

CPE 325: Intro to Embedded Computer System

Lab02

Second Lab Assignment

Submitted by: Michael Agnew

Date of Experiment: 1/22/2024

Report Deadline: 1/24/2024

Demonstration Deadline: 1/30/2024

Introduction:

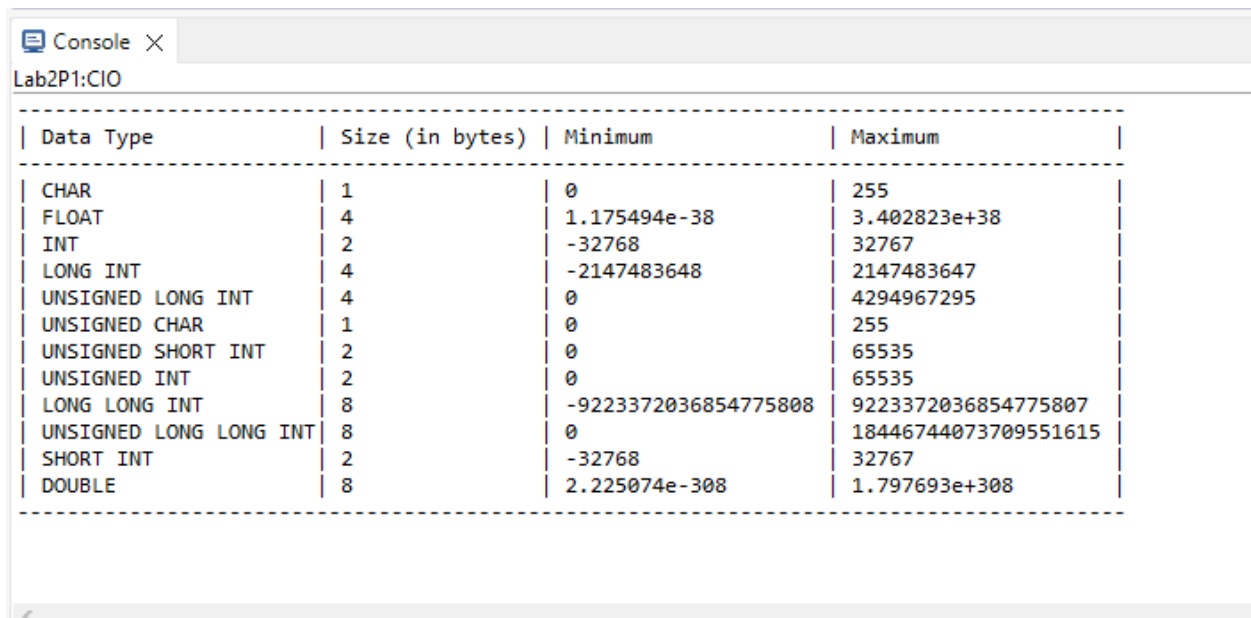
The aim of this lab was to better understand the different data types and different number base systems that exist especially in terms of programming languages. In this case, the goal was to find the size and the range of different data types in C. This is what the first two parts of the lab assignment were, the third part had to do with arrays. With these arrays, the sum and dot product of the given arrays had to be found. The last part of this lab was the bonus assignment which consisted of finding the product of two 8x8 matrices multiplied together.

Theory:

The main theory of this lab is different numerical number bases that are used for representing data as well as the various different data types that are utilized in C. The tutorial lays out the three most-used number bases which are decimal (base 10), binary (base 2), and hexadecimal (base 16). These three number bases are used a lot, and when it comes to computers hexadecimal and binary are used extensively. The tutorial provided for this lab also talks briefly about how to convert numbers from one base to another, and, when seeing how it is laid out, it is evident that for ease of calculation, it can be easier to convert a number to a different base before converting it to its final desired base. It also briefly talks about how memory is managed inside of the board that is provided for this lab, the MSP430. It talks about how the MSP430's memory can be viewed as an array and that it is shown in "little endian" form, which means that the least significant byte is placed at the lowest address. Lastly, the tutorial goes over the different data types that are used in the C programming language, which is the language that will be used for this lab. The focus of the lab is to explore the size and range of the different C data types, as well as explore some operations that can be performed with arrays.

