



CS 208

F, 6 Feb 2024

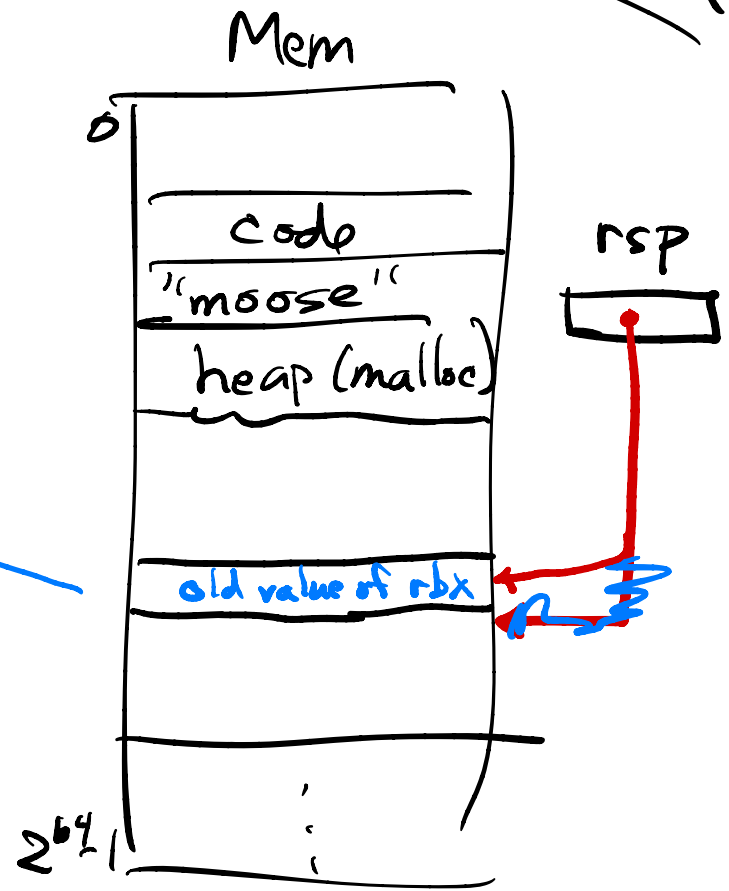
Function 1: "caller-owned local var."

callee pushq %rbx

puzzled

save the old value of rbx

restore old value to rbx  
-> popq %rbx  
ret



function1 ( \_ \*a, \_ \*b )

Answer

testq %rdi, %rdi ) if ~~rdi~~<sup>a</sup> == 0,  
sete %al al = 1, else al = 0

testq %rsi, %rsi ) if ~~rsi~~<sup>b</sup> = 0  
sete %dl dl = 1 else dl = 0

orb %dl, %al )

je L3 ← if !(a == 0 || b == 0)  
jump to L3

testq %rdi, %rdi

Pretend  $rdi = 5$

0	0101
AND 0	0101
<hr/>	
0	0101

$ZF = 0$   
 $SF = 0$

sete %al

if  $ZF = 0$ , set  $al = 0$

else set  $al = 1$

function1 (char \*a, char \*b)

Puzzle 3

fill in the rest of the bytes w/ 0's  
put result into a 32-bit box  
32-bit box

mov zbl (%rsi), %eax


copy

↑  
byte

go to memory where  
rsi/b is pointing  
and get the byte  
you find there

# Puzzle 5

leal <sup>①</sup> (%rcx, %rax), <sup>②</sup> %esi

$\%rcx + \%rax$    
64 bit 32 bit

- ① LEA does computations only on 64-bit #s (thus rex not ecx)
- ② Loads ~~the~~ bottom 32 bits of sum into destination

LEA "load effective address"

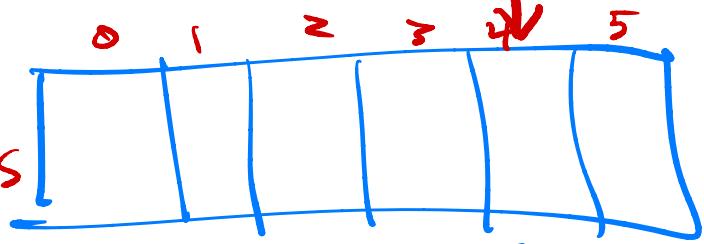
$(\%rax, \%rbx, 4)$

$$rax + rbx * 4$$

$rax$



$nums$



$rbx$  index  
into the  
array

array of  
ints

$$rax + rbx * 4 = \text{address of } \mathit{nums}[4]$$

$0x8(\%rsp, \%rbx, 4)$

$rsp + 8 + 4 * rbx$

$(rsp + 8)$  is where the array  
of ints lives