Interrupt and Exception

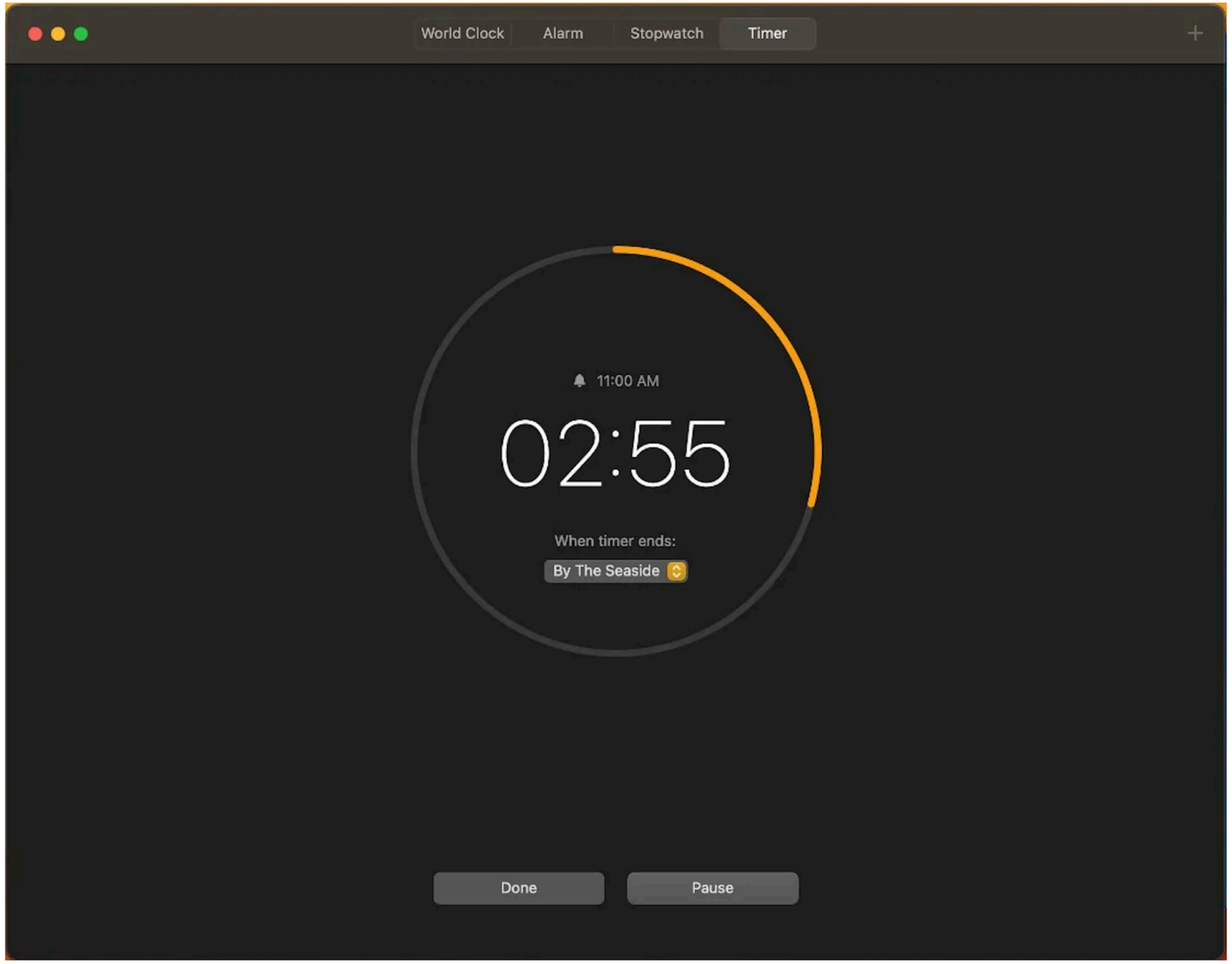
High-level roadmap

- [basic CPU instructions] user-level threading
- [+ timer interrupt] timeshare threading
- [+ ecall exception] system call
- [+ privilege levels] memory protection
- [+ I/O bus control] disk driver, cache and file systems

