Homework 2 – ECEN 3350 – Programming Digital Systems – Fall 2017

Due: By date specified in D2L dropbox.
To hand in via D2L dropbox: Submit 1 file named: LastnameFirstname_homework2.s which contains the assembly source for your program. I assume you'll include address_map_nios2.s in your source file.

The purpose of this homework assignment is to learn how to write programmed IO (also known as polling IO) code to control a UART (Universal Asynchronous Receiver Transmitter) - a hardware device capable of transmitting characters in two directions. When developing embedded systems, it is useful to have the ability to print strings to a terminal window and/or enter strings and commands that can be interpreted by the code you are writing.

Assignment: This program is not an infinite loop, it runs through 1 time, and then halts. The UART's input and output is via the terminal window in AMP. When you type a character in the terminal window, that character is transmitted over the USB cable from the PC to the receiver port of the UART in the PDS Computer System. When software running on the PDS Computer System transmits a character it is transmitted over the USB cable and displayed in the AMP terminal window. See the UART section in the DE10-Lite_PDS_Computer.pdf. See the relevant section in the DE10-Lite_User_manual.pdf.

Program Requirements:

1) The program shall prompt the user in the terminal window:

Press the enter key to begin

Note: I show the characters in this document in bold only to make the characters stand out. There is no requirement to display the characters in bold.

Only the newline character 0x0a will be accepted in order to proceed. All other characters entered by the user will be ignored. When the newline character is entered, the program proceeds.

2) The program shall print the string in the terminal window on a new line:

Hello World!

3) The program shall prompt the user in the terminal window on a new line:

Press the enter key to continue

Only the newline character 0x0a will be accepted in order to proceed. All other characters entered will be ignored. When the newline character is entered, the program proceeds.

4) The program shall print the string in the terminal window on a new line:

We are done!

5) The program shall execute a branch to self to implement the halt function. Example:

```
self:        br       self
```

The characters displayed in the terminal window should look like:

```
Press the enter key to begin

Hello World!

Press the enter key to continue

We are done!
```

A useful assembler directive for creating the strings in this assignment is:

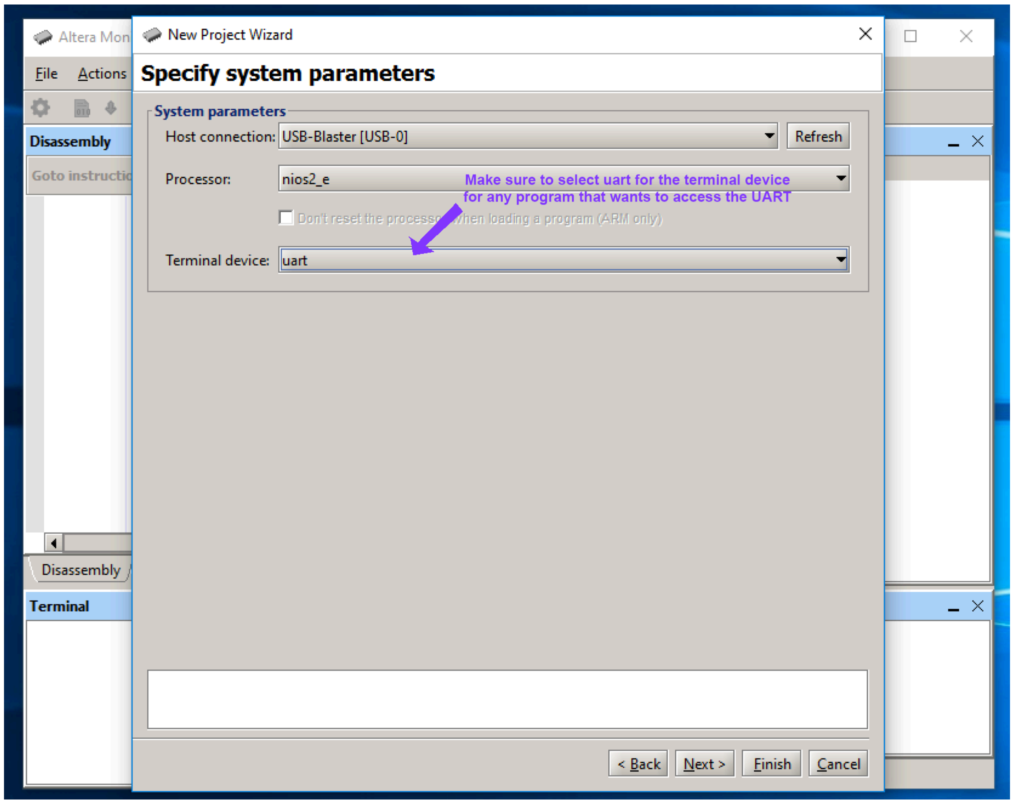
```
                 .data


                 # A null (0) terminiated string.
press_ent_str:   .asciz       "Press the enter key to begin\n"
```

Note: Students are strongly encouraged to write a subroutine that accepts a pointer to a null (zero) terminated ASCII (UTF-8) string (passed in r4) and sends these characters to the transmit UART. The alternative is writing straight-line code to implement the requirements for this assignment. Straight-line code is not as reusable as a subroutine. Students are strongly encouraged to write a subroutine to poll for the enter key being typed in the terminal window. These two subroutines will be useful to you in future homework assignments.

To get started, build a new AMP project, just as you did for the previous homework assignment, using the DE10-Lite PDS Computer system and the **HW Template No Interrupts** assembly file (the filename that gets created is **name_homeworkN.s**). Edit/Add your code to this file, and when complete you can copy that file into a new file for submission to D2L using the above naming convention.

Note:



Host connection: USB-Blaster [USB-0]

Processor: nios2_e

Terminal device: uart

Make sure to select uart for the terminal device for any program that wants to access the UART

Important: Please please please spend time designing your program before you start coding. I cannot emphasize this enough. Create a flow chart, write some pseudo-code, write a C-like program, think through your logic. Think about what variables need to be held in CPU registers. Also, whatever you do, do not wait until the last minute to start designing your program and coding up your solution. Your first few programs will take more effort than you may initially estimate.