```
k_array_test.ko:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <dead_compute01>:
dead_compute01():
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:27
- int *a
- int size

*/

int dead_compute01( int *a, int size){
   0:   41 57                   push   %r15
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:32

   int j=0 ;
   long count=0;

   for (j=0; j<size*3; j++){
   2:   8d 0c 76                lea    (%rsi,%rsi,2),%ecx
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:27
- int *a
- int size

*/

int dead_compute01( int *a, int size){
   5:   41 56                   push   %r14
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:43
        a[i] = i*i-tmp;
            tmp += i;
    }

    // do some reading
    if(j%10 == 0) printk(KERN_INFO "%s:", __FUNCTION__);
   7:   41 be 0a 00 00 00       mov    $0xa,%r14d
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:27
- int *a
- int size

*/

int dead_compute01( int *a, int size){
   d:   41 55                   push   %r13
   f:   41 54                   push   %r12
  11:   49 89 fc                mov    %rdi,%r12
  14:   55                      push   %rbp
  15:   89 f5                   mov    %esi,%ebp
  17:   53                      push   %rbx
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:32

   int j=0 ;
   long count=0;

   for (j=0; j<size*3; j++){
  18:   31 db                   xor    %ebx,%ebx
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:27
- int *a
- int size

*/

int dead_compute01( int *a, int size){
  1a:   48 83 ec 18             sub    $0x18,%rsp
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:32

   int j=0 ;
   long count=0;

   for (j=0; j<size*3; j++){
  1e:   89 4c 24 0c             mov    %ecx,0xc(%rsp)
  22:   e9 9d 00 00 00          jmpq   c4 <dead_compute01+0xc4>
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:38

    int i, tmp=0;

    // first loop to write a[i]s.
    for (i = 0; i<size; i++){
        a[i] = i*i-tmp;
  27:   89 d6                   mov    %edx,%esi
  29:   0f af f2                imul   %edx,%esi
  2c:   29 ce                   sub    %ecx,%esi
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:39
            tmp += i;
  2e:   01 d1                   add    %edx,%ecx
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:38

    int i, tmp=0;

    // first loop to write a[i]s.
    for (i = 0; i<size; i++){
        a[i] = i*i-tmp;
  30:   41 89 34 84             mov    %esi,(%r12,%rax,4)
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:39
            tmp += i;
  34:   48 ff c0                inc    %rax
  37:   eb 04                   jmp    3d <dead_compute01+0x3d>
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:32
int dead_compute01( int *a, int size){

   int j=0 ;
   long count=0;

   for (j=0; j<size*3; j++){
  39:   31 c0                   xor    %eax,%eax
  3b:   31 c9                   xor    %ecx,%ecx
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:37

    int i, tmp=0;

    // first loop to write a[i]s.
    for (i = 0; i<size; i++){
  3d:   39 e8                   cmp    %ebp,%eax
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:39
        a[i] = i*i-tmp;
            tmp += i;
  3f:   89 c2                   mov    %eax,%edx
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:37
   for (j=0; j<size*3; j++){

    int i, tmp=0;

    // first loop to write a[i]s.
    for (i = 0; i<size; i++){
  41:   7c e4                   jl     27 <dead_compute01+0x27>
  43:   45 31 ed                xor    %r13d,%r13d
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:43
        a[i] = i*i-tmp;
            tmp += i;
    }

    // do some reading
    if(j%10 == 0) printk(KERN_INFO "%s:", __FUNCTION__);
  46:   89 d8                   mov    %ebx,%eax
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:37
   for (j=0; j<size*3; j++){

    int i, tmp=0;

    // first loop to write a[i]s.
