Lab 1 – Code Entry, Assembly, and Execution

ASSEMBLY PROGRAMMING WITH CODE WARRIOR

The purpose of this lab is to introduce you to the layout and structure of assembly language programs and their format, as well as to the use of the Code Warrior development tool. You will write your own programs later on in the semester with similar structure.

In the following example source code, from left to right, you will notice four columns. The first column contains label names. Leave a space if there is no label. The second column contains assembly opcode. The third column contains data or operands. The fourth column is used for comments. Comments start with semi colon. Your programs should always be in this format.

INTRODUCTION TO Code Warrior V5.1

Code Warrior (a free program from Freescale available on the Internet) is a windows based program, which allows assembly programmer to assemble, debug, and download a program onto the Dragon12 HCS12 board. This lab familiarizes the student with all the steps involved in assembling, running, and debugging assembly language programs using Code Warrior. The student will be required to understand and remember all the steps to download and debug programs in future labs. Please download and install Code warrior V5.1 (Special Edition: CodeWarrior for HCS12(X) Microcontrollers (Classic)) on your PC before coming to the lab.

http://www.freescale.com/webapp/sps/site/overview.jsp?code=CW_SPECIALEDITIONS

If your laptop's operating system is not Windows, e.g., MAC. You should use Windows virtual machine or dual operating systems to use the code warrior.

PROCEDURE:

I. USING CODE WARRIOR TO CREATE A NEW PROJECT

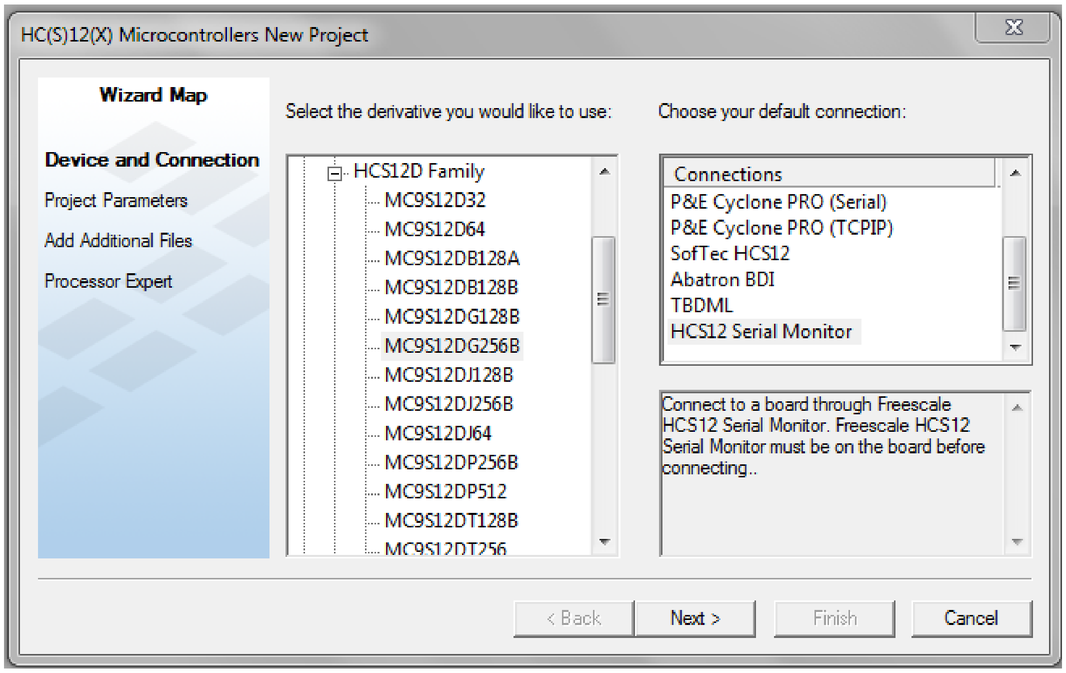
1. Run the code warrior program as follows:

Start → All Programs → Freescale CodeWarrior → CodeWarrior Development Studio for S12(X) V5.1→ Code Warrior IDE

2. Close the "Tip of the Day" window if it comes up, and Click on File → New. This opens the New Project window as shown in Figure 1.

