

ECET310 - INTRO TO MICROPROCESSORS

Lab Experiment #: 3

Team #:

Number of weeks: 2

Instructor: Mr. E. Paterno



LABORATORY EXPERIMENT

Starting Date:

Date Submitted:

Course: ECET310 Sect:

Semester:



Experiment Title:

Laboratory Partners:

- 1:
2:
3:

Responsibilities:

- 1:
2:
3:

Grading Criteria:

Part 1: 50%

Part 2: 50%

Total: 100%

Comments:

Lateness: Grade:

LAB Experiment #3: 2 weeks

Pre-Lab activities:

- ✚ View the Dragon12 tutorial, asmIDE tutorial, CodeWarrior tutorial, D-Bug12 Command list and, other useful documents found using the link below:

<http://web.njit.edu/~paterno/>

[Click on Classes / ECET 310 / Labs]

Some of the D-Bug12 commands of interest are:

ASM, BF, BR, NOBR, G, HELP, LOAD, MD, MDW, MM, MMW, RM, RD, FCC, FCB, PC and T.

- ✚ Review chapter 4 material and lecture notes on STACK, SUBROUTINES and, Debug-12 calling functions (i.e. **printf()** function).
- ✚ Students need to show the instructor at a minimum, the algorithm (flowchart + pseudocode) for each program prior to downloading it to the Dragon12 board. . Use the SIMHCS12 Java API to simulate your program. Optionally, students can show the instructor the CodeWarrior simulation result of their code (Give the CodeWarrior S19 resulting file to the instructor).

LAB Experiment #3 Objectives:

- ✚ Become familiar with the Dragon12 Evaluation board, the asmIDE software tool, and the Code Warrior IDE.
- ✚ Become familiar with the HCS12 assembly language and the D-Bug12 monitor commands
- ✚ Become familiar with directives (**org, equ, db, dw, rmb, rmw, swi, fcc, fcb**)
- ✚ Become familiar with loop structures, the HCS12 stack, and operation thereof (i.e. **PSH[A,B,C,D,X,Y], PUL[[A,B,C,D,X,Y], LEAS, LDS, INS, DES, STS**).
- ✚ Become familiar with HCS subroutine operations. Passing parameters to subroutines (i.e. passing by value, passing by reference – using registers, stack and global variables).
- ✚ Become familiar with the DEBUG-12 I/O calling functions, (i.e. specifically the **printf()** utility), and the **FCC** and **FCB** directives.

To achieve these objectives LAB#3 consists of two parts. Students will:

1. Modify lab experiment #2 part 1 HCS12 assembly program
2. Write a program satisfying certain requirements

LAB EXPERIMENT #3 – PART 1

Work item: Modify lab experiment #2 part 1:

The function to calculate the sum of **N** squared numbers and the function to compute the average of the **N** squared values are performed with the help of the two subroutines - **sum** subroutine and **average** subroutine. The sum subroutine accepts one parameter, (i.e. array), which is the starting address of the array holding the **N** numbers. As well, the **sum** subroutine returns one value (i.e. **total**). The average subroutine accepts 2 parameters, (i.e. **total**, **N**), and returns one value (**average**). The **sum** function's parameter is a global variable and the **average** function's parameters are registers. Furthermore, your program displays, using the **printf()** callable C utility function, the following message on the D-BUG-12 Terminal screen:

The sum of the squared **N** values is: **total**
The average is: **average**

Whereby **total** is the sum of the squares of the **N** given numbers, and average is the **average** thereof.

For example, for **N=4** and **\$1, \$3, \$5, \$7**, being the four numbers, the screen output looks like:

The sum of the squared **4** values is: **84**
The average is: **21**

The original program specifications are listed below:

Write a program to compute the average of the square of **N** 8-bit unsigned numbers – That is, given **N** numbers, compute the squared value for each number, compute the sum of the **N** squared values, and finally compute the average.

- ✚ The **N** 8-bit unsigned numbers are stored starting in memory location \$2000. The necessary memory block is thus, \$2000:\$2000 + (**N**-1).
- ✚ The program must use a loop to perform the computation.
- ✚ The computed average is to be stored in memory location \$2000 + **N**.
- ✚ Your program must be able to handle a value of **N** such that: $4 < \mathbf{N} < 255$.
- ✚ The program must be written such that it can handle 16 bit values when computing the sum and the average.

Tasks to be performed:

1. Define your algorithm by drawing a flowchart. **Show the instructor [].**
2. Write down your pseudocode. **Show the instructor [].**
3. Write the AS12 assembly code that implements your algorithm.
4. Enter/assemble/download/debug your code using CodeWarrior / AsmIDE / Dragon12.
5. **Show the instructor your working code.**
The instructor will provide the value N and the N numbers to enter in the program [].

LAB EXPERIMENT #3 – PART 2

Work item: Write a short program satisfying certain requirements

Write a program that sums up **N** products each of two 8-bit unsigned numbers. Each pair of numbers is obtained from two distinct arrays of numbers called respectively as **array1**, **array2**. Thus, element **J** of array1 is multiplied respectively with element **J** of array2, whereby $0 \leq J \leq N-1$, and the resulting products are summed up with the final sum stored in one memory location called **sum**.

This computation is summarized in the formula below:

$$sum = \sum_{J=0}^{J=N-1} array\ 1[J] \bullet array\ 2[J]$$

Assume that the HCS12 CPU does not have a multiplication instruction so that you need to write a subroutine that multiplies two 8-bit unsigned integer numbers and return the result of the multiplication as a 16-bit number. The algorithm for the multiplication function using the C++ programming language format is shown below:

To evaluate:

```
product = num1 * num2;
```

Perform the repeated addition computation:

```
For (K = 0, product = 0; K < num2; ++K)
{
    product = product + num1;
}
```

For example:

```
Product = 5 * 4;
```

Perform the repeated addition computation:

	<u>K</u>	<u>product</u>
	0	5
+ 5	1	10
+ 5	2	15
+ 5	3	20

This results in a product equal to: 20

ECET310 - INTRO TO MICROPROCESSORS

- ✚ The choice of the multiplication subroutine, *multi*, interface is left up to the students (i.e. passing by value / passing by reference using registers or the stack or global variables).
- ✚ The specific memory choice of locations for the data, program and stack is also left up to the students.
- ✚ The program will display the sum result on the D-BUG12 Terminal screen using the **print()** callable C function with the help of the **FCC** and **FCB** directives.

Tasks to be performed:

1. Define your algorithm by drawing a flowchart. **Show the instructor** [].
2. Write down your pseudocode. **Show the instructor** [].
3. Enter/assemble/download/debug your code using CodeWarrior / AsmIDE / Dragon12 evaluation board.
4. **Show the instructor your working code. The instructor will give you the value of the number that needs to be converted to BCD/ASCII format** [].