ISCAS NIST

循序渐进，学习开发一个 RISC-V 上的操作系统

第 12 章 硬件定时器

汪辰

2021/4

本章内容

- RISC-V 定时器中断
- RISC-V CLINT 介绍
- 硬件定时器的应用

参考

- 【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

本章内容

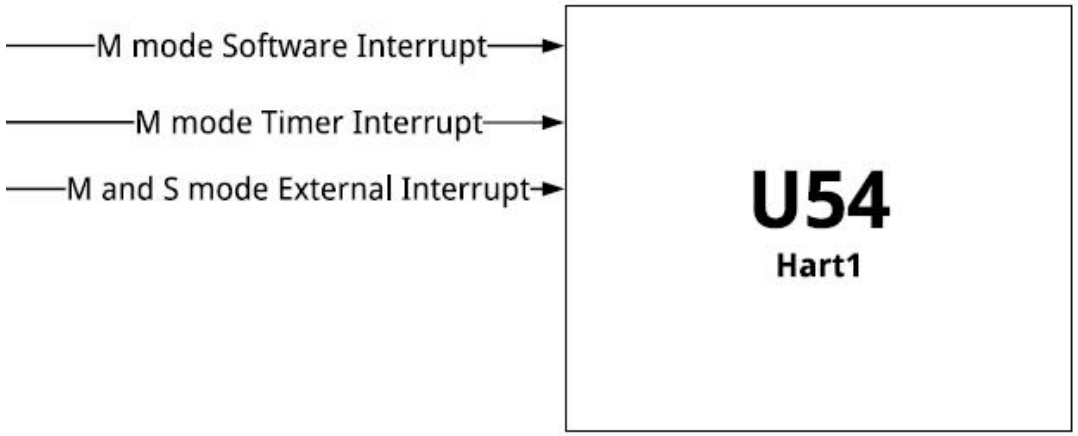
- RISC-V 定时器中断
- RISC-V CLINT 介绍
- 硬件定时器的应用

RISC-V 中断（Interrupt）的分类

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

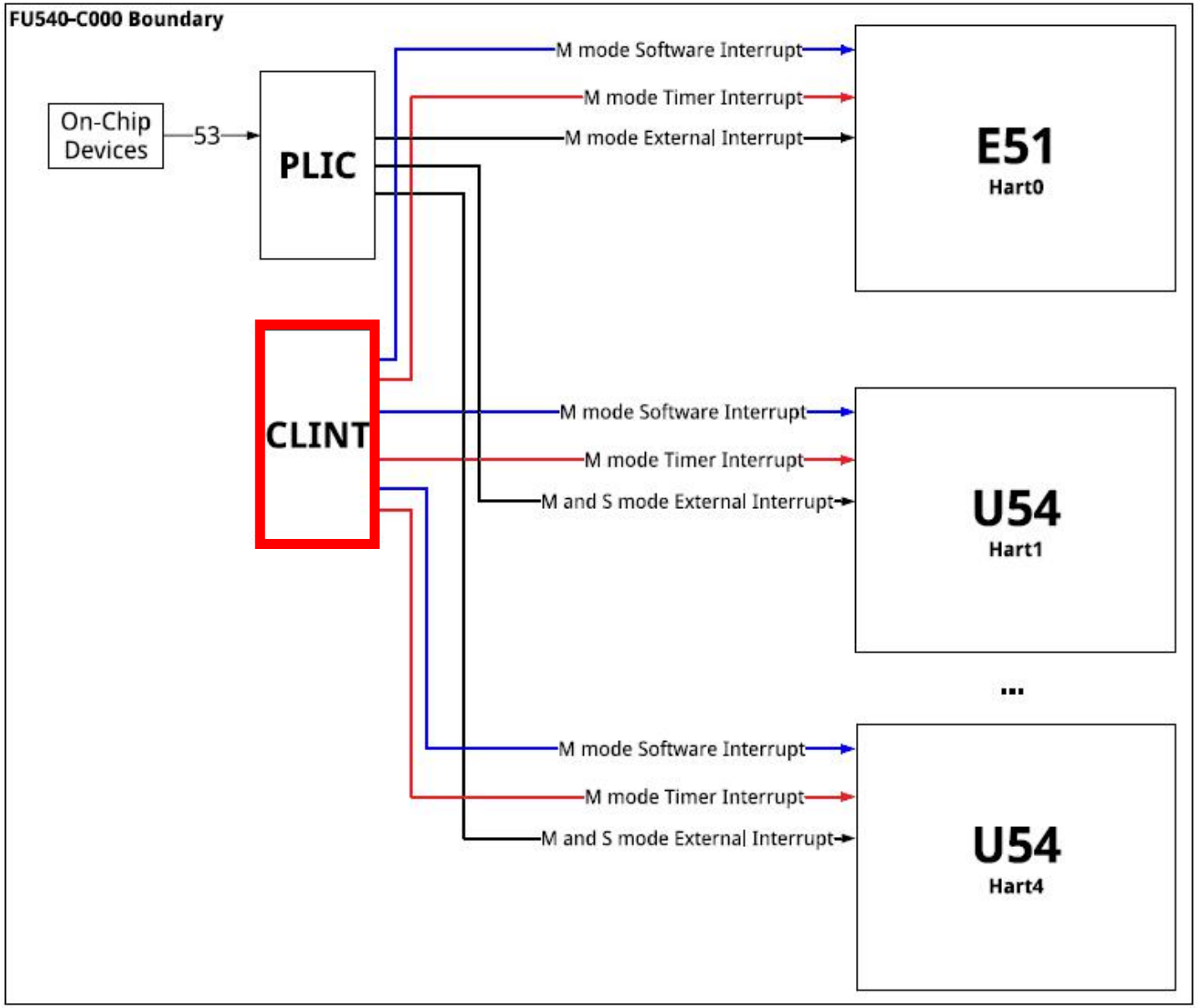
Interrupt	Exception Code	Description
1	0	User software interrupt
1	1	Supervisor software interrupt
1	2	*Reserved for future standard use*
1	3	Machine software interrupt
1	4	User timer interrupt
1	5	Supervisor timer interrupt
1	6	*Reserved for future standard use*
1	7	Machine timer interrupt
1	8	User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

