



# 又见异常：Bad mode in Synchronous Abort

by HappySeeker

March 3, 2016

in [Kernel](#)

## 闲话

最近自己的Arm64环境频繁崩溃，又见新的异常Bad mode in Synchronous Abort，哎，谁来拯救这脆弱的国产硬件。

## 问题现象

环境中出现了panic，dmesg中有如下打印：

```
[ 1259.780974] Bad mode in Synchronous Abort handler detected, code 0x8600000f -- IABT (current EL)
[ 1259.789716] CPU: 12 PID: 2293 Comm: mate-settings-d Not tainted 4.1.15-1.el7.aarch64 #2
[ 1259.797678] Hardware name: xxx
[ 1259.803567] task: fffffffc8c9bd1700 ti: fffffffc8c9e4c000 task.ti: fffffffc8c9e4c000
[ 1259.811011] PC is at 0x7f942f9828
[ 1259.814308] LR is at 0x7f942f9828
[ 1259.817605] pc : [<0000007f942f9828>] lr : [<0000007f942f9828>] pstate: 600003c5
[ 1259.824962] sp : fffffffc8c9e4fed0
[ 1259.828258] x29: 0000007fc5ef6be0 x28: 0000000000000000
[ 1259.833563] x27: 0000007f8dfc92c0 x26: 0000007f8de63000
[ 1259.838868] x25: 0000007f8de47e48 x24: 0000007f8de47e40
[ 1259.844172] x23: 0000000080000000 x22: 0000007f9c51abd8
[ 1259.849478] x21: fffffffc8c9e4fff0 x20: 0000007f94344000
[ 1259.854784] x19: 0000000013e20a10 x18: 0000000000000004
[ 1259.860090] x17: 0000007f9c51abd0 x16: 0000007f943448c8
[ 1259.865395] x15: 0000000000000028 x14: 0xfffffffffffffffe
[ 1259.870699] x13: 0000000000000020 x12: 0000000000000002
[ 1259.876004] x11: 0000000000000000 x10: 0000000000000250
[ 1259.881308] x9 : 0000000000000251 x8 : 0000000000000000
[ 1259.886614] x7 : fffffffc0000b69a8 x6 : fffffffc8c9e4fe80
[ 1259.891920] x5 : fffffffc000030001 x4 : 0000000000000000
[ 1259.897225] x3 : fffffffc8c9e4fcb0 x2 : 0000000000001c55
[ 1259.902530] x1 : 000000000000003c0 x0 : 0000000000000000
[ 1259.907835]
[ 1259.909320] Internal error: Oops - bad mode: 0 [#1] SMP
[ 1259.914517] Modules linked in: fuse snd_hda_codec_hdmi snd_hda_in
```

从打印看，应该是先出现了Synchronous Abort，然后进入了内核oops流程。

## 分析

### 什么是Bad mode in Synchronous Abort？

内核中搜一下相关打印，确认是在bad\_mode函数中打印的：

```
/*
 * bad_mode handles the impossible case in the exception vector.
 */
asmlinkage void bad_mode(struct pt_regs *regs, int reason, unsigned int esr)
{
    siginfo_t info;
    /*获取异常时的PC指针*/
    void __user *pc = (void __user *)instruction_pointer(regs);
    console_verbose();
    /*打印异常信息，messages中可以看到。*/
    pr_crit("Bad mode in %s handler detected, code 0x%08x -- %s\n",
            handler[reason], esr, esr_get_class_string(esr));
    /*打印寄存器内容*/
    __show_regs(regs);
    /*如果发生在用户态，需要向其发送信号，这种情况下，发送SIGILL信号，所以就不会有core文件产生了*/
    info.si_signo = SIGILL;
    info.si_errno = 0;
    info.si_code = ILL_ILLOPC;
    info.si_addr = pc;
    /*给用户态进程发生信号，或者die然后panic*/
    arm64_notify_die("Oops - bad mode", regs, &info, 0);
}
```

看看Arm64对应的entry.S，可以知道可以在很多中情况进入，通常情况下，内核无法进一步处理的情况，都会进入bad\_mode。

再仔细看看错误打印的内容：

```
80974] Bad mode in Synchronous Abort handler detected, code 0x8600000f -- IABT (current EL)
```

这里面有几个关键点：

- 1. “Synchronous Abort”对应异常的reason。
- 2. code为0x8600000f，对应的是ESR寄存器中的内容，这个需要参考芯片手册解析。
- 3. “IABT (current EL)”对应异常的分类。

