsi, 0x2
   mov rdx, 0x3                fun:
   call 400450 <fun> -------> push rbp
-> mov rdi, rdx   <----       mov rbp,rsp
                      |       .
                      |       .
                      |       leave
                      ------- ret
```

0xffffffc0	ff aa 12 cd 27 af de 58
0xffffffc8	ff aa 12 cd 27 af de 58
0xffffffd0	ff aa 12 cd 27 af de 58
0xffffffd8	ff aa 12 cd 27 af de 58
0xffffffe0	RBP
0xffffffe8	RET

~/DCcurity/intro_reversing

Challenges

- Hay 3 y vamos a ver juntos el primero.
- El objetivo es lograr imprimir la flag.
- Mandarla al bot @dccurity_ctf_bot para recibir puntos:



--Más--

~/DCcurity/intro_reversing

```
# challenge_00

int main(int argc, char** argv) {
      long int debug = 0;    <---- Variable local
      char buf[8];           <---- Variable local

      gets(buf);

      if (debug) {
        print_flag();
      }
}
```

```
main:
 push rbp
 mov rbp,rsp
 sub rsp,0x20
 mov [rbp-0x14],edi              ; argc
 mov [rbp-0x20],rsi              ; argv
 mov [rbp-0x8],0x0               ; debug
 lea rax,[rbp-0x10]              ; buf
 mov rdi,rax
 call 400450 <gets@plt>
 cmp [rbp-0x8],0x0
 je 400583 <main+0x2f>
 call 400544 <print_flag>
 mov eax,0x0
 leave
 ret
```

~/DCcurity/intro_reversing

```
# challenge_00

int main(int argc, char** argv) {
      long int debug = 0;
      char buf[8];

      gets(buf);

      if (debug) {
        print_flag();   <------- !!!
      }
}
```

Puedo lograr que se ejecute print_flag()?

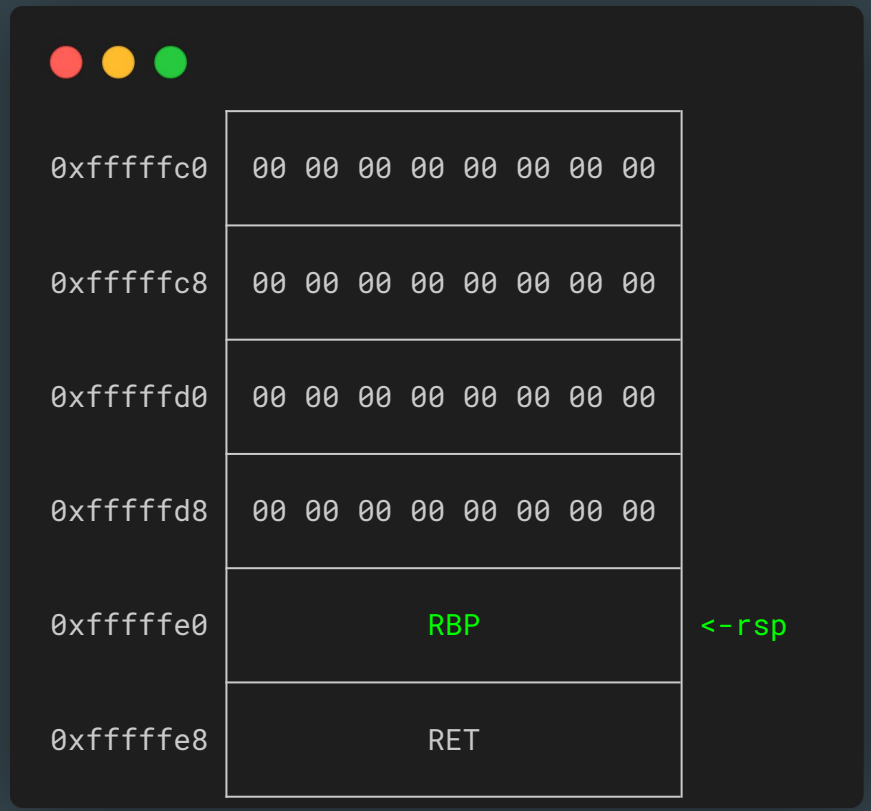
```
main:
 push rbp
 mov rbp,rsp
 sub rsp,0x20
 mov [rbp-0x14],edi              ; argc
 mov [rbp-0x20],rsi              ; argv
 mov [rbp-0x8],0x0               ; debug
 lea rax,[rbp-0x10]              ; buf
 mov rdi,rax
 call 400450 <gets@plt>
 cmp [rbp-0x8],0x0
 je 400583 <main+0x2f> --------
 call 400544 <print_flag>     |
 mov eax,0x0 <-----------------
 leave
 ret
```

~/DCcurity/intro_reversing

