**MPLAB Introduction and PIC24 Assembly Language**

**4th Laboratory Report for ECE 383**

**Microcomputers**

**Submitted by**

**Yichen Huang**

**11906882**

**Shomari Thomas**

**A picture containing object

Description automatically generated11672867**

**The University of Alabama**

**Tuscaloosa, Alabama 35487**

**February 12, 2020**

# Table of Contents

[Table of Contents - 2 -](#_Toc33035113)

[Abstract - 3 -](#_Toc33035114)

[Introduction - 4 -](#_Toc33035115)

[Pre-Lab - 4 -](#_Toc33035116)

[Task 1- MPLAB Introduction - 4 -](#_Toc33035117)

[Task 2 - myadd.s - 5 -](#_Toc33035118)

[Procedure/Results - 6 -](#_Toc33035119)

[Task 3 - mysub.s - 6 -](#_Toc33035120)

[Task 4 - mylogicops.s - 6 -](#_Toc33035121)

[Conclusion - 7 -](#_Toc33035122)

[Appendixes - 7 -](#_Toc33035123)

[Appendix A – Task I - 7 -](#_Toc33035124)

[Appendix B – Task II - 12 -](#_Toc33035125)

[Appendix C – Task III - 18 -](#_Toc33035126)

[Appendix D – Task IV - 24 -](#_Toc33035127)

[Appendix E – Manual Scrips - 33 -](#_Toc33035128)

# Abstract

Lab 4 was an introduction to basic PIC24 assembly language and a means of practice using the MPLAB Integrated Development Environment (IDE). During this lab we used MPLAB to simulate the PIC24 assembly language program in a project, as well as implemented simple programming tasks using PIC24 assembly language. These tasks were functional practice in becoming more familiar with the MPLAB environment.

For Task 1, we followed the step-by-step instructions provided in the lab document to activate the MPLAB Simulator and watch variable values, special function register values, special function register, memory locations, and window locations. For Task 2, we watched variables aa, bb, lsp, msp, and sum and their corresponding memory locations when space is reserved for variable sum to hold the sum of lsp and msp. For task 3, we wrote an assembly language program that implemented the C program provided in the lab document monitored the memory locations corresponding to variables i, j, k, l, m, xx, and yy. Similar to task 3, task 4 consisted of we write an assembly language program that implemented another C program provided in the lab document and monitoring the memory locations corresponding to

variables u16\_x, u8\_a, u8\_b, u8\_c, u8\_d, u8\_e, and u8\_f.

In Lab 4, we became familiar with the MPLAB environment by translating C programs into program assembly language and completing simulations of these programs that show the relationships between corresponding changes in data location and data memory for variables.

# Introduction

In Lab 4, we were introduced to the basic PIC24 assembly language and the MPLAB Integrated Development Environment (IDE). The PIC24 assembly language was introduced by translating each line of a C program into the corresponding assembly language program. The MPLAB Integrated Development Environment (IDE) was used as a helpful tool to simulate the assembly language program and provide the results in the form of registers and values. Additionally, an assembly language program was downloaded onto a PIC24 device, giving a physical example of the capabilities of the assembly language program and their application in the appropriate environments. This lab exemplified the sensitivity of simulation to different values (hexadecimal in this case) and showed how we can use the PIC24 assembly language and MPLAB Integrated Development Environment together to solve problems.

# Pre-Lab

## Task 1- MPLAB Introduction

For Task 1 of Lab 4, we first moved the files in C:\microchip\chap3\ to our custom directory. In the MPLAB IDE we opened the “mpst\_word.mcp” project and selected the PIC24HJ128GP502 device. After assembling the project, we scrolled through the program memory window to find our program in memory. Then, we opened the file registers window to view the data memory where our variables are listed, and the special function registers window. Lastly, we opened the watch window, and after adding the SFR symbol we were able to watch variable values and special function register values of the i, j, k variables and the W0 special function register. We used this information to apply it to the MPLAB Simulator. In the simulator, we were able to watch and correlate the changing values of the memory and watch window locations with the instructions causing the changes. Additionally, we changed the avalue equate to be the last four digits of your student ID, 6882. The program was assembled and simulated once again to represent this. Code, Figure and flag result is in [Appendix A](#_Appendix_I).

