

循序渐进，学习开发一个 RISC-V 上的操作系统



第 11 章 外部设备中断

汪辰

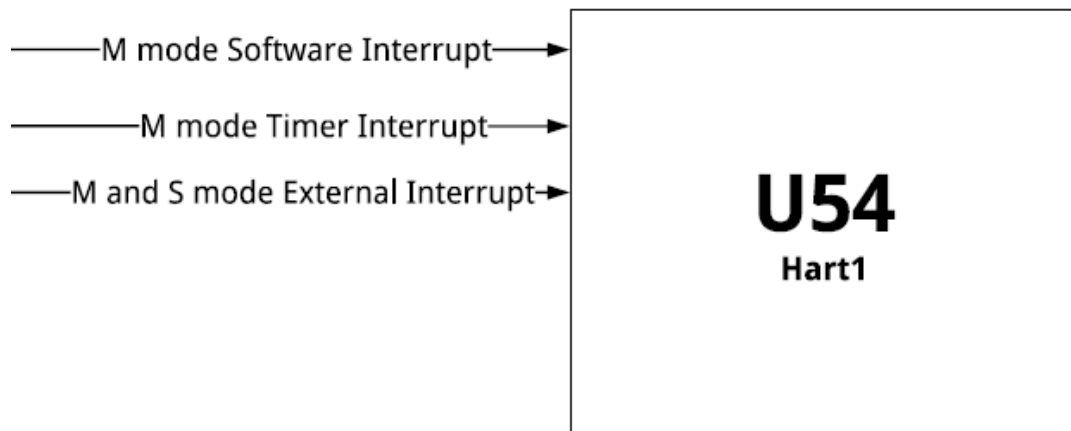
- RISC-V 中断（Interrupt）的分类
- RISC-V 中断编程中涉及的寄存器
- RISC-V 中断处理流程
- PLIC 介绍

- 【参考 1】 : The RISC-V Instruction Set Manual , Volume I: Unprivileged ISA , Document Version 20191213
- 【参考 2】 : The RISC-V Instruction Set Manual , Volume II: Privileged Architecture , Document Version 20190608-Priv-MSU-Ratified
- 【参考 3】 : SiFive FU540-C000 Manual , v1p0
- 【参考 4】 : RISC-V Platform-Level Interrupt Controller Specification :
<https://github.com/riscv/riscv-plic-spec>

- **RISC-V 中断（Interrupt）的分类**
- RISC-V 中断编程中涉及的寄存器
- RISC-V 中断处理流程
- PLIC 介绍

RISC-V 中断（Interrupt）的分类

- 本地（Local）中断
 - software interrupt
 - timer interrupt
- 全局（Global）中断
 - external interrupt



Interrupt	Exception Code	Description
1	0	User software interrupt
1	1	Supervisor software interrupt
1	2	<i>Reserved for future standard use</i>
1	3	Machine software interrupt
1	4	User timer interrupt
1	5	Supervisor timer interrupt
1	6	<i>Reserved for future standard use</i>
1	7	Machine timer interrupt
1	8	User external interrupt
1	9	Supervisor external interrupt
1	10	<i>Reserved for future standard use</i>
1	11	Machine external interrupt
1	12–15	<i>Reserved for future standard use</i>
1	≥16	<i>Reserved for platform use</i>

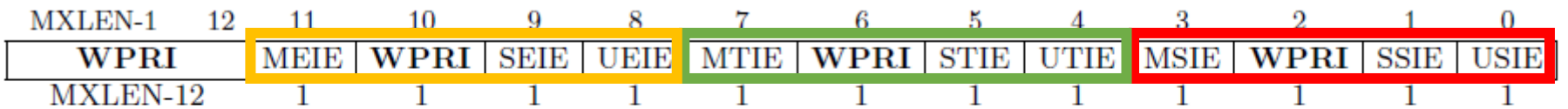
【参考 2】Table 3.6: Machine cause register (mcause) values after trap.