    for (i = 0; i<size; i++){
  48:   85 ed                   test   %ebp,%ebp
  4a:   44 0f 49 ed             cmovns %ebp,%r13d
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:43
        a[i] = i*i-tmp;
            tmp += i;
    }

    // do some reading
    if(j%10 == 0) printk(KERN_INFO "%s:", __FUNCTION__);
  4e:   99                      cltd
  4f:   41 f7 fe                idiv   %r14d
  52:   85 d2                   test   %edx,%edx
  54:   41 89 d7                mov    %edx,%r15d
  57:   75 15                   jne    6e <dead_compute01+0x6e>
  59:   48 c7 c6 00 00 00 00    mov    $0x0,%rsi
            5c: R_X86_64_32S    .rodata
  60:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
            63: R_X86_64_32S    .rodata.str1.1
  67:   31 c0                   xor    %eax,%eax
  69:   e8 00 00 00 00          callq  6e <dead_compute01+0x6e>
            6a: R_X86_64_PC32   printk+0xfffffffffffffffc
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:44
    printk(KERN_CONT "(j=%d)\t%d",j, a[(i+j)/4]);
  6e:   41 8d 44 1d 00          lea    0x0(%r13,%rbx,1),%eax
  73:   b9 04 00 00 00          mov    $0x4,%ecx
  78:   89 de                   mov    %ebx,%esi
  7a:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
            7d: R_X86_64_32S    .rodata.str1.1+0x7
  81:   99                      cltd
  82:   f7 f9                   idiv   %ecx
  84:   48 98                   cltq
  86:   41 8b 14 84             mov    (%r12,%rax,4),%edx
  8a:   31 c0                   xor    %eax,%eax
  8c:   e8 00 00 00 00          callq  91 <dead_compute01+0x91>
            8d: R_X86_64_PC32   printk+0xfffffffffffffffc
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:45
    if(j%10 == 0) printk(KERN_INFO "\n");
  91:   45 85 ff                test   %r15d,%r15d
  94:   75 0e                   jne    a4 <dead_compute01+0xa4>
  96:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
            99: R_X86_64_32S    .rodata.str1.1+0x14
  9d:   31 c0                   xor    %eax,%eax
  9f:   e8 00 00 00 00          callq  a4 <dead_compute01+0xa4>
            a0: R_X86_64_PC32   printk+0xfffffffffffffffc
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:32
int dead_compute01( int *a, int size){

   int j=0 ;
   long count=0;

   for (j=0; j<size*3; j++){
  a4:   31 c9                   xor    %ecx,%ecx
  a6:   eb 14                   jmp    bc <dead_compute01+0xbc>
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:49
    printk(KERN_CONT "(j=%d)\t%d",j, a[(i+j)/4]);
    if(j%10 == 0) printk(KERN_INFO "\n");
```

```
    // second loop to write a[i]s
    for (i =0 ; i< size; i++){
        a[i] = i*i/2 - 1;
  a8:   0f af c0                imul   %eax,%eax
  ab:   be 02 00 00 00          mov    $0x2,%esi
  b0:   99                      cltd
  b1:   f7 fe                   idiv   %esi
  b3:   ff c8                   dec    %eax
  b5:   41 89 04 8c             mov    %eax,(%r12,%rcx,4)
  b9:   48 ff c1                inc    %rcx
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:48
    if(j%10 == 0) printk(KERN_INFO "%s:", __FUNCTION__);
    printk(KERN_CONT "(j=%d)\t%d",j, a[(i+j)/4]);
    if(j%10 == 0) printk(KERN_INFO "\n");

    // second loop to write a[i]s
    for (i =0 ; i< size; i++){
  bc:   39 e9                   cmp    %ebp,%ecx
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:49
        a[i] = i*i/2 - 1;
  be:   89 c8                   mov    %ecx,%eax
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:48
    if(j%10 == 0) printk(KERN_INFO "%s:", __FUNCTION__);
    printk(KERN_CONT "(j=%d)\t%d",j, a[(i+j)/4]);
    if(j%10 == 0) printk(KERN_INFO "\n");

    // second loop to write a[i]s
    for (i =0 ; i< size; i++){
  c0:   7c e6                   jl     a8 <dead_compute01+0xa8>
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:32
int dead_compute01( int *a, int size){

   int j=0 ;
   long count=0;

   for (j=0; j<size*3; j++){
  c2:   ff c3                   inc    %ebx
  c4:   3b 5c 24 0c             cmp    0xc(%rsp),%ebx
  c8:   0f 8c 6b ff ff ff       jl     39 <dead_compute01+0x39>
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:57

    count += tmp * 2 - tmp / 3;

    }
    return 0;
}
  ce:   48 83 c4 18             add    $0x18,%rsp
  d2:   31 c0                   xor    %eax,%eax
  d4:   5b                      pop    %rbx
  d5:   5d                      pop    %rbp
  d6:   41 5c                   pop    %r12
  d8:   41 5d                   pop    %r13
  da:   41 5e                   pop    %r14
  dc:   41 5f                   pop    %r15
  de:   c3                      retq

00000000000000df <dead_compute02>:
dead_compute02():
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:82
int dead_compute02( int *a, int size){

   int j=0 ;
   long count=0;

   for (j=0; j<size*3; j++){
  df:   44 8d 14 76             lea    (%rsi,%rsi,2),%r10d
  e3:   45 31 c0                xor    %r8d,%r8d
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:94
            tmp += i;
    }

    // second loop to write a[i]s
    for (i =0 ; i< size; i++){
        a[i] = i*i/2 + 1;
  e6:   41 b9 02 00 00 00       mov    $0x2,%r9d
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:82
int dead_compute02( int *a, int size){

   int j=0 ;
   long count=0;

   for (j=0; j<size*3; j++){
  ec:   eb 3b                   jmp    129 <dead_compute02+0x4a>
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:88

    int i, tmp=0;

    // first loop to write a[i]s.