## Task 2 - myadd.s

For Task 2 of Lab 4, we used a given C program to execute assembly instructions that created changes in data memory and memory locations of variables. After removing all instructions from mov #avalue, W0 through mov WREG, k, we started using the myadd.s file as a start for a new program. Next, we converted the number 11906882 into an eight-digit hexadecimal number. Using the C code provided in the lab document, we wrote a program to add the four digit hex number formed by the last four digits of the student ID number (6882) to the four-digit hex number formed by the first four digits of the student ID number (1190). We did so by translating the given C program into the appropriate assembly instructions line-by-line, reserving space for the lsp and msp variable to hold the hex values. Lastly, we opened the watch window, and watched the variable values of aa, bb, lsp, msp, and sum. As in task 1, we used the resulting information to apply it to the MPLAB Simulator. In the simulator, we were able to watch and correlate the changing values of the memory and watch window locations with the instructions causing the changes. Code, figures and flag result will be shown in [Appendix B](#_Appendix_II) .

# Procedure/Results

## Task 3 - mysub.s

For task 3, we created a new project named mysub. We then wrote an assembly language program corresponding to the C program provided in the lab document and used the last 6 digits of the student ID number (906882). Next we opened the watch window and watched the variable values of i, j, k, l, m, xx, and yy, with i, j, k, l, m being 8-bit variables and xx and yy being 16-bit variables. As in the previous tasks, we used the resulting information to apply it to the MPLAB Simulator. In the simulator, we were able to watch and correlate the changing values of the memory and watch window locations with the instructions causing the changes. Code, figures and flag result will be shown in [Appendix C](#_Appendix_III). The flag table in this section will be shown in [Appendix E](#_Flag_Table_in).

## Task 4 - mylogicops.s

In the 4th and final task of lab 4, we created a new project named mylogicops. We then wrote an assembly language program corresponding to the C program provided in the lab document. Next we opened the watch window and watched the variable values of u16\_x, u8\_a, u8\_b, u8\_c, u8\_d, u8\_e, and u8\_f. We then used the MPLAB Simulator to simulate our program. In the simulator, we were able to watch and correlate the changing values of the memory and watch window locations with the instructions causing the changes. Following our simulation, we downloaded our program onto the PIC24 device and showed our demonstration to the TA. Code, figures and flag result will be shown in [Appendix D](#_Appendix_IV)

# Conclusion

We are now confident in our ability to translate C code into its corresponding assembly language program and complete simulations and device downloads using MPLAB. In addition, we were successful in our simulations for tasks 1, 2, 3, and 4, as well as the program download onto the PIC24 device. The results of simulations provided the expected flag values and memory locations of the values. This lab provided an introduction to the PIC24 assembly language and MPLAB and allowed for us to be confident in their use for testing components in future labs.

# Appendixes

## Appendix A – Task I

### Code for “mptst\_word.s”

; Just check out MPLAB

.include "p24Hxxxx.inc"

.global \_\_reset ;The label for the first line of code.

.bss ;unitialized data section

;;These start at location 0x0800 because 0-0x07FF reserved for SFRs

i: .space 2 ;Allocating space (in bytes) to variable.

j: .space 2 ;Allocating space (in bytes) to variable.

k: .space 2 ;Allocating space (in bytes) to variable.

;..............................................................................

;Code Section in Program Memory

;..............................................................................

.text ;Start of Code section

\_\_reset: ; first instruction located at \_\_reset label

mov #\_\_SP\_init, w15 ;Initalize the Stack Pointer

mov #\_\_SPLIM\_init,W0

mov W0, SPLIM ;Initialize the stack limit register

;\_\_SP\_init set by linker to be after allocated data

;User Code starts here.

; C Program equivalent

; #define avalue 2047

; uint16\_t i,j,k;

;

; i = avalue; /\* myvalue = 2047 (0x7FF) \*/

; i = i + 1; /\* i++, i = 2048 (0x800) \*/

; j = i; /\* j is 2048 (0x0800) \*/

; j = j - 1; /\* j--, j is 2047 \*/

; k = j + i; /\* k = 4095 (0x0FFF) \*/

.equ avalue, 6882

;i = avalue; /\* myvalue = 6882 \*/

mov #avalue, w0 ; w0 = 6882 (w0 is wreg)

mov wreg,i ; i = 6882

; i = i + 1;

inc i ; i = i + 1(i = 6882 + 1 = 6883)

; j = i

mov i,wreg ; w0 = i

mov wreg,j ; j = w0 = 6883

; j = j - 1; /\* j--, j is 100 \*/

dec j ; j= j - 1(j = 6883 - 1 = 6882)