Problem 01:

The first problem that had to be solved was writing a C code that printed the size (in bytes), and range of C data types. The data types that were presented were: char, float, int, long int, unsigned long int, unsigned char, unsigned short int, unsigned int, long long int, unsigned long long int, short int, and double. There were no operations that were needed for this assignment as making use of the <float.h> and <limits.h> headers allowed the use of various variables which are set to the maximum and minimum values of these data types. The following illustrates the output and the results obtained from this program:



Data Type	Size (in bytes)	Minimum	Maximum
CHAR	1	0	255
FLOAT	4	1.175494e-38	3.402823e+38
INT	2	-32768	32767
LONG INT	4	-2147483648	2147483647
UNSIGNED LONG INT	4	0	4294967295
UNSIGNED CHAR	1	0	255
UNSIGNED SHORT INT	2	0	65535
UNSIGNED INT	2	0	65535
LONG LONG INT	8	-9223372036854775808	9223372036854775807
UNSIGNED LONG LONG INT	8	0	18446744073709551615
SHORT INT	2	-32768	32767
DOUBLE	8	2.225074e-308	1.797693e+308

Figure 1

Looking at the results, the program clearly states which data type, the size and the exact range.

After stepping through this program with the debug feature in CCS the amount of clock cycles

needed to complete this program are 2,849,852.

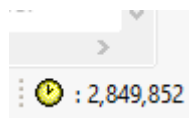


Figure 2.

Problem 02:

The second problem was to find, either by hand or by a typeset calculation, the range of a data type with a size of 2 bytes. The following figure shows the work that was done to achieve this:

a)

For a 2 byte (16 bit) signed data type, the following formula can be used for the max and min:

$$\max : 2^{n-1} - 1$$

$$\min : -(2^{n-1})$$

This is with #n being equal to the number of bits.

Therefore, for a 2 byte data type, this formula can be utilized and the following answer is found:

$$\max = 2^{16-1} - 1 = 32,767$$

$$\min = -(2^{16-1}) = -32,768$$

b)

For a 2 byte unsigned data type, the smallest value is 0. The formula used for calculating max is:

$$\max = 2^n - 1$$

This, when considering the parameters of the problem, is:

$$\max = 2^{16} - 1 = 65,535$$

c)

When considering the question of whether or not these values match the output of the written code from part one for data types with a size of 2 bytes, the answer is yes, these values exactly match the output of the code that was written.

Figure 3

The data types from Q1 that are 2 bytes are: int, unsigned int, unsigned short int, and short int.

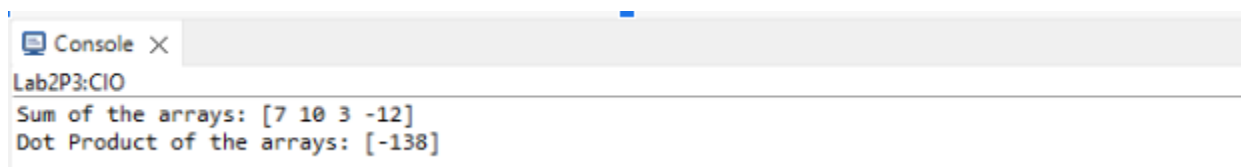
Problem 03:

The third problem that had to be solved for this assignment was one regarding operations done with arrays in C. The goals that were set out for this assignment were that the student was to set up two arrays and find the sum and dot product of the two. In order to do this, two arrays were initialized with the given values in the assignment; and a for loop was constructed that adds the current index of the two arrays based on an initialized counter, which is then added together and inserted into a “sum” array. The counter is then incremented until it reaches the end of the array. In order to obtain the dot product, the same technique was used but instead of adding, the arrays were multiplied together at the same index before adding that value to a variable for each element of the array. The assignment given illustrates this idea well:

Note: Sample Calculation for Dot Product is shown below
Array #1: (X_1 , X_2 , X_3)
Array #2: (Y_1 , Y_2 , Y_3)
Dot Product = $(X_1 * Y_1) + (X_2 * Y_2) + (X_3 * Y_3)$

Figure 4

After this was coded, the code was ran in CCS and the following output was found:



```
Console X
Lab2P3:CIO
Sum of the arrays: [7 10 3 -12]
Dot Product of the arrays: [-138]
```

Figure 5

Then, after stepping through the program the total clock cycles were found

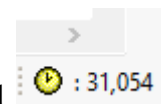


Figure 6.

This code also works for other values other than the ones given in the assignment. The arrays x and y were changed to {9, 0, 0, 4} and {3, 6, 1, -2} respectively and the results were as follows:

```

Console X
Lab2P3:CIO
Sum of the arrays: [12 6 1 2]
Dot Product of the arrays: [19]

```

Figure 7

These are the correct answers for the values chosen. A flowchart was also made for this problem. It dictates the flow of logic that the code exhibits.

