# CDA 4203 Sec 001 Spring 2015
# Computer System Design
# Instructor: Dr. Srinivas Katkoori
## Homework 2: PicoBlaze Microcontroller
*Handed out on Monday, 26th January*
*DUE: 11:59:59PM, Monday, 2nd February*

Note:
1) Recommended submission is by Canvas.
2) If you handwrite the answers, you may want to scan and upload to Canvas.
3) For all questions, that need assembly program, you should validate in pBlazeIDE and submit the code. You will get credit only if it works in pBlazeIDE.
4) Deliverables:
   (a) Your answers to the problem in a PDF document
   (b) Your PicoBlaze code validated in pBlaze IDE

**Your Name:** **Your U#:**

1. **(5 pts.) PicoBlaze Architecture**

   Answer the following questions about the PicoBlaze Architecture:
   a. List all storage units and what kind of data they can store (data, instructions, addresses)?
   b. What is the address range (in hex) for the instruction PROM?
   c. How many cycles does an instruction consume?
   d. What is the purpose of NOP instruction?
   e. List all instructions that ***directly*** manipulate the call/return stack.
   f. How many clock cycles does instruction decode take?
   g. What is the purpose of the interrupt input? How can we disable interrupts?
   h. What flags does the test instruction affect? Does it affect the register contents?
   i. When we execute a call instruction, does the PicoBlaze save the register file contents?
   j. What is the difference between ADD and ADDCY instructions? Explain why we need two variants of ADD instructions?

2. **(5 pts.) PicoBlaze – Word Parity:** Write an assembly program to compute the parity of a given word. A word has an *even* parity if it has even number of 1's (e.g., 01101001). Otherwise, it has odd parity (e.g., 10101110). Assume that the word is presented on the IN_PORT (for only two clock cycles). The parity result must be presented on the OUT_PORT. The procedure must output 000hex if the word has even parity and 001hex if it has odd parity. Disable interrupt handling during the parity computation and re-enable when done. Simulate your program using pBlazeIDE simulator.

3. **(15 pts.) Simple Loopback System**[1]**:** Consider a simple "loopback" system as in Fig. 1. In such a setup, a PC can send a message which is returned (looped) back to PC by the processor. Such a loopback test is helpful to test if the processor is alive or not.

   A PC (not shown) interfaces with PicoBlaze processor using an RS232 (serial) connection. Let us examine the case when the PC acts as a transmitter. In this case the data arrives serially on RS232_RX line. But we know that PicoBlaze accepts only byte-size data. So we need to convert the serial data into parallel (8-bit) data. In such a case, a UART (Universal Asynchronous Receiver Transmitter) block can help. UART_RX does the serial-to-parallel conversion i.e., the data arriving on RS232_RX line is buffered in a 16-byte buffer. Whenever new data is presented on rx_data output, UART_RX

[1] Adapted from the assignment created by Eric Crabill at San Jose State University.

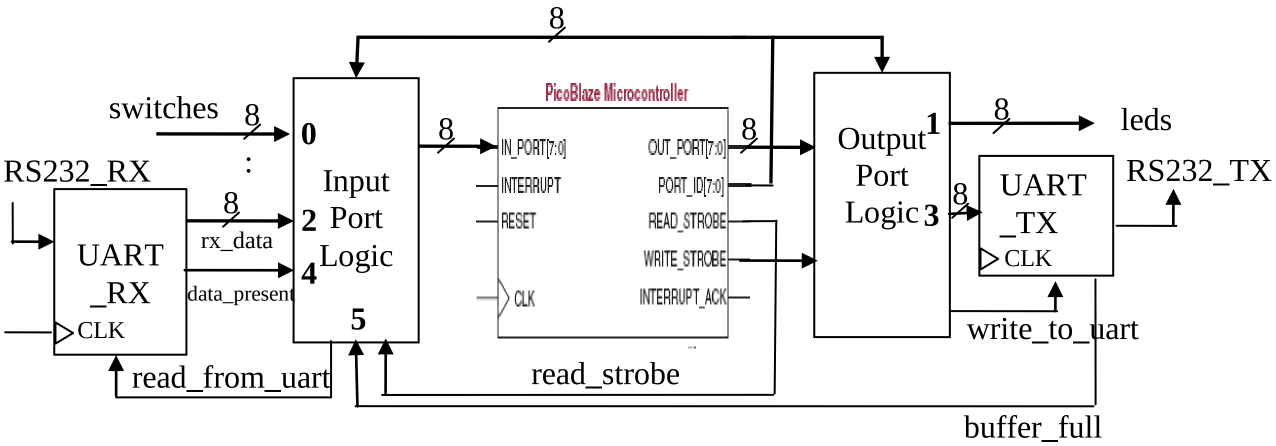

Figure 1: A simple loopback system

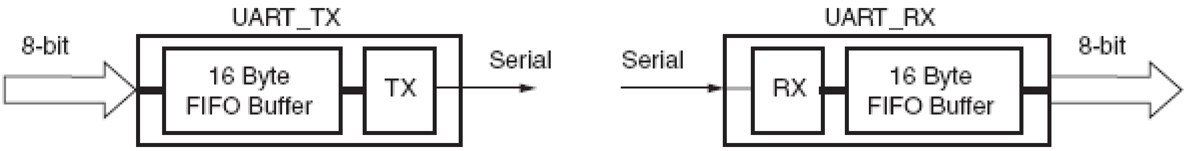

Figure 2 : UART Transmit and Receive Blocks – High Level View
*Note that for the sake of clarity, we have not shown the clock and reset signal connections.*

will assert data_present i.e., data_present = '1' in other words data_present validates rx_data. The "Input Port Logic" block consists of glue logic that connects UART_RX to the IN_PORT. After PicoBlaze reads the data, it can request the next data item by asserting read_from_uart signal.

Now, let us consider the case when PicoBlaze acts as a transmitter i.e., PC is the receiver. In this case, PicoBlaze puts out 8-bit data while PC is expecting serial data. Again, UART (UART_TX in Fig. 1) comes to our rescue! The output bus with id=3 is connected to UART_TX. Output Port Logic block interfaces PicoBlaze with leds and UART_TX.

In this system, we can also manually provide 8-bit data via 8 switches (switches bus in Fig. 1). PicoBlaze can write 8-bit data to drive 8 leds (leds bus in Fig. 1).

Recall that PicoBlaze can accept up to 256 input ports each of which can addressed by 8-bit PORT_ID bus. Similarly, it can drive up to 256 ports and the address of the port being driven appears on PORT_ID bus. In Fig. 1, for example, the id of the input port, switches, is 0. Similarly, the rx_data has id of 2. The port ids are shown in **bold** font in the Fig.1. The high-level views of UART in transmit and received modes are shown in Fig. 2.

Answer the following questions:

a) **(2 pts.)** Explain under what condition(s) should the following signals be asserted?
   i. read_from_uart
   ii. write_to_uart
b) **(4 pts.)** cold_start: Write code to output a message "Welcome to Loopback!" to the serial port. You need encode the message in ASCII format. For ASCII table see http://www.asciitable.com
   *Caution: UART_TX buffer size is only 16 bytes!!*
c) **(4 pts.)** led_echo: Write code to read switches and write it, inverted, to the LEDs.
d) **(5 pts.)** rs232_echo: Write code to check if a byte has been received by UART_RX. If so, send it back to PC via UART_TX.