in arch/arm/mm/fault.c

```
1.  /*
2.   * Dispatch a data abort to the relevant handler.
3.   */
4.  asmlinkage void __exception
5.  do_DataAbort(unsigned long addr, unsigned int fsr, struct pt_regs *regs)
6.  {
7.          const struct fsr_info *inf = fsr_info + fsr_fs(fsr);        ①
8.          struct siginfo info;
9.
10.         if ((fsr == 0xc06) || (fsr == 0xa11)) {
11.                 printk(KERN_EMERG "FIX ignoring exception %#x addr=%lx %s:%d\n\n"
    , fsr, addr, current->comm, current->pid);
12.                 return;
13.         }
14.
15.         if (!inf->fn(addr, fsr & ~FSR_LNX_PF, regs))
16.                 return;
17.
18.         printk(KERN_ALERT "Unhandled fault: %s (0x%03x) at 0x%08lx\n",
19.                 inf->name, fsr, addr);
20.
21.         info.si_signo = inf->sig;
22.         info.si_errno = 0;
23.         info.si_code  = inf->code;
24.         info.si_addr  = (void __user *)addr;
25.         arm_notify_die("", regs, &info, fsr, 0);
26. }
27.
28. asmlinkage void __exception
29. do_PrefetchAbort(unsigned long addr, unsigned int ifsr, struct pt_regs *regs)
30. {
```

```
31.          const struct fsr_info *inf = ifsr_info + fsr_fs(ifsr);        ②
32.          struct siginfo info;
33.
34.          if (!inf->fn(addr, ifsr | FSR_LNX_PF, regs))
35.                  return;
36.
37.          printk(KERN_ALERT "Unhandled prefetch abort: %s (0x%03x) at 0x%08lx\n",
38.                  inf->name, ifsr, addr);
39.
40.          info.si_signo = inf->sig;
41.          info.si_errno = 0;
42.          info.si_code  = inf->code;
43.          info.si_addr  = (void __user *)addr;
44.          arm_notify_die("", regs, &info, ifsr, 0);
45.  }
```

```
1.  #define FSR_FS3_0               (15)
2.  #define FSR_FS4                 (1 << 10)
3.
4.  static inline int fsr_fs(unsigned int fsr)
5.  {
6.          return (fsr & FSR_FS3_0) | (fsr & FSR_FS4) >> 6;
7.  }
```

fsr & FSR_FS3_0不知道啥意思？ ARM ARM中bit 15没有说明

(fsr & FSR_FS4) >> 6

fault status 用 5 bits来identify fault encoding

(bit 10)(bit 3)(bit 2)(bit 1)(bit 0)

最高位位于fault status的bit 10,所以这里把bit与第4bit合并，这样fsr_fs()返回的就是

0 to 31的fault encoding，并用其作为下面两个struct fsr_info array的index。

addr是产生abort的instruction的address

该值应该来自于p15,c6

mrc p15, 0, r0, c6, c0, 0

get data fault address

mrc p15, 0, r0, c6, c0, 2

get instruction fault address

①

fsr是data fault status register中指示的具体是那种fault.

fsr --> 来自p15,c5,0

mrc p15, 0, r0, c5, c0, 0

r0 contains data fault status value

②

ifsr是instruction fault status register中指示的具体是那种fault.

ifsr --> 来自p15,c5,1

mrc p15, 0, r0, c5, c0, 1

r0 contains instruction fault status value

XXXX,XXXX,XXXX,XXXX,XXXX,X**X**XX,XXXX,**XXXX**

bit 0 - bit 3 + bit 10 are for fault status

XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,**XXXX**,XXXX

bit 4 - bit 7, domain, 即引起access fault的domain

但在ARM ARM(Architecture Reference Manual) ARMv7-A and ARMv7-R中由如下描述：

```
From ARMv7 use of this field is deprecated
```

关键的data structure是arch/arm/mm/fsr-2level.c中的struct fsr_info的数组。

```
struct fsr_info {
        int     (*fn)(unsigned long addr, unsigned int fsr, struct pt_regs *regs)
;
        int     sig;
        int     code;
        const char *name;
};
```

fn是某种specific fault handler

sig and code是当kernel不能处理该fault,即fn()返回非0时，要发送的signal信息。

XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX,XXXX

构成的5-bit fault encoding (最高位好像没实质性内容)

