第4季

RISC-V中断处理

本节课主要内容

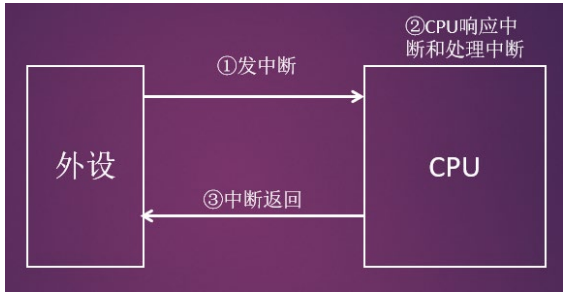
- 本章主要内容
  - 中断处理过程
  - CLINT中断控制器
  - PLIC中断控制器
  - 实验

技术手册：

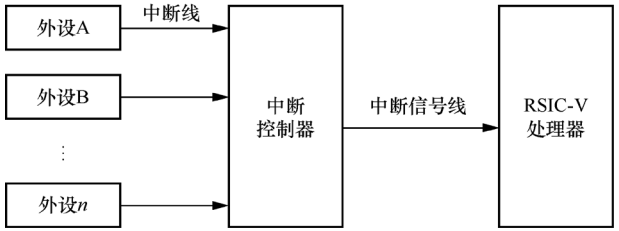
1. The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Document Version 20211203
2. RISC-V Platform-Level Interrupt Controller Specification, Version 1.0
3. SiFive U74-MC Core Complex Manual, 21G2.01.00

本节课主要讲解书上第9章内容

中断基本概念

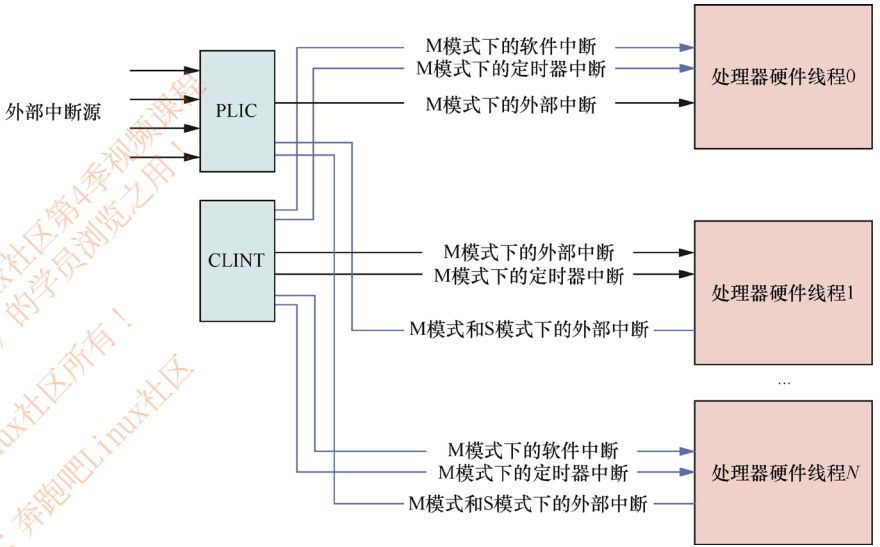


- 中断与轮询（polling）
- 外设通知CPU和OS的一种方式：中断发生的时刻和当前正在执行的指令无关
- 中断模型
- RISC-V架构把中断分成4类：
  - ✓ 软件中断（software interrupt）
  - ✓ 定时器中断（timer interrupt）
  - ✓ 外部中断（external interrupt）
  - ✓ 调试中断（debug interrupt）



