You will learn the following in this lab:

- Designing a memory-mapped I/O system
- Integrating your I/O devices from an earlier lab with the MIPS CPU and memories

Part 0: Thoroughly test your design of the MIPS CPU from Lab 8

Before you proceed with the tasks below, make sure that your MIPS CPU from Lab 8 is working correctly. The tester provided for Lab 8 should result in all correct outputs (i.e., all ERROR signals should be green).

Part 1: Integrate the CPU and the display unit using memory mapping

As discussed in Lecture 14 (figures reproduced below), you will integrate the CPU and the display using memory-mapped I/O. The desired memory mapping scheme was discussed in class (see next page).
Assigning the data memory to start at address 0x1001_0000 allows you to use the MARS assembler with the "Default" configuration, which places code at 0x0040_0000.

We will simply drop the upper bits of the address coming into instruction, data and screen memories. This happens automatically if you use the memory templates provided earlier, which have the number of address bits automatically calculated using the number of memory locations, i.e., $A_{bits} = \lceil \log_2(N_{Loc}) \rceil$. 

Assigning the data memory to start at address 0x1001_0000 allows you to use the MARS assembler with the “Default” configuration, which places code at 0x0040_0000.
To implement this memory map, use the block diagram below. (A more detailed figure appears on the next page.) Put the “Memory and I/O unit” in a module called `memIO`, and name the file `memIO.sv`. You should start with integrating the screen memory into `memIO`, and then proceed to also integrate the keyboard, accelerometer, sound and LED lights modules (from Lab 5).

![Block Diagram](image)

NOTE: The screen memory from Lab 7 Part 3 has now been placed inside the “Memory and I/O unit”. Therefore, the display driver now outputs the address for the screen memory, which goes into the memory-I/O unit through a port (shown on the right side in the figure above). This port is distinct from the port used by the MIPS processor. Thus, the screen memory now has separate interfaces to the display driver and to the MIPS CPU.

Below is a more detailed version of the figure showing all of the connections between the modules.
Carefully design your memIO unit by adhering to the hierarchy and connectivity shown in the detailed figure. You may, of course, choose different names for the wires inside the memIO module, as long as you use them consistently.

Note that the name of a wire may be different inside a module compared with outside. For example, the memory address generated by the MIPS CPU is labeled mem_addr outside the memIO unit, but called cpu_addr inside. The name used inside, cpu_addr, was conveniently chosen to distinguish it from the other address input to the memIO unit, called vga_addr, which is generated by the VGA display driver for looking up the current character code for display. Outside the memIO unit, vga_addr also has a different name; coming out of the display driver, it is called smem_addr (see picture on previous page).

Implement this part on the boards! Write a short program to have your MIPS write characters to Screen Memory and see if they show up on the monitor! Sample programs are provided on the website as well. If all goes well, you will see some activity on the screen. And you will have a full-function computer.

Good luck!
Start working on your final project demo idea. As explained in class, submit a short proposal on what you would like to implement for your final project demo via Sakai by Monday April 9, before start of lab so we can give you timely feedback.

What to submit:

- The Verilog sources for the top module (top.sv) and the memIO module (memIO.sv).
- In a couple of sentences, state if everything works as you expect, or if there are some problems you still need to resolve.
- Show a working demo of your design Apr 11 in the lab session.

How to submit: Please submit your work by email by 1:25pm Apr 11 (before lab session):

- Send email to: comp541-submit-s18@cs.unc.edu
- Use subject line: Project PART A
- Include the Verilog file as attachment as specified above, and your statement about any unresolved bugs.