P2: interrupt and exception

- [basic CPU instructions] user-level threading
- [+ timer interrupt] timeshare threading
- [+ ecall exception] system call
- [+ privilege levels] memory protection
- [+ I/O bus control] disk driver, cache and file systems

Set a timer


World Clock
Alarm
Stopwatch
Timer
11:00 AM
02:55
When timer ends:
By The Seaside
Done
Pause

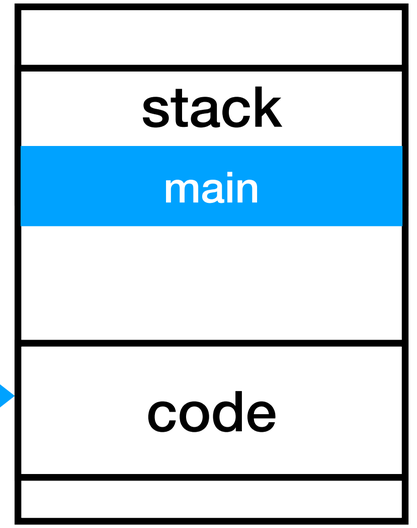
First glance of timer interrupt

```
void handler() {
    earth->tty_info("Got timer interrupt.");
    // start another timer
}

int main() {
    // register handler() as interrupt handler
    // enable timer interrupt
    // start a timer

    while(1);
}
```

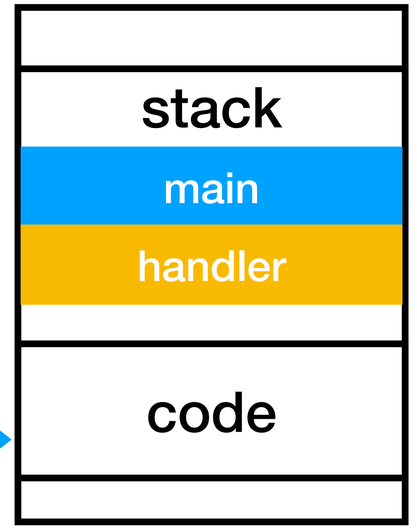
Execution of this program

```
void handler() {
    earth->tty_info("...");
    // start another timer
}

int main() {
    // register handler()
    // enable timer interrupt
    // start a timer

    while(1);
}
```

When timer is running



When timer runs out



- How to register `handler()` as interrupt handler?
- How to start a timer?
- How to enable timer interrupt?

CSR: control and status registers

- There are many registers other than x0 .. x31.
  - *machine ISA*: 32-bit or 64bit?
  - *hart ID*: the ID number of a core in a multi-core CPU
  - *interrupt control: timer, I/O device …*

The `mtvec` CSR

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

Table 3.5: Encoding of `mtvec` MODE field.

Page 26 of RISC-V manual, volume2, v1.10
https://github.com/yhzhang0128/egos-2000/blob/main/references/riscv-privileged-v1.10.pdf

