```
1.
      #include <linux/module.h>
2.
      #include <linux/kernel.h>
3.
      #include <linux/init.h>
4.
5.
      static uint test 1 = 0;
6.
      module_param(test_1, uint, 0644);
7.
8.
      static int __init test_module_init(void)
9.
10.
          int
                  d1, d2;
11.
12.
          d1 = 1234;
13.
          d2 = d1 / test_1;
14.
15.
          return d2;
16.
17.
18.
      static void __exit test_module_exit(void)
19.
20.
      }
21.
22.
      module_init(test_module_init);
23.
      module_exit(test_module_exit);
24.
      MODULE LICENSE("GPL");
```

```
1.
      obj-m := test.o
2.
3.
      ccflags-y += -00 -march=armv7-a
4.
5.
      SRC := $(shell pwd)
6.
7.
      \#ccflags-y := \$(shell echo \$(ccflags-y) | sed -e 's/-0[0123s]//g')
8.
9.
10.
      all:
          echo "test build"
11.
12.
          echo $(KERNEL SRC)
13.
          echo $(ccflags-y)
          $(MAKE) -C $(KERNEL_SRC) M=$(SRC)
14.
15.
      modules install:
          echo "install test"
16.
17.
          echo $(KERNEL_SRC)
18.
          $(MAKE) -C $(KERNEL_SRC) M=$(SRC) modules_install
19.
20.
      clean:
21.
          rm -f *.o *~ core .depend .*.cmd *.ko *.mod.c
          rm -f Module.markers Module.symvers modules.order
22.
23.
          rm -rf .tmp_versions Modules.symvers
```

```
1.
     00000000 <init module>:
2.
3.
     static uint test_1 = 0;
4.
     module_param(test_1, uint, 0644);
5.
     static int    init test module init(void)
6.
7.
8.
       0: e52de004 push {lr} ; (str lr, [sp, #-4]!)
9.
      4: e24dd00c sub sp, sp, #12
10.
       int d1, d2;
11.
       d1 = 1234;
12.
13.
      8: e30034d2 movw r3, #1234 ; 0x4d2
14.
      c: e58d3004 str r3, [sp, #4]
15.
       d2 = d1 / test_1;
16.
      10: e59d2004 ldr r2, [sp, #4]
       14: e3003000 movw r3, #0
17.
18.
       18: e3403000 movt r3, #0
19.
       1c: e5933000 ldr r3, [r3]
       20: e1a00002 mov r0, r2
20.
21.
       24: e1a01003 mov r1, r3
       28: ebfffffe bl 0 <__aeabi_uidiv>
22.
23.
      2c: e1a03000 mov r3, r0
24.
      30: e58d3000 str r3, [sp]
25.
26.
      return d2;
27.
     34: e59d3000 ldr r3, [sp]
    }
28.
       38: e1a00003 mov r0, r3
29.
30.
       3c: e28dd00c add sp, sp, #12
       40: e49df004
                      pop {pc} ; (ldr pc, [sp], #4)
31.
```

ccflags-y += -O1 -march=armv7-a

```
1.
     00000000 <init module>:
2.
3.
     static uint test_1 = 0;
4.
     module_param(test_1, uint, 0644);
5.
     static int    init test module init(void)
6.
7.
8.
        0: e92d4008
                       push \{r3, lr\}
9.
        int d1, d2;
10.
11.
        d1 = 1234;
12.
        d2 = d1 / test_1;
13.
        4: e3003000 movw r3, #0
14.
       8: e3403000 movt r3, #0
        c: e30004d2 movw r0, #1234 ; 0x4d2
15.
16.
       10: e5931000 ldr r1, [r3]
       14: ebfffffe bl 0 <__aeabi_uidiv>
17.
18.
19.
        return d2;
20.
21.
      18: e8bd8008 pop {r3, pc}
```

ccflags-y += -O2 -march=armv7-a

```
1.
     00000000 <init module>:
2.
3.
     static uint test_1 = 0;
     module_param(test_1, uint, 0644);
4.
5.
     static int __init test_module_init(void)
6.
8.
        0: e92d4008 push {r3, lr}
9.
        int d1, d2;
10.
11.
       d1 = 1234;
        d2 = d1 / test_1;
12.
13.
        4: e3003000 movw r3, #0
14.
       8: e3403000 movt r3, #0
15.
                              r0, #1234 ; 0x4d2
       c: e30004d2 movw
       10: e5931000 ldr r1, [r3]
16.
       14: ebfffffe bl 0 <__aeabi_uidiv>
17.
18.
19.
        return d2;
20.
    }
       18: e8bd8008 pop {r3, pc}
21.
```

即使是整数/,kernel也调用了 \_\_aeabi\_uidiv 库函数。

kernel code中对/(除法)的禁止真是彻底。