MPLAB Introduction and PIC24 Assembly Language

4th Laboratory Report for ECE 383

Microcomputers

Submitted by

Yichen Huang

11906882

Shomari Thomas

11672867

The University of Alabama

Tuscaloosa, Alabama 35487

February 12, 2020

Table of Contents

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.

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 .

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. The flag table in this section will be shown in Appendix E.

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

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

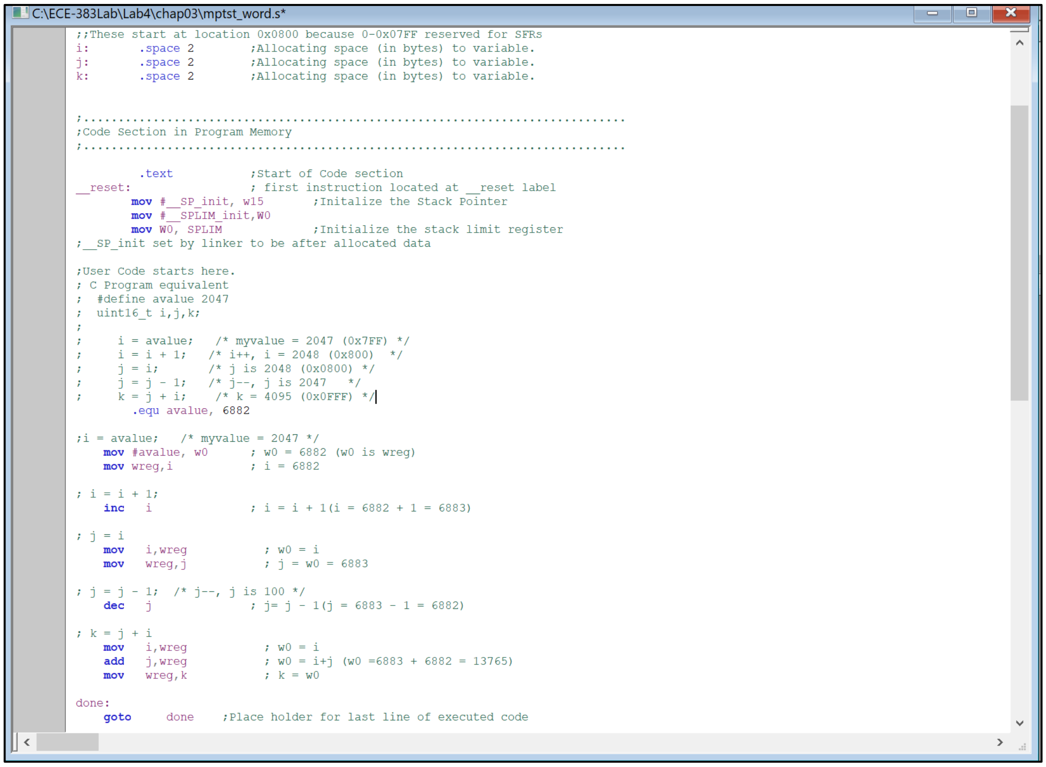
Code for “mptst_word.s”

```
; 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]

Instruct	Variables[2]			Resisters[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)



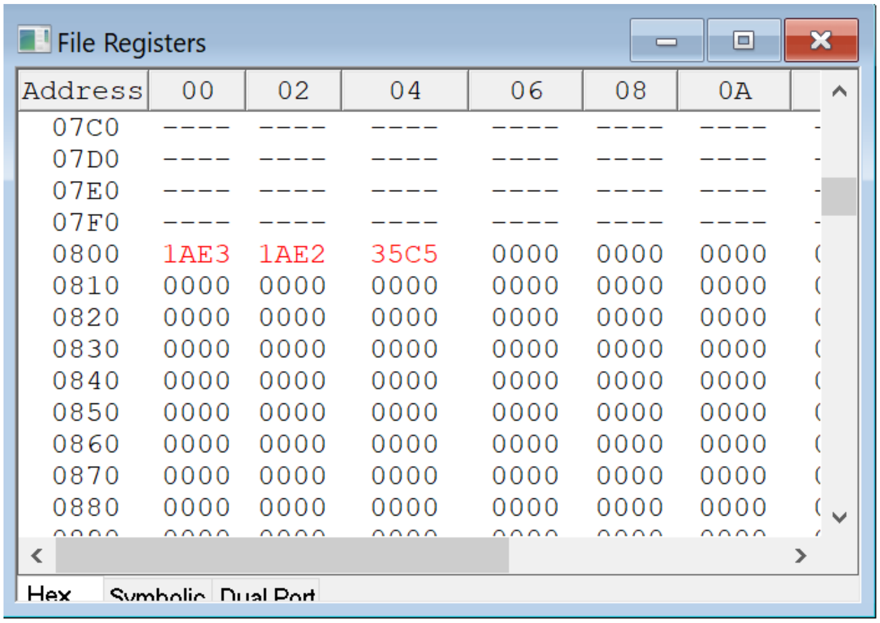
```
;;These start at location 0x0800 because 0-0x07FF reserved for SFRs
i:        .space 2        ;Allocating space (in bytes) to variable.
j:        .space 2        ;Allocating space (in bytes) to variable.
k:        .space 2        ;Allocating space (in bytes) to variable.


;..............................................................................
;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 = 2047 */
    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
```

Figure 1 . mptst_word.s

[1] The manual scrip is in Appendix E

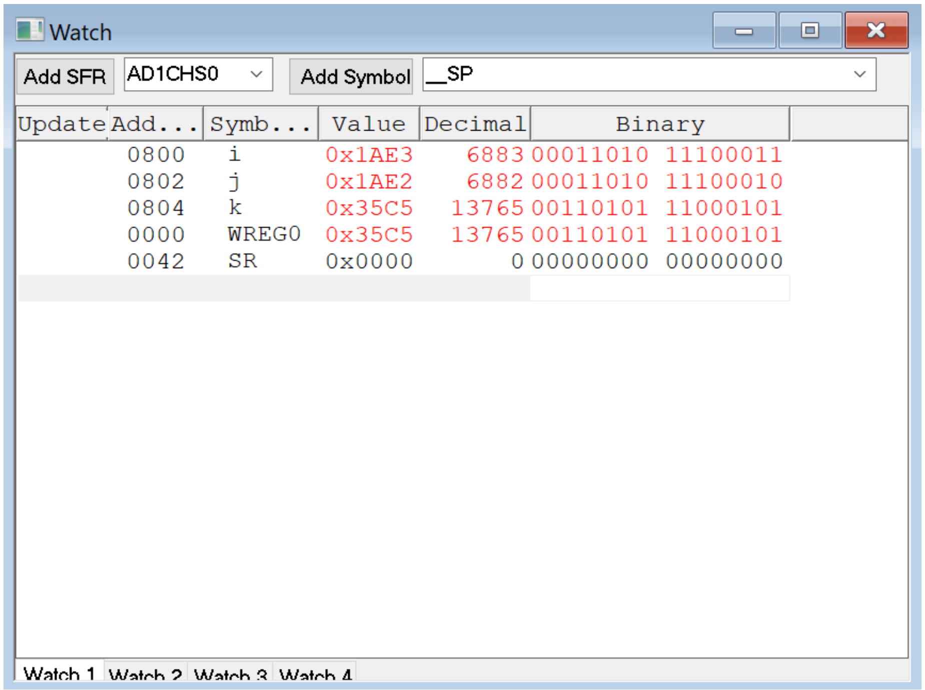
[2] The value of variables will be performed in unsigned decimal.

[3] The value of registers will be performed in unsigned decima.



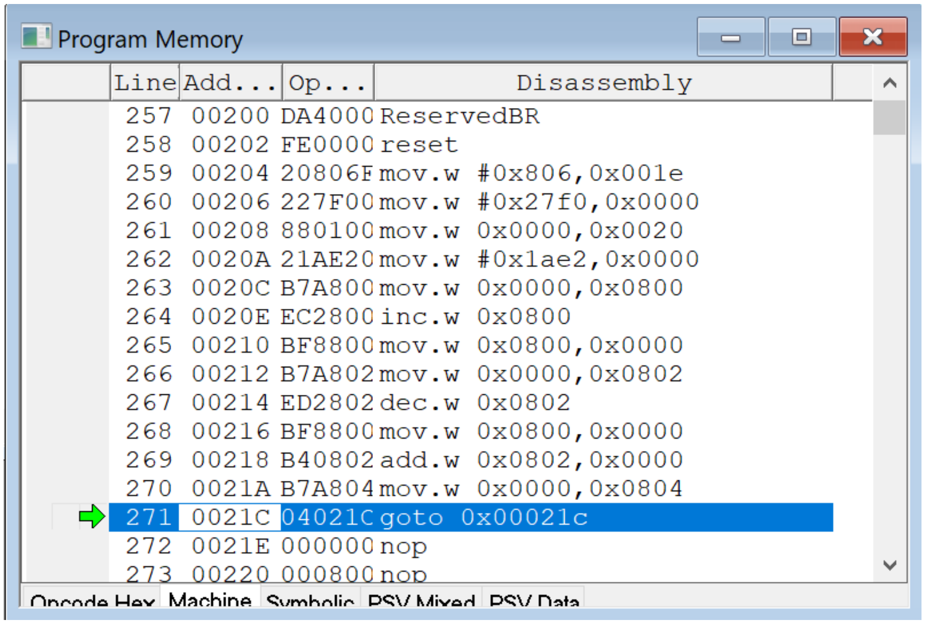
```
Address   00    02    04    06    08    0A
 07C0    ----  ----  ----  ----  ----  ----
 07D0    ----  ----  ----  ----  ----  ----
 07E0    ----  ----  ----  ----  ----  ----
 07F0    ----  ----  ----  ----  ----  ----
 0800    1AE3  1AE2  35C5  0000  0000  0000
 0810    0000  0000  0000  0000  0000  0000
 0820    0000  0000  0000  0000  0000  0000
 0830    0000  0000  0000  0000  0000  0000
 0840    0000  0000  0000  0000  0000  0000
 0850    0000  0000  0000  0000  0000  0000
 0860    0000  0000  0000  0000  0000  0000
 0870    0000  0000  0000  0000  0000  0000
 0880    0000  0000  0000  0000  0000  0000
```

Figure 2. File Registers



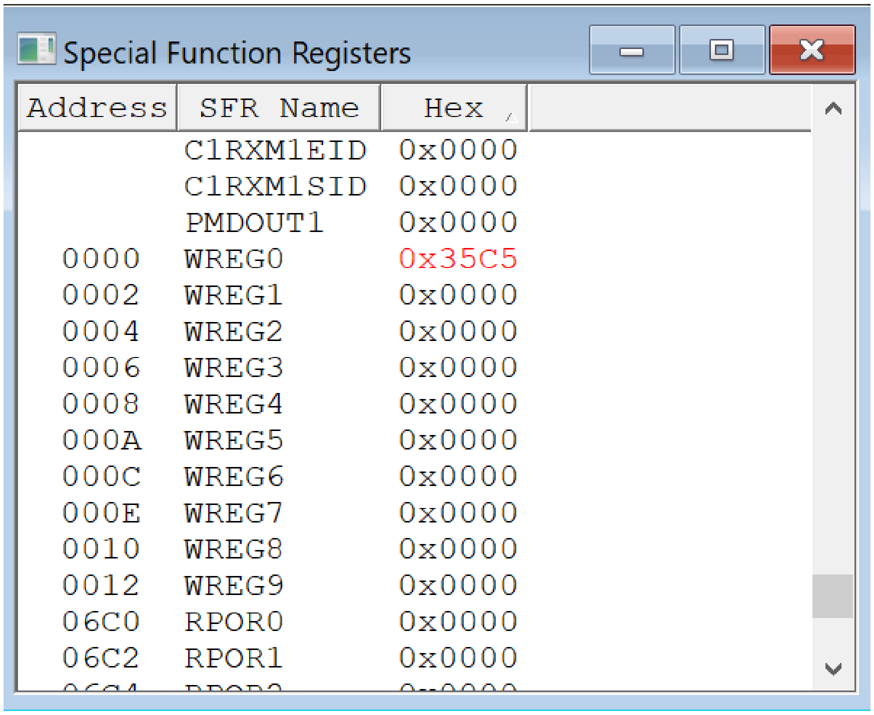
```
Update Add...  Symb...  Value    Decimal         Binary
        0800    i        0x1AE3    6883 00011010  11100011
        0802    j        0x1AE2    6882 00011010  11100010
        0804    k        0x35C5   13765 00110101  11000101
        0000    WREG0    0x35C5   13765 00110101  11000101
        0042    SR       0x0000       0 00000000  00000000
```

Figure 3. Watch Window



