Laboratory Assignment #2

Autumn 2013 TCES 430 – Microprocessor System Design

by

Alvin Baldemeca and Edward Bassan

Date: 10/22/2013

Table of Contents

I. Objective …………………………………………………………………………………pg 3
II. Procedure ………………………………………………………………………………...pg 3
III. Data & Analysis ………………………………………………………………………..pg 4
IV. Observation & Conclusion …………………………………………………………...pg 5
V. Appendix A - Lab2.asm Source Code ………………………………………………pg 6

I. Objective

Our objective for this lab is to write a C program using the MPLAB/MPLAB X IDE. We need to write, compile and simulate a simple program that prints “Hello World” to an output that we can see and verify. After we get this simple program to work, we will write a program to find the solution to the polynomial equation : $x^3 + 2x^2 – 10x + 40 = 0$. The requirement for this lab is that we find all the real solutions from x= -10 to x= +10 by increasing the values of x by 0.01 and looking for a change of sign (from positive to negative or vise-versa).

II. Procedure

The first thing we needed to do is to install a compiler. On one of the computer we installed the XC8 compiler from microchip. We created a new project for the MPLABX IDE, by choosing a standalone project and the device we selected is the PIC18F4520. For our hardware tools we selected the simulator, finally we selected the XC8 compiler. We then created a hello.c which prints “Hello World!”. The code below was compiled:

```
#include <stdio.h>
#include <stdlib.h>
#include <p18f4520.h>
#pragma config WDT=OFF

int main(int argc, char** argv){


   printf("Hello World!\n");
   while(1);
   return (EXIT_SUCCESS);
}
```

The compilation was successful however the problem is that we can’t see the output. We have to add some code in order for the compiler to show the output. Below is the modified code:

```
#include <stdio.h>
#include <stdlib.h>
#include <p18f4520.h>
#pragma config WDT=OFF

void putch(char c)
{
    while(!TRMT);
    TXREG = c;
}

int main(int argc, char** argv){

    SPEN = 1;
    TXEN = 1;
    printf("Hello World!\n");
    while(1);
    return (EXIT_SUCCESS);
}
```

The code added is needed for the compiler to enable the UART I/O to be simulated. We then need to change some settings for the simulator by enabling the UART I/O option and sending the output to window.

Once we got our simple “Hello World!” program to work we created a program to solve the polynomial : $x^3 + 2x^2 – 10x + 40 = 0$. Appendix A contains the code for this program.

III. Data and Analysis

Compiling the C code in the MPLABX IDE using the XC8 wasn’t difficult, however debugging the program so that we can observe if the program was written correctly gave us difficulties. Although we can debug the program we can only watch what is happening in the registers and/or memory. This is very taxing to find which register is associated with the given variable and then have to convert the machine code to a data type so we can compare what we expect. On the other hand the MPLAB IDE was not that bad at all except for the major problem that it wont support double, float, long, etc, which is quite disappointed. The bottom line is, we finally managed to run and compile the code both on either platfrom successfully and debug it as well. The MPLAB IDE simulator was quite friendly because we do not have to add any code for the compiler to display the output.

We were finally able to link the simulator to the UART output so that we can display the output to the MPLABX IDE window. This proved to be easy now but we had to go to read the manual for the compiler, search the web, go to different discussion forum and finally asked our fellow students in order for us to see the output of a printf statement.

To solve for our polynomial problem we use a brute force approach, since this was what the lab instructions asked for. The way we test if the value of ***x*** is a root was to observe the sign change for the current value of ***x*** and the next value we tested which is ***x*** + 0.01. We substituted the value of ***x*** starting from -10.0 to +10.0 incrementing ***x*** by 0.01. Below is the pseudo code we used to test where the zero crossing occurs:

Pseudo Code:

```
if [(x < 0) and (x + 0.01 > 0) ] or [(x < 0) and (x + 0.01 > 0) ]
then the root is between x and x + 0.01
```

C code used:

```
((myX < 0 && myNextX > 0) ||( myX > 0 && myNextX < 0) ? 1 : 0)
```

