

# 循序渐进，学习开发一个 RISC-V 上的操作系统



## 第 10 章 Trap 和 Exception

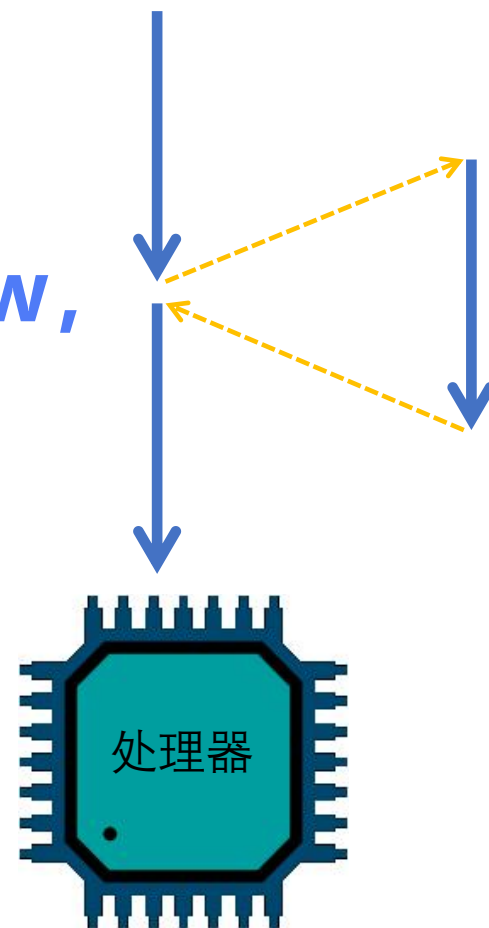
汪辰

- 控制流 (Control Flow) 和 Trap
- RISC-V Trap 处理中涉及的寄存器
- RISC-V Trap 处理流程

- 【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

- **控制流 (Control Flow) 和 Trap**
- **RISC-V Trap 处理中涉及的寄存器**
- **RISC-V Trap 处理流程**

- 控制流 (Control Flow)
  - branch, jump
- 异常控制流 (Exceptional Control Flow, 简称 ECP)
  - exception
  - interrupt
- RISC-V 把 ECP 统称为 Trap



- 控制流 (Control Flow) 和 Trap
- **RISC-V Trap 处理中涉及的寄存器**
- RISC-V Trap 处理流程
- RISC-V Trap 代码实现

## ➤ Machine 模式下的 CSR 列表

Number	Privilege	Name	Description
Machine Information Registers			
0xF11	MRO	mvendorid	Vendor ID.
0xF12	MRO	marchid	Architecture ID.
0xF13	MRO	mimpid	Implementation ID.
0xF14	MRO	mhartid	Hardware thread ID.
Machine Trap Setup			
0x300	MRW	mstatus	Machine status register.
0x301	MRW	misa	ISA and extensions
0x302	MRW	medeleg	Machine exception delegation register.
0x303	MRW	mideleg	Machine interrupt delegation register.
0x304	MRW	mie	Machine interrupt-enable register.
0x305	MRW	mtvec	Machine trap-handler base address.
0x306	MRW	mcounteren	Machine counter enable.
Machine Trap Handling			
0x340	MRW	mscratch	Scratch register for machine trap handlers.
0x341	MRW	mepc	Machine exception program counter.
0x342	MRW	mcause	Machine trap cause.
0x343	MRW	mtval	Machine bad address or instruction.
0x344	MRW	mip	Machine interrupt pending.
Machine Memory Protection			
0x3A0	MRW	pmpcfg0	Physical memory protection configuration.
		⋮	
0x3BF	MRW	pmpaddr15	Physical memory protection address register.

【参考 2】Table 2.4: Currently allocated RISC-V machine-level CSR addresses.

