ECE 3731 Experiment-1
(Software Development Tools)

Lab Instructor: Azeem Hafeez

Email: azeemh@umich.edu

Contents

1 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.

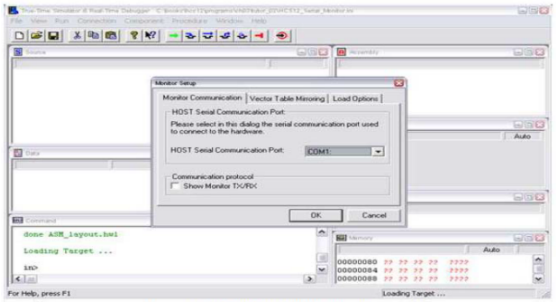
2 Introduction To Code Warrior (using Serial Monitor)

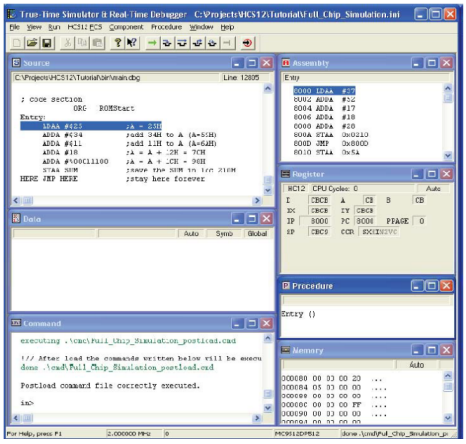
Code Warrior (a free program from Freescale available on the Internet) is a windows based program, which allows assembly programmer to assemble, debug, and download a program onto the Dragon12 HCS12 board. This lab familiarizes the student with all the steps involved in assembling, running, and debugging 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. A sample project file has been created for you. Codewarrior is based on the concept of a collection of files called a "Project".The actual program that is in the project is in the file main.asm. Your instructor has provided a sample project that you can modify the source code in. A condensed version of the source code in main.asm is shown below. Please note that the code must be modified to meet the criterion for a Codewarrior Project in order to run under Codewarrior.(This has been done for you.) So the code will appear somewhat differently in main .asm

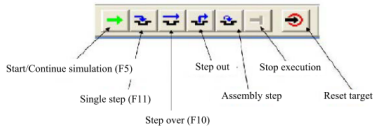
```
;---------------------------------------
;Lab#1
;Code Entry, Assembly, and Execution

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

; Code Section
     org $2000 ;Program code starts at address $2000
      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-------------
```

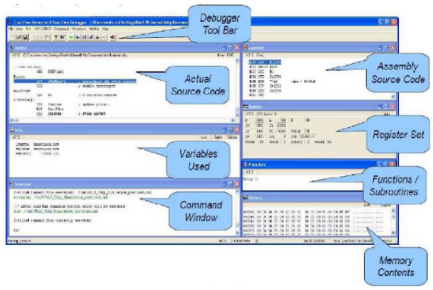
3 Assembling And Running The Program On The Board

1. Make sure the Dragon12 board is connected to power and to the PC. 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”.
2. 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.
3. Download and debug the code: from main menu select Project —Debug (F5). This downloads the user’s machine code into the HCS12 micro-controller.
4. When CodeWarrior works with a demo board with the Serial Monitor, the response is similar to that shown in the Figure. The Data window shows all variables present in the current source module or procedure.
5. The Register window displays the names, values and details device registers.
6. The Memory window displays unstructured memory content, or memory dump, that is, continuous memory words without distinction between variables.
7. 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.
8. 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.
9. Use single step repeatedly to execute the code. Observe the content of the PC, A, B, and CCR bits after each single step execution.
10. Record these values in Table 1. Note that the black flag means 1 and the gray one means 0.






Start/Continue simulation (F5)
Single step (F11)
Step over (F10)
Step out
Assembly step
Stop execution
Reset target


Debugger Tool Bar
Actual Source Code
Assembly Source Code
Register Set
Variables Used
Functions / Subroutines
Command Window
Memory Contents

4 Things to turn in as your Lab1 Report

4.1 Part (a)

Debug the given program and record the values in Table 1.

Table 1

PC	A	B	[$1000]	N	Z	V	C

4.2 Part (b)

Assuming that all the values are in Hexadecimal, rewrite the code so that it calculates the result of:

1B + 2F + 13 -55 -20 and include the re-written code in your report.

4.3 Part (c)

Answers to the following questions: (original program given is being referred to – not Part(b) program)

1. What was the original content of memory location sum after downloading but before executing the complete program? Why does it have this value ?

2. What is the final content of memory location sum?

3. Explain how to find the content of memory location sum using the debug windows.

4. After execution of the last adda instruction, explain why the four status bits Z,N,C,V have those particular values. Explain in detail.

Remember:

Z: Zero flag it is 1 when the result is zero otherwise 0

N: Negative flag it is 1 when the result is negative

C: Carry flag it is 1 when the there is a carry last operation

V: overflow flag it is 1 when there is signed overflow

5.

What are the results of the following arithmetic operations? List the N, Z, V and C bits of the condition codes. Assume that all numbers are in Hex.

Operation	Result and Values of N, Z, V and C bits
a. AC + B5	
b. 7F + BC	
c. FA + AB	

Determine the machine code (in hex) for the following assembly instructions:

Instruction	Machine Code (in Hex)
a. ADDA #$A6	
b. LDAA $0A,X	
c. STAA $2A3B	
d. LDX #$1B2D	
e. LDAB $23	

5 Submission Deadline:

As announced on canvas