

## Assignment 2C (50 marks) – Lab Week Eight (Due: End of your week Nine's lab period).

**Due: End of your week Nine's lab period: Week of 3 – 6 Nov 2015**

Late submissions will not be accepted and will receive a mark of zero (0).

### Third Assembly Program – A Simple Hardware Programming Exercise

This portion of the lab exercise leads you through the programming of Light Emitting Diodes (LEDs) using 68HCS12 Assembly Language as the target language using *AsmIDE* and *Simulator – Dragon12 & Student Mode*.

### Fourth Assembly Program – The Little Endian Challenge

This portion of the lab confirms your understanding on how to write assembly language code to copy data from one memory location to another while at the same time swapping the values so that the data is changed from 16-bit Big Endian to 16-bit Little Endian. It will also provide you with the opportunity to use the Assembler Help files and select HCS12 instructions.

### Fifth Assembly Program – Those were the Memories

This portion of the lab confirms your understanding of pre and post program run memory maps as well as number conversions.

### Number Conversions and BCD Arithmetic

This portion of the lab confirms your understanding of number conversions and BCD Arithmetic.

### 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 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](#)

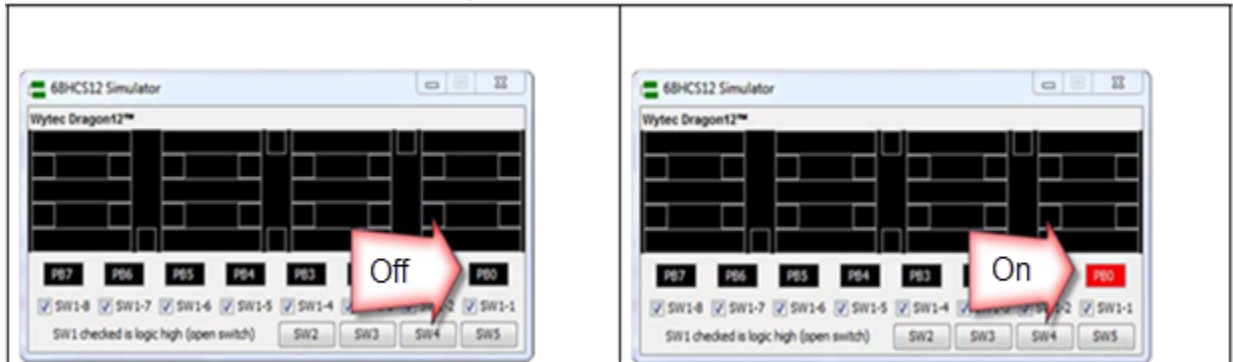
### TASK One – Flash\_PB0 – Correct the Code; Assemble; Code Inspection; Demo (10 Marks)

The purpose of the supplied assembly language is to flash an LED on and off every 250 ms. We will use the simulator to observe the correct program run.

### Procedure:

- Download the compressed Library Files package (*Library\_Files.zip*). Create a new folder called **Lib** in your **C:\68HCS12** folder (this was the folder where you originally installed the assembly language software). Unzip *Library\_Files.zip* into **C:\68HCS12\Lib**.
- Download the corrupted *Flash\_PB0.asm* into your Week Eight source directory (e.g. CST8216\Lab8) and correct the assembly language code source code listing so that the labels, opcodes, operands and comments are in the correct columns. You will also have to add the two missing lines of code that will include the two library files found in **C:\68HCS12\Lib** (read those files for instructions on how to use them – ensure that their "include" statements are placed just before the "end" statement in your code). **Do not change any other lines of code in this source file, noting that if you receive the following error, then you incorrectly installed the software package and that you must reinstall it:** Fatal error -- Can't open #include file C:\68HCS12\registers.inc
- Once you have the previous steps complete, assemble *Flash\_PB0.asm* ensuring that there are no errors or warnings.
- Load the .s19 file into the  **Simulator - Dragon12 & Student Mode** and take the following action to run it:
  - On the main menu, click on View → Parallel Ports
  - For your convenience, I have included a short video presentation on the expected behaviour of this program. Observe that the simulator and parallel ports are identical to the ones in that video. If they are not, then you are using the wrong simulator and you will have to select the correct one before proceeding any further

- III. Next, click the “Go” button on the simulator and observe that the Port B LED (PB0) on the simulator Parallel Port flashes on and off at a rate of about 250 ms, or ¼ of a second:



- IV. Your solution should very closely match it. Once the program run is correct, demonstrate it, printing out the Lab Week Eight Hand—In Sheet BEFORE doing so. Note that your timing **may** be slightly different from that illustrated in the video.

## TASK Two – The Little Endian Challenge – Write the Code; Demo Solution in Simulator (10 Marks)

This portion of the lab confirms your understanding on how to copy data from one memory location to another while at the same time swapping the values so that the data is changed from 16-bit Bit Endian to 16-bit Little Endian. It will also provide you with the opportunity to use the Assembler Help files and select HCS12 instructions.

Building upon your knowledge of assembly language from supporting lecture material/your lecture notes, you are to write a complete assembly language program (**Big\_To\_Little\_Endian.asm**) that takes an array that is stored as Big Endian data and converts it to Little Endian data.

