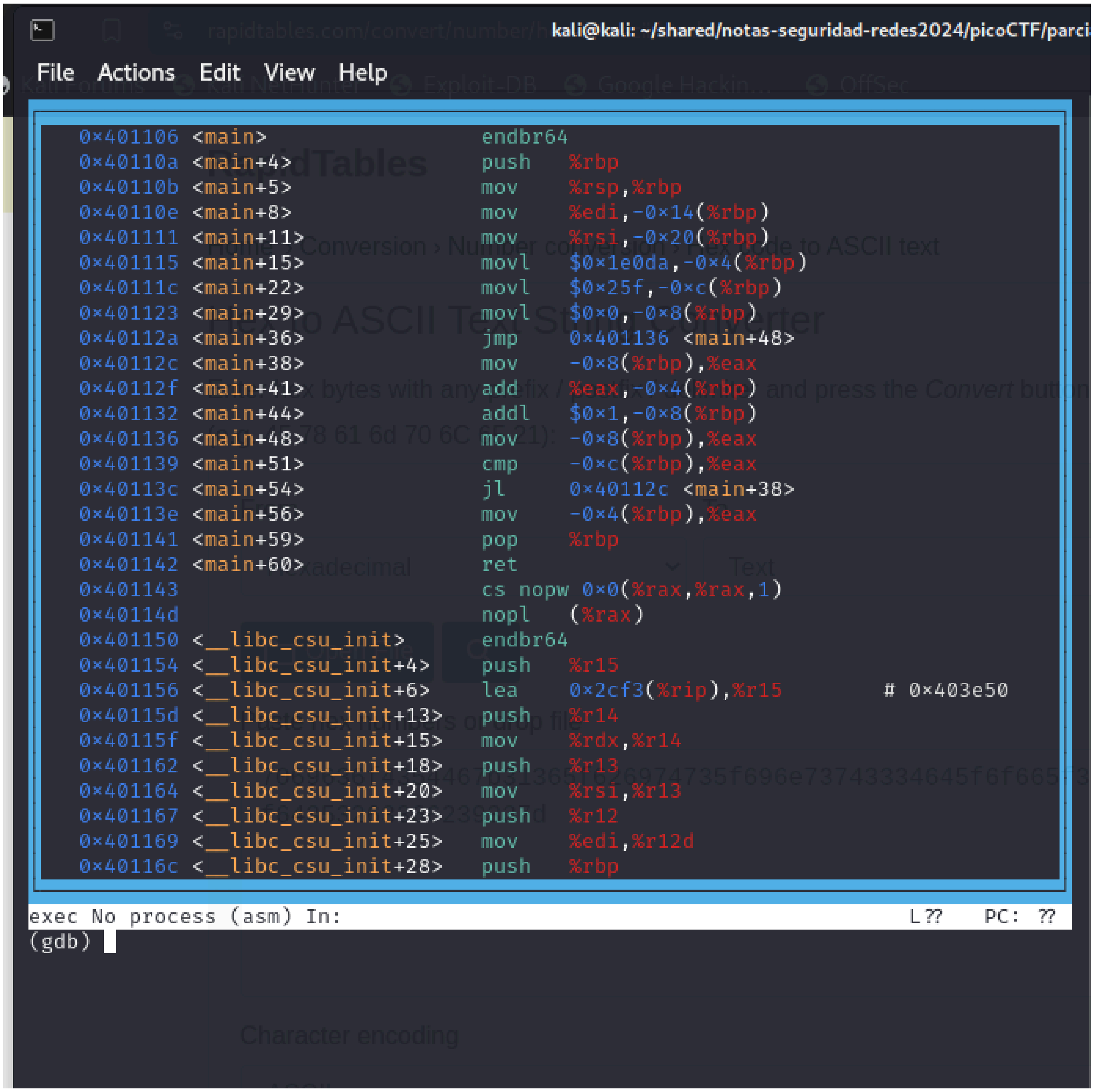
Description

Can you figure out what is in the `eax` register at the end of the `main` function? Put your answer in the picoCTF flag format: `picoCTF{n}` where `n` is the contents of the `eax` register in the decimal number base. If the answer was `0x11` your flag would be `picoCTF{17}`.Debug this.