Register an interrupt handler

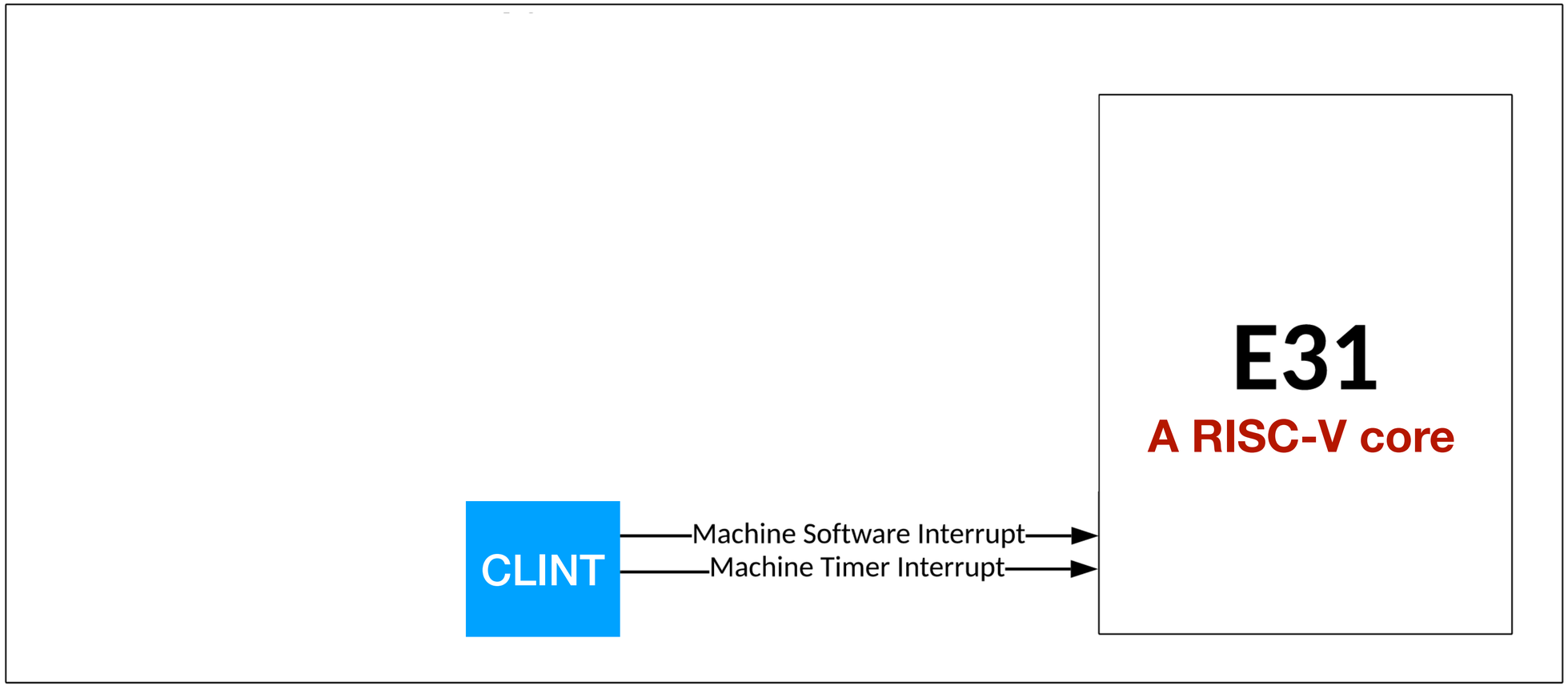
```
0800280c <handler>:
    . . .

08002914 <main>:
    . . .
    lui     a5,0x8003    # now a5 == 0x08003000
    addi    a5,a5,-2036 # now a5 == 0x0800280c
    # csrw: control and status register write
    csrw    mtvec,a5     # now mtvec == 0x0800280c
    . . .
```

Register an interrupt handler in C

```
void handler() {
    . . .
}

int main() {
    /* Register handler with direct mode */
    asm("csrw mtvec, %0" ::"r"(handler));
    . . .
}
```