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



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

Core Local INTerruptor



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

本章内容

- RISC-V 定时器中断
- RISC-V CLINT 介绍
- 硬件定时器的应用

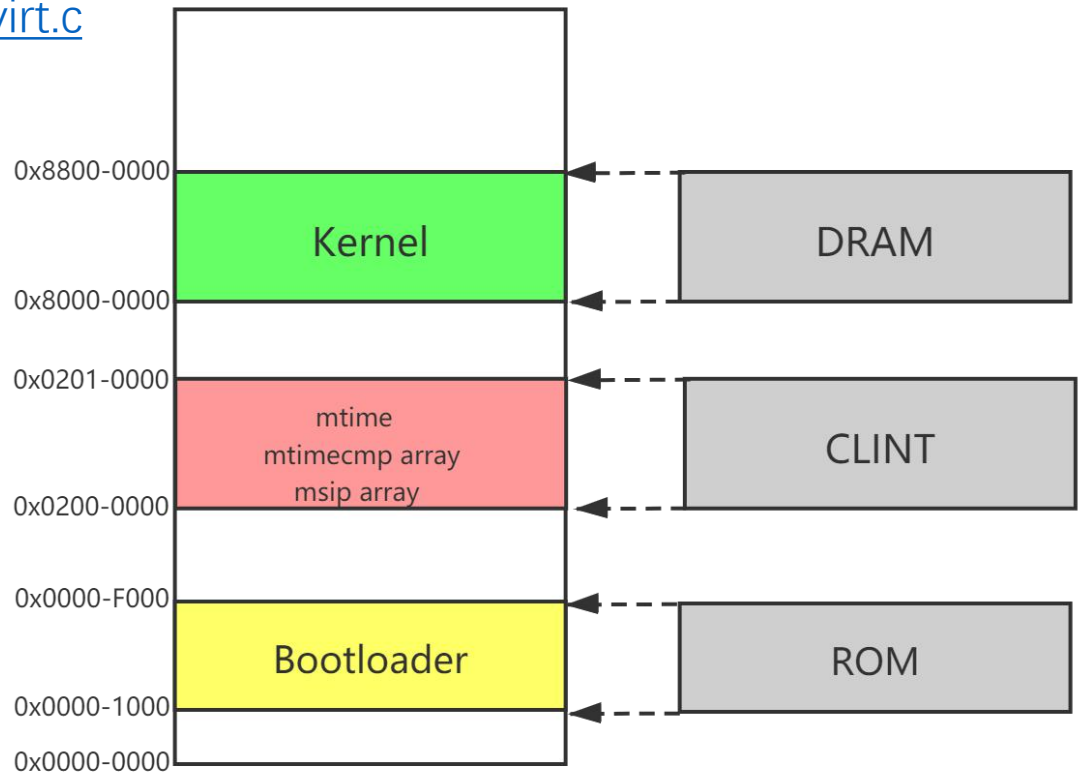
CLINT 编程接口 - 寄存器

- RISC-V 规范规定，CLINT 的寄存器编址采用内存映射（memory map）方式。
- 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 CLINT 的设计参考了 SFIVE，base 为 0x2000000。

```
#define CLINT_BASE 0x2000000L
```