    for (i = 0; i<size; i++){
        a[i] = i*i-tmp;
  ee:   41 89 d3                mov    %edx,%r11d
  f1:   44 0f af da             imul   %edx,%r11d
  f5:   41 29 cb                sub    %ecx,%r11d
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:89
            tmp += i;
  f8:   01 d1                   add    %edx,%ecx
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:88

    int i, tmp=0;

    // first loop to write a[i]s.
    for (i = 0; i<size; i++){
        a[i] = i*i-tmp;
  fa:   44 89 1c 87             mov    %r11d,(%rdi,%rax,4)
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:89
            tmp += i;
  fe:   48 ff c0                inc    %rax
 101:   eb 04                   jmp    107 <dead_compute02+0x28>
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:82
int dead_compute02( int *a, int size){

   int j=0 ;
   long count=0;

   for (j=0; j<size*3; j++){
 103:   31 c0                   xor    %eax,%eax
 105:   31 c9                   xor    %ecx,%ecx
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:87

    int i, tmp=0;

    // first loop to write a[i]s.
    for (i = 0; i<size; i++){
 107:   39 f0                   cmp    %esi,%eax
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:89
        a[i] = i*i-tmp;
            tmp += i;
 109:   89 c2                   mov    %eax,%edx
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:87
   for (j=0; j<size*3; j++){

    int i, tmp=0;

    // first loop to write a[i]s.
    for (i = 0; i<size; i++){
 10b:   7c e1                   jl     ee <dead_compute02+0xf>
 10d:   31 c9                   xor    %ecx,%ecx
 10f:   eb 0f                   jmp    120 <dead_compute02+0x41>
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:94
            tmp += i;
    }

    // second loop to write a[i]s
    for (i =0 ; i< size; i++){
        a[i] = i*i/2 + 1;
 111:   0f af c0                imul   %eax,%eax
 114:   99                      cltd
 115:   41 f7 f9                idiv   %r9d
 118:   ff c0                   inc    %eax
 11a:   89 04 8f                mov    %eax,(%rdi,%rcx,4)
 11d:   48 ff c1                inc    %rcx
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:93
        a[i] = i*i-tmp;
            tmp += i;
    }

    // second loop to write a[i]s
    for (i =0 ; i< size; i++){
 120:   39 f1                   cmp    %esi,%ecx
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:94
        a[i] = i*i/2 + 1;
 122:   89 c8                   mov    %ecx,%eax
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:93
        a[i] = i*i-tmp;
            tmp += i;
    }

    // second loop to write a[i]s
    for (i =0 ; i< size; i++){
 124:   7c eb                   jl     111 <dead_compute02+0x32>
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:82
int dead_compute02( int *a, int size){

   int j=0 ;
   long count=0;

   for (j=0; j<size*3; j++){
 126:   41 ff c0                inc    %r8d
 129:   45 39 d0                cmp    %r10d,%r8d
 12c:   7c d5                   jl     103 <dead_compute02+0x24>
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:102

    count += tmp * 2 - tmp / 3;

    }
    return 0;
}
 12e:   31 c0                   xor    %eax,%eax
 130:   c3                      retq
 131:   00 00                   add    %al,(%rax)
    ...

