```
1.  int main(void)
2.  {
3.      float f1, f2, f3;
4.
5.      f1 = 1.2;
6.      f2 = 1.3;
7.      f3 = f1 / f2;
8.
9.      return 0;
10. }
```

- `-mfpu=vfp` option

> arm-linux-gnueabi-gcc -g -o test -march=armv7-a -mfpu=vfp test.c
>
> arm-linux-gnueabi-objdump -Sd test

```
1.  0000840c <main>:
2.  int main(void)
3.  {
4.      840c:   e92d4800    push    {fp, lr}
5.      8410:   e28db004    add fp, sp, #4
6.      8414:   e24dd010    sub sp, sp, #16
7.      float f1, f2, f3;
8.
9.      f1 = 1.2;
10.     8418:   e309399a    movw    r3, #39322  ; 0x999a
11.     841c:   e3433f99    movt    r3, #16281  ; 0x3f99
12.     8420:   e50b3010    str r3, [fp, #-16]
13.     f2 = 1.3;
14.     8424:   e3063666    movw    r3, #26214  ; 0x6666
15.     8428:   e3433fa6    movt    r3, #16294  ; 0x3fa6
16.     842c:   e50b300c    str r3, [fp, #-12]
17.     f3 = f1 / f2;
18.     8430:   e51b0010    ldr r0, [fp, #-16]
19.     8434:   e51b100c    ldr r1, [fp, #-12]
20.     8438:   eb00006b    bl  85ec <__aeabi_fdiv>
21.     843c:   e1a03000    mov r3, r0
22.     8440:   e50b3008    str r3, [fp, #-8]
23.
24.     return 0;
25.     8444:   e3a03000    mov r3, #0
26. }
27.     8448:   e1a00003    mov r0, r3
28.     844c:   e24bd004    sub sp, fp, #4
29.     8450:   e8bd8800    pop {fp, pc}
```

