Microcomputers 1

Lab 2
Assembly Language Programming and Debugging

Winter 2022

Due Date: Midnight Tuesday Feb. /08/2022

Objectives:

- Create and execute the **HCS12 assembly program** in CodeWarrior.
- Learn how to read and modify the memory and CPU registers in HCS 12 microcontroller for debugging.
- Learn how to trace the program execution by setting breakpoints.

Total Points: 100 points

Part I. *An Assembly program to add numbers*: in full simulation

Follow the procedures given below to create your project with the help of the project wizard.

1. Create a Lab02 folder on your computer. Next, open CodeWarrior IDE by double-clicking on its desktop icon. You will see the Startup window if 'Display on Startup' is turned on. Next, click the 'Create New Project' button. If you do not see the Startup window, click on "New Project …" under the File menu. This starts the New Project wizard.

2. In the HC(S)12(X) Microcontrollers New Project wizard window, select "**HCS12**" →"**HCS12D Family**" → "**MC9S12DG256B**" as shown in Figure 1.

3. In the Connections pane, select "Full Chip Simulation". Then, click Next.

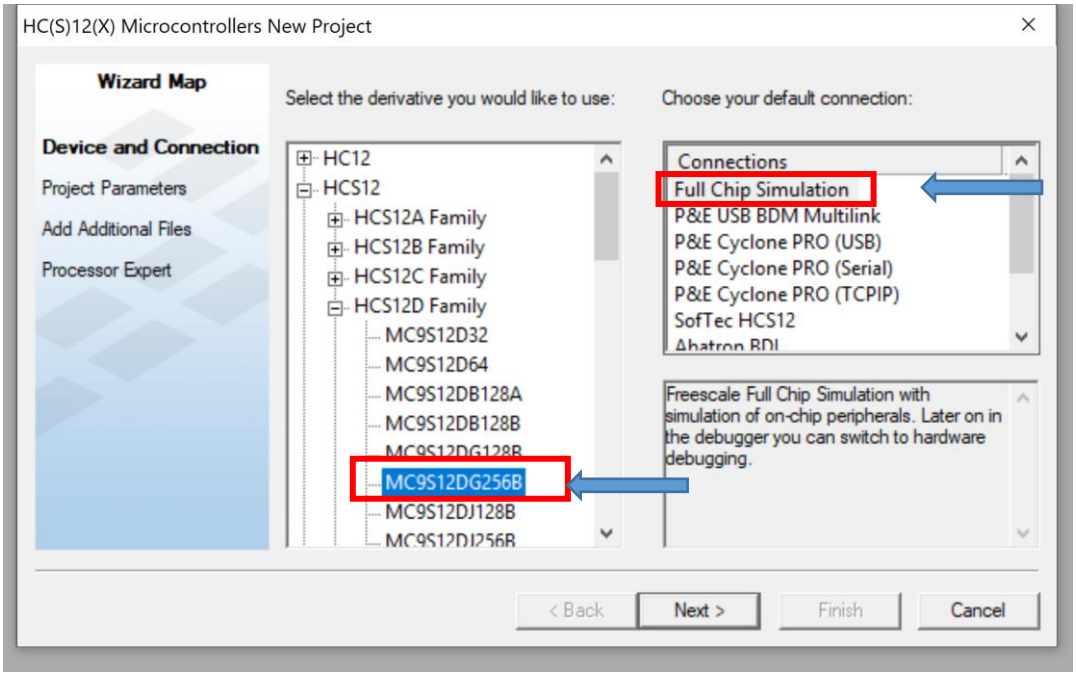


Figure 1.

4. The second **New Project** window opens up, as shown in Figure 2. Use the **SET…** button to point to folder Lab02 in the **Location** field. Then, in the **Project name** field, type your (arbitrary but in general meaningful) project name, say, **Lab2_part1.mcp**. Note that your project name is appended to the **Location** field.

5. Uncheck all the checkboxes and then check the **Absolute assembly** checkbox as shown in Figure 2. Finally, click on **Finish** (NOT Next!!).

