1：使用 gdb 定位

```
[root@maple 1]# gdb -q test
Reading symbols from /mnt/data/work/improved/1/test...done.
(gdb) b *main
Breakpoint 1 at 0x80483de: file test.s, line 25.
(gdb) r
Starting program: /mnt/data/work/improved/1/test

Breakpoint 1, main () at test.s:25
25              pushl   %ebp
Missing separate debuginfos, use: debuginfo-install glibc-2.12-1.7.el6.i686
(gdb) x/x $ebp
0xbffff6b8:     0x00000000
(gdb) x/x $eip
0x80483de <main>:       0x83e58955
(gdb) x/32x $esp
0xbffff63c:     0x007aacc6      0x00000001      0xbffff6e4      0xbffff6ec
0xbffff64c:     0xb7ffe3d0      0x08048310      0xffffffff      0x0078cfc4
0xbffff65c:     0x0804822c      0x00000001      0xbffff6a0      0x0077c7c5
0xbffff66c:     0x0078dab0      0xb7ffe6b0      0x0091aff4      0x00000000
0xbffff67c:     0x00000000      0xbffff6b8      0xada67dfb      0xa713ea84
0xbffff68c:     0x00000000      0x00000000      0x00000000      0x00000001
0xbffff69c:     0x08048310      0x00000000      0x00782c60      0x007aabeb
0xbffff6ac:     0x0078cfc4      0x00000001      0x08048310      0x00000000
(gdb)
```

main函数执行前，其进程空间的栈里已经有了相当多的数据。我的系统里此时栈顶指针esp的值是0xbffff63c，栈基址指针ebp的值0xbffff6b8，指令寄存器eip的值是0x80483de正好是下一条马上即将执行的指令，

通过 objdump -D 程序名 > a.s ,反汇编程序

说明：ebp 和 esp 是内存地址，不是代码中地址，而 eip 是 代码地址，程序加载进内存变量在内存上的地址是不固定的，而代码地址是不变的，是编译器设置的。
内存地址 ：0xbffff6b8，这个地址 一共32位，就是cpu的可寻址地址范围
代码地址 ：0x80483de， 这个地址在 反汇编出的汇编代码中是可以找到的

2：release 版本崩溃
在 ubuntu 上 测试，其它平台没试
通过 dmesg 可以找到奔溃的 代码地址，
[371.548936] a[2907]: segfault at 0 ip 00000000004005b6 sp 00007ffe56fb4ea0 error 6 in a[400000+1000]

ip 00000000004005b6 ，这个地址就是 代码地址，在 反汇编出的汇编代码中是可以找到的