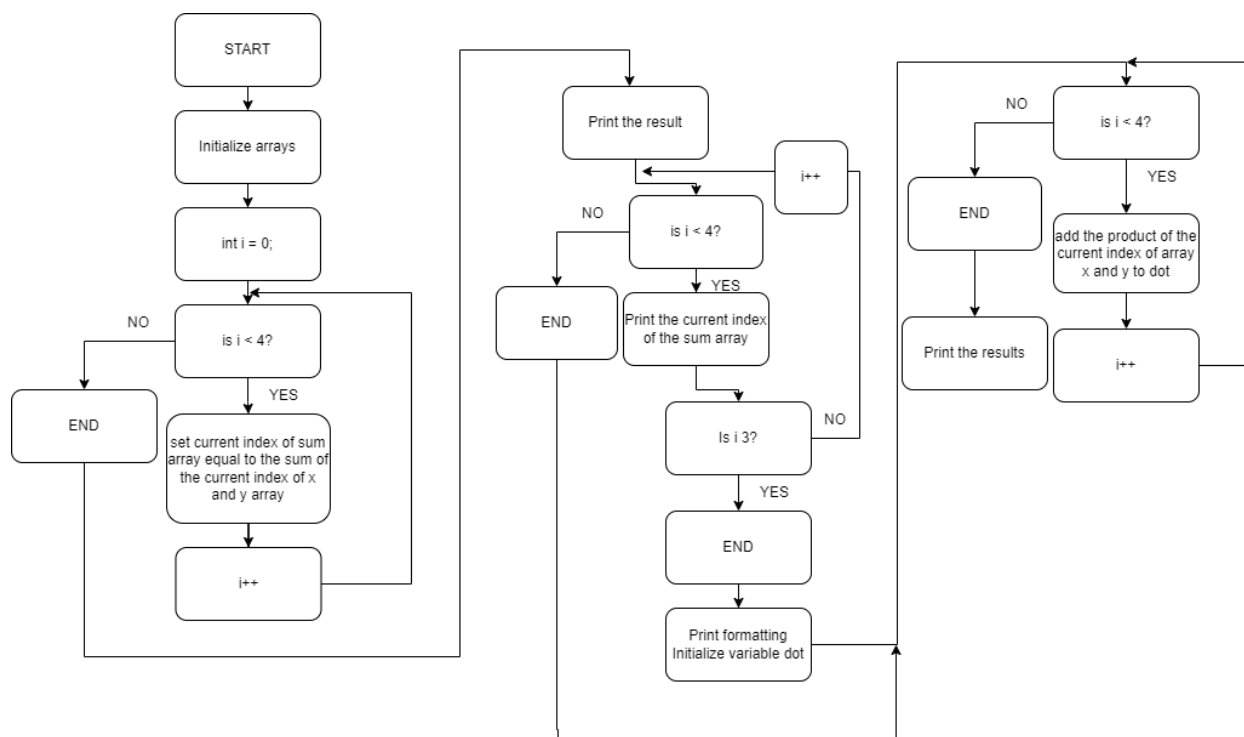


Figure 8

Bonus Problem:

The assignment included a bonus problem which consisted of taking two 8x8 matrices and multiplying them together. In order to complete this, a matrix A and a matrix B were initialized and printed to the console before being multiplied together forming a matrix C which was then also printed to the console as the final result. The output is shown as such:

```
Console X
Lab2Bonus:CIO
The matrix A is:
-----
| 1 | | 2 | | 3 | | 4 | | 1 | | 2 | | 3 | | 4 |
-----
| 1 | | 2 | | 3 | | 4 | | 1 | | 2 | | 3 | | 4 |
-----
| 1 | | 2 | | 3 | | 4 | | 1 | | 2 | | 3 | | 4 |
-----
| 1 | | 2 | | 3 | | 4 | | 1 | | 2 | | 3 | | 4 |
-----
| 1 | | 2 | | 3 | | 4 | | 1 | | 2 | | 3 | | 4 |
-----
| 1 | | 2 | | 3 | | 4 | | 1 | | 2 | | 3 | | 4 |
-----
| 1 | | 2 | | 3 | | 4 | | 1 | | 2 | | 3 | | 4 |
-----
| 1 | | 2 | | 3 | | 4 | | 1 | | 2 | | 3 | | 4 |
-----
The matrix B is:
-----
| 1 | | 2 | | 3 | | 4 | | 1 | | 2 | | 3 | | 4 |
-----
| 1 | | 2 | | 3 | | 4 | | 1 | | 2 | | 3 | | 4 |
-----
| 1 | | 2 | | 3 | | 4 | | 1 | | 2 | | 3 | | 4 |
-----
| 1 | | 2 | | 3 | | 4 | | 1 | | 2 | | 3 | | 4 |
-----
| 1 | | 2 | | 3 | | 4 | | 1 | | 2 | | 3 | | 4 |
-----
| 1 | | 2 | | 3 | | 4 | | 1 | | 2 | | 3 | | 4 |
-----
| 1 | | 2 | | 3 | | 4 | | 1 | | 2 | | 3 | | 4 |
-----
| 1 | | 2 | | 3 | | 4 | | 1 | | 2 | | 3 | | 4 |
-----
The product matrix of matrix A and B is:
-----
| 20 | | 40 | | 60 | | 80 | | 20 | | 40 | | 60 | | 80 |
-----
| 20 | | 40 | | 60 | | 80 | | 20 | | 40 | | 60 | | 80 |
-----
| 20 | | 40 | | 60 | | 80 | | 20 | | 40 | | 60 | | 80 |
-----
| 20 | | 40 | | 60 | | 80 | | 20 | | 40 | | 60 | | 80 |
-----
| 20 | | 40 | | 60 | | 80 | | 20 | | 40 | | 60 | | 80 |
-----
| 20 | | 40 | | 60 | | 80 | | 20 | | 40 | | 60 | | 80 |
-----
| 20 | | 40 | | 60 | | 80 | | 20 | | 40 | | 60 | | 80 |
-----
| 20 | | 40 | | 60 | | 80 | | 20 | | 40 | | 60 | | 80 |
-----
```

Figure 9

Conclusion:

Fortunately, no issues were encountered over the course of the lab and everything that was sought to be completed was finished without any sort of issue. All of the code that was required to be written was implemented in the correct way and was able to achieve the correct answer. What one can learn from this lab is the size and range of different data types in C, how to calculate them, and how to do basic calculations with arrays such as summation and dot product. Not only this, but through the bonus problem, one may learn how to do matrix multiplication in C, using 2d arrays. It is evident that a student who did not know of these things could learn them through this lab.

Appendix:

Problem 01:

/*-----

* File: Lab2P1Code.c

* Function: Prints the size and range of data types

* Description: Finds the size and range of various data types and outputs them to the console

* Input: None

* Output: Size and range of different data types

* Author(s): Michael Agnew, ma0133@uah.edu

* Date: Jan 23, 2024

-----/

```
#include <msp430.h>
```

```
#include <stdio.h>
```

```
#include <float.h>
```

```
#include <limits.h>
```

```
int main(void)
```

```
{
```

```
    WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer
```

```
    printf("-----\n");
```

```
    printf("| Data Type      | Size (in bytes) | Minimum      | Maximum      |\n");
```

```
    printf("-----\n");
```

```
    printf("| CHAR          | %-15d | %-20d | %-20u |\n", sizeof(char), CHAR_MIN,  
CHAR_MAX);
```

```
    printf("| FLOAT         | %-15d | %-20e | %-20e |\n", sizeof(float), FLT_MIN, FLT_MAX);
```

```
    printf("| INT           | %-15d | %-20d | %-20d |\n", sizeof(int), INT_MIN, INT_MAX);
```

```
    printf("| LONG INT      | %-15d | %-20ld | %-20ld |\n", sizeof(long int), LONG_MIN,  
LONG_MAX);
```

```
    printf("| UNSIGNED LONG INT | %-15d | %-20u | %-20lu |\n", sizeof(unsigned long int),  
0, ULONG_MAX);
```

```

printf("| UNSIGNED CHAR      | %-15d | %-20d | %-20u |\n", sizeof(unsigned char), 0,
    UCHAR_MAX);

printf("| UNSIGNED SHORT INT   | %-15d | %-20d | %-20hu |\n", sizeof(unsigned short int),
    0, USHRT_MAX);

printf("| UNSIGNED INT          | %-15d | %-20d | %-20u |\n", sizeof(unsigned int), 0,
    UINT_MAX);

printf("| LONG LONG INT        | %-15d | %-20ld | %-20ld |\n", sizeof(long long int),
    LLONG_MIN, LLONG_MAX);

printf("| UNSIGNED LONG LONG INT| %-15d | %-20u | %-20llu |\n", sizeof(unsigned long
    long int), 0, ULLONG_MAX);

printf("| SHORT INT            | %-15d | %-20d | %-20d |\n", sizeof(short int), SHRT_MIN,
    SHRT_MAX);

printf("| DOUBLE                | %-15d | %-20e | %-20e |\n", sizeof(double), DBL_MIN,
    DBL_MAX);

printf("-----\n");

return 0;
}

