Timer and Interrupt Handling
High-level roadmap

- [ basic RISC-V CPU ] non-preemptive multi-threading
- [ + timer interrupt ] preemptive scheduling
- [ + privilege levels ] protection and isolation for processes
- [ + I/O bus controllers ] disk driver and file systems
Overview of a timer handler program

```c
int main() {
    // register handler() as interrupt handler
    // set a timer
    // enable timer interrupt

    while(1);
}

void handler() {
    earth->tty_info("Got timer interrupt.");
    // set a timer
}
```
How to register handler() as interrupt handler?

• How to set a timer?

• How to enable timer interrupt?
Control and status registers (CSR)

- There are many registers other than the 32 user-level ones:
  - *machine ISA*: 32-bit or 64bit?
  - *hart ID*: the ID number of a core in a multi-core CPU
  - *interrupt control*: timer, …
The `mtvec` CSR

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Direct</td>
<td>All exceptions set pc to BASE.</td>
</tr>
<tr>
<td>1</td>
<td>Vectored</td>
<td>Asynchronous interrupts set pc to BASE+4×cause.</td>
</tr>
<tr>
<td>≥2</td>
<td>—</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 3.5: Encoding of `mtvec` MODE field.
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

void handler() {
    . . .
}

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

• How to `register handler()` as interrupt handler?

• How to `set` a timer?

• How to `enable` timer interrupt?
Core-local Interrupt (CLINT)

Page 38 of Sifive FE310 manual, v19p04
The \texttt{mtime} and \texttt{mtimecmp} CSRs

A \textbf{timer interrupt} is triggered when \texttt{mtime} > \texttt{mtimecmp}

\begin{itemize}
  \item CLINT
    \begin{itemize}
      \item Machine time (\texttt{mtime})
      \item Machine time compare (\texttt{mtimecmp})
    \end{itemize}
  \end{itemize}

E31
A RISC-V core

Page 38 of Sifive FE310 manual, v19p04
Set a timer

```c
int quantum = 50000;

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

int main() {
    /* Set a timer */
    mtimecmp_set(mtime_get() + quantum);
    ...
}
```
CLINT CSRs are **memory-mapped**

<table>
<thead>
<tr>
<th>Address</th>
<th>Width</th>
<th>Attr.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x20000000</td>
<td>4B</td>
<td>RW</td>
<td>msip for hart 0</td>
</tr>
<tr>
<td>0x2004008</td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>mtimecmp for hart 0</td>
</tr>
<tr>
<td>0x200b7</td>
<td>8B</td>
<td>RW</td>
<td>mtimecmp for hart 0</td>
</tr>
<tr>
<td>0x2004000</td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x2004008</td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x200b7</td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x200b8</td>
<td>8B</td>
<td>RW</td>
<td>mtime</td>
</tr>
<tr>
<td>0x200c000</td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**mtime_get()** reads 8 bytes from

**mtimecmp_set()** writes 8 bytes to

---

Page 44 of Sifive FE310 manual, v19p04
• How to `register handler()` as interrupt handler?

• How to set a timer?

→ How to `enable` timer interrupt?
The **mstatus** CSR

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>SD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WPRI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>XS[1:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FS[1:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPP[1:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WPRI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>2</th>
<th>2</th>
<th>2</th>
<th>2</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
</table>

**MIE** stands for machine interrupt enable
Enable all (not only timer) interrupts

08002914 <main>:
  ...
  csrr    a5,mstatus    # read CSR mstatus to a5
  ori     a5,a5,8       # set bit3 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));
  ...
}
The **wie CSR (not mstatus.MIE)**

<table>
<thead>
<tr>
<th>XLEN-12</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>WPRI</td>
<td>MEIE</td>
<td>WPRI</td>
<td>SEIE</td>
<td>UEIE</td>
<td>MTIE</td>
<td>WPRI</td>
<td>STIE</td>
<td>UTIE</td>
<td>MSIE</td>
<td>WPRI</td>
<td>SSIE</td>
</tr>
</tbody>
</table>

**MTIE** stands for machine timer interrupt enable
Enable **timer** interrupt

```c
int main() {
    . . .
    int mie;
    asm("csrr %0, mie" : ":r"(mie));
    asm("csrw mie, %0" : ":r"(mie | 0x80));
    . . .
}
```
Enable timer interrupt: altogether

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

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 function handler() to mtvec

• How to set a timer?
  • write \((mtime + quantum)\) to mtimecmp

• How to enable timer interrupt?
  • set certain bit of mstatus and mie to 1
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("csr 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);
}

A timer handler program

Register handler
Set a timer

Enable timer interrupt
Demo on a RISC-V board

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

demo code in microSD card

earth layer code in boot ROM
1. Load demo from microSD
2. Print strings to the screen
Guard against mtime rollover