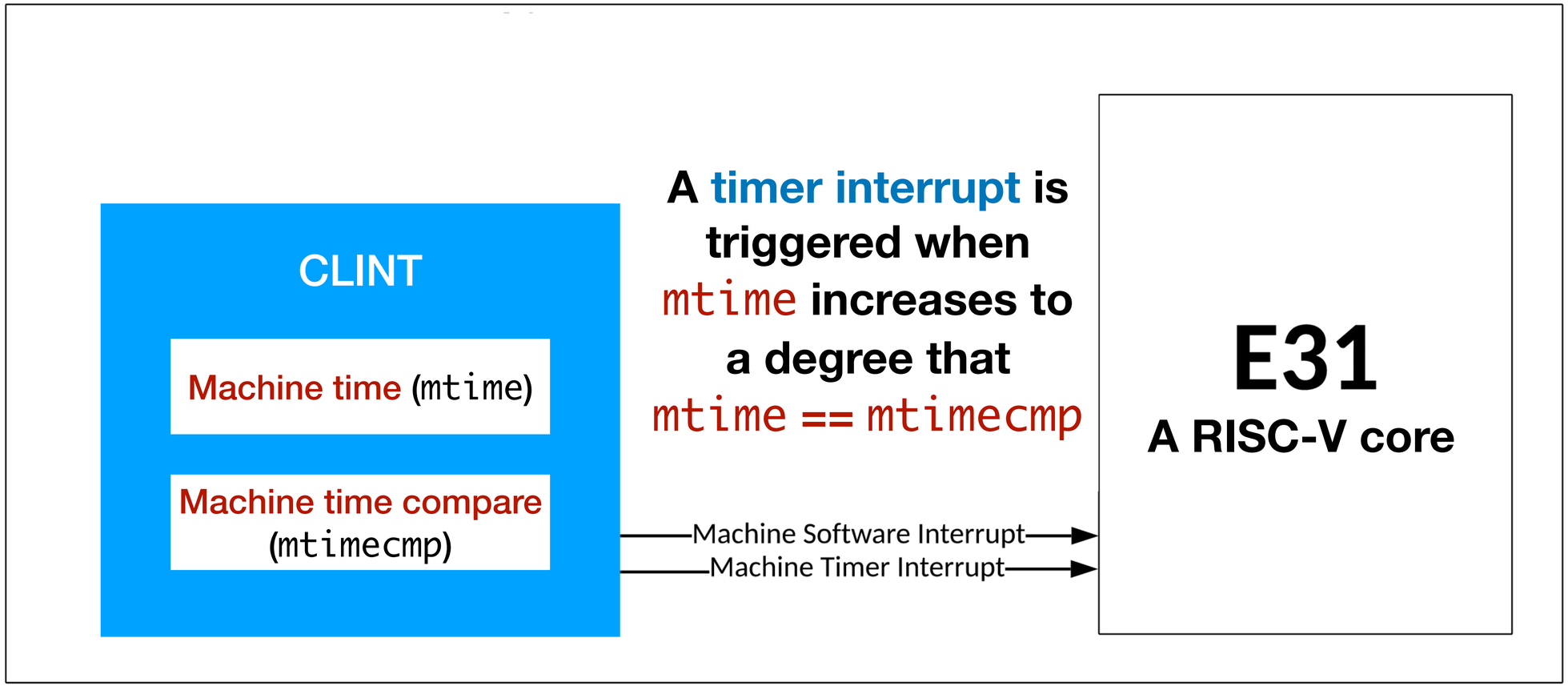
- How to register `handler()` as interrupt handler?
- How to start a timer?
- How to enable timer interrupt?

Core-local Interrupt (CLINT)



Page 38 of Sifive FE310 manual, v19p04
https://github.com/yhzhang0128/egos-2000/blob/main/references/sifive-fe310-v19p04.pdf

mtime and mtimecmp



Page 38 of Sifive FE310 manual, v19p04
https://github.com/yhzhang0128/egos-2000/blob/main/references/sifive-fe310-v19p04.pdf

