UNIVERSITY OF BRITISH COLUMBIA
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

EECE 259: Introduction to Microcomputers
Assignment 2: Programming the UBC-DE259 Nios II Computer
Handed out February 4, 2011. Revised Feb 5,6. Current Version 1.02

Overview

In this homework, you should learn how to write simple Nios II programs that can perform simple I/O to read switches, write to LEDs, and measure elapsed time. You learn how to control execution and debug your using the Altera Monitor Program.

PART 0: Reference Material

Here are some good sources of information that can help you with this assignment. You should look at all of this information in the order presented below:

1. Course lecture notes.
   http://courses.ece.ubc.ca/259/lectures/

2. ***ubc-de1media-macros.s*** This file defines the special I/O addresses and constants. Place this file with your program and put **.include "ubc-de1media-macros.s"** in your program.
   http://courses.ece.ubc.ca/259/homework/files/ubc-de1media/ubc-de1media-macros.s

3. ***Altera Monitor Program*** tutorial. This tutorial explains how to assemble, download, and execute your program on the DE1 board.
   http://courses.ece.ubc.ca/259/homework/files/altera/Altera_Monitor_Program_Tutorial.pdf

4. ***Introduction to the Altera Nios II Soft Processor*** tutorial. This tutorial explains key assembler directives, important Nios II instructions, and describes a Nios II computer system. The example program in this tutorial has a small error, so use the corrected version below.
   http://courses.ece.ubc.ca/259/homework/files/altera/Altera_Nios2_Introduction.pdf
   http://courses.ece.ubc.ca/259/homework/files/altera/Altera_Nios2_Introduction.s

5. ***Altera DE1 Media Nios II Computer*** files for a specific Nios II computer system designed by Altera. The UBC-specific variation of this computer adds the COUNTER feature.
   http://courses.ece.ubc.ca/259/homework/files/ubc-de1media.zip (zip file for download)
   http://courses.ece.ubc.ca/259/homework/files/ubc-de1media (unzipped version)

6. **The *Nios II Processor Reference Handbook*** is the authoritative reference manual for all details about the Nios II processor. Chapter 8, the ***Instruction Set Reference***, carefully explains each instruction and pseudoinstruction in full detail.
   http://courses.ece.ubc.ca/259/homework/files/altera/n2cpu_nii5v1.pdf
   http://courses.ece.ubc.ca/259/homework/files/altera/n2cpu_nii51017.pdf

7. The full ***Nios II Documentation*** available from Altera at:
   http://www.altera.com/literature/lit-nio2.jsp

PART 1: Study Questions

Write a Nios II program *and verify your solution* by running the program on your board.

Do these problems before attempting the Practical Assignment !

1. The program below behaves like a wire, copying the switch values to the red LEDs.
   a. Try it out.
   b. Modify the program so the LED ***remains on*** even after you move the switch to "off". There are several solutions, but the simplest changes 1 instruction and adds 2 more.
   ```
   .include "ubc-de259-macros.s"
   .global _start
   .text
   _start:    movia r23, IOBASE
   loop:      ldwio r2, SWITCH(r23)
              stwio r2, LEDR(r23)
              br    loop
   ```
2. Write a short program to compute A*B+C/D. Assume the values have been defined according to the constants and data segment shown below.
   ```
   .equ  B, 5
   .equ  D, 6
   .text
        /* write your program here */
   .data
   A:
   .word 7
   C:
   .word 24
   ```
3. Modify the program in Question 1 so it copies the switch settings to the red LEDs only when KEY3 changes from a "0" to a "1". That is, your program should behave like a "DFF" where the inputs (switches) are captured only on the rising edge of clock (KEY3). Be sure the program does not behave like a "flow-through" latch.

4. Write a program to count the number of times switch SW0 is moved to the "1" position, and display this count in binary on the red LEDs. Make sure you wait for the switch to go back to "0" before counting the next "1". When testing your program, you will notice the count sometimes goes up by more than 1. Why do you think this happens?