Hints

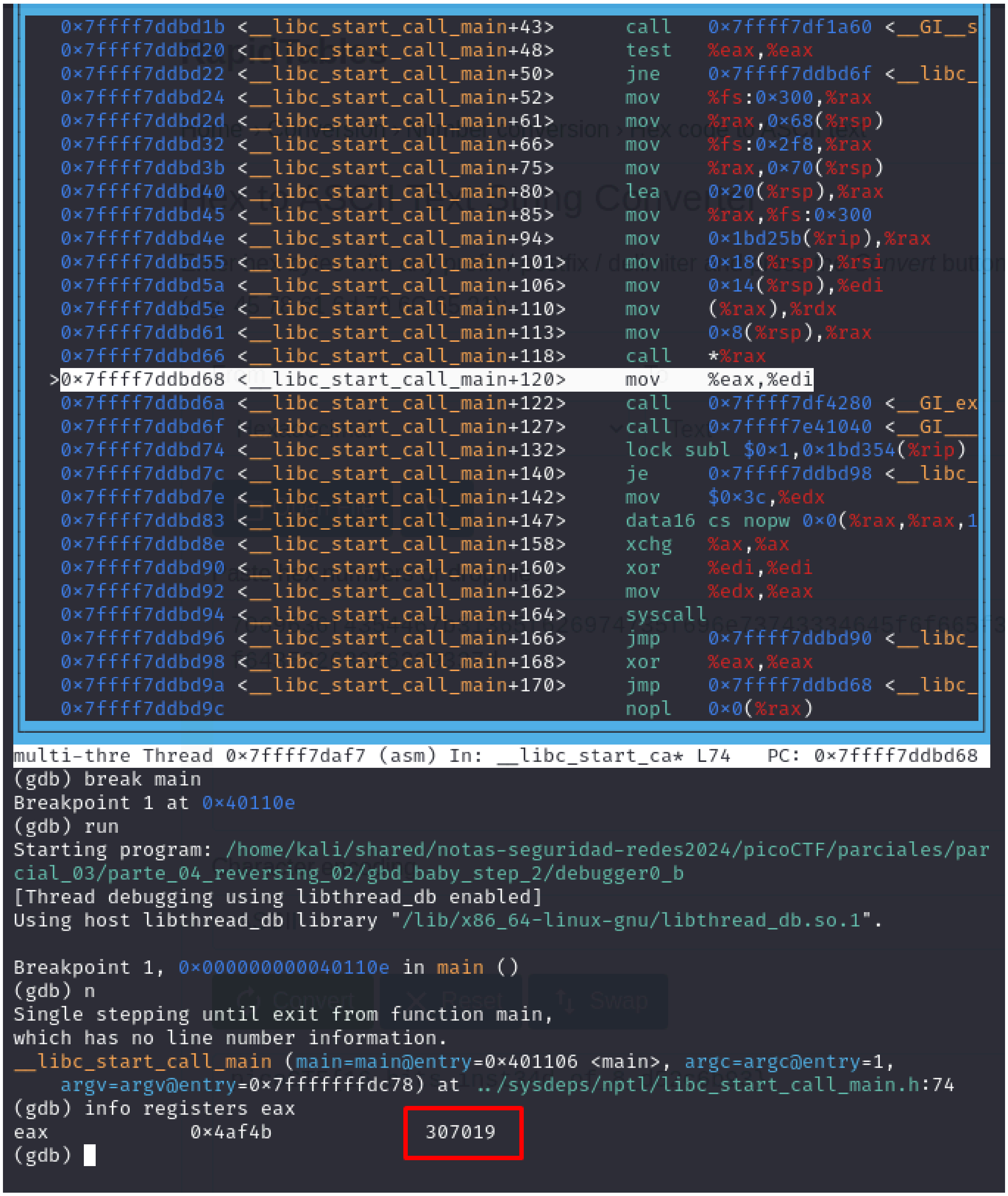
- You could calculate `eax` yourself, or you could set a breakpoint for after the calculcation and inspect `eax` to let the program do the heavy-lifting for you.

