Analysing scanf

Load integer variable into rsi – 2nd argument

Load format specifier into rdi – 1st argument

```
 →   0x40060c <main+37>        lea    rax, [rbp-0x4]
     0x400610 <main+41>        mov    rsi, rax
     0x400613 <main+44>        lea    rdi, [rip+0x117]        # 0x400731
     0x40061a <main+51>        mov    eax, 0x0
     0x40061f <main+56>        call   0x4004f0 <__isoc99_scanf@plt>
```

When the format specifier is integer, the value stored in that particular memory location is also in integer as opposed to string

```
gef➤  x/wx $rbp-4
0x7fffffffe41c: 0x00000004
gef➤  |
```

Printing newline on the console

```
 →   0x400624 <main+61>        mov    edi, 0xa
     0x400629 <main+66>        call   0x4004c0 <putchar@plt>
```

Load current count value into rsi register: 2nd argument

Load message to be displayed and format specifier into rdi register: 1st argument

```
 →   0x40062e <main+71>        mov    eax, DWORD PTR [rbp-0x4]
     0x400631 <main+74>        mov    esi, eax
     0x400633 <main+76>        lea    rdi, [rip+0xfa]         # 0x400734
     0x40063a <main+83>        mov    eax, 0x0
     0x40063f <main+88>        call   0x4004e0 <printf@plt>
```

Load current count value into eax, decrement it by 1.

After decrementing it by one, store the results into the same memory location of which is loaded before.

After that, a bitwise and is being done on eax register and if zero flag isnt set, continue looping

```
→   0x400644 <main+93>        mov    eax, DWORD PTR [rbp-0x4]
    0x400647 <main+96>        sub    eax, 0x1
    0x40064a <main+99>        mov    DWORD PTR [rbp-0x4], eax
    0x40064d <main+102>       mov    eax, DWORD PTR [rbp-0x4]
    0x400650 <main+105>       test   eax, eax
    0x400652 <main+107>       jg     0x40062e <main+71>
```

17

Register `eax` will contain **the return code from** `strcmp`, after the call. The `test eax, eax` is the same as `and eax, eax` (bitwise `and`) *except* that it doesn't store the result in `eax`. So `eax` isn't affected by the test, but the zero-flag is, for example.

```
→   0x400652 <main+107>       jg     0x40062e <main+71>        TAKEN [Reason: !Z && S==O]
  ↳    0x40062e <main+71>        mov    eax, DWORD PTR [rbp-0x4]
```

If count is 0, then test eax, eax will set the zero flag

```
→   0x400652 <main+107>       jg     0x40062e <main+71>        NOT taken [Reason: !(!Z && S==O)]
    0x400654 <main+109>       lea    rdi, [rip+0xe5]        # 0x400740
```