Week 4 | Lab Assignment | Condition Code Register

September 17, 2023

Timothy McNamara

**Objectives**

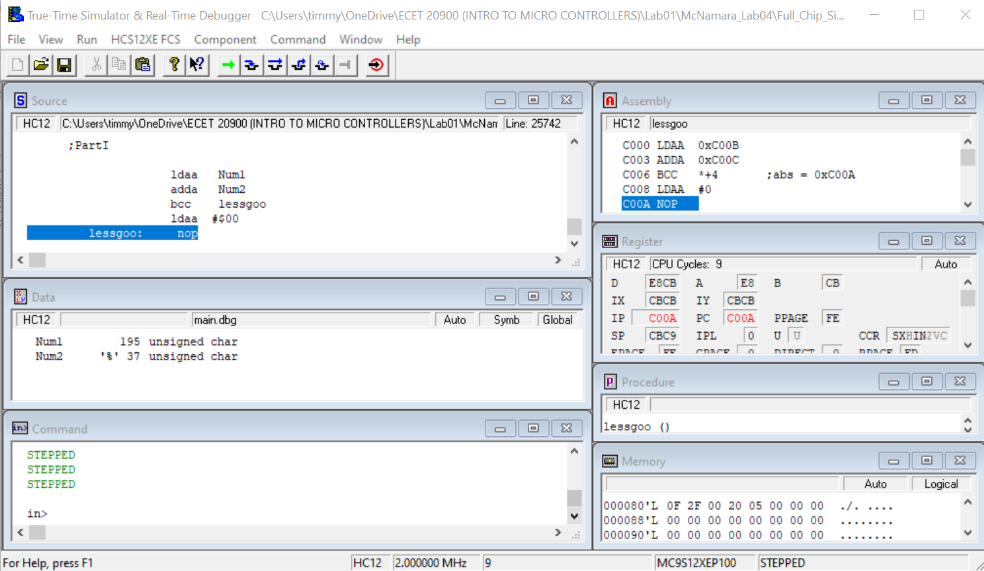
Withing the MC9S12 chip lies the Condition Code Register otherwise known as the CCR. The CCR consists of eight bits that change from a one or a zero depending on whether register A, B, or D was loaded with a value, or a math operation occurred changing the value within the register. For this lab, we focused on the first four bits as the other bits were for higher level programming in assembly. We focused on the first four flags of the register which are labeled as N, Z, V and C. N stands for negative. Z for Zero. O for overflow. C for carry. Utilizing the CCR we can create if then loops or even while loops in assembly and understand how setting each of the register flags works!

**Procedure**

The first step was completing the prelab. By completing the prelab, understanding how each of the four bits works makes the whole lab a breeze. Going through the assignment we were instructed to write lines of code by either copying prewritten code or reading an instruction that tells us to write our own program to achieve a desired result. Writing of code was done swiftly and screenshots were taken when necessary. Coming to the last question, we coded in binary for the first time which made the guesswork of the math operation a breeze. There was some guess work for the lab to further our understanding of the CCR.

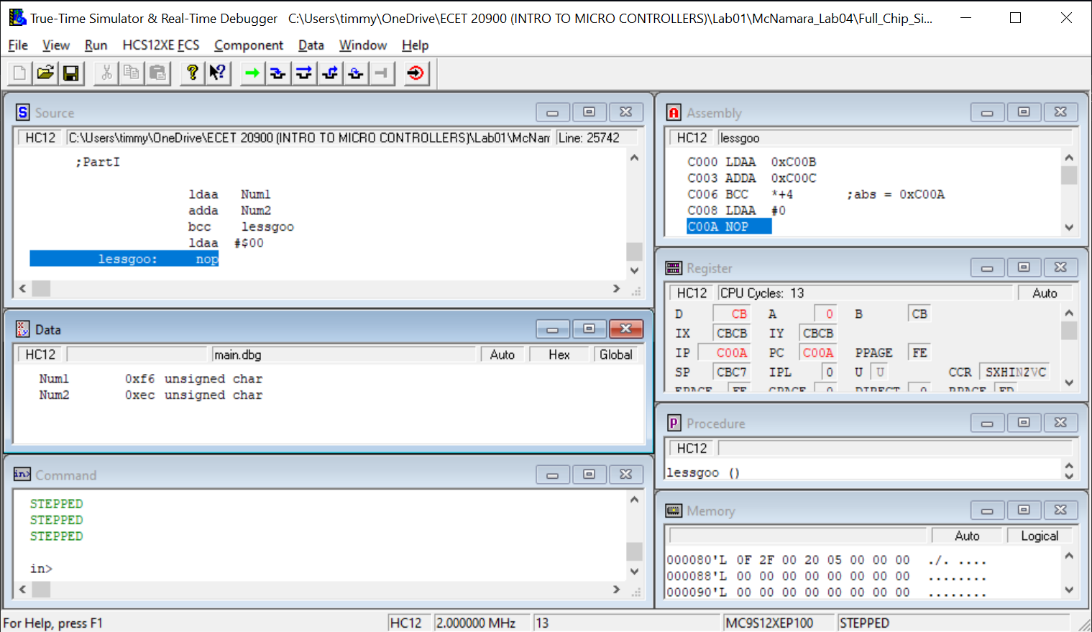
**Analysis (Answers to the questions)**

**Part I**

2a. 

