Old Dominion University

ECE 346

Salem Chemlal

Spring 2016

Lab Report 1

Bradley McKee (UIN: 00975338)

Introduction:

With this being the first lab of the semester, our main objective was to grasp some sort of understanding the basics of our DE0-NAN0 board. Since this is the first lab, I had to first install all of the programs and drivers according to the tutorial (lab 1). Next I followed through the set of instructions on how to set up my board with my pc. The main objective of the lab was to figure out and familiarize myself with the NAN0 board and Quartus 15.0.

Background:

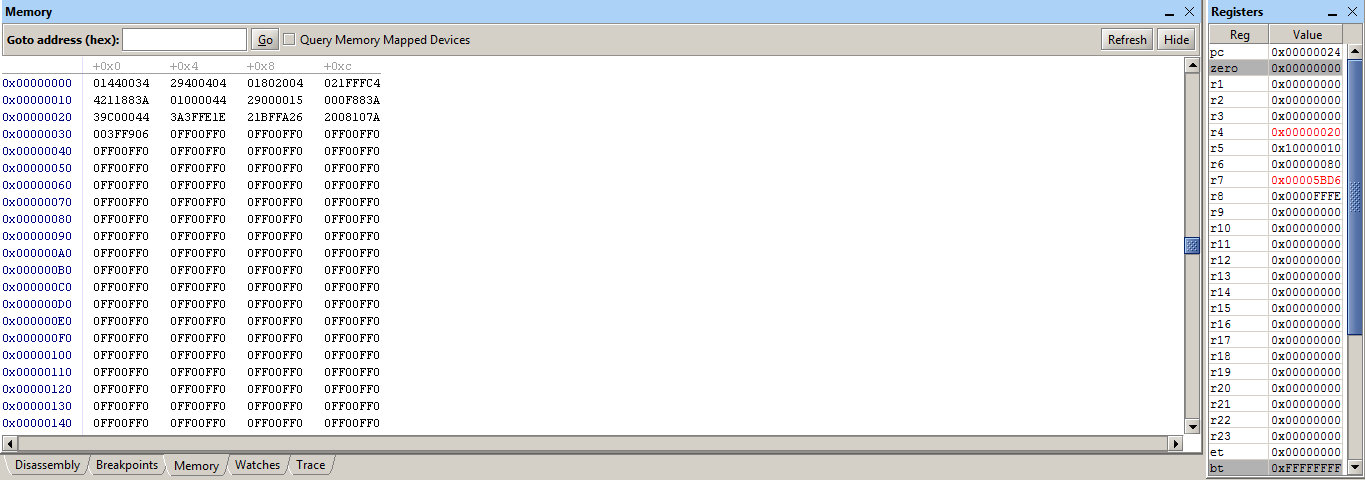
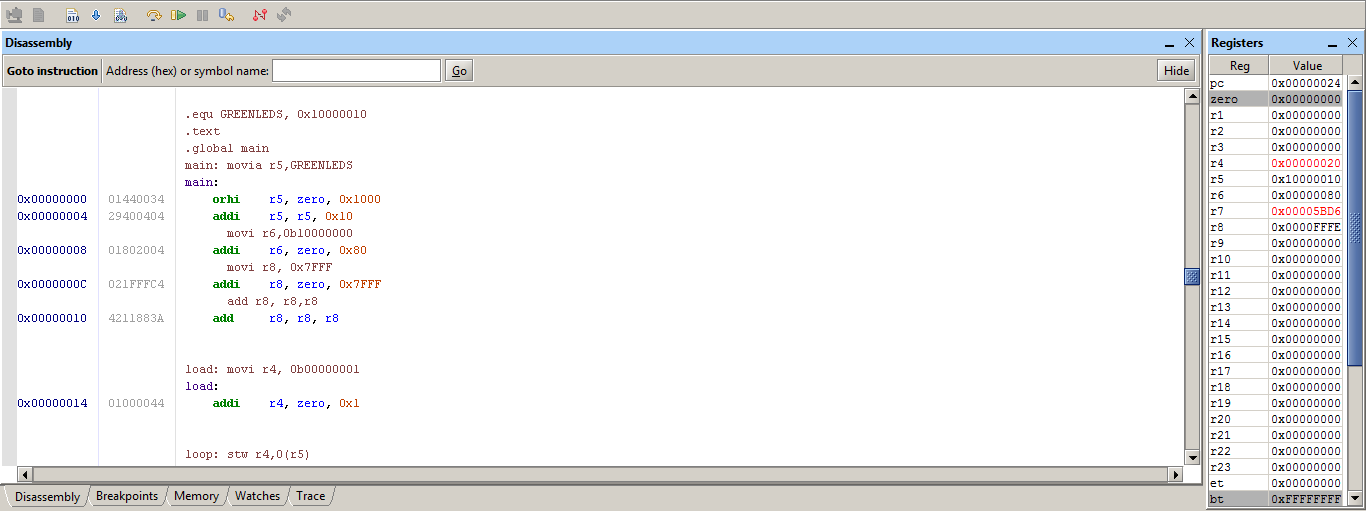
The DE0-NAN0 board is a Field Programmable Gate Array (FPGA). For our board, the processor is a Nios II Processor, which gives us a register structure to follow, as well as an instruction set. An instruction set is how you manipulate the binary code for registers.