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 Pending)	它列出目前已发生等待处理的中断。



# mtvec (Machine Trap-Vector Base-Address)



【参考 2】 Figure 3.8: Machine trap-vector

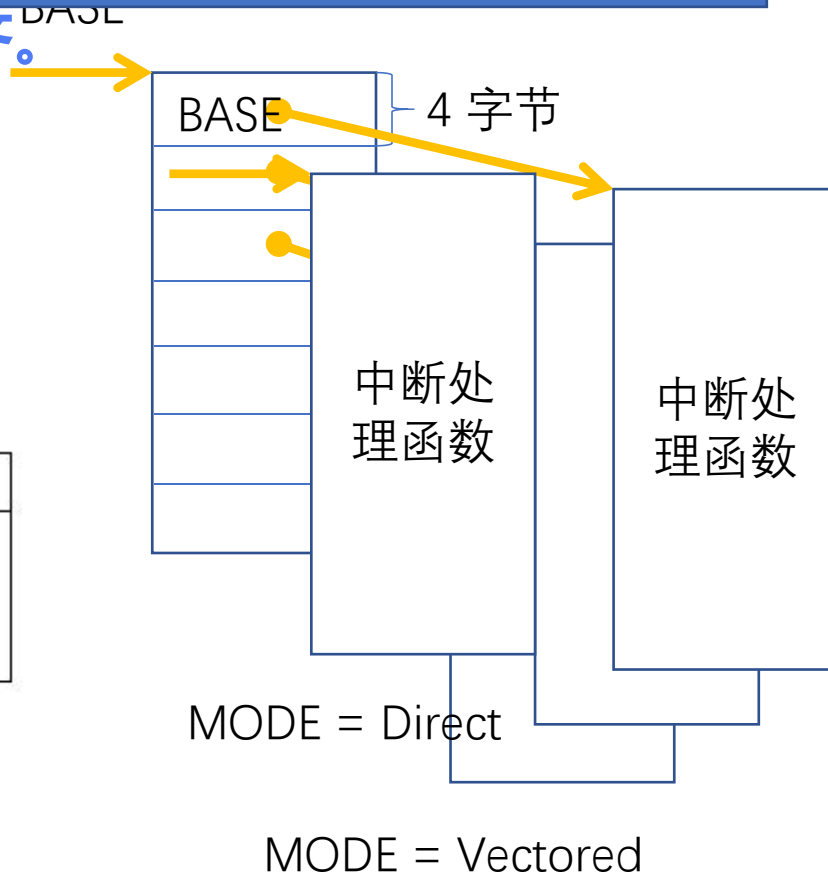
Write Any Values, Reads Legal Values (WARL)

【参考2】 2.3 CSR Field Specifications

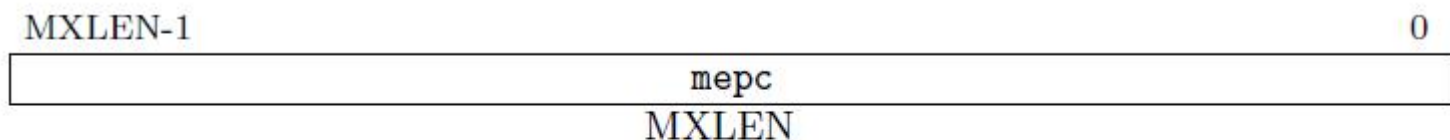
- **BASE:** trap 入口函数的基地址，必须保证四字节对齐。
- **MODE:** 进一步用于控制入口函数的地址配置方式：
  - **Direct:** 所有的 exception 和 interrupt 发生后 PC 都跳转到 BASE 指定的地址处。
  - **Vectored:** exception 处理方式同 Direct; 但 interrupt 的入口地址以数组方式排列。

Value	Name	Description
0	Direct	All exceptions set pc to BASE.
1	Vectored	Asynchronous interrupts set pc to BASE+4×cause.
≥2	—	Reserved

【参考 2】 Table 3.5: Encoding of mtvec MODE field.