5. Modify the program in Question 4 to display the count on the 7-segment display in hexadecimal format. You only need to display one digit (counting from 0 to F), but make sure you turn off the LEDs for the other digits.

6. Write a program that "decompresses" by reading two words from the COMPRESSED section below representing COUNT and VALUE, then writing out the word VALUE exactly COUNT times in the DECOMPRESSED section. Repeat this until you find "COUNT=0".
   ```
   .data
   COMPRESSED:
   .word 3, 0xEECE, 2, 0x0259, 4, 0xF00D, 5, 0xCAFE, 0
   DECOMPRESSED:
   .skip 4*(3+2+4+5)
   ```

PART 2: Practical Assignment

Before you begin…

- Don't forget to install all of the software tools for this assignment. The instructions are here:

  http://courses.ece.ubc.ca/259/homework/installing.htm

- Take a moment to look over the DE1 user manual that came on the CD-ROM with your kit:

  http://courses.ece.ubc.ca/259/homework/files/altera/DE1_UserManual_v1018.pdf

- Find the little *zip-lock baggie* in your DE1 kit. You should install the 6 small white silicone feet. It stops you board from sliding around on the desk! The baggie also contains two small wires that can be used to connect your 281/282 project to GND (the black wire can be attached under a screw) and one of the GPIO0 pins (the red wire can be placed on a pin).

A. Programming the DE1/DE2

7. Using your DE1 board is a two-step process:
   a. First, the DE1 board needs to be programmed with a correct **.pof** or **.sof** file. For this step, you can use *Quartus II* to program your board using the **persistent** method using a **.pof** file. This file configures the Altera Cyclone II FPGA on the board with a particular hardware circuit. In the first homework, we configured 3 different circuits into the FPGA. For this homework, we need to configure a Nios II computer system into the FPGA. If you want to use a **.sof** file, you can skip this step (see the next step).
   b. Second, the Nios II computer system needs to be loaded with software. For this step, you can use the *Altera Monitor Program* (AMP) to assemble, download, and execute your program on your Nios II computer system. The AMP will want to read a **.ptf** file, which gives it important details about the Nios II computer system (devices, memory addresses, etc). The AMP can also configure your DE1 board using a **.sof** file (containing the Nios II computer system) before it attempts to send any software.
8. At first, you can try using the preconfigured DE1 Media Computer (comes with the Altera Monitor Program). To use the COUNTER for precise hardware timing, we have made a few changes to this Nios II computer system, and provide a new set of **.sof/.pof/.ptf** files here:

   http://courses.ece.ubc.ca/259/homework/files/ubc-de1media.zip (zip for download)
   http://courses.ece.ubc.ca/259/homework/files/ubc-de1media/ (unzipped version)

B. Starting the Altera Monitor Program

9. Using the *Altera Monitor Program* (AMP), assemble, download, and execute programs on your board. Before AMP can communicate with the board, it needs to know about the details of the Nios II computer system on your DE1 board, so you must first load a **.ptf** file. This file describes all of the I/O, memory locations, and processor features that were built into the computer. The AMP needs to know this so it can properly assemble and download your program into the DE1.

   a. Start the AMP software: **Start → Programs → Altera → University Program → Altera Monitor Program → Altera Monitor Program.**
   b. Start a new Project in the AMP software. Choose **File → New Project… .** Choose a directory (where you will store all of your files) and project name.
   c. Configure the AMP for the UBC DE1 Media Computer. Under "Select a system" choose **<Custom System>** and then manually **Browse…** to find the downloaded files **ubc-de1media-v1.sof** and **ubc-de1media-v1.ptf**. Press **Next** >.
   d. Configure the AMP for the type software you want to use. Under "Program Type" choose **Assembly Program**. Press **Next** >.
   e. Configure the AMP for your software program. Press **Add…** and select an assembly program. For example, go into "hw2 examples", select "L13_demo.s", press **Select.** Notice that you can add multiple files here, and even reorder them. Press **Next** >.
   f. Under "Host connection" make sure "USB-Blaster [USB-0]" is selected. If not, connect your DE1 board to the computer, turn it on, and press **Refresh**. Press **Next** >.
   g. Under "Memory options", you can choose to relocate the ".text" and ".data" sections of your program to different addresses. To use default addresses, just leave them unchanged. Press **Finish**.
   h. AMP will now ask you if you want to send the **.sof** file to the board. Press **Yes**. It will take 1-3s to configure your board and respond with a Success dialog. Press **OK**.
   i. In the lower-right corner of the AMP window, look in the pane entitled "Info & Errors". In this window, you will see a list of messages from AMP, and any possible warnings or errors that it may have encountered. You can resize the pane to make it bigger. Hopefully, there are no errors so far.
   j. To avoid repeating steps (b) and (i) every time, you can save your current configuration. Choose **File → Save Project**. This saves other information too, which can help you quickly restore a debugging session later.