; k = j + i

mov i,wreg ; w0 = i

add j,wreg ; w0 = i+j (w0 =6883 + 6882 = 13765)

mov wreg,k ; k = w0

done:

goto done ;Place holder for last line of executed code

.end ;End of program code in this file

### Manual Calculation for instruction and flags [[1]](#footnote-1)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruct | Variables[[2]](#footnote-2) | | | Resisters[[3]](#footnote-3) | Flags | | | |
| Instruction | i | j | k | W0 | N | OV | Z | C |
| mov #avalue, w0 | - | - | - | 6882 | 0 | 0 | 0 | 0 |
| mov wreg, i | 6882 | - | - | 6882 | 0 | 0 | 0 | 0 |
| inc i | 6883 | - | - | 6882 | 0 | 0 | 0 | 0 |
| mov i, wreg | 6883 | - | - | 6883 | 0 | 0 | 0 | 0 |
| mov wreg, j | 6883 | 6883 | - | 6883 | 0 | 0 | 0 | 0 |
| dec j | 6883 | 6882 | - | 6883 | 0 | 0 | 0 | 1 |
| mov i, wreg | 6883 | 6882 | - | 6883 | 0 | 0 | 0 | 1 |
| add j, wreg | 6883 | 6882 | - | 13765 | 0 | 0 | 0 | 0 |
| mov wreg, k | 6883 | 6882 | 13765 | 13765 | 0 | 0 | 0 | 0 |