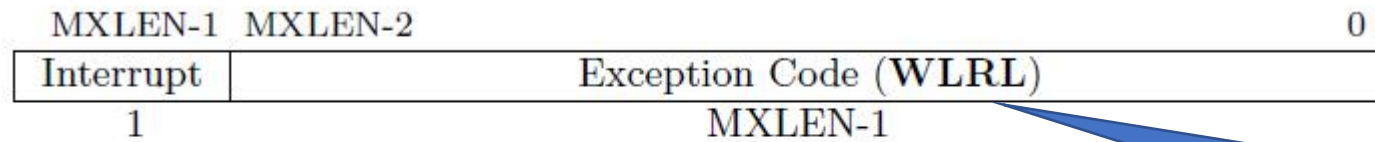
# mepc (Machine Exception Program Counter)



【参考 2】Figure 3.21: Machine exception program counter register.

- 当 trap 发生时，pc 会被替换为 mtvec 设定的地址，同时 hart 会设置 mepc 为当前指令或者下一条指令的地址，当我们需要退出 trap 时可以调用特殊的 mret 指令，该指令会将 mepc 中的值恢复到 pc 中（实现返回的效果）。
- 在处理 trap 的程序中我们可以修改 mepc 的值达到改变 mret 返回地址的目的。

# mcause (Machine Cause)



【参考 2】 Figure 3.22: Machine Cause register mca

Write/Read Only Legal Values (WLRL)  
【参考2】 2.3 CSR Field Specifications

- 当 trap 发生时，hart 会设置该寄存器通知我们 trap 发生的原因。
- 最高位 Interrupt 为 1 时标识了当前 trap 为 interrupt，否则是 exception。
- 剩余的 Exception Code 用于标识具体的 interrupt 或者 exception 的种类。

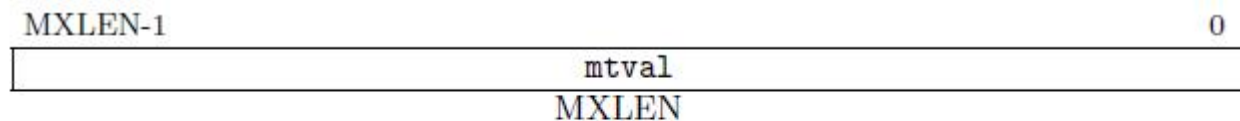
# mcause (Machine Cause)

中断

异常

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>
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode
0	9	Environment call from S-mode
0	10	<i>Reserved</i>
0	11	Environment call from M-mode
0	12	Instruction page fault
0	13	Load page fault
0	14	<i>Reserved for future standard use</i>
0	15	Store/AMO page fault
0	16–23	<i>Reserved for future standard use</i>
0	24–31	<i>Reserved for custom use</i>
0	32–47	<i>Reserved for future standard use</i>
0	48–63	<i>Reserved for custom use</i>
0	≥64	<i>Reserved for future standard use</i>

# mtval (Machine Trap Value)



【参考 2】 Figure 3.23: Machine Trap Value register.

- 当 trap 发生时，除了通过 mcause 可以获取 exception 的种类 code 值外，hart 还提供了 mtval 来提供 exception 的其他信息来辅助我们执行更进一步的操作。
- 具体的辅助信息由特定的硬件实现定义，RISC-V 规范没有定义具体的值。但规范定义了一些行为，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等，具体阅读【参考2】。

# mstatus (Machine Status)



【参考 2】 Figure 3.6: Machine-mode status register (mstatus) for RV32.