本地中断和全局中断

- 本地（local）中断：私有中断，直接发送给CPU
- 全局（global）中断：外设中断，经过中断控制器路由之后发送给CPU

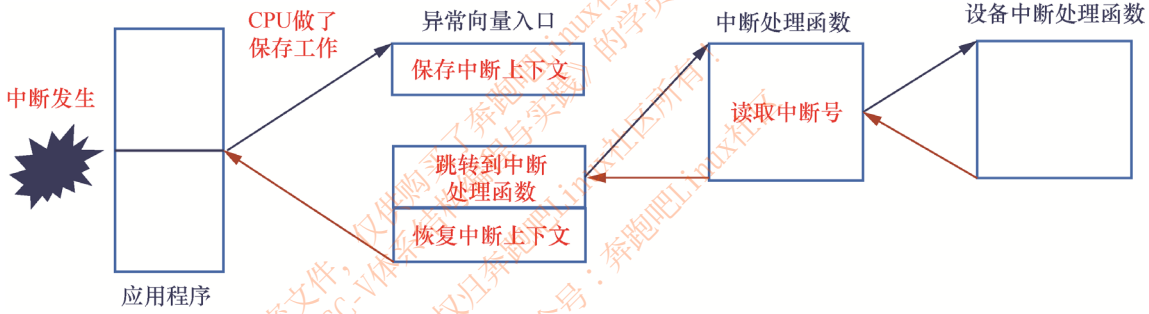


中断处理过程

- 触发中断后，默认情况下由回 M模式响应和处理，也可以委派给S模式处理。
- 中断触发之后，CPU所做的事情与异常处理类似。
  1. PC值 -> sepc寄存器
  2. 中断的类型 -> scause寄存器。
  3. 中断虚拟地址 -> stval寄存器。
  4. SIE字段 -> SPIE字段。
  5. 处理器模式 -> SPP字段。
  6. SIE字段 -> 0
  7. 设置处理器模式为S模式。
  8. 跳转到异常向量表里执行，即PC -> stvec寄存器的值

- OS需要做的事情：
  1. 保存中断上下文
  2. 查询scause寄存器，跳转到合适的中断处理程序中。
  3. 恢复中断上下文
  4. sret返回

- 异常返回CPU要做的事情：
  1. SPIE -> SIE
  2. 从SPP中恢复处理器模式
  3. spec的值 写入到 PC



中断的委托和注入

- 中断委托

表 9.1　　mideleg 寄存器中的字段

字段	位	说明
SSIP	Bit[1]	把软件中断委托给 S 模式
STIP	Bit[5]	把时钟中断委托给 S 模式
SEIP	Bit[9]	把外部中断委托给 S 模式

- 中断注入：mip寄存器用来注入中断到S模式

中断优先级

- 中断类型按优先级从高到低排序：
  - ✓ M模式下的外部中断
  - ✓ M模式下的软件中断
  - ✓ M模式下的定时器中断
  - ✓ S模式下的外部中断
  - ✓ S模式下的软件中断
  - ✓ S模式下的定时器中断

CLINT本地中断控制器

- CLINT支持：
  - ✓ 软件中断
  - ✓ 时钟中断

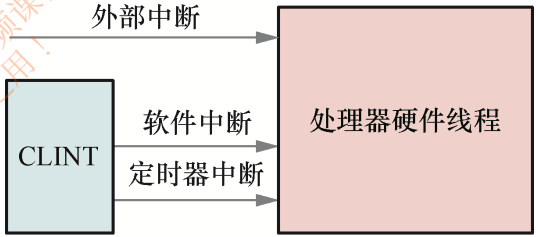


表 9.2 CLINT 支持的中断

名称	中断号	说明
ssip	1	S 模式下的软件中断
msip	3	M 模式下的软件中断
stip	5	S 模式下的时钟中断
mtip	7	M 模式下的时钟中断
seip	9	S 模式下的外部中断
meip	11	M 模式下的外部中断

- mtimer定时器在CLINT中设置
  - ✓ MTIMECMP寄存器用来设置时间间隔，当MTIME返回的时间大于或者等于MTIMECMP寄存器的值时便会触发定时器中断。

