ECE 372: Introduction To Microprocessors

Lab 1: Software Development Tools

May 26, 2018

Summer 2018

Honor Code: I have neither given nor received unauthorized assistance on this graded report.

x Brian Vera-Burgos

x Srinivas Simhan

Table of Contents

Objective ---------- Page 3

Equipment ---------- Page 3

Procedure ---------- Page 3

Code ---------- Page 4

Part A ---------- Page 4

Part B ---------- Page 6

Part C ---------- Page 7

Arithmetic Operations ---------- Page 8

Machine Code ---------- Page 8

Conclusion ---------- Page 9

Objective

- To become familiar with CodeWarrior
- To become familiar with basic assembly language
- To become familiar with memory access

Equipment Used

- CodeWarrior

Procedure

- In part A, we opened up CodeWarrior and went over some basic commands. Then we downloaded and ran the example program and ran it to demonstrate hexadecimal and binary arithmetic. In Part B, we made modifications to the code in order to run a different arithmetic problem involving hexadecimal, binary, and decimal numbers. Then we logged the results to an excel spreadsheet. In part C, we answered several questions about our results from part A and part B.

Code

Part A:

```
;---------------------------------------------------------

; Insert your code here

;---------------------------------------------------------

       lds   #ROMStart ; load stack pointer


              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-------------
```

Figure 1:

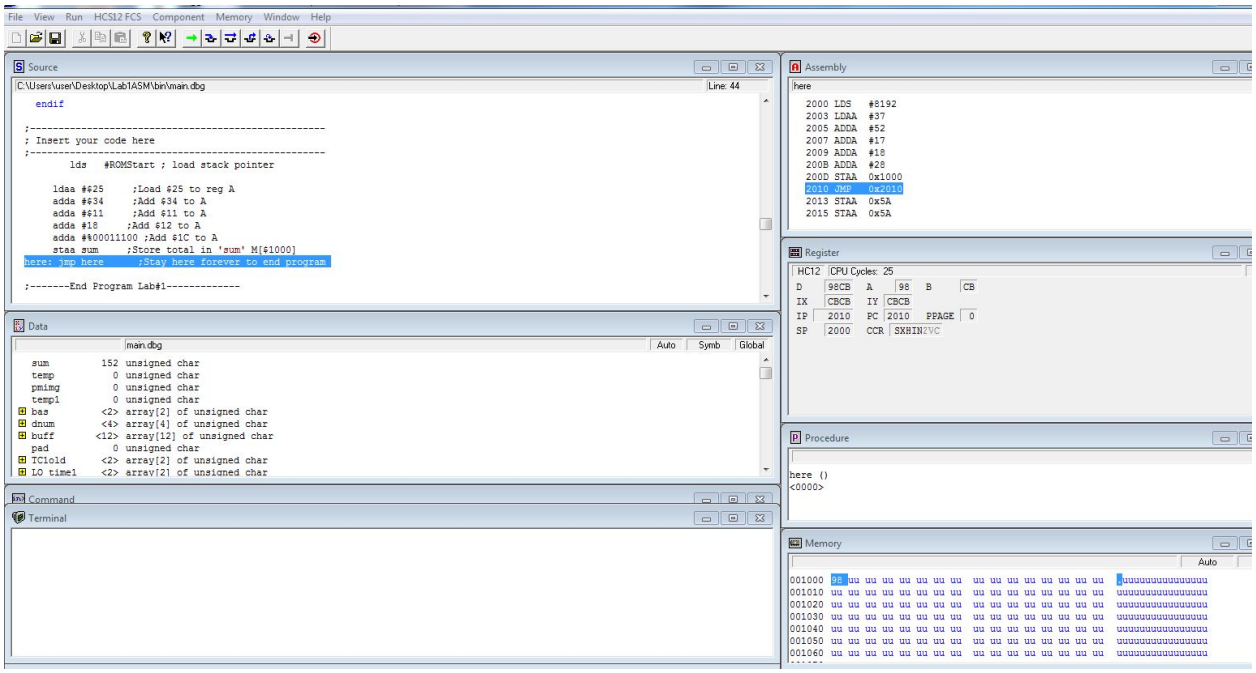

Figure 2:

PC	A	B	[$1000]	N	Z	V	C
2000	CB	CB	0	0	0	0	0
2003	CB	CB	0	0	0	0	0
2005	25	CB	0	0	0	0	0
2007	59	CB	0	0	0	0	0
2009	6A	CB	0	0	0	0	0
200B	7C	CB	0	0	0	0	0
200D	98	CB	0	0	0	0	0
2010	98	CB	98	1	0	0	0

Part B:

```
;---------------------------------------------------------
; Insert your code here
;---------------------------------------------------------
       lds   #ROMStart ; load stack pointer


             ldaa #$1b              ;Load $1b to reg A
             adda #$2f              ;Add $2f to A
             adda #$13              ;Add $13 to A
             adda #$ab              ;Add $ab to A
             adda #e0    ;Add $e0 to A
             staa sum  ;Store total in 'sum' M[$1000]
here: jmp here         ;Stay here forever to end program


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

;The answer is $1B + $2F + $13 - $55 - $22 = $E8

Part C:

1) The original content of the memory sum location is uu. The reason for this is because there is nothing written in the memory location during the initial stages before executing the program.
2) The final memory of memory location sum is $98
3) Open debug with F5 key. Right click on the memory box in the bottom right corner and click address, enter 1000 (to access the address in the memory location of [$1000]. The walk through the program using F10 to step over the code, and at the end when you state to store the value of sum, it will be visible in memory position [$1000].
4) The last addition is $7C + $1C. This converted to binary and evaluated is:

   +0111 1100

   0001 1100

   = 1001 1000

- Z-Flag: Because the number is not 0000 0000 or zero in decimal, the Z flag (zero flag) is set to 0.
- N-Flag: Because the most significant bit is 1, the N flag (negative flag) is set to 1.
- C-Flag: Because the number does not overflow when the two unsigned values are added, the C flag (unsigned overflow) is set to 0.
- V-Flag: Because both numbers have a most significant bit of 0, they are simply added together to show that the value Thus the V flag (signed overflow) is set to 1.

Arithmetic Operations

	Operation	Result	N	Z	V	C
a)	$AC + $B5	= $61	0	0	0	1
b)	$7F + $BC	= $3B	0	0	0	1
c)	$FA + $AB	= $A5	1	0	0	1

Machine Code (in hex)

	Instruction	Machine Code
a)	ADDA #$A6	$8B $A6
b)	LDAA $0A,X	$A6 $0A
c)	STAA $2A3B	$7A $2A $3B
d)	LDX #$1B2D	$CE $1B $2D
e)	LDAB $23	$F6 $01 $23

Conclusion:

In part A, we learned the basics of navigating the CodeWarrior IDE and practiced arithmetic using binary, hexadecimal, and decimal numbers in both signed and unsigned format. In Part B, we practiced writing in assembly language by rewriting the code in part A to solve another arithmetic problem. In Part C, we answered questions regarding the nature of the data we were looking at, and focused on the operation of our status bits and the basic memory location of the CodeWarrior IDE as well as the machine code in notepad.