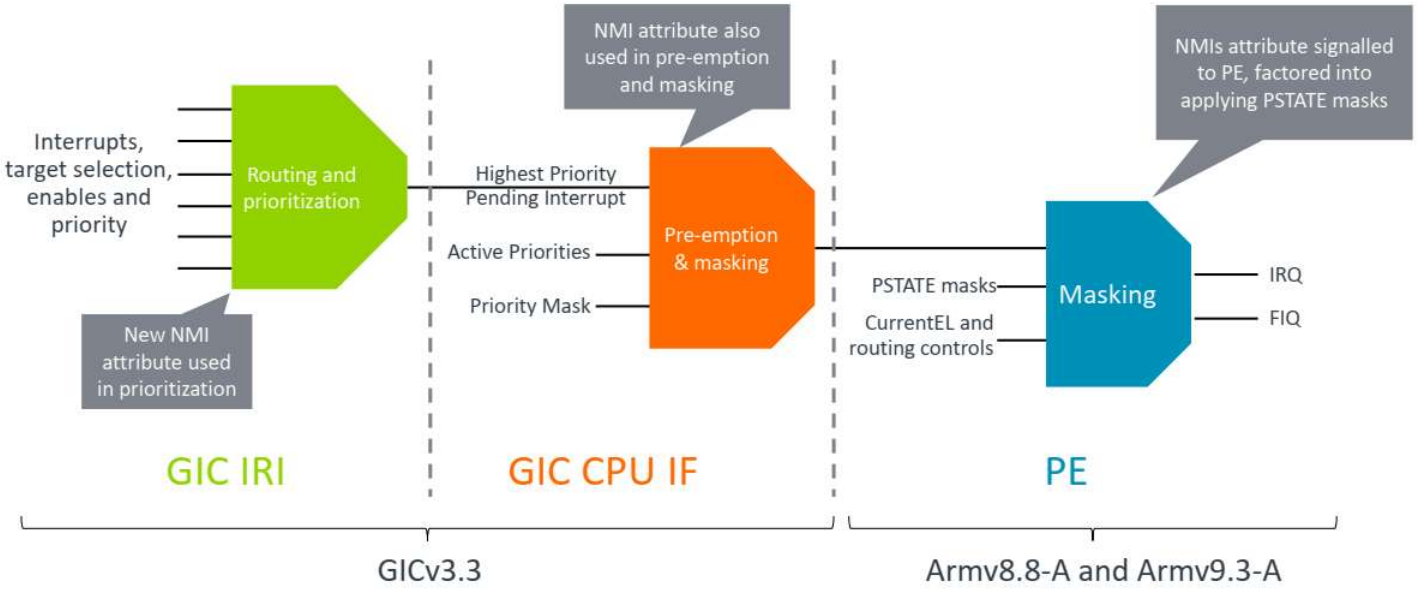
快速链接:

👉👉👉 ARMv8/ARMv9架构入门到精通-[目录] 👈👈👈

- 付费专栏-付费课程 **【购买须知】:**
- 联系方式-加入交流群 ----**联系方式-加入交流群**
- 个人博客笔记导读目录(全部)

目录

Arm A-profile 架构的有一个长期缺陷就是不支持不可屏蔽中断 (NMI) 。2021年，ARM宣布讲支持NMI，所谓支持其实就是看CPU是否支持？GIC是否支持？ 但是究竟什么是 NMI，操作系统软件如何使用这些功能，以及当有多种方法可以屏蔽它们时，为什么它们被称为不可屏蔽？这篇博文更详细地探讨了这些问题。



对 NMI 的需求

让我们首先探讨为什么需要一类特殊的中断以及为什么我们称它们为不可屏蔽。首先要意识到，所有中断在某些情况下都是可屏蔽的。例如，如果中断控制器完全关闭，则不会向 CPU 传递任何中断。术语不可屏蔽中断实际上涵盖了一类中断，即使“正常”中断被屏蔽，仍可以将其传递给 CPU。NMI 仍然可以被屏蔽，但是通过标准内核代码难以访问的单独控制状态。在某些情况下屏蔽所有中断的能力也存在于其他具有悠久 NMI 历史的架构上。

操作系统软件需要仅在特定情况下被屏蔽的单独中断类别有两个主要原因。第一个原因很简单，**中断可能会意外地被禁用**，从而使系统处于无响应状态。乍一看，似乎可以轻松编写软件以在禁用中断后始终启用中断，但事实并非如此。现代操作系统内核通常跨越数百万行可以直接操作中断标志的代码。中断处理例程可能会导致同步异常，从而导致非线性代码路径，并可能导致禁用中断的死锁。还要考虑以完全内核权限运行并能够直接操纵 CPU 中断标志的第三方驱动程序。应该清楚的是，简单的代码检查或静态分析不足以防止中断被禁用。**在一些场景中，能够快速传递中断至关重要，例如调试、跨 PE 同步和热补丁**。