```
Line Add... Op...  Disassembly
257  00200  DA4000 ReservedBR
258  00202  FE0000 reset
259  00204  20806F mov.w  #0x806,0x001e
260  00206  227F00 mov.w  #0x27f0,0x0000
261  00208  880100 mov.w  0x0000,0x0020
262  0020A  21AE20 mov.w  #0x1ae2,0x0000
263  0020C  B7A800 mov.w  0x0000,0x0800
264  0020E  EC2800 inc.w  0x0800
265  00210  BF8800 mov.w  0x0800,0x0000
266  00212  B7A802 mov.w  0x0000,0x0802
267  00214  ED2802 dec.w  0x0802
268  00216  BF8800 mov.w  0x0800,0x0000
269  00218  B40802 add.w  0x0802,0x0000
270  0021A  B7A804 mov.w  0x0000,0x0804
271  0021C  04021C goto   0x00021c
272  0021E  000000 nop
273  00220  000800 nop
```

Figure 4. Program Memory



```
Address  SFR Name    Hex
         C1RXM1EID   0x0000
         C1RXM1SID   0x0000
         PMDOUT1     0x0000
0000     WREG0       0x35C5
0002     WREG1       0x0000
0004     WREG2       0x0000
0006     WREG3       0x0000
0008     WREG4       0x0000
000A     WREG5       0x0000
000C     WREG6       0x0000
000E     WREG7       0x0000
0010     WREG8       0x0000
0012     WREG9       0x0000
06C0     RPOR0       0x0000
06C2     RPOR1       0x0000
06C4     RPOR2       0x0000
```

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]

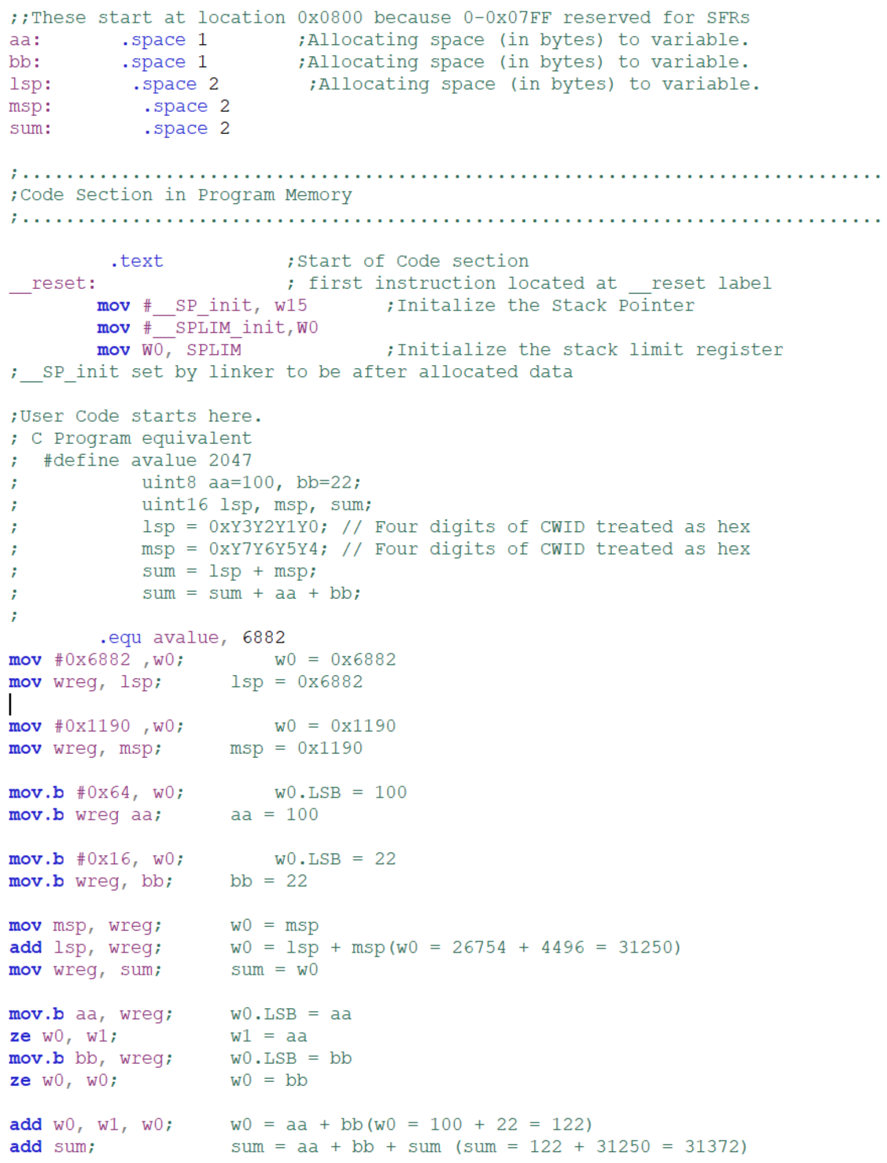
Instruct	Variables[5]					Registers[6]		Flags			
Instruction	aa	bb	lsp	msp	sum	W0	W1	N	OV	Z	C
mov #0x6882 ,w0	-	-	-	-	-	26754	-	0	0	0	0
mov wreg, lsp	-	-	26754	-	-	26754	-	0	0	0	0
mov #0x1190 ,w0	-	-	26754	-	-	4496	-	0	0	0	0
mov wreg, msp	-	-	26754	4496	-	4496	-	0	0	0	0
mov.b #0x64, w0	-	-	26754	4496	-	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

[4] The manual scrip is in Appendix E.

[5] The value of variables will be performed in unsigned decimal.

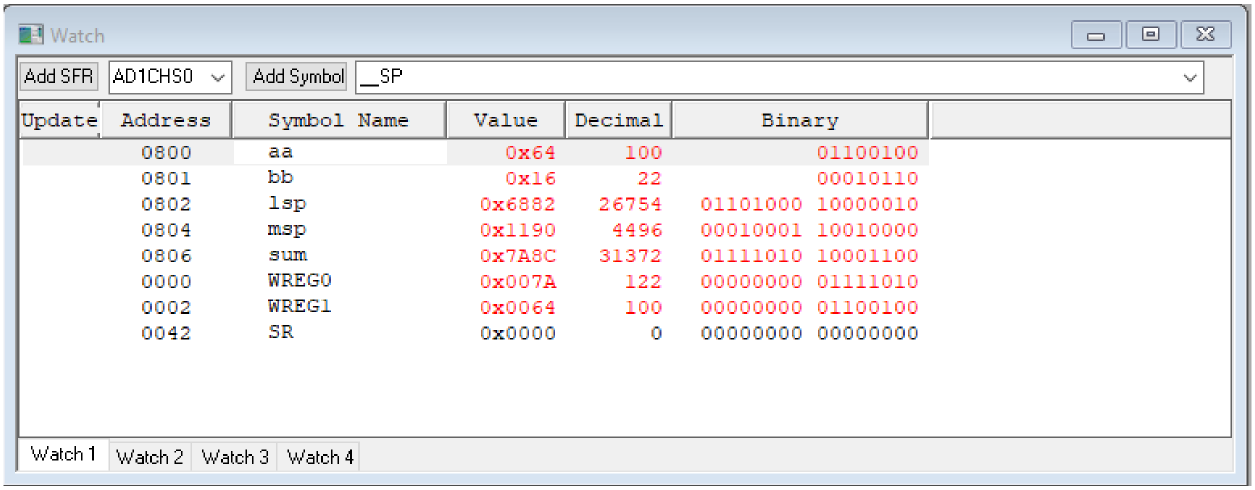
[6] The value of Register will be performed in unsigned decimal.

Figures(6-10)



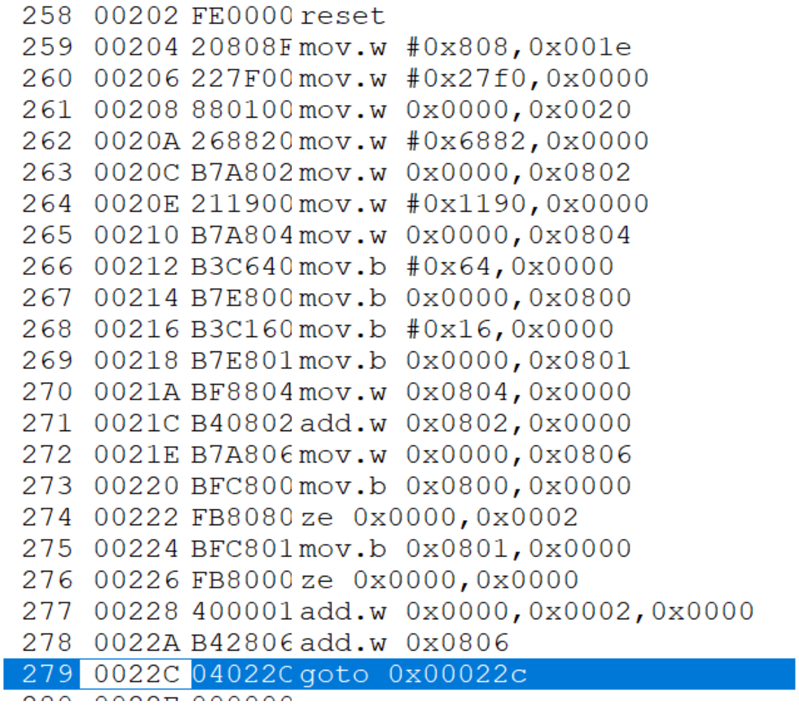
```
;;These start at location 0x0800 because 0-0x07FF reserved for SFRs
aa:       .space 1        ;Allocating space (in bytes) to variable.
bb:       .space 1        ;Allocating space (in bytes) to variable.
lsp:       .space 2        ;Allocating space (in bytes) to variable.
msp:        .space 2
sum:        .space 2