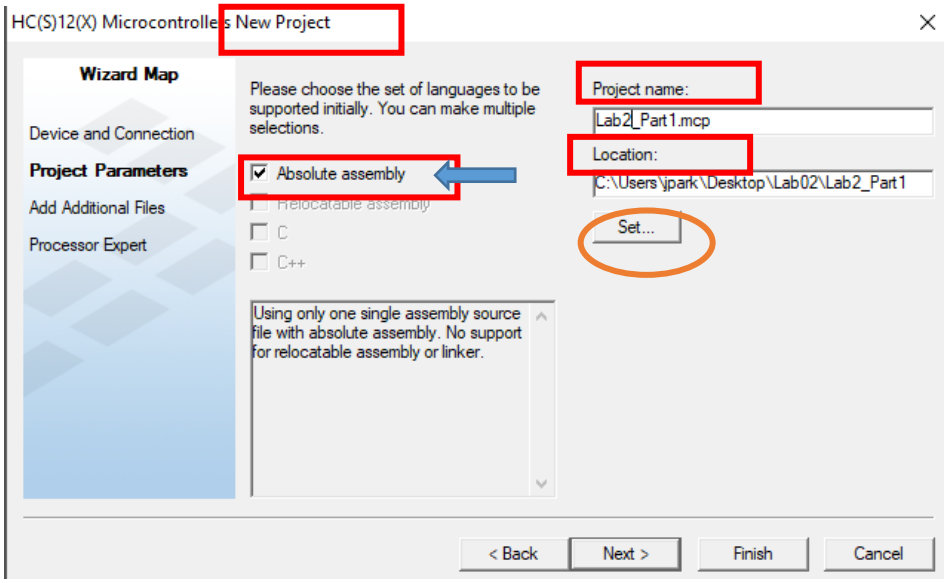


Figure 2.

6. CodeWarrior main window opens up as shown in Figure3. From the dropdown list on the left, select "**Full Chip Simulation**" if it is not so yet. Double click the file, 'main.asm' in Figure 3.

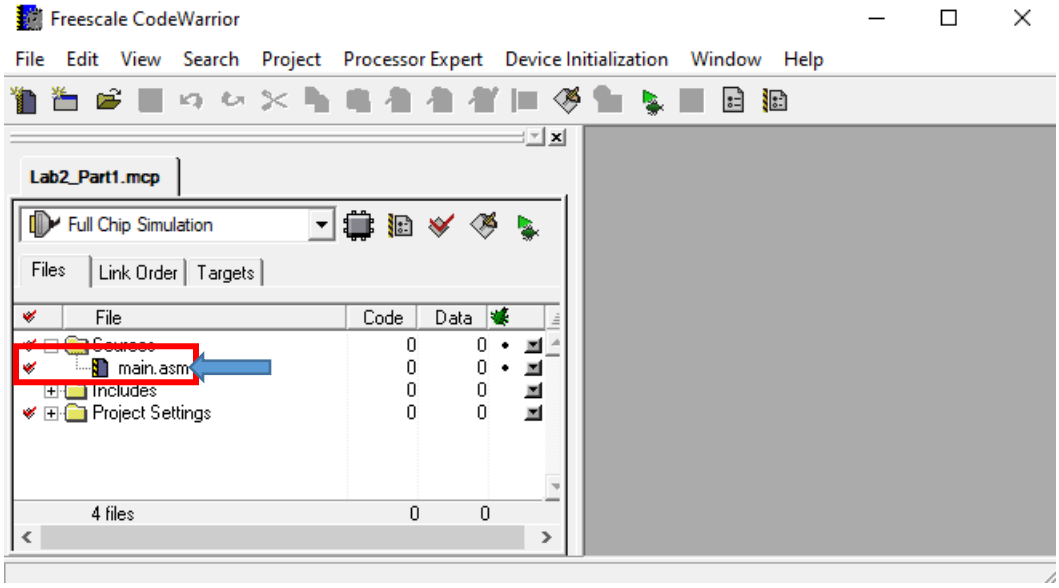


Figure 3.