```

Problem 03:

```

/*-----

* File:    Lab2P3Code.c

```

* Function: Finds the dot product and sum of two arrays

* Description: Finds the dot product and sum of two arrays

* Input: None

* Output: Sum and dot product of 2 given arrays

* Author(s): Michael Agnew, ma0133@uah.edu

* Date: Jan 23, 2024

-----/

```
#include <msp430.h>
```

```
#include <stdio.h>
```

```
#include <math.h>
```

```
int main(void)
```

```
{
```

```
    WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer
```

```
    int x[4] = {5,2,-1,8};
```

```
    int y[4] = {2,8,4,-20};
```

```
    int sum[4];
```

```
    int i = 0;
```

```
    for(i = 0; i < 4; i++)
```

```
{  
    sum[i] = x[i] + y[i];  
}
```

```
printf("Sum of the arrays: [");
```

```
for(i = 0; i < 4; i++)  
{  
    printf("%d", sum[i]);  
    if (i == 3)  
    {  
        break;  
    }  
    printf(" ");  
}
```

```
printf("]");
```

```
printf("\n");
```

```
int dot = 0;
```

```
for(i = 0; i < 4; i++)  
{
```

```

        dot += x[i]*y[i];
    }

    printf("Dot Product of the arrays: [%d]", dot);

    printf("\n");

    return 0;
}

```

Bonus Problem:

```

/*-----
* File:    Lab2BonusCode.c
* Function: Multiplies Matrices together
* Description: Takes two matrices that are given, and multiplies them together before outputting
* the result
* Input:    None
* Output:   Outputs the given matrices A and B and the resulting product matrix C
* Author(s): Michael Agnew, ma0133@uah.edu
* Date:     Jan 23, 2024
*-----*/

#include <msp430.h>

#include <stdio.h>

#include <math.h>

```

```

int main(void)

{
    WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer


    const int rows = 8;

    const int cols = 8;

    int mA[8][8] = {{1,2,3,4,1,2,3,4}, {1,2,3,4,1,2,3,4}, {1,2,3,4,1,2,3,4}, {1,2,3,4,1,2,3,4},
{1,2,3,4,1,2,3,4}, {1,2,3,4,1,2,3,4}, {1,2,3,4,1,2,3,4}, {1,2,3,4,1,2,3,4}};

    int mB[8][8] = {{1,2,3,4,1,2,3,4}, {1,2,3,4,1,2,3,4}, {1,2,3,4,1,2,3,4}, {1,2,3,4,1,2,3,4},
{1,2,3,4,1,2,3,4}, {1,2,3,4,1,2,3,4}, {1,2,3,4,1,2,3,4}, {1,2,3,4,1,2,3,4}};

    int mC[8][8];


    int i = 0;

    int j = 0;

    int k = 0;


    printf("The matrix A is:\n");

    printf("-----\n");

    for (i = 0; i < rows; i++)

    {

```

```

    for (j = 0; j < cols; j++)
    {
        printf("| %d | ", mA[i][j]);

        if (j == cols - 1)
        {
            printf("\n");

            printf("-----\n");
        }
    }
}

```

```

printf("The matrix B is:\n");

printf("-----\n");

for (i = 0; i < rows; i++)
{
    for (j = 0; j < cols; j++)
    {
        printf("| %d | ", mA[i][j]);

        if (j == cols - 1)
        {
            printf("\n");

            printf("-----\n");
        }
    }
}

```

```
    }  
  }  
}
```

```
for (i = 0; i < rows; i++)  
{  
  for (j = 0; j < cols; j++)  
  {  
    mC[i][j] = 0;  
  
    for (k = 0; k < rows; k++)  
    {  
      mC[i][j] += mA[i][k]*mB[k][j];  
    }  
  
  }  
  
}
```

```
printf("The product matrix of matrix A and B is:\n");
```



```
printf("-----\n");

for (i = 0; i < rows; i++)

{

    for (j = 0; j < cols; j++)

    {

        printf("| %d | ", mC[i][j]);

        if (j == cols - 1)

        {

            printf("\n");

            printf("-----\n");

        }

    }

}

return 0;

}
```