Microcomputers 1
Lab 1

HCS12 Machine Language Programming
Getting Started with CodeWarrior

Winter 2022

Due Date: Tuesday midnight Feb. /01/2022

Objectives: Create and execute the HCS12 machine language program in the CodeWarrior Debugger.

- Learn how to use the CodeWarrior to simulate HCS12 instruction execution
- Learn how to use the trace and breakpoints to monitor a program's execution.
- Learn how to read and modify memory and CPU registers during debugging.
- Understand how to detect overflow for signed and unsigned binary numbers.

Assignments:

In Lab1, you will **execute some basic machine instructions** of the HCS12 microcontroller manually and then run them using a simulator/debugger called CodeWarrior. **Working in a simulation environment will be useful in the early development of software and not having access to the actual hardware**. For many software projects, you can write and test your programs on the simulator running on your personal computer or laptop. Once a program is tested for correct execution in a simulation environment, it can also be easily downloaded to the actual microcontroller hardware (the HCS12 chip on the Dragon-12+ board) for execution on the target device.

Task1. How to use CodeWarrior:

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

1. Create a folder, 'Lab01', on your computer. Then, open CodeWarrior IDE by double-clicking on its desktop icon or from the Start menu.

2. You will see the Startup window if **'Display on** Startup' is turned on. 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.

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

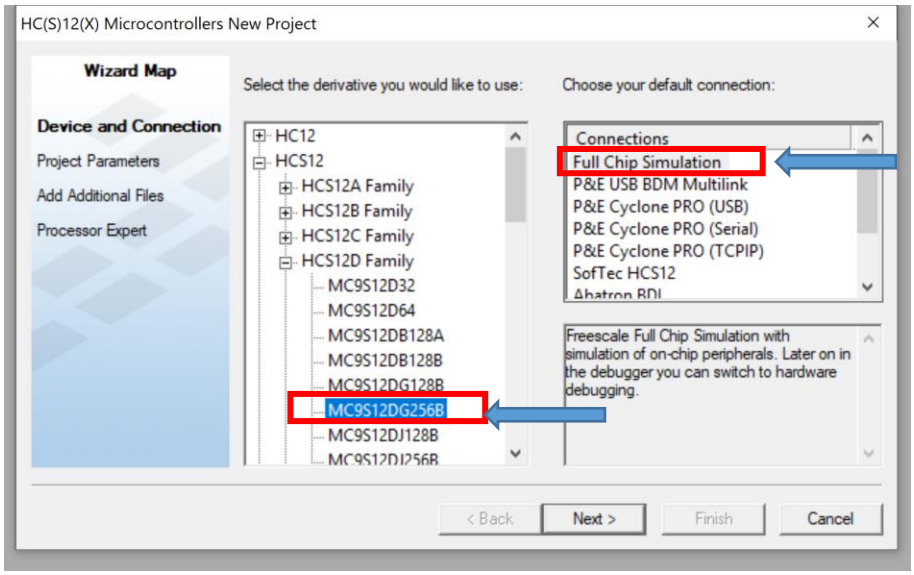


Figure 1.

4. In the Connections pane, select “Full Chip Simulation”. Click Next.

5. The second **New Project** window opens up, as shown in Figure 2. Use the **SET…** button to point to folder Lab01 in the **Location** field. Next, in the **Project name** field, type your (arbitrary but in general meaningful) project name, say, **Lab1.mcp**. Note that your project name is appended to the **Location** field.

