

**Western Michigan university**

Electrical and Computer Engineering department

**ECE-2510-560-Intro to microprocessor**

**LAB 2**

**Introduction to HCS12 Assembly**

Kaushik Mojumder and Sutopa Suhrid

Name of Laboratory Instructor- Husam Beitello  
Day/Time of your lab section- 2.30 PM-5.10 PM, Thursday

Date Laboratory was Performed-September 16,2021

Table of Contents

**PROCEDURES3**

Task 13

Task 211

Task 315

Task 422

Task4a……………………………………………………………………………………22

Task4b……………………………………………………………………………...…….27

Task4c……………………………………………………………………………...…….35

Task4d………………………………………………………………………………...….38

**Analysis and Conclusion………………………………………………………………43**

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

* 1. ***Task 1:***

Below we can see the addressing mode for different instructions and their Register transfer level.

|  |  |  |  |
| --- | --- | --- | --- |
| # | Instruction | Addressing mode | RTL |
|  | CLRA | Inherent mode | Clear accumulator A |
|  | LDAA #$27 | Immediate mode | [A]🡨[$27] |
|  | LDAB #$F5 | Immediate mode | [B]🡨[F5] |
|  | ABA | Inherent mode | [A]🡨[A]+[B] |
|  | STAA $1000 | Extended mode | M[$1000]🡨[A] |
|  | STAB $1001 | Extended mode | M[$1001]🡨[B] |
|  | LDX $1000 | Extended mode | X🡨M[$1000]=1CF5 |
|  | LDAA 0, X | Indexed addressing mode constant offset | X=M[1CF5]  A🡨M[1CF5] = 16 |
|  | LDAA 1, X+ | Auto post increment indexed addressing mode | [A]🡨M[1CF5] = 16  [X]=M[1CF6] |
|  | LDAB -1, X | Indexed addressing mode constant offset | [X]=M[1CF6]  [B]🡨M[1CF5] = 16 |
|  | LDAA 2, X- | Auto post decrement indexed addressing mode | [A]🡨M[1CF6] = A4  [X]=M[1CF4] |

A new project ‘lab2-task1’ was created and the sample Fibonacci values are cleared.

Then we compiled the program and found no errors.

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

A picture containing table

Description automatically generated

Figure: default code to calculate Fibonacci values

We wrote the following code under endless loop.

LDAA #$27

LDAB #$F5

ABA

STAA $1000

STAB $1001

LDX $1000

LDAA 0,X

LDAA 1,X+

LDAB -1,X

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

LDAA 2,X-

Text

Description automatically generated with medium confidence

Figure: assembly code for task 1

After clicking on make the following message was displayed.

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, text, application, Word

Description automatically generated

Figure: message for lab 2 task 1

We connected the DRAGON12 Plus-2 to upload the bootloader after resetting it.. Then pressed the debug button. We set a breakpoint to the source code section and iterated step by step.

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

A screenshot of a computer

Description automatically generated with medium confidence

Figure: LDAA #$27

Graphical user interface, application

Description automatically generated

Figure: LDAB #$F5

Graphical user interface, application

Description automatically generated

Figure: ABA

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application

Description automatically generated

Figure: STAA $1000

Graphical user interface, application

Description automatically generated

Figure: STAB $1001

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application

Description automatically generated

Figure: LDX $1000

Graphical user interface, application, Word

Description automatically generated

Figure: LDAA 0,X

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application, Word

Description automatically generated

Figure: LDAA 1,X+

Graphical user interface, application, Word

Description automatically generated

Figure: LDAB -1,X

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application, Word

Description automatically generated

Figure: LDAA 2, X-

* 1. ***Task 2:***

All the additions are done in hexadecimal base system.

The following part was included from pre-lab.