### Figures(1 - 5)

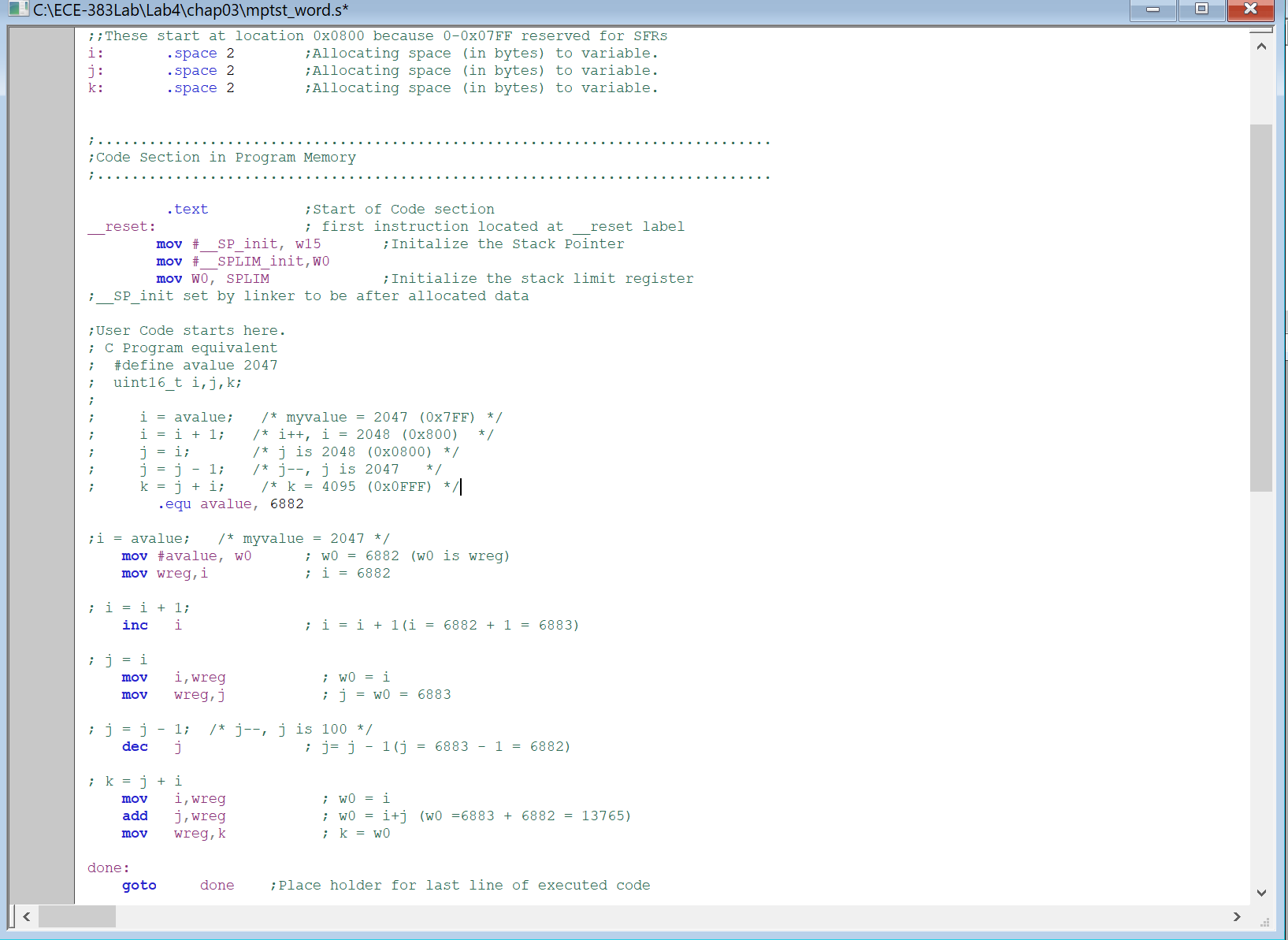


Figure 1 . mptst\_word.s

A screenshot of a social media post

Description automatically generated

Figure 2. File Registers

A screenshot of a social media post

Description automatically generated

Figure 3. Watch Window

A screenshot of a social media post

Description automatically generated

Figure 4. Program Memory

A screenshot of a cell phone

Description automatically generated

Figure 5. Special Function Register

## Appendix B – Task II

### Code for “myadd.s”

;

; Just check out MPLAB

.include "p24Hxxxx.inc"

.global \_\_reset ;The label for the first line of code.

.bss ;unitialized data section

;;These start at location 0x0800 because 0-0x07FF reserved for SFRs

aa: .space 1 ;Allocating space (in bytes) to variable.

bb: .space 1 ;Allocating space (in bytes) to variable.

lsp: .space 2 ;Allocating space (in bytes) to variable.

msp: .space 2

sum: .space 2

.text ;Start of Code section

\_\_reset: ; first instruction located at \_\_reset label

mov #\_\_SP\_init, w15 ;Initalize the Stack Pointer

mov #\_\_SPLIM\_init,W0

mov W0, SPLIM ;Initialize the stack limit register

;\_\_SP\_init set by linker to be after allocated data

;User Code starts here.

; C Program equivalent

; #define avalue 2047

; uint8 aa=100, bb=22;

; uint16 lsp, msp, sum;

; lsp = 0xY3Y2Y1Y0; // Four digits of CWID treated as hex

; msp = 0xY7Y6Y5Y4; // Four digits of CWID treated as hex

; sum = lsp + msp;

; sum = sum + aa + bb;

;

.equ avalue, 6882

mov #0x6882 ,w0; w0 = 0x6882

mov wreg, lsp; lsp = 0x6882

mov #0x1190 ,w0; w0 = 0x1190

mov wreg, msp; msp = 0x1190

mov.b #0x64, w0; w0.LSB = 100

mov.b wreg aa; aa = 100

mov.b #0x16, w0; w0.LSB = 22

mov.b wreg, bb; bb = 22

mov msp, wreg; w0 = msp

add lsp, wreg; w0 = lsp + msp(w0 = 26754 + 4496 = 31250)

mov wreg, sum; sum = w0

mov.b aa, wreg; w0.LSB = aa

ze w0, w1; w1 = aa

mov.b bb, wreg; w0.LSB = bb

ze w0, w0; w0 = bb

add w0, w1, w0; w0 = aa + bb(w0 = 100 + 22 = 122)

add sum; sum = aa + bb + sum (sum = 122 + 31250 = 31372)

done:

goto done ;Place holder for last line of executed code

.end ;End of program code in this file

### Manual Calculation for instruction and flags[[4]](#footnote-4)

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruct | Variables[[5]](#footnote-5) | | | | | Registers[[6]](#footnote-6) | | Flags | | | |
| Instruction | aa | bb | lsp | msp | sum | W0 | W1 | N | OV | Z | C |
| mov #0x6882 ,w0 | - | - | - | - | - | 26754 | - | 0 | 0 | 0 | 0 |
| mov wreg, lsp | - | - | 26754 | - | - | 26754 | - | 0 | 0 | 0 | 0 |
| mov #0x1190 ,w0 | - | - | 26754 | - | - | 4496 | - | 0 | 0 | 0 | 0 |
| mov wreg, msp | - | - | 26754 | 4496 | - | 4496 | - | 0 | 0 | 0 | 0 |
| mov.b #0x64, w0 | - | - | 26754 | 4496 | - | 100 | - | 0 | 0 | 0 | 0 |
| mov.b wreg, aa | 100 | - | 26754 | 4496 | - | 100 | - | 0 | 0 | 0 | 0 |
| mov.b #0x16, w0 | 100 | - | 26754 | 4496 | - | 22 | - | 0 | 0 | 0 | 0 |
| mov.b wreg, bb | 100 | 22 | 26754 | 4496 | - | 22 | - | 0 | 0 | 0 | 0 |
| mov msp, wreg | 100 | 22 | 26754 | 4496 | - | 4496 | - | 0 | 0 | 0 | 0 |
| add lsp, wreg | 100 | 22 | 26754 | 4496 | - | 31250 | - | 0 | 0 | 0 | 0 |
| mov wreg, sum | 100 | 22 | 26754 | 4496 | 31250 | 31250 | - | 0 | 0 | 0 | 0 |
| mov.b aa, wreg | 100 | 22 | 26754 | 4496 | 31250 | 100 | - | 0 | 0 | 0 | 0 |
| ze w0, w1 | 100 | 22 | 26754 | 4496 | 31250 | 100 | 100 | 0 | 0 | 0 | 1 |
| mov.b bb, wreg | 100 | 22 | 26754 | 4496 | 31250 | 22 | 100 | 0 | 0 | 0 | 1 |
| ze w0, w0 | 100 | 22 | 26754 | 4496 | 31250 | 22 | 100 | 0 | 0 | 0 | 1 |
| add w0, w1, w0 | 100 | 22 | 26754 | 4496 | 31250 | 122 | 100 | 0 | 0 | 0 | 0 |
| add sum | 100 | 22 | 26754 | 4496 | 31372 | 122 | 100 | 0 | 0 | 0 | 0 |

