

# 基于 SPARC 的地址非对齐异常处理研究

祝晓林<sup>1,2</sup>, 张 健<sup>2</sup>, 李 楠<sup>2</sup>

(1. 中国科学院大学, 北京 100049;

2. 中国科学院 空间应用工程与技术中心, 北京 100094)

**摘 要:**高可靠嵌入式系统中的异常处理策略是评估该系统可靠性的一个重要指标,尤其对于军工、航天等方面的应用。文中基于宇航级 CPU AT697F 和军用实时操作系统 VxWorks,分析了 SPARC V8 体系结构下 VxWorks 的异常处理机制,并详细介绍 SPARC V8 体系结构特有的寄存器窗口机制以及该机制结合 VxWorks 操作系统的具体实现。在此基础上,提出了针对 SPARC V8 体系结构的一种异常处理策略,并基于该策略完成了内存地址非对齐访问的异常处理,提高了嵌入式实时系统的可靠性。

**关键词:**VxWorks; SPARC V8; 异常处理; 地址非对齐

**中图分类号:**TP31

**文献标识码:**A

**文章编号:**1673-629X(2015)04-0018-04

**doi:**10.3969/j.issn.1673-629X.2015.04.005

## Research on Memory Access to Un-aligned Addresses Exception Based on SPARC

ZHU Xiao-lin<sup>1,2</sup>, ZHANG Jian<sup>2</sup>, LI Nan<sup>2</sup>

(1. University of Chinese Academy of Sciences, Beijing 100049, China;

2. Technology and Engineering Center for Space Utilization, Chinese Academy of Sciences, Beijing 100094, China)

**Abstract:**Exception handling strategy is a very important indicator in evaluating reliability level of embedded system, especially in the field of war industry, space industry and so on. Based on space flight standard CPU AT697F and real time operating system VxWorks, analyze the VxWorks exception handling strategy for SPARC, then illustrate register window mechanism of SPARC V8 architecture in detail and special implementation of this mechanism combined with VxWorks. Based on this, introduce an exception handling strategy according to SPARC V8, and implement the exception handling function to solve memory access to un-aligned addresses exception, improving the reliability of the system effectively.

**Key words:**VxWorks; SPARC V8; exception handling; memory access to un-aligned addresses

## 1 概 述

VxWorks 操作系统是美国 WindRiver 公司设计开发的一种嵌入式实时操作系统,它以其良好的可靠性和卓越的实时性被广泛地应用在通信、军事、航空、航天等对实时性要求极高的领域中<sup>[1]</sup>。VxWorks 支持优先级抢占的多任务调度方式,没有严格的内核态与用户态的区分,任务调度时上下文切换延时小,支持多种 CPU 架构的板级支持包(BSP),同时提供快速灵活的与 ANSI 兼容的 IO 系统,方便用户编程<sup>[2]</sup>。基于这些优点,VxWorks 在航天及控制领域已经得到越来越广

泛的应用,比如在美国的 F-16、FA-18 战斗机、B-2 隐形轰炸机和爱国者导弹上,甚至在火星表面登陆的火星探测器上也使用到了 VxWorks。

SPARC 全称为“可扩充处理器架构”(Scalable Processor ARChitecture),是 RISC 微处理器架构之一,它的设计目标是为了优化编译和更好地实现硬件流水。SUN 和 TI 公司于 1987 年合作开发了 RISC 微处理器—SPARC,这是业界出现的第一款有可扩展性功能的微处理器。文中使用的 AT697F 是一款高性能、高集成度的 SPARC V8 处理器,它的实现是以欧空局

的 LEON2 处理器为模板,并且已被成功应用于我国的航天项目中<sup>[3]</sup>。

基于 SPARC 独特的寄存器窗口机制,文中提出一种基于 VxWorks 的异常处理策略,并基于该策略实现了内存地址非对齐访问的异常处理,并在 AT697F 处理器上进行了验证。