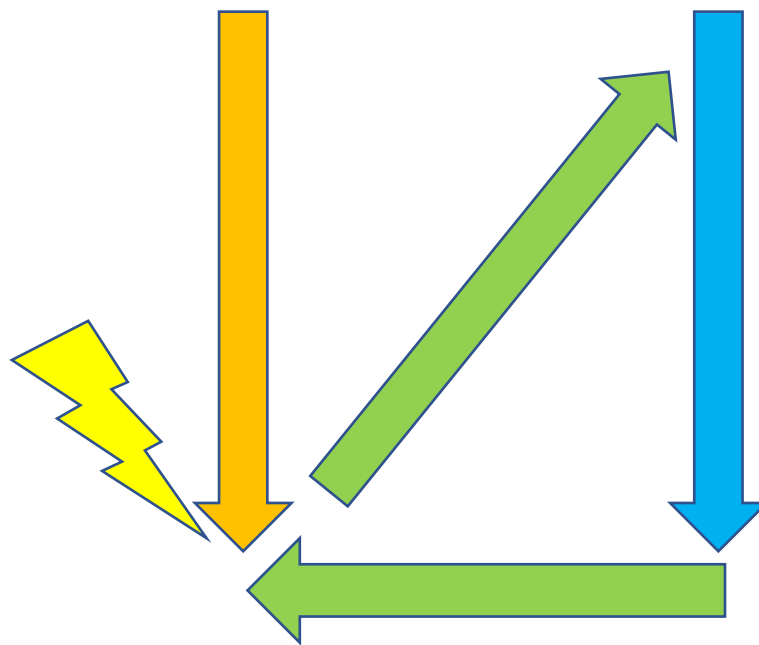
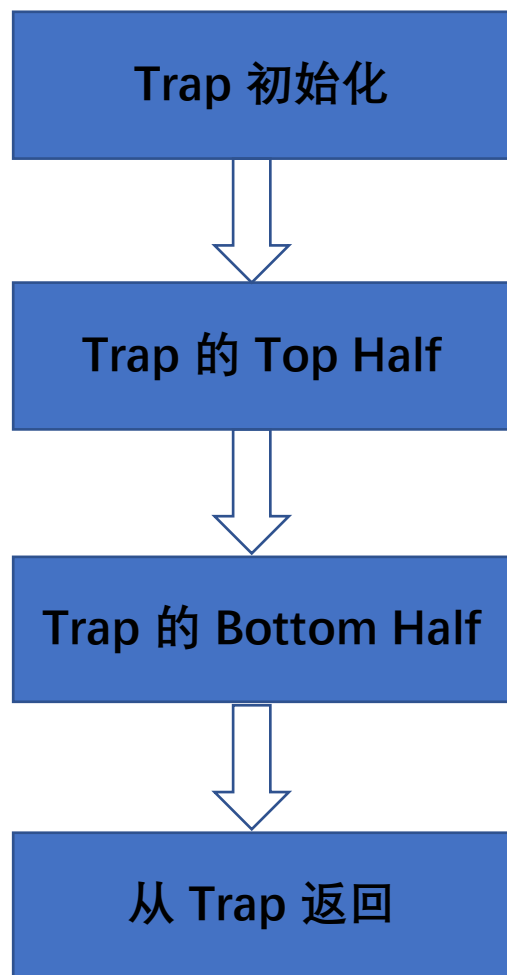
- **xIE (x=M/S/U)** : 分别用于打开 (1) 或者关闭 (0) M/S/U 模式下的全局中断。当 trap 发生时, hart 会自动将 xIE 设置为 0。
- **xPIE (x=M/S/U)** : 当 trap 发生时用于保存 trap 发生之前的 xIE 值。
- **xPP (x=M/S)** : 当 trap 发生时用于保存 trap 发生之前的权限级别值。注意没有 UPP。
- 其他标志位涉及内存访问权限、虚拟内存控制等, 暂不考虑。

Level	Encoding	Name	Abbreviation
0	00	User/Application	U
1	01	Supervisor	S
2	10	<i>Reserved</i>	
3	11	Machine	M

【参考 2】 Table 1.1: RISC-V privilege levels.