;..............................................................................
;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
;           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)
```

Figure 6. myadd.s



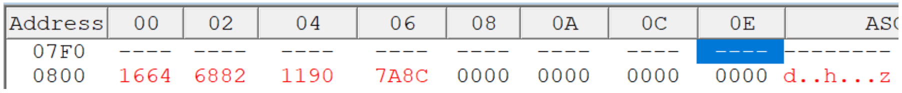
Update	Address	Symbol Name	Value	Decimal	Binary
	0800	aa	0x64	100	01100100
	0801	bb	0x16	22	00010110
	0802	lsp	0x6882	26754	01101000 10000010
	0804	msp	0x1190	4496	00010001 10010000
	0806	sum	0x7A8C	31372	01111010 10001100
	0000	WREG0	0x007A	122	00000000 01111010
	0002	WREG1	0x0064	100	00000000 01100100
	0042	SR	0x0000	0	00000000 00000000

Figure 7. Watch Window



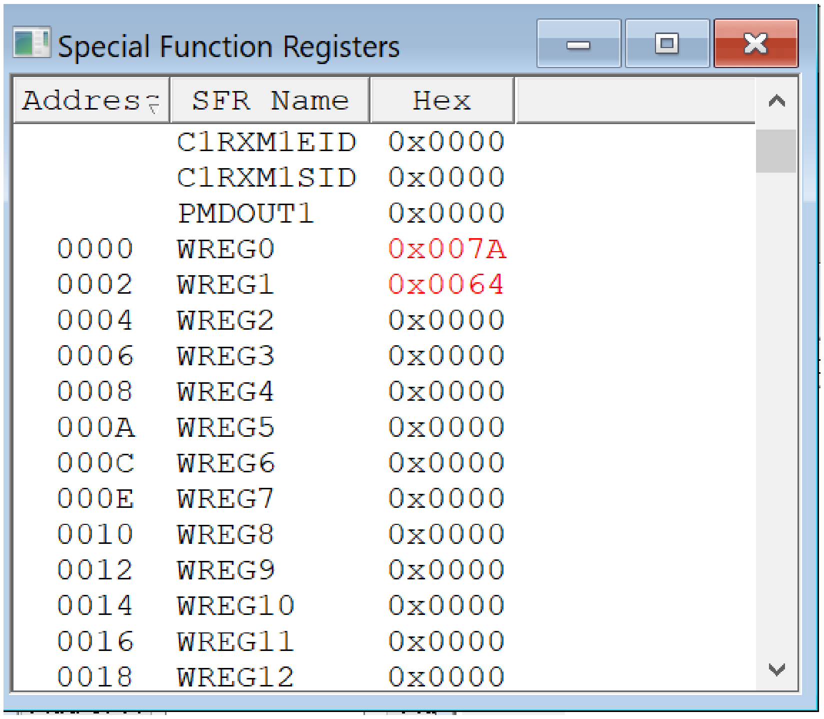
```
00202 FE0000 reset
00204 20808F mov.w  #0x808,0x001e
00206 227F00 mov.w  #0x27f0,0x0000
00208 880100 mov.w  0x0000,0x0020
0020A 268820 mov.w  #0x6882,0x0000
0020C B7A802 mov.w  0x0000,0x0802
0020E 211900 mov.w  #0x1190,0x0000
00210 B7A804 mov.w  0x0000,0x0804
00212 B3C640 mov.b  #0x64,0x0000
00214 B7E800 mov.b  0x0000,0x0800
00216 B3C160 mov.b  #0x16,0x0000
00218 B7E801 mov.b  0x0000,0x0801
0021A BF8804 mov.w  0x0804,0x0000
0021C B40802 add.w  0x0802,0x0000
0021E B7A806 mov.w  0x0000,0x0806
00220 BFC800 mov.b  0x0800,0x0000
00222 FB8080 ze  0x0000,0x0002
00224 BFC801 mov.b  0x0801,0x0000
00226 FB8000 ze  0x0000,0x0000
00228 400001 add.w  0x0000,0x0002,0x0000
0022A B42806 add.w  0x0806
0022C 04022C goto  0x00022c
```

Figure 8. Program Memory