表 9.3　　CLINT 中的寄存器

名称	地址	属性	位宽	描述
MSIP	0x200 0000	RW	32	机器特权模式下的软件触发寄存器，用于处理器硬件线程 0
MSIP	0x200 0004	RW	32	机器特权模式下的软件触发寄存器，用于处理器硬件线程 1
MSIP	0x200 0008	RW	32	机器特权模式下的软件触发寄存器，用于处理器硬件线程 2
MSIP	0x200 000C	RW	32	机器特权模式下的软件触发寄存器，用于处理器硬件线程 3
MSIP	0x200 0010	RW	32	机器特权模式下的软件触发寄存器，用于处理器硬件线程 4
MTIMECMP	0x200 4000	RW	64	定时器比较寄存器，用于处理器硬件线程 0
MTIMECMP	0x200 4008	RW	64	定时器比较寄存器，用于处理器硬件线程 1
MTIMECMP	0x200 4010	RW	64	定时器比较寄存器，用于处理器硬件线程 2
MTIMECMP	0x200 4018	RW	64	定时器比较寄存器，用于处理器硬件线程 3
MTIMECMP	0x200 4020	RW	64	定时器比较寄存器，用于处理器硬件线程 4
MTIME	0x200 BFF8	RW	64	定时器寄存器

RISC-V架构中的mtimer缺点：每次设置定时器mtimercmp寄存器都需要陷入到M模式，导致系统性能下降

案例分析1：定时器中断

```
BenOS image layout:
  .text.boot: 0x80200000 - 0x80200044 (    68 B)
       .text: 0x80200048 - 0x80203178 ( 12592 B)
     .rodata: 0x80203178 - 0x80204b00 (  6536 B)
       .data: 0x80204b00 - 0x80206000 (  5376 B)
        .bss: 0x80206020 - 0x80226488 (132200 B)
sstatus:0x0
sstatus:0x2
Core0 Timer interrupt received, jiffies=1
Core0 Timer interrupt received, jiffies=2
Core0 Timer interrupt received, jiffies=3
Core0 Timer interrupt received, jiffies=4
Core0 Timer interrupt received, jiffies=5
Core0 Timer interrupt received, jiffies=6
Core0 Timer interrupt received, jiffies=7
Core0 Timer interrupt received, jiffies=8
Core0 Timer interrupt received, jiffies=9
Core0 Timer interrupt received, jiffies=10
```

早期的中断控制器

- 传统的中断控制器，例如树莓派4b上的legacy interrupt controller
  - ✓ 中断enable寄存器
  - ✓ 中断disable寄存器
  - ✓ 中断状态寄存器

- 传统的使用简单状态寄存器的方式来管理中断，变得越来越难管理
  - 中断源变得原来越多
  - 不同类型的中断，比如多核间的中断，中断优先级，软件定义的中断等

三星2410上的中断控制器：
通过几个简单的寄存器来控制这个中断控制器：
中断pending寄存器：每个bit位表示一个中断源
中断mask寄存器：用来屏蔽某个中断源