这里的 `__aeabi_fdiv` 是c library中的function.

```
000085ec <__aeabi_fdiv>:
    85ec:   e3a0c0ff    mov ip, #255    ; 0xff
    85f0:   e01c2ba0    ands    r2, ip, r0, lsr #23
    85f4:   101c3ba1    andsne  r3, ip, r1, lsr #23
    85f8:   1132000c    teqne   r2, ip
    85fc:   1133000c    teqne   r3, ip
    8600:   0a00003a    beq 86f0 <__aeabi_fdiv+0x104>
    8604:   e0422003    sub r2, r2, r3
    8608:   e020c001    eor ip, r0, r1
    860c:   e1b01481    lsls    r1, r1, #9
    8610:   e1a00480    lsl r0, r0, #9
    8614:   0a00001c    beq 868c <__aeabi_fdiv+0xa0>
    8618:   e3a03201    mov r3, #268435456  ; 0x10000000
    861c:   e1831221    orr r1, r3, r1, lsr #4
    8620:   e1833220    orr r3, r3, r0, lsr #4
    8624:   e20c0102    and r0, ip, #-2147483648    ; 0x80000000
    8628:   e1530001    cmp r3, r1
    862c:   31a03083    lslcc   r3, r3, #1
    8630:   e2a2207d    adc r2, r2, #125    ; 0x7d
    8634:   e3a0c502    mov ip, #8388608    ; 0x800000
    8638:   e1530001    cmp r3, r1
    863c:   20433001    subcs   r3, r3, r1
    8640:   2180000c    orrcs   r0, r0, ip
    8644:   e15300a1    cmp r3, r1, lsr #1
    8648:   204330a1    subcs   r3, r3, r1, lsr #1
    864c:   218000ac    orrcs   r0, r0, ip, lsr #1
    8650:   e1530121    cmp r3, r1, lsr #2
    8654:   20433121    subcs   r3, r3, r1, lsr #2
    8658:   2180012c    orrcs   r0, r0, ip, lsr #2
    865c:   e15301a1    cmp r3, r1, lsr #3
    8660:   204331a1    subcs   r3, r3, r1, lsr #3
    8664:   218001ac    orrcs   r0, r0, ip, lsr #3
    8668:   e1b03203    lsls    r3, r3, #4
    866c:   11b0c22c    lsrsne  ip, ip, #4
    8670:   1affffe0    bne 8638 <__aeabi_fdiv+0x4c>
    8674:   e35200fd    cmp r2, #253    ; 0xfd
    8678:   8afffe9d    bhi 84f4 <__aeabi_fmul+0xa0>
    867c:   e1530001    cmp r3, r1
    8680:   e0a00b82    adc r0, r0, r2, lsl #23
    8684:   03c00001    biceq   r0, r0, #1
    8688:   e12fff1e    bx  lr
    868c:   e20cc102    and ip, ip, #-2147483648    ; 0x80000000
    8690:   e18c04a0    orr r0, ip, r0, lsr #9
    8694:   e292207f    adds    r2, r2, #127    ; 0x7f
    8698:   c27230ff    rsbsgt  r3, r2, #255    ; 0xff
    869c:   c1800b82    orrgt   r0, r0, r2, lsl #23
    86a0:   c12fff1e    bxgt    lr
    86a4:   e3800502    orr r0, r0, #8388608    ; 0x800000
    86a8:   e3a03000    mov r3, #0
    86ac:   e2522001    subs    r2, r2, #1
    86b0:   eafffe8f    b   84f4 <__aeabi_fmul+0xa0>
    86b4:   e3320000    teq r2, #0
    86b8:   e200c102    and ip, r0, #-2147483648    ; 0x80000000
```

```
54.      86bc:   01a00080    lsleq   r0, r0, #1
55.      86c0:   03100502    tsteq   r0, #8388608    ; 0x800000
56.      86c4:   02422001    subeq   r2, r2, #1
57.      86c8:   0afffffb    beq 86bc <__aeabi_fdiv+0xd0>
58.      86cc:   e180000c    orr r0, r0, ip
59.      86d0:   e3330000    teq r3, #0
60.      86d4:   e201c102    and ip, r1, #-2147483648    ; 0x80000000
61.      86d8:   01a01081    lsleq   r1, r1, #1
62.      86dc:   03110502    tsteq   r1, #8388608    ; 0x800000
63.      86e0:   02433001    subeq   r3, r3, #1
64.      86e4:   0afffffb    beq 86d8 <__aeabi_fdiv+0xec>
65.      86e8:   e181100c    orr r1, r1, ip
66.      86ec:   eafffffc4   b   8604 <__aeabi_fdiv+0x18>
67.      86f0:   e00c3ba1    and r3, ip, r1, lsr #23
68.      86f4:   e132000c    teq r2, ip
69.      86f8:   1a000005    bne 8714 <__aeabi_fdiv+0x128>
70.      86fc:   e1b02480    lsls    r2, r0, #9
71.      8700:   1afffffb6   bne 85e0 <__aeabi_fmul+0x18c>
72.      8704:   e133000c    teq r3, ip
73.      8708:   1afffffaf   bne 85cc <__aeabi_fmul+0x178>
74.      870c:   e1a00001    mov r0, r1
75.      8710:   eafffffb2   b   85e0 <__aeabi_fmul+0x18c>
76.      8714:   e133000c    teq r3, ip
77.      8718:   1a000003    bne 872c <__aeabi_fdiv+0x140>
78.      871c:   e1b03481    lsls    r3, r1, #9
79.      8720:   0afffff97   beq 8584 <__aeabi_fmul+0x130>
80.      8724:   e1a00001    mov r0, r1
81.      8728:   eafffffac   b   85e0 <__aeabi_fmul+0x18c>
82.      872c:   e3d0c102    bics    ip, r0, #-2147483648    ; 0x80000000
83.      8730:   13d1c102    bicsne  ip, r1, #-2147483648    ; 0x80000000
84.      8734:   1afffffde   bne 86b4 <__aeabi_fdiv+0xc8>
85.      8738:   e3d02102    bics    r2, r0, #-2147483648    ; 0x80000000
86.      873c:   1afffffa2   bne 85cc <__aeabi_fmul+0x178>
87.      8740:   e3d13102    bics    r3, r1, #-2147483648    ; 0x80000000
88.      8744:   1afffff8e   bne 8584 <__aeabi_fmul+0x130>
89.      8748:   eafffffa4   b   85e0 <__aeabi_fmul+0x18c>
```