```
Address  00    02    04    06    08    0A    0C    0E    ASC
 07F0   ----  ----  ----  ----  ----  ----  ----  ----  --------
 0800   1664  6882  1190  7A8C  0000  0000  0000  0000  d..h...z
```

Figure 9. File Registers



Addres	SFR Name	Hex
	C1RXM1EID	0x0000
	C1RXM1SID	0x0000
	PMDOUT1	0x0000
0000	WREG0	0x007A
0002	WREG1	0x0064
0004	WREG2	0x0000
0006	WREG3	0x0000
0008	WREG4	0x0000
000A	WREG5	0x0000
000C	WREG6	0x0000
000E	WREG7	0x0000
0010	WREG8	0x0000
0012	WREG9	0x0000
0014	WREG10	0x0000
0016	WREG11	0x0000
0018	WREG12	0x0000

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 – 1
;           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]

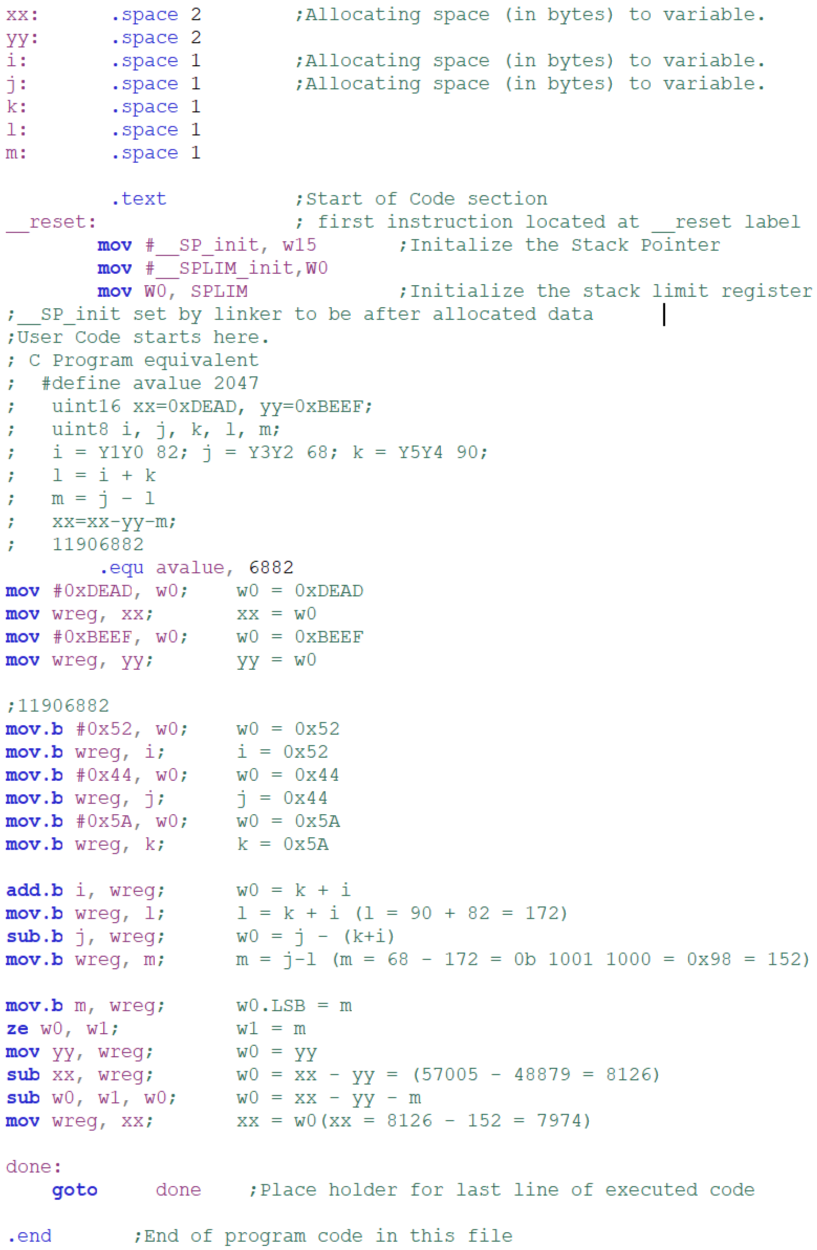
Instruct	Variables[8]							Registers[9]		Flags			
Instruction	i	j	k	l	m	xx	yy	W0	W1	N	OV	Z	C
mov #0xDEAD, w0	-	-	-	-	-	-	-	57005	-	0	0	0	0
mov wreg, xx	-	-	-	-	-	57005	-	57005	-	0	0	0	0
mov #0xBEEF, w0	-	-	-	-	-	57005	-	48879	-	0	0	0	0
mov wreg, yy	-	-	-	-	-	57005	48879	48779	-	0	0	0	0
mov.b #0x52, 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

[7] The manual scrip is in Appendix E.

[8] Value of variables will be performed in unsigned decimal.

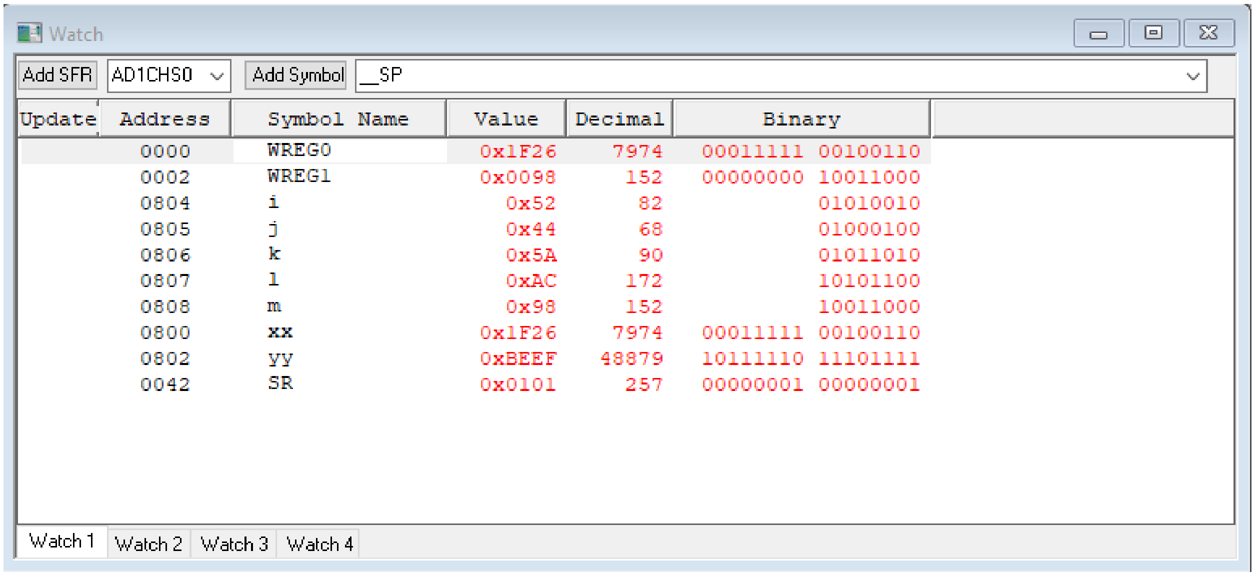
[9] Value of registers will be performed in unsigned decimal.

Figures(11 - 15)



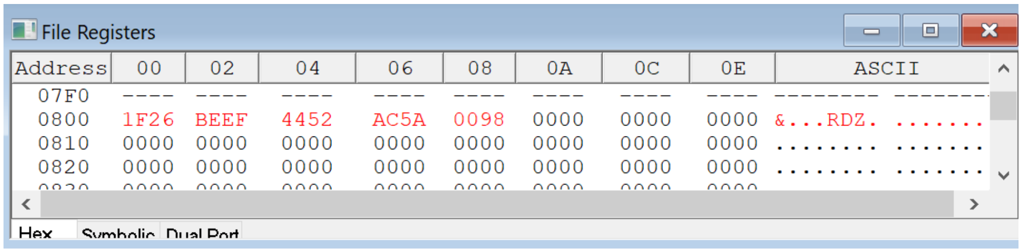
```
xx:      .space 2        ;Allocating space (in bytes) to variable.
yy:      .space 2
i:       .space 1        ;Allocating space (in bytes) to variable.
j:       .space 1        ;Allocating space (in bytes) to variable.
k:       .space 1
l:       .space 1
m:       .space 1

