ECE 375 Lab 2

C $\rightarrow Assembler \rightarrow MachineCode \rightarrow TekBot$

Lab Time: Friday 2 PM - 3:50 PM

Student 1: Winnie Woo
Student 2: Joseph Borisch

TA Signature

1 Introduction

The purpose of the second lab was to learn how to read and write a C program for the ATmega32U4 microcontroller based off the BumpBot routine in lab 1. We also learned how to configure the I/O ports of the microcontroller.

2 Program Overview

This section provides an overview of the Tekbot source code used in lab 2. The BumpBot routine from lab 1 was followed, but done in C programming. In the MAIN routine, the I/O pins were initialized and the Tekbot moves forward in a default state until either or both whiskers are triggered. Within the while loop, if the right whisker and/or both whiskers are triggered the Tekbot will backup, turn left and resume forward motion. If the left whisker is triggered, the Tekbot will backup, turn right and resume forward motion.

3 Main Routine

The main routine initializes PORTB for output and to have the initial starting position as halt. PORTD was configured for whisker input. Then PORTB was set to a default forward motion.
Within the while loop, we are reading the input into the temporary register MPR and an equal-to operator (==) was used to compare the right and left sides of the operator. If for the case of the right whisker: PIND input and left whisker are the same value as the left whisker bit then it performs the set routine. Else if for the case of the left whisker: PIND input and right whisker are not the same as the left whisker bit, then it performs the set routine. If both whiskers are triggered, then it will also the same routine as if the right whisker was triggered, because the PIND input and left whisker are equal to each other.

4 Additional Questions

1. This lab required you to compile two C programs (one given as a sample, and another that you wrote) into a binary representation that allows them to run directly on your mega32 board. Explain some of the benefits of writing code in a language like C that can be "cross compiled". Also, explain some of the drawbacks of writing this way

   It is beneficial to write code that can be cross compiled because because writing code in a higher assembly language such as C allows for more powerful operations to be carried out in fewer lines of code. This means that programs can efficiently be written and compiled in less amount of time. There are also several drawbacks to cross compiling such as a higher assembly language making assumptions about the hardware specs. In addition, writing in a language like C can make it harder to understand what is actually happening and how the hardware is communicating.

2. The C program you just wrote does basically the same thing as the sample assembly program you looked at in Lab 1. What is the size (in bytes) of your Lab 1 Lab 2 output .hex files?

Can you explain why there is a size difference between these two files, even though they both perform the same BumpBot behavior?

The Lab 1 output .hex files was 485 bytes and the Lab 2 output .hex files was 909 bytes. There is a size difference because higher level code typically requires the use of libraries in order to convert the code into assembly language. These libraries typically include excess functions that are not actually used by the machine, but required in order to compile the code. Assembly language is arranged in a way that allows the computer to efficiently run the code. While assembly language looks longer and more tedious, it is typically the most optimized approach to coding.

5 Difficulties

Upon loading the program into the Tekbot, the Tekbot LED on on D8 just kept blinking no matter if the right/left/both whiskers were triggered. The problem was that PIND was not getting input, so after reworking the code and taking a look at the lab 1 skeleton code the problem was fixed.

6 Conclusion

In conclusion, this lab was meant to teach us how to convert AVR code to C code and compile it on the Tekbot. The trickiest part of the lab was understanding how the bit shift works and implementing it.

7 Source Code

```
/*
Lab 2: C- > Assembler -> Machine Code -> Tekbot
Author: Winnie Woo and Joseph Borisch
Date: 10/11/2022

This code will cause a TekBot connected to the AVR board to
move forward and when if right whisker is hit: move backwards, turn left, resume
    forward
If the left whisker is hit: move backwards, turn right, resume forward

PORT MAP
Port B, Pin 5 -> Output -> Right Motor Enable
Port B, Pin 4 -> Output -> Right Motor Direction
Port B, Pin 6 -> Output -> Left Motor Enable
Port B, Pin 7 -> Output -> Left Motor Direction
Port D, Pin 5 -> Input -> Left Whisker
Port D, Pin 4 -> Input -> Right Whisker
*/
```

```
#define F_CPU 16000000
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>

int main(void)
{
  DDRB=0b11110000; //configure PORTB pins for input/output (LED)
  PORTB=0b11110000; //set initial value for PORTB outputs as halt

  DDRD=0b00000000; //configure PORTD for input (Button)
  PORTD=0b1111111; //enable pull-up resistor
  PIND = 0b00000000;   // set initial value for PIN D inputs

  PORTB=0b10010000; // make TekBot move forward

  while (1) // loop forever
  {
    uint8_t mpr = PIND & 0b00110000;    //read the input into temporary register MPR
    //if right whisker was triggered
    //PIND input and whskrL are same value
    //PIND = 0b00001000
    if (mpr == 0b00000000 || mpr == 0b00100000)   //Check if both whiskers are hit
        or the right whisker respectively
    {
      PORTB=0b00000000;      //Move backwards
      _delay_ms(500);          // wait 500 ms
      PORTB=0b00010000;      //Turn left
      _delay_ms(1000);      //wait for 1000 ms
      PORTB = 0b10010000;   //move forward
    }
    //if left whisker was triggered
    //if PIND input and whskrR and whskrL are same value
    //PIND = 0b00001010
    else if (mpr == 0b00010000)     //check if left whisker has been hit
    {
      PORTB=0b00000000;  //Move backwards
      _delay_ms(500);   //wait 500 ms
      PORTB=0b10000000; //Turn right
      _delay_ms(2000);  //wait for 2000 ms
      PORTB = 0b10010000; //move forward

    }
  }
}
```