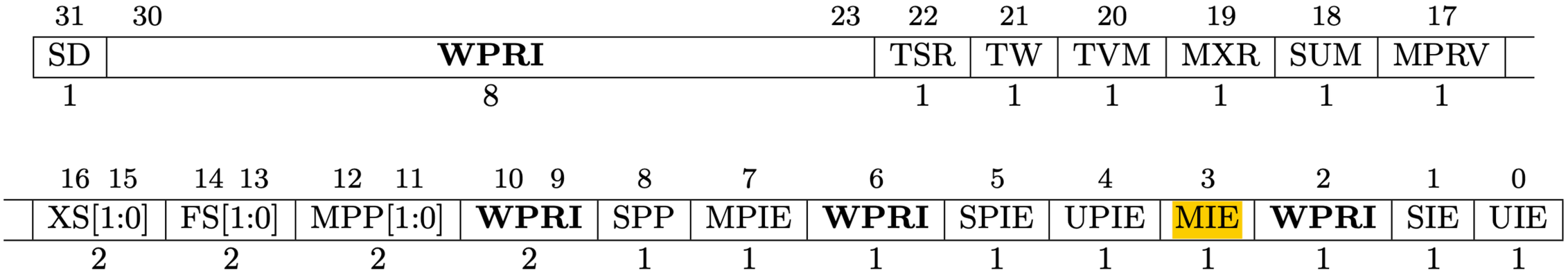
```
int quantum = 500000;

void handler() {
    ...
    mtimecmp_set(mtime_get() + quantum);
}

int main() {
    ...
    mtimecmp_set(mtime_get() + quantum);
    ...
}
```

Read current time

Set timer

- How to register `handler()` as interrupt handler?
- How to start a timer?
- How to enable timer interrupt?

The mstatus CSR



MIE stands for machine interrupt enable

Page 20 of RISC-V manual, volume2, v1.10
https://github.com/yhzhang0128/egos-2000/blob/main/references/riscv-privileged-v1.10.pdf

Enable machine interrupts

```
08002914 <main>:
    . . .
    csrr    a5,mstatus  # read CSR mstatus to a5
    ori     a5,a5,8     # set bit#3 of a5 to 1
    csrw    mstatus,a5  # write CSR mstatus
    . . .
```

```
int main() {
    . . .
    int mstatus;
    asm("csrr %0, mstatus" : "=r"(mstatus));
    asm("csrw mstatus, %0" ::"r"(mstatus | 0x8));
   . . .
}
```

Another CSR `mie` (not `mstatus`)

- mstatus.MIE is bit #3 in mstatus
- mie is another CSR, and mie.MTIE is bit #7 in mie

MTIE stands for machine timer interrupt enable

XLEN-1 12	11	10	9	8	7	6	5	4	3	2	1	0
WPRI	MEIE	**WPRI**	SEIE	UEIE	MTIE	**WPRI**	STIE	UTIE	MSIE	**WPRI**	SSIE	USIE
XLEN-12	1	1	1	1	1	1	1	1	1	1	1	1

Page 29 of RISC-V manual, volume2, v1.10
https://github.com/yhzhang0128/egos-2000/blob/main/references/riscv-privileged-v1.10.pdf

Enable timer interrupt

```
08002914 <main>:
    . . .
    csrr    a5,mie      # read CSR mie to a5
    ori     a5,a5,128   # set bit#7 of a5 to 1
    csrw    mie,a5      # write CSR mie
    . . .
```

```
int main() {
    . . .
    int mie;
    asm("csrr %0, mie" : "=r"(mie));
    asm("csrw mie, %0" ::"r"(mie | 0x80));
   . . .
}
```

Put together: Enable timer interrupt

```
int main() {
    . . .
    int mstatus, mie;
    asm("csrr %0, mstatus" : "=r"(mstatus));
    asm("csrw mstatus, %0" ::"r"(mstatus | 0x8));
    asm("csrr %0, mie" : "=r"(mie));
    asm("csrw mie, %0" ::"r"(mie | 0x80));
    . . .
}
```

Summary of timer interrupt

- How to register an interrupt handler?
  - write the address of `handler()` to `mtvec`
- How to set a timer?
  - write (`mtime` + `quantum`) to `mtimecmp`
- How to enable timer interrupt?
  - set bit#3 of CSR `mstatus` and bit#7 of CSR `mie`

CSR is a key CPU support for OS

- How to register an interrupt handler?
  - write the address of `handler()` to `mtvec`
- How to set a timer?
  - write (`mtime` + *quantum*) to `mtimecmp`
- How to enable timer interrupt?
  - set bit#3 of CSR `mstatus` and bit#7 of CSR `mie`