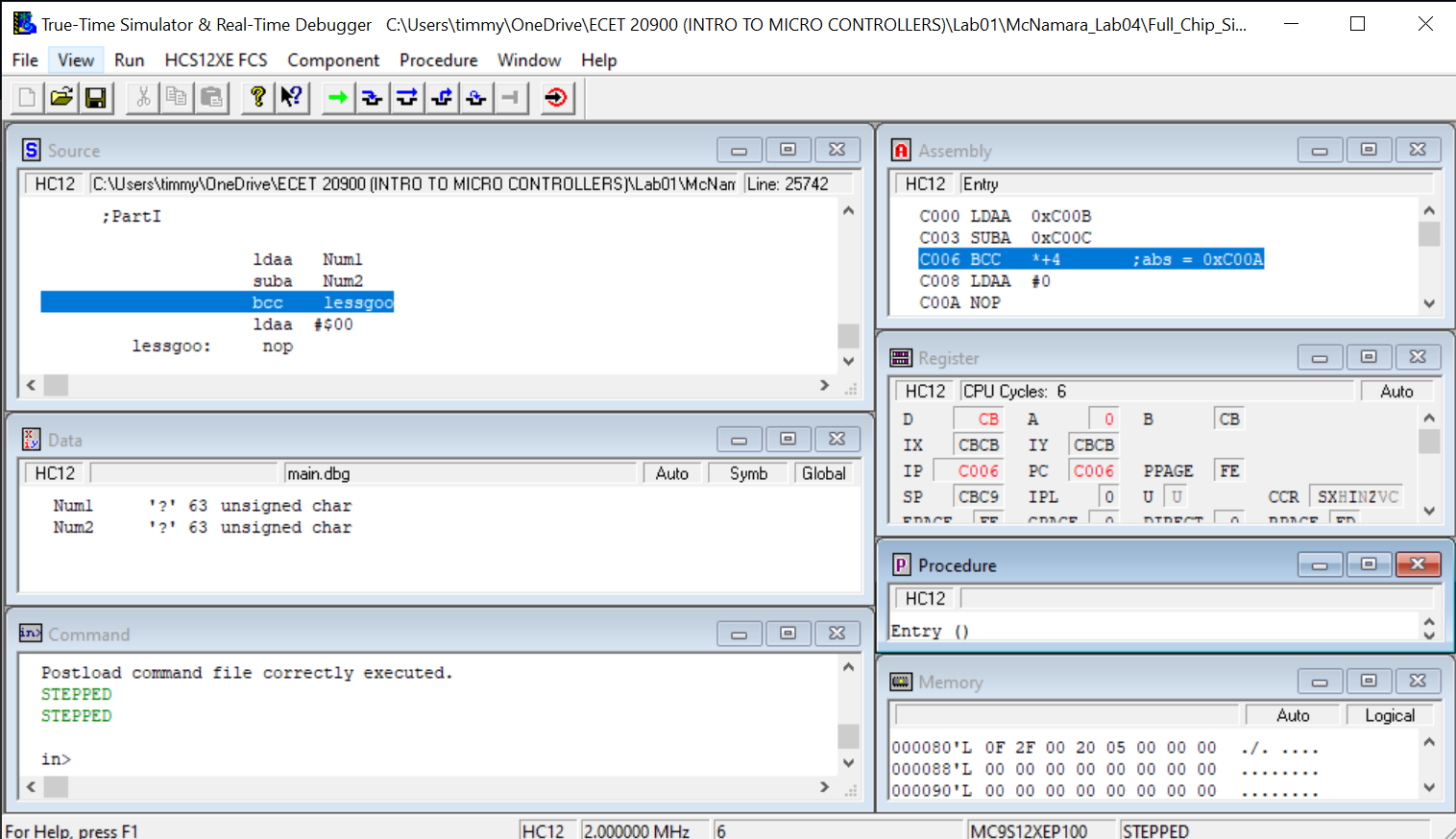
2b. Are your Pre-Lab 04 predicted/calculated answers for 1.a-f the same or different than the values shown in the debugger windows? Explain.

They are the same due to my analysis of the prewritten code to be spot on.

3a. 