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

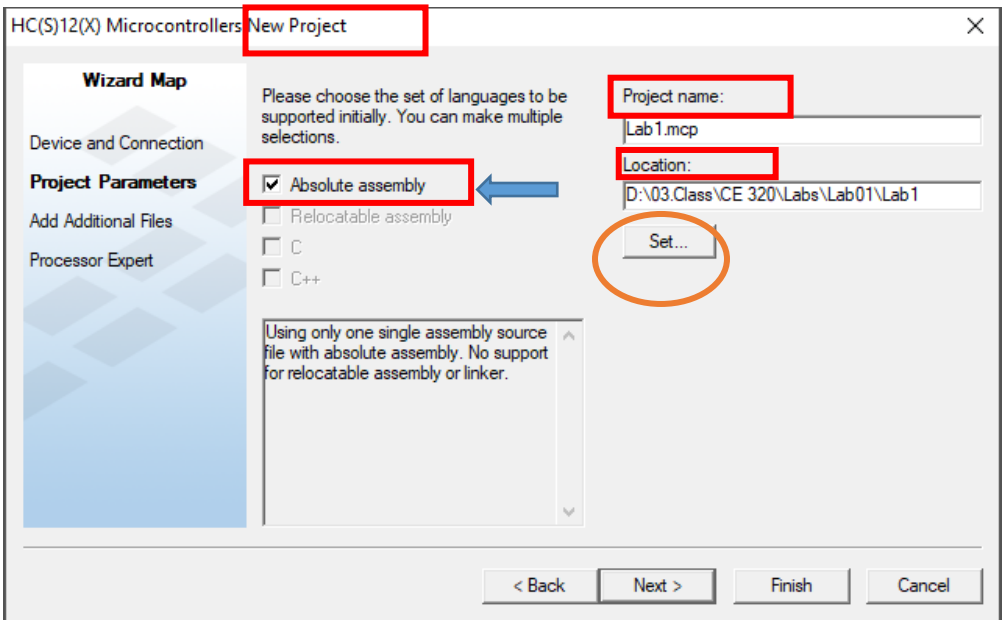


Figure 2.

7. CodeWarrior main window opens up, as shown in Figure3. First, select “**Full Chip Simulation**” from the dropdown list on the left if it is not so yet. Next, click on the green **debug arrow** on the menu bar.

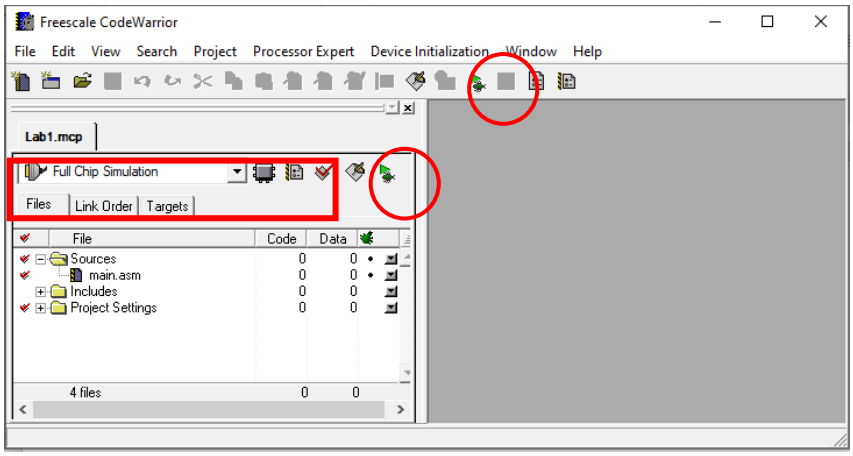


Figure 3

8. The **simulator/Debugger window** opens up as shown in Figure 4. There are 7 panes (or sub-windows) in this window, but today you will use only 3 of them: **Assembly**, **Register**, and **Memory** panes.

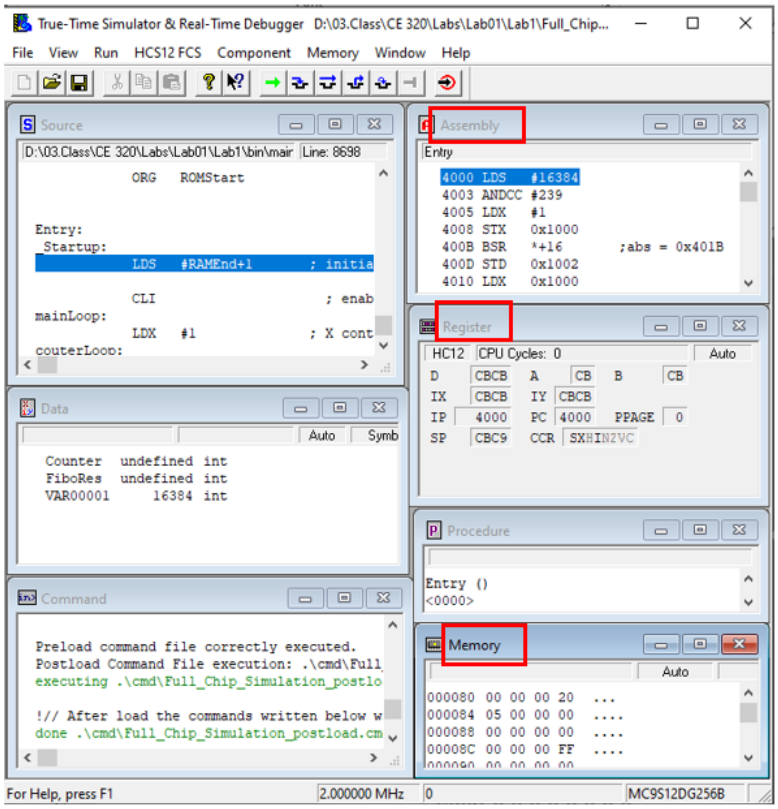


Figure 4

Task II. Machine Code for Simple Arithmetic Operations

Q1: Do the following addition and subtraction by hand:

