

142301013
K Srirama Srikar

MISSION 1

Answer: 50

```
Register group: general
rax      0x1      1
rcx      0x10     16
rsi      0x1      1
rbp      0x7fffffffdd30 0x7fffffffdd30
r8        0x7fffffffdd7e2 140737488345058
r10      0x7ffffb77f00 140737349385984
r12      0x55555555140 93824992235840
r14      0x0       0
rip      0x55555555279 0x55555555279 <mission1+80>
cs       0x33     51
ds       0x0       0
fs       0x0       0
rbx      0x0       0
rdx      0x7ffff7dc9870 140737351817328
rdi      0x0       0
rsp      0x7fffffffdd20 0x7fffffffdd20
r9        0x0       0
r11      0x0       0
r13      0x7fffffffde20 140737488346656
r15      0x0       0
eflags   0x202     [ IF ]
ss       0x2b     43
es       0x0       0
gs       0x0       0

0x55555555274 <mission1+75> callq 0x55555555130 <_isoc99_scanf@plt>
> 0x55555555279 <mission1+80> cmp    $0x1,%eax
0x5555555527c <mission1+83> je     0x555555552a2 <mission1+121>
0x5555555527e <mission1+85> lea    0xdd5(%rip),%rdi    # 0x55555555605a
0x55555555285 <mission1+92> mov    $0x0,%eax
0x5555555528a <mission1+97> callq 0x55555555130 <_isoc99_scanf@plt>
0x5555555528f <mission1+102> lea    0xdca(%rip),%rdi    # 0x555555556060
0x55555555296 <mission1+109> callq 0x5555555550c0 <puts@plt>
0x5555555529b <mission1+114> mov    $0x0,%eax
0x555555552a0 <mission1+119> jmp    0x555555552d8 <mission1+175>
0x555555552a2 <mission1+121> mov    -0xc(%rbp),%eax
0x555555552a5 <mission1+124> cmp    $0x32,%eax
0x555555552a8 <mission1+127> jne    0x555555552c7 <mission1+158>

native process 5306 In: mission1
(gdb) ni
0x000055555555261 in mission1 ()
(gdb) ni
0x000055555555265 in mission1 ()
(gdb) ni
0x000055555555268 in mission1 ()
(gdb) ni
0x00005555555526f in mission1 ()
(gdb) ni
0x000055555555274 in mission1 ()
(gdb) ni
0x000055555555279 in mission1 ()
(gdb) refresh
(gdb) █
```

Here we can see a comparison operation at address <mission+124>, where we are comparing the value in %eax to 0x32 (50). Now, if it is not equal to 50, as we can see it jumps to <mission+158>, and from the image below, we can see that the <mission_done> is at the address <mission+141>, which will only be called if %eax is equal to 50.

rax	0x32	50	rbx	0x0	0	rcx	0x10	16
rdx	0x2dfff7dc9870	450737351817328	rsi	0x1	1	rdi	0x7ffff7b25628	140737349047848
rbp	0x7fffffffdd60	0x7fffffffdd312	rsp	0x5555555592a0	93824992252576	r8	0x0	0 8
r9	0x0	0	r10	0x7ffff7b77f20	140737349385204	r11	0x0	7fe6540 0 354032448
r12	0x55555555140	93824992235840	r13	0x7ffff7b77f20	140737488346656	r14	0x246	582
r15	0x0	0	rip	0x7ffff7a7ebd0	0x7ffff7a7ebd0 <puts>	eflags	0x246	[PF ZF IF]
cs	0x33	51	ss	0x555555552b6	43555555552b6 <mission1+14>	ds	0x002	0 IF]
es	0x0	0	fs	0x0	0	gs	0x0	0

```

B+> 0x555555552a2 <mission1+121> mov    -0xc(%rbp),%eax
0x555555552a5 <mission1+124> cmp    $0x32,%eax
0x555555552a8 <mission1+127> jne    0x555555552c7 <mission1+158>
0x555555552aa <mission1+129> lea    0xdd7(%rip),%rdi    # 0x555555556088
0x555555552b1 <mission1+136> callq  0x5555555550c0 <puts@plt>
> 0x555555552b6 <mission1+141> movl    $0x1,0x2d5c(%rip)    # 0x55555555801c <mission1_done>
0x555555552c0 <mission1+151> movlq  0x71,%eax18a50 <ABS*+0x9f500@plt>
0x555555552c5 <mission1+156> jmp    0x555555552d8 <mission1+175>    ff7dc87c8 <stdout>
0x555555552c7 <mission1+158> lea    %rdea(%rip),%rdi    # 0x5555555560b8
0x555555552ce <mission1+165> callq  0x5555555550c0 <puts@plt>
0x555555552d3 <mission1+170> mov    $0x0,%eax
0x555555552d8 <mission1+175> and    -0x8(%rbp),%rdx
0x555555552dc <mission1+179> xor    %fs:0x28,%rdx