7. Delete the prewritten sample code marked with black background in main.asm provided by the CodeWarrior IDE.

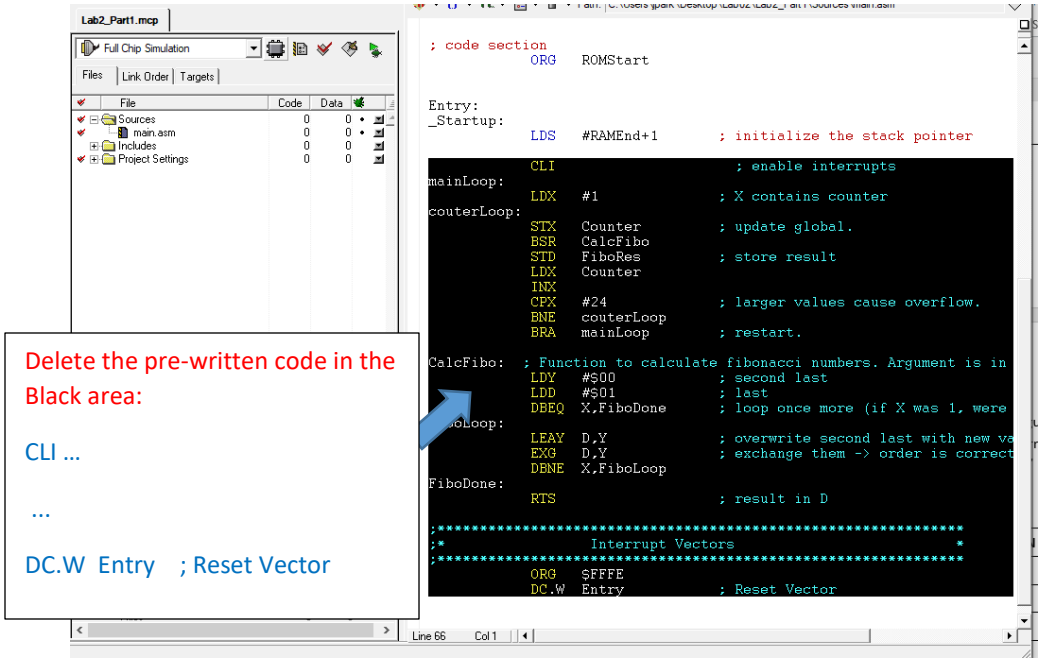


Figure 4.

8. The assembly program for Lab2-Part I is shown in the table of Figure 5. Your task is to replace **instructions (only with the extended addressing mode) in column 2** with the instructions that do the same job but **use the *indexed* addressing mode with offset**. For the task, you will

complete the assembly code in the 3rd column in the table using the *indexed* addressing mode with offset. Then, write your assembly code in main.asm.

Code Line #	Extended Assembly	Indexed with offset		PC	N	Z	V	C
		Assembly	Machine code					
1	LDAA $3000	LDX #$3000						
2		LDAA 0,X						
3	LDAB $3001							
4	ABA	ABA						
5	STAA $3002							
6	LDAA $3003							
7	LDAB $3004							
8	SBA	SBA						
9	STAA $3005							
endmain: BRA endmain								

Figure 5 : Program Code.

9. To place your assembly instructions starting the memory address $4100, you need to **add the assembly directive, ORG $4100, just before code line 1**, as shown in Figure 6. Click on the green **debug arrow** on the menu bar in Figure 6. The simulator/Debugger window opens up.

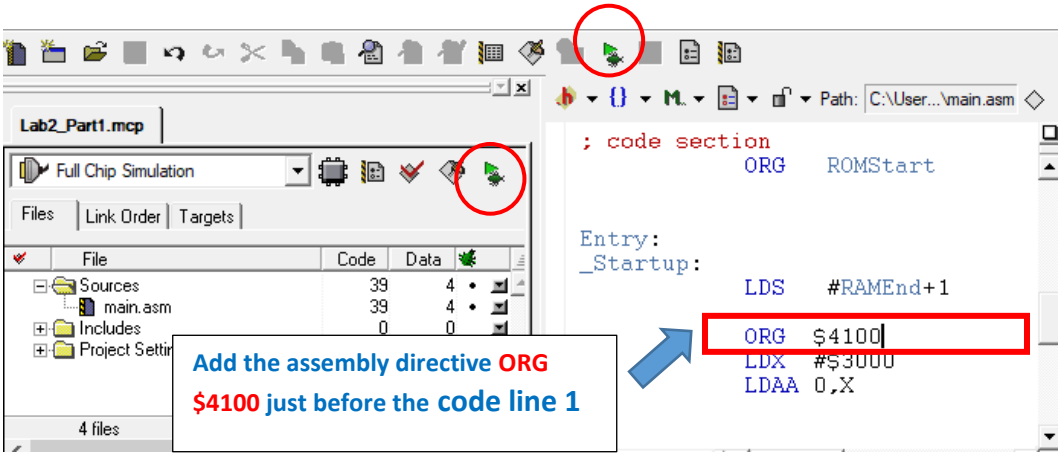
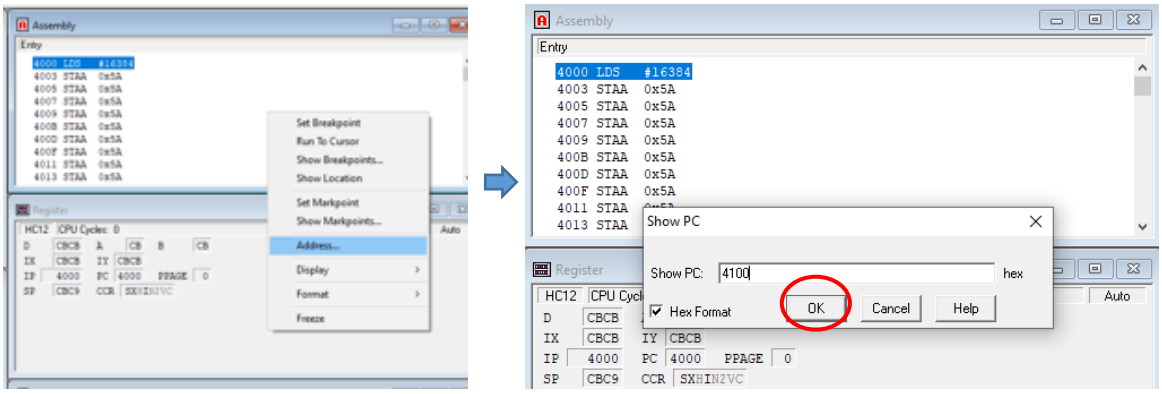