3. From that window select: HCS12 then HCS12D family, then the microcontroller type (MC9S12DG256B) and the default connection (HCS12 serial monitor). Then Click **Next**

4. Change the project name to Lab one and Select 'Absolute Assembly' indicating that you are writing all your assembly code in one single file using fixed addresses, as shown in figure 2. Then click **next**.



HC(S)12(X) Microcontrollers New Project

Wizard Map: **Device and Connection**, Project Parameters, Add Additional Files, Processor Expert

Select the derivative you would like to use:

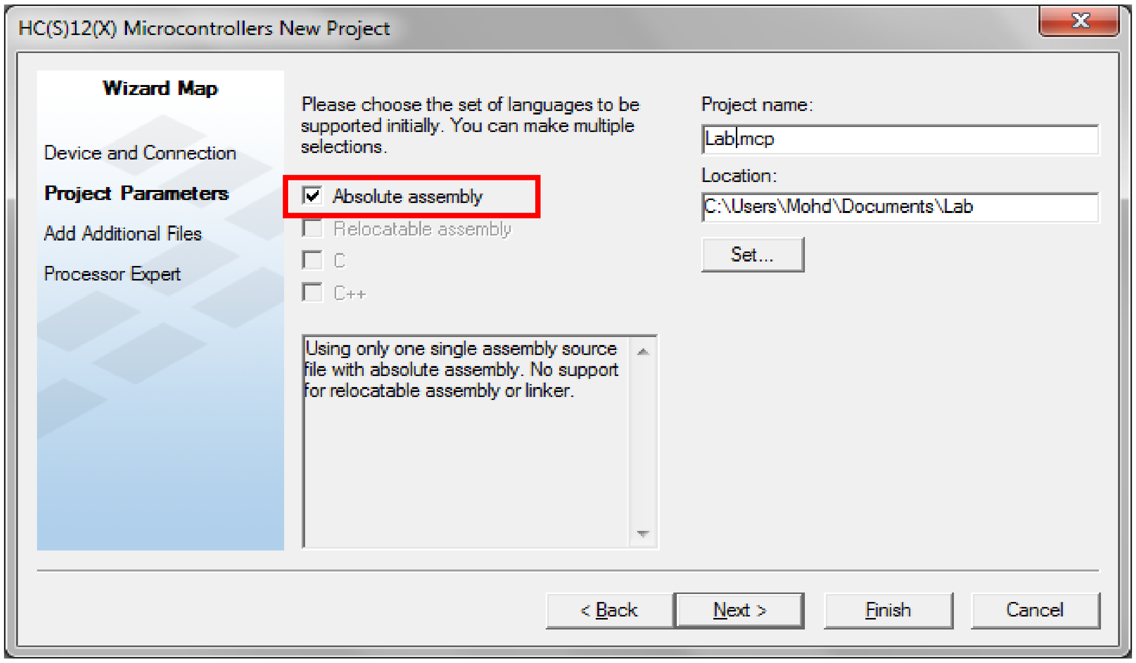
- HCS12D Family
  - MC9S12D32
  - MC9S12D64
  - MC9S12DB128A
  - MC9S12DB128B
  - MC9S12DG128B
  - MC9S12DG256B
  - MC9S12DJ128B
  - MC9S12DJ256B
  - MC9S12DJ64
  - MC9S12DP256B
  - MC9S12DP512
  - MC9S12DT128B
  - MC9S12DT256

Choose your default connection:

Connections
- P&E Cyclone PRO (Serial)
- P&E Cyclone PRO (TCPIP)
- SofTec HCS12
- Abatron BDI
- TBDML
- HCS12 Serial Monitor

Connect to a board through Freescale HCS12 Serial Monitor. Freescale HCS12 Serial Monitor must be on the board before connecting..

Figure 1



HC(S)12(X) Microcontrollers New Project

Wizard Map: Device and Connection, **Project Parameters**, Add Additional Files, Processor Expert

Please choose the set of languages to be supported initially. You can make multiple selections.