## 2 SPARC 寄存器窗口机制及异常处理策略

### 2.1 SPARC 寄存器窗口机制

AT697F 处理器具有 8 个寄存器窗口,5 级流水线,16 K 字节大小的两路组相联数据 Cache、32 字节大小的四路组相联指令 Cache、支持双精度浮点数据类型的浮点处理单元<sup>[4]</sup>。支持完善的片上外设,比如 PROM,SRAM,SDRAM 和 I/O 映射空间访问的存储控制器,两个 24 位定时器,一个看门狗,三个串行通信接口,4 个可编程的中断控制器,32 个通用 IO 接口,33 MHz PCI 接口等,同时具有完全的三模冗余设计、EDAC 和奇偶校验,功能接口完善,可以很好地满足航天控制的需求。SPARC 总体结构框图如图 1 所示<sup>[5]</sup>。

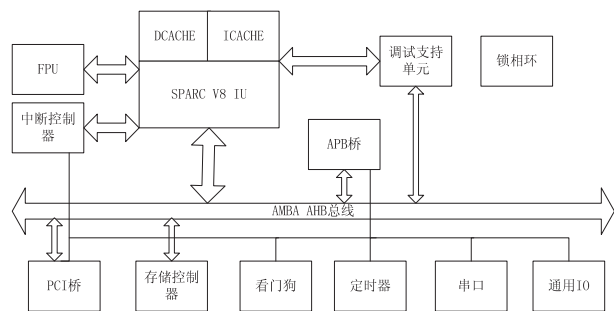


图 1 SPARC 结构框图

寄存器窗口是 SPARC V8 架构处理器中十分重要的概念。每一个寄存器窗口可以访问的寄存器包括 32 个非浮点寄存器:(输入寄存器)in0-in7,(输出寄存器)out0-out7,(全局寄存器)global0-global7,(局部寄存器)local0-local7。根据实际情况可以实现窗口个数 2~32 个(NWINDOWS),程序调用时,调用程序的输出寄存器成为被调用程序的输入寄存器,输入、输出寄存器主要用于向子程序传递参数,接收子程序的运行结果和跟踪存储器堆栈<sup>[6]</sup>。本地寄存器用于存储临时值,全局寄存器不属于任何窗口,用于存储全局变量。这种设计可以显著减少耗时的访存指令并明显提高指令运行性能<sup>[7]</sup>。

处理器状态寄存器(Processor State Register,PSR)是最重要的 IU 相关的内部寄存器,低 5 位 CWP 组成了当前窗口的指针,该寄存器还可以使能和屏蔽陷阱,使能协处理器和浮点处理器,以及指定处理器将要接受陷阱的级别<sup>[8]</sup>。窗口无效屏蔽寄存器(Window Invalid Mask register,WIM)可以决定一条 SAVE、RE-

STORE 或者 RETT 指令是否会产生窗口上溢或者下溢陷阱,WIM[n]对应着 CWP=n 的寄存器组。WIM[n]=1 代表该窗口无效,WIM[n]=0 则代表该窗口有效。AT697F 处理器具有 8 个寄存器窗口,因此 W8-W31 的值始终为 0,W0-W7 的初始值为 0,但会随着程序的运行而变化。

当 SAVE 指令执行时,硬件自动将新窗口(CWP-1)mod nWindows 与窗口无效掩码寄存器 WIM 比较,若新的窗口无效,即 WIM[CWP-1]=1,则会产生上溢陷阱,调用 windowOverflow 异常处理程序。

RESTORE 和 RETT 指令执行时增加 CWP,(CWP+1)mod nWindows 也要与窗口无效掩码寄存器比较,若新的窗口无效,则会产生下溢陷阱,调用 windowUnderflow 异常处理程序<sup>[9]</sup>。需要注意的是,在 trap 发生时 CWP 也会减 1,但不进行窗口溢出的检查。

windowOverflow 是由 SAVE 指令引起的窗口上溢陷阱处理函数,该异常产生后处理流程如下:

(1)硬件将 CWP 置成(CWP-1)mod nWindows,进入该异常处理函数。

(2)WIM 寄存器清零,然后把 CWP-1 的窗口寄存器保存起来。

(3)再将 WIM[CWP]置零,WIM[CWP-1]置 1,即使 CWP 窗口有效,CWP-1 窗口无效。

(4)跳回到引起陷阱的 SAVE 指令继续执行。

首先知道,WIM 是由软件更改的,在初始化时软件将 WIM[1]置成 1,将 CWP 置成 0,这样每进行一次函数调用,CWP 变为(CWP-1)mod nWindows,当 CWP 减到 2 后继续 SAVE 时会触发一次 windowOverflow,因为 WIM[1]=1,而这时在 windowOverflow 函数里就要把 WIM[1]置 0,并同时保存 CWP=0 的窗口,将 WIM[0]置 1,即将窗口 1 置成有效,窗口 0 置成无效。这样的设计保证函数调用深度大于窗口个数时,可以将窗口寄存的内容保存到堆栈中。

windowUnderflow 函数是由 RESTORE 指令引起的窗口下溢陷阱处理函数,异常产生后处理流程如下:

(1)硬件将 CWP 置成(CWP+1)mod nWindows,进入该异常处理函数。

(2)将 WIM 寄存器清零,并且 RESTORE 两次(注意,异常产生后 CWP 会自动减 1)即将 CWP+2 的窗口寄存器恢复出来。

(3)将 WIM[CWP+3]置 1,使 CWP+3 窗口无效。

(4)跳回到引起陷阱的 RESTORE 指令继续执行。

windowOverflow 和 windowUnderflow 两个函数巧妙配合,完成了深度函数调用时现场的恢复和保存,使得函数可以深层次调用,这也是 SPARC 寄存器窗口机制的核心,后面完成的内存访问地址非对齐的异常处理

也是基于该机制完成的<sup>[10]</sup>。

但是可以看到,当函数调用深度大于窗口个数时,寄存器现场会保存在内存里,这会使得函数调用的效率降低,因此可以通过尽量减少函数的调用深度来提高程序的效率。

## 2.2 基于 SPARC 的 VxWorks 异常处理策略

SPARC V8 产生异常的原因有很多种,例如程序断点,除零异常,数据访问异常,写内存错误,总线异常,加载指令异常等<sup>[11]</sup>。通常情况下产生异常后,会停止当前产生异常的任务,并通过函数接口报错,通知上层应用程序,如果处理不及时可能导致系统崩溃死机,这在高可靠系统中是不允许的。

根据 The SPARC Architecture Manual 手册,SPARC V8 包括两种类型的 trap:默认异常处理和增强型异常处理。默认异常处理是通用的异常处理,适用于所有的异常类型,增强型异常处理是在默认异常处理的基础上,由用户安装具体的异常处理程序<sup>[12]</sup>。文中将地址非对齐处理看做是增强型异常处理,用户自己定制的异常处理程序,并安装到异常向量表中,实现个性化处理。

异常处理的核心有两点:处理和恢复。文中采用钩子函数机制,异常处理函数在系统初始化时被系统挂接到异常处理的钩子函数中,由系统异常触发。该函数的功能是进行现场保存和用户自定义的处理工作,然后调用异常恢复函数。处理和恢复函数根据不同的硬件和软件环境,处理方式也不同。

针对 VxWorks5.4 操作系统中,CPU 检测到异常后,跳到相应的向量表处,首先执行 excEnter 汇编函数,进行异常处理前的窗口检测,现场保存工作,然后调用 excExcHandle 函数,该函数是 C 语言实现的函数,方便上层用户对钩子函数进行挂接,对不同的 trap 进行个性化处理。

## 3 地址非对齐异常处理的实现

处理器在读取或写入内存数据单元时,目标地址必须是所访问基本数据类型字节数的整数倍,这个就叫做地址对齐,如果访问非对齐的地址,CPU 会触发异常。ARM,POWERPC,SPARC 等体系架构都不支持非对齐地址访问。

以 AT697 为例,地址非对齐的异常向量号为 0x7,地址非对齐异常主要有以下几种情况引起:

(1) SPARC 规定,半字是两个字节,字是四个字节,双字是八个字节,load/store 内存操作指令,当执行读取或存储半字指令时内存地址非二字节对齐,执行读取或存储字指令时内存地址非四字字节对齐,执行读取或存储双字指令时内存地址非八字节对齐,都会产

生该异常。

(2) JMWPL/RETT 指令跳到的地址非四字字节对齐时,会产生该异常。

一般情况下,SPARC 地址非对齐会产生异常,结束当前任务,为了不妨碍当前任务继续执行,继续对非对齐的地址进行访问,对于第一种异常情况可以做如下处理:对非对齐的地址拆分成两次对齐访问,并将两次数据进行拼接,完成对非对齐地址的访问。第二种情况可以认为是跳转指令中地址个别位翻转带来的地址非对齐,AT697F 核心寄存器组增加三模冗余功能防止位翻转,同时 EDAC 校验可以纠正一位寄存器或内存数据错误,如果遇这种异常,可以尝试强制将跳转地址的低两位清零,使其强制对齐<sup>[13]</sup>。