Figure 6.

10. Check out the **Assembly** pane: The assembly code that you entered in main.asm listed by **CodeWarrior**. In the **Assembly** pane, prefix **0x** signifies base 16. If you can not see the address $4100, right-click on the **Assembly** pane and select **Address…** in the pop-up menu. The **Show PC** window pops up. Type 4100 in the **Show PC** field and then hit OK.



11. The data (operands) of the addition and subtraction instructions are shown in the table in Figure 7. Initialize four memory locations as shown in the table in Figure 7. These are the data to be used by your instructions:

Memory address	Contents
$3000	**$9C**
$3001	**$B5**
$3002	--
$3003	**$3E**
$3004	**$F7**
$3005	--
$3006	--
$3007	--
$3008	--

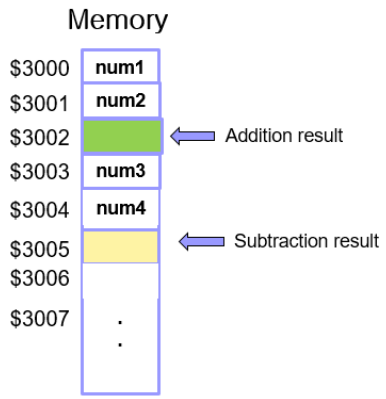
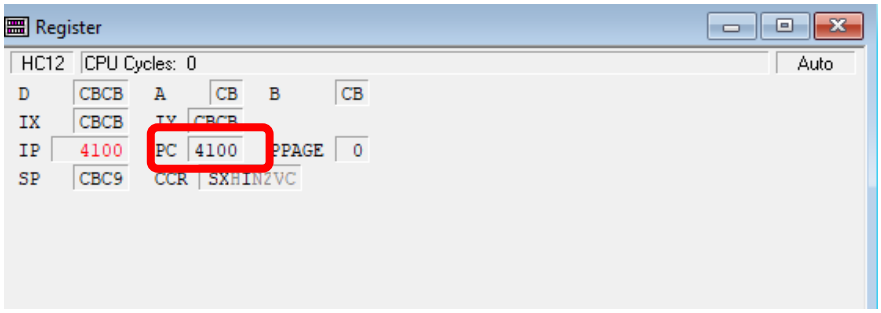


Figure7. Initial Data

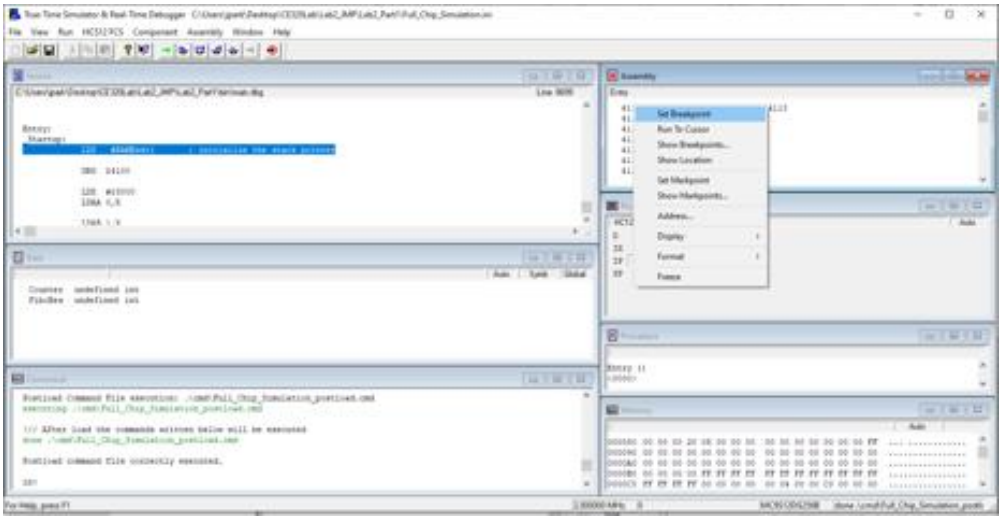
Q1: Do the following addition and subtraction by hand. Mark whether there is overflow or not.