- [✓] Absolute assembly
- [ ] Relocatable assembly
- [ ] C
- [ ] C++

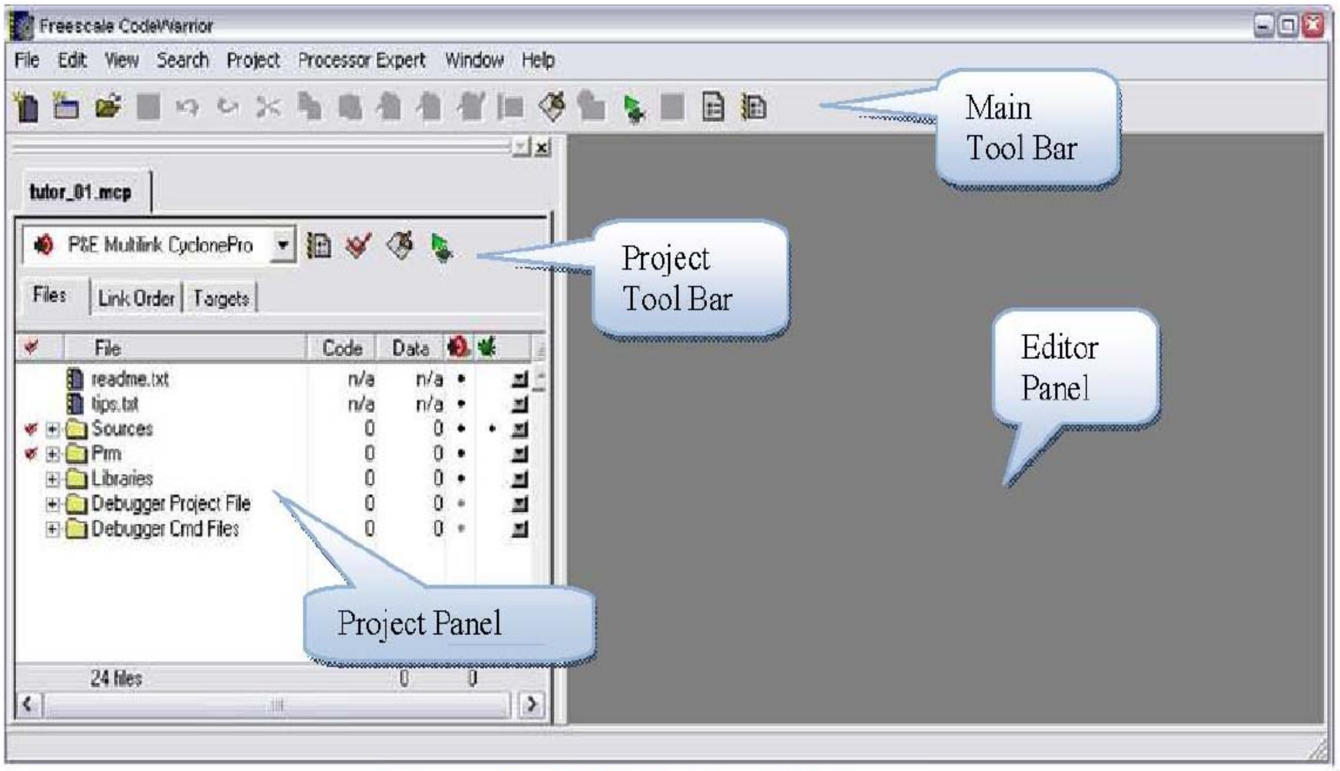
Using only one single assembly source file with absolute assembly. No support for relocatable assembly or linker.

Project name: Lab.mcp

Location: C:\Users\Mohd\Documents\Lab

Figure 2

5. Click on **Finish** to complete the project setup. The resultant screen will be similar to that shown in Figure 3.



File	Code	Data
readme.txt	n/a	n/a
tips.txt	n/a	n/a
Sources	0	0
Prm	0	0
Libraries	0	0
Debugger Project File	0	0
Debugger Cmd Files	0	0
24 files	0	0

Figure 3: Initial Project Window

II. USING CODEWARRIOR TO ENTER THE PROGRAM

There are two methods to enter your code into Code Warrior.

a. Modify the pre-written *main.asm* file or

b. Write code in a new text file, add it to your project, and remove *main.asm* from the project.

1. (Using method a.) Open *main.asm* which is already present in the *Sources* folder of the Project panel. Figure 4 and 5 gives the directions to open the *main.asm* file.