### Figures(6-10)

A screenshot of a cell phone

Description automatically generated

Figure 6. myadd.s

A screenshot of a cell phone

Description automatically generated

Figure 7. Watch Window

A close up of text on a white background

Description automatically generated

Figure 8. Program Memory

A screenshot of a cell phone

Description automatically generated

Figure 9. File Registers

A screenshot of a social media post

Description automatically generated

Figure 10. Special Function Registers

## Appendix C – Task III

### Code for “mysub.s”

; Just check out MPLAB

.include "p24Hxxxx.inc"

.global \_\_reset ;The label for the first line of code.

.bss ;unitialized data section

;;These start at location 0x0800 because 0-0x07FF reserved for SFRs

xx: .space 2 ;Allocating space (in bytes) to variable.

yy: .space 2

i: .space 1 ;Allocating space (in bytes) to variable.

j: .space 1 ;Allocating space (in bytes) to variable.

k: .space 1

l: .space 1

m: .space 1

.text ;Start of Code section

\_\_reset: ; first instruction located at \_\_reset label

mov #\_\_SP\_init, w15 ;Initalize the Stack Pointer

mov #\_\_SPLIM\_init,W0

mov W0, SPLIM ;Initialize the stack limit register

;\_\_SP\_init set by linker to be after allocated data

;User Code starts here.

; C Program equivalent

; #define avalue 2047

; uint16 xx=0xDEAD, yy=0xBEEF;

; uint8 i, j, k, l, m;

; i = Y1Y0 82; j = Y3Y2 68; k = Y5Y4 90;

; l = i + k

; m = j – l

; xx=xx-yy-m;

; 11906882

mov #0xDEAD, w0; w0 = 0xDEAD

mov wreg, xx; xx = w0

mov #0xBEEF, w0; w0 = 0xBEEF

mov wreg, yy; yy = w0

;11906882

mov.b #0x52, w0; w0 = 0x52

mov.b wreg, i; i = 0x52

mov.b #0x44, w0; w0 = 0x44

mov.b wreg, j; j = 0x44

mov.b #0x5A, w0; w0 = 0x5A

mov.b wreg, k; k = 0x5A

add.b i, wreg; w0 = k + i

mov.b wreg, l; l = k + i (l = 90 + 82 = 172)

sub.b j, wreg; w0 = j - (k+i)

mov.b wreg, m; m = j-l (m = 68 - 172 = 0b 1001 1000 = 0x98 = 152)

mov.b m, wreg; w0.LSB = m

ze w0, w1; w1 = m

mov yy, wreg; w0 = yy

sub xx, wreg; w0 = xx - yy = (57005 - 48879 = 8126)

sub w0, w1, w0; w0 = xx - yy - m

mov wreg, xx; xx = w0(xx = 8126 - 152 = 7974)

done:

goto done ;Place holder for last line of executed code

.end ;End of program code in this file

### Manual Calculation for instruction and flags[[7]](#footnote-7)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruct | Variables[[8]](#footnote-8) | | | | | | | Registers[[9]](#footnote-9) | | Flags | | | |
| Instruction | i | j | k | l | m | xx | yy | W0 | W1 | N | OV | Z | C |
| mov #0xDEAD, w0 | - | - | - | - | - | - | - | 57005 | - | 0 | 0 | 0 | 0 |
| mov wreg, xx | - | - | - | - | - | 57005 | - | 57005 | - | 0 | 0 | 0 | 0 |
| mov #0xBEEF, w0 | - | - | - | - | - | 57005 | - | 48879 | - | 0 | 0 | 0 | 0 |
| mov wreg, yy | - | - | - | - | - | 57005 | 48879 | 48779 | - | 0 | 0 | 0 | 0 |
| mov.b #0x52, w0 | - | - | - | - | - | 57005 | 48879 | 82 | - | 0 | 0 | 0 | 0 |
| mov.b wreg, i | 82 | - | - | - | - | 57005 | 48879 | 82 | - | 0 | 0 | 0 | 0 |
| mov.b #0x44, w0 | 82 | - | - | - | - | 57005 | 48879 | 68 | - | 0 | 0 | 0 | 0 |
| mov.b wreg, j | 82 | 68 | - | - | - | 57005 | 48879 | 68 | - | 0 | 0 | 0 | 0 |
| mov.b #0x5A, w0 | 82 | 68 | - | - | - | 57005 | 48879 | 90 | - | 0 | 0 | 0 | 0 |
| mov.b wreg, k | 82 | 68 | 90 | - | - | 57005 | 48879 | 90 | - | 0 | 0 | 0 | 0 |
| add.b i, wreg | 82 | 68 | 90 | - | - | 57005 | 48879 | 172 | - | 1 | 1 | 0 | 0 |
| mov.b wreg, l | 82 | 68 | 90 | 172 | - | 57005 | 48879 | 172 | - | 1 | 1 | 0 | 0 |
| sub.b j, wreg | 82 | 68 | 90 | 172 | - | 57005 | 48879 | 152 | - | 1 | 1 | 0 | 0 |
| mov.b wreg, m | 82 | 68 | 90 | 172 | 152 | 57005 | 48879 | 152 | - | 1 | 1 | 0 | 0 |
| mov.b m, wreg | 82 | 68 | 90 | 172 | 152 | 57005 | 48879 | 152 | - | 1 | 1 | 0 | 0 |
| ze w0, w1 | 82 | 68 | 90 | 172 | 152 | 57005 | 48879 | 152 | 152 | 0 | 1 | 0 | 1 |
| mov yy, wreg | 82 | 68 | 90 | 172 | 152 | 57005 | 48879 | 48779 | 152 | 1 | 1 | 0 | 1 |
| sub xx, wreg | 82 | 68 | 90 | 172 | 152 | 57005 | 48879 | 8126 | 152 | 0 | 0 | 0 | 1 |
| sub w0, w1, w0 | 82 | 68 | 90 | 172 | 152 | 57005 | 48879 | 7974 | 152 | 0 | 0 | 0 | 1 |
| mov wreg, xx | 82 | 68 | 90 | 172 | 152 | 7974 | 48879 | 7974 | 152 | 0 | 0 | 0 | 1 |

### Figures(11 - 15)

A screenshot of a cell phone

Description automatically generated

Figure 11. mysub.s

A screenshot of a social media post

Description automatically generated

Figure 12. Watch Window

A screenshot of a cell phone

Description automatically generated

Figure 13. File Register

A screenshot of a cell phone

Description automatically generated

Figure 14. Special Function Register

A screenshot of a social media post

Description automatically generated

Figure 15. Program Memory

## Appendix D – Task IV

### Code for “mylogicops.s”

; Just check out MPLAB

.include "p24Hxxxx.inc"

.global \_\_reset ;The label for the first line of code.

.bss ;unitialized data section

;;These start at location 0x0800 because 0-0x07FF reserved for SFRs

a: .space 1 ;Allocating space (in bytes) to variable.

b: .space 1 ;Allocating space (in bytes) to variable.

c: .space 1 ;Allocating space (in bytes) to variable.

d: .space 1

e: .space 1

f: .space 1

x: .space 2

.text ;Start of Code section

\_\_reset: ; first instruction located at \_\_reset label

mov #\_\_SP\_init, w15 ;Initalize the Stack Pointer

mov #\_\_SPLIM\_init,W0

mov W0, SPLIM ;Initialize the stack limit register

;\_\_SP\_init set by linker to be after allocated data

;User Code starts here.

; C Program equivalent

; uint8 u8\_a, u8\_b, u8\_c, u8\_d, u8\_e, u8\_f;

; uint16 u16\_x=0x0001;

; u8\_a=0xAF;

; u8\_b=0x50;