### Procedure:

- Create the Source\_Array (starts at \$1000) using an **origin** pseudo op code and the **define word** pseudo op code.
- Reserve memory space for the Destination\_Array (starts at \$1020) by using an **origin** pseudo op code and the **define storage word** pseudo op code.
- Use an **origin** pseudo op code so that the program code starts at \$2000. Write your code starting at that address
- Initialize the stack to \$2000 (e.g. lds #\$2000)
- Without using iteration (e.g. no loops), take one 16-bit Big Endian value at a time from the Source Array and store it as a 16-bit Little Endian value in the Destination Array. There are a total of three 16 bit values to convert.
- Don't forget to include a program header and appropriate documentation of your code

### Pre-Execution Memory Map

Memory Display Address 1000 Show																
ADDR	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1000	10	fe	28	88	aa	55	00	00	00	00	00	00	00	00	00	00
1010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

### Post-Execution Memory Map

Memory Display Address 1000 Show																
ADDR	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1000	10	fe	28	88	aa	55	00	00	00	00	00	00	00	00	00	00
1010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1020	fe	10	88	28	55	aa	00	00	00	00	00	00	00	00	00	00

### Task Three – Those were the Memories (16 Marks)

To complete this task, analyze the given code and complete the memory maps *as if you were using the actual hardware, not just the simulator* and complete the Pre-Execution and Post-Execution Memory Maps.

Note that dw = define 16-bit word and the values after dw occupy 2-eight bit contiguous memory locations. Pay particular attention to the **base** of the numbers in the source code and how they must be stored in memory.

Hint: Perform this task MANUALLY, and THEN check your answers using AsmIDE and the *Dragon12 & Student Mode Simulator*. Use your lab period to ask questions if you don't understand the results.

**PLACE YOUR ANSWERS ON THE HAND-IN SHEET.**

Given the following source code listing, complete the Pre-Execution Memory Map and Post-Execution Memory Map tables below.

- Ensure all values are expressed in Hexadecimal
- Where memory contents are unknown by the hardware, fill in the entry with two dashes – e.g. "--".
- Ensure all memory addresses contain a value or "--"

Pre-Execution Memory Map

	← 8 bits →
\$1000	
\$1001	
\$1002	
\$1003	

Post-Execution Memory Map

	← 8 bits →
\$1020	
\$1021	
\$1022	
\$1023	

```

1 ; Those_Were_The_Memories.asm
2
3 Value3 equ 128
4
5 org $1000
6 Value1 db $AA
7 Value2 dw 9876
8
9 org $1020
10 Result1 ds 1
11 Result2 ds 2
12 Result3 ds 1
13
14 org $2000
15 ldaa Value1
16 staa Result3
17 ldab #Value3
18 stab Result1
19 ldab $1002
20 ldaa $1001
21 std Result2
22 swi
23 end

```

### Task Four – Number Conversions and BCD Arithmetic (14 Marks)

To complete this task, complete the material on page 2 of the Hand-In Sheet.

**Assignment 2C (50 marks) – Lab Week Eight Hand-In Sheet (Page 1 of 2)****Due: End of your week Nine's lab period: Week of 3 – 6 Nov 2015**Please  
staple the  
pages  
together.**Name:** \_\_\_\_\_**Circle Your Lab Period/Time****Student Number:** \_\_\_\_\_

Tue: 10 – 12    Wed: 1 – 3    Wed 3 – 5    Fri 2 – 4

**TASK One – Flash\_PB0 – Correct the Code; Assemble; Code Inspection; Demo (10 Marks)**

Code Inspection (6 marks):

Your Mark:

/50

- The filename for the code should be "as provided" in this assignment and your Student Information should be evident in the program header.
- The code should be correctly aligned with all Labels, Opcodes, Operands and Comments in their correct columns as discussed in a previous lecture.
- The program code should contain the missing two lines of code in the correct location in the source code.
- The two supplied library files must be in the correct folder on your system.

Program Run (4 marks): The program must run in the simulator without error on the first demonstration attempt.

Professor's Initials \_\_\_\_\_ /10 (In-class Code Inspection + Demo)

**TASK Two – The Little Endian Challenge – Write the Code; Demo Solution in Simulator (10 marks)**

Code Inspection (6 marks):

- Source\_Array starts at \$1000
- define word** pseudo op code used to create 16-bit array values
- Destination\_Array starts at \$1020
- define storage word** pseudo op code used to create the correct amount of 16-bit reserved memory space
- program code starts at \$2000
- stack initialized to \$2000
- Without using iteration 16-bit Big Endian values are taken one at a time from the Source Array and stored as 16-bit Little Endian values in the Destination Array. There are a total of three 16 bit values to convert.

Program Run (4 marks): The program must run in the simulator without error on the first demonstration attempt.

Professor's Initials \_\_\_\_\_ /10 (In-class Code Inspection + Demo)

**Task Three – Those were the Memories (16 marks)**

Complete the memory maps here according to the instructions contained within this assignment:

**Pre-Execution Memory Map**

	← 8 bits →
\$1000	
\$1001	
\$1002	
\$1003	

**Post-Execution Memory Map**

	← 8 bits →
\$1020	
\$1021	
\$1022	
\$1023	

Assessed Post-Lab \_\_\_\_\_ /16

