ECE 375 LAB 2

C->Assembly->Machine Code->TekBot

Lab Time: Tuesday 4-6

Sonia Camacho

Owen Markley

INTRODUCTION

The purpose of this lab was to start using our ATmega128 boards and start getting familiar with the input and output ports and the pins. We were given the task to make the certain PORTs light up and move around. This was done in C code to get us more familiar with assembly code and how it is not that different than other languages that we know.

PROGRAM OVERVIEW

This lab had us using our ATmega128 boards and getting more familiar with the ports and pins. We started off by looking at the PORT map and seeing what ports were for input and which were for output. We then went into the lab-1 source code and read through the comments and made a plan of what our C code needed to do. We made a list of the steps needed which were: if the left whisker was hit then we need to go backwards and then turn right and then keep moving forward, if the right whisker was hit then backwards and then turn left. If they were both pressed then we would need to go back and then turn right and then keep moving forward. After figuring out the steps we implemented our design by checking to see if the left or right whisker was pushed and if it was then the corresponding lights would be activated followed by a motion. We did this all inside of the main function where we had 2 if statements that check if it was a left or both then execute the back and turn right portion, the other if statement checked for the right.

The lab-1 program overview which we were just putting into C code "The BumpBot program provides the basic behavior that allows the TekBot to react to whisker input. The TekBot has two forward facing buttons, or whiskers, a left and a right whisker. By default the TekBot will be moving forward until one of the whiskers are triggered. If the left whisker is hit, then the TekBot will backup and then turn right for a bit, while a right whisker hit will backup and turn left. After the either whisker routine completes, the TekBot resumes its forward motion."

INITIALIZATION ROUTINE

The initialization routine is only executed once, and initializes the key registers that allow the bot/board to perform correctly. First, the *Port x Data Direction Register* (DDR) B is initialized. Pins 4-7 are set to 1 or output since they are the only pins. From the port map we see that port D has to be able to handle input so we set it to all zeros so this represents input because the DDRX is responsible for the input and output features. We then set the pull up resistor. Next the DDRB was changed to make sure that it can handle all outputs. Then the Port B is set to move forward inside the while loop.

MAIN ROUTINE

The main routine is a basic while loop that continues on forever, moving the "bot" forwards and will adjust course if there is an object in the way of the bot. This is accomplished by setting the bits 5 and 6 from PORTB to 1 and the rest to zero. This indicates that both the left and right motors are moving forwards. The loop then continues on to check whether any of the whiskers have been hit. If one or both of the whiskers has been hit, the HitRight or HitLeft routine will run. Once this process is complete, the loop runs again placing the bot in forwards motion.

HITLEFT ROUTINE / BOTH HIT ROUTINE

The first conditional checks to see if the left whisker or both whiskers have been hit. The first two bits of the PIND register are checked for this. If these conditions are met, the sequence to turn right begins. First, the bot is stopped by setting all the bits of PORTB to 0, this should reverse the bot. A pause is then enacted to hold this state for a half second. Next, only the left motor is activated, this will turn the bot to the right. A second pause allows this to continue. Once this has been completed, the statement completes and will return to the loop, setting the bits into the pattern for forwards motion.

HitRight Routine

The HitRight routine is the same as the HitLeft routine, except that the right motor is turned on rather than the left as seen in the above function.

Wait Routine

The Wait routine is a function in the assembly code but C provides a function `_delay_ms();` inside the parenthesis you put however many milliseconds you want it to wait.

Study 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.

Some benefits to having the code be "cross compiled" makes it so that the code for one computer system can be compiled on a different system. This can be good for programming on different systems and can make the coders life easier. A disadvantage would be not knowing if the program will be read the same exact way that one system will read it. One system might also have different space and efficiency so it could really change depending on the system you run it on.

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?

Lab1 file size 1K

Lab 2 file size 1K

Our code files are the same size, this may be for the reason that we wrote our program in about the same amount of lines.

Difficulties

Some difficulties were us being overwhelmed by the idea of converting assembly code to C, which in the end was not that hard to do it just took us a minute to write out all of our thoughts and plans. We also got confused with the pins and ports for a bit but then got it all figured out.

Conclusion

Overall this lab was a great way to get us more hands on with assembly code and understanding how it works and how it is just like any other language just with more steps. We were able to get more familiar with our ATmega128 boards and understanding the way that input and output work. We were able to visually make our tekbot to work like a bumpbot and that was pretty cool. We used our Universal.GUI.ex to run the code just how we did in the first lab where the settings were all the same.

SOURCE CODE:

```
/*

This code will cause a TekBot connected to a mega128 board to 'dance' in a cool
pattern. No pins are used as input, and four Port B pins are used for output.

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
*/
#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
      PORTB = 0b11110000;     // set initial value for Port B outputs
                              // (initially, disable both motors)
      while (1) { // loop forever

            PORTB = 0b01100000;     // make TekBot move forward
            _delay_ms(500);         // wait for 500 ms
            PORTB = 0b00000000;     // move backward
            _delay_ms(500);         // wait for 500 ms
            PORTB = 0b00100000;     // turn left
            _delay_ms(1000);        // wait for 1 s
            PORTB = 0b01000000;     // turn right
            _delay_ms(2000);        // wait for 2 s
            PORTB = 0b00100000;     // turn left
            _delay_ms(1000);        // wait for 1 s
      }
}
```

Our Lab work :

```
/*
SONIA CAMACHO OWEN MARKLEY 1/14/2020
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)
{
         //from the starter code
         DDRB = 0b11110000;      // configure Port B pins for input/output
         PORTB = 0b11110000;     // set initial value for Port B outputs
         // (initially, disable both motors)

         //from the port map we see that port D has to be able to handle input
         //so we set it to all zeros so this represents input because the DDRX is responsible
for the input and output features
         DDRD = 0b00000000;
```

```
          //setting the pull up resistor
          PORTD = 0b11111111;

          //for the B making sure that it is all outputs
          DDRB = 0b11111111;

          while (1) // loop forever
          {
                    //this makes it so that PORTB is constantly moving
                    PORTB = 0b01100000;

                    //this is to see if the left has been hit
                    //and check to see if they are both being hit
                    if((PIND == 0b11111101 )||( PIND == 0b11111100)) //if it was hit this is from
the TA slides
                    {
                              //modified source code
                              PORTB = 0b00000000;     // move backward
                              _delay_ms(500);         // wait for 500 ms
                              PORTB = 0b01000000;          //turn right
                              _delay_ms(1000);        // wait for 1 s
                               PORTB = 0b00100000;         //move it forward

                    }
                    //if the right has been hit
                    if(PIND == 0b11111110)
                    {
                              //modified source code
                              PORTB = 0b00000000;     // move backward
                              _delay_ms(500);         // wait for 500 ms
                              PORTB = 0b00100000;     // turn left
                              _delay_ms(1000);        // wait for 1 s
                               PORTB = 0b00010000;         //move it forward

                    }
          }
}
```