第二个原因是**操作系统依赖中断来支持性能分析**。比如Linux的perf AArch64 上的子系统依赖于 PMU 溢出中断。通过将循环计数器编程为以特定间隔溢出来分析代码路径，并且在每次溢出时，都会将 PMU 中断传递给 CPU。perf 处理中断并对正在运行的进程的 PC 进行采样。当程序员只有一个中断掩码可用时，任何禁用中断的关键部分都不能被分析。用户的一个常见抱怨是他们的分析结果显示在 local_irq_enable() 期间花费了很多 CPU 时间。显然情况并非如此，因为该功能将由操纵 PSTATE.I 标志的单个指令实现。误导性信息是由于 PMU 中断总是在 IRQ 重新启用时传递，并且 perf 总是在此功能上对 PC 进行采样。

为了在引入架构 NMI 支持之前解决这个问题，Linux 依赖于**伪NMI**，它使用了 GIC 架构中的中断优先级特性。Linux 对 PMU 溢出中断进行编程，其优先级高于所有其他中断。这重写了arm64特定的中断启用和禁用函数来更改CPU中断优先级掩码（ICC_PMR_EL1），而不是直接操作CPU IRQ异常标志（PSTATE.I）。提醒一下，Arm 架构将 CPU 异常（IRQ 和 FIQ 异常）与在 GIC 中完成的单个中断的

处理和配置分开。尽管当今大多数系统都使用 GIC，但可以使用不同的中断控制器架构构建基于 Arm 的系统。

使用优先级实现伪 NMI 可以很好地支持 Linux 上的分析，但不幸的是在操作系统的关键路径中引入了额外的软件复杂性。例如，在 KVM 内置的 hypervisor 中进入 guest 虚拟机时，hypervisor 必须仔细执行几个步骤。首先，优先级掩码用于屏蔽非分析中断，以便可以分析 VM 进入路径。其次，在设置 CPU 返回状态以执行向 VM 的异常返回之前，必须使用 PSTATE.I 屏蔽中断以防止破坏异常返回状态。第三，也是最后，必须降低优先级掩码，以允许任何主机中断在稍后的时间点导致从 VM 退出。此流程会在 VM 处理代码的最关键路径中产生开销，最好避免，

最新架构扩展支持 NMI

不可屏蔽有点用词不当，因为 NMI 可以被屏蔽。关键是某些中断必须与其他中断分开屏蔽，并且必须能够防止软件轻易直接屏蔽特定中断。一些操作系统只需要单独的屏蔽，而另一些则需要单独的屏蔽和防止容易屏蔽中断的机制。

在 RISC 风格的架构（如 Arm）上，必须能够在异常进入和异常返回期间屏蔽异常级别内的所有中断。这是因为异常信息存储在 ELR_ELx 和 SPSR_ELx 寄存器中。如果 IRQ 异常在另一个异常的异常处理路径中足够早地传递，这些寄存器将被覆盖。关键信息将丢失，无法恢复。

这可能是提醒读者另一个 Arm 架构特性 PSTATE.SP 的好时机。在对 ELx 产生异常时，目标异常级别的专用堆栈指针 SP_ELx 用于访问堆栈。然后，软件可以选择在初始异常进入后切换到使用 SP_EL0 堆栈指针，例如，如果它想重用用户空间线程堆栈指针来执行内核。这允许使用专用的异常入口堆栈，与在线程上下文中执行分开。通过写入专用寄存器SPSel来切换堆栈指针. 一些操作系统使用此模型，而其他操作系统则不使用。例如，Linux 在分别运行在 EL1 或 EL2 时总是使用专用堆栈指针 SP_EL1 或 SP_EL2，并使用 SP_EL0 作为附加寄存器。

2021 扩展引入了在 GIC 中使用专用 NMI 优先级配置中断的能力，该优先级严格高于其他优先级。（这忽略了跨多个安全状态的优先级。例如，仍然可以将安全世界中断配置为比非安全世界中的 NMI 具有更高的优先级），配置有 NMI 优先级的中断，从这里称为“NMI”，与其他中断分开屏蔽。然后，CPU 架构允许操作系统使用 SCTLR_ELx 系统寄存器中的新选择控件来选择如何屏蔽 NMI。没有 NMI 优先级的正常中断使用现有的 PSTATE.I 位进行屏蔽。