3b. Are your predicted/calculated answers for 2.a-f, Pre-Lab 04 the same or different than the values shown in the debugger windows? Explain.

I got the right values. Due to the carry being set and not cleared, it loaded the accumulator with 0.

4a.

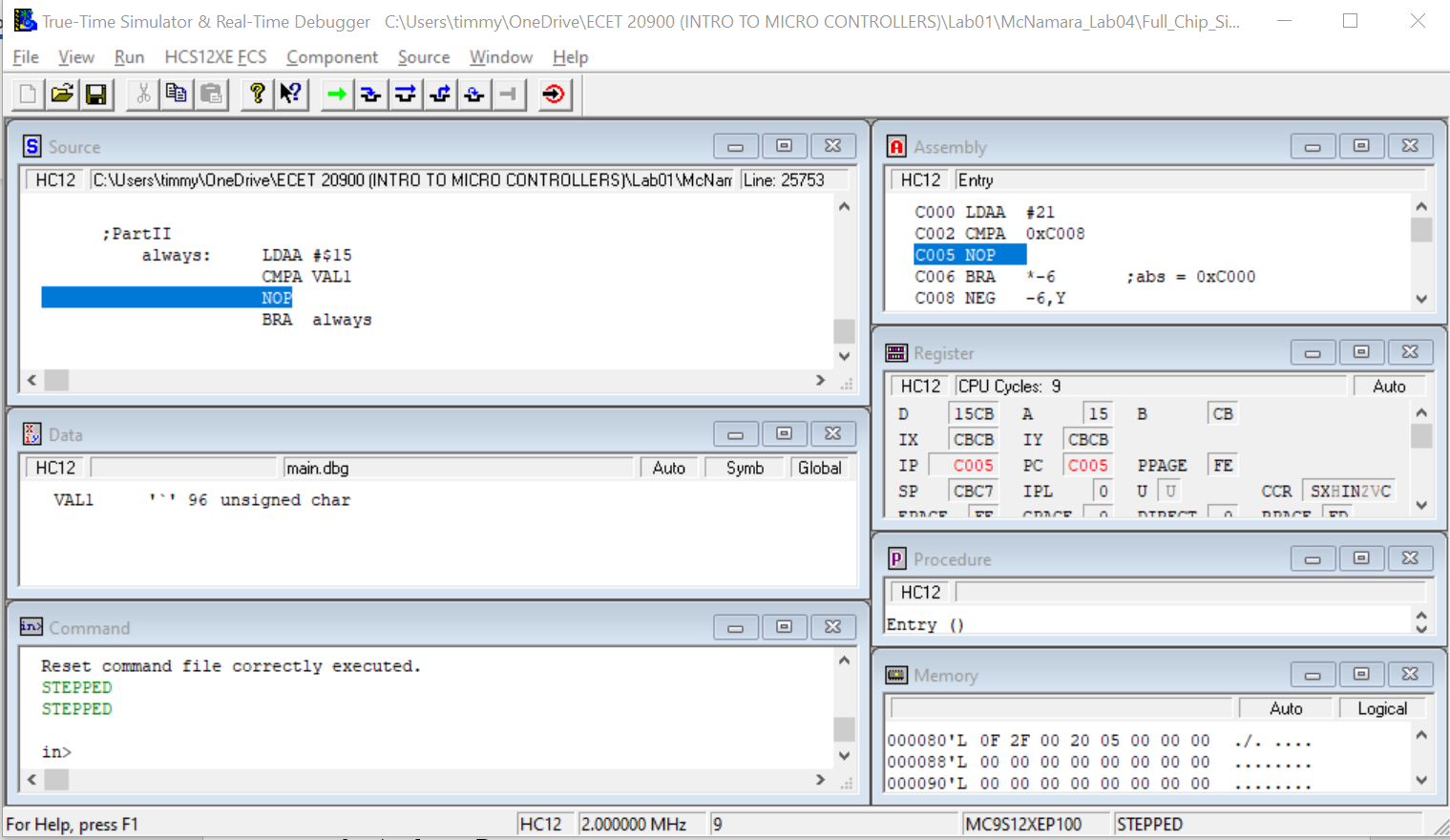
4b. Are your predicted/calculated answers for 2.a-f Pre-Lab 04 the same or different than the values shown in the debugger windows? Explain.

3F -3F does not create a carry therefore, it will branch due to the carry being clear.

**Part II:**

1b.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Observed Values when the program reaches the branch always instruction | | | | |
| Accumulator A | N | Z | V | C |
| 15 | 1 | 0 | 0 | 1 |

1c.

