# Lecture 31, Nov 28, 2022

## Using Interrupts

- When an interrupt occurs:
  1. The CPU saves the pc of the interrupted instruction to `uepc`
  2. The CPU jumps to the interrupt handler address, as specified by the interrupt vector table `utvec`
  3. The interrupt handler code executes
  4. Execution returns to interrupted instruction
- To enable interrupts:
  1. Configure the specific device being used as the interrupt source
  2. Set `utvec` to the address of the interrupt handler
  3. Set interrupt enable bit in `ustatus`
  4. Set interrupt bit for interrupt source in `uie`
- The interrupt handler is like a subroutine, except it cannot use any registers without saving them, as interrupts can occur at any time
- Example: timer interrupt
  - `0xFFFF0018` and `0xFFFF001C` hold the current time count
  - `0xFFFF0020` and `0xFFFF0024` hold the timer comparison value (interrupt on hitting this value)

```
.data
curr_time:  .word 0xffff0018
time_cmp:   .word 0xffff0020
counter:    .word 0

.text
.global _start
_start:
    # Set up timer
    la s0, time_cmp
    li s1, 1000
    sw s1, 0(s0)
    # Enable interrupts
    la t0, timer_handler
    csrrw zero, utvec, t0
    csrrsi zero, utstatus, 1
    csrrsi zero, uie, 0x10
LOOP:
    # The processor can now do anything
    j LOOP

timer_handler:
    # Save registers as necessary
    addi sp, sp -12
    sw t0, 0(sp)
    sw t1, 4(sp)
    sw t2, 8(sp)
    # Increment counter
    la t1, counter
    lw t2, 0(t1)
    addi t2, t2, 1
    sw t2, 0(t1)
    # Set a new timer comparision value (TODO)
    lw t0, 0(sp)
    lw t1, 4(sp)
lw t2, 8(sp)
addi sp, sp, 12
uret
```