第一种屏蔽模式利用现有的 SP 选择逻辑。这会在异常进入和退出路径期间屏蔽包括 NMI 在内的所有中断，并防止操作系统内核的其余部分屏蔽 NMI。当使用专用 SP_ELx 堆栈指针时，此模式通过屏蔽 NMI 来工作。当操作系统完成异常入口路径并将堆栈指针切换到 SP_EL0 时，只能通过将堆栈指针更改回 SP_ELx 来屏蔽 NMI，这实际上阻止了软件直接屏蔽 NMI。通过使用 Armv8.6 架构扩展中引入的细粒度陷阱，在 EL2 上运行的管理程序可以防止在 EL1 上运行的操作系统更改异常入口路径之外的堆栈指针。

第二种屏蔽模式只是引入了一个单独的屏蔽位 PSTATE.AllInt，它在异常进入期间设置，与设置 PSTATE.I 的方式相同。PSTATE.AllInt 独立于堆栈指针选择，并由软件直接设置和清除，尽管运行在 EL2 的管理程序也可能捕获从 EL1 到 PSTATE.AllInt 的写入。这种屏蔽模式预计将由主要需要 NMI 来

进行分析支持并希望避免实现基于优先级的伪 NMI 的复杂性的操作系统使用。Arm 期望 Linux 内核从这种屏蔽模式中受益。例如，KVM 中的 VM 入口路径会在即将运行 VM 时通过设置 PSTATE.I 来屏蔽正常中断。KVM 随后会在设置异常返回寄存器以进入 VM 之前设置 PSTATE.AllInt。在 EL1 中运行 VM 时，针对 EL2 的物理中断不会被 PSTATE 位屏蔽（它们现在只屏蔽虚拟中断）。通过使用这个新的 PSTATE.AllInt 控件，管理程序避免了必须临时管理优先级掩码和 PSTATE.I，如上一节所述。

2021 扩展还为 GIC 添加了一个新的中断确认寄存器 ICC_NMIAR1_EL1，可用于将 NMI 与其他中断分开确认。引入此功能是为了避免软件无意中从无法处理该中断的上下文中确认中断的情况。例如，当在设置了 PSTATE.I 的关键部分期间发出对电平敏感的 NMI 信号时，可能会发生这种情况，但在软件确认中断之前 NMI 的电平被取消断言。

NMI 的虚拟化支持

2021 扩展支持在 EL2 上运行的虚拟机管理程序的控制下，在 EL1 的虚拟机中运行的来宾操作系统的虚拟 NMI。CPU 架构支持分别为 EL1 和 EL2 选择掩码模式。屏蔽模式分别使用 SCTLR_EL1 和 SCTRL_EL2 为 EL1 和 EL2 选择，PSTATE 中的屏蔽状态自然由管理程序通过 SPSR_EL2 保存和恢复。GIC 的虚拟 CPU 接口获得与物理 CPU 接口相同的能力，用于单独确认和识别虚拟 NMI。

在支持 NMI 的硬件系统上运行的 Hypervisor 软件需要扩展以支持虚拟 NMI。2021 扩展包括通过扩展 EL2 列表寄存器 (LR) 对 NMI 的虚拟化支持，其中包含要传递给 VM 的虚拟中断。LR 获得一个新的 NMI 位，以指示虚拟中断在呈现给虚拟 CPU 接口时具有 NMI 优先级。支持直接注入虚拟 SGI 的 GIC 实现也支持配置状态以直接注入具有 NMI 优先级的虚拟 SGI。

虚拟 GIC 分配器和再分配器由软件中的管理程序模拟。必须使用用于配置中断的 NMI 优先级的新寄存器来扩展软件仿真，最后需要进行一些小的更改以在虚拟功能寄存器中公开对 NMI 的支持。

结论

总之，在 2021 架构扩展中引入 NMI 提供了一个简单的编程模型，可以实现常见的 NMI 用例。在设计此功能时，与生态系统合作，以确保我们能够提供必要的灵活性来满足通用内核和虚拟机管理程序对 NMI 的不同要求。

Armv8/Armv9架构从入门到精通，Armv8/Armv9架构从入门到精通（一期），Armv8/Armv9架构从入门到精通（二期）Armv8/Armv9架构从入门到精通（三期），Arm一期、Arm二期、学习资料、免费、下载，全套资料，Secureboot从入门到精通，secureboot训练营，ATF架构从入门到精通、optee系统精讲、secureboot精讲，Trustzone/TEE/安全快速入门班，Trustzone/TEE/安全标准版，Trustzone/TEE/安全高配版。全套资料。周贺贺，baron，代码改变世界，coding_the_world，Arm精选，arm_2023，安全启动，加密启动

optee、ATF、TF-A、Trustzone、optee3.14、MMU、VMSA、cache、TLB、arm、armv8、armv9、TEE、安全、内存管理、页表，Non-cacheable,Cacheable, non-shareable,inner-shareable,outer-shareable, optee、ATF、TF-A、Trustzone、optee3.14、MMU、VMSA、cache、TLB、arm、armv8、armv9、TEE、安全、内存管理、页表…