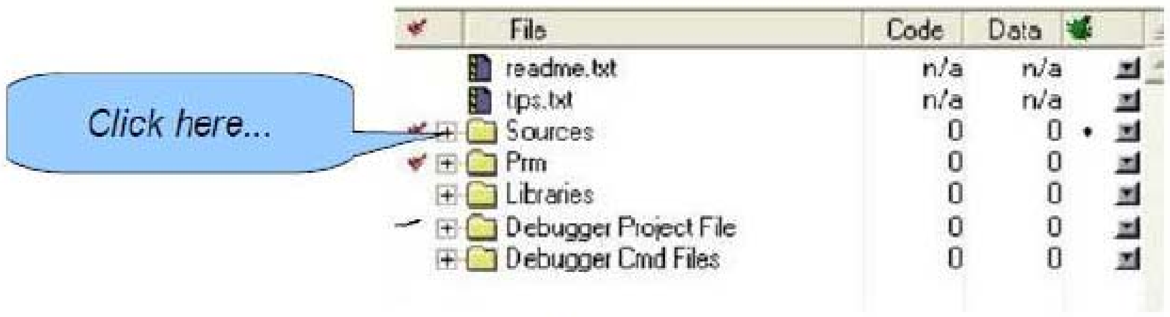


Figure 4

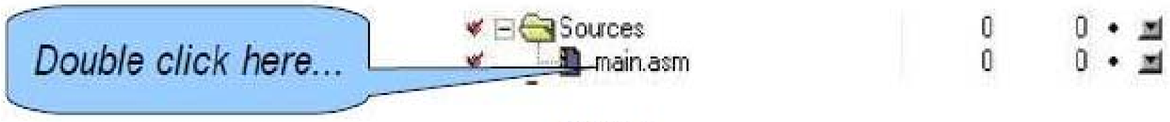


Figure 5

2- Replace the template code in *main.asm* provided by CodeWarrior with the example code shown below and save the program. Keep the columns carefully aligned as shown to enhance reading and understanding! (If you wish, you can use method b. instead).

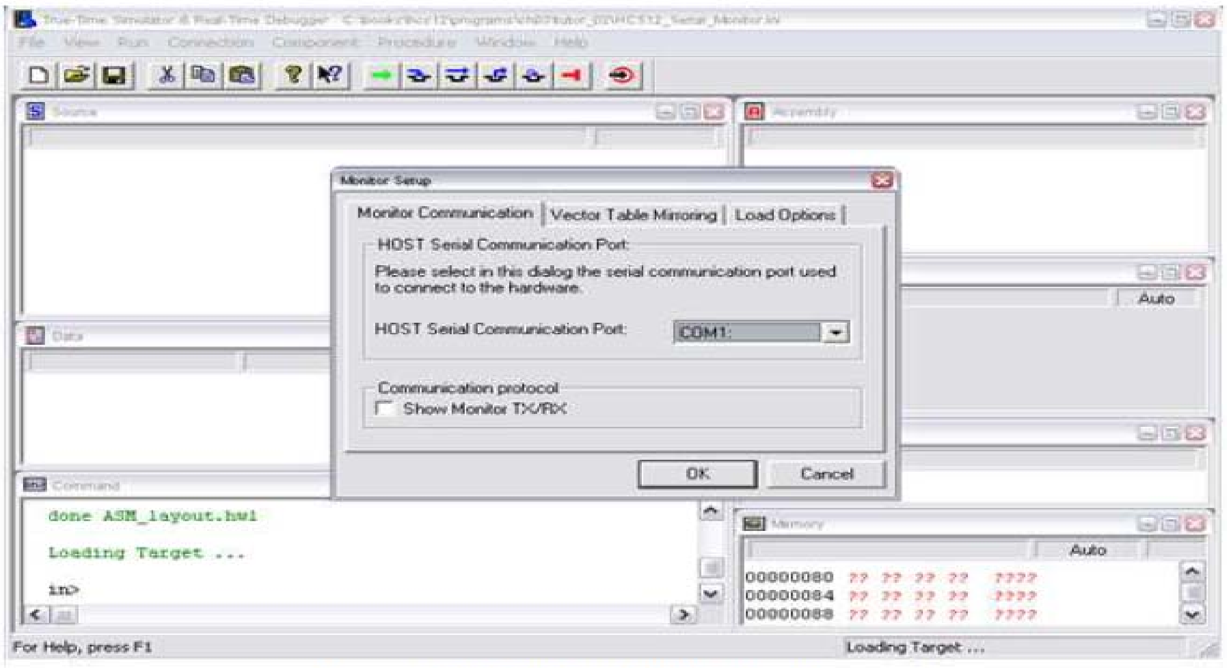
```
;---------------------------------------
;Lab#1
;Code Entry, Assembly, and Execution
;(Put your name and date here)
;---------------------------------------
 absentry Entry ;to indicate the application entry point
 include 'mc9s12dg256.inc'

 org $1000 ;Data starts at RAM address $1000
sum dc.b 0 ;Sum byte stored here

 org $2000 ;Program code starts at address $2000
Entry:
 ldaa #$25 ;Load $25 to reg A
 adda #$34 ;Add $34 to A
 adda #$11 ;Add $11 to A
 adda #18 ;Add $12 to A
 adda #%00011100 ;Add $1C to A
 staa sum ;Store total in 'sum' M[$1000]
here: jmp here ;Stay here forever to end program

;-------End Program Lab#1-------------
```

