Lab 02
MARS polled keyboard and display memory-mapped IOs

1 Objectives

1. To understand the basic polled Input/Output (IO) control method in the Microprocessor without Interlocked Pipeline Stage (MIPS) Microprocessor (μP).
2. To understand the basic concept of memory-mapped IOs.

2 Pre-requisites

For this lab, MIPS Assembler and Runtime Simulator (MARS) must be already installed and you must be familiar the basic steps for running and debugging MIPS assembly programs.

3 Background - MIPS memory-mapped IOs

MIPS μP employs memory-mapped IO control mechanism. More specifically, MIPS reserves memory locations `0xffff0000` to `0xffffffff` within the main memory for communicating with IO devices. From these 64K addresses available, address `0xffff0000` and `0xffff0004` handle inputs from the keyboard. These two addresses are structured as shown in Figure 1.

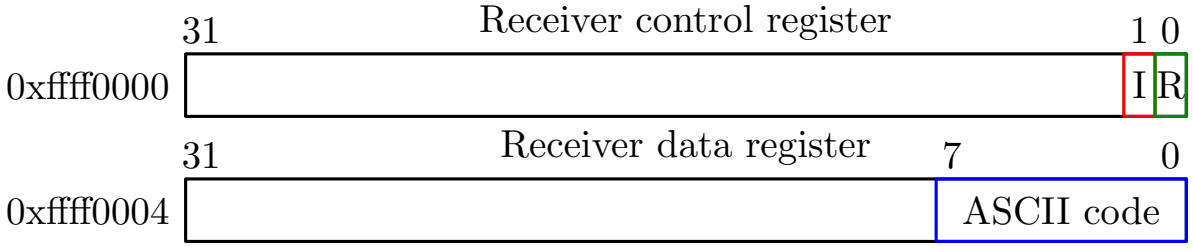

Figure 1: MIPS keyboard input registers.

As shown in Figure 1, address `0xffff0000` is the Receiver Control Register (RxCR). Here, only the two Least Significant Bits (LSBs) are used for indicating whether a key has been pressed. More specifically, bit 0 of RxCR is a ready bit, labelled as `R` in Figure 1. This is a read-only bit, which is set to 1 whenever a key has been pressed. This ready bit is then cleared when the input data from the keyboard is read. Bit 1 of RxCR is an interrupt-enable bit, labelled as `I` in Figure 1. If this bit is set to 1 by the programmer, it will indicate that the μP must acknowledge interrupts whenever a key is pressed.

Address `0xffff0004` of Figure 1 corresponds to Receiver Data Register (RxDR). The 8 LSBs of this read-only register contain the American Standard Code for Information Interchange (ASCII) code of the last pressed key.

4 Lab work

The following sections specify the requirements for this lab.

4.1 Exercise 1 - Polled keyboard input

For this part of the lab, you must create a MIPS assembly language program for polling keyboard inputs. Use the information provided in Section 3, or the slides provided during our lectures, in order to identify the MIPS memory locations that are relevant to the keyboard input. You program for this part of the lab should echo (print) in the **Run I/O** terminal pane, the key pressed in the keyboard using `syscall` instructions. Your program should not finish, *i.e.*, it should continuously run and print out the pressed keys without having to reset the simulation.

In order to help you debug your program, it may be useful to select the **MMIO** segment in the **Data Segment** pane within MARS. Additionally, it might be useful to check the ASCII box, as shown in Figure 2.

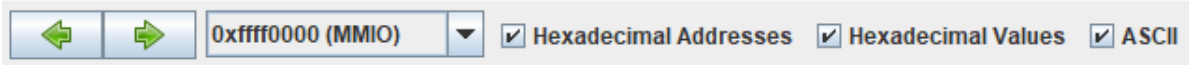

Figure 2: MARS memory-mapped IO data segment.

The following section presents details on how we can simulate a keyboard interface in MARS.

MARS memory-mapped IO simulator

MARS simulator features a tool for simulating the communication between a MIPS μP and IO devices, such as a keyboard. This tool may be accessed by the menu **Tools** $\rightarrow$ **Keyboard and Display MMIO Simulator**. This should bring up the **Keyboard and Display MMIO Simulator** window show in Figure 3.

