Computer Architecture

Assignment 4

Problem 1

Let's follow up on Lab 5. In that lab, you wrote the *polling loop* for the keyboard. However, we still used `syscall` 11 to echo the character to the console window. For this problem, modify your Lab 5 code (or use the code I provided) to also implement the polling loop for the display. It works in an analagous fashion to the keyboard. The computer puts the data to be displayed in a Transmitter Data Register (TDR). The monitor will then read that data and set bit 0 of the Transmitter Control Register (TCR) to let the computer know it's time to load the next byte into the TDR. The TDR is *memory-mapped* to address `0xffff000c` and the TCR to `0xffff0008`. Your loop should load a byte into the TDR (the byte you received from the keyboard), wait for the TCR's bit 0 to turn to a 1, and then load another byte into the TDR, ad infinitum. In the MARS Keyboard and Display MMIO Simulator, you should see the keystrokes from the lower window echoed to the display in the upper window.

Problem 2

For each of the following design problems, (1) write a logic function to describe its behavior and (2) draw the gate-level implementation of your logic function. This material is covered in sections B.1 and B.2 of your text.

(a) A security system has an active-low master switch (M) and active-high sensors (S1 and S2) as well as an active-high test button (T). The active-low alarm (A) should sound (logic-0) when the master switch is active and either sensor is active or when the test button is pressed (even if the master switch is inactive.)

(b) The controller for an automatic door at a grocery store will open the door (D is logic-1) when it detects activity on an active-low presure sensor (P), when an active-high override switch is turned on, or when the active-low safety stop switch is tripped.

(c) Consider the controller for a lighting system. There is an active-high motion sensor (M) and an active-low sensor (S) that turns on when the level of light filtering into the space from outside is above a certain threshold. The system needs to turn on active-high internal lights (L) when the motion is detected and the outside light is low.

Problem 3

For the designs in Problem 2, we can come up with Boolean Algebra models by thinking through the logic and writing down the equation. For many problems, however, the logic is too complex for that. For these situations, we can use a **truth table** to help guide us. To build a truth table, we write down all possible inputs and then assign each of those inputs a corresponding output value. We call this process **specifying** the logic function. We can then take the function we have

specified and compute its logic function from the truth table. (Note that a better name for the truth table would be the *function specification table*, but we are stuck with the name *truth table* due to historical reasons.)

For each of the following problems, (1) write down the truth table for the requested functions, (2) write the corresponding logic functions, and (3) draw the gate-level implementation.

(a) Design a prime number detector. Input a four-bit unsigned integer $x_3x_2x_1x_0$ and output a 1 if the number is prime.

(b) Design a device to compute complements. Input a 3-bit two's complement number $x_2x_1x_0$ and output its three-bit complement $y_2y_1y_0$. Note that for this one we have three output functions, as each bit y_2, y_1, and y_0 in the complement is a separate column that needs to be specified in the truth table. Draw the gate-level implementation of all three functions in the same circuit in the manner shown in figure B.3.4 on page B-15 of the text.