```
 main:
-> push rbp
   mov rbp,rsp
   sub rsp,0x20
   mov [rbp-0x14],edi            ; argc
   mov [rbp-0x20],rsi            ; argv
   mov [rbp-0x8],0x0             ; debug
   lea rax,[rbp-0x10]            ; buf
   mov rdi,rax
   call 400450 <gets@plt>
   cmp [rbp-0x8],0x0
   je 400583 <main+0x2f> --------
   call 400544 <print_flag>     |
   mov eax,0x0 <-----------------
   leave
   ret
```

~/DCcurity/intro_reversing

```
 main:
  push rbp
-> mov rbp,rsp
  sub rsp,0x20
  mov [rbp-0x14],edi            ; argc
  mov [rbp-0x20],rsi            ; argv
  mov [rbp-0x8],0x0             ; debug
  lea rax,[rbp-0x10]            ; buf
  mov rdi,rax
  call 400450 <gets@plt>
  cmp [rbp-0x8],0x0
  je 400583 <main+0x2f> --------
  call 400544 <print_flag>     |
  mov eax,0x0 <-----------------
  leave
  ret
```



~/DCcurity/intro_reversing

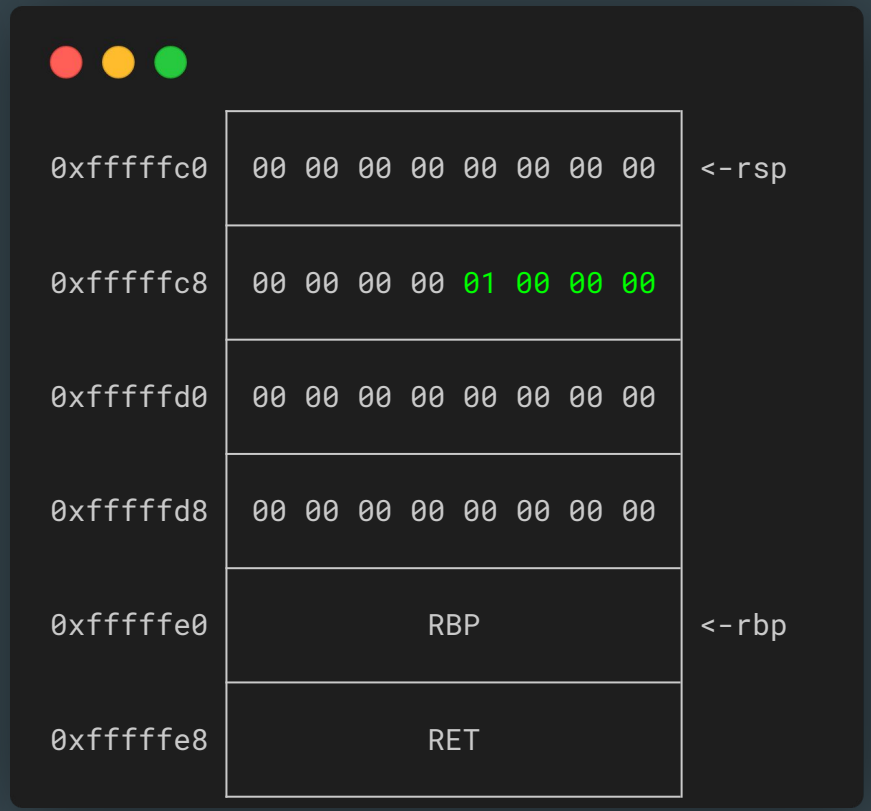
```
 main:
  push rbp
  mov rbp,rsp
-> sub rsp,0x20
  mov [rbp-0x14],edi            ; argc
  mov [rbp-0x20],rsi            ; argv
  mov [rbp-0x8],0x0             ; debug
  lea rax,[rbp-0x10]            ; buf
  mov rdi,rax
  call 400450 <gets@plt>
  cmp [rbp-0x8],0x0
  je 400583 <main+0x2f> --------
  call 400544 <print_flag>     |
  mov eax,0x0 <-----------------
  leave
  ret
```

~/DCcurity/intro_reversing

```
 main:
  push rbp
  mov rbp,rsp
  sub rsp,0x20
-> mov [rbp-0x14],edi            ; argc
  mov [rbp-0x20],rsi             ; argv
  mov [rbp-0x8],0x0              ; debug
  lea rax,[rbp-0x10]             ; buf
  mov rdi,rax
  call 400450 <gets@plt>
  cmp [rbp-0x8],0x0
  je 400583 <main+0x2f> --------
  call 400544 <print_flag>     |
  mov eax,0x0 <-----------------
  leave
  ret
```

~/DCcurity/intro_reversing