         .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
        .equ avalue, 6882
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
```

Figure 11. mysub.s



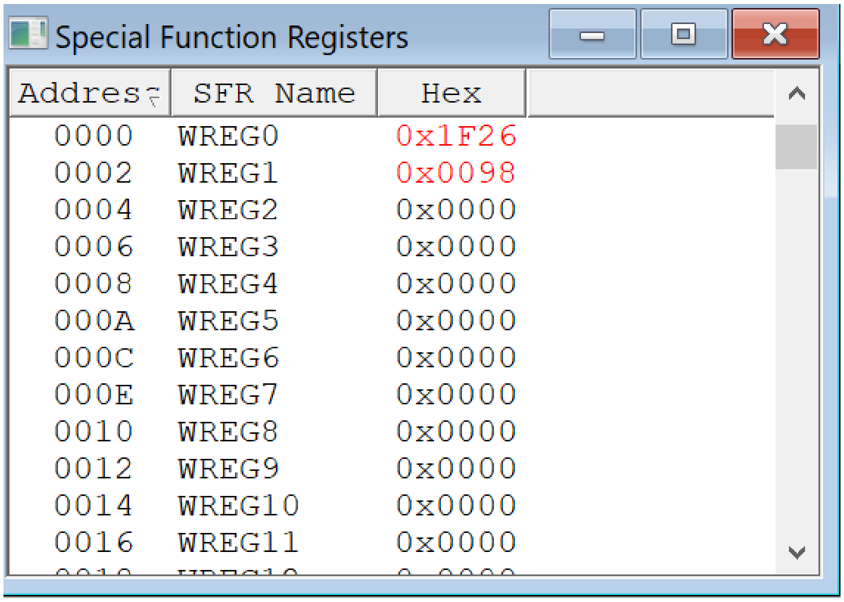
Update	Address	Symbol Name	Value	Decimal	Binary
	0000	WREG0	0x1F26	7974	00011111 00100110
	0002	WREG1	0x0098	152	00000000 10011000
	0804	i	0x52	82	01010010
	0805	j	0x44	68	01000100
	0806	k	0x5A	90	01011010
	0807	l	0xAC	172	10101100
	0808	m	0x98	152	10011000
	0800	xx	0x1F26	7974	00011111 00100110
	0802	yy	0xBEEF	48879	10111110 11101111
	0042	SR	0x0101	257	00000001 00000001

Figure 12. Watch Window



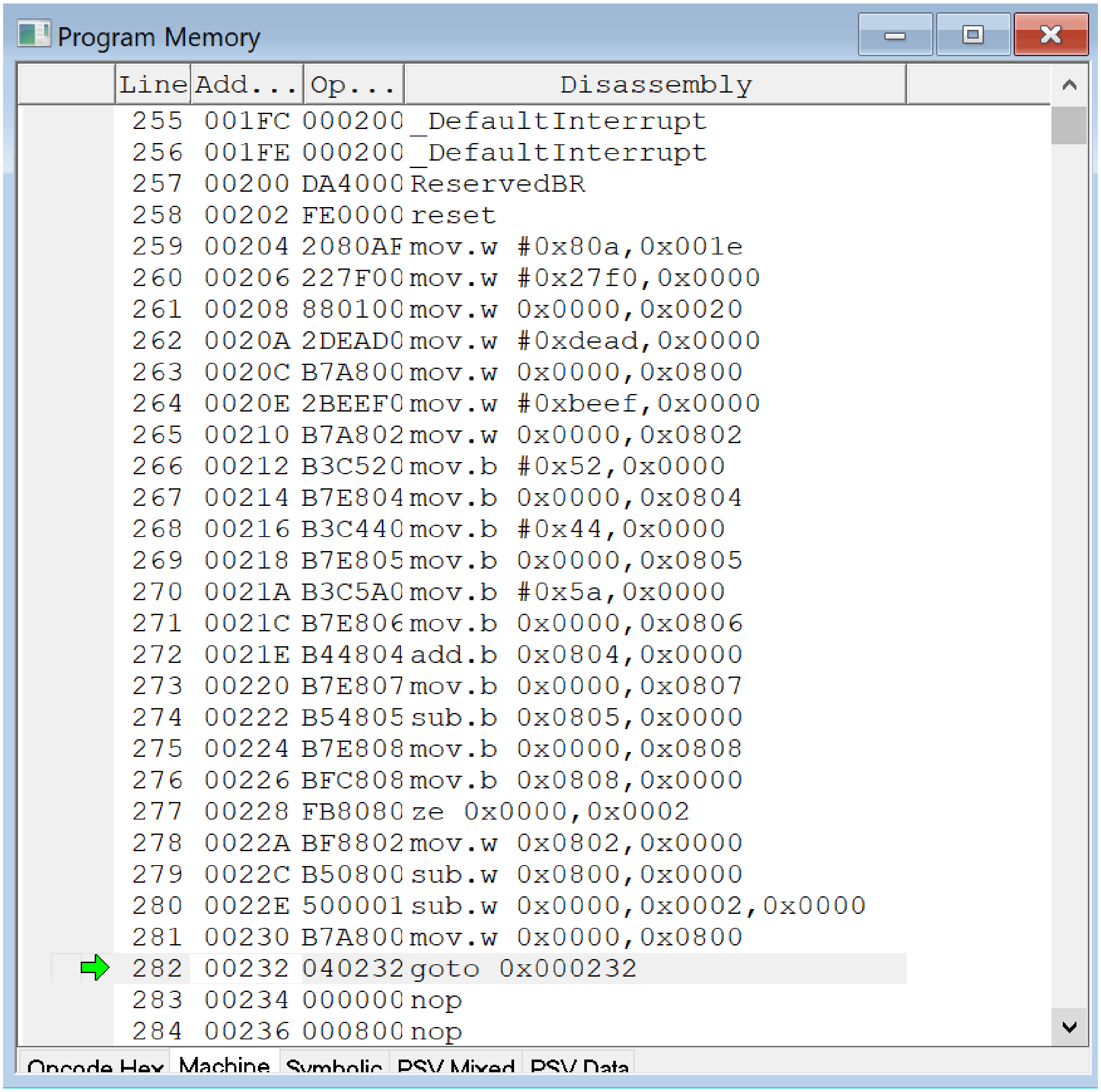
Address	00	02	04	06	08	0A	0C	0E	ASCII
07F0	----	----	----	----	----	----	----	----	-------- --------
0800	1F26	BEEF	4452	AC5A	0098	0000	0000	0000	&...RDZ. ....
0810	0000	0000	0000	0000	0000	0000	0000	0000	....
0820	0000	0000	0000	0000	0000	0000	0000	0000	....

Figure 13. File Register



Addres	SFR Name	Hex
0000	WREG0	0x1F26
0002	WREG1	0x0098
0004	WREG2	0x0000
0006	WREG3	0x0000
0008	WREG4	0x0000
000A	WREG5	0x0000
000C	WREG6	0x0000
000E	WREG7	0x0000
0010	WREG8	0x0000
0012	WREG9	0x0000
0014	WREG10	0x0000
0016	WREG11	0x0000

Figure 14. Special Function Register



```
Line Add... Op...   Disassembly
 255 001FC 000200 _DefaultInterrupt
 256 001FE 000200 _DefaultInterrupt
 257 00200 DA4000 ReservedBR
 258 00202 FE0000 reset
 259 00204 2080AF mov.w  #0x80a,0x001e
 260 00206 227F00 mov.w  #0x27f0,0x0000
 261 00208 880100 mov.w  0x0000,0x0020
 262 0020A 2DEAD0 mov.w  #0xdead,0x0000
 263 0020C B7A800 mov.w  0x0000,0x0800
 264 0020E 2BEEF0 mov.w  #0xbeef,0x0000
 265 00210 B7A802 mov.w  0x0000,0x0802
 266 00212 B3C520 mov.b  #0x52,0x0000
 267 00214 B7E804 mov.b  0x0000,0x0804
 268 00216 B3C440 mov.b  #0x44,0x0000
 269 00218 B7E805 mov.b  0x0000,0x0805
 270 0021A B3C5A0 mov.b  #0x5a,0x0000
 271 0021C B7E806 mov.b  0x0000,0x0806
 272 0021E B44804 add.b  0x0804,0x0000
 273 00220 B7E807 mov.b  0x0000,0x0807
 274 00222 B54805 sub.b  0x0805,0x0000
 275 00224 B7E808 mov.b  0x0000,0x0808
 276 00226 BFC808 mov.b  0x0808,0x0000
 277 00228 FB8080 ze  0x0000,0x0002
 278 0022A BF8802 mov.w  0x0802,0x0000
 279 0022C B50800 sub.w  0x0800,0x0000
 280 0022E 500001 sub.w  0x0000,0x0002,0x0000
 281 00230 B7A800 mov.w  0x0000,0x0800
⇨282 00232 040232 goto  0x000232
 283 00234 000000 nop
 284 00236 000800 nop
```

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
```

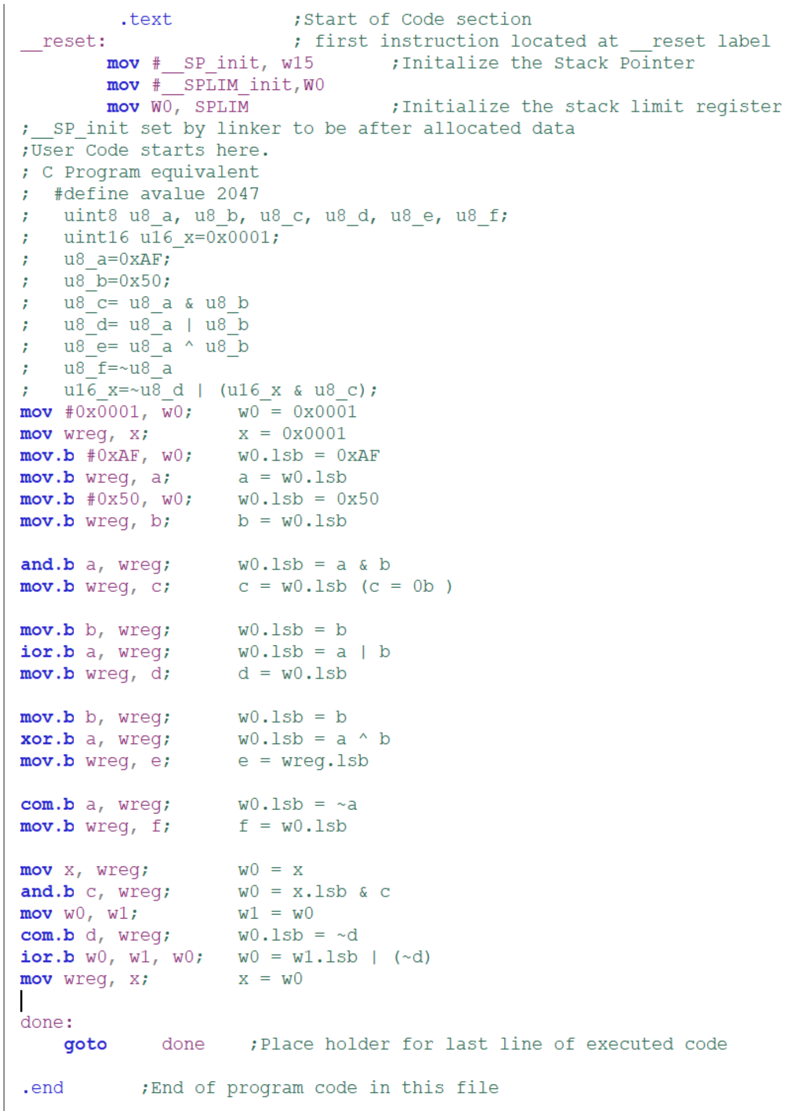
Manual Calculation for instruction and flags[10]

Instruct	Variables[11]							Registers[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

[10] The manual scrip is in Appendix V.
[11] Value of variables will be performed in 8 bits or 16 bits Binary. Figure "-" represent 4 bits.
[12] Value of registers will be performed in Binary. Figure "-" represent 4 bits.