接下来逐个分析这3点。

## Synchronous Abort

该打印的来源：`handler[reason]`

```
static const char *handler[] = {
    "Synchronous Abort",
    "IRQ",
    "FIQ",
    "Error"
};
```

说明传入bad\_mode()函数的reason为0，再看看entry.S代码:

```
/*
 * Bad Abort numbers
 * -----
 */
#define BAD_SYNC      0
#define BAD_IRQ      1
#define BAD_FIQ      2
#define BAD_ERROR    3
```

0对应为BAD\_SYNC，结合entry.S中定义的中断向量，可知如下中断向量可能进入：

```
el1_sync_invalid
el1_sync
el0_sync
```

由于在发生这个错误之后，紧接着进入了内核的OOPS流程，可见异常发生在内核态，所以排除el0\_sync，而el1\_sync\_invalid是当使用SP\_EL0时才进入的，基本不可能，所以，只可能从el1\_sync进入，看看el1\_sync的代码：

```
/*
 * EL1 mode handlers.
 */

.align 6
el1_sync:
    kernel_entry 1
    mrs    x1, esr_el1           // read the syndrome register
    lsr    x24, x1, #ESR_ELx_EC_SHIFT // exception class
    cmp    x24, #ESR_ELx_EC_DABT_CUR // data abort in EL1
    /*数据异常*/
    b.eq   el1_da
    cmp    x24, #ESR_ELx_EC_SYS64    // configurable trap
    /*未定义异常*/
    b.eq   el1_undef
    cmp    x24, #ESR_ELx_EC_SP_ALIGN // stack alignment exception
    /*栈对齐异常*/
    b.eq   el1_sp_pc
    cmp    x24, #ESR_ELx_EC_PC_ALIGN // pc alignment exception
    b.eq   el1_sp_pc
    cmp    x24, #ESR_ELx_EC_UNKNOWN  // unknown exception in EL1
    b.eq   el1_undef
    cmp    x24, #ESR_ELx_EC_BREAKPT_CUR // debug exception in EL1
    /*调试异常*/
    b.ge   el1_dbg
    /*其他异常，包括指令异常，目前没有实现相应的特定的处理接口，默认进入bad_mode*/
    b      el1_inv
    ...

el1_inv:
    /*指令异常会进入这里*/
    // TODO: add support for undefined instructions in kernel mode
    enable_dbg
    mov    x0, sp
    mov    x1, #BAD_SYNC
    mrs    x2, esr_el1
    /*最终进入bad mode*/
    b      bad_mode
ENDPROC(el1_sync)
```

由于本问题最终进入了bad\_mode，所以只可能是指令异常(或是其他没有单独处理的异常)才会进入这里。

## code分析

code即错误码，即ESR(异常状态寄存器)中的内容，具体为0x8600000f，参考Armv8的手册，前面六位确定错误类型，86对应的错误类型为：

```
100001 Instruction Abort taken without a change in Exception level
```

就是不改变当前异常级别的指令异常，对应非虚拟化和安全环境来说，就是在内核态发生的指令异常，因为如果发生在用户态，异常级别会改变。

最后6位为代表具体错误的错误码，0f对应为，具体解释为：

```
001111 Permission fault, level 3
```

也就是访问第3级页表时出现权限错误。

对于指令异常的使用，手册上有如下描述，建议自己理解下：

```
This encoding is used by:
• Instruction Abort that caused entry from a lower Exception level, where that Exception level must be EL0 or EL1 on AArch64 or using AArch32.
Used for MMU faults generated by instruction accesses and Synchronous external aborts, including synchronous parity or ECC errors. Not used for debug related exceptions.
• Instruction Abort from the current Exception level, where the current Exception level must be EL2 or EL3 on AArch64.
Used for MMU faults generated by instruction accesses and Synchronous external aborts, including synchronous parity or ECC errors. Not used for debug related exceptions
```