; u8\_c= u8\_a & u8\_b

; u8\_d= u8\_a | u8\_b

; u8\_e= u8\_a ^ u8\_b

; u8\_f=~u8\_a

; u16\_x=~u8\_d | (u16\_x & u8\_c);

mov #0x0001, w0; w0 = 0x0001

mov wreg, x; x = 0x0001

mov.b #0xAF, w0; w0.lsb = 0xAF

mov.b wreg, a; a = w0.lsb

mov.b #0x50, w0; w0.lsb = 0x50

mov.b wreg, b; b = w0.lsb

and.b a, wreg; w0.lsb = a & b

mov.b wreg, c; c = w0.lsb (c = 0b )

mov.b b, wreg; w0.lsb = b

ior.b a, wreg; w0.lsb = a | b

mov.b wreg, d; d = w0.lsb

mov.b b, wreg; w0.lsb = b

xor.b a, wreg; w0.lsb = a ^ b

mov.b wreg, e; e = wreg.lsb

com.b a, wreg; w0.lsb = ~a

mov.b wreg, f; f = w0.lsb

mov x, wreg; w0 = x

and.b c, wreg; w0 = x.lsb & c

mov w0, w1; w1 = w0

com.b d, wreg; w0.lsb = ~d

ior.b w0, w1, w0; w0 = w1.lsb | (~d)

mov wreg, x; x = w0

done:

goto done ;Place holder for last line of executed code

.end ;End of program code in this file

### Manual Calculation for instruction and flags[[10]](#footnote-10)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruct | Variables[[11]](#footnote-11) | | | | | | | Registers[[12]](#footnote-12) | | Flags | | | |
| Instruction | a | b | c | d | e | f | X | W0 | W1 | N | OV | Z | C |
| mov #0x0001, w0 | - - | - - | - - | - - | - - | - - | - - - - | 0000 0000 0000 0001 | - - - - | 0 | 0 | 0 | 0 |
| mov wreg, x | - - | - - | - - | - - | - - | - - | 0000 0000 0000 0001 | 0000 0000 0000 0001 | - - - - | 0 | 0 | 0 | 0 |
| mov.b #0xAF, w0 | - - | - - | - - | - - | - - | - - | 0000 0000 0000 0001 | 0000 0000 1010 1111 | - - - - | 0 | 0 | 0 | 0 |
| mov.b wreg, a | 1010 1111 | - - | - - | - - | - - | - - | 0000 0000 0000 0001 | 0000 0000 1010 1111 | - - - - | 0 | 0 | 0 | 0 |
| mov.b #0x50, w0 | 1010 1111 | - - | - - | - - | - - | - - | 0000 0000 0000 0001 | 0000 0000 0101 0000 | - - - - | 0 | 0 | 0 | 0 |
| mov.b wreg, b | 1010 1111 | 0101 0000 | - - | - - | - - | - - | 0000 0000 0000 0001 | 0000 0000 0101 0000 | - - - - | 0 | 0 | 0 | 0 |
| and.b a, wreg | 1010 1111 | 0101 0000 | - - | - - | - - | - - | 0000 0000 0000 0001 | 0000 0000 0000 0000 | - - - - | 0 | 0 | 1 | 0 |
| mov.b wreg, c | 1010 1111 | 0101 0000 | 0000 0000 | - - | - - | - - | 0000 0000 0000 0001 | 0000 0000 0000 0000 | - - - - | 0 | 0 | 1 | 0 |
| mov.b b, wreg | 1010 1111 | 0101 0000 | 0000 0000 | - - | - - | - - | 0000 0000 0000 0001 | 0000 0000 0101 0000 | - - - - | 0 | 0 | 0 | 0 |
| ior.b a, wreg | 1010 1111 | 0101 0000 | 0000 0000 | - - | - - | - - | 0000 0000 0000 0001 | 0000 0000 1111 1111 | - - - - | 1 | 0 | 0 | 0 |
| mov.b wreg, d | 1010 1111 | 0101 0000 | 0000 0000 | 1111 1111 | - - | - - | 0000 0000 0000 0001 | 0000 0000 1111 1111 | - - - - | 1 | 0 | 0 | 0 |
| mov.b b, wreg | 1010 1111 | 0101 0000 | 0000 0000 | 1111 1111 | - - | - - | 0000 0000 0000 0001 | 0000 0000 0101 0000 | - - - - | 0 | 0 | 0 | 0 |
| xor.b a, wreg | 1010 1111 | 0101 0000 | 0000 0000 | 1111 1111 | - - | - - | 0000 0000 0000 0001 | 0000 0000 1111 1111 | - - - - | 1 | 0 | 0 | 0 |
| mov.b wreg, e | 1010 1111 | 0101 0000 | 0000 0000 | 1111 1111 | 1111 1111 | - - | 0000 0000 0000 0001 | 0000 0000 1111 1111 | - - - - | 1 | 0 | 0 | 0 |
| com.b a, wreg | 1010 1111 | 0101 0000 | 0000 0000 | 1111 1111 | 1111 1111 | - - | 0000 0000 0000 0001 | 0000 0000 0101 0000 | - - - - | 0 | 0 | 0 | 0 |
| mov.b wreg, f | 1010 1111 | 0101 0000 | 0000 0000 | 1111 1111 | 1111 1111 | 0101 0000 | 0000 0000 0000 0001 | 0000 0000 0101 0000 | - - - - | 0 | 0 | 0 | 0 |
| mov x, wreg | 1010 1111 | 0101 0000 | 0000 0000 | 1111 1111 | 1111 1111 | 0101 0000 | 0000 0000 0000 0001 | 0000 0000 0000 0001 | - - - - | 0 | 0 | 0 | 0 |
| and.b c, wreg | 1010 1111 | 0101 0000 | 0000 0000 | 1111 1111 | 1111 1111 | 0101 0000 | 0000 0000 0000 0001 | 0000 0000 0000 0000 | - - - - | 0 | 0 | 1 | 0 |
| mov w0, w1 | 1010 1111 | 0101 0000 | 0000 0000 | 1111 1111 | 1111 1111 | 0101 0000 | 0000 0000 0000 0001 | 0000 0000 0000 0000 | 0000 0000 0000 0000 | 0 | 0 | 1 | 0 |
| com.b d, wreg | 1010 1111 | 0101 0000 | 0000 0000 | 1111 1111 | 1111 1111 | 0101 0000 | 0000 0000 0000 0001 | 0000 0000 0000 0000 | 0000 0000 0000 0000 | 0 | 0 | 1 | 0 |
| ior.b w0, w1, w0 | 1010 1111 | 0101 0000 | 0000 0000 | 1111 1111 | 1111 1111 | 0101 0000 | 0000 0000 0000 0001 | 0000 0000 0000 0000 | 0000 0000 0000 0000 | 0 | 0 | 1 | 0 |
| mov wreg, x | 1010 1111 | 0101 0000 | 0000 0000 | 1111 1111 | 1111 1111 | 0101 0000 | 0000 0000 0000 0001 | 0000 0000 0000 0000 | 0000 0000 0000 0000 | 0 | 0 | 1 | 0 |