即默认情况下，gcc只用到模拟浮点运算的库函数，而不是产生VFP instruction。

`-mfpu=vfp -mfloat-abi=softfp` options

> arm-linux-gnueabi-gcc -g -o test2 -march=armv7-a -mfpu=vfp -mfloat-abi=softfp test.c
>
> arm-linux-gnueabi-objdump -Sd test2

```
0000840c <main>:
int main(void)
{
    840c:   e52db004    push    {fp}        ; (str fp, [sp, #-4]!)
    8410:   e28db000    add fp, sp, #0
    8414:   e24dd014    sub sp, sp, #20
    float f1, f2, f3;

    f1 = 1.2;
    8418:   e309399a    movw    r3, #39322  ; 0x999a
    841c:   e3433f99    movt    r3, #16281  ; 0x3f99
    8420:   e50b3010    str r3, [fp, #-16]
    f2 = 1.3;
    8424:   e3063666    movw    r3, #26214  ; 0x6666
    8428:   e3433fa6    movt    r3, #16294  ; 0x3fa6
    842c:   e50b300c    str r3, [fp, #-12]
    f3 = f1 / f2;
    8430:   ed1b7a04    vldr    s14, [fp, #-16]
    8434:   ed5b7a03    vldr    s15, [fp, #-12]
    8438:   eec77a27    vdiv.f32    s15, s14, s15
    843c:   ed4b7a02    vstr    s15, [fp, #-8]

    return 0;
    8440:   e3a03000    mov r3, #0
}
    8444:   e1a00003    mov r0, r3
    8448:   e28bd000    add sp, fp, #0
    844c:   e8bd0800    ldmfd   sp!, {fp}
    8450:   e12fff1e    bx  lr
```

这里 `f3 = f1 / f2` generate VFP instruction.

```
    f3 = f1 / f2;
    8430:   ed1b7a04    vldr    s14, [fp, #-16]
    8434:   ed5b7a03    vldr    s15, [fp, #-12]
    8438:   eec77a27    vdiv.f32    s15, s14, s15
    843c:   ed4b7a02    vstr    s15, [fp, #-8]
```

`-mfpu=vfpv3 -mfloat-abi=hard` options

> arm-linux-gnueabi-gcc -g -o test3 -march=armv7-a -mfpu=vfpv3 -mfloat-abi=hard test.c

```
/usr/lib/gcc-cross/arm-linux-gnueabi/4.7/../../../../arm-linux-gnueabi/bin/ld: error: /tmp/ccj5hP8J.o uses VFP register arguments, test3 does not
/usr/lib/gcc-cross/arm-linux-gnueabi/4.7/../../../../arm-linux-gnueabi/bin/ld: failed to merge target specific data of file /tmp/ccj5hP8J.o
collect2: 错误： ld 返回 1
```

compile成功了，但link失败了，???

可能是lib库使用的是 `-mfloat-abi=soft` 的调用接口,这样生成的test.c的obj在与库链接使

不兼容了。
-mfloat-abi=hard生成的代码采用硬浮点(FPU)调用接口。这样要求所有库和应用程序必须采用这同一个参数来编译，
否则连接时会出现接口不兼容错误。

> arm-linux-gnueabi-gcc -v
> 其中可看到 `--with-float=soft` ,是不是原因?