A timer handler program

```
int quantum = 50000;

void handler() {
    earth->tty_info("Got timer interrupt.");
    mtimecmp_set(mtime_get() + quantum);
}

int main() {
    earth->tty_success("A timer interrupt example.");

    asm("csrw mtvec, %0" ::"r"(handler));
    mtimecmp_set(mtime_get() + quantum);

    int mstatus, mie;
    asm("csrr %0, mstatus" : "=r"(mstatus));
    asm("csrw mstatus, %0" ::"r"(mstatus | 0x8));
    asm("csrr %0, mie" : "=r"(mie));
    asm("csrw mie, %0" ::"r"(mie | 0x80));

    while(1);
}
```

Start another timer

Register handler

Start a timer

Enable timer interrupt

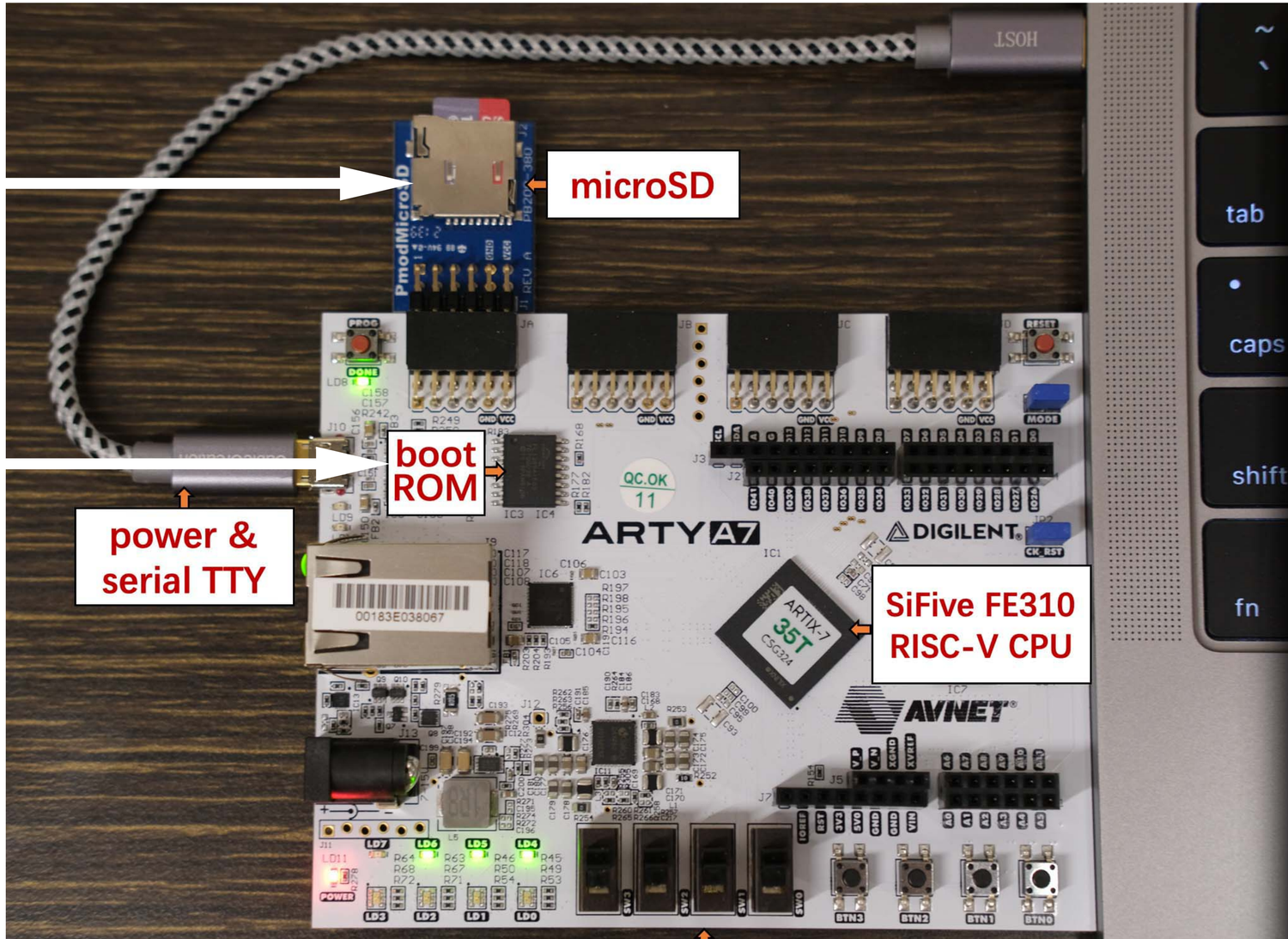
Demo

https://github.com/yhzhang0128/egos-2000/tree/timer_demo/grass

demo code in microSD card

earth layer in boot ROM containing

1. SD card driver for loading demo
2. TTY driver for printing to screen



Timer is interrupt #7

Interrupts

Exceptions

Interrupt Exception Codes		
Interrupt	**Exception Code**	**Description**
1	0–2	Reserved
1	3	Machine software interrupt
1	4–6	Reserved
1	7	Machine timer interrupt
1	8–10	Reserved
1	11	Machine external interrupt
1	≥ 12	Reserved
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	9–10	Reserved
0	11	Environment call from M-mode
0	≥ 12	Reserved

Page 42 of Sifive FE310 manual, v19p04
https://github.com/yhzhang0128/egos-2000/blob/main/references/sifive-fe310-v19p04.pdf

System call is exception #8, #11

Interrupts

Exceptions

Interrupt Exception Codes		
Interrupt	**Exception Code**	**Description**
1	0–2	Reserved
1	3	Machine software interrupt
1	4–6	Reserved
1	7	Machine timer interrupt
1	8–10	Reserved
1	11	Machine external interrupt
1	≥ 12	Reserved