### Figures

#### Simulated (16 - 20)

A screenshot of a cell phone

Description automatically generated

Figure 16. mylogicops.s

A screenshot of a social media post

Description automatically generated

Figure 17. Watch Window

A screenshot of a cell phone

Description automatically generated

Figure 18. Special Function Register

A screenshot of a social media post

Description automatically generated

Figure 19. Program Memory

A screen shot of a social media post

Description automatically generated

Figure 20. File Registers

#### On board

A screenshot of a social media post

Description automatically generated

Figure 21. Output Window

A screenshot of a cell phone

Description automatically generated

Figure 22. File Register

A screenshot of a social media post

Description automatically generated

Figure 23. Special Function Register

A screenshot of a social media post

Description automatically generated

Figure 24. Program Memory

A screenshot of a social media post

Description automatically generated

Figure 25. Watch Window

## Appendix E – Manual Scrips

### Manual Computation(26 - 27)

A close up of text on a whiteboard

Description automatically generated

Figure 26. Manual Scrip 1(Top left: Task1, Top right: Task2, Bottom: Task3)

A close up of text on a whiteboard

Description automatically generated

Figure 27. Manual Scrip 2 (Task4)

### Flag Table in Task-3(28)

A close up of text on a white surface

Description automatically generated

Figure 28. Task3 Flag Form

1. The manual scrip is in [Appendix E](#_Manual_Computation(26_-) [↑](#footnote-ref-1)
2. The value of variables will be performed in unsigned decimal. [↑](#footnote-ref-2)
3. The value of registers will be performed in unsigned decima. [↑](#footnote-ref-3)
4. The manual scrip is in [Appendix E](#_Manual_Computation(26_-). [↑](#footnote-ref-4)
5. The value of variables will be performed in unsigned decimal. [↑](#footnote-ref-5)
6. The value of Register will be performed in unsigned decimal. [↑](#footnote-ref-6)
7. The manual scrip is in [Appendix E](#_Manual_Computation(26_-). [↑](#footnote-ref-7)
8. Value of variables will be performed in unsigned decimal. [↑](#footnote-ref-8)
9. Value of registers will be performed in unsigned decimal. [↑](#footnote-ref-9)
10. The manual scrip is in [Appendix V](#_Manual_Computation(26_-). [↑](#footnote-ref-10)
11. Value of variables will be performed in 8 bits or 16 bits Binary. Figure “-” represent 4 bits. [↑](#footnote-ref-11)
12. Value of registers will be performed in Binary. Figure “-” represent 4 bits. [↑](#footnote-ref-12)