```

```

native process 25779 In: puts                               L??  PC: 0x7ffff7a7ebd0
Single stepping until exit from function puts,
0x000055555555250 in mission1 ()                             555555552b6
(gdb) refresh
(gdb) n
Single stepping until exit from function mission1,
which has no line number information.

Breakpoint 2, 0x00007ffff7a7ebd0 in puts () from /lib64/libc.so.6
(gdb) refresh
(gdb) n
Single stepping until exit from function puts,
which has no line number information.
Tractor beam deactivated! Hoth is safer now.
0x0000555555552b6 in mission1 ()
(gdb) █

```

MISSION 2
Answer:2187

```
rax      0x28      40
rcx      0x88bfff7b25628 218737349047848
rsi      0x10      96
rbp      0x1      1
r8       0x7ffff7fe6540 140737354032448
r10      0x7ffff7d7e4 140737488345060
r12      0x55555555140 93824992235840
r14      0x0       0
rip      0x55555555315 0x55555555315 <mission2+39>
cs       0x7ffff7a7ebd0 517ffff7a7ebd0 <puts>
ds       0x0       0
fs       0x0       0
rbx      0x0       0
rdx      0x7ffff7dc9860 140737351817312
rdi      0x0       7      0      28
rsp      0x555555556198 93824992240024
r9       0x0       18      0      18
r11      0x246      582
r13      0x0       fffffde20 0 737488346656
r15      0x0       0
eflags   0x202      [ IF ]
ss       0x246      43PF ZF IF ]
es       0x0       0
gs       0x0       0

B+> 0x7ffff7a7ebd0 <puts>      endbr64
0x7ffff7a7ebd4 <puts+4>      push %r13 p),%rdi # 0x555555556120
0x7ffff7a7ebd6 <puts+6>      push %r12 puts@plt
> 0x7ffff7a7ebd8 <puts+8>      mov %rdi,%r12 0x555555556148
0x7ffff7a7ebdb <puts+11>     push %rbp
0x7ffff7a7ebdc <puts+12>     push %rbx 550e0 <printf@plt>
B+ 0x7ffff7a7ebdd <puts+13>     sub $0x8,%rsp
0x7ffff7a7ebe1 <puts+17>     callq 0x7ffff7a18a50 <ABS*+0x9f500@plt>
0x7ffff7a7ebe6 <puts+22>     lea 0x349bdb(%rip),%rbp # 0# 0x7ffff7dc87c8 <stdout>
0x7ffff7a7ebed <puts+29>     mov %rax,%rbx
0x7ffff7a7ebf0 <puts+32>     callq 0x0(%rbp),%eax <_isoc99_scanf@plt>
0x7ffff7a7ebf3 <puts+35>     mov %rbp,%rdi
0x7ffff7a7ebf6 <puts+38>     and $0x0,%eax

native process 25779 In: mission2 L?? PC: 0x55555555315
Single stepping until exit from function puts,
Breakpoint 2, 0x00007ffff7a7ebd0 in puts () from /lib64/libc.so.6 7ffff7a7ebd0
(gdb) refresh
(gdb) n
Single stepping until exit from function puts,
which has no line number information.
0x000055555555315 in mission2 ()
(gdb) refresh
(gdb) n
Single stepping until exit from function mission2,
which has no line number information.
Enter the 4-digit override code: 2187
Breakpoint 2, 0x00007ffff7a7ebd0 in puts () from /lib64/libc.so.6
(gdb) █
```