Preliminary:

The preliminary work for this lab was to basically follow the instructions until the end and then modify the code accordingly. I had to do a lot of personal research to try and figure out the delay until a good example was done in class. I messed around a lot with the LEDS to figure out the functionality and how it works.

Results:

1)



2)

movia – Move Immediate Address (loads a 32-bit value that goes to a label. Hi/Lo order. This breaks up into orhi/ori)

movi – Move Immediate (sign entends the immediate 16 bits to 32 bit and loatds it into the new register.)

add – adds contents of two registers

stw – Store Word instruction. addresses it by an offset value and store it's contents into the new register

addi – add immediate (adds content of a register then sign-extends 16 bits. Same for signed/unsigned values.

bne – Branch if not equal (A !== B) (Boolean function)

roli – rotate left immediate. Rotates the bits of a register from right to left by the # of bits given (IMMED5) then stores it into the new register

br – samething as a jump instruction, but executes unconditionally to the address it has.. Addresses the label as well.

Functions that change (pseudo code changes):

movia – orhi/addi

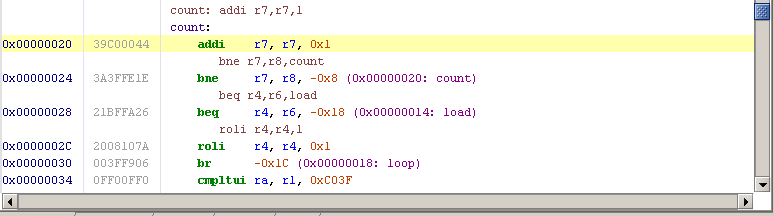
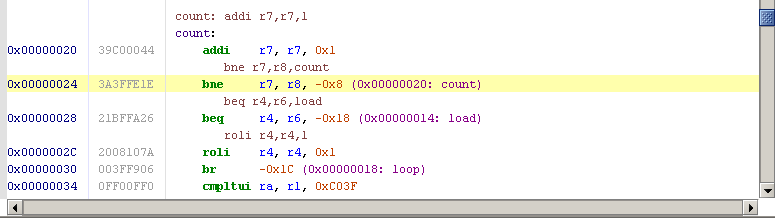
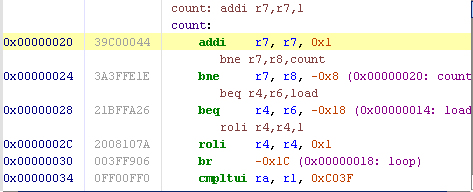
movi – addi

add r7,zero,zero = mov r7,r0

bne changed to show the address of the labels when broken down.

3)

The program gets stuck between two instructions because it is acting as a loop and has a secondary step between each instruction. Once it hits the bne (Branch Not ) it will continue to loop until r7 = r8. Once this happens it then exits the loop then loads the memory/ turn on the next LED.