只编译不链接

> arm-linux-gnueabi-gcc -c -g -o test3.o -march=armv7-a -mfpu=vfpv3 -mfloat-abi=hard test.c
> arm-linux-gnueabi-objdump -Sd test3.o

```
00000000 <main>:
int main(void)
{
   0:   e52db004    push    {fp}        ; (str fp, [sp, #-4]!)
   4:   e28db000    add fp, sp, #0
   8:   e24dd014    sub sp, sp, #20
    float f1, f2, f3;

    f1 = 1.2;
   c:   eddf7a0b    vldr    s15, [pc, #44]  ; 40 <main+0x40>
  10:   ed4b7a04    vstr    s15, [fp, #-16]
    f2 = 1.3;
  14:   eddf7a0a    vldr    s15, [pc, #40]  ; 44 <main+0x44>
  18:   ed4b7a03    vstr    s15, [fp, #-12]
    f3 = f1 / f2;
  1c:   ed1b7a04    vldr    s14, [fp, #-16]
  20:   ed5b7a03    vldr    s15, [fp, #-12]
  24:   eec77a27    vdiv.f32    s15, s14, s15
  28:   ed4b7a02    vstr    s15, [fp, #-8]

    return 0;
  2c:   e3a03000    mov r3, #0
}
  30:   e1a00003    mov r0, r3
  34:   e28bd000    add sp, fp, #0
  38:   e8bd0800    ldmfd   sp!, {fp}
  3c:   e12fff1e    bx  lr
  40:   3f99999a    .word   0x3f99999a
  44:   3fa66666    .word   0x3fa66666
```

`-mfloat-abi=hard` option比 `-mfloat-abi=softfp` 更加的**VFP**化。

`f1 = 1.2;`
`f2 = 1.3;`
这两行code也**VFP**化了。

`-mfloat-abi=hard` generate the following instruction

```
    f1 = 1.2;
 c:   eddf7a0b    vldr    s15, [pc, #44]  ; 40 <main+0x40>
10:   ed4b7a04    vstr    s15, [fp, #-16]
```

`-mfloat-abi=softfp` generate the following instruction

```
    f1 = 1.2;
    8418:   e309399a    movw    r3, #39322  ; 0x999a
    841c:   e3433f99    movt    r3, #16281  ; 0x3f99
    8420:   e50b3010    str r3, [fp, #-16]
```

不指定任何 `vfp` 相关options

> arm-linux-gnueabi-gcc -c -g -o test4.o -march=armv7-a test.c
>
> arm-linux-gnueabi-objdump -Sd test4.o

```
00000000 <main>:
int main(void)
{
   0:   e92d4800    push    {fp, lr}
   4:   e28db004    add fp, sp, #4
   8:   e24dd010    sub sp, sp, #16
    float f1, f2, f3;

    f1 = 1.2;
   c:   e309399a    movw    r3, #39322  ; 0x999a
  10:   e3433f99    movt    r3, #16281  ; 0x3f99
  14:   e50b3010    str r3, [fp, #-16]
    f2 = 1.3;
  18:   e3063666    movw    r3, #26214  ; 0x6666
  1c:   e3433fa6    movt    r3, #16294  ; 0x3fa6
  20:   e50b300c    str r3, [fp, #-12]
    f3 = f1 / f2;
  24:   e51b0010    ldr r0, [fp, #-16]
  28:   e51b100c    ldr r1, [fp, #-12]
  2c:   ebfffffe    bl  0 <__aeabi_fdiv>
  30:   e1a03000    mov r3, r0
  34:   e50b3008    str r3, [fp, #-8]

    return 0;
  38:   e3a03000    mov r3, #0
}
  3c:   e1a00003    mov r0, r3
  40:   e24bd004    sub sp, fp, #4
  44:   e8bd8800    pop {fp, pc}
```

也是调用的 `__aeabi_fdiv` libc function。