CPE 325: Intro to Embedded Computer System

Lab02

Data Types and Average of Arrays

Submitted by: Caleb Keller

Date of Experiment:_____08/30/2022_____

Report Deadline:_____08/31/2022_____

Demonstration Deadline: _____09/01/2022_____

Introduction

Write a brief discussion on what the lab is about. (Use the tutorial and write in your own words. DO NOT copy text).

In this lab, I am creating 2 programs in CCS and performing a hand calculation. For the first program, I will be writing a program that will print out the sizes (in bytes), as well as the minimum and maximum ranges of 12 common data types. For the hand calculation, I will be finding the minimum and maximum values of a 4-byte data type. For the last part of the lab, I will be writing a program that finds the average of the corresponding elements of two arrays and printing the results as a new array.

Theory

Write short notes on each topic discussed in lab.

Topic 1: Different Data Types

a.) There are several different data types such as bool, char, short, and int. Each data type has a particular size (in bytes) in which it uses in memory.
b.) Each data type has its own characteristics that explain how and why it is used. These characteristics determine which data type is necessary to perform a certain task within a program.

Topic 2: Size limit of data types

a.) Each data type takes up a certain amount of storage and memory. Several data types require equal storage compared to other data types but are still differentiable when it comes to their ranges.

Topic 3: Endianness

a.) Endianness describes how data is stored. Data types have two kinds of significant values. To describe these values, the largest value is measured as the “Big-Endian”, whereas the least significant value is described as the “Little-Endian”.
b.) Endianness is usually specified when data is needing to be stored with a unique endianness rather than the typical data already given in a platform.

Results & Observation

Program 1:

Program Description:

Explain your approach in solving the problem.

To begin writing program 1, I first decided what exactly the program needed to do. Once I had an idea, I began to write code. I started with reviewing the ‘float.h’ and ‘limits.h’ libraries to find out how to display the sizes and ranges of the data types. Once I was familiar with the information given in

those libraries, I implemented them in code. I built the table interface using many strings. This program was quite challenging for me as an amateur C programmer, but the final product works as required!

Program Output:

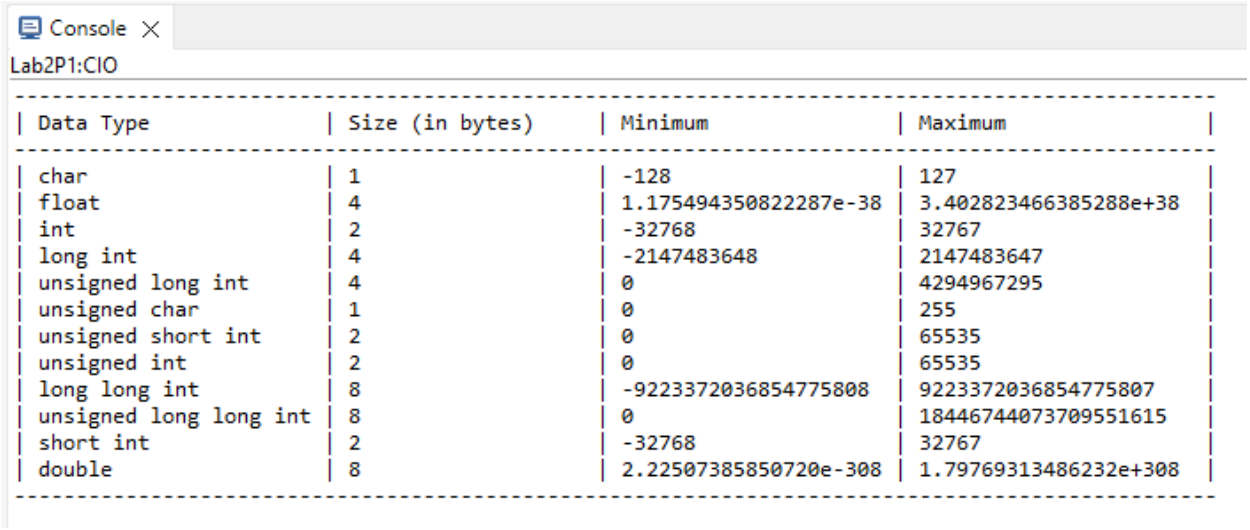

Figure #1: Program 1 Console Output

Program 2:

Program Description:

Explain your approach in solving the problem.