$76 + $3A

If these numbers are unsigned numbers, **overflow:** yes no

If these numbers are signed numbers, **overflow**: yes no

$A3-$6C

If these numbers are unsigned numbers, **overflow:** yes no

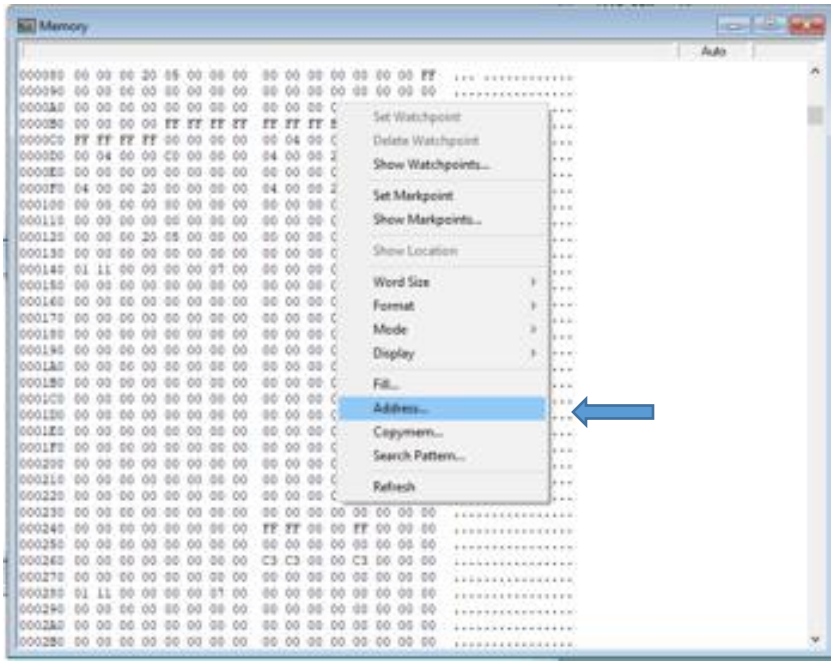
If these numbers are signed numbers, **overflow**: yes no

Take a close look at the table shown in Figure 5. It shows an assembly language program and the corresponding machine codes to load the data from the memory locations into the CPU registers, add and subtract numbers, and save the data in CPU registers into different memory locations.

Code Line	Memory Address	Machine Codes	Assembly Code
1:	$4100	B6	LDAA $3000
	$4101	30	
	$4102	00	
2:	$4103	F6	LDAB $3001
	$4104	30	
	$4105	01	
3:	$4106	18	ABA
	$4107	06	
4:	$4108	7A	STAA $3002
	$4109	30	
	$410A	02	
5:	$410B	B6	LDAA $3003
	$410C	30	
	$410D	03	
6:	$410E	F6	LDAB $3004
	$410F	30	
	$4110	04	
7:	$4111	18	SBA
	$4112	16	
	$4113	7A	STAA $3005
	$4114	30	
	$4115	05	
6:	$4116	20	BRA endmain
	$4117	FE	

Figure 5. Program Code

1. Enter the **Machine codes in** Figure 5 into memory at the addresses shown. You are supposed to use the "**Memory**" pane to enter the machine language code. Clicking the right mouse button over the **Memory** pane will display the pop-up menu. In the pop-up menu, choose **Address….**



And then, **the Display Address** window pops up. Enter the address you want to jump to.

