**Lab 4 Morse Code**

**Joseph Tomalewski**

**Date: 10/18/2022**

**Purpose of Lab**

I must utilize AVR assembly conditionals, registers, pointers, and a combination of a C function to blink morse code on the onboard LED on the Arduino. I must also finish most of the logic and programming in assembly to compute a decoded form of morse code. This means I will be using my tools of the AVR Microcontroller and its language to convert the binary form to human readable text and numbers (ASCII).

**Description of Solution(s)**

To program the solution for the morse coder, I would have to make my code very organized and readable by using extensive comments. It is harder for me to get confused on what is going on in my own code. For my Eureka code I used I created multiple functions to make this easy to read to code. My first function would of course be to turn on the onboard LED by setting up the ports to output and sending a high value, vice versa to turn off the LED. Then I would use blink functions which in turn will use the delay and LED functions that I made to turn on the LED for a specified amount of time depending on if it should be a long or short blink which is determined inside of the delay function I created. One of the frustrating bugs I have encountered was when one of my assembler directives was slightly out of order which somehow ended up making the onboard LED stay lit and would not turn off at all unless I fixed the order.

To decode the given binary morse code value I first create a for loop in the C file to convert my binary to decimal to feed into my AVR code. The for-loop would catch when a word would finish so it would feed all the numbers to be converted and sent off to my AVR. Then my AVR would then match and branch the binary input to decode into ascii characters to be read. This part of the lab would be very difficult for me to finish on my own as I would have to use different register addressing methods and utilize X, Y, and Z registers which would take up registers R26:R31. This created another level of abstraction that would make it hard for me to test certain things exactly whenever I was progressing through my assembly coding. The longest time I spent on an error for this lab was when I noticed that val and width was not being passed from the serial to my assembly code.

**Test Results**

For this lab I would make endless tests on the decoding morse part for multiple reasons. However, for just blink Eureka in morse code was not so hard of a task and didn’t require much testing. For most of the time for the Eureka part I was just testing to make sure if the LED light would work as I intended it to for certain branches. Eventually realizing that my code was not wrong at all other than the ordering of my assembler directives being slightly wrong. Most of the testing for this part would be for making sure that my values I input were correct for the delays, etc.

For decoding morse code on the other hand it would be quite the task because it took a while for me to get the conversion properly so I could use the values in the AVR. I would mostly test each branch to see if it would be branching properly for me to realize that they would never branch since my values from the C file seemed to never to passed onto the AVR. I realized that they weren’t passing because I initialized them as integers instead of keeping them as bytes. So, the AVR would not read the correct values at all. I then had problems with what I needed to do with the table length, but the decoder is mostly functional.

**Answers to Questions**

None

**Discussion**

I think personally in my opinion that this lab was very difficult for me to do at least for the decoding part. The C file that I had to program in was very confusing and would be very helpful if the video provided talked more about the C functions that we could use and where to find them. However, the video was still very useful to me and perhaps I should have just asked more questions during the lab times I had. This lab was mostly fun but for me it was still slightly confusing coding in C.

**Contribution to Teamwork**

This was a solo lab.

**References**

For this lab I mainly used the resources provided in the lab4 assignment which includes a stack macro to push and pop registers onto and off the stack. This algorithm was very useful as it seems slightly tedious and complicated to make my own.