```
 main:
  push rbp
  mov rbp,rsp
  sub rsp,0x20
  mov [rbp-0x14],edi            ; argc
-> mov [rbp-0x20],rsi           ; argv
  mov [rbp-0x8],0x0             ; debug
  lea rax,[rbp-0x10]            ; buf
  mov rdi,rax
  call 400450 <gets@plt>
  cmp [rbp-0x8],0x0
  je 400583 <main+0x2f> --------
  call 400544 <print_flag>     |
  mov eax,0x0 <-----------------
  leave
  ret
```

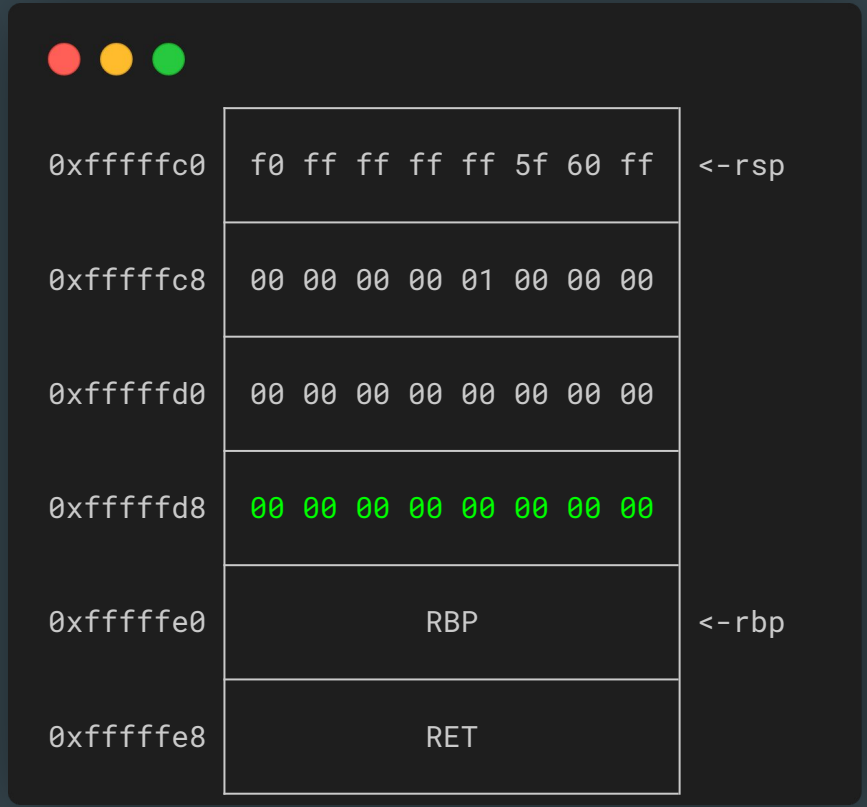


~/DCcurity/intro_reversing

```
 main:
  push rbp
  mov rbp,rsp
  sub rsp,0x20
  mov [rbp-0x14],edi            ; argc
  mov [rbp-0x20],rsi            ; argv
-> mov [rbp-0x8],0x0            ; debug
  lea rax,[rbp-0x10]            ; buf
  mov rdi,rax
  call 400450 <gets@plt>
  cmp [rbp-0x8],0x0
  je 400583 <main+0x2f> --------
  call 400544 <print_flag>     |
  mov eax,0x0 <-----------------
  leave
  ret
```

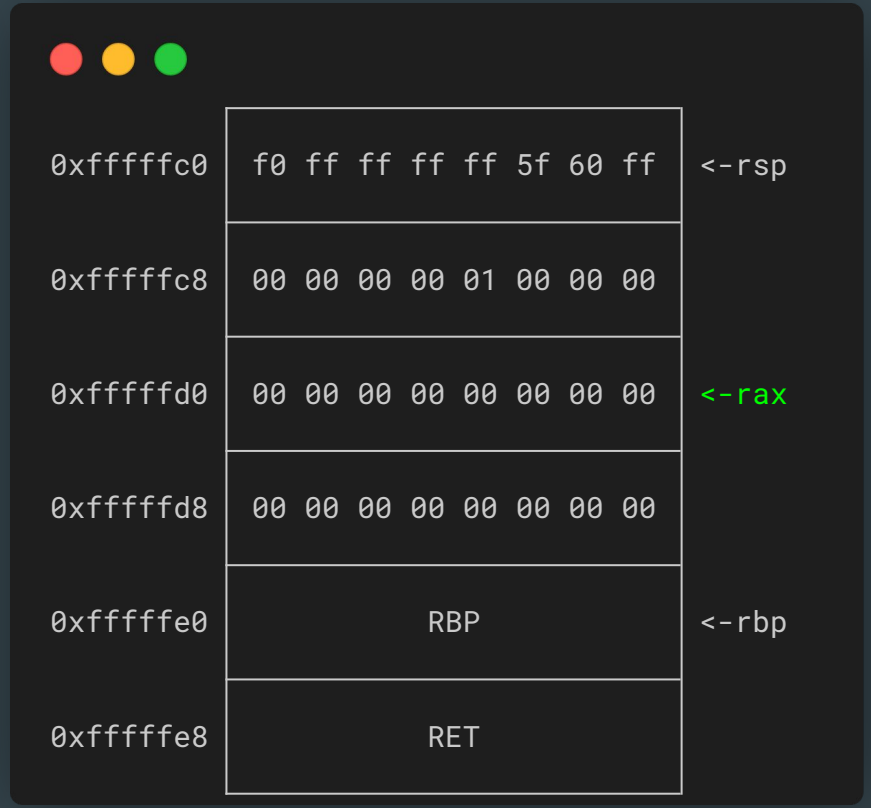
~/DCcurity/intro_reversing