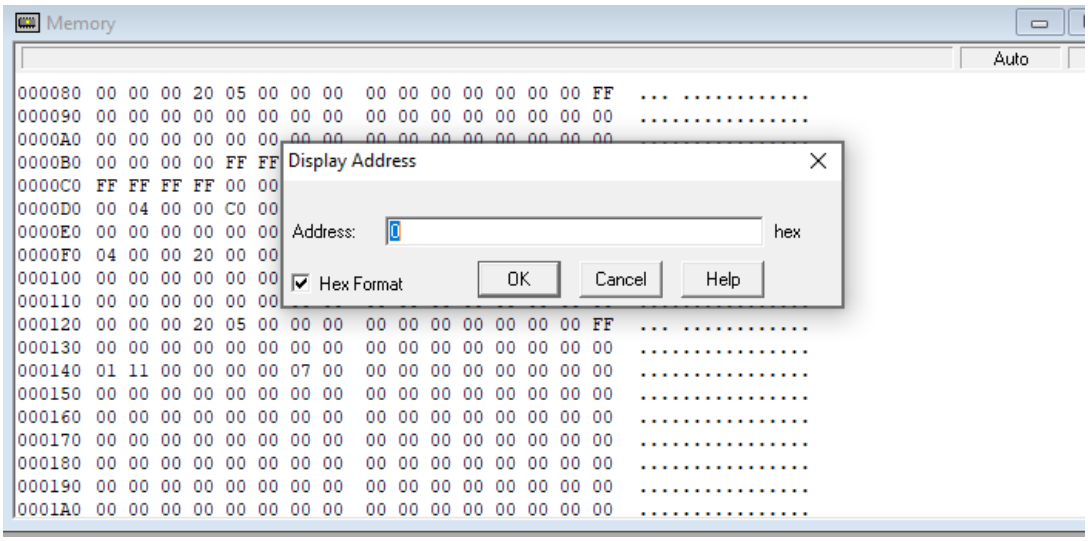


Figure 6. Memory Pane

Also, you can scroll down to the address if you want. Click the position where you want to edit. You will be staying in Edit Mode as long as you are typing. **Use the Tab key** if **you're going to move the following address** without exiting out from Edit Mode**.** Note that as you enter the hex values for the program code, the assembly pane shows the corresponding instructions in an assembly format.

2. Following similar steps, initialize 4 memory locations as shown in Figure 7. These are the data to be used by your instructions:

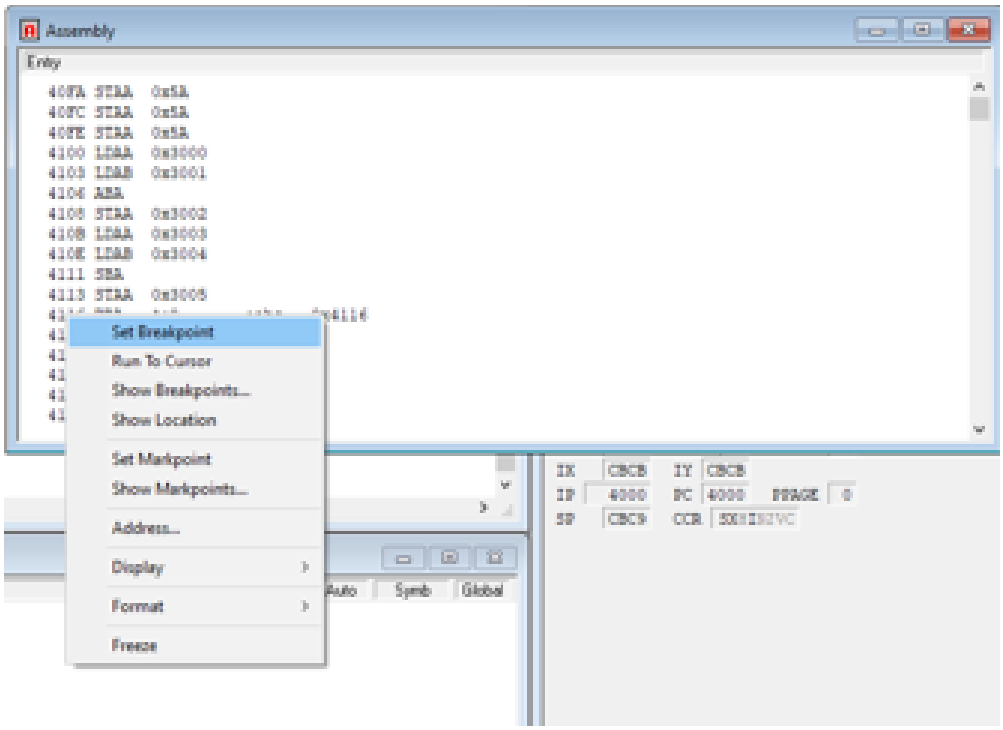
Memory address	Contents
$3000	**$76**
$3001	**$3A**
$3002	--
$3003	**$A3**
$3004	**$6C**
$3005	--
$3006	--
$3007	--
$3008	--

Figure7. Initial Data

3. Check out the **Assembly** pane: The machine code you entered starting at the address $4100 has been disassembled by **CodeWarrior**. In this pane, prefix **0x** signifies base 16. If you can not see the address $4100, click the right mouse button 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.