If there is no label, leave a space otherwise you will get error

Note: The minimal requirements for an absolute assembly program are:
a) The **ABSENTRY** declaration at the top;
b) Inclusion of the ***mc9s12dg256.inc*** file that defines the registers and memory of this particular microcontroller;
c) The **ORG** declarations for the starting memory addresses of the data and code.

III. ASSEMBLING AND RUNNING THE PROGRAM ON THE BOARD

1. Make sure the Dragon12 board is connected to power and to the PC.
2. Hit the **RESET** button on the Dragon12 board. This pushbutton is located close to the middle of the bottom edge of the board, labeled "Reset SW6".
3. **Assemble the code**: From the main menu select Project → Make (F7). This will assemble the project source code into object code and display errors, if any; the screen does not change if there are no errors.
4. **Download and debug the code**: from main menu select Project → Debug (F5). This downloads the user's machine code into the HCS12 microcontroller.
5. When CodeWarrior works with a demo board with the Serial Monitor, the response is similar to that shown in Figure 6.



HOST Serial Communication Port
Please select in this dialog the serial communication port used to connect to the hardware.
HOST Serial Communication Port: COM1:
Communication protocol
Show Monitor TX/RX

```
done ASM_layout.hwl

Loading Target ...

in>
```

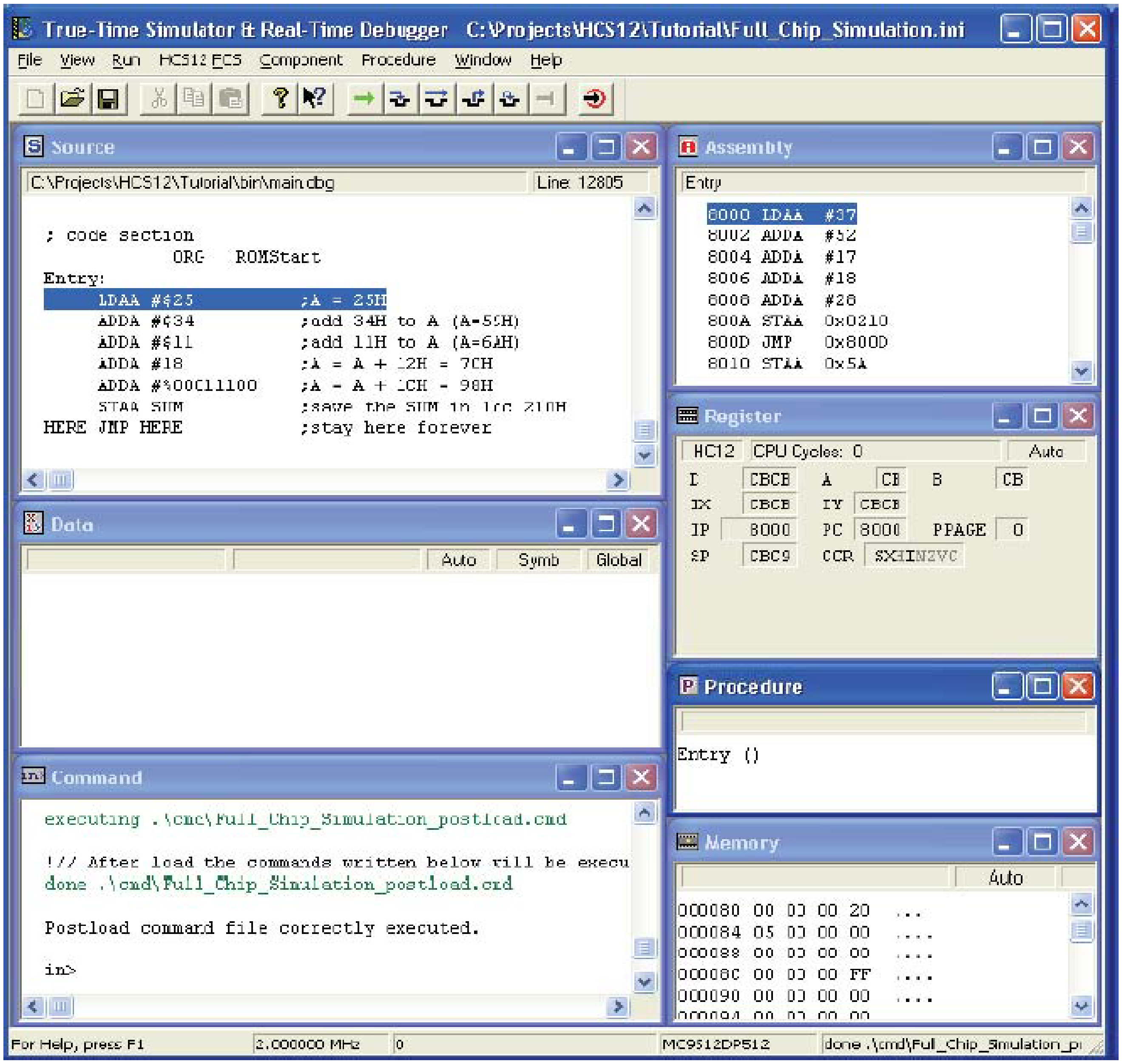
```
00000080 ?? ?? ?? ??   ????
00000084 ?? ?? ?? ??   ????
00000088 ?? ?? ?? ??   ????
```

Loading Target ...

Figure 6: HCS12 Serial download response

6. Select the com number that is assigned to the board. Check your laptop's device manager to know the com number. If the com number is more than 8 you have to change it. To do that right click on the board in the device manger and select properties – select port settings tab – select advanced.

