vector_irq

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

1. vector table entry

```
/*
 * Interrupt dispatcher
 */
        vector_stub     irq, IRQ_MODE, 4

        .long   __irq_usr                       @  0  (USR_26 / USR_32)
        .long   __irq_invalid                   @  1  (FIQ_26 / FIQ_32)
        .long   __irq_invalid                   @  2  (IRQ_26 / IRQ_32)
        .long   __irq_svc                       @  3  (SVC_26 / SVC_32)
        .long   __irq_invalid                   @  4
        .long   __irq_invalid                   @  5
        .long   __irq_invalid                   @  6
        .long   __irq_invalid                   @  7
        .long   __irq_invalid                   @  8
        .long   __irq_invalid                   @  9
        .long   __irq_invalid                   @  a
        .long   __irq_invalid                   @  b
        .long   __irq_invalid                   @  c
        .long   __irq_invalid                   @  d
        .long   __irq_invalid                   @  e
        .long   __irq_invalid                   @  f
```

__irq_usr() --> Application运行时发生interrupt

__irq_svc() --> kernel运行时发生interrupt

2. __irq_usr()

```
        .align  5
__irq_usr:
        usr_entry
        kuser_cmpxchg_check
        irq_handler
        get_thread_info tsk
        mov     why, #0
        b       ret_to_user_from_irq
 UNWIND(.fnend          )
ENDPROC(__irq_usr)
```

3. irq_handler macro

```
/*
 * Interrupt handling.
 */
        .macro  irq_handler
#ifdef CONFIG_MULTI_IRQ_HANDLER
        ldr     r1, =handle_arch_irq
        mov     r0, sp
        adr     lr, BSYM(9997f)
        ldr     pc, [r1]
#else
        arch_irq_handler_default
#endif
9997:
        .endm
```

4. 在G2 LSP中CONFIG_MULTI_IRQ_HANDLER=y

所以就是运行如下code

```
        ldr     r1, =handle_arch_irq
        mov     r0, sp
        adr     lr, BSYM(9997f)
        ldr     pc, [r1]
```

5. call handle_arch_irq()

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

```
#ifdef CONFIG_MULTI_IRQ_HANDLER
        .globl  handle_arch_irq
handle_arch_irq:
        .space  4
#endif
```

在static vmlinux中handle_arch_irq指示function pointer，而且还未初始化。还function pointer一般在各个interrupt controller driver中被初始化。即最终直接跳转到与特定interrupt controller相关的handler中。

2.1 __irq_svc()

在处理interrupt上，与__irq_usr()几乎相同。区别是在从hardware interrupt handler返回后的不同处理。

```
__irq_svc:
        svc_entry
        irq_handler

#ifdef CONFIG_PREEMPT
        get_thread_info tsk
        ldr     r8, [tsk, #TI_PREEMPT]          @ get preempt count
        ldr     r0, [tsk, #TI_FLAGS]            @ get flags
        teq     r8, #0                          @ if preempt count != 0
        movne   r0, #0                          @ force flags to 0
        tst     r0, #_TIF_NEED_RESCHED
        blne    svc_preempt
#endif

        svc_exit r5, irq = 1                    @ return from exception
 UNWIND(.fnend          )
ENDPROC(__irq_svc)
```

???