Laboratory 7: Inlab

Date ______________________ Section ______________________

Name ____

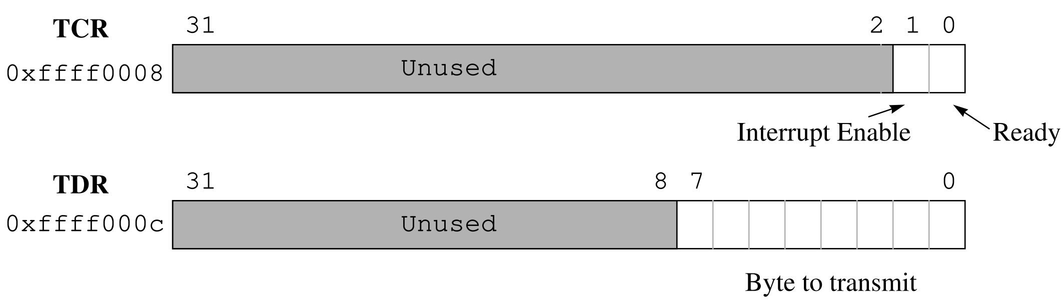
Interrupt Driven Output

Your job is to build the set of procedures needed to do interrupt driven output to the terminal.

Introduction

As described in the Prelab exercise, SPIM simulates a memory-mapped terminal connected to the processor. Memory mapped means that accessing some memory locations accesses the I/O devices in reality. Writing to a specific memory location will actually write to an output device, while reading from some specific memory location actually reads from an I/O device. The terminal device SPIM simulates consists of two independent units: the *receiver* and the *transmitter*. The receiver reads characters from the keyboard as they are typed. The transmitter unit writes characters to the terminal's display.

The processor accesses the transmitter using two memory-mapped device registers:



The Transmitter Control Register (TCR) is located at address 0xffff0008. Bit 0 (LSB) called 'Ready' is read-only. If it is one, then the transmitter is ready to accept a new character. Bit 1 called 'Interrupt Enable' can be read or written. If the user sets it to one, then an interrupt will be requested on level one whenever the ready bit is one (whenever the transmitter is ready to accept a new character).

The Transmitter Data Register (TDR) is located at address 0xffff000c. The least significant byte can be written and will be interpreted as an ASCII character to output to the display.

Step 1

Write a program called *lab7.2.asm* which does:

- loops 100 times: inside the loop it calls the procedure *PrintStr* to print the message "Hello world"
- the procedure 'PrintStr' receives a pointer to the string to print in **`$a0`**, and returns nothing. The pro-

cedure will copy the output characters in a buffer shared with the interrupt routine

The 'PrintStr' procedure will not manipulate the TDR: the interrupt routine will do it. The only interaction PrintStr has with the device registers is to make sure that interrupts are enabled.

The buffer 'PrintStr' shares with the interrupt routine is a circular buffer with a capacity of 256 bytes. The buffer is in the user space. The names for the two pointers in the circular buffer will be `t_in` (to indicate where to put a new char in the buffer) and `t_out` (to indicate where is the next character to send).

Since the program generates characters to print much faster than the transmitter can print, the buffer will be eventually filled: in a real system the operating system would block the currently executing process (the one who has filled up the buffer) and would let another process run. Since the SPIM simulator is a single-user mono-programming environment (as opposed to a multi-programming one), the procedure 'PrintStr' should handle the case when the buffer is full. One approach is to repeatedly check the buffer until it is no longer full.

Turn the interrupts on when 'PrintStr' deposits something in the buffer. This works together with the interrupt routine which will turn the interrupts off when the buffer is empty.

Step 2

Modify the exception handler as to accept interrupts at level 1. If the handler is invoked because of such an interrupt, then call a procedure named *PutChar* which will take a character from the circular buffer and store it into TDR. Of course, if the transmitter is not ready, then just return: an interrupt should not have been produced in the first place in this case, but it does not hurt to test.

If the buffer is empty then turn the interrupts off. Otherwise interrupts will be generated all the time to indicate the transmitter is ready.

Make also sure you properly set up the interrupt mask at level 1 (IM1) and the interrupt enable (IEc), every time you enable or disable interrupts.