【参考 3】 Figure 3: FU540-C000 Interrupt Architecture Block Diagram.

- RISC-V 中断（Interrupt）的分类
- RISC-V 中断编程中涉及的寄存器
- RISC-V 中断处理流程
- PLIC 介绍

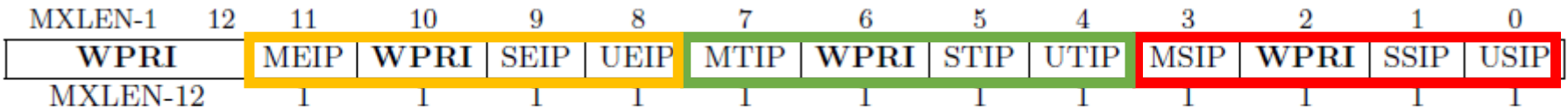
寄存器	用途说明
mtvec （ Machine Trap-Vector Base-Address ）	它保存发生异常时处理器需要跳转到的地址。
mepc （ Machine Exception Program Counter ）	当 trap 发生时，hart 会将发生 trap 所对应的指令的地址值（pc）保存在 mepc 中。
mcause （ Machine Cause ）	当 trap 发生时，hart 会设置该寄存器通知我们 trap 发生的原因。
mtval （ Machine Trap Value ）	它保存了 exception 发生时的附加信息：譬如访问地址出错时的地址信息、或者执行非法指令时的指令本身，对于其他异常，它的值为 0。
mstatus （ Machine Status ）	用于跟踪和控制 hart 的当前操作状态（特别地，包括关闭和打开全局中断）。
mscratch （ Machine Scratch ）	Machine 模式下专用寄存器，我们可以自己定义其用法，譬如用该寄存器保存当前在 hart 上运行的 task 的上下文（context）的地址。
mie （ Machine Interrupt Enable ）	用于进一步控制（打开和关闭）software interrupt/timer interrupt/external interrupt
mip （ Machine Interrupt	它列出目前已发生等待处理的中断。

- mie(Machine Interrupt Enable)：打开（1）或者关闭（0） M/S/U 模式下对应的 External/Timer/Software 中断



【参考 Figure 3.12: Machine interrupt-enable register (mie).

- mip(Machine Interrupt Pending)^{2]}：获取当前 M/S/U 模式下对应的 External/Timer/Software 中断是否发生



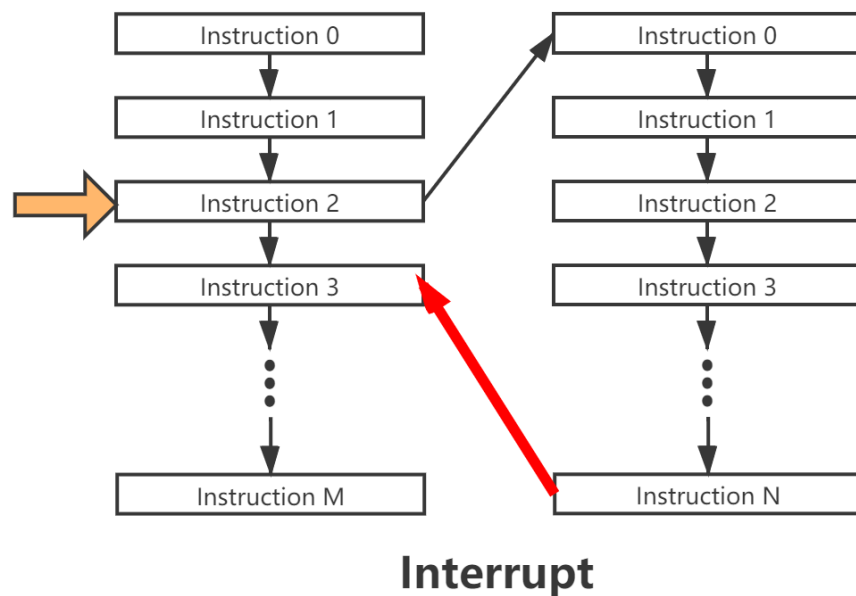
【参考 Figure 3.11: Machine interrupt-pending register (mip).

