1、 总结 RISC-V 体系结构对中断的支持

RISC-V 定义了 4 种中断类型，大体上和 ARM 架构的中断类似，中断类型选取都是通过屏蔽寄存器来控制。

- 外部中断:

来自核心外的中断，常见的 GPIO、UART 中断都属于这个中断。

由 CSR 寄存器 mie 中的 MEIE 控制，等待标志反映在 CSR 寄存器 mip 中的 MEIP 域。

- 定时器中断:

来自定时器的中断。

由 mie 寄存器的 MTIE 域控制，等待标志反映在 mip 寄存器中的 MTIP 域。

- 软件中断:

来自软件自己触发的中断。

由 mie 寄存器中的 MSIE 域控制，等待标志反映在 mip 寄存器的 MSIP 域。

- 调试中断:

用于实现调试器的中断。

RISC-V 定义了一个平台级别中断控制器 PLIC（Platform Level Interrupt Controller），可用于多个外部中断源的优先级仲裁和派发。

PLIC 可以将多个外部中断源仲裁为一个 bit 的中断信号送入处理器核，处理器核接收到中断进入异常服务程序后可以通过读 PLIC 的相关寄存器查看中断源的编号和信息，在处理完响应中断服务程序后可以通过写 PLIC 的相关九年起和外部中断源寄存器来清除中断源。

PLIC 就相当于 RISC-V 上的弱化版 NVIC。

RISC-V 规定系统平台至少有一个定时器，并应配由 2 个 64 位寄存器 mtime（反映当前定时器的计数值）和 mtimecmp（设置计时器比较值），当 mtime 中的计数值大于等于 mtimecmp 中设置的比较值时，计时器就会产生计时器中断，中断期间会一直拉高直到重写 mtimecmp 的值大于 mtime 中的值。

特别地，两个定时器寄存器不归属于 CSR 寄存器，而是定义为存储器地址映射的系统寄存器，由配套 SoC 控制，这样就使得位于内核中的定时器变成了“一半外设”。

这个定时器的时钟必须是为低速（意味着省电）的电源常开（意味着准确稳定）的时钟，它是一种实时计时器。

2、 了解 Linux 的中断处理流程，解释为什么引入上半部和下半部处理?

上半部:完成尽可能少的比较紧急的功能，它往往只是简单的读取寄存器中的中断状态并清除中断标志后就进行“登记中断”(也就是将底半部处理程序挂在到设备的底半部执行队列中的工作)，特点：响应速度快。

下半部:中断处理的大部分工作都在底半部，它几乎做中断处理程序的所有事情，特点：处理相对来说不是非常紧急的事件。

中断处理的一部分过程对及时性要求高不能被打断，并且与硬件相关，而另一部分相对就随意得多，将这两部分分成上半部和下半部有利于更高效地实现中断处理。

3、 系统调用与函数/过程调用的区别是什么?

系统调用

1.使用 INT 和 IRET 指令，内核和应用程序使用的是不同的堆栈，因此存在堆栈的切换，从用户态切换到内核态，从而可以使用特权指令操控设备。

2.依赖于内核，不保证移植性。

3.在用户空间和内核上下文环境间切换，开销较大。

4. 是操作系统的一个入口点。

函数调用

1.使用 CALL 和 RET 指令，调用时没有堆栈切换。

2.平台移植性好。

3.属于过程调用，调用开销较小。

4.一个普通功能函数的调用。