引起访问失效的原因	状态标识	域标识	C6
终端异常(Terminal Exception)	0010 (0x2)	无效	生产商定义
中断向量访问异常(Vector Exception)	0000 (0x0)	无效	有效
地址对齐	00x1	无效	有效
一级页表访问失效	1100 (0xc)	无效	有效
二级页表访问失效	1110 (0xe)	有效	有效
基于段的地址变换失效	0101 (0x5)	无效	有效
基于页的地址变换失效	0111 (0x7)	有效	有效
基于段的存储访问中域控制失效	1001 (0x9)	有效	有效
基于页的存储访问中域控制失效	1101 (0xd)	有效	有效
基于段的存储访问中访问权限控制失效	1111 (0xf)	有效	有效
基于页的存储访问中访问权限控制失效	0100 (0x4)	有效	有效
基于段的 cache 预取时外部存储系统失效	0110 (0x6)	有效	有效
基于页的 cache 预取时外部存储系统失效	1000 (0x8)	有效	有效
基于段的非 cache 预取时外部存储系统失效	1010 (0xa)	有效	有效

```
1.  static struct fsr_info fsr_info[] = {
2.          /*
3.           * The following are the standard ARMv3 and ARMv4 aborts.  ARMv5
4.           * defines these to be "precise" aborts.
5.           */
6.          { do_bad,               SIGSEGV, 0,             "vector exception"
               },
7.          { do_bad,               SIGBUS,  BUS_ADRALN,    "alignment exception"
               },
8.          { do_bad,               SIGKILL, 0,             "terminal exception"
               },
9.          { do_bad,               SIGBUS,  BUS_ADRALN,    "alignment exception"
               },
10.         { do_bad,               SIGBUS,  0,             "external abort on linefe
    tch"       },
11.         { do_translation_fault, SIGSEGV, SEGV_MAPERR,   "section translation faul
    t"         },
12.         { do_bad,               SIGBUS,  0,             "external abort on linefe
    tch"       },
13.         { do_page_fault,        SIGSEGV, SEGV_MAPERR,   "page translation fault"
               },
14.         { do_bad,               SIGBUS,  0,             "external abort on non-li
    nefetch"   },
15.         { do_bad,               SIGSEGV, SEGV_ACCERR,   "section domain fault"
               },
16.         { do_bad,               SIGBUS,  0,             "external abort on non-li
    nefetch"   },
17.         { do_bad,               SIGSEGV, SEGV_ACCERR,   "page domain fault"
               },
18.         { do_bad,               SIGBUS,  0,             "external abort on transl
    ation"     },
19.         { do_sect_fault,        SIGSEGV, SEGV_ACCERR,   "section permission fault
    "          },
20.         { do_bad,               SIGBUS,  0,             "external abort on transl
    ation"     },
21.         { do_page_fault,        SIGSEGV, SEGV_ACCERR,   "page permission fault"
               },
22.         /*
23.          * The following are "imprecise" aborts, which are signalled by bit
24.          * 10 of the FSR, and may not be recoverable.  These are only
25.          * supported if the CPU abort handler supports bit 10.
26.          */
27.         { do_bad,               SIGBUS,  0,             "unknown 16"
               },
28.         { do_bad,               SIGBUS,  0,             "unknown 17"
               },
29.         { do_bad,               SIGBUS,  0,             "unknown 18"
               },
30.         { do_bad,               SIGBUS,  0,             "unknown 19"
               },
31.         { do_bad,               SIGBUS,  0,             "lock abort"
               }, /* xscale */
32.         { do_bad,               SIGBUS,  0,             "unknown 21"
```

```
            },
33.         { do_bad,               SIGBUS,  BUS_OBJERR,    "imprecise external abort
    "           }, /* xscale */
34.         { do_bad,               SIGBUS,  0,             "unknown 23"
            },
35.         { do_bad,               SIGBUS,  0,             "dcache parity error"
            }, /* xscale */
36.         { do_bad,               SIGBUS,  0,             "unknown 25"
            },
37.         { do_bad,               SIGBUS,  0,             "unknown 26"
            },
38.         { do_bad,               SIGBUS,  0,             "unknown 27"
            },
39.         { do_bad,               SIGBUS,  0,             "unknown 28"