The program takes quite a while to finish executing. The following are the a snippet of the last three lines of output for the program :

```
x = 9.981936  and y is = 1134.062500
x = 9.991936  and y is = 1137.375000
The solution/s are x = -5.310424,
```

One thing to note is that since we are working with floats there are roundoff errors or precision errors for example observe the first lines of program output:

```
x = -10.000000  and y is = -660.000000
x = -9.989984  and y is = -657.515624
x = -9.979984  and y is = -655.000000
x = -9.969968  and y is = -652.515624
```

It is evident that from the second line -9.9 is not equal to -9.989984 although it is very close, a small error is unavoidable and should be taken note of. Also for the PIC18 we can only be precises up to 10^{-6} or 0.000001 . Once we observe that the function change signs we added 0.005 since we know the answer lies between ***x*** and ***x*** + 0.01 we take the middle value which is ***x*** + 0.005. The solution we got for the equation $x^3 + 2x^2 - 10x + 40 = 0$ is : ***x*** = -5.310424 with an error of $\mp$ 0.005.

IV. Conclusion

We were able write a PIC18F C program that solves the equation $x^3 + 2x^2 - 10x + 40 = 0$. The answer we got is ***x*** = -5.310424 with an error of $\mp$ 0.005. Using data type float causes some round off errors. The simulation for the output to the window took sometime to figure out but we were able to compile, simulate and debug our C program. We gained valuable experience writing, debugging and simulating a C program for the PIC micro controller. But yet I have to understand why the MPLAB IDE won't support certain data types, which can result to constraints that can not allow us to solve mathematical problems to precision.

Appendix A Code for Lab2.c

```
/*
 * File:   Lab2.c
 * Author: Alvin Baldemeca, Edward Bassan
 * UWT TCES 430
 * Lab 2
 * Prof. Sheng
 * Created on October 15, 2013, 10:35 AM
 */
#include <p18f4520.h>
#pragma config WDT=OFF
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/**
 * This is a helper function for main() which determines if the function
 * x^3 + 2x^2 - 10x + 40 = 0 crosses the y-axis.
 * @param thX the value to substitue x in the equation
 * @return returns 1 if x is a root of the function 0 otherwise.
 */
int findx(float thX)
{
   float myX = 0.00;
   float myNextX;
   float theX = thX;
   myX = pow(theX,3) + 2*pow(theX,2) - 10*theX + 40;
   theX = theX + 0.01;
   myNextX = pow(theX,3) + 2*pow(theX,2) - 10*theX + 40;
   printf(" and y is = %f\n", myX);
   return ((myX < 0 && myNextX > 0) ||( myX > 0 && myNextX < 0) ? 1 : 0);
}

/**
 * This functions enables the simulator to output the UART to a window for the
 * simulator on MPLABX IDE
 * @param c the character to print.
 */
void putch(char c)
{
   while(!TRMT);
   TXREG = c;
```

```
}

/**
 * This solves for the equation x^3 + 2x^2 - 10x + 40 = 0.  It prints out the
 * value of x and y tested and any real solutions found.
 * @param argc number of command line arguments(not used)
 * @param argv the pointer to the string argument values (not used)
 * @return returns 0 on success
 */
int main(int argc, char** argv) {

    SPEN = 1;
    TXEN = 1;

    float x = -10.0;
    float roots[10];
    int y = 0;
    int i = 0;
    while(x <=10.0){
        printf("x = %f ", x);
        y = findx(x);
        if(y)
        {
            x = x + 0.005;
            roots[i] = x;
            i++;
            printf("Solution x = %f\n", x);
            //The solution when this was ran is x = -5.310424
        }
        x = x + 0.01;
    }

    printf("The solution/s are x = ");
    int j = 0;
    if(roots[0]!= NULL){
        for( j=0; j < i; j++ )
        {
            printf("%f, ", roots[j]);
        }
        printf("\n");
    }else{
        printf("No roots found\n");
    }
```

```
   //While loop is needed when running the simulator otherwise the program
   // starts over and executs.
   while(1);

    return (EXIT_SUCCESS);
}
```