Solución



kali@kali: ~/shared/notas-seguridad-redes2024/picoCTF/parci

```
    0x401106 <main>                 endbr64
    0x40110a <main+4>               push   %rbp
    0x40110b <main+5>               mov    %rsp,%rbp
    0x40110e <main+8>               mov    %edi,-0x14(%rbp)
    0x401111 <main+11>              mov    %rsi,-0x20(%rbp)
    0x401115 <main+15>              movl   $0x1e0da,-0x4(%rbp)
    0x40111c <main+22>              movl   $0x25f,-0xc(%rbp)
    0x401123 <main+29>              movl   $0x0,-0x8(%rbp)
    0x40112a <main+36>              jmp    0x401136 <main+48>
    0x40112c <main+38>              mov    -0x8(%rbp),%eax
    0x40112f <main+41>              add    %eax,-0x4(%rbp)
    0x401132 <main+44>              addl   $0x1,-0x8(%rbp)
    0x401136 <main+48>              mov    -0x8(%rbp),%eax
    0x401139 <main+51>              cmp    -0xc(%rbp),%eax
    0x40113c <main+54>              jl     0x40112c <main+38>
    0x40113e <main+56>              mov    -0x4(%rbp),%eax
    0x401141 <main+59>              pop    %rbp
    0x401142 <main+60>              ret
    0x401143                        cs nopw 0x0(%rax,%rax,1)
    0x40114d                        nopl   (%rax)
    0x401150 <__libc_csu_init>      endbr64
    0x401154 <__libc_csu_init+4>    push   %r15
    0x401156 <__libc_csu_init+6>    lea    0x2cf3(%rip),%r15        # 0x403e50
    0x40115d <__libc_csu_init+13>   push   %r14
    0x40115f <__libc_csu_init+15>   mov    %rdx,%r14
    0x401162 <__libc_csu_init+18>   push   %r13
    0x401164 <__libc_csu_init+20>   mov    %rsi,%r13
    0x401167 <__libc_csu_init+23>   push   %r12
    0x401169 <__libc_csu_init+25>   mov    %edi,%r12d
    0x40116c <__libc_csu_init+28>   push   %rbp
exec No process (asm) In:                                       L??   PC: ??
(gdb) 
```



```
   0x7ffff7ddbd1b <__libc_start_call_main+43>     call   0x7ffff7df1a60 <__GI__s
   0x7ffff7ddbd20 <__libc_start_call_main+48>     test   %eax,%eax
   0x7ffff7ddbd22 <__libc_start_call_main+50>     jne    0x7ffff7ddbd6f <__libc_
   0x7ffff7ddbd24 <__libc_start_call_main+52>     mov    %fs:0x300,%rax
   0x7ffff7ddbd2d <__libc_start_call_main+61>     mov    %rax,0x68(%rsp)
   0x7ffff7ddbd32 <__libc_start_call_main+66>     mov    %fs:0x2f8,%rax
   0x7ffff7ddbd3b <__libc_start_call_main+75>     mov    %rax,0x70(%rsp)
   0x7ffff7ddbd40 <__libc_start_call_main+80>     lea    0x20(%rsp),%rax
   0x7ffff7ddbd45 <__libc_start_call_main+85>     mov    %rax,%fs:0x300
   0x7ffff7ddbd4e <__libc_start_call_main+94>     mov    0x1bd25b(%rip),%rax
   0x7ffff7ddbd55 <__libc_start_call_main+101>    mov    0x18(%rsp),%rsi
   0x7ffff7ddbd5a <__libc_start_call_main+106>    mov    0x14(%rsp),%edi
   0x7ffff7ddbd5e <__libc_start_call_main+110>    mov    (%rax),%rdx
   0x7ffff7ddbd61 <__libc_start_call_main+113>    mov    0x8(%rsp),%rax
   0x7ffff7ddbd66 <__libc_start_call_main+118>    call   *%rax
  >0x7ffff7ddbd68 <__libc_start_call_main+120>    mov    %eax,%edi
   0x7ffff7ddbd6a <__libc_start_call_main+122>    call   0x7ffff7df4280 <__GI_ex
   0x7ffff7ddbd6f <__libc_start_call_main+127>    call   0x7ffff7e41040 <__GI___
   0x7ffff7ddbd74 <__libc_start_call_main+132>    lock subl $0x1,0x1bd354(%rip)
   0x7ffff7ddbd7c <__libc_start_call_main+140>    je     0x7ffff7ddbd98 <__libc_
   0x7ffff7ddbd7e <__libc_start_call_main+142>    mov    $0x3c,%edx
   0x7ffff7ddbd83 <__libc_start_call_main+147>    data16 cs nopw 0x0(%rax,%rax,1
   0x7ffff7ddbd8e <__libc_start_call_main+158>    xchg   %ax,%ax
   0x7ffff7ddbd90 <__libc_start_call_main+160>    xor    %edi,%edi
   0x7ffff7ddbd92 <__libc_start_call_main+162>    mov    %edx,%eax
   0x7ffff7ddbd94 <__libc_start_call_main+164>    syscall
   0x7ffff7ddbd96 <__libc_start_call_main+166>    jmp    0x7ffff7ddbd90 <__libc_
   0x7ffff7ddbd98 <__libc_start_call_main+168>    xor    %eax,%eax
   0x7ffff7ddbd9a <__libc_start_call_main+170>    jmp    0x7ffff7ddbd68 <__libc_
   0x7ffff7ddbd9c                                 nopl   0x0(%rax)
multi-thre Thread 0x7ffff7daf7 (asm) In: __libc_start_ca* L74    PC: 0x7ffff7ddbd68
(gdb) break main
Breakpoint 1 at 0x40110e
(gdb) run
Starting program: /home/kali/shared/notas-seguridad-redes2024/picoCTF/parciales/par
cial_03/parte_04_reversing_02/gbd_baby_step_2/debugger0_b
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, 0x000000000040110e in main ()
(gdb) n
Single stepping until exit from function main,
which has no line number information.
__libc_start_call_main (main=main@entry=0x401106 <main>, argc=argc@entry=1,
    argv=argv@entry=0x7fffffffdc78) at ../sysdeps/nptl/libc_start_call_main.h:74
(gdb) info registers eax
eax            0x4af4b             307019
(gdb) 
```

Bandera

```
flag: picoCTF{307019}
```

Notas Adicionales

Referencias

-