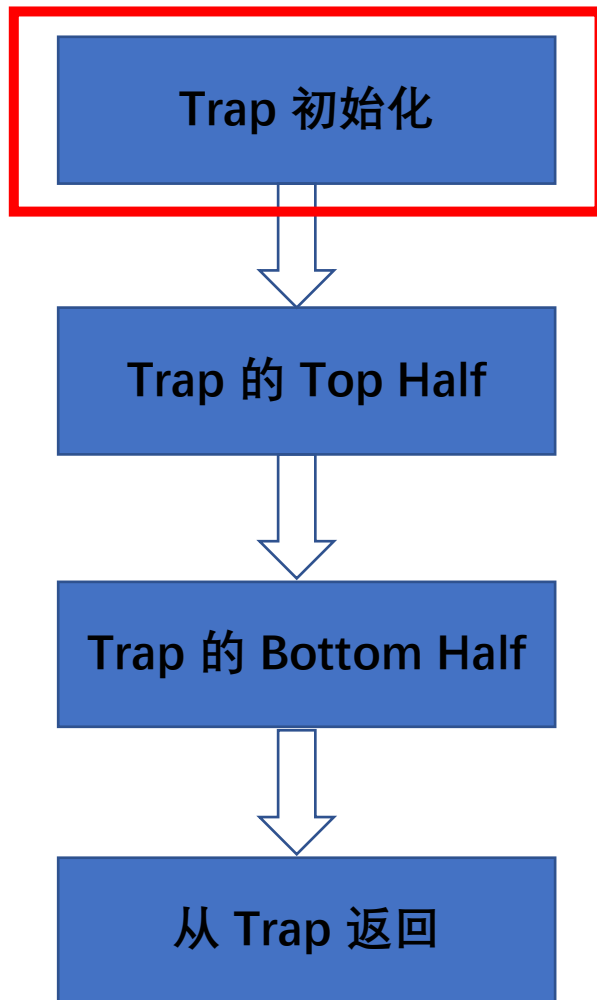
- 控制流 (Control Flow) 和 Trap
- RISC-V Trap 处理中涉及的寄存器
- **RISC-V Trap 处理流程**

# RISC-V Trap 处理流程





# RISC-V Trap 处理流程



# Trap 初始化

code/os/05-traps/trap.c

```
void trap_init()
{
    /*
     * set the trap-vector base-address for machine-mode
     */
    w_mtvec((reg_t)trap_vector);
}
```