Disassembly of section .init.text:

0000000000000000 <init_module>:
k_array_test_init():
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:110
MODULE_LICENSE("GPL");
MODULE_AUTHOR("LELE MA");
```

```
MODULE_DESCRIPTION("A sample kernel module with simple deadwrites");

static int __init k_array_test_init(void)
{
   0:   41 56                   push   %r14
kmalloc():
/usr/src/linux-headers-3.2.0-4-common/include/linux/slab_def.h:155
            cachep = malloc_sizes[i].cs_dmacachep;
        else
#endif
            cachep = malloc_sizes[i].cs_cachep;

        ret = kmem_cache_alloc_trace(size, cachep, flags);
   2:   48 8b 35 00 00 00 00    mov    0x0(%rip),%rsi        # 9 <init_module
+0x9>
            5: R_X86_64_PC32    malloc_sizes+0xdc
   9:   ba d0 00 00 00          mov    $0xd0,%edx
   e:   bf 40 1f 00 00          mov    $0x1f40,%edi
k_array_test_init():
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:128

    for (i = size-1; i>=0; i--){
        printk(KERN_CONT "a[%d]=%d\t", i, a[i]);
        printk(KERN_CONT "b[%d]=%d\t", i, b[i]);

        if(i%10==0) printk(KERN_CONT "\n");
  13:   41 be 0a 00 00 00       mov    $0xa,%r14d
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:110
MODULE_LICENSE("GPL");
MODULE_AUTHOR("LELE MA");
MODULE_DESCRIPTION("A sample kernel module with simple deadwrites");

static int __init k_array_test_init(void)
{
  19:   41 55                   push   %r13
  1b:   41 54                   push   %r12
  1d:   55                      push   %rbp
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:122

    printk(KERN_INFO "Starting k_array_test with deadwrites\n");

    dead_compute01( a, size);

    dead_compute02( b, size);
  1e:   31 ed                   xor    %ebp,%ebp
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:110
MODULE_LICENSE("GPL");
MODULE_AUTHOR("LELE MA");
MODULE_DESCRIPTION("A sample kernel module with simple deadwrites");

static int __init k_array_test_init(void)
{
  20:   53                      push   %rbx
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:124

    dead_compute01( a, size);

    dead_compute02( b, size);

    for (i = size-1; i>=0; i--){
  21:   bb cf 07 00 00          mov    $0x7cf,%ebx
kmalloc():
/usr/src/linux-headers-3.2.0-4-common/include/linux/slab_def.h:155
  26:   e8 00 00 00 00          callq  2b <init_module+0x2b>
            27: R_X86_64_PC32   kmem_cache_alloc_trace+0xfffffffffffffffc
  2b:   48 8b 35 00 00 00 00    mov    0x0(%rip),%rsi        # 32 <init_modul
e+0x32>
            2e: R_X86_64_PC32   malloc_sizes+0xdc
  32:   ba d0 00 00 00          mov    $0xd0,%edx
  37:   bf 40 1f 00 00          mov    $0x1f40,%edi
  3c:   49 89 c5                mov    %rax,%r13
  3f:   e8 00 00 00 00          callq  44 <init_module+0x44>
            40: R_X86_64_PC32   kmem_cache_alloc_trace+0xfffffffffffffffc
k_array_test_init():
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:118
    int *a, *b;

    a = kmalloc (size * sizeof(int), GFP_KERNEL);
    b = kmalloc (size * sizeof(int), GFP_KERNEL);

    printk(KERN_INFO "Starting k_array_test with deadwrites\n");
  44:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
            47: R_X86_64_32S    .rodata.str1.1+0x19
kmalloc():
/usr/src/linux-headers-3.2.0-4-common/include/linux/slab_def.h:155
  4b:   49 89 c4                mov    %rax,%r12
k_array_test_init():
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:118
  4e:   31 c0                   xor    %eax,%eax
  50:   e8 00 00 00 00          callq  55 <init_module+0x55>
            51: R_X86_64_PC32   printk+0xfffffffffffffffc
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:120

    dead_compute01( a, size);
  55:   be d0 07 00 00          mov    $0x7d0,%esi
  5a:   4c 89 ef                mov    %r13,%rdi
  5d:   e8 00 00 00 00          callq  62 <init_module+0x62>
            5e: R_X86_64_PC32   dead_compute01+0xfffffffffffffffc