## 异常分类(“IABT (current EL)”)

这次报错对应的异常分类的打印为：“IABT (current EL)”，看看该错误类型的来源：

```
static const char *esr_class_str[] = {
    [0 ... ESR_ELx_EC_MAX] = "UNRECOGNIZED EC",
    [ESR_ELx_EC_UNKNOWN] = "Unknown/Uncategorized",
    [ESR_ELx_EC_WFx] = "WFI/WFE",
    [ESR_ELx_EC_CP15_32] = "CP15 MCR/MRC",
    [ESR_ELx_EC_CP15_64] = "CP15 MCRR/MRRC",
    [ESR_ELx_EC_CP14_MR] = "CP14 MCR/MRC",
    [ESR_ELx_EC_CP14_LS] = "CP14 LDC/STC",
    [ESR_ELx_EC_FP_ASIMD] = "ASIMD",
    [ESR_ELx_EC_CP10_ID] = "CP10 MRC/VMRS",
    [ESR_ELx_EC_CP14_64] = "CP14 MCRR/MRRC",
    [ESR_ELx_EC_ILL] = "PSTATE.IL",
    [ESR_ELx_EC_SVC32] = "SVC (AArch32)",
    [ESR_ELx_EC_HVC32] = "HVC (AArch32)",
    [ESR_ELx_EC_SMC32] = "SMC (AArch32)",
    [ESR_ELx_EC_SVC64] = "SVC (AArch64)",
    [ESR_ELx_EC_HVC64] = "HVC (AArch64)",
    [ESR_ELx_EC_SMC64] = "SMC (AArch64)",
    [ESR_ELx_EC_SYS64] = "MSR/MRS (AArch64)",
    [ESR_ELx_EC_IMP_DEF] = "EL3 IMP DEF",
    [ESR_ELx_EC_IABT_LOW] = "IABT (lower EL)",
    [ESR_ELx_EC_IABT_CUR] = "IABT (current EL)",
    [ESR_ELx_EC_PC_ALIGN] = "PC Alignment",
    [ESR_ELx_EC_DABT_LOW] = "DABT (lower EL)",
    [ESR_ELx_EC_DABT_CUR] = "DABT (current EL)",
    [ESR_ELx_EC_SP_ALIGN] = "SP Alignment",
    [ESR_ELx_EC_FP_EXC32] = "FP (AArch32)",
    [ESR_ELx_EC_FP_EXC64] = "FP (AArch64)",
    [ESR_ELx_EC_SERR0R] = "SError",
    [ESR_ELx_EC_BREAKPT_LOW] = "Breakpoint (lower EL)",
    [ESR_ELx_EC_BREAKPT_CUR] = "Breakpoint (current EL)",
    [ESR_ELx_EC_SOFTSTP_LOW] = "Software Step (lower EL)",
    [ESR_ELx_EC_SOFTSTP_CUR] = "Software Step (current EL)",
    [ESR_ELx_EC_WATCHPT_LOW] = "Watchpoint (lower EL)",
    [ESR_ELx_EC_WATCHPT_CUR] = "Watchpoint (current EL)",
    [ESR_ELx_EC_BKPT32] = "BKPT (AArch32)",
    [ESR_ELx_EC_VECTOR32] = "Vector catch (AArch32)",
    [ESR_ELx_EC_BRK64] = "BRK (AArch64)",
};
```

结合前面的错误码的分析，可以确认这是“发生在内核态的指令异常”。

## 疑问

- 从代码在出现这个异常后，立即打印了OOPS，而且触发了kdump，说明是发生在内核态，但为何PC和LR寄存器指向的地址还是用户态的地址呢？

可能答案：应该是由于前一个SError，应该是在前一个发生system error时，程序处于用户态，而此时保存了PC和LR，但在SError的处理过程中再次发生了这个异常，此时，虽然已经处于内核态，但PC和LR中的值还是发生SError之前的值。

- 为何是“001111 Permission fault, level 3”这种错误码，按理当前已经是内核态，是特权模式，应该不存在权限问题才对。 可能答案：由于之前已经发生过SError，这个数据可能不准确

Subscribe [via RSS](#)


Share: 

happyseeker.github.io





1 登录

❤

评分最高



开始讨论...



姓名

在 上还有

New Mission on Graphic

1条评论 • 2年前

Jiangbiao

我的第一篇GitHub Blog

1条评论 • 2年前

Jiangbiao

图形栈&架构

1条评论 • 2年前

Jiangbiao

闲聊Framebuffer

2条评论 • 2年前

Jiangbiao

HappySeeker

Site Map

About  
Posts  
Typography

Contact

✉ jiang.biao@hotmail.com  
🔗 HappySeeker  
in JiangBiao

Subscribe [via RSS](#)

HappySeeker's blog & My Tech Life.