```
 main:
  push rbp
  mov rbp,rsp
  sub rsp,0x20
  mov [rbp-0x14],edi            ; argc
  mov [rbp-0x20],rsi            ; argv
  mov [rbp-0x8],0x0             ; debug
-> lea rax,[rbp-0x10]           ; buf
  mov rdi,rax
  call 400450 <gets@plt>
  cmp [rbp-0x8],0x0
  je 400583 <main+0x2f> --------
  call 400544 <print_flag>     |
  mov eax,0x0 <-----------------
  leave
  ret
```



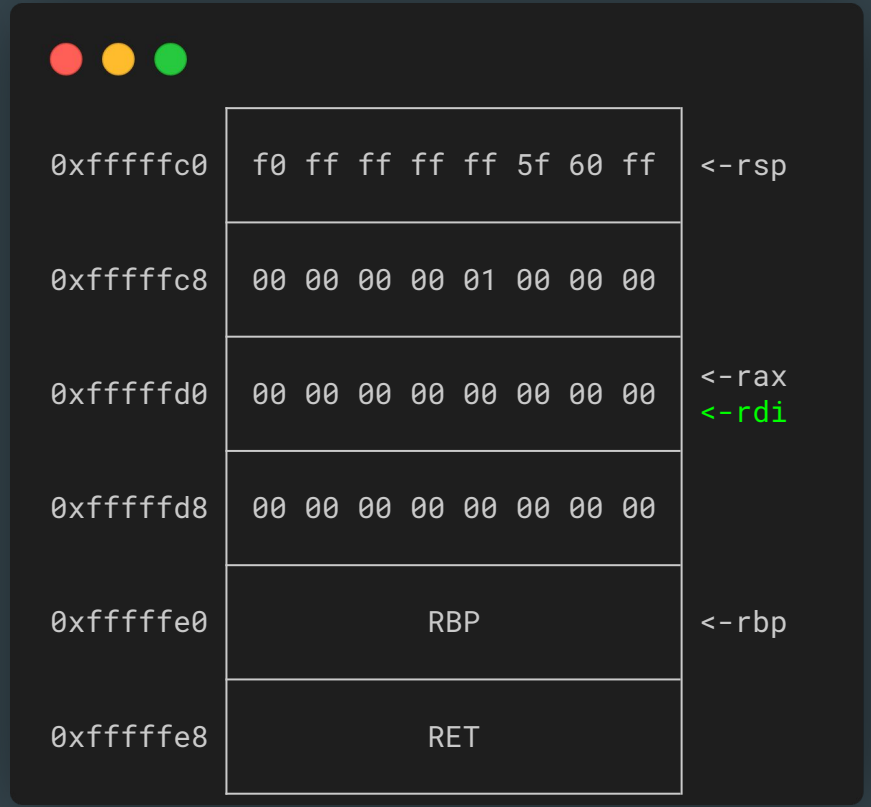
~/DCcurity/intro_reversing

```
 main:
  push rbp
  mov rbp,rsp
  sub rsp,0x20
  mov [rbp-0x14],edi            ; argc
  mov [rbp-0x20],rsi            ; argv
  mov [rbp-0x8],0x0             ; debug
  lea rax,[rbp-0x10]            ; buf
-> mov rdi,rax
  call 400450 <gets@plt>
  cmp [rbp-0x8],0x0
  je 400583 <main+0x2f> --------
  call 400544 <print_flag>     |
  mov eax,0x0 <-----------------
  leave
  ret
```



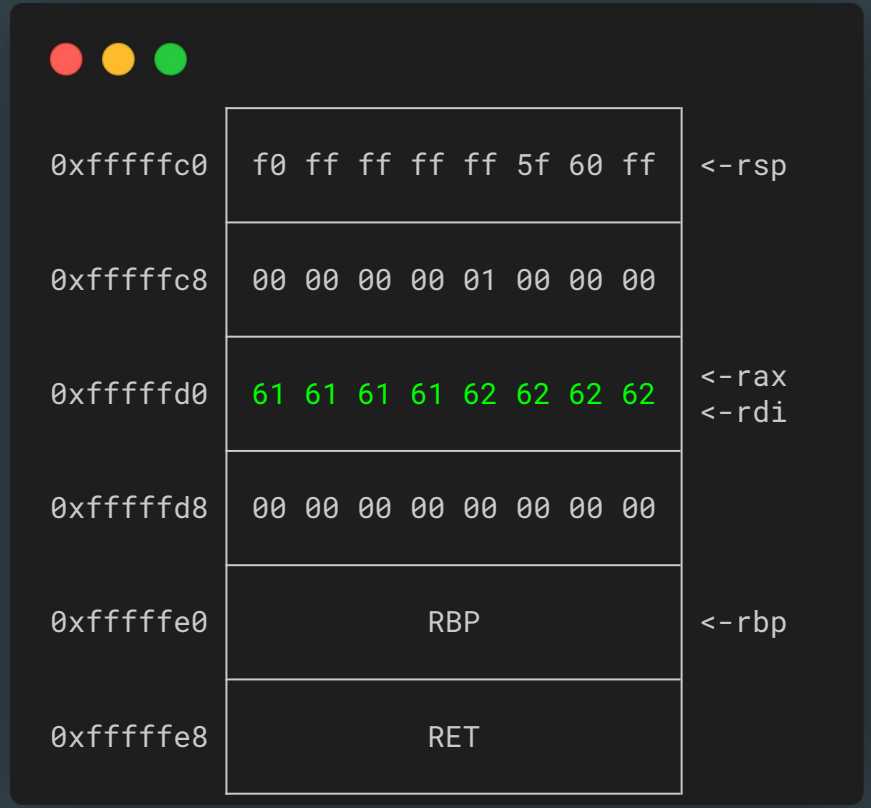
~/DCcurity/intro_reversing