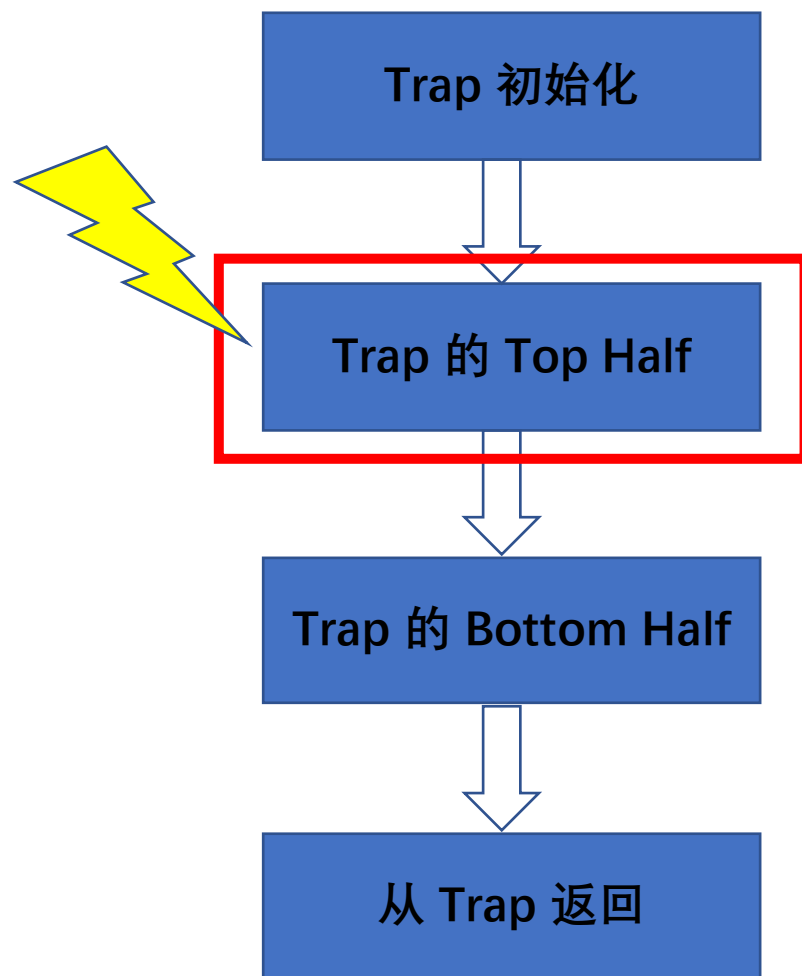


code/os/05-traps/entry.S

```
trap_vector:
    # save context(registers).
    csrrw    t6, mscratch, t6          # swap t6 and mscratch
    reg_save t6

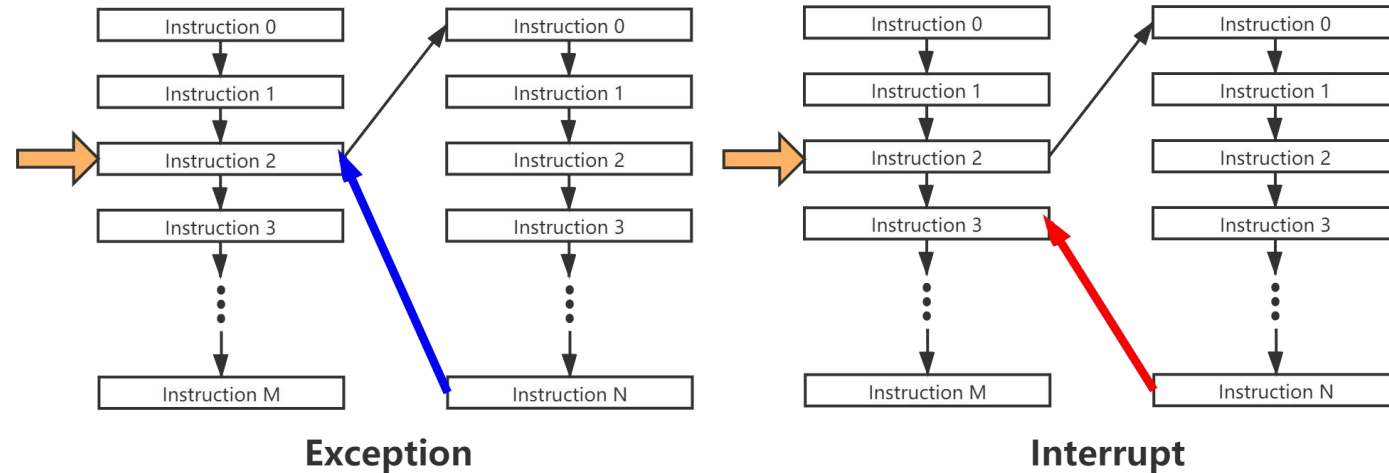
    # Save the actual t6 register, which we swapped into
    # mscratch
    mv       t5, t6                    # t5 points to the context of current task
    csrr     t6, mscratch              # read t6 back from mscratch
    cu      t6, 120(t5)                # save t6 with t5 as base
```

