用如下code做测试

```c
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>

static int __init hello_start(void)
{
    printk(KERN_INFO "Loading hello module...\n");
    printk(KERN_INFO "Hello world\n");
    return  0;
}

static void __exit hello_end(void)
{
    printk(KERN_INFO "Goodbye Mr.\n");
}

module_init(hello_start);
module_exit(hello_end);
```

---

test-1

> ccflags-y = -O0 -g

查看反汇编结果

```
walterzh@walterzh-Precision-T1650:~/work/x64-module/hello$ objdump -dS hello
.ko

hello.ko:     file format elf64-x86-64


Disassembly of section .init.text:

0000000000000000 <init_module>:
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>

static int __init hello_start(void)
{
   0:   55                      push   %rbp
   1:   48 89 e5                mov    %rsp,%rbp
    printk(KERN_INFO "Loading hello module...\n");
   4:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
   b:   b8 00 00 00 00          mov    $0x0,%eax
  10:   e8 00 00 00 00          callq  15 <init_module+0x15>
    printk(KERN_INFO "Hello world\n");
  15:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
  1c:   b8 00 00 00 00          mov    $0x0,%eax
  21:   e8 00 00 00 00          callq  26 <init_module+0x26>
    return  0;
  26:   b8 00 00 00 00          mov    $0x0,%eax
}
  2b:   5d                      pop    %rbp
  2c:   c3                      retq

Disassembly of section .exit.text:

0000000000000000 <cleanup_module>:

static void __exit hello_end(void)
{
   0:   55                      push   %rbp
   1:   48 89 e5                mov    %rsp,%rbp
    printk(KERN_INFO "Goodbye Mr.\n");
   4:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
   b:   b8 00 00 00 00          mov    $0x0,%eax
  10:   e8 00 00 00 00          callq  15 <cleanup_module+0x15>
}
  15:   5d                      pop    %rbp
  16:   c3                      retq
```

test-2: Remove -g option

> ccflags-y = -O0

```
walterzh@walterzh-Precision-T1650:~/work/x64-module/hello$ objdump -dS hello.ko

hello.ko:     file format elf64-x86-64


Disassembly of section .init.text:

0000000000000000 <init_module>:
   0:   55                      push   %rbp
  25:   48 89 e5 48 c7 c7       add    %bh,0x0(%rax)
  2b:   00                      pop    %rbp
  2c:   00                      retq
  2d:   Address 0x000000000000002d is out of bounds.


Disassembly of section .exit.text:

0000000000000075 <cleanup_module>:
  75:   55                      push   %rbp
  76:   48 89 e5                mov    %rsp,%rbp
  79:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
  80:   b8 00 00 00 00          mov    $0x0,%eax
  85:   e8 00 00 00 00          callq  8a <cleanup_module+0x15>
  8a:   5d                      pop    %rbp
  8b:   c3                      retq
```

虽然objdump加了-S选项，但没有源码对应了，ccflags-y应该是有效的。

---

test-3: add optimization option

> ccflags-y = -O3 -g

```
walterzh@walterzh-Precision-T1650:~/work/x64-module/hello$ objdump -dS hello
.ko

hello.ko:     file format elf64-x86-64


Disassembly of section .init.text:

0000000000000000 <init_module>:
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>

static int __init hello_start(void)
{
   0:   55                      push   %rbp
    printk(KERN_INFO "Loading hello module...\n");
   1:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
   8:   31 c0                   xor    %eax,%eax
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>

static int __init hello_start(void)
{
   a:   48 89 e5                mov    %rsp,%rbp
    printk(KERN_INFO "Loading hello module...\n");
   d:   e8 00 00 00 00          callq  12 <init_module+0x12>
    printk(KERN_INFO "Hello world\n");
  12:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
  19:   31 c0                   xor    %eax,%eax
  1b:   e8 00 00 00 00          callq  20 <init_module+0x20>
    return  0;
}
  20:   31 c0                   xor    %eax,%eax
  22:   5d                      pop    %rbp
  23:   c3                      retq

Disassembly of section .exit.text:

0000000000000000 <cleanup_module>:

static void __exit hello_end(void)
{
   0:   55                      push   %rbp
    printk(KERN_INFO "Goodbye Mr.\n");
   1:   48 c7 c7 00 00 00 00    mov    $0x0,%rdi
   8:   31 c0                   xor    %eax,%eax
    printk(KERN_INFO "Hello world\n");
    return  0;
}

static void __exit hello_end(void)
```

```
{
   a:   48 89 e5                mov    %rsp,%rbp
    printk(KERN_INFO "Goodbye Mr.\n");
   d:   e8 00 00 00 00          callq  12 <cleanup_module+0x12>
}
  12:   5d                      pop    %rbp
  13:   c3                      retq
```

整个反汇编很混乱，应该是优化后的结果。

结论：
ccflags-y设定的编译options是有效的，可以用于去除kernel source中的部分目录的优化option，那样最起码这部分code可以用kgdb调试了。