4)

From my understanding the branches and jump instructions are the reasons why the LEDs cycle (LED0-LED7) The branches will make sure the values of r4 and r6 are the same before loading/cycling to the next LED. (0x0000FFFF). It affects the PC because it had to automatically allocate the space for the labels (count, load, loop). The address of PC changes according to the last used label from my observation and understanding.

For example:

0x0000020 and 0x0000024 (addi/bne for-loop)

5)

Main: stores and creates the location of LEDS. I'm pretty sure this is where we would add our delay for problem 6). R4 holds the addresses for the LEDS which gives the addresses shown below. R7 is a counter that will go until it equals R8 (which from my understanding R8 is a constant 0x7FFF)

R4 = LEDS

LED0 – 0x00000001

LED1 – 0x00000002

LED2 – 0x00000004

LED3 – 0x00000008

LED4 – 0x00000010

LED5 – 0x00000020

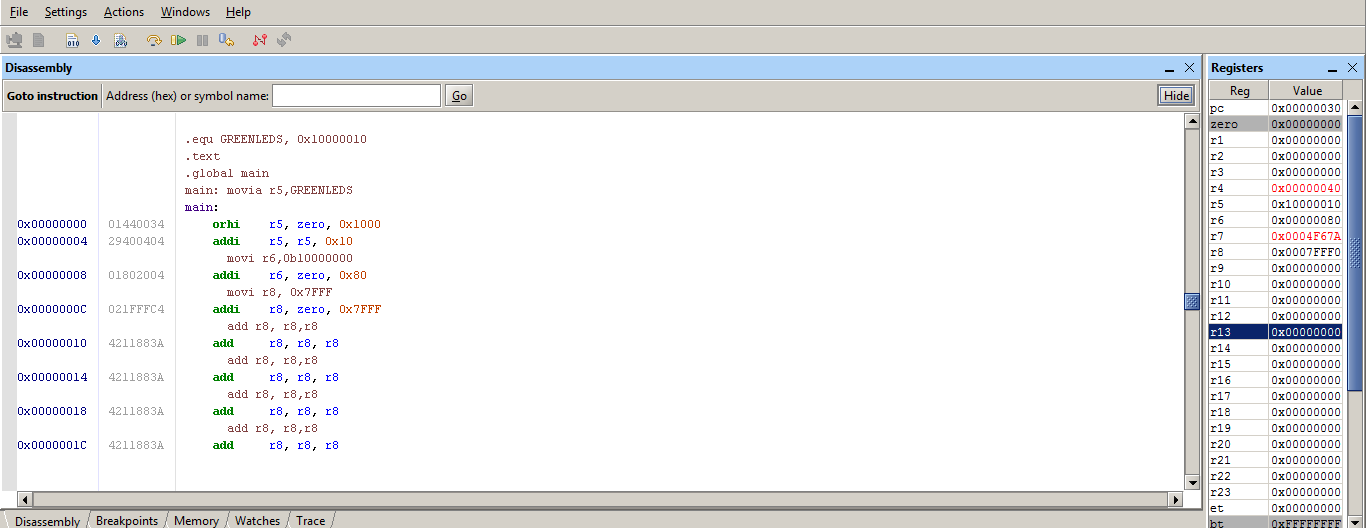
LED6 – 0x00000040

LED7 – 0x00000080

The LEDs flash when there is a bit loaded into it. The label load, loads a bit into the register r4. The label loop: makes the program go into an infinite loop until it is interrupted by the user. The label count: counts up a temporary variable r7 until it equals r8, then it load a bit into the next LED.

6)

I wanted to add a delay like we learned in class recently, it's a helpful tip especially since we know that our DE0 board runs @ 50 Mhz. Sadly, I didn't have a chance to copy it down from the board to use it for my code. I took note that r8 is what set the speed of the LEDs, I then repeated to add r8, r8, r8. The result of add r8, r8, r8 is a delay because the processor has to waste more processes therefore slowing down the LED so it is much easier to observe.



7) My program was incredibly close to functioning as asked. My LEDs lit up in the order of LED0 ,LED1, LED2, LED3 ,LED7. Sadly the flashing does stop because of the MSB rotation not really working like it’s suppose to. I tried to figure it out, but I couldn’t successfully accomplish the non-stop flashing of the LEDs. The assembly program will be attached.