https://github.com/qemu/qemu/blob/master/hw/riscv/virt.c

```
static const MemMapEntry virt_memmap[] = {
    [VIRT_DEBUG] =       {        0x0,         0x100 },
    [VIRT_MROM] =        {     0x1000,         0xf000 },
    [VIRT_TEST] =        {   0x100000,         0x1000 },
    [VIRT_RTC] =         {   0x101000,         0x1000 },
    [VIRT_CLINT] =       {  0x2000000,        0x10000 },
    [VIRT_PCIE_PIO] =    {  0x3000000,        0x10000 },
    [VIRT_PLIC] =        {  0xc000000, VIRT_PLIC_SIZE(VIRT_CPUS_MAX * 2) },
    [VIRT_UART0] =       { 0x10000000,         0x100 },
    [VIRT_VIRTIO] =      { 0x10001000,         0x1000 },
    [VIRT_FLASH] =       { 0x20000000,     0x4000000 },
    [VIRT_PCIE_ECAM] =   { 0x30000000,    0x10000000 },
    [VIRT_PCIE_MMIO] =   { 0x40000000,    0x40000000 },
    [VIRT_DRAM] =        { 0x80000000,           0x0 },
};
```



CLINT 编程接口 - 寄存器 (Timer 部分)

可编程寄存器	功能描述	内存映射地址
mtime	real-time 计数器（counter）	BASE + 0xbff8

- 系统全局唯一，在 RV32 和 RV64 上都是 64-bit。系统必须保证该计数器的值始终按照一个固定的频率递增。
- 上电复位时，硬件负责将 mtime 的值恢复为 0。

```
#define CLINT_MTIME (CLINT_BASE + 0xBFF8) // cycles since boot.
```

CLINT 编程接口 - 寄存器 (Timer 部分)

可编程寄存器	功能描述	内存映射地址
mtimecmp	timer compare register	BASE + 0x4000 + (hart) * 8)

- 每个 hart 一个 mtimecmp 寄存器，64-bit。
- 上电复位时，系统不负责设置 mtimecmp 的初值。

```
#define CLINT_MTIMECMP(hartid) (CLINT_BASE + 0x4000 + 8 * (hartid))
```

```
void timer_load(int interval)
{
        int id = r_mhartid();
        *(uint64_t*)CLINT_MTIMECMP(id) =
                *(uint64_t*)CLINT_MTIME + interval;
}
```

```
void timer_init()
{
        timer_load(TIMER_INTERVAL);
        ......
}
```

```
void start_kernel(void)
{
        ......
        timer_init();
        ......
        while (1) {}; // stop here!
}
```

CLINT 编程接口 - 寄存器 (Timer 部分)

