

Name:

Login:

EXERCISE

Stack memory and instruction-level execution

In the current execution frame...

1. What line of C code is about to be executed? line __ in function _____
2. What is the value in the base pointer? 0x7fffffff___
3. What is the value in the stack pointer? 0x7fffffff___
4. What is the return address? 0x400___ == line ___ in function _____
5. What is address of the return address? 0x7fffffff___

Upon return to the calling execution frame...

6. What line of C code will execute next? line ___
7. What will the value in the base pointer be? 0x7fffffff___
8. What will the value in the stack pointer be? 0x7fffffff___

In the memory dump below...

9. Label all 8 local variables, 2 parameters, 2 return addresses, and 2 saved base pointers. (Assignments/initializations and function calls will help.)

(gdb) info registers

```

[...]
rbp      0x7fffffffde00  0x7fffffffde00
rsp      0x7fffffffde00  0x7fffffffde00
[...]
rip      0x400638 0x400638 <greet_visitor+66>
[...]

```

(gdb) x/80bx \$rsp

```

0x7fffffffde0: 0x41 0x6c 0x65 0x78 0x61 0x6e 0x64 0x65
0x7fffffffde8: 0x72 0x00 0x40 0x00 0x00 0x00 0x00 0x00
0x7fffffffddf0: 0x28 0xdf 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffddf8: 0xaa 0xbb 0xbb 0xaa 0xaa 0xaa 0xaa 0xaa
0x7fffffffde00: 0x30 0xde 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffde08: 0x99 0x06 0x40 0x00 0x00 0x00 0x00 0x00
0x7fffffffde10: 0x18 0xdf 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffde18: 0x90 0x04 0x40 0x00 0x01 0x00 0x00 0x00
0x7fffffffde20: 0x10 0xdf 0xff 0xff 0xff 0x7f 0x00 0x00
0x7fffffffde28: 0x00 0x00 0x00 0x00 0xaa 0xee 0xee 0xaa
0x7fffffffde30: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x7fffffffde38: 0x1d 0xed 0x41 0x2f 0x38 0x00 0x00 0x00

```

(gdb) disas /s main

Dump of assembler code for function main:
boa.c:

```

19  int main(int argc, char *argv[]) {
    0x00000000400679 <+0>:  push %rbp
    0x0000000040067a <+1>:  mov  %rsp,%rbp
    0x0000000040067d <+4>:  sub  $0x20,%rsp
    0x00000000400681 <+8>:  mov  %edi,-0x14(%rbp)

```

```

0x00000000400684 <+11>:  mov  %rsi,-0x20(%rbp)
20  int mn_top = 0xAEEEEEA;
    0x00000000400688 <+15>:  movl $0xaeeeeea,-0x4(%rbp)
21  greet_visitor();
    0x0000000040068f <+22>:  mov  $0x0,%eax
    0x00000000400694 <+27>:  callq 0x4005f6 <greet_visitor>
22  int mn_btm = 0xAFFFFFAA;
    0x00000000400699 <+32>:  movl $0xaaffffaa,-0x8(%rbp)
23  return 0;
    0x000000004006a0 <+39>:  mov  $0x0,%eax
24  }
    0x000000004006a5 <+44>:  leaveq
    0x000000004006a6 <+45>:  retq

```

(gdb) disas /s greet_visitor

Dump of assembler code for function greet_visitor:
boa.c:

```

3  void greet_visitor() {
    0x000000004005f6 <+0>:  push %rbp
    0x000000004005f7 <+1>:  mov  %rsp,%rbp
    0x000000004005fa <+4>:  sub  $0x20,%rsp
4  int gt_top = 0xAaaaaaaa;
    0x000000004005fe <+8>:  movl $0xaaaaaaa,-0x4(%rbp)
5  char name[10]; // BAD!!!
6  printf("Hello. What is your name?\n");
    0x00000000400605 <+15>:  mov  $0x40079c,%edi
    0x0000000040060a <+20>:  callq 0x400450 <puts@plt>
7  gets(name); // VERY, VERY, VERY BAD!!!
    0x0000000040060f <+25>:  lea -0x20(%rbp),%rax
    0x00000000400613 <+29>:  mov  %rax,%rdi
    0x00000000400616 <+32>:  callq 0x400480 <gets@plt>
8  printf("Hello, %s.", name);
    0x0000000040061b <+37>:  lea -0x20(%rbp),%rax
    0x0000000040061f <+41>:  mov  %rax,%rsi
    0x00000000400622 <+44>:  mov  $0x4007b7,%edi
    0x00000000400627 <+49>:  mov  $0x0,%eax
    0x0000000040062c <+54>:  callq 0x400460 <printf@plt>
9  int gt_btm = 0xAABBBBAA;
    0x00000000400631 <+59>:  movl $0xaabbbbaa,-0x8(%rbp)
10 }
    0x00000000400638 <+66>:  nop
    0x00000000400639 <+67>:  leaveq
    0x0000000040063a <+68>:  retq

```

(gdb) disas /s scare_visitor

Dump of assembler code for function scare_visitor:
boa.c:

```

12 void scare_visitor() {
    0x0000000040063b <+0>:  push %rbp
    0x0000000040063c <+1>:  mov  %rsp,%rbp
    0x0000000040063f <+4>:  sub  $0x20,%rsp
13 int sv_top = 0xAACCCCAA;
    0x00000000400643 <+8>:  movl $0xaaccccaa,-0x4(%rbp)
14 char message[10] = "BRAH!!!\n"; // OKAY
    0x0000000040064a <+15>:  movabs $0xa21212148415242,%rax
    0x00000000400654 <+25>:  mov  %rax,-0x20(%rbp)
    0x00000000400658 <+29>:  movw $0x0,-0x18(%rbp)
15 printf(message);
    0x0000000040065e <+35>:  lea -0x20(%rbp),%rax
    0x00000000400662 <+39>:  mov  %rax,%rdi
    0x00000000400665 <+42>:  mov  $0x0,%eax
    0x0000000040066a <+47>:  callq 0x400460 <printf@plt>
16 int sv_btm = 0xAADDDDA;
    0x0000000040066f <+52>:  movl $0xaaddddaa,-0x8(%rbp)
17 }
    0x00000000400676 <+59>:  nop
    0x00000000400677 <+60>:  leaveq
    0x00000000400678 <+61>:  retq

```