LDAA #$00; [A] = 00

ADDA #$27; [A]=[A] + 00 = 27

ADDA #$45; [A] = [A]+ 45 = 27 + 45 = 6C

ADDA #4; [A] = [A]+4 = 6C + 4 =70

ADDA #%0101111; 01011111 = 5F in hex. [A]= [A] + 5F = 70 + 5F = CF

ADDA #$31; [A] = [A] + 31 = 100. So, A = [00]

So,

00+27=27

27 + 45 = 6C

6C + 4 =70

01011111 = 5F

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

70 + 5F = CF

CF + 31 = 100

A new project ‘lab2-task1’ was created and the sample Fibonacci values are cleared.

Then we compiled the program and found no errors. We wrote the following code under endless loop.

LDAA #$00

ADDA #$27

ADDA #$45

ADDA #4

ADDA #%01011111

ADDA #$31

A picture containing text

Description automatically generated

Figure: code for task 2

After clicking the make option, the code was successfully compiled.

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application

Description automatically generated

Figure: messages for lab 2 task 2

We invoked the bootloader for DRAGON-Plus 2 and pressed the reset button. Then we debugged the program and set a break point in order to iterate step by step.

Graphical user interface, application, Word

Description automatically generated

Figure: LDAA #00

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, text, application, Word

Description automatically generated

Figure: ADAA #$27

Graphical user interface, text, application

Description automatically generated

Figure: ADAA #$45

Graphical user interface, text, application

Description automatically generated

Figure: ADAA #4

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application

Description automatically generated

Figure: ADDA #%01011111

Graphical user interface, application, Word

Description automatically generated

Figure: ADDA #$31

* 1. ***Task 3:***

We created a new project ‘lab 2 task 3’ and deleted the default Fibonacci code. Then we compiled the code without any errors. Below we pasted the code from prelab:

LDAA #55

LDAB #75

ABA

LDAB #41

SBA

STAA $1100

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Then we compiled the code and got the successfully compile message in the dialogue box.

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, text, application

Description automatically generated

Figure: Task 3 assembly code

55 = #$ 37

75 = #$4B

55 + 75 = #$37 + #$4B = 130 = #$82

41 = #$29

130 – 41 = 89= #$82 - #$29 = $59

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

A 🡨 [37]

[B] 🡨 [4B]

A 🡨 [A] + [B] = [37] + [4B] = [82]

[B] 🡨 [29]

[A] 🡨 [A] - [B] = [82] – [29] = [59]

M [$1100] 🡨 [A] = [59]

Stop

We uploaded the bootloader into DRAGON12-Plus 2 and debugged the code.

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application

Description automatically generated

Figure: LDAA #55

Graphical user interface, application, Word

Description automatically generated

Figure: LDAB $75

Graphical user interface, application, Word

Description automatically generated

Figure: ABA

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application

Description automatically generated

Figure: LDAB #41

Graphical user interface, application

Description automatically generated

Figure: SBA

Graphical user interface, application, Word

Description automatically generated

Figure: STAA $1100

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

* 1. ***Task 4: (part a)***

For part 1, we created a new project and erased the default Fibonacci code, Then the following code was added under endless loop. The code was compiled and the message was shown.

LDAA #165

LDAB #164

STAA $1200

STAB $1201

LDAA #32

LDAB #10

STAA $1202

STAB $1203

Then the bootloader was invoked and DRAGON12 plus 12 board was reset. Then we debugged the program.

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application

Description automatically generated

Figure: code for task 4 part 1

Graphical user interface, application, Word

Description automatically generated

Figure: LDAA #165

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application, Word

Description automatically generated

Figure: LDAB #164

Graphical user interface, application, Word

Description automatically generated

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Figure: STAA $1200

Graphical user interface, application

Description automatically generated

Figure: STAB $1201

Graphical user interface, application, Word

Description automatically generated

Figure: LDAA #32

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application, Word

Description automatically generated

Figure: LDAB #10

Graphical user interface, application, Word

Description automatically generated

Figure: STAA $1202

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application, Word

Description automatically generated

Figure: STAB $1203

* 1. ***Task 4 (part B):***

For part 2, we created a new project and erased the default Fibonacci code, Then the following code was added under endless loop. The code was compiled, and the message was shown.