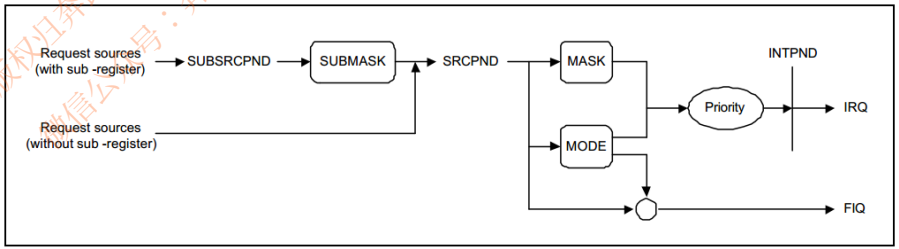


Figure 14-1. Interrupt Process Diagram

ARM阵营：GIC中断控制器发展历史

- 在Cortex-A时期就开发了GIC中断控制器

GICV1：
- ✓ 支持8核
- ✓ 支持多大1020个中断源
- ✓ 8bit优先级
- ✓ 支持软件触发中断
- ✓ TrustZone支持

IP：
GIC-390

应用场景：
Cortex-A9 MPCore

GICV2：
- ✓ 虚拟化支持
- ✓ 支持secure software

IP：
GIC-400

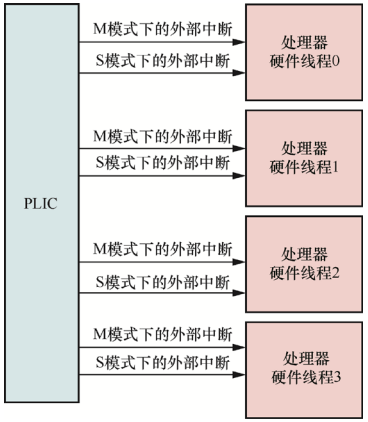
应用场景：
Cortex-A7 MPCore
树莓派4b

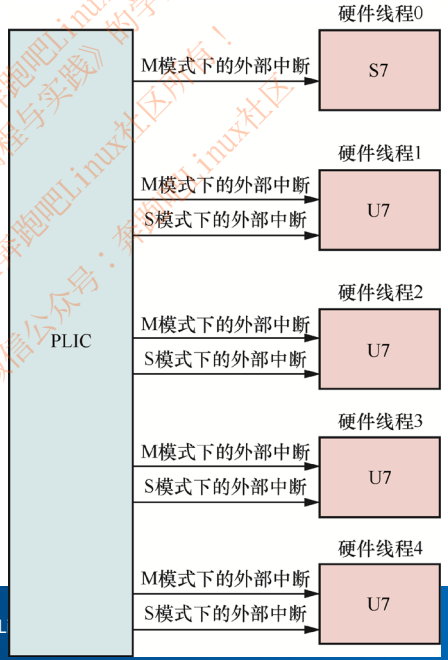
GICV3：
- ✓ 支持CPU核心数量大于8
- ✓ 支持基于消息的中断
- ✓ 支持更多的中断ID

IP：
GIC-500
GIC-600

RISC-V阵营： PLIC中断控制器

- 简化版的GIC
- 最多可以支持1024个中断源与15 872个中断硬件上下文（context）
- FU740处理器，它支持5个处理器内核，并且不支持SMT。一共支持9个中断硬件上下文。
  - ✓ S7处理器硬件线程0的M模式；
  - ✓ U7处理器硬件线程1的M模式；
  - ✓ U7处理器硬件线程1的S模式；
  - ✓ U7处理器硬件线程2的M模式；
  - ✓ U7处理器硬件线程2的S模式；
  - ✓ U7处理器硬件线程3的M模式；
  - ✓ U7处理器硬件线程3的S模式；
  - ✓ U7处理器硬件线程4的M模式；
  - ✓ U7处理器硬件线程4的S模式。





中断号分配

- 每一款SoC的中断号分配不一样
- FU740处理器中的PLIC支持69个中断源以及7级中断优先级
- QEMU Virt开发板中断分配

表 9.4 FU740 处理器的中断号分配

中断号	描述
1～10	MSI
11～18	DMA
19	L2 高速缓存目录错误
20	L2 高速缓存目录失败
21	L2 高速缓存数据错误
22	L2 高速缓存数据失败
23～38	GPIO
39	串口 0
40	串口 1
41	SPI 0
42	SPI 1
43	SPI 2
44～47	PWM 0
48～51	PWM 1
52	I2C 0
53	I2C 1
54	DDR
55	MAC
56～64	PCIE
65～69	总线错误