$9C + $B5	**Overflow** (if unsigned number):	Yes	No
	Overflow (if signed number):	Yes	No
$3E - $F7	**Overflow** (if unsigned number):	Yes	No
	Overflow (if signed number):	Yes	No

12. Use CodeWarrior to execute the instructions and then verify your results obtained above. In the CPU **Register pane**, change the value of **PC to $4100**, the starting address of your code by **clicking on the textbox for the program counter (PC).**



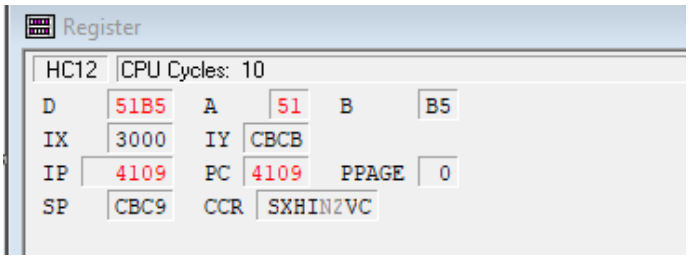
13. Add a breakpoint at the final instruction's address "**$4113**" to stop the debugger at a logical point for the user's inspection. In the **Assembly pane,** select the line starting with $**4113** that is an address. Then, right-click the line and choose the 'Set Breakpoint' item.



14. F11 is the single-step button. Use it to execute one instruction at a time. Hit F11, then the first instruction executes (**LDX #$3000**), and the next one (**LDAA 0,X**) is highlighted. Check the affected registers. Note that **a CCR flag is pulled up (i.e., set to logic 1) if it is dark.** Use F11 to execute the instructions one at a time. Before you proceed to the next instruction, look at the simulation results.

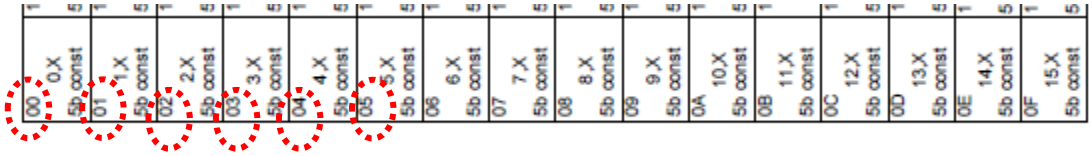
    a) Fill the N Z V C bits in CCR in the table in Figure 5 by using **F11** to execute the instruction one at a time. Use the provided Lab2_Workbook.doc file to fill the table in Figure 5. Note that **a CCR flag is pulled up (i.e., set to logic 1) if it is dark.** Use F11 to execute all the instructions once at a time.

    Ex) The example below: N=0, Z=0, V=1, C=1



    b) Fill the machine code in the table in Figure 5 by checking the memory locations for the machine code starting from $4100.

    Note: For the machine code for indexing address mode, see the reference manual page 20.



    Ex) The machine code for 0,X indexing address mode is 00.

    The machine code for 1,X indexing address mode is 01, etc.

15. *Then c*lose the simulator/Debugger window after you complete Part1. Close the project.

Part II. Assembly directives:

1. Create the new project **Lab2_part2.mcp** by following the steps (1-7) explained in Part I.
2. In Part2, instead of entering data *directly* and *manually* in the memory, you will **use assembler *directives*** to let CodeWarrior do it for you.

3. The data definitions are placed in the data declaration part in main.asm as shown in Figure 8. First, delete the prewritten sample code provided by the CodeWarrior IDE (marked with black background) in main.asm.