2]

- RISC-V 中断（Interrupt）的分类
- RISC-V 中断编程中涉及的寄存器
- **RISC-V 中断处理流程**
- PLIC 介绍

中断发生时 Hart 自动执行如下状态转换

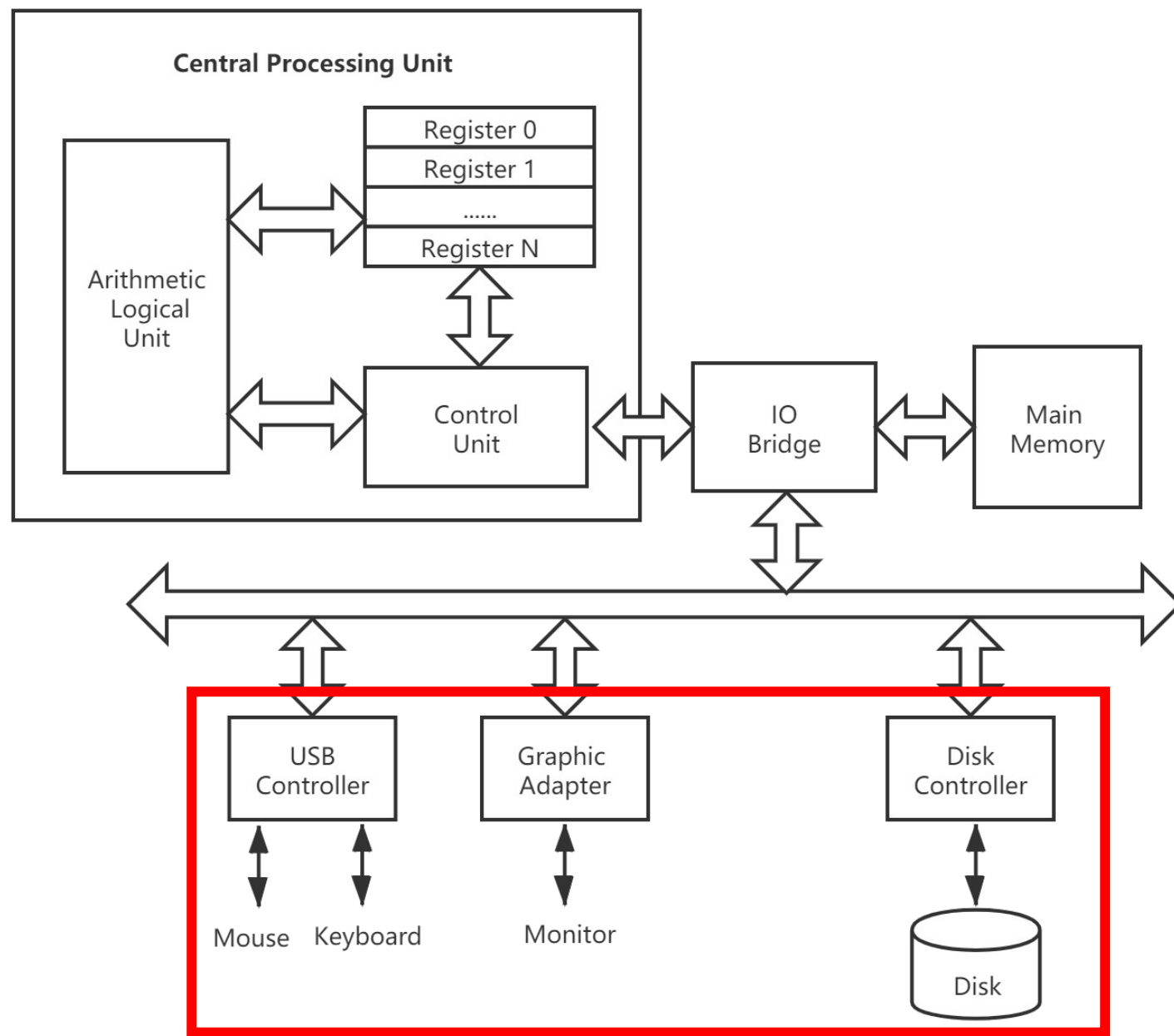
- 把 **mstatus** 的 **MIE** 值复制到 **MPIE** 中，清除 **mstatus** 中的 **MIE** 标志位，效果是中断被禁止。
- 当前的 **PC** 的下一条指令地址被复制到 **mepc** 中，同时 **PC** 被设置为 **mtvec**。注意如果我们设置 **mtvec.MODE = vetcored**， $PC = mtvec.BASE + 4 \times \text{exception-code}$ 。
- 根据 **interrupt** 的种类设置 **mcause**，并根据需要为 **mtval** 设置附加信息。
- 将 **trap** 发生之前的权限模式保存在 **mstatus** 的 **MPP** 域中，再把 **hart** 权限模式更改为 **M**。