```
 main:
  push rbp
  mov rbp,rsp
  sub rsp,0x20
  mov [rbp-0x14],edi            ; argc
  mov [rbp-0x20],rsi            ; argv
  mov [rbp-0x8],0x0             ; debug
  lea rax,[rbp-0x10]            ; buf
  mov rdi,rax
-> call 400450 <gets@plt>
  cmp [rbp-0x8],0x0
  je 400583 <main+0x2f> --------
  call 400544 <print_flag>     |
  mov eax,0x0 <-----------------
  leave
  ret
```



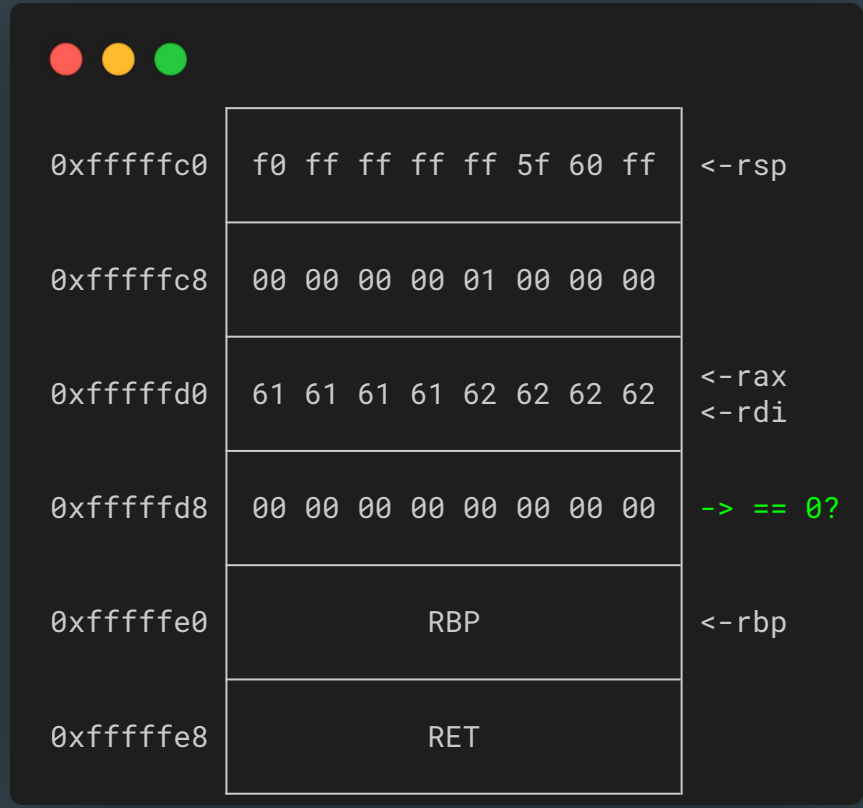
~/DCcurity/intro_reversing