7. After clicking on OK, the screen changes to one similar to Figure 7


True-Time Simulator & Real-Time Debugger C:\Projects\HCS12\Tutorial\Full_Chip_Simulation.ini
File View Run HCS12 FCS Component Procedure Window Help
Source
Assembly
Register
Data
Auto Symb Global
Procedure
Command
Memory

```
C:\Projects\HCS12\Tutorial\bin\main.dbg                Line: 12805

; code section
            ORG   ROMStart
Entry:
     LDAA #$25              ;A = 25H
     ADDA #$34              ;add 34H to A (A=59H)
     ADDA #$11              ;add 11H to A (A=6AH)
     ADDA #18               ;A = A + 12H = 7CH
     ADDA #%00011100        ;A = A + 1CH = 98H
     STAA SUM               ;save the SUM in loc 210H
HERE JMP HERE               ;stay here forever
```

```
Entry
   8000 LDAA   #37
   8002 ADDA   #52
   8004 ADDA   #17
   8006 ADDA   #18
   8008 ADDA   #28
   800A STAA   0x0210
   800D JMP    0x800D
   8010 STAA   0x5A
```

```
HC12   CPU Cycles: 0                Auto
D    CBCB     A    CB      B      CB
IX   CBCB     IY   CBCB
IP   8000     PC   8000    PPAGE  0
SP   CBC9     CCR  SXHINZVC
```

```
Entry ()
```

```
executing .\cmd\Full_Chip_Simulation_postload.cmd

!// After load the commands written below will be execu
done .\cmd\Full_Chip_Simulation_postload.cmd

Postload command file correctly executed.

in>
```

```
                          Auto
000080  00 00 00 20   ...
000084  05 00 00 00   ....
000088  00 00 00 00   ....
00008C  00 00 00 FF   ....
000090  00 00 00 00   ....
000094  00 00 00 00
```

```
For Help, press F1   2.000000 MHz   0   MC9S12DP512   done .\cmd\Full_Chip_Simulation_p
```