After successfully assembling your program, click on **Connect to MIPS** to interface the keyboard simulator with your code. Run your program step-by-step and press a key inside the keyboard simulator - you may have to click inside the text box before pressing a key. Verify that the **Run I/O** terminal echoes the key that is pressed in the keyboard.

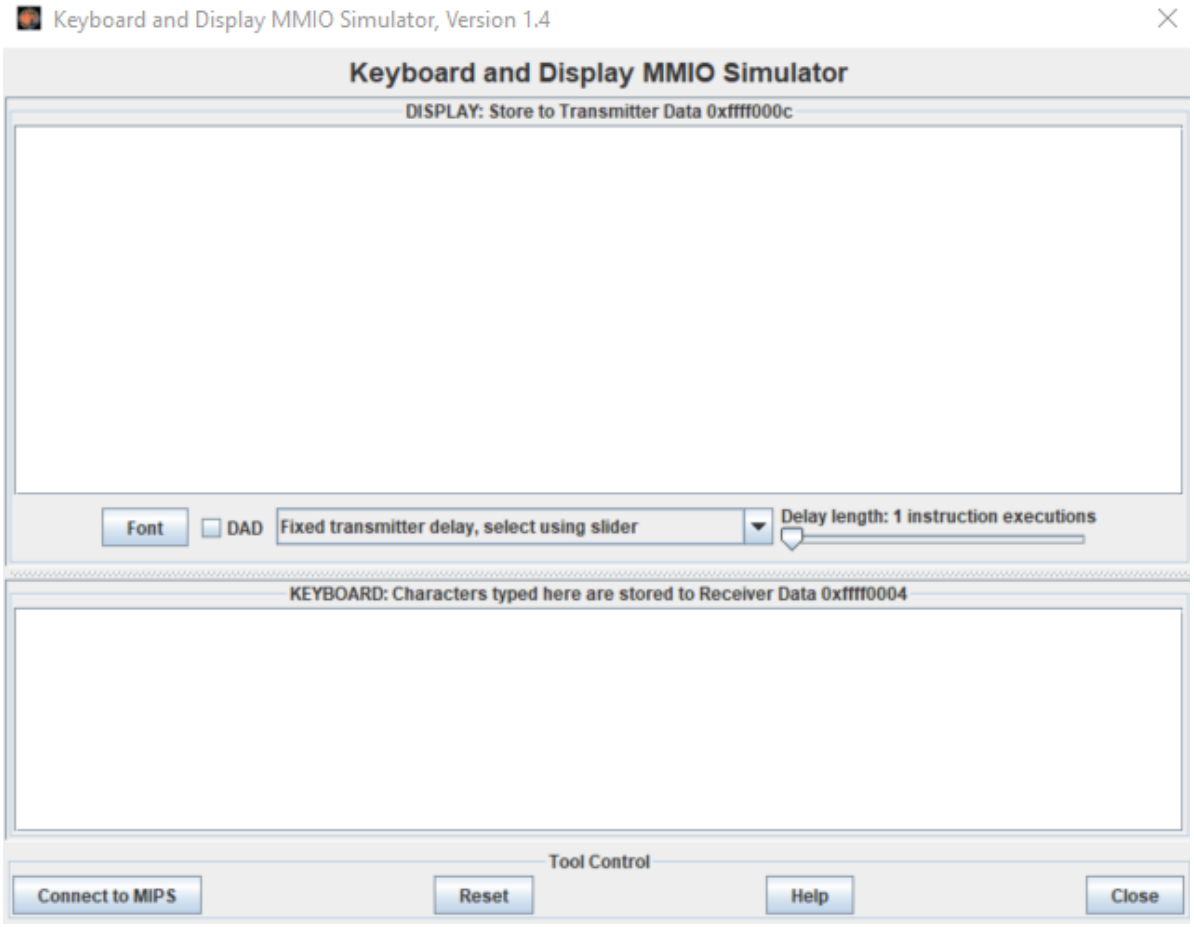

Figure 3: MARS IO simulator .