10. The Nios II program still has to be "assembled" and downloaded to your DE1 board:
    a. Compile by choosing **Actions → Compile**. You will see several messages appear in the "Info & Errors" section of the window. Since you do not need the "Terminal" section of the window for a while, you can close it (click the X). You can also resize the Info&Errors section (make it taller). The last message should be something like "**SREC generated at C:\259\hw2\L13_demo.srec.**" Of course, the exact pathname after the C:\ will be different on your computer.
    b. Download by choosing **Actions → Load**. The AMP should bring up an assembly-language listing of your program. The first instruction to be executed is shown highlighted with a pale yellow bar.
11. Single-step the program one instruction at a time by pressing F2 on the keyboard (or selecting **Actions → Single Step**). As each instruction executes, you can see the affect it has on the registers in the right-hand window named **Registers**.

12. Eventually, the HEX display will change and display the letter "E". If you keep single-stepping, you will get to a long delay loop. Instead of stepping through each item in the loop, we'll set a *breakpoint* for the first instruction after the loop.

    a. Scroll down the program listing to find the "ret" instruction in the delay loop. This should be at address 0x00000054. Set a "breakpoint" at this instruction by clicking in the vertical gray bar to the left of the "0x…" in the address. If successful, a red dot will appear there. A breakpoint tells a running CPU to stop *just before* executing this instruction. Now, press F3 or **Actions → Continue**. The CPU will stop and register r2, which was being decremented inside the loop, will contain 0. Clear the breakpoint by clicking on the red dot again.

    b. If you single-step past the first breakpoint, the HEX display will show "EE" and you will end up at the delay loop again.

    c. Press "F3" to continue running (without a breakpoint). The CPU will scroll a message across the HEX display.

13. For more information on the Altera Monitor Program features, select **Help → Tutorial**. This will open the PDF-based tutorial. Sections 3 to 4 cover the basics. Section 11.3 describes watchpoints, an advanced debugging feature that may help you keep track of values stored in specific registers or specific memory locations.

C. Playing Around (this can be addictive!)

14. Try the different example programs in the "**hw2 examples**" folder. These are all pretty simple, but they give you the general idea about what is possible. For example, example0.s copies the switch settings to the RED and GREEN LEDs (also to the HEX display!). This shows you how to do both input and output. You can probably come up with more complex goals and write your own tiny programs. As a simple practice example, try writing an "LED chaser" program ala Knight Rider. This should flash the LEDs in sequence from left to right (about ¼ second each); when it reaches the right edge (eg, LEDR0) it should bounce back and go to the left (LEDR9). To do this, you can use simple shift instructions and a delay loop.

D. Getting Serious (do this for your PA2 marks!)

15. Write a Nios II program and demonstrate it on your DE1/DE2 board. The program is a ***reaction timer***: after you see the red LEDs turn on, you should turn switch SW0 on as quickly as possible. Your reaction time will be displayed on the 7-segment display.