```
 main:
  push rbp
  mov rbp,rsp
  sub rsp,0x20
  mov [rbp-0x14],edi            ; argc
  mov [rbp-0x20],rsi            ; argv
  mov [rbp-0x8],0x0             ; debug
  lea rax,[rbp-0x10]            ; buf
  mov rdi,rax
  call 400450 <gets@plt>
-> cmp [rbp-0x8],0x0
  je 400583 <main+0x2f> --------
  call 400544 <print_flag>     |
  mov eax,0x0 <-----------------
  leave
  ret
```



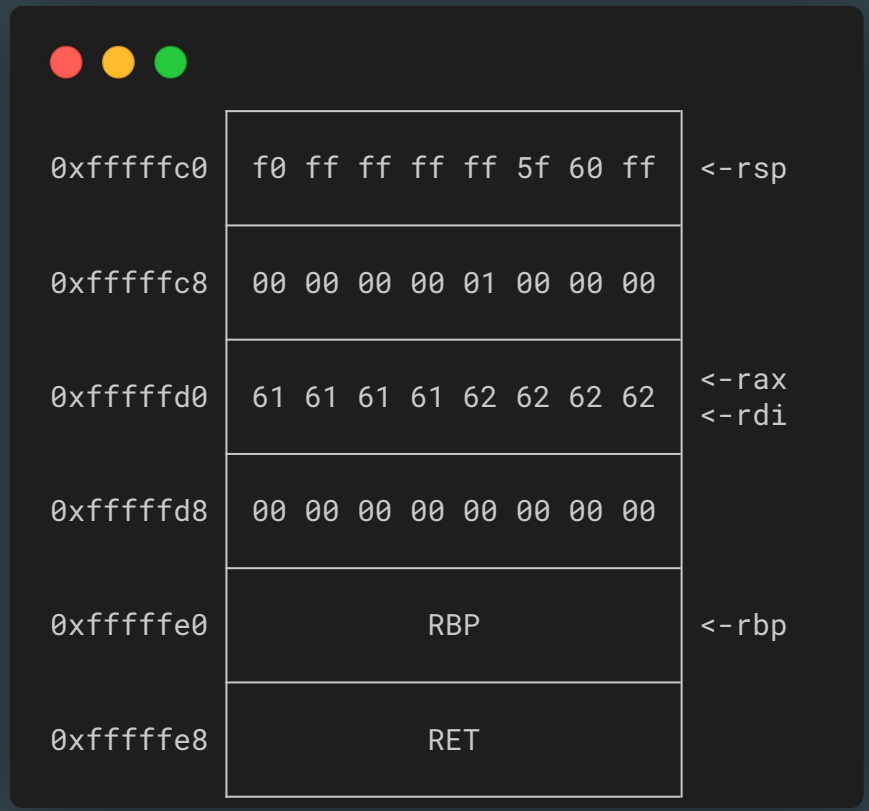
~/DCcurity/intro_reversing

```
main:
  push rbp
  mov rbp,rsp
  sub rsp,0x20
  mov [rbp-0x14],edi            ; argc
  mov [rbp-0x20],rsi            ; argv
  mov [rbp-0x8],0x0             ; debug
  lea rax,[rbp-0x10]            ; buf
  mov rdi,rax
  call 400450 <gets@plt>
  cmp [rbp-0x8],0x0
-> je 400583 <main+0x2f> --------
  call 400544 <print_flag>     |
  mov eax,0x0 <-----------------
  leave
  ret
```



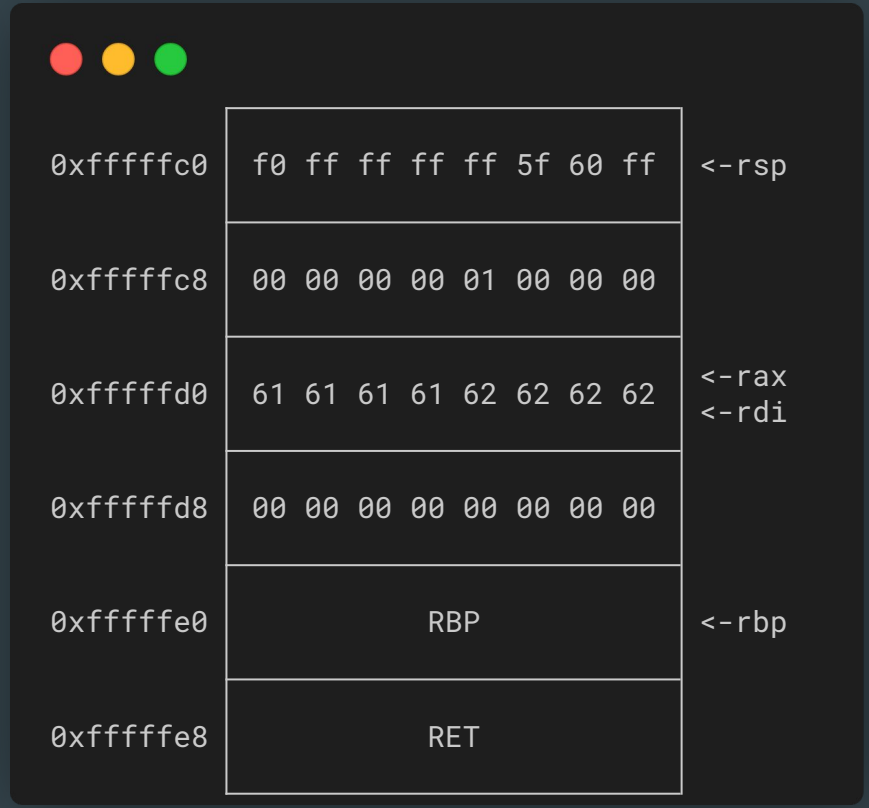
~/DCcurity/intro_reversing