4.2 Exercise 2 - Polled display output

Similarly to the MIPS keyboard input registers of Figure 1, we may simulate a display terminal using MIPS memory-mapped IO registers. More specifically, memory addresses `0xffff0008` and `0xffff000c` of Figure 4 correspond to the Transmitter Control Register (TxCR) and Transmitter Data Register (TxDR), respectively.

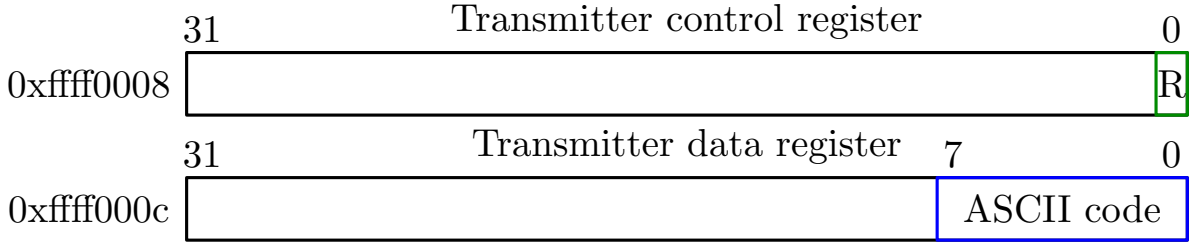

Figure 4: MIPS display output registers.

Here, the LSBs of TxCR is a ready bit, labelled as `R` in Figure 4. This read-only ready bit is used for indicating whether data is ready to be sent to the display. This ready bit is then automatically cleared by MARS when data is displayed. Address `0xffff000c` of Figure 4 corresponds to TxDR. The 8 LSBs of this register are used as the ASCII code to be sent to the display.

For this exercise, you must create an assembly program for displaying the key pressed using

polled IO method. More specifically, for this part of the lab you should poll both the keyboard and the display. You may be able to poll the display since MARS should automatically set the ready bit of TxCR after each successful display output. Your program should not finish, *i.e.*, it should continuously run and print out the pressed keys without having to reset the simulation.

TIPS

- In order to help you debug this part of the lab, make sure that the **DAD** box within the **Keyboard and Display MMIO Simulator** is unchecked and that the **Display length** slider is at the farthest left, as shown in Figure 3. This will avoid any delay between the moment you write data into TxDR and the moment it is displayed.
- Reset the display each time you have to re-assemble your code (after each time you modify something in your code) and after each time you have to reset your simulation. MARS interface is not 100% bug free and I've encountered situations where the ready bit of TxCR is stuck at zero and data cannot be written into the display. This is detailed in the following help message provided by MARS.

  IMPORTANT NOTE: The Transmitter Controller Ready bit is set to its initial value of 1 only when you click the tool's 'Connect to MIPS' button ('Assemble and Run' in the stand-alone version) or the tool's Reset button! If you run a MIPS program and reset it in MARS, the controller's Ready bit is cleared to 0! Configure the Data Segment Window to display the MMIO address range so you can directly observe values stored in the MMIO addresses given above.

5 Deliverables

1. Exercise 1 - Polled keyboard input
   (a) Your `.asm` or `.s` file with your assembly code.
   (b) A short video demonstrating your program working correctly.
      i. Your video should show MARS **Run I/O** pane echoing (printing) the key pressed in the keyboard pane within the **Keyboard and Display MMIO Simulator**, *i.e.*, your video should show both the **Run I/O** pane and the **Keyboard and Display MMIO Simulator** at the same time.
      ii. In your video, you should type your student ID number as proof of your own work.
2. Exercise 2 - Polled display output
   (a) Your `.asm` or `.s` file with your assembly code.
   (b) A short video demonstrating your program working correctly.
      i. Your video should show the display pane on MARS **Keyboard and Display MMIO Simulator** window echoing (printing) the key pressed in the keyboard pane within the **Keyboard and Display MMIO Simulator**.
      ii. In your video, you should type your student ID number as proof of your own work.

Submit your assignment through Canvas before 23:59 hours on Thursday September 3rd 2020.
Please send any questions to isaac.perez.andrade@tec.mx.