Computer Science 330
Project: Interrupt Handling in MIPS

Due: Fri. Apr. 23, at 11:59 p.m.. This project is worth 100 points.

Write a program that reads characters from the keyboard, and then prints the character, possibly modified, to the console. All reading and printing of the characters must be done in an exception handler that resides in kernel memory. Read and write to the console using the memory mapped reciever and transmitter control and data registers. *You must not use syscalls to read and write the characters.*

Your program should consist of three different files: A preamble, a main program, and an exception handler. The preamble and exception handler are partially done for you. You can download them from the course webpage.

The three files each have separate tasks.

The Preamble: Call this file "preamble.s". The preamble should contain code to allow your to process interrupts in your program. Write code to perform the following steps, in this order:

1. Enable interrupts in the co-processor status register by setting bit 0. Leave all other bits unchanged. The status register is co-processor register 12.
2. Enable interrupts in the receiver control register by setting bit 1. Leave all other bits unchanged.
3. Enable interrupts in the transmitter control register by setting bit 1. Leave all other bits unchanged.
4. Execute the main function by executing a jump-and-link to the label `main`.
5. When the main function returns, exit using syscall 10.

The Main Program Call this file "main.s". The first instruction in this program should be labeled "`main`". The main program should count down from 2^{23} to 0 in a tight loop. Once 0 is reached, return from `main()` by executing the instruction `jr $ra`.

The Exception Handler Call this file "exception.s". Add code to the exception handler to handle interrupts from the keyboard and console. When an interrupt occurs, check to see that both the reciever and transmitter are ready. If not, return from the interrupt handler without doing anything. If both devices are ready, read a character from the receiver data register. If the character is an an upper case letter, leave it alone. If he character is a lowercase letter, change it to upper case. If it is any other character, change it to a dash '-' (ASCII char 45). Then, print the possibly modified character on the console by storing it in the transmitter data register. After printing the character, return from the exception.

Important: Before you load your program into SPIM, make sure that the emulator is configured properly to use memory mapped IO and a custom exception handler. To do so

1. Open the `Simulator` menu and choose `Settings`.
2. Click on the tab labeled `MIPS`.
3. Under `MIPS Simulator Settings`, make sure that `Enable Mapped IO` and `Accept Pseudo-Instructions` are both checked, but nothing else is. Under `Exception Handler`, make sure `Load Exception Handler` is not checked.

When you load your source files into MIPS, load "`preamble.s`" using the `Reinitialize and Load File` command. Load the other two using the `Load File` command. `Reinitilize and Load File` clears all of SPIM's memory (text, data, and stack segments in both user and kernel space) before loading a source file - `Load File` does not.

What to turn in: When you have finished with the program, upload a tarball containing all three files to the appropriate place on blackboard. Then, print a hard copy of the source to turn in. Assemble them in the order 1) preamble, 2) main, 3) exception handler. You must submit your hardcopy within one class day of your electronic submission for the submission to be considered as one time.