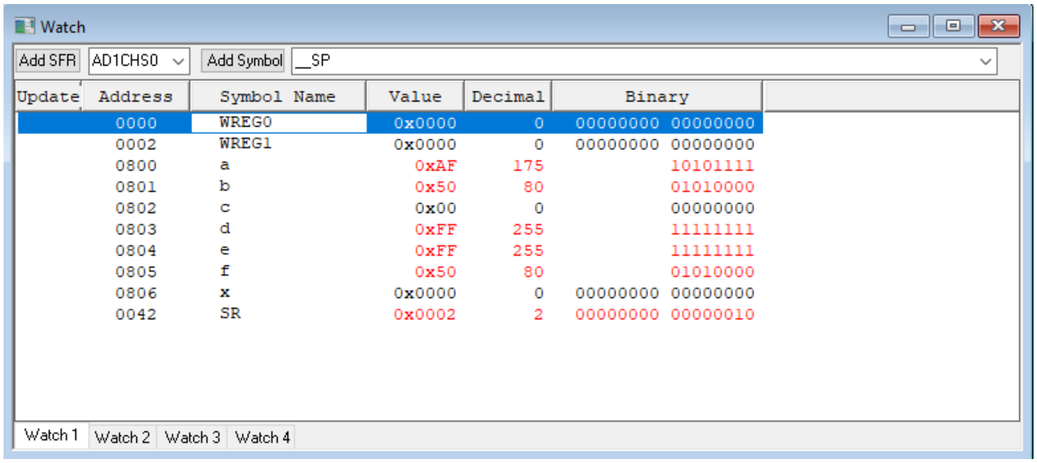
Figures

Simulated (16 - 20)



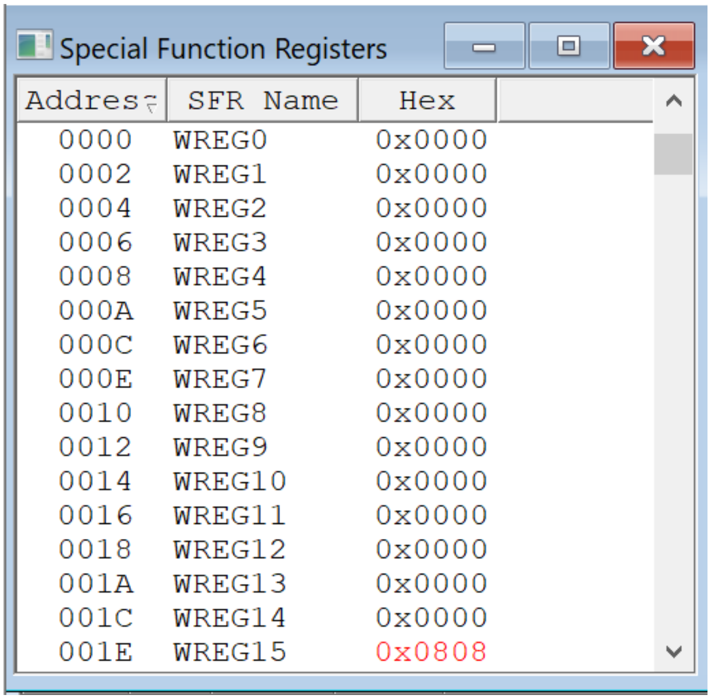
```
         .text               ;Start of Code section
__reset:                     ; first instruction located at __reset label
        mov #__SP_init, w15       ;Initalize the Stack Pointer
        mov #__SPLIM_init,W0
        mov W0, SPLIM             ;Initialize the stack limit register
;__SP_init set by linker to be after allocated data
;User Code starts here.
; C Program equivalent
;  #define avalue 2047
;   uint8 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
```

Figure 16. mylogicops.s



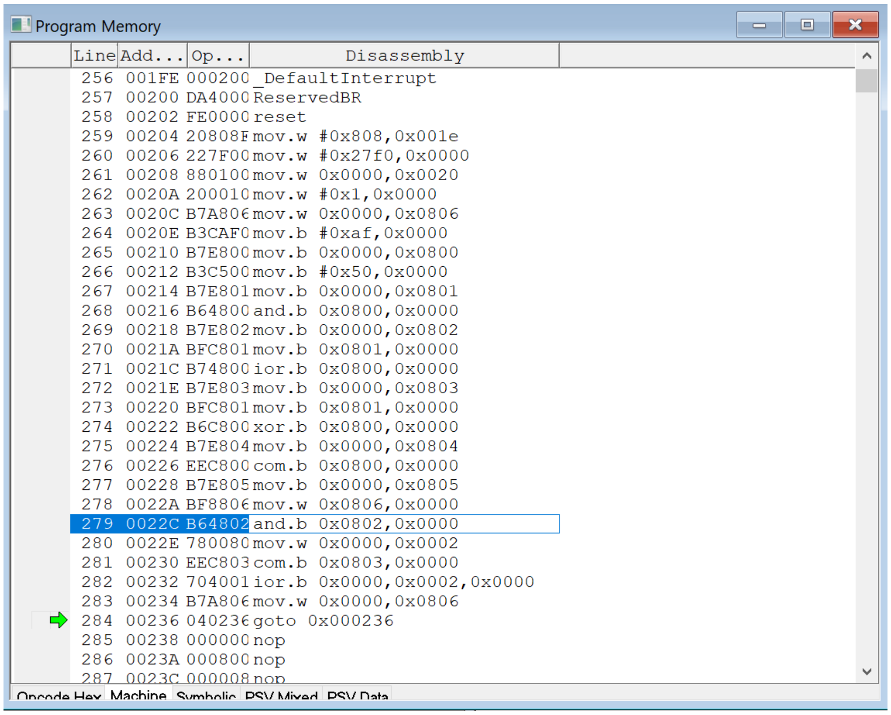
Update	Address	Symbol Name	Value	Decimal	Binary
	0000	WREG0	0x0000	0	00000000 00000000
	0002	WREG1	0x0000	0	00000000 00000000
	0800	a	0xAF	175	10101111
	0801	b	0x50	80	01010000
	0802	c	0x00	0	00000000
	0803	d	0xFF	255	11111111
	0804	e	0xFF	255	11111111
	0805	f	0x50	80	01010000
	0806	x	0x0000	0	00000000 00000000
	0042	SR	0x0002	2	00000000 00000010

Figure 17. Watch Window



Address	SFR Name	Hex
0000	WREG0	0x0000
0002	WREG1	0x0000
0004	WREG2	0x0000
0006	WREG3	0x0000
0008	WREG4	0x0000
000A	WREG5	0x0000
000C	WREG6	0x0000
000E	WREG7	0x0000
0010	WREG8	0x0000
0012	WREG9	0x0000
0014	WREG10	0x0000
0016	WREG11	0x0000
0018	WREG12	0x0000
001A	WREG13	0x0000
001C	WREG14	0x0000
001E	WREG15	0x0808

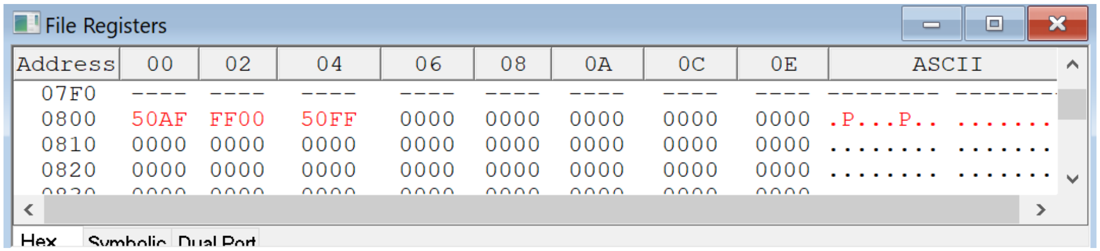
Figure 18. Special Function Register



Program Memory

```
Line Add... Op...           Disassembly
 256 001FE 000200 _DefaultInterrupt
 257 00200 DA4000 ReservedBR
 258 00202 FE0000 reset
 259 00204 20808F mov.w  #0x808,0x001e
 260 00206 227F00 mov.w  #0x27f0,0x0000
 261 00208 880100 mov.w  0x0000,0x0020
 262 0020A 200010 mov.w  #0x1,0x0000
 263 0020C B7A806 mov.w  0x0000,0x0806
 264 0020E B3CAF0 mov.b  #0xaf,0x0000
 265 00210 B7E800 mov.b  0x0000,0x0800
 266 00212 B3C500 mov.b  #0x50,0x0000
 267 00214 B7E801 mov.b  0x0000,0x0801
 268 00216 B64800 and.b  0x0800,0x0000
 269 00218 B7E802 mov.b  0x0000,0x0802
 270 0021A BFC801 mov.b  0x0801,0x0000
 271 0021C B74800 ior.b  0x0800,0x0000
 272 0021E B7E803 mov.b  0x0000,0x0803
 273 00220 BFC801 mov.b  0x0801,0x0000
 274 00222 B6C800 xor.b  0x0800,0x0000
 275 00224 B7E804 mov.b  0x0000,0x0804
 276 00226 EEC800 com.b  0x0800,0x0000
 277 00228 B7E805 mov.b  0x0000,0x0805
 278 0022A BF8806 mov.w  0x0806,0x0000
 279 0022C B64802 and.b  0x0802,0x0000
 280 0022E 780080 mov.w  0x0000,0x0002
 281 00230 EEC803 com.b  0x0803,0x0000
 282 00232 704001 ior.b  0x0000,0x0002,0x0000
 283 00234 B7A806 mov.w  0x0000,0x0806
 284 00236 040236 goto  0x000236
 285 00238 000000 nop
 286 0023A 000800 nop
 287 0023C 000008 nop
Opcode Hex  Machine  Symbolic  PSV Mixed  PSV Data
```