# RISC-V Trap 处理流程



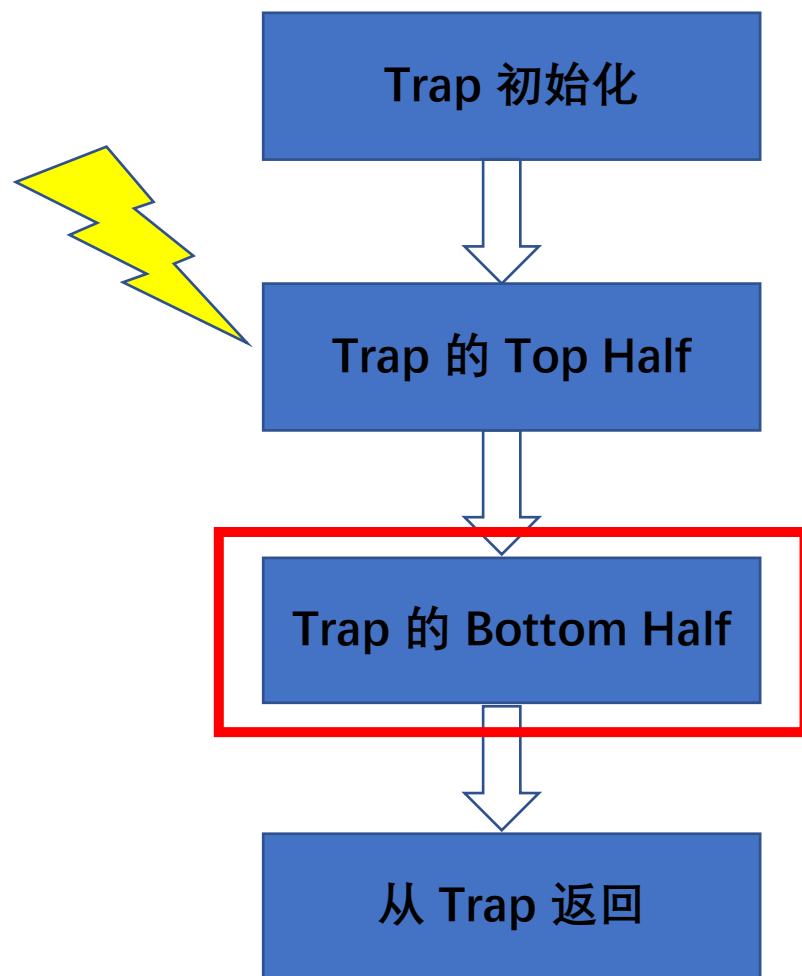
# Trap 发生, Hart 自动执行如下状态转换

- 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。
- 设置 **mepc**, 同时 **PC** 被设置为 **mtvec**。(需要注意的是, 对于 **exception**, **mepc** 指向导致异常的指令; 对于 **interrupt**, 它指向被中断的指令的下一条指令的位置。)



- 根据 **trap** 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** (也就是说无论在任何 **Level** 下触发 **trap**, **hart** 首先切换到 **Machine** 模式)。

