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

Lab #5- Advanced branching.

Description

This lab will introduce the student to the methods of programming used with looping and branching instructions and will be explained and practiced in the lab.

Learning objectives

1. Create the proper file structure needed by Eclipse.
2. Download an assembly language program into the processor.
3. Use the Nios II debugger to edit and view the program.
4. Use the stepping functions to debug a program.
5. Insert a “break” point to help debug a program.
6. Use pseudo codes to help develop a program.
7. Use the branching instructions to make program decisions.
8. Use branching instructions to create loops.
9. Use the “for” loop to execute a loop a set number of times.

Procedure

As in the previous labs you will need to setup the project again.

Remember, to get started:

Close any projects you have open.

Create a new project and chose the name 'lab5'.

Create a new source file lab5.s, and copy the contents of the provided lab5.s

Remember to setup the debugger the first time you wish to run the program, create a new debug entry called 'lab5 dbg'.

For this program, we will be writing some code to drive a stepper motor. After building the circuit (see the end of this lab). You can simply run the existing lab5.s provided and tap any of the pushbuttons to step the motor a single step each time.

In the next two steps make the following two major modifications to this program.

Step 1) Create a loop around the step & delay code in the center of the program. (Where to create the loop is clearly indicated in the comments.) You will want to read the slider switches, initialize a count variable, and then run the step and delay procedures the number of times indicated on the slider switches. For instance, putting in the binary value 500 on the slider switches and then tapping a pushbutton will step the motor 500 times (just over 10 full rotations).

Step 2) After you have Step 1 complete and working, add code to keep track of the TOTAL number of steps performed in a memory location. You will need to add a reserved .word location in the .data section at the beginning (place it after the pattern: word) Be sure to give it a label, and initialize it to zero. Then in the program, after the “for” loop is complete, but before the 'br loop' returns to the beginning, add code to read from your reserved memory location, increment it by the number of steps (whatever the slider switches were set to) and store it back into memory.

Step 3) Challenge: Modify the program so that momentarily pressing the S0 pushbutton makes it turn clockwise and pressing the S1 button makes the motor turn counter-clockwise the predetermined amount in step 2.

You can observe this value in memory later by doing the following,

1. Pause the program execution at any time, or insert a breakpoint somewhere
2. Examine the register you loaded your count total's address into, make a note of it
3. Use the memory debugger to view this memory address.

STEPPER MOTORS

A stepper motor is a permanent magnet or variable reluctance dc motor that has the following performance characteristics:

1. Rotation in both directions,

2. Precision angular incremental changes,

3. Repetition of accurate motion or velocity profiles,

4. A holding torque at zero speed, and

5. Capability for digital control.

A stepper motor can move in accurate angular increments known as steps in response to the application of digital pulses to an electric drive circuit from a digital controller. The number and rate of the pulses control the position and speed of the motor shaft. Generally, stepper motors are manufactured with steps per revolution of 12, 24, 72, 144, 180, and 200, resulting in shaft increments of 30, 15, 5, 2.5, 2, and 1.8 degrees per step.

Stepper motors are either **bipolar**, requiring two power sources or a switchable polarity power source, or **unipolar**, requiring only one power source. They are powered by dc current sources and require digital circuitry to produce the coil energizing sequences for rotation of the motor. Feedback is not always required for control, but the use of an encoder or other position sensor can ensure accuracy when it is essential. The advantage of operating without feedback is that a closed loop control system is not required. Generally, stepper motors produce less than 1 horsepower (746W) and are therefore frequently used in low-power position control applications.

**Theory**

**Unipolar stepper motor**

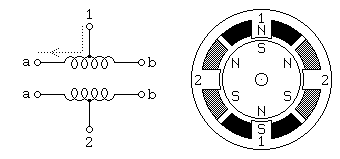
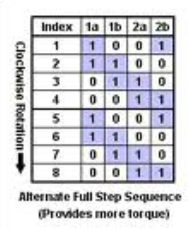


Fig.1 A unipolar stepper motor

Unipolar stepping motors with 5 or 6 wires are usually wired as shown in the schematic in Figure 1, with a center tap on each of two windings. In use, the center taps of the windings are typically wired to the positive supply, and the two ends of each winding are alternately grounded to reverse the direction of the field provided by that winding. The motor cross section shown in Figure 1 is of a 30 degree per step motor -- the difference between these two motor types is not relevant at this level of abstraction. Motor winding number 1 is distributed between the top and bottom stator pole, while motor winding number 2 is distributed between the left and right motor poles. The rotor is a permanent magnet with 6 poles, 3 south and 3 north, arranged around its circumference. For higher angular resolutions, the rotor must have proportionally more poles. The 30 degree per step motor in the figure is one of the most common permanent magnet motor designs, although 15 and 7.5 degree per step motors are widely available. As shown in the figure, the current flowing from the center tap of winding 1 to terminal a causes the top stator pole to be a north pole while the bottom stator pole is a south pole. This attracts the rotor into the position shown. If the power to winding 1 is removed and winding 2 is energized, the rotor will turn 30 degrees, or one step. To rotate the motor continuously, we just apply power to the two windings in sequence. Assuming positive logic, where a 1 means turning on the current through a motor winding.

A typical bit pattern to drive the unipolar stepper motor would be 0011, 0110, 1100, 1001, etc...) >



DRIVER CIRCUIT

We can use a simple IO port on the NIOS II along with an open-collector driver (the ULN2003AN) to drive a small unipolar stepper. By using a rotate instruction on a repeated pattern (00110011... etc..) the pattern (0011, 0110, 1100, 1001) will appear on the 4 lowest bits. This pattern will drive the stepper motor.