For program 2, I started by deciding what the program needed to do. Once I had a good understanding, I began to write code. I created 3 arrays, each with 6 elements (including positive and negative integers, one array was empty). I printed the output using 'for loops' that iterated through each array and displayed their contents to the console. For the final array I used a 'for loop' to display the averages within the empty array. To perform the average, I added each of the corresponding elements, and divided them by 2.0 (float value to allow result to be accurate).

Program Input:

```
int arrayX[6] = {-9,6,5,4,3,8};
int arrayY[6] = {-4,13,56,6,77,15};
```

Figure #2: Program 2 Input Array Values

Program Output:

```
Console
Lab2P3:CIO
Input Array x: [ -9 6 5 4 3 8 ]
Input Array y: [ -4 13 56 6 77 15 ]
Output Array z: [ -6.50 9.50 30.50 5.00 40.00 11.50 ]
```

Figure #3: Program 2 Console Output

Program Flowchart:

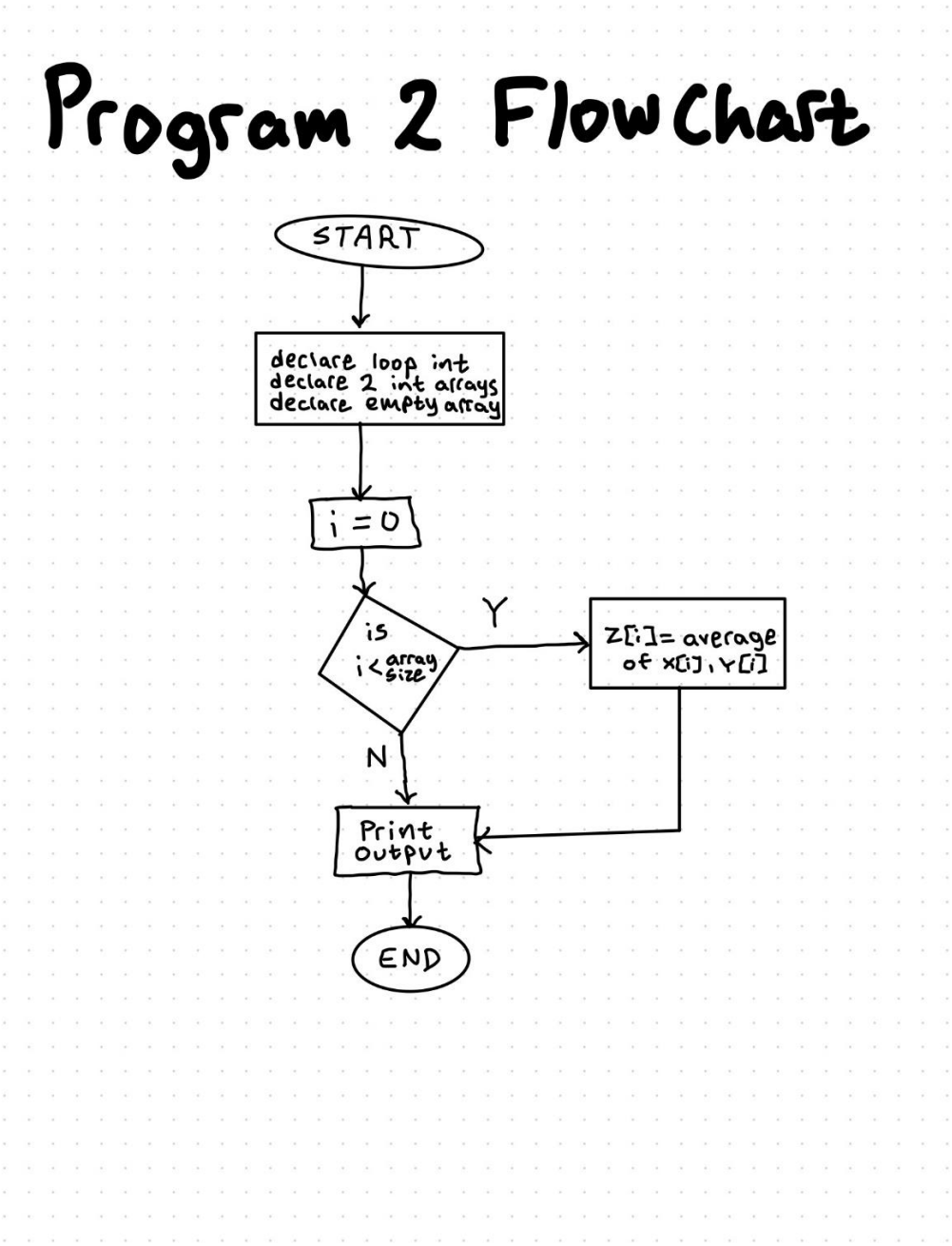

Figure #4: Program 2 Flow Chart

Conclusion

In conclusion, data types are unique in several ways. Seeing the different sizes and ranges of data types using program 1 has helped me to understand how data types work in memory, and why their characteristics are important. Program 2 demonstrated how float values differ from typical integer values. These 2 programs implemented techniques that were new to me; however, I am now more fluent in programming using the C programming language.

Part 2 Hand Calculation:

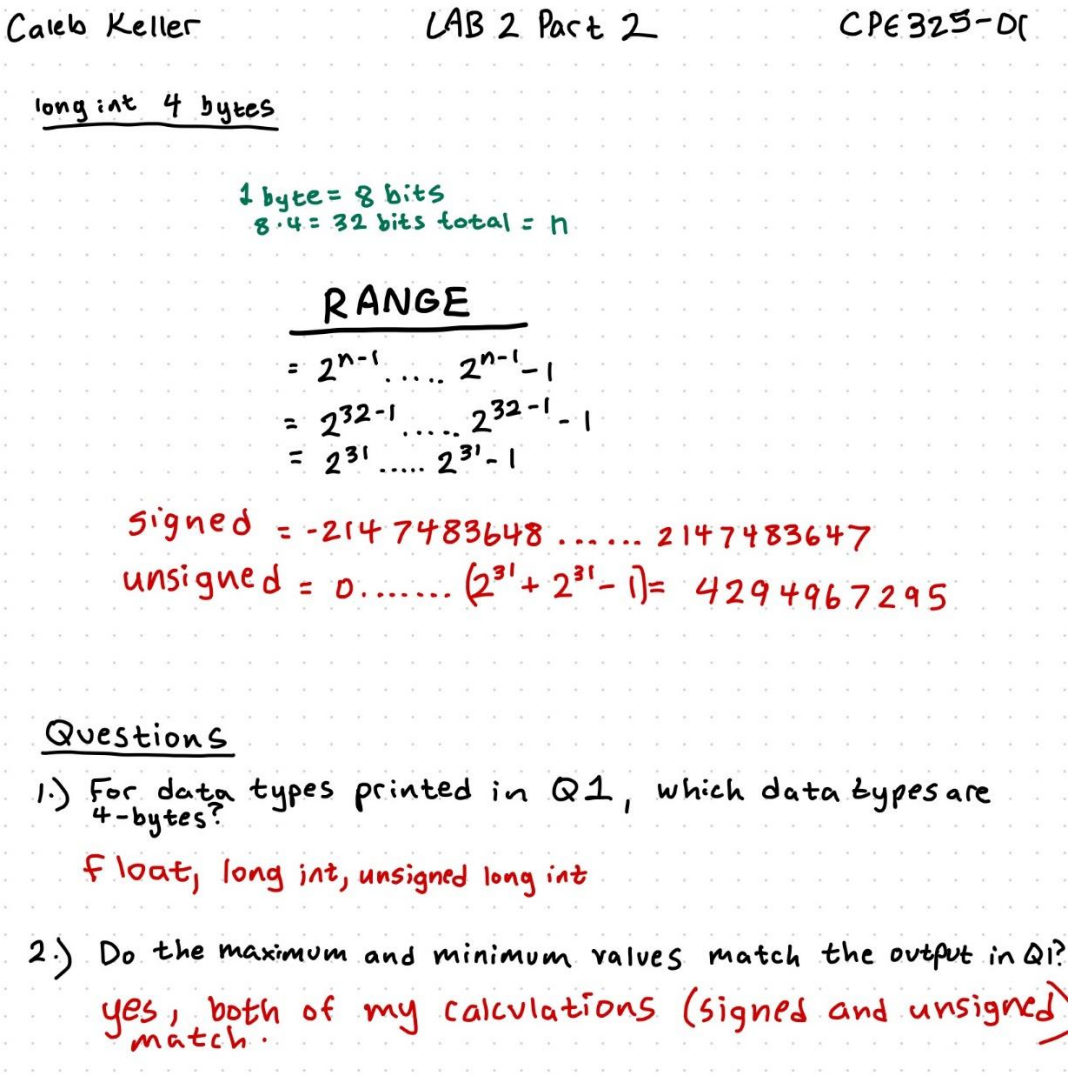

Figure #5: Part 2 Hand Calculation

Appendix

Your first code goes here, if any. Make sure you use a 1X1 table for this.