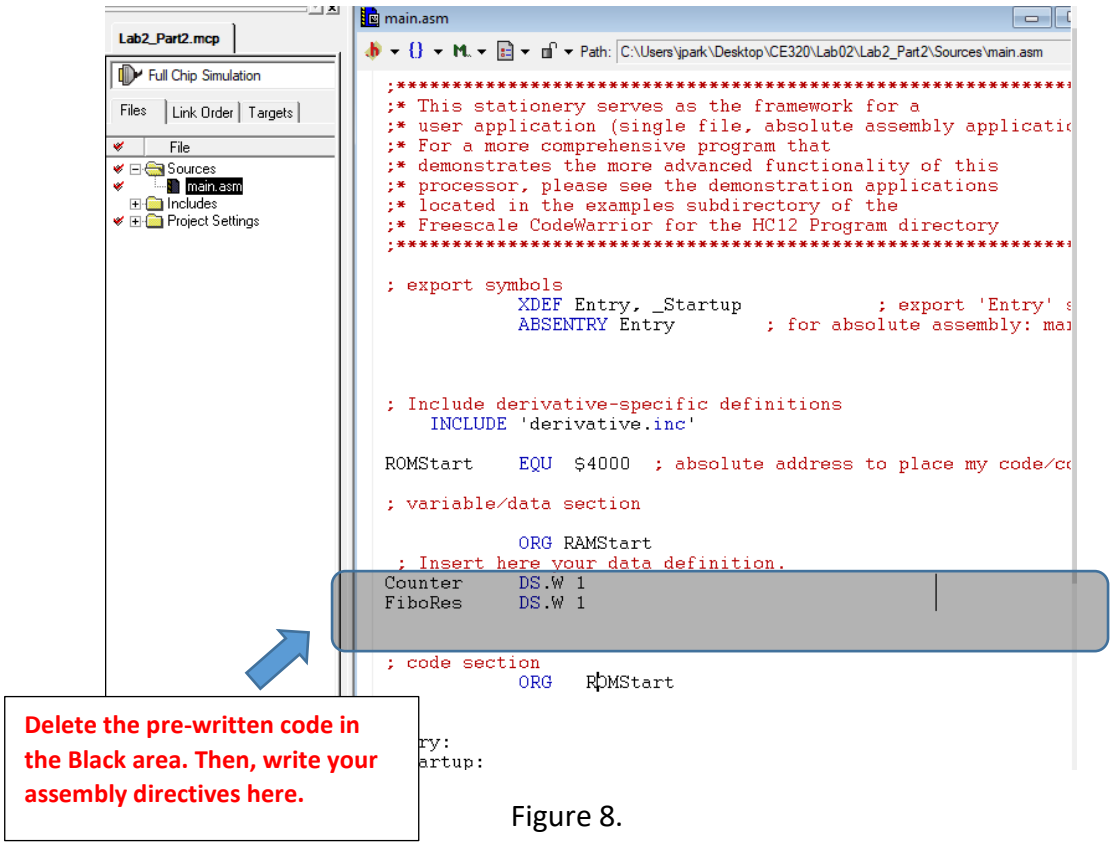


Figure 8.

4. Type the below assembly directives in main.asm as shown in Figure 9.

```
ORG    $3000   ; tells assembler where to place the data declarations after this org directive
               ; data will be placed starting from the memory location $3000

operandA:  DC.B   $9C      ;operandA  is a meaningful name of the memory address $3000
operandB:  DC.B   $B5      ;operandB  is a meaningful name of the memory address $3001
Result1:   DS.B   1       ;Result1 is a meaningful name of the memory address $3002
operandC:  DC.B   $3E      ;operandC  is a meaningful name of the memory address $3003
operandD:  DC.B    $F7     ;operandD is a meaningful name of the memory address $3004
Result2:   DS.B    1       ;Result2 is a meaningful name of the memory address $3005
```

Note: All labels (green color) should be placed starting in column 1.