以 4 字节的非对齐地址读操作为例。反汇编显示,产生异常的指令为 ld [%o0],%o1,其中 %o0 为非四字字节对齐的内存地址,则执行该语句,会产生 memory address not aligned 异常,异常向量号为 0x7,CPU 会跳转到异常向量表的指定位置,处理过程如下:

(1) 执行 excEnter,将异常现场保存到 ESF 结构体中(包括产生异常指令的 PC,NPC 等),同时判断当前 WIM[CWP]是否为 1,如果为 1,则将其清 0,并将 CWP-1 窗口的寄存器保存到自己的堆栈中,并置 WIM[CWP-1]为 1。

(2) 跳到 excExcHandle 函数,这个函数是由 C 语言实现的,有通用的异常处理函数,可以通知上层用户或者打印异常信息等。

(3) 对于地址非对齐异常,调用自己定义的初始化好的钩子函数\_func\_excHandler,这个函数由汇编实现,假若要读取的地址是 0x8000,0001,则该函数会分别对地址 0x8000,0000 和 0x8000,0004 进行两次读取操作,并进行移位,拼接读出 0x8000,0001 地址处的 32 位数据。

(4) 数据读完后,需要将异常现场的内容恢复出来,这里选择当前窗口的上一个窗口 CWP+1 作为恢复窗口,因为这个窗口是刚刚调用过的,肯定是有效的窗口,需要将异常产生前的现场全部恢复到 CWP+1 窗口,并要将读出的 32 位数据恢复到 CWP+1 窗口的 %o1,然后返回到 NEXT PC 地址继续执行。

代码修改完成后需要重新编译 VxWorks 库文件,生成新的库文件代码后再重新编译新的内核。具体代码及步骤不再详述。地址非对齐处理流程见图 2。

## 4 实验验证

将以下的测试程序加载入内核。

```
void excTest2()  
{
```

```
char buff[8] = {0x12, 0x34, 0x56, 0x78, 0x9a, 0xbc, 0xde, 0xf0};
int v32=0;
int *p32=NULL;
p32=(int*)&(buff[1]);
v32=*p32;
}
```

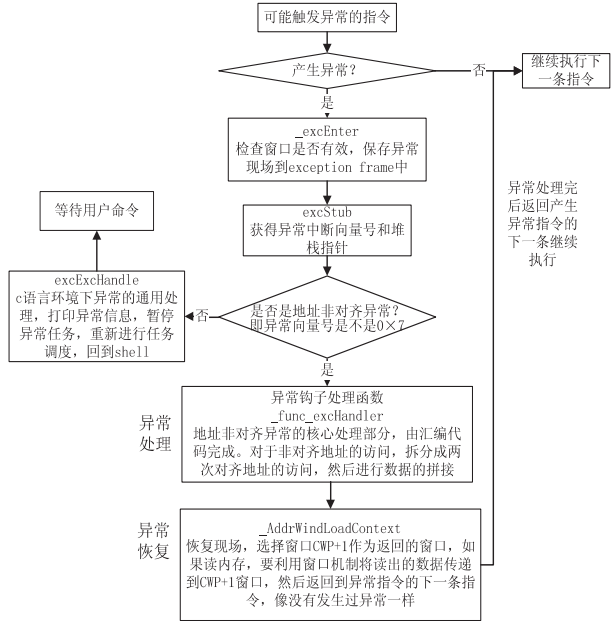


图2 地址非对齐处理

程序原意是读取 0x3456789a 四个字节,显然,此函数中这四个字节的起始地址 p32 是非四字节对齐的地址,对它进行四字节读操作,加入异常处理之前,运行 excTest 函数会报告如图 3 所示的异常。

```
->
-> excTest2

Exception: Memory Address Not Aligned
program counter:      0x402003dc
next program counter: 0x402003e0
processor status register: 0x000010c5

0x4020d2f0 vxTaskEntry +c : shell (1, 0, 0, 0, 0, 0, 0)
0x4023d438 shell      +154: 4023d64 (0, 3, 7f, 1, 40293400, c350)
0x40232f58 shell      +374: execute (4038a558, 0, 80, 0, 4038a558, 4038a348)
0x402330ac execute    +c4 : yyparse (403fe28, 40293400, ffffffff, 4038a340, 403ffffc0)
0x4026c23c yyparse    +74c: 4026a488 (403fe9f8, 403fe9f8, 10, 2bb, 4, 0)
0x4026a5d4 yystart    +880: 0 (0, 0, 0, 0, 0, 0)
shell restarted

->
->
```