(Note: Make sure the code is readable, have comments. Also reduce spacing between lines to avoid lengthy reports.

Table #1: Program 1 source code

```
/*
 * Title: Lab 2 Part 1 Data Types
 * Author: Caleb Keller
 * Written: 08/30/2022
 * Description: Prints sizes and ranges of 12 common data types
 */

#include <msp430.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#include <float.h>

int main(void)
{
	WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer

// top of table
	char dash[] = "-----------------------------------------------------------------
-----------------------------------\n"; // dash for table
	char header[] = "| Data Type              | Size (in bytes)     | Minimum
| Maximum                |\n"; // header for table
	char newline[] = "\n"; // new line character for table format use
	char zero[] = "0 "; // for when unsigned min values = 0
	char sizespace[] = "                    | "; // table space after printing
size
	printf("%s", dash);
	printf("%s",header);
    printf("%s", dash);
// char print
	char charsize[] = "| char                    | "; // char size
	char linespace1[] = "                   | ";
	char linespace2[] = "                     |";

    printf("%s",charsize);
    printf("%i",sizeof(char));
    printf("%s", sizespace);
    printf("%1.d", SCHAR_MIN);
    printf("%s", linespace1);
    printf("%1.d", SCHAR_MAX);
```

```
    printf("%s", linespace2);
    printf("%s", newline);
// float print
       char floatsize[] = "| float                  | "; // float size
       char linespace3[] = " | ";
       char linespace4[] = "  |";
    printf("%s",floatsize);
    printf("%d",sizeof(float));
    printf("%s", sizespace);
    printf("%.15e", FLT_MIN);
    printf("%s", linespace3);
    printf("%.15e", FLT_MAX);
    printf("%s", linespace4);
    printf("%s", newline);
// int print
    char intsize[] = "| int                    | "; // size of int
    char linespace5[] = "                 | ";
    char linespace6[] = "                   | ";
    printf("%s",intsize);
    printf("%d",sizeof(int));
    printf("%s", sizespace);
    printf("%d", INT_MIN);
    printf("%s", linespace5);
    printf("%d", INT_MAX);
    printf("%s", linespace6);
    printf("%s", newline);
// long int print
    char longintsize[] = "| long int               | "; // size of long int
    char linespace7[] = "            | ";
    char linespace8[] = "              | ";
    printf("%s", longintsize);
    printf("%d",sizeof(long int));
    printf("%s", sizespace);
    printf("%ld", LONG_MIN);
    printf("%s", linespace7);
    printf("%ld", LONG_MAX);
    printf("%s", linespace8);
    printf("%s", newline);
// unsigned long int print
    char ulongintsize[] = "| unsigned long int      | "; // size of unsigned int
    char linespace9[] = "                    | ";
    char linespace10[] = "              | ";
    printf("%s", ulongintsize);
    printf("%d",sizeof(long int));
    printf("%s", sizespace);
    printf("%s", zero); // since unsigned values min = 0
    printf("%s", linespace9);
    printf("%lu", ULONG_MAX);
    printf("%s", linespace10);
    printf("%s", newline);
// unsigned char print
    char ucharsize[] = "| unsigned char          | "; // size of unsigned char
    char linespace11[] = "                    | ";
    char linespace12[] = "                    | ";
    printf("%s", ucharsize);
```

```
    printf("%d",sizeof(char));
    printf("%s", sizespace);
    printf("%s", zero); // since unsigned values min = 0
    printf("%s", linespace11);
    printf("%d", UCHAR_MAX);
    printf("%s", linespace12);
    printf("%s", newline);
// unsigned short int print
    char usintsize[] = "| unsigned short int     | "; // size of unsigned short int
    char linespace13[] = "                     | ";
    char linespace14[] = "                   | ";
    printf("%s", usintsize);
    printf("%d",sizeof(short));
    printf("%s", sizespace);
    printf("%s", zero); // since unsigned values min = 0
    printf("%s", linespace13);
    printf("%hu", USHRT_MAX);
    printf("%s", linespace14);
    printf("%s", newline);
// unsigned int print
    char uintsize[] = "| unsigned int           | "; // size of unsigned int
    char linespace15[] = "                     | ";
    char linespace16[] = "                   | ";
    printf("%s", uintsize);
    printf("%d",sizeof(int));
    printf("%s", sizespace);
    printf("%s", zero); // since unsigned values min = 0
    printf("%s", linespace15);
    printf("%u", UINT_MAX);
    printf("%s", linespace16);
    printf("%s", newline);
// long long int print
    char LLintsize[] = "| long long int          | "; // size of long long int
    char linespace17[] = "  | ";
    char linespace18[] = "    | ";
    printf("%s", LLintsize);
    printf("%d",sizeof(long long));
    printf("%s", sizespace);
    printf("%lld", LLONG_MIN);
    printf("%s", linespace17);
    printf("%lld", LLONG_MAX);
    printf("%s", linespace18);
    printf("%s", newline);
// unsigned long long int print
    char uLLintsize[] = "| unsigned long long int | "; // size of unsigned long
long int
    char linespace19[] = "                     | ";
    char linespace20[] = "   | ";
    printf("%s", uLLintsize);
    printf("%d",sizeof(long long));
    printf("%s", sizespace);
    printf("%s", zero); // since unsigned values min = 0
    printf("%s", linespace19);
    printf("%llu", ULLONG_MAX);
    printf("%s", linespace20);
```

```
    printf("%s", newline);
// short int print
    char sintsize[] = "| short int              | "; // size of short in
    char linespace21[] = "                 | ";
    char linespace22[] = "                   | ";
    printf("%s", sintsize);
    printf("%d",sizeof(short));
    printf("%s", sizespace);
    printf("%d", SHRT_MIN);
    printf("%s", linespace21);
    printf("%d", SHRT_MAX);
    printf("%s", linespace22);
    printf("%s", newline);
// double print
    char doublesize[] = "| double                 | "; // size of double
    char linespace23[] = " | ";
    char linespace24[] = "  | ";
    printf("%s", doublesize);
    printf("%d",sizeof(double));
    printf("%s", sizespace);
    printf("%.14e", DBL_MIN);
    printf("%s", linespace23);
    printf("%.14e", DBL_MAX);
    printf("%s", linespace24);
    printf("%s", newline);

// end of table
    printf("%s", dash);

      return 0;
}
```

