ECE 375 LAB 2

C -> Assembler -> Machine Code -> Tekbot

Lab Time: Tuesday 5-7

Drake Vidkjer

INTRODUCTION

The purpose of this lab it to learn how code written in C and other high level languages is translated into assembly, as well as to learn how to interface with I/O ports on the ATmega 128 dev board. First we learned how to assemble and load a C program onto the board. Next, using the assembly program from the first lab, we wrote a C program that performs the same tasks. These tasks included controlling a Tekbot, and making the Tekbot react when encountering obstacles.

PROGRAM OVERVIEW

The C program I wrote is supposed to have the same functionality as the assembly program provided in the first lab. This functionality includes controlling the two motors of a Tekbot, and responding to inputs from the two buttons on the front of the Tekbot commonly called "whiskers." The program begins by initializing ports B and D based on the port map provided. It then goes into an infinite while loop. In this while loop, the Tekbot checks to see if either, or both of the whiskers have been bumped. If so, then it moves accordingly.

INITIALIZATION ROUTINE

The initialization routine provides a one-time initialization of key registers that allow the program to execute correctly. First it initializes port B as outputs, and port D as inputs. Next it sets the pullup resistors on port D.

MAIN ROUTINE

The Main routine is an infinite loop that continuously checks to see if either or both of the whiskers have been bumped. While it does this, the program keeps the Tekbot moving forward until one of the conditions is met. If the right whisker is hit, then the HitRight routine is run, and if the left whisker is hit, the HitLeft routine is run. If both whiskers are bumped, then the HitBoth routine is run. The determination of whether one of the whiskers is bumped is made based on two bytes in Port D. If the byte is low, that means that the whisker was bumped.

HITRIGHT ROUTINE

The HitRight routine first moves the TekBot backwards for roughly 1 second by first sending the Move Backwards command to PORTB followed by a delay of 1000ms. After the delay, the Turn Left command is sent to PORTB to get the TekBot to turn left and then another delay to have the TekBot turn left for roughly another second. Finally, the program returns to the top of the loop, sending another move forward command to PORTB before checking the whiskers again.

HITLEFT ROUTINE

The HitLeft routine is identical to the HitRight routine, except that a Turn Right command is sent to PORTB instead. This then fills the requirement for the basic BumpBot behavior.

HITBOTH

The HitBoth routine does exactly the same as HitLeft, just as is asked for in the project specification.

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 mega128 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.

Being able to cross compile gives much more portability to a program. It allows the program to basically be compiled for any system of any architecture as long as there is a compiler for that language and that system. For C for example, it is possible to write a program and compile if for both 32 and 64 bit processors, instead of having to write the program in each processors respective assembly. Some of the drawbacks are that the program is not as optimized for a given system. A compiler is only so good and so the program can be slower and larger if it is written in a higher level language and then compiled.

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 Assembly program hex file was 485 bytes, while the C program generated hex file was 935 bytes. The programs don't do exactly the same things. The C program has a little more functionality, though not enough to justify the almost double size. The compiling process leads to larger file sizes, and that is why it is so much larger.

DIFFICULTIES

One difficulty I encountered was using PORTD as an input. I did not realize at first that the port would be held high until a button was pressed. This caused some difficulties when trying to test the program but was quickly resolved when the TA informed us of this fact.

CONCLUSION

The lab helped to learn how to create and load programs onto the AVR board. It also helped to learn some of the assembly language through having to reverse engineer the assembly program given to us in the first lab. We learned how to configure ports on the board, as well as see the comparison between a high level and low level programming language.

SOURCE CODE

```
/*
This code will cause a TekBot connected to the AVR board to
move forward and when it touches an obstacle, it will reverse
and turn away from the obstacle and resume forward motion.

PORT MAP
Port B, Pin 4 -> Output -> Right Motor Enable
Port B, Pin 5 -> Output -> Right Motor Direction
Port B, Pin 7 -> Output -> Left Motor Enable
Port B, Pin 6 -> Output -> Left Motor Direction
Port D, Pin 1 -> Input -> Left Whisker
Port D, Pin 0 -> Input -> Right Whisker
*/

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

```
#include <util/delay.h>
#include <stdio.h>

int main(void)
{
	/*
	DDRB = 0b11110000;	// configure Port B pins for input/output (0xF0)
	DDRD = 0b00000000;		//configure Port D pins as inputs (0x00)
	PORTB = 0b11110000;	// set initial value for Port B outputs (0xF0)
	PORTD = 0b11111111;		//set pullup resisoters (0xFF)
	*/

	//hex version
	DDRB = 0xF0;	// configure Port B pins for input/output (0xF0)
	DDRD = 0x00;		//configure Port D pins as inputs (0x00)
	PORTB = 0xF0;	// set initial value for Port B outputs (0xF0)
	PORTD = 0xFF;		//set pullup resisoters (0xFF)

	while (1) // loop forever
	{
		//move Tekbot forward
		PORTB = 0b01100000;

		//if both whiskers pressed
		if (PIND == 0b11111100)
		{
			//move back
			PORTB = 0b00000000;	// move backward
			_delay_ms(1000);	 // wait for 1000 ms

			//move left
			PORTB = 0b01000000;	// turn right
			_delay_ms(1000);	// wait for 1 s
		}

		//if left whisker pressed
		else if(PIND == 0b11111101)
		{
			//move back
			PORTB = 0b00000000;	// move backward
			_delay_ms(1000);	 // wait for 1000 ms

			//move right
			PORTB = 0b01000000;	// turn right
			_delay_ms(1000);	// wait for 1 s
		}

		//if right whisker pressed
		else if (PIND == 0b11111110)
		{
			//move back
			PORTB = 0b00000000;	// move backward
			_delay_ms(1000);	 // wait for 1000 ms

			//move left
			PORTB = 0b00100000;	// turn left
			_delay_ms(1000);	// wait for 1 s
		}
	}
}
```