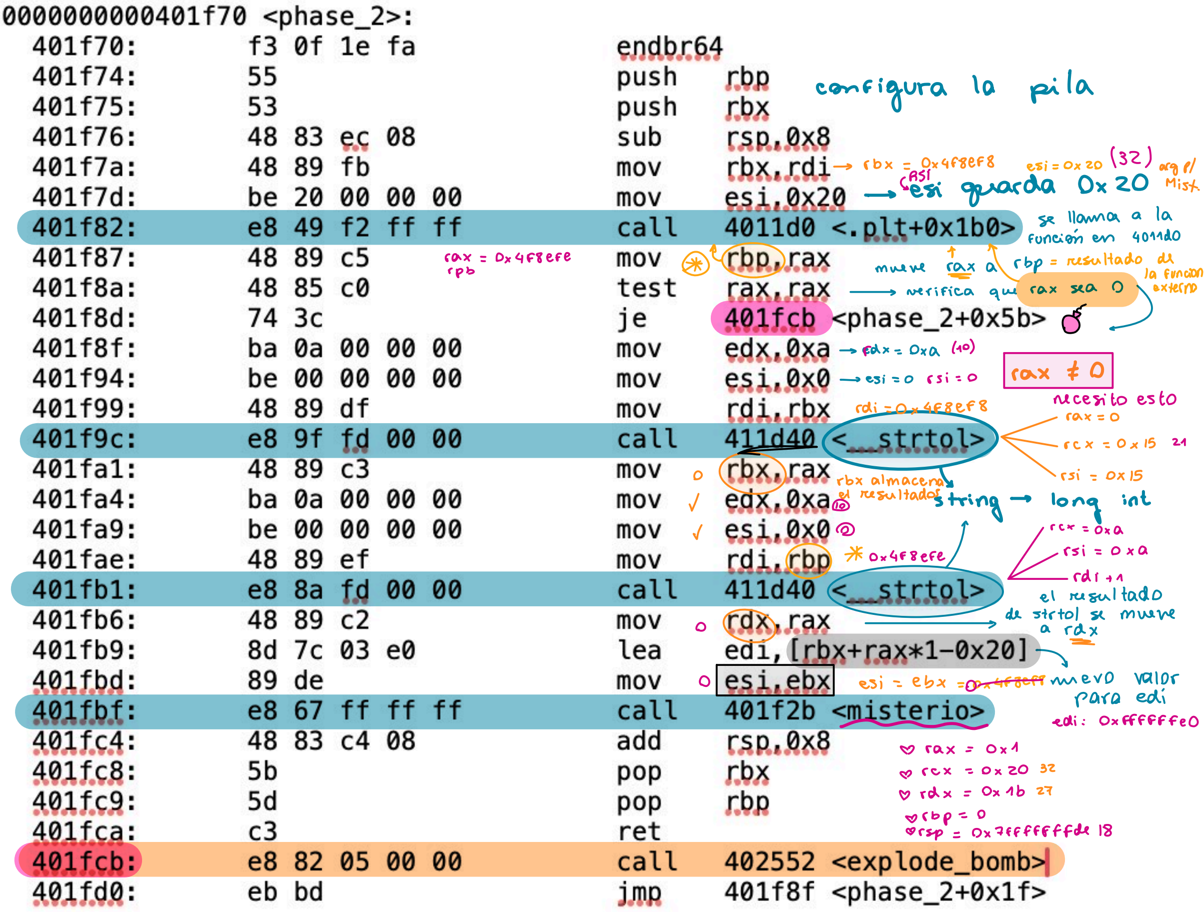


```
0000000000401f70 <phase_2>:
  401f70:       f3 0f 1e fa             endbr64
  401f74:       55                      push   rbp
  401f75:       53                      push   rbx
  401f76:       48 83 ec 08             sub    rsp,0x8
  401f7a:       48 89 fb                mov    rbx,rdi
  401f7d:       be 20 00 00 00          mov    esi,0x20
  401f82:       e8 49 f2 ff ff          call   4011d0 <.plt+0x1b0>
  401f87:       48 89 c5                mov    rbp,rax
  401f8a:       48 85 c0                test   rax,rax
  401f8d:       74 3c                   je     401fcb <phase_2+0x5b>
  401f8f:       ba 0a 00 00 00          mov    edx,0xa
  401f94:       be 00 00 00 00          mov    esi,0x0
  401f99:       48 89 df                mov    rdi,rbx
  401f9c:       e8 9f fd 00 00          call   411d40 <__strtol>
  401fa1:       48 89 c3                mov    rbx,rax
  401fa4:       ba 0a 00 00 00          mov    edx,0xa
  401fa9:       be 00 00 00 00          mov    esi,0x0
  401fae:       48 89 ef                mov    rdi,rbp
  401fb1:       e8 8a fd 00 00          call   411d40 <__strtol>
  401fb6:       48 89 c2                mov    rdx,rax
  401fb9:       8d 7c 03 e0             lea    edi,[rbx+rax*1-0x20]
  401fbd:       89 de                   mov    esi,ebx
  401fbf:       e8 67 ff ff ff          call   401f2b <misterio>
  401fc4:       48 83 c4 08             add    rsp,0x8
  401fc8:       5b                      pop    rbx
  401fc9:       5d                      pop    rbp
  401fca:       c3                      ret
  401fcb:       e8 82 05 00 00          call   402552 <explode_bomb>
  401fd0:       eb bd                   jmp    401f8f <phase_2+0x1f>
```