表 2.2 QEMU Virt 模拟的开发板的中断号分配

中断类型	中断号
VIRTIO_IRQ	1～8
UART0	10
RTC	11
PCIe	32～35

中断优先级

- PLIC支持7级中断优先级。0表示不会触发中断或者关闭中断，1表示最低中断优先级，7表示最高中断优先级
- 如果两个相同优先级的中断同时触发，那么编号小的中断具有较高优先级。
- PLIC为每个中断号提供一个寄存器来设置相应中断的优先级
- 中断优先级寄存器的基地址为0xC0 0000。当中断号为N时，中断优先级寄存器的地址为0xC0 0000 + 4 N。

表 9.5 使用寄存器来设置相应中断的优先级

字段	位	属性	说明
Priority	Bit[2:0]	RW	设置中断优先级
Reserved	Bit[31:3]	RO	保留

中断使能寄存器

- 为每个处理器内核使能和关闭中断源
- 每位用来表示一个中断源。一个寄存器就可以表示32个中断源
- PLIC最多支持1024个中断源，一共需要128个字节来管理中断源
- FU740处理器，只支持53个有效的中断源，只需要2个寄存器

表 9.6　FU740 处理器中 PLIC 的中断使能寄存器的布局

地址	位宽	属性	说明
0xC00 2000	4	RW	处理器硬件线程 0 中 M 模式下的中断使能寄存器 0
0xC00 2004	4	RW	处理器硬件线程 0 中 M 模式下的中断使能寄存器 1
0xC00 2080	4	RW	处理器硬件线程 1 中 M 模式下的中断使能寄存器 0
0xC00 2084	4	RW	处理器硬件线程 1 中 M 模式下的中断使能寄存器 1
0xC00 2100	4	RW	处理器硬件线程 1 中 S 模式下的中断使能寄存器 0
0xC00 2104	4	RW	处理器硬件线程 1 中 S 模式下的中断使能寄存器 1
0xC00 2180	4	RW	处理器硬件线程 2 中 M 模式下的中断使能寄存器 0
0xC00 2184	4	RW	处理器硬件线程 2 中 M 模式下的中断使能寄存器 1
0xC00 2200	4	RW	处理器硬件线程 2 中 S 模式下的中断使能寄存器 0
0xC00 2204	4	RW	处理器硬件线程 2 中 S 模式下的中断使能寄存器 1
0xC00 2280	4	RW	处理器硬件线程 3 中 M 模式下的中断使能寄存器 0
0xC00 2284	4	RW	处理器硬件线程 3 中 M 模式下的中断使能寄存器 1
0xC00 2300	4	RW	处理器硬件线程 3 中 S 模式下的中断使能寄存器 0
0xC00 2304	4	RW	处理器硬件线程 3 中 S 模式下的中断使能寄存器 1
0xC00 2380	4	RW	处理器硬件线程 4 中 M 模式下的中断使能寄存器 0
0xC00 2384	4	RW	处理器硬件线程 4 中 M 模式下的中断使能寄存器 1
0xC00 2400	4	RW	处理器硬件线程 4 中 S 模式下的中断使能寄存器 0
0xC00 2404	4	RW	处理器硬件线程 4 中 S 模式下的中断使能寄存器 1

中断待定寄存器

- 表示每个中断源的待定状态
- 每位用来表示一个中断源。一个寄存器就可以表示32个中断源
- PLIC最多支持1024个中断源，一共需要128个字节来管理中断源
- FU740处理器，只支持53个有效的中断源，只需要2个寄存器

