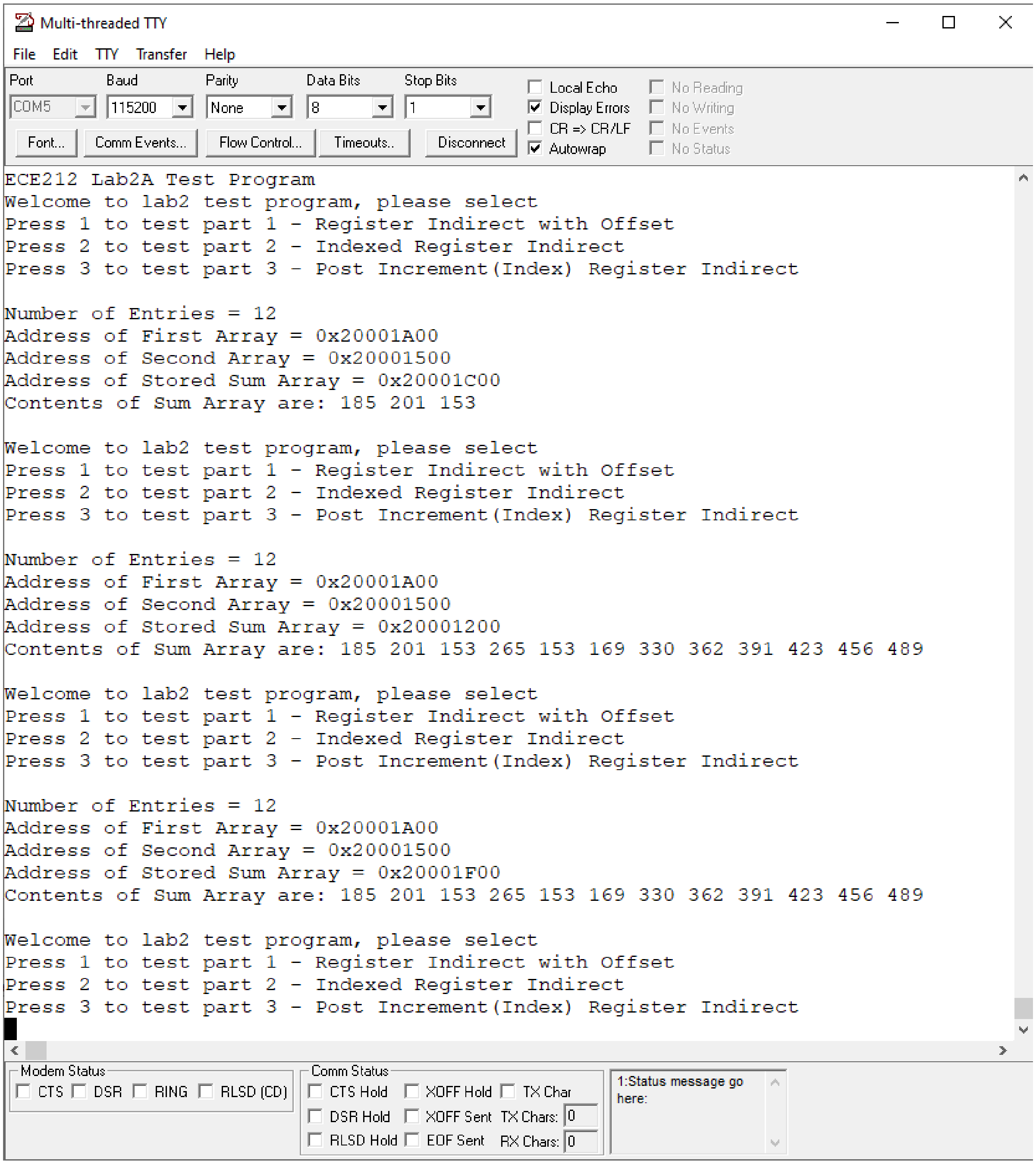
Lab 2: Introduction to Addressing Modes



Port: COM5
Baud: 115200
Parity: None
Data Bits: 8
Stop Bits: 1

□ Local Echo
☑ Display Errors
□ CR => CR/LF
☑ Autowrap
□ No Reading
□ No Writing
□ No Events
□ No Status

```
ECE212 Lab2A Test Program
Welcome to lab2 test program, please select
Press 1 to test part 1 - Register Indirect with Offset
Press 2 to test part 2 - Indexed Register Indirect
Press 3 to test part 3 - Post Increment(Index) Register Indirect

Number of Entries = 12
Address of First Array = 0x20001A00
Address of Second Array = 0x20001500
Address of Stored Sum Array = 0x20001C00
Contents of Sum Array are: 185 201 153

Welcome to lab2 test program, please select
Press 1 to test part 1 - Register Indirect with Offset
Press 2 to test part 2 - Indexed Register Indirect
Press 3 to test part 3 - Post Increment(Index) Register Indirect

Number of Entries = 12
Address of First Array = 0x20001A00
Address of Second Array = 0x20001500
Address of Stored Sum Array = 0x20001200
Contents of Sum Array are: 185 201 153 265 153 169 330 362 391 423 456 489

Welcome to lab2 test program, please select
Press 1 to test part 1 - Register Indirect with Offset
Press 2 to test part 2 - Indexed Register Indirect
Press 3 to test part 3 - Post Increment(Index) Register Indirect

Number of Entries = 12
Address of First Array = 0x20001A00
Address of Second Array = 0x20001500
Address of Stored Sum Array = 0x20001F00
Contents of Sum Array are: 185 201 153 265 153 169 330 362 391 423 456 489

Welcome to lab2 test program, please select
Press 1 to test part 1 - Register Indirect with Offset
Press 2 to test part 2 - Indexed Register Indirect
Press 3 to test part 3 - Post Increment(Index) Register Indirect
█
```

□ CTS □ DSR □ RING □ RLSD (CD)

□ CTS Hold □ XOFF Hold □ TX Char
□ DSR Hold □ XOFF Sent TX Chars: 0