    The precise specifications are:

    a. If *any* of the switches is in the "1" or "up" position, keep waiting.

    b. After all switches are "0", turn off all red LEDs.

    c. Wait about 3 seconds – if SW0 goes to "1" too early, go back to step (a).

    d. Turn on all red LEDs and start measuring time.

    e. When SW0 goes to "1", stop the timer.

    f. Display how many *epochs* have elapsed using the red LEDs. We will define 1 epoch as 65536 clock cycles, which is approximately 1.31ms. To display this, bits 25 to 16 of the elapsed clock cycles should appear on LEDR9 to LEDR0. By ignoring bits 15 to 0, you are automatically dividing by 65536 (and discarding any fractional part).

    g. Display the number of elapsed *milliseconds* as a 4-digit decimal number on the 7-segment display. You will have to do some calculations for this – and it may not be as easy as it seems at first! This calculation should be as accurate as possible, so the value displayed will be roughly 31% larger than the number of epochs.

    h. Go back to step (a) which waits for the switches to be returned to 0.

    i. OPTIONAL: Update the 7-segment display while waiting for SW0 to go to "1".

    j. OPTIONAL: Beat a buddy! Determine whether SW9 or SW0 goes to "1" first. Use the green LEDs to indicate the winner. Display the reaction time of the winner on the 7-segment display, but show *how much faster* the winner is on the red LEDs.

    k. OPTIONAL: Modify the program to use a blue pushbutton, KEY3, instead of SW0. Because the buttons are spring-loaded, you can try changing the program to measure the duration between *two* successive button presses. How quickly can you "double-click" KEY3?

WARNING: The program specification above sounds very simple. However, it will likely take you a long time to get it working exactly right. You will want to get started as soon as possible!

HINTS: You will probably want to write this in several stages, expanding. Here is one method:

a. Write a short subroutine to delay about 3 seconds. Test it by writing a main program that blinks the red LEDs every 3 seconds – use a stopwatch to verify the delay.

b. Change the main program to perform (a) through (e) and (h), but instead of displaying elapsed time simply turn off all the LEDs.

c. Figure out how to measure time, then fix steps (d) and (e) and add step (f).

d. Display the # of epochs on the 7-segment display in hex. For this, write a subroutine to convert the # of epochs into a 32-bit value that you can write to HEX7SEG. Verify the value on the 7-segment display matches the red LEDs.

e. Modify the subroutine so it displays in milliseconds in decimal, completing step (g).

You will also want to test out several small programs to test each of your ideas out. Do not be afraid to experiment – but remember, the easiest way is to learn from small examples!

E. After Marking… (MANDATORY!)

16. **Immediately** after you have been marked by the TA, **you must electronically submit your program** using the Linux *handin* facility, described below.

    <u>Note: You can only hand in your assignment once!!! Read carefully!!</u>

17. Add the comment below to the top of your program. It must appear ***exactly as it is shown,*** except for the name/number/userid/email content. We will use automated tools to extract your ID information. If we cannot extract these details automatically, you will get 0 on your homework!

```
/* EECE259 Homework 2 Solution
 * Date: February 10/11, 2011
 *
 * Student1Name: John Smith
 * Student1Number: 12345678
 * Student1UserID: userid_john@ece.ubc.ca
 * Student1Email: john_smith@hotmail.com
 *
 * Student2Name: Jane C.Y. Doe
 * Student2Number: 87654321
 * Student2UserID: userid_jane@ece.ubc.ca
 * Student2Email: jane_doe@interchange.ubc.ca
 */
```

    If you do not have a partner, you should remove the four lines for "Student2".

18. The steps to hand in your program are:

    a. In MCLD348/358, open up your personal Z:\ drive. Name your solution ***<u>program.s</u>*** and copy it into a folder exactly as follows: ***<u>Z:\eece259\hw2\program.s</u>***

    b. From the Desktop Software icon, start **SSH Secure Shell Client** and press the **Quick Connect** button. Enter *ssh* for the "Host Name" and your ECE *userid*. Say "Yes" to allow **Host Identification**, then enter your password.

    c. At the "ssh-linux2:~>" prompt, type in:

       handin eece259 hw2

       exit

    d. You can only hand in once! (However, if the handin failed, you can try again.)

    e. After handin, you can delete the file/folders you created in your *Z:\* drive.

19. Hand in your program *<u>immediately</u>* after marking, or you will be assigned a grade of 0!