Your next code goes here, if any.

Table #2: Program 2 source code

```
/*
 * Title: Lab 2 Part 3 Array Average Program
 * Author: Caleb Keller
 * Written: 08/31/2022
 * Description: This program finds the average of the corresponding elements of two
arrays
 */

#include <msp430.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(void)
{
      WDTCTL = WDTPW | WDTHOLD; // stop watchdog timer
```

```
	int i = 0;

	int arrayX[6] = {-9,6,5,4,3,8};
	int arrayY[6] = {-4,13,56,6,77,15};
	float arrayZ[6];

	char X[] = "Input Array x: [ "; // output for arrayX
	char Y[] = "Input Array y: [ "; // output for arrayY
	char Zout[] = "Output Array z: [ "; // output for arrayZ
	char bracket[] = "]\n"; // bracket with new line to be placed at end of each
output

//finding average of corresponding arrayX and arrayY elements
	for(i = 0; i < 6; i++)
	{
	    arrayZ[i] = (arrayX[i] + arrayY[i])/2.0; // dividing by 2.0 to get float
values
	}
//begin output
	printf("%s", X);
	for(i = 0; i < 6; i++)
	{
	    printf("%d ", arrayX[i]); // iterating through arrayX to print each
value contained in the array
	}
	printf("%s", bracket);

	printf("%s", Y);
	for(i = 0; i < 6; i++)
	{
	    printf("%d ", arrayY[i]); // iterating through arrayY to print each
value contained in the array
	}
	printf("%s", bracket);

	printf("%s", Zout);
	for(i = 0; i < 6; i++)
	{
	    printf("%0.2f ", arrayZ[i]); // printing average for each element with a
precision of 2 after decimal
	}
	printf("%s", bracket);

	return 0;
}
```

Make sure to submit a single pdf file only