            },
40.         { do_bad,               SIGBUS,  0,             "unknown 29"
            },
41.         { do_bad,               SIGBUS,  0,             "unknown 30"
            },
42.         { do_bad,               SIGBUS,  0,             "unknown 31"
            },
43. };
44.
45. static struct fsr_info ifsr_info[] = {
46.         { do_bad,               SIGBUS,  0,             "unknown 0"
            },
47.         { do_bad,               SIGBUS,  0,             "unknown 1"
            },
48.         { do_bad,               SIGBUS,  0,             "debug event"
            },
49.         { do_bad,               SIGSEGV, SEGV_ACCERR,   "section access flag faul
    t"          },
50.         { do_bad,               SIGBUS,  0,             "unknown 4"
            },
51.         { do_translation_fault, SIGSEGV, SEGV_MAPERR,   "section translation faul
    t"          },
52.         { do_bad,               SIGSEGV, SEGV_ACCERR,   "page access flag fault"
            },
53.         { do_page_fault,        SIGSEGV, SEGV_MAPERR,   "page translation fault"
            },
54.         { do_bad,               SIGBUS,  0,             "external abort on non-li
    nefetch"  },
55.         { do_bad,               SIGSEGV, SEGV_ACCERR,   "section domain fault"
            },
56.         { do_bad,               SIGBUS,  0,             "unknown 10"
            },
57.         { do_bad,               SIGSEGV, SEGV_ACCERR,   "page domain fault"
            },
58.         { do_bad,               SIGBUS,  0,             "external abort on transl
    ation"      },
59.         { do_sect_fault,        SIGSEGV, SEGV_ACCERR,   "section permission fault
    "           },
60.         { do_bad,               SIGBUS,  0,             "external abort on transl
    ation"      },
```

```
61.          { do_page_fault,        SIGSEGV, SEGV_ACCERR,   "page permission fault"
                },
62.
63.          { do_bad,               SIGBUS,  0,             "unknown 16"
                },
64.          { do_bad,               SIGBUS,  0,             "unknown 17"
                },
65.          { do_bad,               SIGBUS,  0,             "unknown 18"
                },
66.          { do_bad,               SIGBUS,  0,             "unknown 19"
                },
67.          { do_bad,               SIGBUS,  0,             "unknown 20"
                },
68.          { do_bad,               SIGBUS,  0,             "unknown 21"
                },
69.          { do_bad,               SIGBUS,  0,             "unknown 22"
                },
70.          { do_bad,               SIGBUS,  0,             "unknown 23"
                },
71.          { do_bad,               SIGBUS,  0,             "unknown 24"
                },
72.          { do_bad,               SIGBUS,  0,             "unknown 25"
                },
73.          { do_bad,               SIGBUS,  0,             "unknown 26"
                },
74.          { do_bad,               SIGBUS,  0,             "unknown 27"
                },
75.          { do_bad,               SIGBUS,  0,             "unknown 28"
                },
76.          { do_bad,               SIGBUS,  0,             "unknown 29"
                },
77.          { do_bad,               SIGBUS,  0,             "unknown 30"
                },
78.          { do_bad,               SIGBUS,  0,             "unknown 31"
                },
79.  };
```

```
1.  /*
2.   * This abort handler always returns "fault".
3.   */
4.  static int
5.  do_bad(unsigned long addr, unsigned int fsr, struct pt_regs *regs)
6.  {
7.          return 1;
8.  }
```

返回1表示kernel不处理该fault (Data Abort or Prefetch Abort),只是发送SIGBUS signal (Bus Error)

Question: SIGBUS and SIGSEGV之间区别到底是什么呢？

网上有种说法如下：

SIGREGV是权限的问题，即地址是有效的，但是对该地址无权限

SIGBUS是地址的问题，即地址本身是无效的