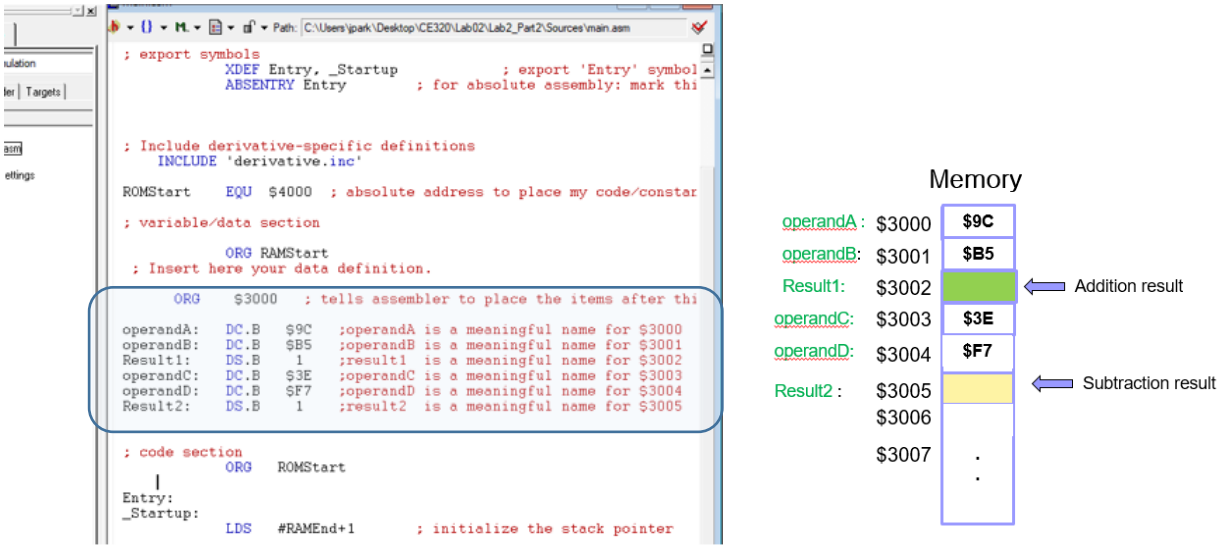


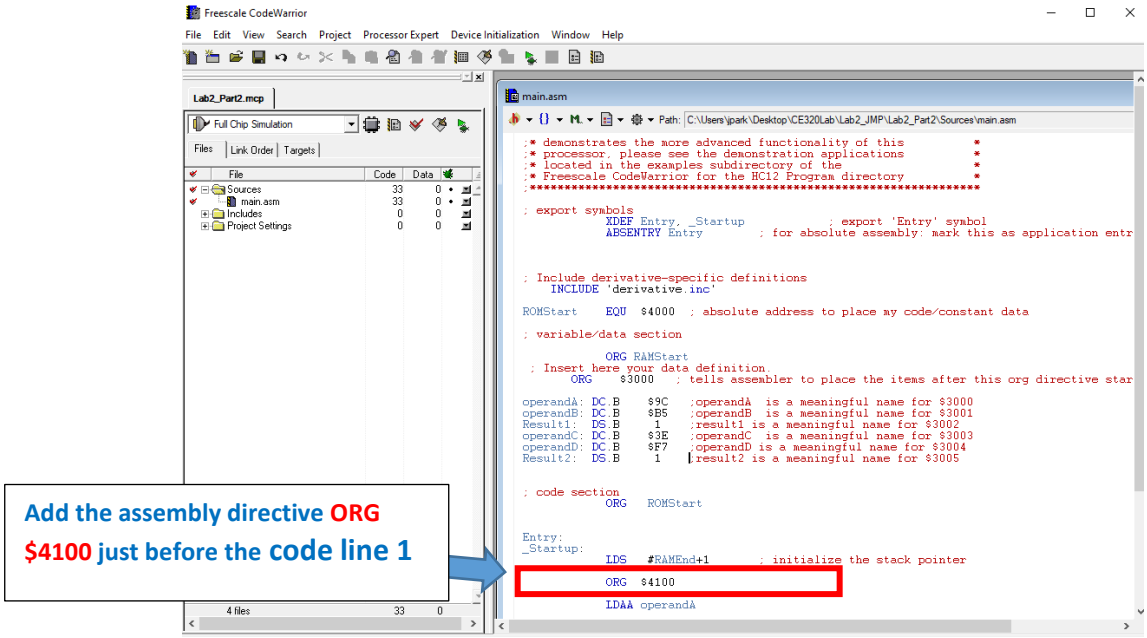
Figure 9.

5. Write your code **in column3** using the labels you defined in the data section. Fill the table in Figure 10. For example, the memory address $3000 will be replaced with the label, 'operandA' and replace other memory addresses similarly.

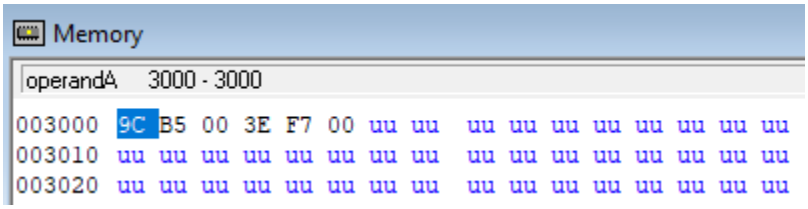
Code Line #	Extended Assembly	Using Labels		PC	N	Z	V	C
		Assembly	Machine Code					
1	**LDAA** $3000	**LDAA** operandA						
2	**LDAB** $3001							
3	**ABA**	**ABA**						
4	**STAA** $3002							
5	**LDAA** $3003							
6	**LDAB** $3004							
7	**SBA**	**SBA**						
8	**ST**AA $3005							
endmain: BRA endmain								

Figure 10 - Program Code.