□ RLSD Hold □ EOF Sent RX Chars: 0

1:Status message go here:

Lab Dates

Refer to the schedule on the ECE212 Laboratories page for the latest schedule

Introduction

In this lab the students will be introduced to a few of the different types of addressing modes in assembly language programming.

Objectives:

1. To gain experience developing code in THUMB2 assembly language
2. To gain experience in fetching/storing content in memory with different addressing modes

Prelab and Preparation:

- Read the lab prior to attending your lab session. Please only attend the section that you are registered in.
- Do the online prelab Quiz **individually** and submit it before the deadline
- Prepare rough drafts of the flow chart(s)/pseudocode for each part of the lab(Part A and Part B). You do not neet to use a CAD tool for this. The final CAD version will be submitted in your lab report.

Lab Work and Specifications

1. Download the template files
    - main_L432KC.c
    - retarget.c
    - retarget.h
    - DataStorage.s
    - SetZeros.s
    - Lab2a_L432KC.s(For part A)
    - Lab2b_L432KC.s(For part B)
2. Review the marking sheet

Specifications

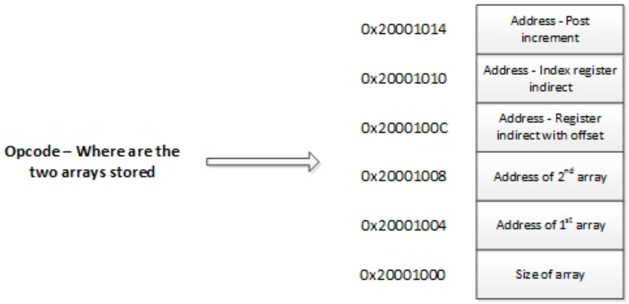
If you recall from the Lab1, the 'main_L432KC.c' is the standard project template that is used to initialize and call standard functions/subroutines including our 'AssemblyProgram'. Do not modify any of the parameters in this file. 'SetZeros.s' and 'DataStorage' are also provided to initialize the memory contents. Do not modify any of the parameters in either file. Lab2a_L432KC.s is provided for you to write your assembly language program for part A and Lab2b_L432KC.s is provided for part B. All 3 parts for Part A should be written in Lab2a_L432KC.s and part B should be in Lab2b_L432KC.s. A more detail description is provided below.

