

Assignment 2B (40 marks) – Lab Week Seven

Due: End of your week Eight's lab period: Week of 11 – 16 March 2018

Prelab work – It is highly advisable to complete the following PRIOR to your lab period.

- *Read the Assignment, noting that you are required to hand-in 2 sheets for this assignment.*
- *Download Math_Operations.asm from the Lab Week Seven folder.*
- *Complete Task One – paragraphs a and b (at a minimum) so that you have code that can correctly assembled*

Second Assembly Program – Assemble and Trace

This portion of the lab exercise has you create and trace a simple program using 68HCS12 Assembly Language as the target language using [AsmIDE](#) and [Simulator – Dragon12 & Student Mode](#).

PURPOSE OF LAB:

The purpose of this lab is to gain more experience with both the assembler and simulator that will be extensively used in this course. Additionally, you will have the opportunity to confirm your understanding of Arithmetic Principles taught during in-class lectures.

Resources:

The following material is available on Blackboard in the Course Resources - Textbooks and User Guides folder to assist you in answering questions contained in this lab exercise. The Help feature in [AsmIDE](#) may also be of assistance to you.



Course Text - 68HCS12Text




HCS12-9S12 Instruction Set Reference



HCS12 Assembly Language Reference Manual

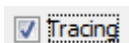
Task One – Assemble; Trace; Fill in the blanks.

- Load the source code for **Math_Operations.asm** into [AsmIDE](#). You may find that you will have to align Labels, Opcodes, Operands and Comments into their correct columns in order to correctly assemble the program. This alignment was explained in a previous lab exercise and discussed in a lecture period.
- Assemble the code, ensure that there are no errors or warnings present.
- Start [Simulator – Dragon12 & Student Mode](#) and load Math_Operations.s19
- As you  through the program, **fill in the Task One – Part One Questions on the A2B Lab Week Seven Hand-In sheet.**
- Once you have completed the above steps, prepare the simulator to rerun the program by using **File → Reset**. Now, click on

```

1 ; Math_Operations.asm
2 ;
3 ; Author:          D. Haley
4 ; Student Number:  nnn-nnn-nnn
5 ; Date:            22 Feb 2018
6 ;
7 ; Purpose:         To Gain Experience with Assembly Language instructions
8
9 ; Program Constants
10 STACK equ        $2000
11
12             org    $1000           ; Data starts at $1000
13 MyArray db      $0C,$81
14 Result ds        2
15
16             org    $2000           ; Program Code starts at $2000
17 Start  lds        #STACK           ; Setup the stack
18
19             ldaa    #$25            ; Note: This code is purposely NOT documented
20             ldab    #25
21             adda    MyArray
22             staa    Result
23             stab    Result+1
24             ldaa    Result+1
25             ldab    Result
26             incb
27             ldaa    #%10101010
28             std     Result
29             swi
30             end

```



then select **View → Console Log** and Step through the program again. What you will observe is a complete trace of your program, which is very handy in debugging Assembly Language programs!

- Now, **answer the Task One – Part Two Questions on the A2B Lab Week Seven Hand-In sheet**

Assignment 2B (40 marks) – Lab Week Seven Hand-In Sheet

Due: End of your week Eight's lab period: Week of 11 – 16 March 2018 Page 1 of 2



Name: _____

Write in your Lab Day and Time (e.g. Wed 10 - 12)

Student Number: _____

Task One – Part One Questions (18 marks)–Place your answers in the allocated space.

Using the **Simulator – Dragon12 & Student Mode**, record the **missing** values of the registers and memory locations **after** **executing each listed instruction** for **Math_Operations.asm**. Watch how each instruction changes the registers and memory locations that are contained in the table below.

As you enter the hexadecimal values into the table, ensure they **are proceeded with a dollar (\$) sign and are in CAPITAL LETTERS** – e.g. \$4F, \$1234 (otherwise, marks may be deducted).

To assist you, I have included the entries for the first line of interest in the supplied source code listing, which is line 19 **Idaa #\$25** and the contents of memory locations \$1000 and \$1001, which do not change during the program run. Note that memory locations \$1000 and \$1001 were filled with their values before line 19 executes, but the contents of memory locations \$1002 and \$1003 are unknown at the point in the program, so we annotate them as “- -”. I have also included the PC value that is evident **after** executing line 28 of the program listing. If you do not end up with that value, then you have not recorded the missing values correctly.

Line, Instruction		Values present AFTER execution of Line, Instruction							
		PC	A	B	D	\$1000	\$1001	\$1002	\$1003
19	Idaa #\$25	\$2005	\$25	\$00	\$2500	\$0C	\$81	--	--
20	Idab #25					\$0C	\$81		
21	adda MyArray					\$0C	\$81		
22	staa Result					\$0C	\$81		
23	stab Result+1					\$0C	\$81		
24	Idaa Result+1					\$0C	\$81		
25	Idab Result					\$0C	\$81		
26	incb					\$0C	\$81		
27	Idaa #%10101010					\$0C	\$81		
28	std Result					\$0C	\$81		

Note that Idaa #% above is Immediate Mode, Binary value, versus #\$, which is Immediate Mode, Hexadecimal Value

Task One – Part Two Questions (12 marks) – Place your answers in the allocated space.

Now, based upon the values in the above table and the code listing on page 1 of this assignment, answer the following questions. Ensure all hexadecimal values are proceeded with a dollar (\$) sign and are in CAPITAL LETTERS – e.g. \$4F, \$1234 (otherwise, marks may be deducted).

Question	Answer
a. How did \$1000 and \$1001 initially have their memory addresses filled with the indicated values at line 24? To answer this question, replicate the complete line of code that initialized those memory locations to the values indicated	
b. What is the 16-bit address associated with the label “MyArray”?	
c. What is the 16-bit address associated with the label “Result+1”?	
d. Line 26 contains the instruction incb. What 16-bit address holds the opcode for this instruction?	
e. What is the opcode for incb?	
f. What addressing mode is used in line 26? To answer this question, look up Increment B (INCB) in the HCS12 Assembly Language Reference Manual. Give both the full name and abbreviated name for the addressing mode. (2 marks)	
g. What addressing mode is used in line 28? To answer this question, look up Store Double Accumulator (STD). This instruction uses a Source Form of opr16a. Give both the full name and abbreviated name for the addressing mode. (2 marks)	
h. What high-level functionality is performed on the values in A and B by program lines 22 – 25? (3 marks)	

Assignment 2B (40 marks) – Lab Week Seven Hand-In Sheet**Due: End of your week Eight's lab period: Week of 11 – 16 March 2018****Page 2 of 2****Task Two – Number Conversions (10 marks) – Place your answers in the allocated space.**

Perform the following number conversions, placing your answers in the **Value** column (10 Marks)

While you do not have to show your work, ensure that you adhere to the following standards for your answers; otherwise, marks may be deducted:

- All hexadecimal values are **8-bits only**, are proceeded with a dollar (\$) sign and are in **CAPITAL LETTERS** – e.g. \$4F
- All binary values are **8-bits only** and are proceed with a percent (%) sign – e.g %00001010

Convert	To	Value
\$F3 (unsigned)	Integer	
\$F3 (signed)	Integer	
\$7FFF	Integer	
82	Hexadecimal (signed 2's complement)	
-82	Hexadecimal (signed 2's complement)	
-3	Hexadecimal (signed 2's complement)	
127	Binary	
-127	Binary (signed 2's complement)	
11001101 (signed)	Integer	
11001101 (unsigned)	Integer	

Note: The Course Resources folder on Blackboard contains a Tutorial that explains how to perform two's complement both manually and with a Sharp Calculator