Figure 7: Debug mode

Lab 1 – Code Entry, Assembly, and Execution

8. Figure 8 and 9 give details of the debug window and the main features to be used to debug the code.

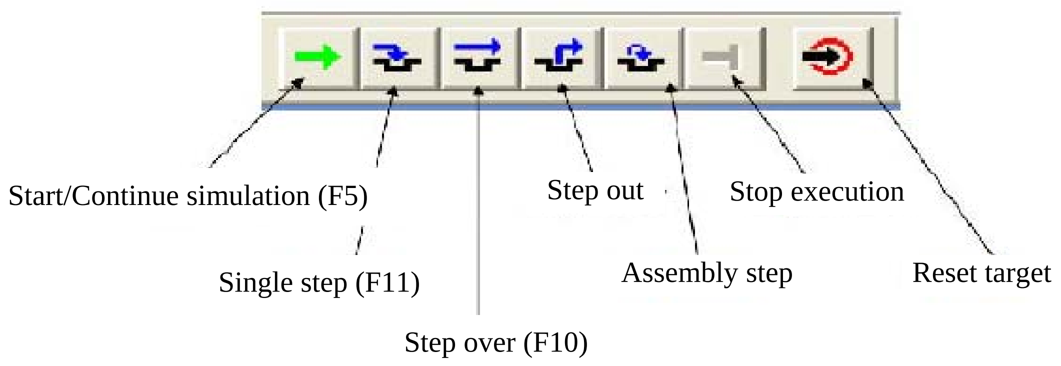


Figure 8

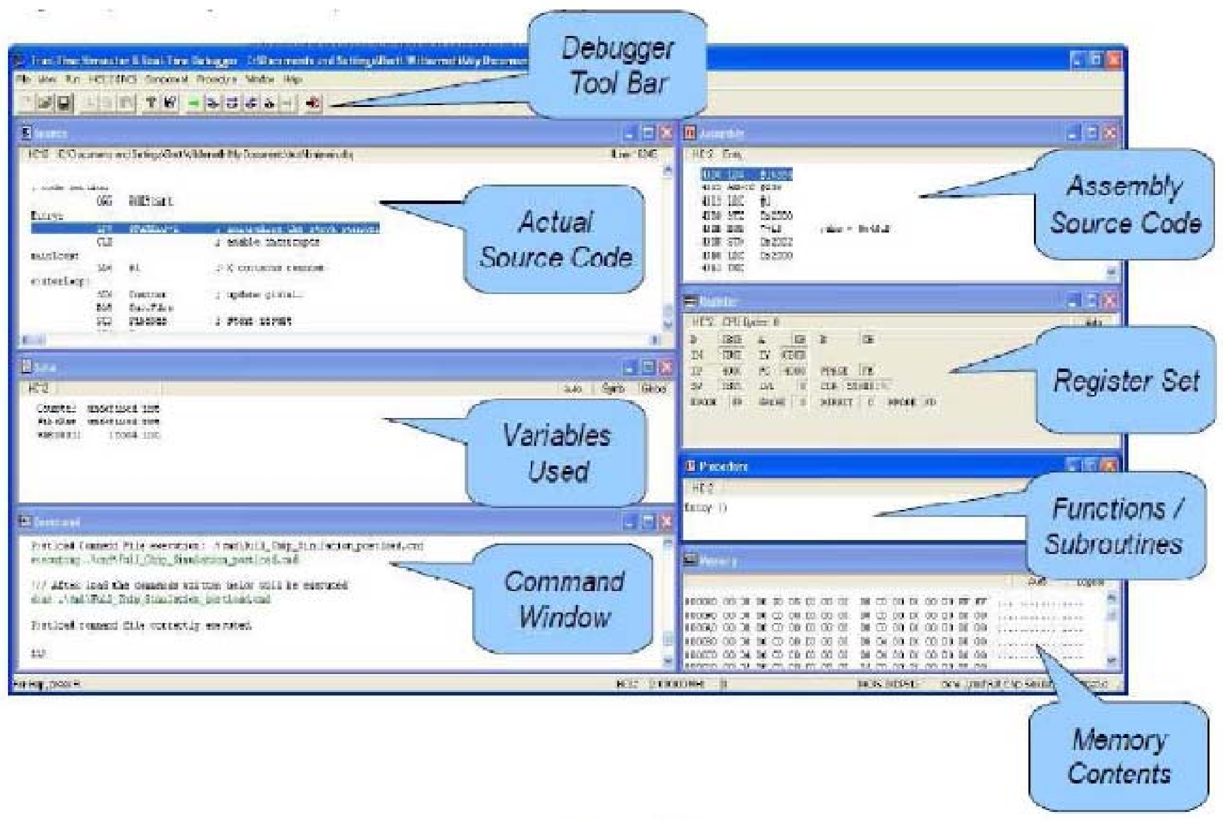


Figure 9

- The Data window shows all variables present in the current source module or procedure

- The Register window displays the names, values and details device registers

- The Memory window displays unstructured memory content, or memory dump, that is, continuous memory words without distinction between variables.

- You can right click on the Data window or Register window to change the data format (binary, decimal, signed, unsigned, character (ASCII)…)

- You can right click on the memory window to go to a specific memory location, to change the memory content, or to change the format.

- Normally, you will run a program at full speed from start to finish by using the 'start/continue simulation' icon or the F5 key. However, for this lab, we want to see what will happen as we execute the program step-by-step.

- Use single step repeatedly to execute the code. ***Observe*** the content of the PC, accumulators A & B, as well as four of the CCR (Condition Code Register) bits after each single step execution. ***Record*** these values in Table 1. Note that the black flag means 1 and the gray one means 0.

Table 1

PC	A	B	[$1000]	N	Z	V	C

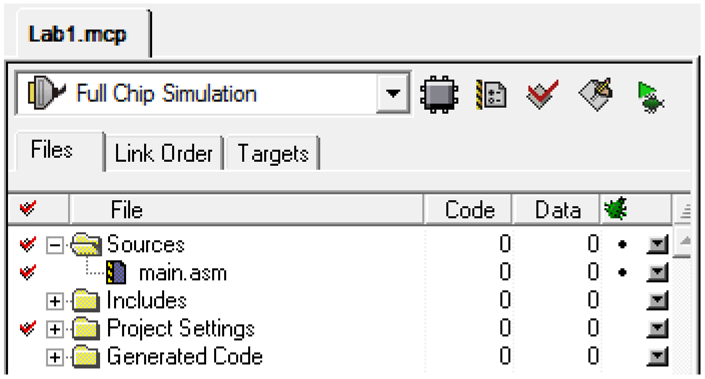
Approve lab TA _________________________

Record the machine code that is equivalent to the assembly program.

Approve lab TA _________________________

IV. ASSEMBLING AND RUNNING THE PROGRAM USING FULL CHIP SIMULATION:

You can use the full chip simulation feature in CODEWARRIOR to debug your code without running it on a board. To do so associated the "Full Chip Simulator" Connection to your project. The fastest way to do that is to select the full chip simulation option from the drop down menu



Full Chip Simulation

File	Code	Data
Sources	0	0
main.asm	0	0
Includes	0	0
Project Settings	0	0
Generated Code	0	0

Figure 10

Things to turn in as your Lab1 Report:

This assignment sheet, with your name at the top, signed by the TA where shown.

A. The answers in Table 1. **[10 marks]**

B. The machine code that is equivalent to the assembly program. **[10 marks]**

C. Answers to the following questions:

1. What was the original content of memory location *sum* after loading but before executing the complete program? Why does it have this value? **[10 marks]**

2. What is the final content of memory location *sum*? **[10 marks]**

3. Explain two different ways to find the content of memory location *sum* using the debug windows. **[10 marks]**

4. After execution of the last adda instruction, explain why the 4 status bits have those particular values. Explain in details **[20 marks]**

Remember:

Z: zero flag it is 1 when the result is zero otherwise it is 0
N: Negative flag it is 1 when the result is negative otherwise it is 0

C: carry flag it is 1 when the there is a carry in the last operation otherwise it is 0
V: overflow flag it is 1 when there is signed overflow (+ve number + +ve number = -ve number) or (-ve number + -ve number = +ve number) otherwise it is 0

5. Explain in details how PC changes in Table 1. Why the increment value is not the same for all the instructions. **[20 marks]**