When reading the lower 4 bytes, mtime is
\[0x00000000 \quad 0xffffffff\]

When reading the higher 4 bytes, mtime is
\[0x00000001 \quad 0x00000000\]

Combine the two \[0x00000001ffffffff\] is wrong!
Guard against rollover

```c
long long mtime_get() {
    int low, high;
    do {
        high = *(int*)(0x200bff8 + 4);
        low  = *(int*)(0x200bff8);
    } while ( *(int*)(0x200bff8 + 4) != high );
    return (((long long)high) << 32) | low;
}

void mtimecmp_set(long long time) {
    *(int*)(0x2004000 + 4) = 0xFFFFFFFF;
    *(int*)(0x2004000 + 0) = (int)time;
    *(int*)(0x2004000 + 4) = (int)(time >> 32);
}
```
Platform Interrupt Controller (PLIC)

FE310-G002 Interrupt Architecture

- AON
- 2x UART
- 3x QSPI
- GPIO
- 3x PWM
- I2C

PLIC

3 types of interrupts

E311
A RISC-V core

CLINT

Page 38 of Sifive FE310 manual, v19p04
mie provides fine-grained control

MTIE: machine `timer` interrupt enable

MEIE: machine `external` interrupt enable

MSIE: machine `software` interrupt enable

Figure 3.12: Machine interrupt-enable register (mie).
Kernel $\approx$ timer handler + system call handler + fault handler
## Timer is interrupt #7

### Interrupt Exception Codes

<table>
<thead>
<tr>
<th>Interrupt</th>
<th>Exception Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0–2</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>Machine software interrupt</td>
</tr>
<tr>
<td>1</td>
<td>4–6</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>Machine timer interrupt</td>
</tr>
<tr>
<td>1</td>
<td>8–10</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>Machine external interrupt</td>
</tr>
<tr>
<td>1</td>
<td>≥ 12</td>
<td>Reserved</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Instruction address misaligned</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Instruction access fault</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>Illegal instruction</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>Breakpoint</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>Load address misaligned</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>Load access fault</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>Store/AMO address misaligned</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>Store/AMO access fault</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>Environment call from U-mode</td>
</tr>
<tr>
<td>0</td>
<td>9–10</td>
<td>Reserved</td>
</tr>
<tr>
<td>0</td>
<td>11</td>
<td>Environment call from M-mode</td>
</tr>
<tr>
<td>0</td>
<td>≥ 12</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Page 42 of Sifive FE310 manual, v19p04**
## System call is exception #8

<table>
<thead>
<tr>
<th>Interrupt</th>
<th>Exception Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0–2</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>Machine software interrupt</td>
</tr>
<tr>
<td>1</td>
<td>4–6</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>Machine timer interrupt</td>
</tr>
<tr>
<td>1</td>
<td>8–10</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>Machine external interrupt</td>
</tr>
<tr>
<td>1</td>
<td>≥ 12</td>
<td>Reserved</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Instruction address misaligned</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Instruction access fault</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>Illegal instruction</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>Breakpoint</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>Load address misaligned</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>Load access fault</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>Store/AMO address misaligned</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>Store/AMO access fault</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>Environment call from U-mode</td>
</tr>
<tr>
<td>0</td>
<td>9–10</td>
<td>Reserved</td>
</tr>
<tr>
<td>0</td>
<td>11</td>
<td>Environment call from M-mode</td>
</tr>
<tr>
<td>0</td>
<td>≥ 12</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
`void kernel() {  // registered to CSR mtvec
    int mcause;
    __asm__ volatile("csrr %0, mcause" : "=r"(mcause));

    int id = mcause & 0x3ff;  // take the last 10 bits
    if (mcause & (1 << 31)) {  // most significant bit is 1?
        if (id == 7) { timer_handler(); } else {
            if (id == 8) { syscall_handler(); } else {
                fault_handler();
            }
        }
    }
}
void kernel() {
    int mcause;
    __asm__ volatile("csrr %0, mcause" : "=r"(mcause));

    int id = mcause & 0x3ff;
    if (mcause & (1 << 31)) {
        // P2: scheduling
        if (id == 7) { timer_handler(); }
    } else {
        // P3: system call and memory protection
        if (id == 8) { syscall_handler(); }
        else { fault_handler(); }
    }
}
Example of scheduling

Thread1: I/O-bound (zoom)
Thread2: CPU-bound (matrix computation)
Homework

• P2 has been released and it is due on Oct 12.
  • scheduling algorithm: multi-level feedback queue
• Read the 4 files of the timer handler program.
  • https://github.com/yhzhang0128/egos-2000/tree/timer_example/grass