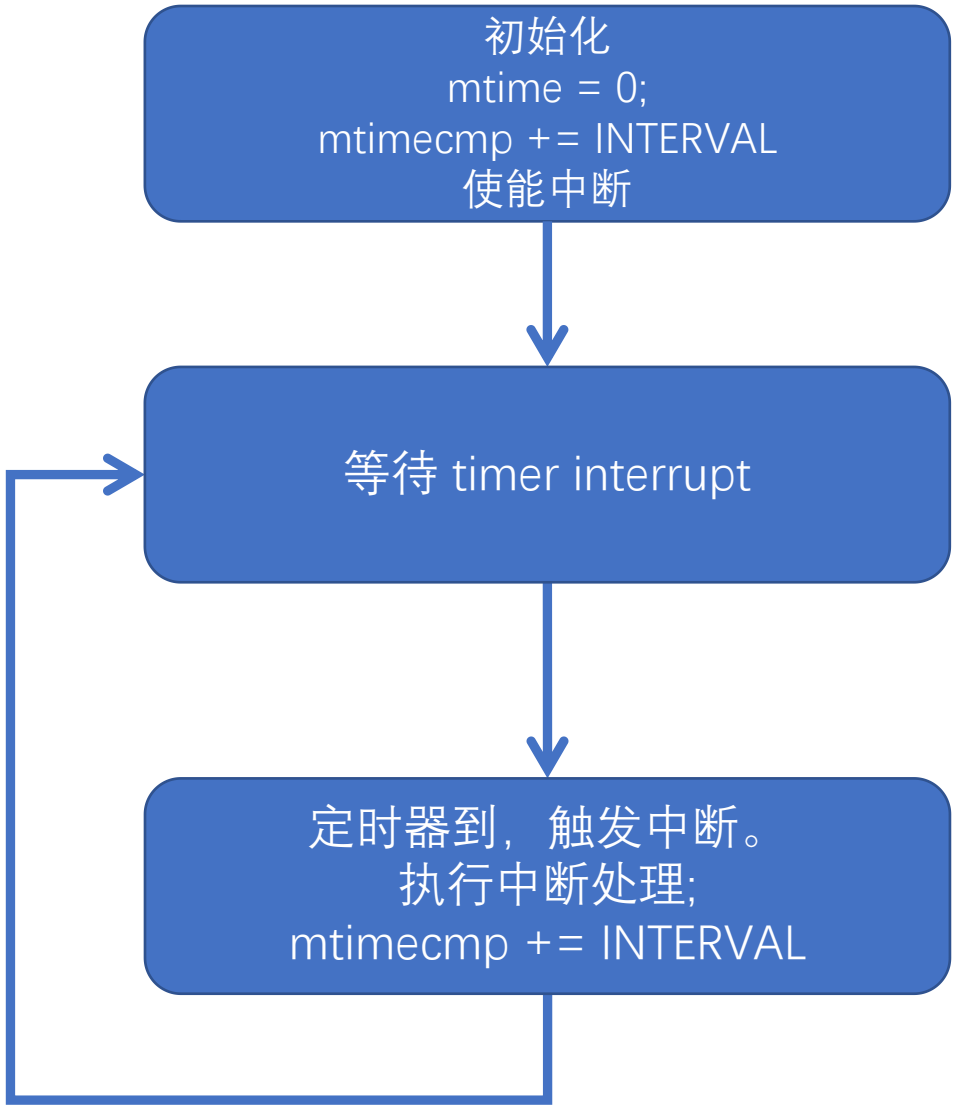
- 当 mtime >= mtimecmp 时，CLINT 会产生一个 timer 中断。如果要使能该中断需要保证全局中断打开并且 mie.MTIE 标志位置 1。
- 当 timer 中断发生时，hart 会设置 mip.MTIP，程序可以在 mtimecmp 中写入新的值清除 mip.MTIP。

```
void timer_handler()
{
        ......
        timer_load(TIMER_INTERVAL);
}
```

```
void timer_init()
{
        timer_load(TIMER_INTERVAL);
        /* enable machine-mode timer interrupts. */
        w_mie(r_mie() | MIE_MTIE);
        /* enable machine-mode global interrupts. */
        w_mstatus(r_mstatus() | MSTATUS_MIE);
}
```

```
reg_t trap_handler(reg_t epc, reg_t cause)
{
        ......
        if (cause & 0x80000000) {
                switch (cause_code) {
                ......
                case 7:
                        timer_handler();
                        break;
                ......
                }
        }
        ......
}
```

CLINT 编程接口 - 总体框架流程



本章内容

- RISC-V 定时器中断
- RISC-V CLINT 介绍
- **硬件定时器的应用**

硬件定时器的应用：时间管理

- 生活离不开对时间的管理；操作系统的运行也是一样。

时钟节拍（Tick）

- 操作系统中最小的时间单位
- Tick 的单位（周期）由硬件定时器的周期决定（通常为 1 ~ 100ms）
- Tick 周期越小，系统的精度越高，但开销越大。

系统时钟

- 操作系统维护的一个整型计数值，记录着系统启动直到当前发生的 Tick 总数。
- 可用于维护系统的墙上时间，所以也称为系统时钟。


```c
/* interval ~= 1s */
#define TIMER_INTERVAL CLINT_TIMEBASE_FREQ

static uint32_t _tick = 0;
```

```c
void timer_init()
{
        timer_load(TIMER_INTERVAL);
```

```c
void timer_handler()
{
        _tick++;
        printf("tick: %d\n", _tick);

        timer_load(TIMER_INTERVAL);
}
```

练习 12-1

谢 谢

欢迎交流合作