Here similar to the previous question, we enter the code 2187. This is because at the address <mission2+124> we compare \$0x88b(2187) with the value in %eax. And if it is not equal to it, we jump to <mission2+160>, as we can see from the image below, but the <mission2_done> is at the address <mission2+143> and hence the value given should be equal to 2187 to complete the mission2.

```

Register group: general
rax      0x2e      46      rbx      0x0      0
rcx      0x7ffff7b25628  140737349047848  rdx      0x7ffff7dc9860  140737351817312
rsi      0x5555555592a0  93824992252576  rdi      0x0      0
rbp      0x7fffffffdd30  0x7fffffffdd30  rsp      0x7fffffffdd20  0x7fffffffdd20
r8       0x7ffff7fe540  140737354032448  r9       0x0      0
r10      0x7ffff7b77f00  140737349385984  r11      0x246     582
r12      0x55555555140  93824992235840  r13      0x7fffffde20  140737488346656
r14      0x0      0      r15      0x0      0
rip      0x5555555537d  0x5555555537d <mission2+143>  eflags   0x202     [ IF ]
cs       0x33      51      ss       0x2b      43
ds       0x0      0      es       0x0      0
fs       0x0      0      gs       0x0      0

0x55555555367 <mission2+121> mov    -0xc(%rbp),%eax
0x5555555536a <mission2+124> cmp    $0x88b,%eax
0x5555555536f <mission2+129> jne    0x5555555538e <mission2+160>
0x55555555371 <mission2+131> lea    0xe20(%rip),%rdi    # 0x555555556198
0x55555555378 <mission2+138> callq  0x5555555550c0 <puts@plt>
> 0x5555555537d <mission2+143> movl   $0x1,0x2c99(%rip)    # 0x555555558020 <mission2 done>
0x55555555387 <mission2+153> mov    $0x1,%eax
0x5555555538c <mission2+158> jmp    0x555555553b0 <mission2+194>
0x5555555538e <mission2+160> movl   $0xda,-0xc(%rbp)
0x55555555395 <mission2+167> mov    -0xc(%rbp),%eax
0x55555555398 <mission2+170> mov    %eax,%esi
0x5555555539a <mission2+172> lea    0xe27(%rip),%rdi    # 0x5555555561c8
0x555555553a1 <mission2+179> mov    $0x0,%eax

```

native process 25779 In: mission2 L?? PC: 0x5555555537d

```

Mission 2: Bypass the Shield Generator
0x000055555555315 in mission2 ()
(gdb) n
Single stepping until exit from function main,
which has no line number information.

Choose a mission to attempt (1-5, or 0 to exit): 2
(gdb) n
Single stepping until exit from function mission2,
which has no line number information.
Enter the 4-digit override code: 2187

Mission 2: Bypass the Shield Generator
Enter the 4-digit override code: 2187
Shield generator bypassed! The path is clear.

```

MISSION 3

Answer:13

```
Mission 3: Navigate the Asteroid Field
The Millennium Falcon made the Kessel Run in less than 12 parsecs.
Enter the number of parsecs to plot the course: 13
Course plotted successfully in 13 parsecs!
```

```
0x55555555469 <mission3+163>  mov    -0x14(%rbp),%eax
0x5555555546c <mission3+166>  cmp    %eax,-0x10(%rbp)
0x5555555546f <mission3+169>  jne    0x55555555498 <mission3+210>
0x55555555471 <mission3+171>  mov    -0x14(%rbp),%eax
0x55555555474 <mission3+174>  mov    %eax,%esi
0x55555555476 <mission3+176>  lea    0xe5b(%rip),%rdi    # 0x5555555562d8
0x5555555547d <mission3+183>  mov    $0x0,%eax
0x55555555482 <mission3+188>  callq 0x555555550e0 <printf@plt>
0x55555555487 <mission3+193>  movl   $0x1,0x2b93(%rip)    # 0x555555558024 <mission3_done>
```

Here we are comparing the given value at `-0x14(%rbp)` with a value in `-0x10(%rbp)`.