LDAA #165

LDAB #164

STAA $1200

INC $1200

STAB $1201

INC $1201

LDAA #32

LDAB #10

STAA $1202

INC $1202

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

STAB $1203

INC $1203

A picture containing text

Description automatically generated

Figure: code for task 4 part 2

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Then the bootloader was invoked and DRAGON12 plus 12 board was reset. Then we debugged the program.

Graphical user interface, application

Description automatically generated

Figure: LDAA #165

Graphical user interface, application, Word

Description automatically generated

Figure: LDAB #164

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application, Word

Description automatically generated

Figure: STAA $1200

Graphical user interface, application

Description automatically generated

Figure: INC $1200

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application

Description automatically generated  
 figure: STAB $1201

Graphical user interface, application, Word

Description automatically generated

Figure: INC 1201

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application

Description automatically generated

Figure: LDAA #32

Graphical user interface, application

Description automatically generated

Figure: LDAB #10

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application

Description automatically generated

Figure: STAA $1202

Graphical user interface, application

Description automatically generated

Figure: INC $1202

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application

Description automatically generated

Figure: STAB $1203

Graphical user interface, application, Word

Description automatically generated

Figure: INC $1203

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

* 1. ***Task 4(part C):***

For part 3, we created a new project and erased the default Fibonacci code, Then the following code was added under endless loop. The code was compiled, and the message was shown.

LDAA #165

LDAB #164

STAA $1200

LDX #$1200

INC 0, X

STAB $1201

LDX #$1201

INC 0, X

LDAA #32

LDAB #10

STAA $1202

LDX #$1202

INC 0, X

STAB $1203

LDX #$1203

INC 0, X

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, text, application, Word

Description automatically generated

Figure: task 4 part c output

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Text

Description automatically generated

Figure: task 4 part c source code

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, text, application

Description automatically generated

Figure: task 4 part c output

* 1. ***Task 4 (part d):***

For part 3, we created a new project and erased the default Fibonacci code, Then the following code was added under endless loop. The code was compiled, and the message was shown.

LDAA #165

LDAB #164

STAA $1200

LDY #$1200

INC 1,Y+

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

STAB $1201

LDY #$1201

INC 1,Y+

LDAA #32

LDAB #10

STAA $1202

LDX #$1202

INC 1,Y+

STAB $1203

LDX #$1203

INC 1,Y+

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Text

Description automatically generated

Figure: task 4 part d assembly code

Then the bootloader was invoked and DRAGON12 plus 12 board was reset. Then we debugged the program.

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Text, letter

Description automatically generated

Figure: task 4 part d source code

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

Graphical user interface, application, Word

Description automatically generated

Figure: task 4 part d memory output

Lab2: Introduction to HCS12 assembly Thursday, 2.30 PM

1. **Analysis and conclusions:**

In this lab, we observed various addressing mode. We were able to manipulate the values in CPU registers and memory addresses through these addressing modes. We also performed various operations such as addition and subtraction. However, there are various limitations to these operations. As for example, for task 3, I decided to subtract the value of Register A from register B using the instruction ‘SAB’. Even if the code makes sense, it is not hardwired to the HCS12 microprocessor.

Again, by using INC instruction and the indexed addressing mode, not only we can manipulate the address that X and Y index register refers to but also, we can change the values in the memory address.

Extended mode can select any values in the 64 kb range. But to cover the range of $0000 and $00FF, direct mode is used since it has a faster response time.

By using the immediate mode, the registers and the memory can take any input directly. We must keep in mind that the values in the CPU registers and the memory addresses belong are all hexadecimal values. Even if the input is decimal or binary, the corresponding hexadecimal value will be stored in the accumulators and the memory addresses.