CS 310H **Assignment #10** **Out: April 19, 2007**
Prof. Keckler **Due: May 3, 2007**

Problem 1:

Explain (using an example) why interrupt priorities are necessary in a modern computer system. You should consider the number and types of devices (disk, keyboard, etc.) that might request an interrupt. What could happen if any device could interrupt any interrupt handler.

Problem 2: P&P 10.10, 10.12, 10.16

Problem 3:

Using the template hw10-4.asm, write a simple program that causes the illegal instruction exception to get triggered (i.e. executing an instruction that uses opcode 1101). You should be able to write just regular LC-3 code and not have to hack any object code to make this work. Turn in a hardcopy and electronic copy as hw10-4.asm.

Problem 4: Timer Interrupt Handler

In this programming assignment, you will implement a timer interrupt handler that swaps two user processes in and out of the LC-3 processor. The basic idea is that a timer interrupts the processor every fixed number of cycles. The timer interrupt handler is responsible for swapping the program that was interrupted with another program waiting to run. This is typically called a context switch in a preemptive multitasking environment. The two programs you will swap between are very simple:

- preempta.asm - this just prints a sequence of the character "A" to the screen
- preemptb.asm - this just prints a sequence of the character "B" to the screen

Since both A and B will be preempted periodically, you should see alternating sequences of "A"s and "B"s printed to the screen. The number of A's or B's printed in a row will depend on the time between timer interrupts. Just to remind you, the additional special registers and counters that are built into the hardware (and the lc3db simulator) are:

- MCR (machine control register). MCR[15] holds the run bit (we've seen this before). MCR[14] holds the timer interrupt enable bit. When this bit is set to 1, then timer interrupts are enabled. MCR[13:0] holds the timer interrupt limit - when the timer interrupt counter reaches this limit, an interrupt is triggered. This means that the maximum number of instructions (the counter counts instructions between interrupts) is 2^{14} = 16K. The memory-mapped address of the MCR is xFFFE.
- MCC is the interrupt counter. You can write any value to it, but after every instruction it is incremented. The memory mapped address of the MCC is xFFFF.

We will provide you with the `preempta.asm`, `preemptb.asm`, and the shell of the multitasking interrupt handler (`scheduler.asm`). To run these, make sure you load `scheduler.obj` last. Your mission (should you choose to accept it - do you have a choice?) is:

- Fill in the timer interrupt handler code so that it correctly swaps between the two programs. Note that you need to make sure that you save and restore all of the state of each program (a and b) every time you swap. Don't forget to reset MCC to zero after every swap. Turn in a hardcopy and an electronic copy (using the turnin program) of your new version of scheduler.asm.
- Now experiment with the interrupt limit register by reducing it to a smaller number. How many A's are printed in a row if you set the limit to decimal 2000? What if you set the limit to decimal 100? What happens when you set it to decimal 10?
- With the limit register set to 100, what fraction of the time in your program is spent in (1) the code that prints A, (2) the code that prints B, and (3) the code that swaps between the A code and the B code. Please describe how you computed this number.

Please note that you will not have to write very much code to solve this problem, but the code is a little tricky - it is very easy to get confused and mess things up. Start early so that you will have ample time to debug.