CSCI-21 Final programming project. Due before start of final exam, no late projects accepted

You may turn off branch and load delays for this program, if you wish. It will make it considerably simpler. Be sure to turn on memory-mapped I/O. Make sure you fully understand what the memory mapped I/O example program from my Web site does before you start designing or programming this. You will not be polling, but you need a clear understanding of memory-mapped I/O.

You may not use syscall for any input or output, because the program would then block on the input. Instead, use an interrupt-driven I/O routine to handle all input and output. **You must do ALL input and output through the memory-mapped input/output ports, NOT through syscall. You may NOT poll for I/O, either.**

Write a program that starts with two arrays of characters (.asciiz strings), of equal length, labeled 'source' and 'display'. Initially, make the source array contain a character string with a '\n' before the terminating NUL. Include upper- and lower-case letters, digits and whitespace freely in the string. The string must be at least 40 characters long. Start your program by copying the source array into the display array. Use a subroutine to do this passing in the addresses of the two arrays on the stack. Use the "real-world" subroutine calling convention.

After copying the string, enable interrupts then loop examining a named variable (which may change value inside the interrupt handler) until the user tells the program to quit. Whenever an output ready (transmitter) interrupt occurs, the interrupt handler will print the next character from the display array (one character per interrupt), wrapping back to the beginning after it prints the '\n'. Whenever the input interrupt (receiver) interrupt occurs, you will extract the user's input (a single lower-case character) and do one of the following tasks, depending on the user's input:

's': sort the display array using any easy sort routine (bubble or ripple is fine).

't': toggle the case of every alphabetic character (for example, 'T' becomes 't', 't' becomes 'T' and all non-alphabetic characters stay unchanged).

'a': replace the display array elements with the source elements once again.

'r': reverse the elements in the display array (excluding the '\n').

'q': quit -- terminate program execution.

Ignore any other character in input. Handle the commands from the user inside the interrupt handler, not inside the main program.

On the 'q' command, the interrupt handler will set the variable being queried by the main program. This will cause the main program to leave its loop and exit with a syscall 10.

Note: an interrupt could happen in the middle of displaying the array, so the characters being displayed could appear to change mid-line. This program is obviously designed to build on previous assignments, so you should reuse code wherever possible. The primary new feature is the interrupt-driven input and output. Instead of reading the entire string in one syscall, you will read and process each character as it arrives. Again, you MAY NOT use syscall for any I/O. This is approximately three or four pages of code in total.