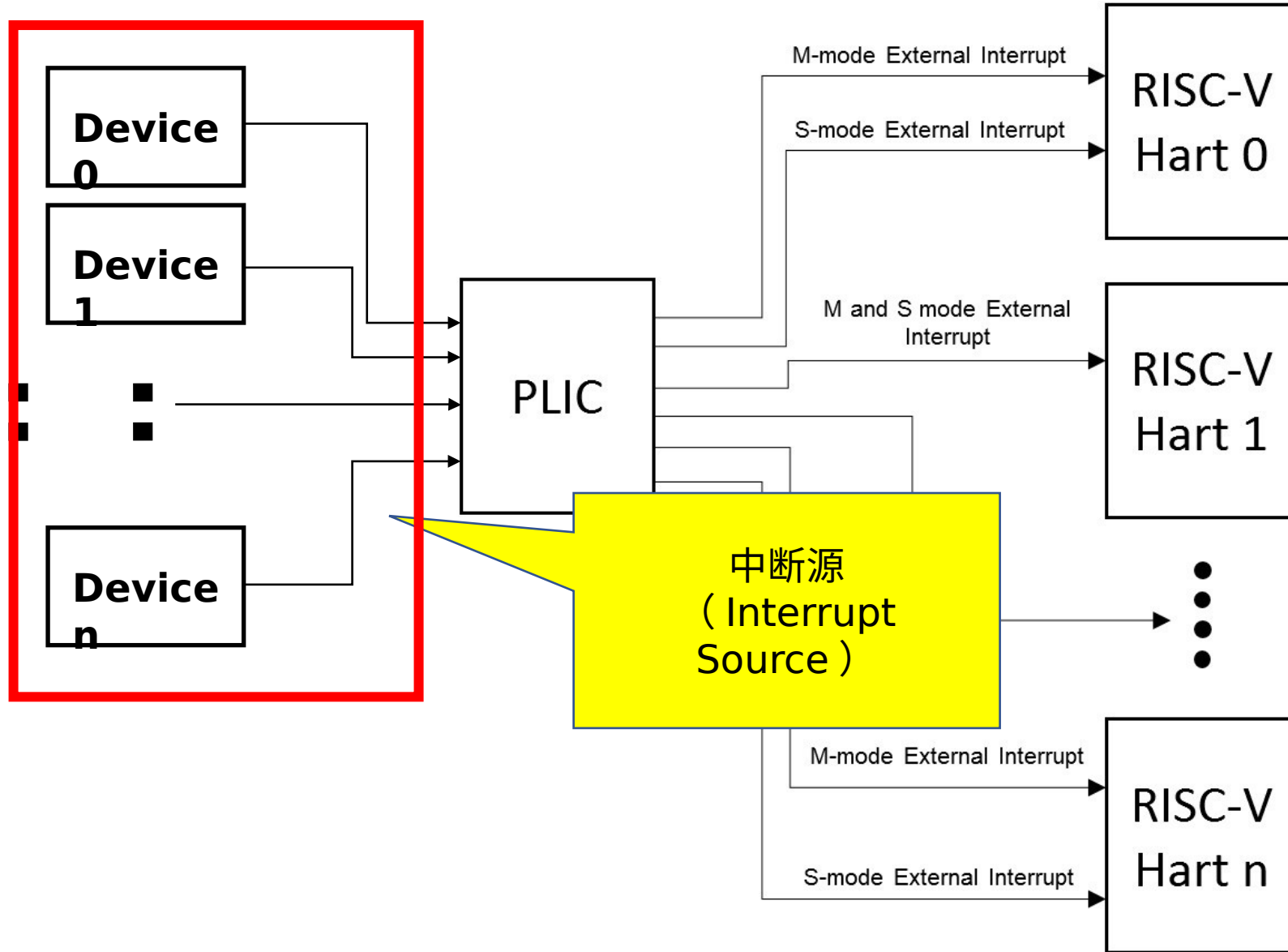
- 以在 M 模式下执行 mret 指令为例，会执行如下操作：
 - 当前 Hart 的权限级别 = mstatus.MPP; mstatus.MPP = U (如果 hart 不支持 U 则为 M)
 - mstatus.MIE = mstatus.MPIE; mstatus.MPIE = 1
 - pc = mepc