configura la pila

→ rbx = 0x4f8ef8

esi = 0x20 (32) arg p/ Mist.

→ RSI esi guarda 0x20

se llama a la función en 4011d0

rax = 0x4f8efe rpb

mueve rax a rbp = resultado de la funcion externo

→ verifica que rax sea 0

→ rdx = 0xa (10)

→ esi = 0 rsi = 0

rax ≠ 0

necesito esto

rdi = 0x4f8ef8

rax = 0

rcx = 0x15 21

rsi = 0x15

rbx almacena el resultados

string → long int

0 ✓ (10)

✓ (0)

✳ 0x4f8efe

rcx = 0xa

rsi = 0xa

rdi +1

el resultado de strtol se mueve a rdx

esi = ebx = 0 ~~0x4f8ef8~~

nuevo valor para edi

edi: 0xffffffe0

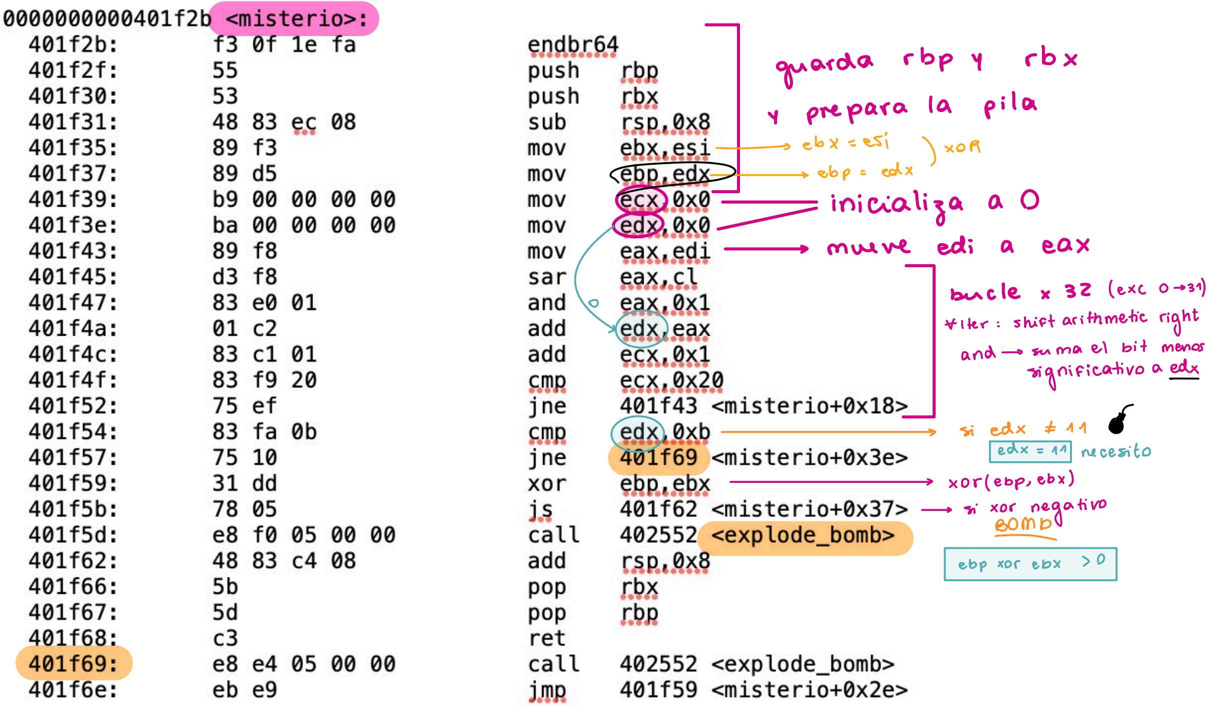
♡ rax = 0x1

♡ rcx = 0x20 32

♡ rdx = 0x1b 27

♡ rbp = 0

♡ rsp = 0x7fffffffde18



```
0000000000401f2b <misterio>:
  401f2b:       f3 0f 1e fa             endbr64
  401f2f:       55                      push   rbp
  401f30:       53                      push   rbx
  401f31:       48 83 ec 08             sub    rsp,0x8
  401f35:       89 f3                   mov    ebx,esi
  401f37:       89 d5                   mov    ebp,edx
  401f39:       b9 00 00 00 00          mov    ecx,0x0
  401f3e:       ba 00 00 00 00          mov    edx,0x0
  401f43:       89 f8                   mov    eax,edi
  401f45:       d3 f8                   sar    eax,cl
  401f47:       83 e0 01                and    eax,0x1
  401f4a:       01 c2                   add    edx,eax
  401f4c:       83 c1 01                add    ecx,0x1
  401f4f:       83 f9 20                cmp    ecx,0x20
  401f52:       75 ef                   jne    401f43 <misterio+0x18>
  401f54:       83 fa 0b                cmp    edx,0xb
  401f57:       75 10                   jne    401f69 <misterio+0x3e>
  401f59:       31 dd                   xor    ebp,ebx
  401f5b:       78 05                   js     401f62 <misterio+0x37>
  401f5d:       e8 f0 05 00 00          call   402552 <explode_bomb>
  401f62:       48 83 c4 08             add    rsp,0x8
  401f66:       5b                      pop    rbx
  401f67:       5d                      pop    rbp
  401f68:       c3                      ret
  401f69:       e8 e4 05 00 00          call   402552 <explode_bomb>
