# **Microprocessors (662-133) NAME: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Lab #2 – Math and Logic Operation

Description

This lab will introduce the student to the registers and the types and functions of the instruction set of the Nios II processor. The student will download a simple assembly language program and modify it to calculate a mathematical expression and perform basic logical operations.

Learning objectives

1. Create the proper file structure needed by Eclipse.
2. Download the Nios II soft processor into the FPGA.
3. Download an assembly language program into the processor.
4. Use the Nios II debugger to edit and view the program.
5. Use the stepping functions to debug a program.
6. Insert a “break” point to help debug a program.
7. Use the “suspend” function to debug a program.

Procedure

Logon to the computer.

1. If you have not already created a folder to hold your projects (usually named 'workspace')do so now, otherwise you may use the same folder. This may be on a USB flash drive for convenience. If you make it on your system's hard drive, remember:
2. Do not use any folders that have spaces in the names, as some of the Altera tools have issues with this. In particular, Program Files, or My Documents
   1. Use the .sopcinfo file that should already be in this folder.
3. Start the NIOS II EDS Software Build Tools (This is a modified version of Eclipse)
4. When EDS starts, it will ask you to choose a workspace. Select the folder you created in step 1. If you like, you can set this as the default, otherwise EDS will ask on every start.
5. Create a new Project by going to File->New-> Nios II Application and BSP from Template.
6. Set the SOPC Information File to the one located in your workspace folder. The CPU name will come up automatically to 'cpu'.
7. Now choose a name for the project. In this case use 'lab2'
8. The Default Location will place all your projects in a subfolder called 'Software'. If you don't wish to have all your labs in the same place, uncheck “Use default location” and change it to simply X:\workspace\lab2
9. For a project template, chose 'Blank Project'
10. Click Next
11. Leave all the settings on this page as default, and click Finish
12. The project will build and after a few seconds, you will see 'lab2' and 'lab2\_bsp' in the Project Explorer. If you see any errors pop up during this step, most likely it is because you used a folder name with a space in it, or you don't have write permissions to the folder.
13. Let’s start building the program:

Right-click on the folder 'lab2' in the Project Explorer and select New->Source File

Set Source file: to 'lab2.s' and change the template to <None>

Click Finish

A window may appear, indicating that the makefiles are being updated, you can check the option to 'do this in the background' from now on, before it closes. A new blank file is created, so paste the contents of the included 'lab2.s' into it.

Now you will make two modifications:

* Add code to take the values in r2 through r10, using the add, sub, mul, and div instructions to calculate the expression provided in the comments, storing the result in r15. Use only the registers r11 through r15, so you may have to re-use some registers. Remember that a register can appear as both the output/result location and as one of the parameters. So for example, mul r11,r11,r12 multiplies r11 by r12 and stores the result back into r11. This will erase the original value of r11. Remember not to overwrite values you still need.
* Add code inside the loop after reading the switches to do the following:

\* Make sure the first light (in binary position 0) is always lit, regardless of the switch positions.

\* Make sure the second light (in binary position 1) is never lit regardless of the switch positions.

\* Make sure the third light (in binary position 2) always displays the opposite of the switch position

You will use a combination of and, or, nor, and xor instructions to do this, storing the result in r20 so that it will be displayed on the lights.

Add a break point on the first instruction you wrote for calculating the math formula.

1. Save the file. Then go to Project → Build All. This may take a minute or two the first time. Any typos or errors you may have will be stated in the Console window, as well as highlighted in your program. Hovering your mouse cursor over the instruction will give a popup describing the error.
2. Run this program on the DE1 hardware. Make sure the board is on and the USB cable is connected.
3. Setup the debugger. Click the small downward arrow next to the debugging icon
4. Choose Debug Configurations
5. Select the Nios II Hardware option in the list on the left, then click on the New Launch Configuration icon just above. You can change the Name at the top to 'Lab2 Debug' or similar
6. On the first Project tab, change the project name to Lab2, the Project ELF file name: should be automatically filled out as something like X:\workspace\lab2\lab2.elf
7. Go to the Target Connection tab and click Refresh Connections once or twice on the far right. The USB-Blaster should show up on both lists. (You may have maximize your window to see this If the USB-blaster still doesn’t show up, close eclipse, turn your board off then on, then restart eclipse.
8. Go to the Debugger tab. You can see the option 'Stop on startup at: main' uncheck that box.
9. Click Apply to save changes, then go ahead and click Debug
10. After a few seconds, the project will be built and downloaded to the board.
11. You may see a warning about this kind of launch begin configured to open the Nios II Debug perspective when it suspends. Check the 'Remember my decision' checkbox and click YES
12. Welcome to the NIOS II debugger view.

In the top left, you will see the Debug controls. Don't worry about the information in the window just yet, but make a note of the controls on this window.

You will use these controls to step through programs to observe your instructions in action.

In the middle window you will see your code (if an extra window titled main() appeared, just close it)

1. Before you start stepping through, on the register window in the top-right, expand the Main tab to see all the registers. Note that R8 and R10 seem unusual because you are seeing the 2's complement of these numbers, and they are negative, so they look like a very large positive number. (Since the MSB is set in 2's complement negative numbers.)
2. You should see the answer (35) visible in r15 when your calculations are done, and that will be displayed on the green LED's (in binary) after the stw r15,0(r21) instruction executes.
3. You can repeatedly step through the loop, or simply press the Resume button (F8) in the debugging controls to let it run continually. Play with the switches and see if the red LED's respond appropriately. Most should simply copy the switch next to them, except the first three should follow the rules above. If that is not the case, you can press the yellow SUSPEND button in the debugging controls, then step through while examining your operations.
4. Have your instructor sign off on the lab, run the program (just press resume if you have a breakpoint, so it runs normally). It should show the correct value on the green LEDs, and the red LEDs should copy the slider switches appropriately.