测试代码

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

static int __init test_module_init(void)
{
    double f1, f2, f3;
    f1 = 1.1;
    f2 = 2.3;
    f3 = f1 / f2;

    return 0;
}
static void __exit test_module_exit(void)
{
}
module_init(test_module_init);
module_exit(test_module_exit);
MODULE_LICENSE("GPL");
```

Makefile

```
obj-m := test.o
ccflags-y = -O0 -march=armv7-a -msoft-float
SRC := $(shell pwd)

all:
    echo "test build"
    echo $(KERNEL_SRC)
    $(MAKE) -C $(KERNEL_SRC) M=$(SRC)

modules_install:
    echo "install test"
    echo $(KERNEL_SRC)
    $(MAKE) -C $(KERNEL_SRC) M=$(SRC) modules_install

clean:
    rm -f *.o *~ core .depend .*.cmd *.ko *.mod.c
    rm -f Module.markers Module.symvers modules.order
    rm -rf .tmp_versions Modules.symvers
```

关键是这里的

> ccflags-y = -O0 -march=armv7-a -msoft-float

生成code如下

```
walterzh@walterzh-Precision-T1650:~/work/victor/module-test/test$ arm-linux-gnueabi-objdump -Sd test.ko

test.ko:     file format elf32-littlearm


Disassembly of section .init.text:

00000000 <init_module>:
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>

static int __init test_module_init(void)
{
   0:   e52de004    push    {lr}        ; (str lr, [sp, #-4]!)
   4:   e24dd01c    sub sp, sp, #28
    double f1, f2, f3;
    f1 = 1.1;
   8:   e309299a    movw    r2, #39322  ; 0x999a
   c:   e3492999    movt    r2, #39321  ; 0x9999
  10:   e3093999    movw    r3, #39321  ; 0x9999
  14:   e3433ff1    movt    r3, #16369  ; 0x3ff1
  18:   e1cd21f0    strd    r2, [sp, #16]
    f2 = 2.3;
  1c:   e3062666    movw    r2, #26214  ; 0x6666
  20:   e3462666    movt    r2, #26214  ; 0x6666
  24:   e3063666    movw    r3, #26214  ; 0x6666
  28:   e3443002    movt    r3, #16386  ; 0x4002
  2c:   e1cd20f8    strd    r2, [sp, #8]
    f3 = f1 / f2;
  30:   e1cd01d0    ldrd    r0, [sp, #16]
  34:   e1cd20d8    ldrd    r2, [sp, #8]
  38:   ebfffffe    bl  0 <__aeabi_ddiv>
  3c:   e1a02000    mov r2, r0
  40:   e1a03001    mov r3, r1
  44:   e1cd20f0    strd    r2, [sp]

    return 0;
  48:   e3a03000    mov r3, #0
}
  4c:   e1a00003    mov r0, r3
  50:   e28dd01c    add sp, sp, #28
  54:   e49df004    pop {pc}        ; (ldr pc, [sp], #4)

Disassembly of section .exit.text:

00000000 <cleanup_module>:

static void __exit test_module_exit(void)
{
}
   0:   e12fff1e    bx  lr
```

.ko是生成了，但肯定是不能工作的，因为

> 38: ebffffe bl 0 <__aeabi_ddiv>
> 这是libgcc实现的模拟除法，但在kernel中肯定是没有的，所以该ko连载入都不行。

> ccflags-y = -O0 -march=armv7-a -mhard-float

build fail

```
arm-poky-linux-gnueabi-gcc: error: -mfloat-abi=soft and -mfloat-abi=hard may not be used together
```

也就是在build out-of-tree module时，kbuild环境已经默认指定了 `-mfloat-abi=soft` ，这样与我们指定的 `-mfloat-abi=hard` 冲突了。