4. Use CodeWarrior to execute the instructions and 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).**

5. Add **a breakpoint at the final instruction's address "4116**" to stop at a logical point for the user's inspection. In the **Assembly pane,** select the line starting with **4116** that is an address. Then, click the right mouse button to display the pop-up window and choose the 'Set Breakpoint' item.



6. F11 is the single-step button. Use it to execute one instruction at a time. Hit F11, then the first instruction (**LDAA 0x3000**) executes, and the next one (**LDAB 0x3001**) 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 all the instructions once at a time. Before you proceed to the next instruction, look at the simulation results as well as your expectations.

Q2: Do your results calculated by hand in Q1 match the simulation results? Yes No

(Check the memory location $3002 and $3005).

7. Enter the new data set in the memory as shown in the table of Figure 8.

Memory address	Contents
3000h	**$F3**
3001h	**$C9**
3002h	--
3003h	**$4B**
3004h	**$8D**
3006h	--

Figure8. New Data

Q3: Do the following addition and subtraction by hand:

$F3 + $C9 If these numbers are unsigned numbers, **overflow:** yes no
If these numbers are signed numbers, **overflow**: yes no

$4B-$8D If these numbers are unsigned numbers, **overflow:** yes no
If these numbers are signed numbers, **overflow**: yes no

8. You need to reset **the program counter (PC)** in the CPU **Register pane** to $4100, the starting address of your code. Using **F11 key, execute** the instructions one at a time and fill the values of **N, Z, V, and C bits in Condition Coder Register (CCR) in Figure 9.**

Code Line	Memory Address	Machine Codes	Assembly Code	N	Z	V	C
1:	$4100	B6	LDAA 3000h				
	$4101	30					
	$4102	00					
2:	$4103	F6	LDAB 3001h				
	$4104	30					
	$4105	01					
3:	$4106	18	ABA				
	$4107	06					
4:	$4108	7A	STAA 3002h				
	$4109	30					
	$410A	02					
5:	$410B	B6	LDAA 3003h				
	$410C	30					
	$410D	03					
6:	$410E	F6	LDAB 3004h				
	$410F	30					
	$4110	04					
7:	$4111	18	SBA				
	$4112	16					
	$4113	7A	STAA 3005h				
	$4114	30					
	$4115	05					
6:	$4116	20	BRA endmain				
	$4117	FE					

Figure 9.

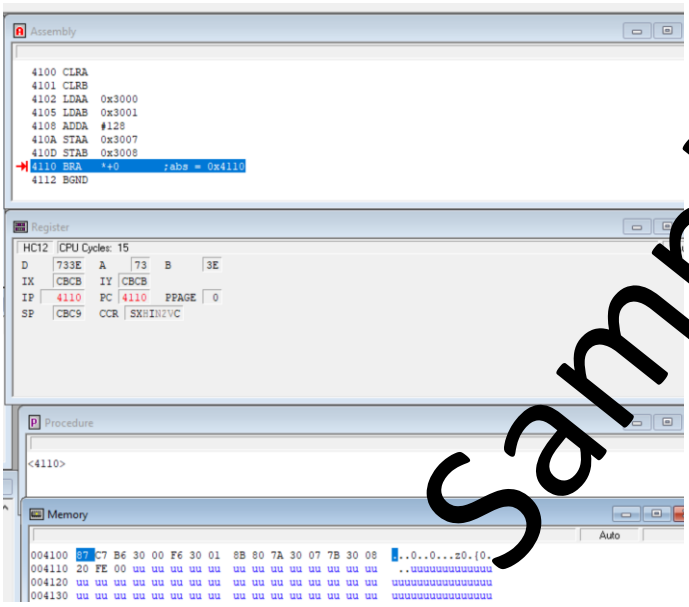
Q4: Do your results calculated by hand in Q3 match the simulation results with the new data?

Yes No

What to Submit through Blackboard

1) **Per Student: one pdf file** that includes :
   - Complete the work book, **Lab1_workbook.doc** (contains the table in Figure 9 and questions)
     - *30 pts: Fill the table in Figure 9*
     - *40pts: Provide answers to all questions: Q1, Q2, Q3, and Q4.*

2) Per Group : 30 pts
Screenshot of *the Assembly pane, the Register pane, and the Memory pane after run the program with the new data as shown below:*



- **Pay attention to the file name convention:**
- **Individual file: Lab1_Student1_Firstnname_Lastname.pdf**
  Ex) Lab1_Smith_Green.pdf

- **Group file: Lab1_Student1 Lastname_Student2 Lastname.pdf**
  Ex) Lab1_Green_Owens.pdf