Figure 19. Program Memory

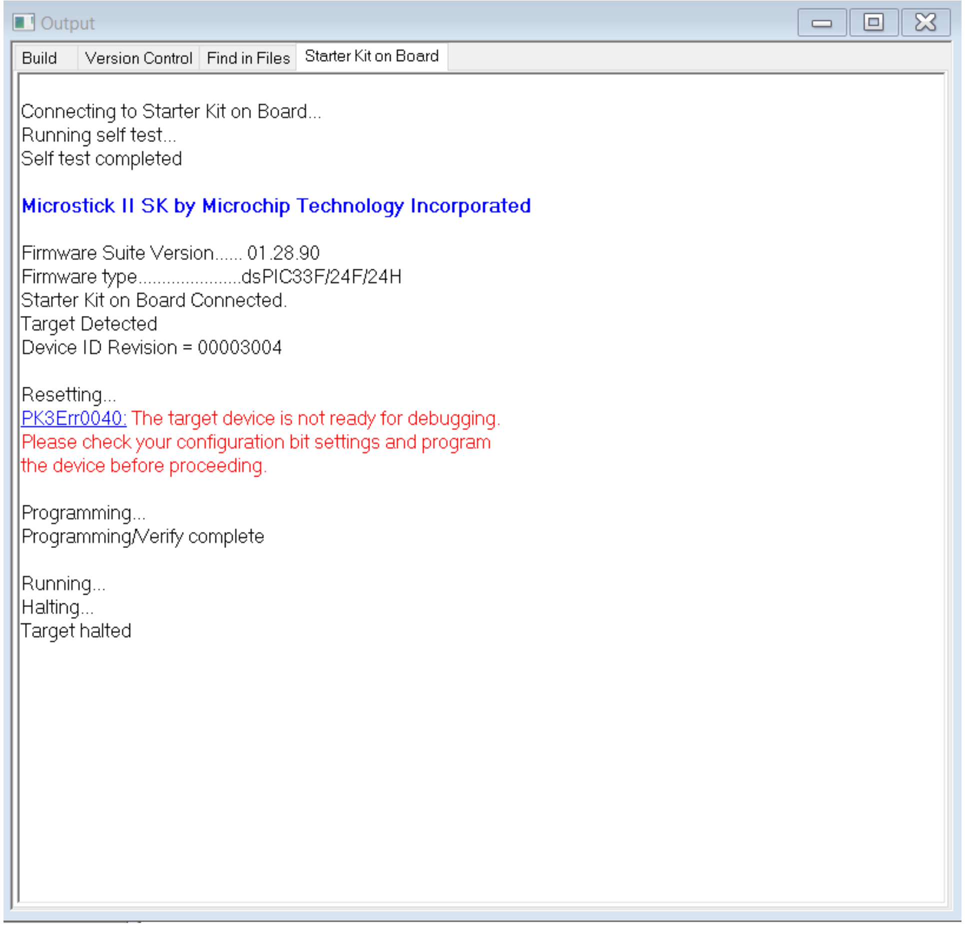


File Registers

```
Address   00    02    04    06    08    0A    0C    0E         ASCII
  07F0   ----  ----  ----  ----  ----  ----  ----  ----  --------  --------
  0800   50AF  FF00  50FF  0000  0000  0000  0000  0000  .P...P..  .......
  0810   0000  0000  0000  0000  0000  0000  0000  0000  ........  .......
  0820   0000  0000  0000  0000  0000  0000  0000  0000  ........  .......
  0830   0000  0000  0000  0000  0000  0000  0000  0000
Hex  Symbolic  Dual Port
```

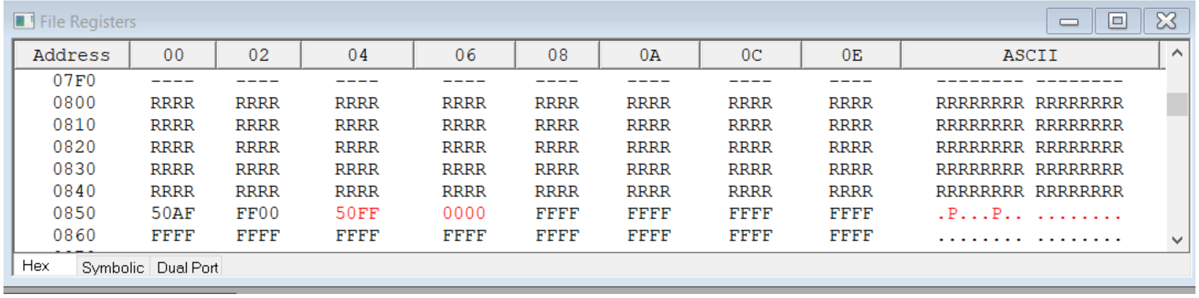
Figure 20. File Registers

On board



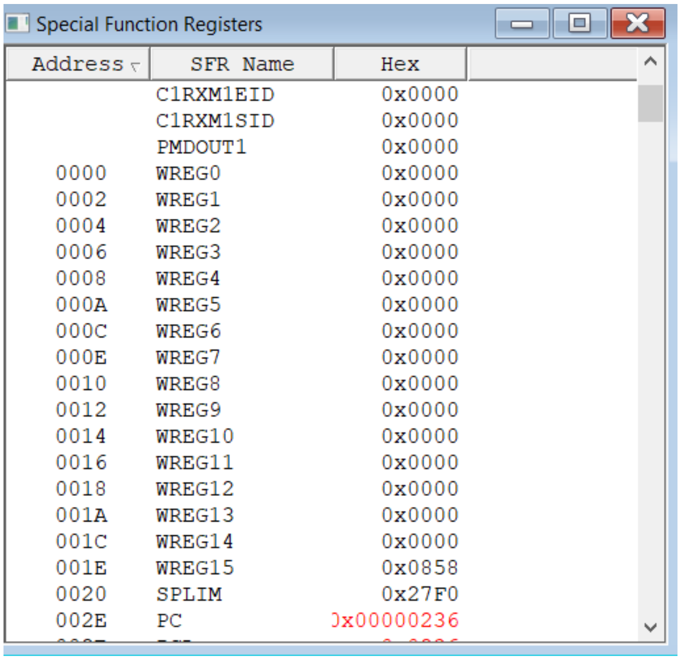
```
Connecting to Starter Kit on Board...
Running self test...
Self test completed

Microstick II SK by Microchip Technology Incorporated

Firmware Suite Version...... 01.28.90
Firmware type.....................dsPIC33F/24F/24H
Starter Kit on Board Connected.
Target Detected
Device ID Revision = 00003004

Resetting...
PK3Err0040: The target device is not ready for debugging.
Please check your configuration bit settings and program
the device before proceeding.

Programming...
Programming/Verify complete

Running...
Halting...
Target halted
```

Figure 21. Output Window



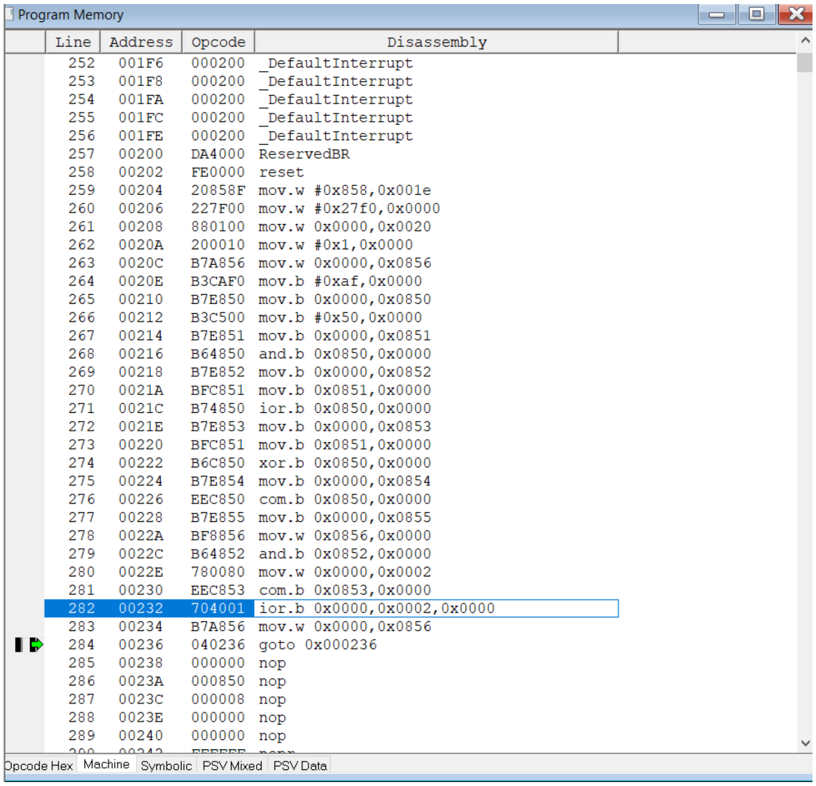
```
Address   00    02    04    06    08    0A    0C    0E         ASCII
  07F0   ----  ----  ----  ----  ----  ----  ----  ----   -------- --------
  0800   RRRR  RRRR  RRRR  RRRR  RRRR  RRRR  RRRR  RRRR   RRRRRRRR RRRRRRRR
  0810   RRRR  RRRR  RRRR  RRRR  RRRR  RRRR  RRRR  RRRR   RRRRRRRR RRRRRRRR
  0820   RRRR  RRRR  RRRR  RRRR  RRRR  RRRR  RRRR  RRRR   RRRRRRRR RRRRRRRR
  0830   RRRR  RRRR  RRRR  RRRR  RRRR  RRRR  RRRR  RRRR   RRRRRRRR RRRRRRRR
  0840   RRRR  RRRR  RRRR  RRRR  RRRR  RRRR  RRRR  RRRR   RRRRRRRR RRRRRRRR
  0850   50AF  FF00  50FF  0000  FFFF  FFFF  FFFF  FFFF   .P...P.. ........
  0860   FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF   ........ ........
```