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:122
    dead_compute02( b, size);
  62:   be d0 07 00 00          mov    $0x7d0,%esi
  67:   4c 89 e7                mov    %r12,%rdi
  6a:   e8 00 00 00 00          callq  6f <init_module+0x6f>
            6b: R_X86_64_PC32   dead_compute02+0xfffffffffffffffc
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:125

    for (i = size-1; i>=0; i--){
        printk(KERN_CONT "a[%d]=%d\t", i, a[i]);
  6f:   41 8b 94 2d 3c 1f 00    mov    0x1f3c(%r13,%rbp,1),%edx
  76:   00
  77:   89 de                   mov    %ebx,%esi
  79:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
            7c: R_X86_64_32S    .rodata.str1.1+0x43
  80:   31 c0                   xor    %eax,%eax
  82:   e8 00 00 00 00          callq  87 <init_module+0x87>
            83: R_X86_64_PC32   printk+0xfffffffffffffffc
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:126
        printk(KERN_CONT "b[%d]=%d\t", i, b[i]);
  87:   41 8b 94 2c 3c 1f 00    mov    0x1f3c(%r12,%rbp,1),%edx
  8e:   00
  8f:   31 c0                   xor    %eax,%eax
  91:   89 de                   mov    %ebx,%esi
  93:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
            96: R_X86_64_32S    .rodata.str1.1+0x50
  9a:   e8 00 00 00 00          callq  9f <init_module+0x9f>
            9b: R_X86_64_PC32   printk+0xfffffffffffffffc
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:128

        if(i%10==0) printk(KERN_CONT "\n");
  9f:   89 d8                   mov    %ebx,%eax
  a1:   99                      cltd
  a2:   41 f7 fe                idiv   %r14d
  a5:   85 d2                   test   %edx,%edx
  a7:   75 0e                   jne    b7 <init_module+0xb7>
  a9:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
            ac: R_X86_64_32S    .rodata.str1.1+0x5d
  b0:   31 c0                   xor    %eax,%eax
  b2:   e8 00 00 00 00          callq  b7 <init_module+0xb7>
            b3: R_X86_64_PC32   printk+0xfffffffffffffffc
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:124

    dead_compute01( a, size);

    dead_compute02( b, size);

    for (i = size-1; i>=0; i--){
  b7:   ff cb                   dec    %ebx
  b9:   48 83 ed 04             sub    $0x4,%rbp
  bd:   83 fb ff                cmp    $0xffffffff,%ebx
  c0:   75 ad                   jne    6f <init_module+0x6f>
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:131
        printk(KERN_CONT "b[%d]=%d\t", i, b[i]);

        if(i%10==0) printk(KERN_CONT "\n");
    }

    kfree(a);
  c2:   4c 89 ef                mov    %r13,%rdi
  c5:   e8 00 00 00 00          callq  ca <init_module+0xca>
            c6: R_X86_64_PC32   kfree+0xfffffffffffffffc
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:132
    kfree(b);
  ca:   4c 89 e7                mov    %r12,%rdi
  cd:   e8 00 00 00 00          callq  d2 <init_module+0xd2>
            ce: R_X86_64_PC32   kfree+0xfffffffffffffffc
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:134
    return 0;    // Non-zero return means that the module couldn't be loaded.
}
  d2:   5b                      pop    %rbx
  d3:   5d                      pop    %rbp
  d4:   41 5c                   pop    %r12
  d6:   41 5d                   pop    %r13
  d8:   31 c0                   xor    %eax,%eax
  da:   41 5e                   pop    %r14
  dc:   c3                      retq

Disassembly of section .exit.text:

0000000000000000 <cleanup_module>:
k_array_test_cleanup():
/root/deadwriteBenchmark/ko_dead_array_test/k_array_test.c:138

static void __exit k_array_test_cleanup(void)
{
    printk(KERN_INFO "Goodbye from k_array_test.\n");
   0:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
            3: R_X86_64_32S .rodata.str1.1+0x62
   7:   31 c0                   xor    %eax,%eax
   9:   e9 00 00 00 00          jmpq   e <cleanup_module+0xe>
            a: R_X86_64_PC32    printk+0xfffffffffffffffc
```