We get the value of `-0x10(%rbp)` from the loop below.

Where we initialize two variables `-0x10(%rbp)` and `-0xc(%rbp)`, and here `-0xc(%rbp)` is similar to the `int i` initialization in a typical for loop. Here the loop runs 13 times and increments `-0x10(%rbp)` by one every time. The loop runs for 13 times because of the `jle` comparison, where we compare `0xc` (12) with `-0xc(%rbp)`. Therefore, we get a value of 13 in `-0x10(%rbp)`.

```
0x5555555543f <mission3+121>    callq 0x5555555550c0 <puts@plt>
0x55555555444 <mission3+126>    mov     $0x0,%eax
0x55555555449 <mission3+131>    jmp     0x555555554a9 <mission3+227>
0x5555555544b <mission3+133>    movl    $0x0,-0x10(%rbp)
0x55555555452 <mission3+140>    movl    $0x0,-0xc(%rbp)
0x55555555459 <mission3+147>    jmp     0x55555555463 <mission3+157>
0x5555555545b <mission3+149>    addl    $0x1,-0x10(%rbp)
0x5555555545f <mission3+153>    addl    $0x1,-0xc(%rbp)
0x55555555463 <mission3+157>    cmpl    $0xc,-0xc(%rbp)
0x55555555467 <mission3+161>    jle     0x5555555545b <mission3+149>
0x55555555469 <mission3+163>    mov     -0x14(%rbp),%eax
0x5555555546c <mission3+166>    cmp     %eax,-0x10(%rbp)
0x5555555546f <mission3+169>    jne     0x55555555498 <mission3+210>
```

The above is the loop which goes from <mission3+133> to <mission3+161>.

MISSION 4

Answer: 5678

```
Mission 4: Launch the Defense Fighters
Enter the launch authorization code: 5678
Launch code accepted! Fighters are away.
```

The answer is 5678.

The important thing to note here is that the function mission4 calls is not verifyLaunchCode but rather verifyLandingCode

```
0x55555555571 <mission4+126> callq 0x5555555554d9 <verifyLandingCode>
0x55555555576 <mission4+131> test  %eax,%eax
0x55555555578 <mission4+133> je   0x555555555597 <mission4+164>
0x5555555557a <mission4+135> lea  0xe4f(%rip),%rdi          # 0x55555555563d0
0x55555555581 <mission4+142> callq 0x5555555550c0 <puts@plt>
0x55555555586 <mission4+147> movl  $0x1,0x2a98(%rip)      # 0x5555555558028 <mission4_done>
0x55555555590 <mission4+157> mov  $0x1,%eax
```

As we can see at the address <mission4+126> we call the function verifyLandingCode and then compare its return value to get the mission done.

```
B+> 0x5555555554d9 <verifyLandingCode> endbr64
    0x5555555554dd <verifyLandingCode+4> push  %rbp
    0x5555555554de <verifyLandingCode+5> mov   %rsp,%rbp
    0x5555555554e1 <verifyLandingCode+8> mov   %edi,-0x4(%rbp)
    0x5555555554e4 <verifyLandingCode+11> cmpl  $0x162e,-0x4(%rbp)
    0x5555555554eb <verifyLandingCode+18> sete  %al
    0x5555555554ee <verifyLandingCode+21> movzbl %al,%eax
    0x5555555554f1 <verifyLandingCode+24> pop   %rbp
    0x5555555554f2 <verifyLandingCode+25> retq
B+ 0x5555555554f3 <mission4>          endbr64
    0x5555555554f7 <mission4+4>      push  %rbp
    0x5555555554f8 <mission4+5>      mov   %rsp,%rbp
    0x5555555554fb <mission4+8>      sub   $0x10,%rsp
```

And in the verify landing code function, we compare it with 0x162e which is equal to 5678 and if it is correct we return the value.

MISSION 5