中断优先级阈值寄存器

- 用来屏蔽所有优先级小于或等于阈值的中断源，只有当中断优先级大于该中断源的优先级阈值时，PLIC才会处理该中断。
- FU740处理器，它有9个中断硬件上下文

表 9.7 FU740 的中断优先级阈值寄存器的布局

地址	位宽	属性	说明
0xC20 0000	4	RW	处理器超线程 0 中 M 模式下的中断优先级阈值寄存器
0xC20 1000	4	RW	处理器超线程 1 中 M 模式下的中断优先级阈值寄存器
0xC20 2000	4	RW	处理器超线程 1 中 S 模式下的中断优先级阈值寄存器
0xC20 3000	4	RW	处理器超线程 2 中 M 模式下的中断优先级阈值寄存器
0xC20 4000	4	RW	处理器超线程 2 中 S 模式下的中断优先级阈值寄存器
0xC20 5000	4	RW	处理器超线程 3 中 M 模式下的中断优先级阈值寄存器
0xC20 6000	4	RW	处理器超线程 3 中 S 模式下的中断优先级阈值寄存器
0xC20 7000	4	RW	处理器超线程 4 中 M 模式下的中断优先级阈值寄存器
0xC20 8000	4	RW	处理器超线程 4 中 S 模式下的中断优先级阈值寄存器

中断请求/完成寄存器

- 中断请求（claim）寄存器和中断完成（complete）寄存器是同一个寄存器。
  - ✓ 当触发中断时，软件通过读取中断请求寄存器可知哪个中断源是当前待定中断源中优先级最高的，软件可以在中断处理程序中处理该中断。
  - ✓ 中断处理程序执行完后，软件可以把中断号写入中断完成寄存器中，PLIC便知道软件已经完成了中断处理，PLIC完成最后的中断处理工作。

表 9.8　　FU740 处理器的中断请求/完成寄存器的布局

地址	位宽	属性	说明
0xC20 0004	4	RW	处理器超线程 0 中 M 模式下的中断请求/完成寄存器
0xC20 1004	4	RW	处理器超线程 1 中 M 模式下的中断请求/完成寄存器
0xC20 2004	4	RW	处理器超线程 1 中 S 模式下的中断请求/完成寄存器
0xC20 3004	4	RW	处理器超线程 2 中 M 模式下的中断请求/完成寄存器
0xC20 4004	4	RW	处理器超线程 2 中 S 模式下的中断请求/完成寄存器
0xC20 5004	4	RW	处理器超线程 3 中 M 模式下的中断请求/完成寄存器
0xC20 6004	4	RW	处理器超线程 3 中 S 模式下的中断请求/完成寄存器
0xC20 7004	4	RW	处理器超线程 4 中 M 模式下的中断请求/完成寄存器
0xC20 8004	4	RW	处理器超线程 4 中 S 模式下的中断请求/完成寄存器

中断处理 case 2：使能PLIC中断来接收串口数据

- 使用串口接收中断来接收串口数据

```
BenOS image layout:
  .text.boot: 0x80200000 - 0x80200044 (    68 B)
       .text: 0x80200048 - 0x80203598 ( 13648 B)
     .rodata: 0x80203598 - 0x80205100 (  7016 B)
       .data: 0x80205100 - 0x80207000 (  7936 B)
        .bss: 0x80207020 - 0x80227488 (132200 B)
sstatus:0x2
sstatus:0x2, sie:0x222
handle_uart_irq occurred
handle_uart_irq occurred
handle_uart_irq occurred
handle_uart_irq occurred
```

接收到回车后输出“handle_uart_irq occurred”

实验1：定时器中断

1．实验目的

熟悉RISC-V处理器的中断流程。

2．实验要求

在QEMU虚拟机中实现mtimer中断处理，并且在QEMU虚拟机上单步调试和观察。

实验2：使用汇编函数保存和恢复中断现场

1．实验目的

熟悉如何保存和恢复中断现场。

2．实验要求

（1）在实验1的基础上，把kernel_entry和kernel_exit两个宏修改成使用汇编函数实现。当修改成用汇编函数实现时，需要注意什么地方？

（2）请使用QEMU虚拟机和GDB或者Eclipse来单步调试中断处理过程，重点观察保存中断现场和恢复中断现场的寄存器的变化以及栈的变化情况。

实验3：实现并调试串口0中断

1．实验目的

熟悉PLIC中断控制器。

2．实验要求

在BenOS中案例分析2的串口中断，并用GDB来单步调试中断处理过程。