CMPUT 229

Lab 4: Clock Implementation

Lab 4

Goal: Implement a clock by creating an exception handler

Interrupts

- Must be enabled for this assignment
- Status register ($12) will need to be modified
- Set bit 0 to enable interrupts
- Set bits 11 and 15 to enable keyboard and timer interrupts

Timer

- Implemented by using registers $9 and $11 of coprocessor 0
- $9 is incremented every 10ms, if it reaches max int value, it resets
- If the value in $9 equals the value in $11, a timer interrupt is raised

Exception Handling

- When an exception occurs, the processor jumps to address 0x80000180
- Place your handler there by using .ktext 0x80000180 before your handler
- Disable default exception handler by running SPIM with -notrap flag
- Your handler will distinguish between timer and keyboard interrupts

Exception Handling

- Cause register ($13) will be useful in distinguishing interrupts
- Exception code is in bits 2-6 of cause register
- An exception code of zero indicates a hardware exception (timer/keyboard)

  Mask bit 11 (level 1 interrupt) to determine keyboard interrupts

  Mask bit 15 (level 5 interrupt) to determine timer interrupts

- On non-interrupt exceptions, increment the EPC by 4 (see exception.s)

Returning from Exceptions

- Restore registers saved into kdata or k0 or k1
- Restore $at
- Clear the cause register
- Re enable interrupts in the status register
- Return using eret

How to Access Coprocessor 0 Registers

- Read from these registers by using mfc0

  Example: mfc0 $k0 $12

- Write to these register by using mtc0

  Example: mtc0 $t1 $11

Memory-Mapped IO

- Allows interaction with external devices by pretending to be system memory
- Keyboard control register: 0xFFFF0000.

  Set bit 1 to enable interrupts. When one occurs, bit 0 is set.

- Keyboard data register: 0xFFFF0004

  The ASCII keycode of the last character typed is stored here

- Display control register: 0xFFFF0008

  Bit 0 is set when the display is ready for the next character

- Display data register: 0xFFFF000C

  When display is ready, storing an ASCII code here will have it written to screen

Assignment

- Upon starting, display 00:00 on screen
- Begin counting up immediately in **mm:ss** format.
- The clock must be updated in-place and not print across the screen.
- When 'r' is pressed, immediately reset the time to 00:00.
- When 'q' is pressed, quit the application.
- For all other key presses, do nothing.

Resources

- exception.s:

The default SPIM exception handler

Notes

- Use a .ktdata section to save registers in your handler
- Remember to save $at (look at exception.s)
- Your program will need a __start: label (look at exception.s)
- Remember to run SPIM using: spim -notrap -mapped_io
- The ASCII code of 8 (backspace) will be useful
    - for your printing to happen in one spot and not move across the screen

Tips

- Test your assignment on the labs machines before submitting.
- Format your code like example.s to get easy style marks.
- Use exceptions.s as a reference/starting point.