6. Then, write your assembly code in main.asm. To start your assembly instruction at the memory location of $4100, you need to **add the assembly directive ORG $4100 just before code line 1,** as shown in Figure 11.

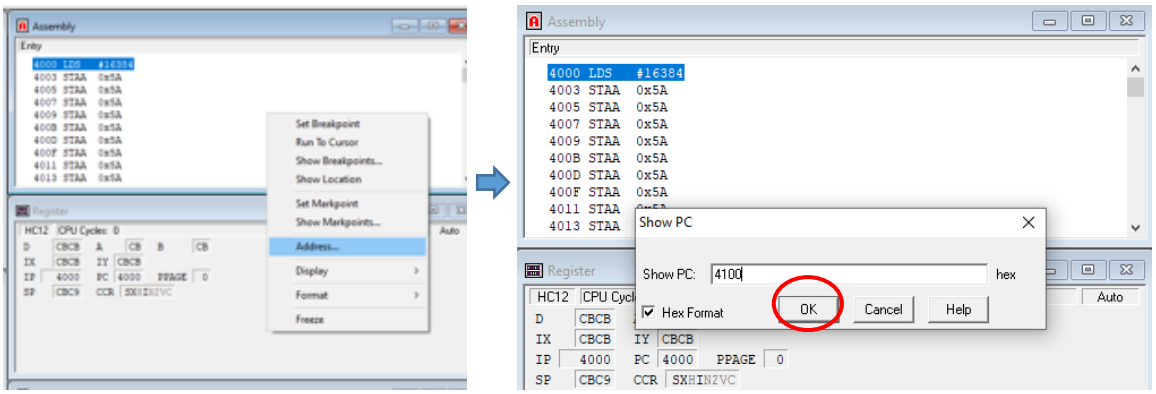


5. Click on the green **debug arrow** on the menu bar. The simulator/Debugger window opens up.
6. Check out the **Memory pane**:
   - Data: The assembler *directives* entered the data directly in the memory. Go to the memory location $3000 and check the values. You should see the data you defined in the data section.

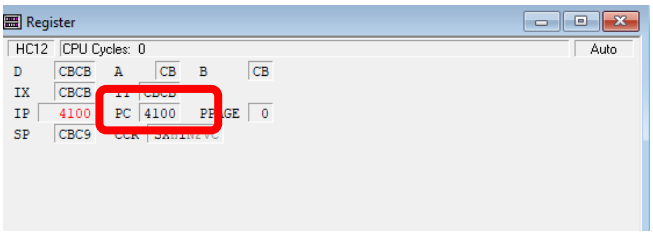


   - Instructions: The assembly instructions are converted into machine code and saved in the memory locations starting from $4100. Go to the memory location $4100 and check the machine codes. Fill the machine code in the table in Figure 10.

7. **Assembly pane**: In this pane, prefix **0x** signifies base 16. If you can not see the address $4100, right-click on the **Assembly** pane and select **Address…** in the pop-up menu. The **Show PC** window pops up. Type 4100 in the **Show PC** field and then hit OK.



8. Add a breakpoint at the final instruction's address "$**4116**" to stop the debugger at a logical point for the user's inspection. In the **Assembly pane,** select the line starting with $**4116** that is an address. Then, right-click the line and choose the 'Set Breakpoint' item.
9. In the CPU **Register pane**, change the value of **PC to $4100**, the starting address of your code, by **clicking on the textbox for the program counter (PC).**



10. Fill in the values in the table in Figure 10 by using **F11** to execute the instruction one at a time.

What to Submit:

- Group submission that includes:
  - 30pts: Part1: main.asm ; main.asm is located under the folder \ Sources.
  - 30pts: Part2: main.asm
- Individual Submission
  - 40pts: Per each member - complete the workbook, **Lab2_workbook.**
    - *Fill the table in Figure 5*
    - *Fill the table in Figure 10*
    - *Provide the answers to the questions in Q1.*

- **Pay attention to the file name convention:**

- **Individual file: Lab2_Student_Firstnname_Lastname.pdf**
  Ex) Lab2_Smith_Green.pdf

- **Group file: Lab2_Student1 Lastname_Student2 Lastname.pdf**
  Ex) Lab2_Green_Owens.pdf