2b. Fill in the table with the machine language for the program. Make sure you include the constants at the end.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C000 | F6 | C0 | 0F | B6 | C0 | 10 | FB | C0 |
| C008 | 11 | 52 | 80 | 04 | 43 | 20 | F1 | C0 |
| C010 | 05 | 3F |  |  |  |  |  |  |

2c. What is the machine code for the instruction INCB? Answer: 52

2d. – 2f.

A screenshot of a document

Description automatically generated

3a. – 3e.

A screenshot of a computer

Description automatically generated

3g. g. Why is it impossible to write a program that results in the N bit, the Z bit, and the C bit all being set to 1 when the program ends?

You can’t have a negative number be zero. There is no possible way for you to have a negative number and for it to be zero.

**Part III:**

1a. A white rectangular box with red text

Description automatically generated

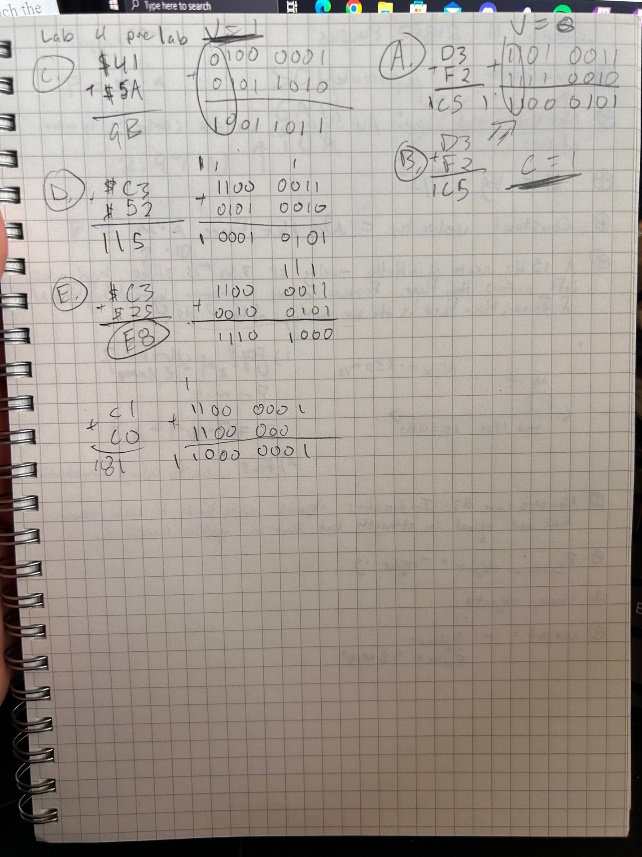
1b. In the table below, without running the program, predict the hex contents of the registers after each instruction is executed.

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | Predicted Values After Instruction Executes | | |
| Program Counter | Accumulator A | Accumulator B |
| LDAA #13 | C002 | 0D | 0 |
| INCA | C003 | E | 0 |
| LDAB #%00100011 | C005 | E | 23 |
| ABA | C007 | 31 | 23 |
| SUBA #$10 | C009 | 21 | 23 |
| BRA ENTRY | C000 | 21 | 23 |

1c. In CodeWarrior IDE editor, comment out the previous code and declarations and enter the program. Then single-step, recording your observed values below.

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | Predicted Values After Instruction Executes | | |
| Program Counter | Accumulator A | Accumulator B |
| LDAA #13 | C002 | 0D | 0 |
| INCA | C003 | E | CB |
| LDAB #%00100011 | C005 | E | 23 |
| ABA | C007 | 32 | 23 |
| SUBA #$10 | C009 | 32 | 23 |
| BRA ENTRY | C000 | 21 | 23 |

3d. Do the observed values agree with your predictions? (Yes or no?) \_\_\_\_\_\_YES\_\_\_\_\_\_ If not, make sure you understand why not.

**** **Observations**

Learning how the 8-bit registers A and B react when it gets a value in hex greater than FF was a key part of the hand calculations for me. Also turning the hex values into binary helps me analyze whether or not overflow occurs.

**Conclusion**

In the prelab, I learned the mnemonics bvc, bvs, bcs, bcc, bzs, bzc, etc. These mnemonics were a key part of understanding how the CCR flags work when doing math operations and loading register a with hex values. Knowing when the flags are set or cleared was critical and I caught on quickly. Part I was just us testing the prelab in the MC9S12 chip simulator. The second part of the lab was analyzing how many of the first four flags could be set at once and how to trigger multiple flags. We also looked at prewritten code and wrote out the machine code for the written code. We also filled out a table with our predictions and actual results of what values Register A or B will hold after an instruction as well as the CCR. Part III was the same. We predicted values, looked at more prewritten code, and wrote out the machine code in the memory window. This was a cool lab and each part of it had its own purpose and reason for research. Every flag value to every screen shot taken had its analysis for the theory behind this lab. While it may be a very full lab. I am prepared to be quizzed over this. This lab helped so much I am that confident.