图3 异常报告

加入异常处理之后,运行结果如图 4 所示,像没发生异常一样,0x3456789a 数据被正确读出,试验成功。

5 结束语

异常处理策略的完善性对于高可靠嵌入式系统有

着十分重要的影响。在 SPARC 架构处理器下进行异常处理需要充分利用 SPARC 独特的窗口机制,可以极大提高异常处理的效率,同时结合嵌入式实时操作系统 VxWorks 的优点,可以极大提高系统的稳定性及可靠性<sup>[14]</sup>。

```
->
-> excTest2
not aligned address is 0x4038a329
read data from the address is 0x3456789a

->
->
```

图4 运行结果

地址非对齐访问的异常处理在很多 CPU 上都没有实现,不同的 CPU 上虽然实现不尽相同,但其核心机理是一样的,文中的异常处理策略也可以应用于其他 CPU 的异常处理过程中。

参考文献:

[1] 刘小军,李秀娟. 嵌入式操作系统 VxWorks 的内存管理技术研究[J]. 电子科技,2008,21(6):62-65.

[2] 李玉深,周祖洋,万 杨. 实时操作系统 VxWorks 下的异常处理[J]. 应用科技,2005,32(5):30-32.

[3] Rad-hard 32 bit SPARC V8 processor manual[M]. [s. l.]: Atmel Corporation,2011.

[4] 黄江泉,陈晓敏,赵勋峰. 基于 SPARC 的 VxWorks 异常处理研究[J]. 微计算机信息,2012,28(5):75-77.

[5] VxWorks 5.4 programmer's guide[M]. USA:Wind River Systems,Inc,2000.

[6] 王泽民,芦东昕,谢 鑫,等. 基于 VxWorks 的异常处理的研究和实现[J]. 计算机工程,2005,31(13):90-92.

[7] 王运盛,王 坚. VxWorks 实时操作系统中的中断处理机制分析[J]. 电讯技术,2007,47(4):178-181.

[8] VxWorks BSP developer's guide 6.6[M]. USA:Wind River Systems,Inc,2007.

[9] 孔祥营,柏桂枝. 嵌入式实时操作系统 VxWorks 及其开发环境 Tornado[M]. 北京:中国电力出版社,2001.

[10] 曹桂平. VxWorks 设备驱动开发详解[M]. 北京:电子工业出版社,2011.

[11] The SPARC architecture manual version 8[M]. [s. l.]: SPARC International Inc,1992.

[12] VxWorks kernel programmer's guide 6.6[M]. USA:Wind River Systems,Inc,2007.

[13] 房同忠,杨卉升,张华年,等. 基于 VxWorks 的异常问题分析及调试方法的研究[J]. 工业控制计算机,2012,25(6):23-24.

[14] 何先波,钟乐海,芦冬昕. 嵌入式操作系统封装层的设计与实现[J]. 计算机应用,2003,23(5):89-91.



# 基于SPARC的地址非对齐异常处理研究

作者:

祝晓林, 张健, 李楠, [ZHU Xiao-lin](#), [ZHANG Jian](#), [LI Nan](#)

作者单位:

[祝晓林, ZHU Xiao-lin\(中国科学院大学, 北京 100049; 中国科学院 空间应用工程与技术中心, 北京 100094\), 张健, 李楠, ZHANG Jian, LI Nan\(中国科学院 空间应用工程与技术中心, 北京, 100094\)](#)

刊名:

[计算机技术与发展](#) 

英文刊名:

[Computer Technology and Development](#)

年, 卷(期):

2015(4)

引用本文格式: [祝晓林, 张健, 李楠, ZHU Xiao-lin, ZHANG Jian, LI Nan](#) [基于SPARC的地址非对齐异常处理研究](#)[期刊论文]-[计算机技术与发展](#) 2015(4)