```
main:
  push rbp
  mov rbp,rsp
  sub rsp,0x20
  mov [rbp-0x14],edi            ; argc
  mov [rbp-0x20],rsi            ; argv
  mov [rbp-0x8],0x0             ; debug
  lea rax,[rbp-0x10]            ; buf
  mov rdi,rax
  call 400450 <gets@plt>
  cmp [rbp-0x8],0x0
  je 400583 <main+0x2f> --------
  call 400544 <print_flag>     |
-> mov eax,0x0 <----------------
  leave
  ret
```



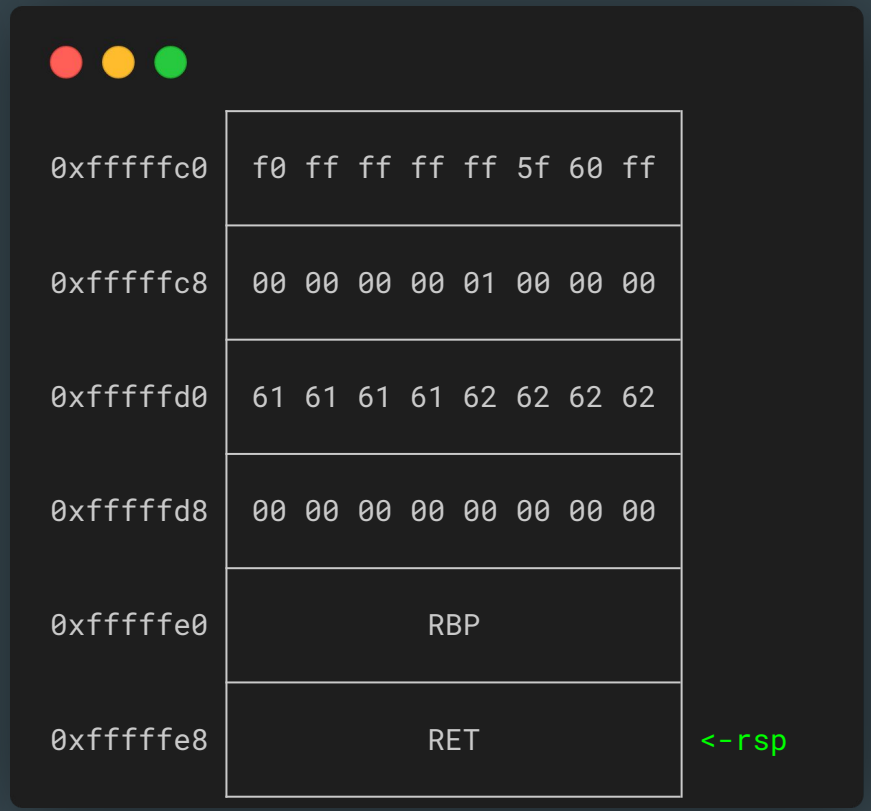
~/DCcurity/intro_reversing

```
 main:
  push rbp
  mov rbp,rsp
  sub rsp,0x20
  mov [rbp-0x14],edi            ; argc
  mov [rbp-0x20],rsi            ; argv
  mov [rbp-0x8],0x0             ; debug
  lea rax,[rbp-0x10]            ; buf
  mov rdi,rax
  call 400450 <gets@plt>
  cmp [rbp-0x8],0x0
  je 400583 <main+0x2f> --------
  call 400544 <print_flag>     |
  mov eax,0x0 <-----------------
-> leave
  ret
```



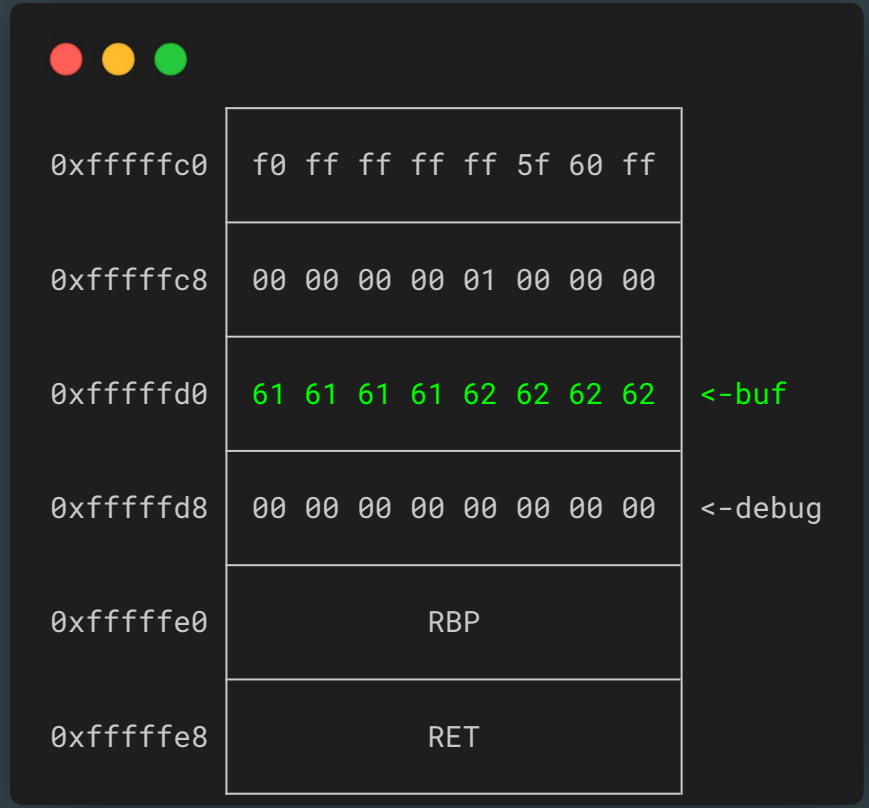
~/DCcurity/intro_reversing

