CONFIG_VFP=y

in arch/arm/Makefile

```
core-$(CONFIG_VFP)      += arch/arm/vfp/
```

难道vfp instruction是通过undefined instruction exception来处理的吗？如果这样不是太低效了吗？

in arch/arm/kernel/entry-armv.S

```
__und_svc:
#ifdef CONFIG_KPROBES
        @ If a kprobe is about to simulate a "stmdb sp..." instruction,
        @ it obviously needs free stack space which then will belong to
        @ the saved context.
        svc_entry 64
#else
        svc_entry
#endif
        @
        @ call emulation code, which returns using r9 if it has emulated
        @ the instruction, or the more conventional lr if we are to treat
        @ this as a real undefined instruction
        @
        @  r0 - instruction
        @
#ifndef CONFIG_THUMB2_KERNEL
        ldr     r0, [r4, #-4]
#else
        mov     r1, #2
        ldrh    r0, [r4, #-2]                   @ Thumb instruction at LR - 2
        cmp     r0, #0xe800                     @ 32-bit instruction if xx >= 0
        blo     __und_svc_fault
        ldrh    r9, [r4]                        @ bottom 16 bits
        add     r4, r4, #2
        str     r4, [sp, #S_PC]
        orr     r0, r9, r0, lsl #16
#endif
        adr     r9, BSYM(__und_svc_finish)        ①
        mov     r2, r4
        bl      call_fpe

        mov     r1, #4                                  @ PC correction to apply        ②
__und_svc_fault:
        mov     r0, sp                                  @ struct pt_regs *regs
        bl      __und_fault

__und_svc_finish:                                    ③
        ldr     r5, [sp, #S_PSR]                        @ Get SVC cpsr
        svc_exit r5                                     @ return from exception
 UNWIND(.fnend          )
ENDPROC(__und_svc)
```

①

从call_fpe() in arch/arm/kernel/entry-armv.S的comments看

```
 * Emulators may wish to make use of the following registers:
 *  r0  = instruction opcode (32-bit ARM or two 16-bit Thumb)
 *  r2  = PC value to resume execution after successful emulation
 *  r9  = normal "successful" return address
 *  r10 = this threads thread_info structure
 *  lr  = unrecognised instruction return address
 * IRQs enabled, FIQs enabled.
```

好像是如果引起该undefined instruction exception的指令实际上是VFP instruction，

那么call_fpe()会handle该exception，则会从call_fpe()返回到__und_svc_finish，即

并不会真正作为invalid instruction处理。

②

这里是call_fpe()并不认识该instruction，也就是这是真正的undefined instruction,那么

要跳转到__und_fault() --- 真正的handler。

③

退出exception。