# RISC-V Trap 处理流程



# trap handler: 软件需要做的事情

code/os/05-traps/entry.S

- 保存 (save) 当前控制流的上下文信息 (利用 mscratch)
- 调用 C 语言的 trap handler
- 从 trap handler 函数返回, mepc 的值有可能需要调整
- 恢复 (restore) 上下文的信息
- 执行 MRET 指令返回到 trap 之前的状态。

```
trap_vector:
    # save context(registers).
    csrrw    t6, mscratch, t6          # swap t6 and mscratch
    reg_save t6

    # Save the actual t6 register, which we swapped into
    # mscratch
    mv       t5, t6                   # t5 points to the context of current task
    csrr     t6, mscratch              # read t6 back from mscratch
    sw       t6, 120(t5)               # save t6 with t5 as base

    # Restore the context pointer into mscratch
    csrw     mscratch, t5

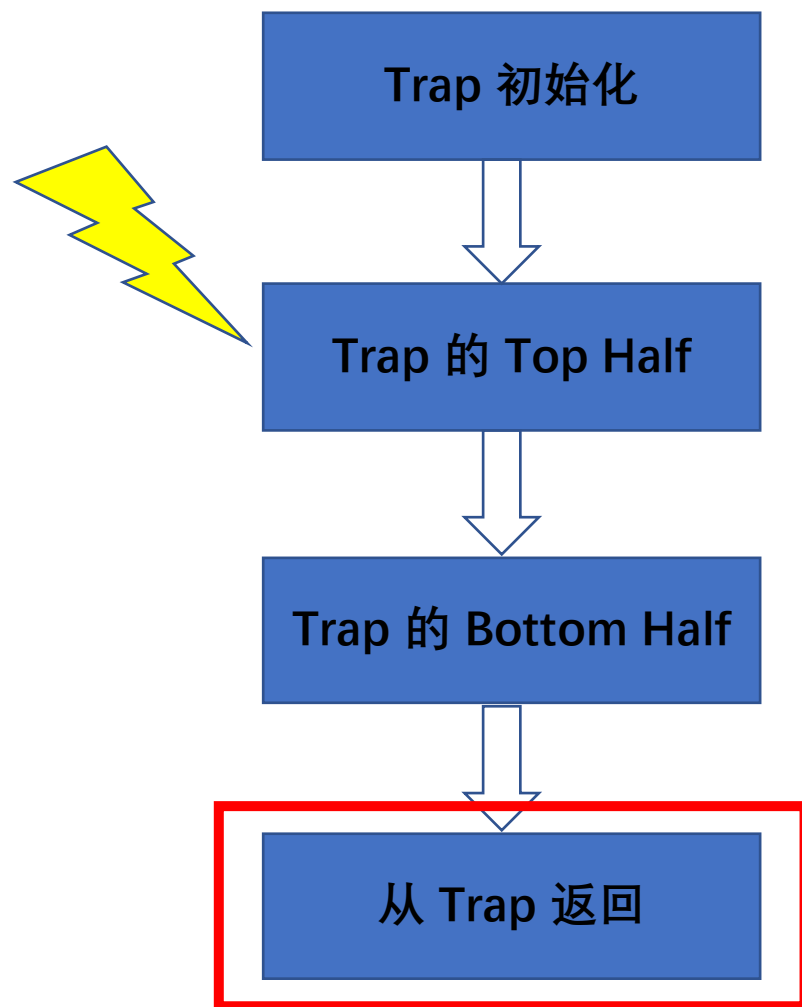
    # call the C trap handler in trap.c
    csrr     a0, mepc
    csrr     a1, mcause
    call     trap_handler

    # trap_handler will return the return address via a0.
    csrw     mepc, a0

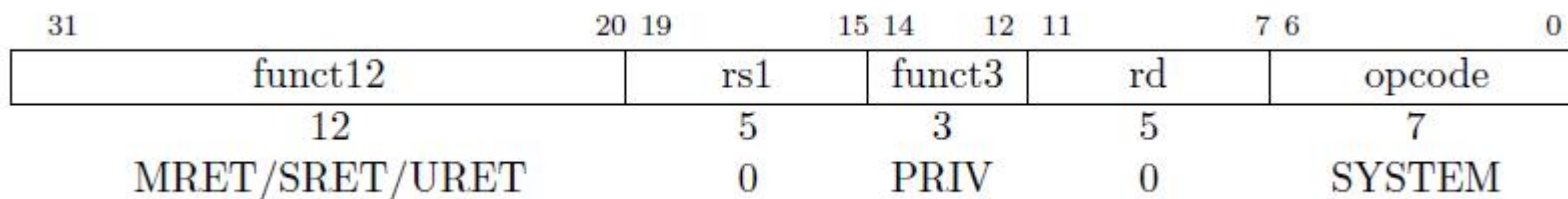
    # restore context(registers).
    csrr     t6, mscratch
    reg_restore t6

    # return to whatever we were doing before trap.
    mret
```

# RISC-V Trap 处理流程



# 退出 trap：编程调用 MRET 指令



【参考 2】 3.2.2 Trap-Return Instructions

- 针对不同权限级别下如何退出 trap 有各自的返回指令 xRET (x = M/S/U)
- 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



# 谢谢

欢迎交流合作