  401f6e:       eb e9                   jmp    401f59 <misterio+0x2e>

0000000000401f70 <phase_2>:
```

guarda rbp y rbx y prepara la pila

ebx = esi
ebp = edx
) xor

inicializa a 0

mueve edi a eax

0

bucle x 32 (exc 0→31)
∀iter: shift arithmetic right
and → suma el bit menos significativo a edx

si edx ≠ 11
edx = 11 necesito

xor(ebp, ebx)

si xor negativo BOMB

ebp xor ebx > 0

?
RAX = 4011d0 <.plt + 0x1b0> —— RAX = 0 —— EXPLOTA

RPB = RAX

EXPLOTA ← edx = 11 ← edi = número que tenga 11 bits de 1 en binario ← edi = rbx + rax - 0x20 (32)

que alguno sea 15 → rbx

con 2047 32

rax = 0
rbx = 2047
rcx = 32 → #1
rdx = 11 → = rbx
rsi = 2097
rdi = 2047
rbp = 0x 7df
rsp = 0x7ffffffdd18

2047 = a + b − 32
15 ↑
2079 = a + b
2079 - 11 = a
2068 = a

b ⊔ a

num 2 (2da)
rax → strtol

num 1 (1era)
rbx = rax → strtol

EXPLOTA —— ebp xor ebx < 0 ← epb ≠ ebx

necesito que rpb = rbx

yo necesito que

♡ [rax + rbx - 32] tiene 11 bits encendidos → rax + rbx = 0......0111111011111111 (4063)

♡ rbp xor rbx > 0

0 1010110000101 1...

rbp → rdx → rax (2)

a xor b = 0....
mismo primer bit

11	10	9	8	7	6	5	4	3	2	1	0	
1	1	1	1	1	1	0	1	1	1	1	1	= a + b - 32 = rdi = 4063 = FDF
2098	1024	512	256	128	64	32	16	8	4	2	1	2047
1	1	1	1	1	1	1	1	1	1	1	1	= a + b = 4095 = FFF

2079

michu -1 2081

pruebo con 3071 y 1024

rax = 0 ?
- rbx = 3071 ✓
rcx = 32 loop
rdx = 11 pasa la condición 1 ✓
- rsi = 3071 ✓
- rdi = 4063 pasa la condición 2 ✓
rbp = FFF = 4095
→ edx → edx - 0x8 → edx = rax

rax :	~~-1~~	~~2080~~	0
rbx :	-1		
rcx :	32		
rdx :	11		
rsi :	0x FFFFFFF		
rdi :	2047		
rbp :			

-1 :	...	1	1	1	1	1	1	1	1	1	1	1	1
2080 :		1	0	0	0	0	0	1	1	0	0	0	0
XOR		0	1	1	1	1	1	0	0	1	1	1	1

992 1087

rax	1080
rbx	-1
rcx	0
rdx	1080
rsi	4....
rdi	1047
rbp	

misterio

11 bits

XOR

NICHU

veamos, $\overset{b}{-1}$ $\overset{a}{2080}$

teoricamente

RDI : a + b - 32 = 2047

a + b = 2079

2079 = $\underbrace{11111111111}$ → rdx = 11

$\underbrace{11111111111}$ = 2047 = 7FF = rdi

→ rdx = 0xb (11 bytes encendidos)

a + b - 0x20 = rdi

a + b = rdi + $\overset{32}{0x20}$ = 2079

-31	2048
-993	3072
-225	2304
-30	2049

-1 : 11 11 11 11 11 11

2080 : 10 00 00 10 00 00

xor : 01 11 11 01 11 11

positivo