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	9–10	Reserved
0	11	Environment call from M-mode
0	≥ 12	Reserved

Page 42 of Sifive FE310 manual, v19p04

https://github.com/yhzhang0128/egos-2000/blob/main/references/sifive-fe310-v19p04.pdf

Kernel ≈ 3 handlers

```
void kernel() {  // registered to CSR mtvec
    int mcause;
    __asm__ volatile("csrr %0, mcause" : "=r"(mcause));

    int id = mcause & 0x3ff;
    if (mcause & (1 << 31)) {
        if (id == 7) { yield(); }
    } else {
        if (id == 8) { syscall_handler(); }
                else { fault_handler(); }
    }
}
```

Kernel ≈ timer handler + system call handler + fault handler

Design of projects P1 and P2

```
void kernel() {
    int mcause;
    __asm__ volatile("csrr %0, mcause" : "=r"(mcause));

    int id = mcause & 0x3ff;
    if (mcause & (1 << 31)) {
        // P1: multi-threading
        if (id == 7) { yield(); }
    } else {
        // P2: system call and memory protection
        if (id == 8) { syscall_handler(); }
                else { fault_handler(); }
    }
}
```

Some details: memory-mapped register

`mtimecmp_set()` writes 8 bytes to

`mtime_get()` reads 8 bytes from

Address	Width	Attr.	Description
0x2000000	4B	RW	msip for hart 0
0x2004008			Reserved
…			
0x200bff7			
0x2004000	8B	RW	mtimecmp for hart 0
0x2004008			Reserved
…			
0x200bff7			
0x200bff8	8B	RW	mtime
0x200c000			Reserved

Page 44 of Sifive FE310 manual, v19p04
https://github.com/yhzhang0128/egos-2000/blob/main/references/sifive-fe310-v19p04.pdf

Homework

- P2 has been released and it is due on Oct 20.
- Read the 4 files of the timer demo program.
  - https://github.com/yhzhang0128/egos-2000/tree/timer_demo/grass