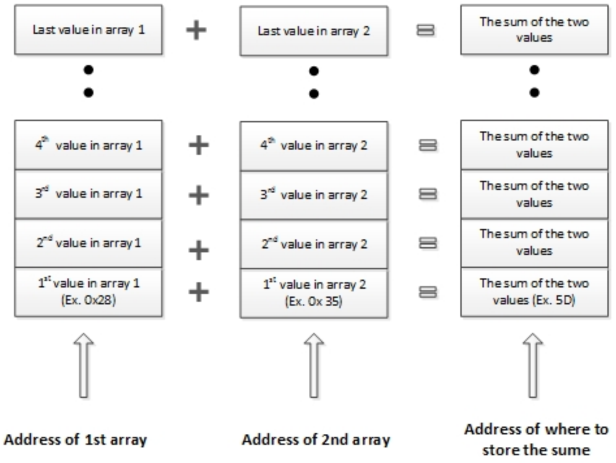
Part A - Different Addressing Modes

You are asked to write a program that will add the contents of two arrays stored at different memory locations and place the result at another specified memory location. The operational code will be provided at memory location 0x20001000. The operational code contains the information for where to access your arrays in memory and where to store the sum. Each piece of information is stored as a word(32bits). For instance the first word(32bits) stored at memory location 0x20001000 will contain the size of the two arrays. The next word(32bit) will contain the address of where the first array is stored in memory. The word(32bits) after that will contain the address of where the second array is stored in memory. A breakdown of the operational code is illustrated as follows

1. 0x20001000 - Size of array
2. 0x20001004 - address of first array
3. 0x20001008 - address of second array
4. 0x2000100C - address of where to store the sum with Register Indirect With Offset
5. 0x20001010 - address of where to store the sum with Indexed Register Indirect
6. 0x20001014 - address of where to store the sum with Postincrement Register Indirect

Note: Each piece of information is stored as a word(32bits)




Last value in array 1
+
Last value in array 2
=
The sum of the two values
4th value in array 1
4th value in array 2
The sum of the two values
3rd value in array 1
3rd value in array 2
The sum of the two values
2nd value in array 1
2nd value in array 2
The sum of the two values
1st value in array 1 (Ex. 0x28)
1st value in array 2 (Ex. 0x 35)
The sum of the two values (Ex. 5D)
Address of 1st array
Address of 2nd array
Address of where to store the sume

Part 1

In part 1, you will perform the addition of the two arrays using the Register Indirect With Offset. You need only perform the operaton on the first 3 values in the array to prove that you understand how to use this mode of operation.

Part 2

Repeat part 1 using the Indexed Register Indirect method. You need to perform the operation on the entire array. An index register must be used

Part 3

Repeat part 1 using the Postincrement Register Indirect. You need to perform the operation on the entire array.

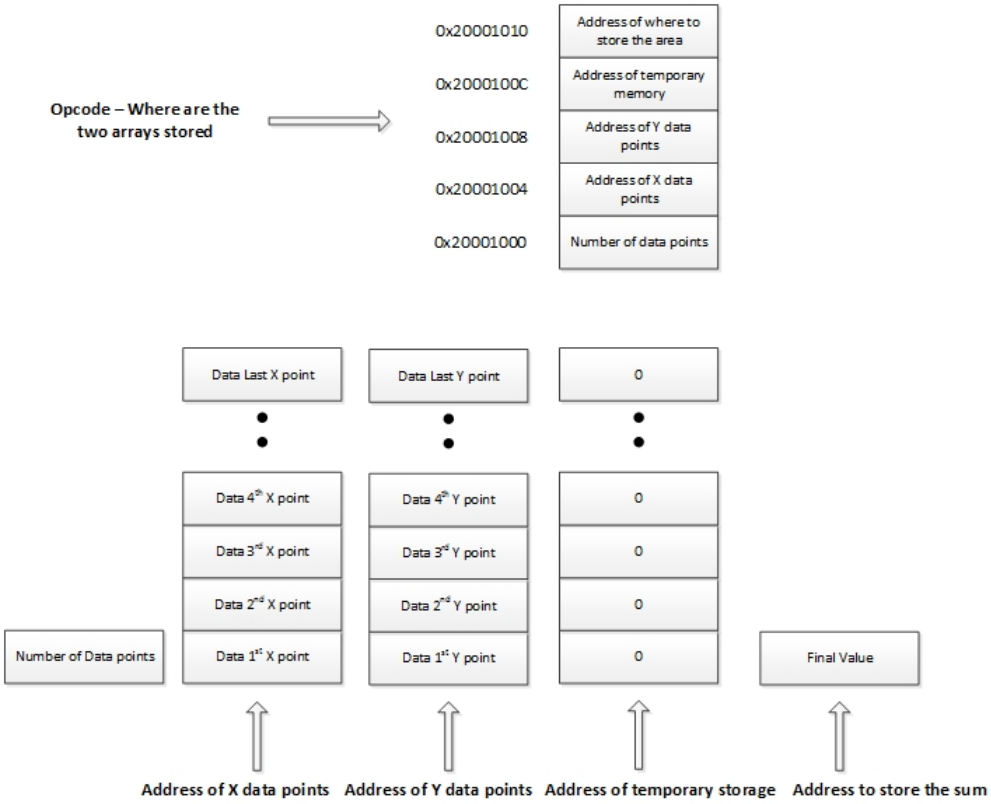
Note: Solutions for each part are written in Lab2a.s