```
main:
 push rbp
 mov rbp,rsp
 sub rsp,0x20
 mov [rbp-0x14],edi            ; argc
 mov [rbp-0x20],rsi            ; argv
 mov [rbp-0x8],0x0             ; debug
 lea rax,[rbp-0x10]            ; buf
 mov rdi,rax
 call 400450 <gets@plt>
 cmp [rbp-0x8],0x0
 je 400583 <main+0x2f> --------
 call 400544 <print_flag>     |
 mov eax,0x0 <-----------------
 leave
-> ret
```



~/DCcurity/intro_reversing

```
int main(int argc, char** argv) {
     long int debug = 0;
     char buf[8];

     gets(buf);     <---- ???

     if (debug) {
        print_flag();
     }
}
```



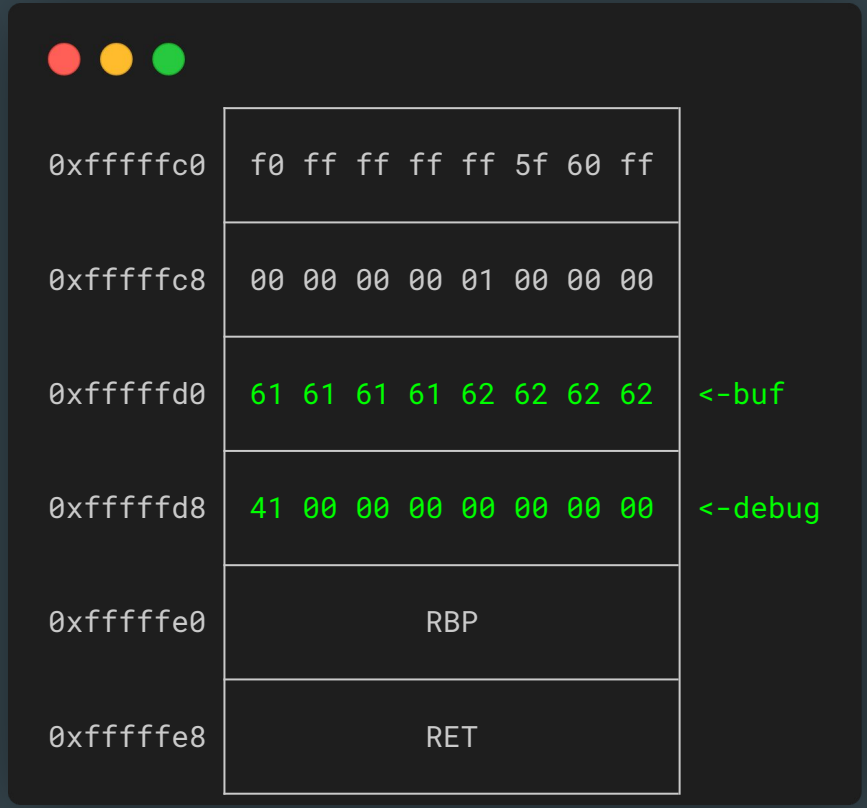
~/DCcurity/intro_reversing

```
$ man gets

NAME
     gets - get a string from standard input (DEPRECATED)

DESCRIPTION
     Never use this function.

     gets()  reads  a  line from stdin into the buffer pointed to by s until either a
     terminating newline or EOF, which it replaces with a null byte ('\0').  No check
     for buffer overrun is performed (see BUGS below).

--Más--
```

~/DCcurity/intro_reversing

```c
int main(int argc, char** argv) {
    long int debug = 0;
    char buf[8];

    gets(buf);

    if (debug) {
        print_flag();
    }
}
```

Buffer overflow!



~/DCcurity/intro_reversing

```c
int main(int argc, char** argv) {
    long int debug = 0;
    char buf[8];

    gets(buf);

    if (debug) {
        print_flag();
    }
}
```

~/DCcurity/intro_reversing

Para los siguientes desafíos

- Endianness
- Convención de llamada
- Dirección de retorno

--Más--