- RISC-V 中断（Interrupt）的分类
- RISC-V 中断编程中涉及的寄存器
- RISC-V 中断处理流程
- **PLIC 介绍**

外部中断 (external interrupt)



Platform-Level Interrupt Controller



<https://github.com/qemu/qemu/blob/master/include/hw/riscv/virt.h>

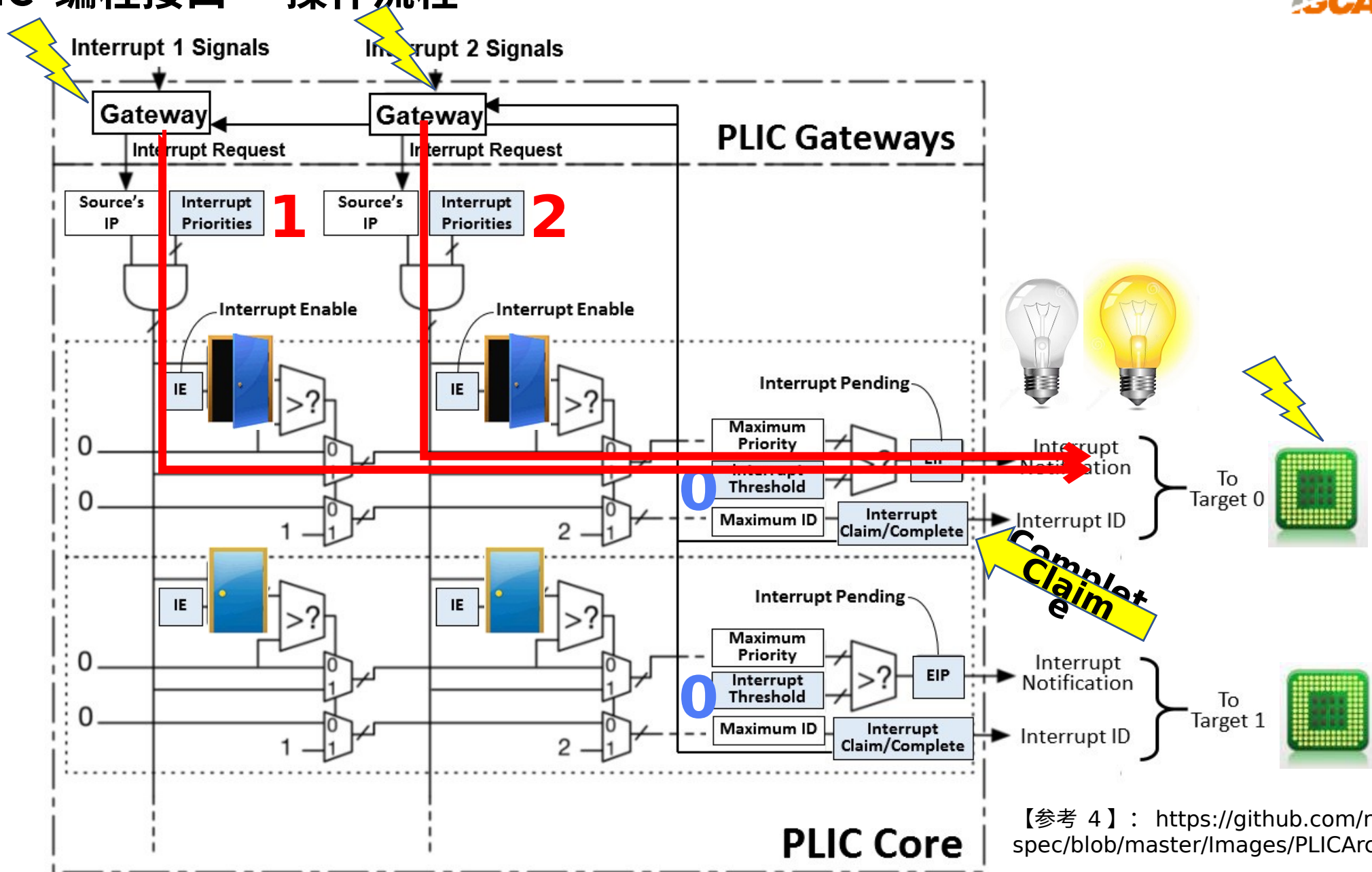
```
enum {  
    UART0_IRQ = 10,  
    RTC_IRQ = 11,  
    VIRTIO_IRQ = 1, /* 1 to 8 */  
    VIRTIO_COUNT = 8,  
    PCIE_IRQ = 0x20, /* 32 to 35 */  
    VIRTIO_NDEV = 0x35 /* Arbitrary maximum number of interrupts */  
};
```

- Interrupt Source ID 范围：1 ~ 53 （0x35）
- 0 预留不用

- RISC-V 规范规定，PLIC 的寄存器编址采用内存映射（memory map）方式。每个寄存器的宽度为 32-bit。
- 具体寄存器编址采用 base + offset 的格式，且 base 由各个特定 platform 自己定义。针对 QEMU-virt，其 PLIC 的设计参考了 FU540-C000，base 为 0x0c000000。

可编程寄存器	功能描述	内存映射地址
Priority	设置某一路中断源的优先级。	$\text{BASE} + (\text{interrupt-id}) * 4$
Pending	用于指示某一路中断源是否发生。	$\text{BASE} + 0x1000 + ((\text{interrupt-id}) / 32)$
Enable	针对某个 hart 开启或者关闭某一路中断源。	$\text{BASE} + 0x2000 + (\text{hart}) * 0x80$
Threshold	针对某个 hart 设置中断源优先级的阈值。	$\text{BASE} + 0x200000 + (\text{hart}) * 0x1000$
Claim/ Complete	对该寄存器执行读操作称之为 Claim，即获取当前发生的最高优先级的中断源 ID 对该寄存器执行写操作称之为 Complete。所谓 Complete 指的是通知 PLIC 对某路中断的处理	$\text{BASE} + 0x200004 + (\text{hart}) * 0x1000$

PLIC 编程接口 - 操作流程



【参考 4】： <https://github.com/riscv/riscv-plic-spec/blob/master/Images/PLICArch.jpg>

谢谢

欢迎交流合作