

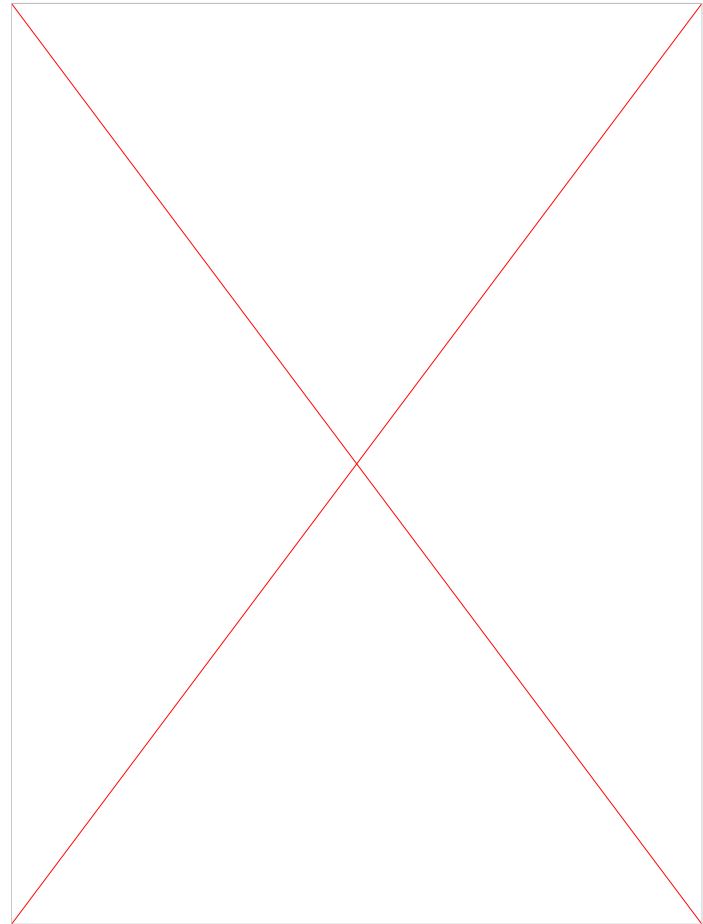
# Time



## ECE 373

# Prelims

- Questions on assignments, class?
  - Datasheets and marketing info, PCI
- Midterm?



# Time

- Measuring
- Storing
- Waiting
- Working

# Jiffies

- Time values
  - jiffies – global kernel time tick
  - HZ – number of jiffies per second
  - (versus tickless kernel)
- Operations
  - msec\_to\_jiffies(m), jiffies\_to\_msec(j)
  - nsec\_to\_jiffies(n), jiffies\_to\_nsec(j)
  - time\_after(a,b), time\_before(a,b)

# Low-level time

- `get_cycles()`
  - Arch independent CPU cycle counter
  - `#define'd` to 0 if not supported
  - Arch specifics underneath
    - Sparc 64: [http://lxr.free-electrons.com/source/arch/sparc/include/asm/timex\\_64.h#L15](http://lxr.free-electrons.com/source/arch/sparc/include/asm/timex_64.h#L15)
    - X86: <http://lxr.free-electrons.com/source/arch/x86/include/asm/tsc.h#L22>
- `rdtsc` – Time Stamp Counter register
  - CPU register in x86 and x86\_64 since pentium
  - 64 bit counter of clock cycles
- Good for low level timing – e.g. code profiling

# get\_cycles() example

- Time an operation

```
u32 c_start, c_done, c_duration;  
u32 answer;
```

```
c_start = get_cycles();  
answer = do_some_timewasting_thing();  
c_done = get_cycles();
```

```
c_duration = c_done - c_start;  
msecs = (1000 * c_duration) / cycles_per_sec;
```

# Time of Day

- `current_kernel_time()`
  - Updated by tick, running time-of-date, init'ed from RTC
- `do_gettimeofday()`
  - Read from HW, adjusted with nanosecs & arch specific tweaks
- `mktime()`
  - Takes min/sec/etc to make seconds since "epoch"
- RTC – i2c devices to keep Time of Day info
  - Standard i2c device interface
  - Standard RTC device interface
  - <http://lxr.linux.no/#linux+v2.6.38/drivers rtc/rtc-rx8025.c>

# Time example

- Time since "the epoch" – Jan 1, 1970, 00:00am

```
struct timeval tv;  
struct rtc_time rt;  
unsigned long secs1, secs2;
```

```
do_gettimeofday(&tv);  
secs1 = tv->tv_sec;
```

```
rtc->get_time(rtc, &rt);  
secs2 = mktime(rt->tm_year, rt->tm_mon, ...);
```

```
if (secs1 != secs2)  
    printk(KERN_INFO "Different times: %d != %d\n",  
           secs1, secs2);
```



# Delay

- Delayers
- Sleepers
- Timers
- Schedulers

# Delayers

- `mdelay()`, `udelay()`, `ndelay()`
  - While loop spin on a counter, no scheduler action
    - <http://lxr.free-electrons.com/source/include/linux/delay.h>
    - <http://lxr.free-electrons.com/source/arch/x86/lib/delay.c>
  - Only way to get short period delays
  - Not very friendly for long periods
  - Could block jiffies update if interrupts disabled
- Become very useful later when locking, interrupting...

# Delayers, nicely

- Scheduler/CPU friendly
  - `cpu_relax()` - arch specific, might not do anything
  - `schedule()` - give up timeslice
- `while (!is_device_finished())`
  - `schedule();`

# Sleepers

- `msleep()`
  - Give up timeslices for specific number of millisecs
- `msleep_interruptible()`
  - Same, but stop if a signal is pending
- <http://lxr.free-electrons.com/source/kernel/timer.c>

# Timers

- Basics
  - `setup_timer(t_var, t_callback, t_data)`
  - `mod_timer(t_var, interval)`
  - `del_timer_sync(t_var)`
- Callback function
  - Called when timer expires
  - Given `t_data` as an argument
  - `timer_cb(unsigned int data)`
- Timer scheduler code
  - <http://lxr.free-electrons.com/source/kernel/time/timer.c#L1205>

# Reading

- LDD3 – Chapter 7
- ELDD – Chapter 2, pgs 31-38
- Linux src
  - ../Documentation/rtc.c