Part B - Trapezoidal rule

You are asked to write a program that will calculate the area underneath a curve(y = f(x)) using the Trapezoidal rule that you learned in your Calculus course. The data points for the curve(X and Y data points) are stored as arrays starting at a specific place in memory. For convenience, the distance between each X data point(delta x) is either 1,2 or 4. The multiplication instruction cannot be used. Instead, think of another way to perfom the multication by 1,2 or 4. The operational code will be provided at memory location 0x20001000. The operational code contains the information such as number of data points, where to access the array of data points(X and Y) in memory, temporary storage space(if required) for use and where to store the final area value. Just like in part A, each piece of information is stored as a word(32bits). The first word(32bits) stored at memory location 0x20001000 will contain the number of data points. The next two words(32bits each) will contain the address of where the X and Y data points are stored in memory. The next word(32bits) after that contain an addresse for where you can store temporary values if you happen to run out of registers and need a place to temporary store some values. The last word(32bits) will contain the address of where to store the final area in memory. A breakdown of the operational code is illustrated as follows. For this part, please download and use only **Datastorage4/5/6.s and SetZeros4.s**. The different Datastorage files are different test cases. You should test them all to confrim functionality of your code. In order to preserve accuracy, do not round fractions during the calculation but you need to keep track of how many so you can add them back to the final value. Finally, if the final sum contains a fraction, please round up.

1. 0x20001000 - Number of Data Points
2. 0x20001004 - address of where the X data points are stored
3. 0x20001008 - address of where the Y data points are stored
4. 0x2000100C - address of temporary storage space
5. 0x20001010 - address of where to store the final value

Note: Each piece of information is stored as a word(32bits)



Questions

1. What are the advantages of using the different addressing modes covered in this lab? Indicate any restrictions or limitations.

2. Do the different Addressing Modes affect the CCR bits in the ASPR register? If yes, which bits are affected?

3. From the data points, what is the function (y=f(x))? What is the percent error between the theoretical calculated area and the one obtain in your program? Do this for all 3 different test cases.

Marking Scheme

Lab 2 is worth 25 % of the final lab mark. Please view the Marking Sheet for this lab to ensure that you have completed all of the requirements of the lab. The Marking Sheet also provides a limited test suite in the demo section for you to think about. Make use of it! When writing the report, please follow the ECE 212 Lab Report Guideline document uploaded to Canvas.

Demo and Report

The **report and demo due dates** are given on the ECE212 Laboratories page on Canvas. Note that you have one week from the dating of your prelab to complete the demo.

During your demo, you solution will be graded on functionality(did you meet the requirements). In addition, we may or may not ask you questions about your solution including specific instructions or ask you to make modificiatons to your program. **Only one demo is allowed** so make sure you thoroughly understand your code before requesting the demo. Late demo will not be accepted.

The report must be submitted using the submission link on Canvas before **4:30 P.M. Do not be late** on your scheduled due date.

Late Submissions

There are late submission penalties for the Report(-10%) per calendar day(2 days max) after the scheduled due date. If you are submitting your report late, please email your report to the lab instructor since the submission link will be closed.