Figure 22. File Register



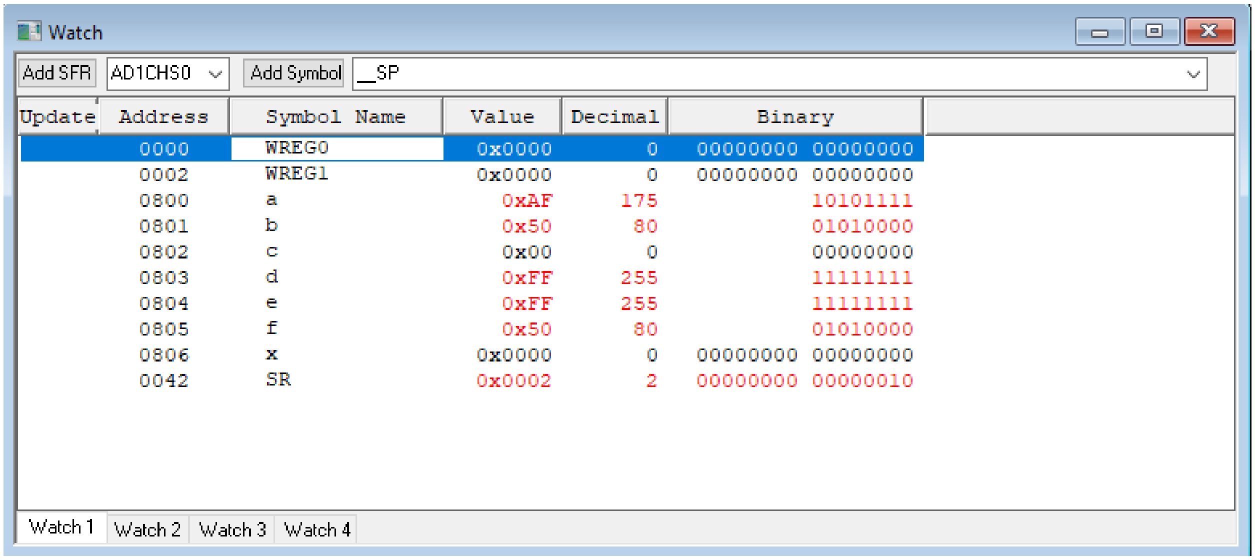
Address	SFR Name	Hex
	C1RXM1EID	0x0000
	C1RXM1SID	0x0000
	PMDOUT1	0x0000
0000	WREG0	0x0000
0002	WREG1	0x0000
0004	WREG2	0x0000
0006	WREG3	0x0000
0008	WREG4	0x0000
000A	WREG5	0x0000
000C	WREG6	0x0000
000E	WREG7	0x0000
0010	WREG8	0x0000
0012	WREG9	0x0000
0014	WREG10	0x0000
0016	WREG11	0x0000
0018	WREG12	0x0000
001A	WREG13	0x0000
001C	WREG14	0x0000
001E	WREG15	0x0858
0020	SPLIM	0x27F0
002E	PC	0x00000236

Figure 23. Special Function Register



Line	Address	Opcode	Disassembly
252	001F6	000200	_DefaultInterrupt
253	001F8	000200	_DefaultInterrupt
254	001FA	000200	_DefaultInterrupt
255	001FC	000200	_DefaultInterrupt
256	001FE	000200	_DefaultInterrupt
257	00200	DA4000	ReservedBR
258	00202	FE0000	reset
259	00204	20858F	mov.w #0x858,0x001e
260	00206	227F00	mov.w #0x27f0,0x0000
261	00208	880100	mov.w 0x0000,0x0020
262	0020A	200010	mov.w #0x1,0x0000
263	0020C	B7A856	mov.w 0x0000,0x0856
264	0020E	B3CAF0	mov.b #0xaf,0x0000
265	00210	B7E850	mov.b 0x0000,0x0850
266	00212	B3C500	mov.b #0x50,0x0000
267	00214	B7E851	mov.b 0x0000,0x0851
268	00216	B64850	and.b 0x0850,0x0000
269	00218	B7E852	mov.b 0x0000,0x0852
270	0021A	BFC851	mov.b 0x0851,0x0000
271	0021C	B74850	ior.b 0x0850,0x0000
272	0021E	B7E853	mov.b 0x0000,0x0853
273	00220	BFC851	mov.b 0x0851,0x0000
274	00222	B6C850	xor.b 0x0850,0x0000
275	00224	B7E854	mov.b 0x0000,0x0854
276	00226	EEC850	com.b 0x0850,0x0000
277	00228	B7E855	mov.b 0x0000,0x0855
278	0022A	BF8856	mov.w 0x0856,0x0000
279	0022C	B64852	and.b 0x0852,0x0000
280	0022E	780080	mov.w 0x0000,0x0002
281	00230	EEC853	com.b 0x0853,0x0000
282	00232	704001	ior.b 0x0000,0x0002,0x0000
283	00234	B7A856	mov.w 0x0000,0x0856
284	00236	040236	goto 0x000236
285	00238	000000	nop
286	0023A	000850	nop
287	0023C	000008	nop
288	0023E	000000	nop
289	00240	000000	nop

Figure 24. Program Memory

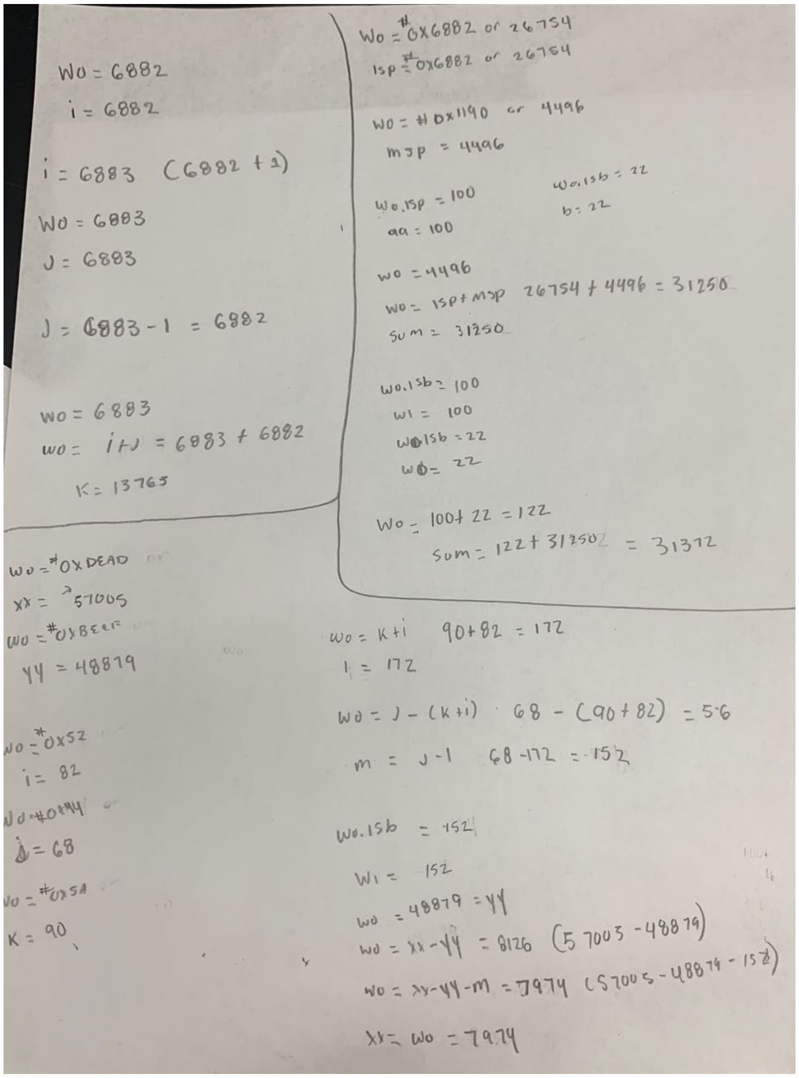


Update	Address	Symbol Name	Value	Decimal	Binary
	0000	WREG0	0x0000	0	00000000 00000000
	0002	WREG1	0x0000	0	00000000 00000000
	0800	a	0xAF	175	10101111
	0801	b	0x50	80	01010000
	0802	c	0x00	0	00000000
	0803	d	0xFF	255	11111111
	0804	e	0xFF	255	11111111
	0805	f	0x50	80	01010000
	0806	x	0x0000	0	00000000 00000000
	0042	SR	0x0002	2	00000000 00000010

Figure 25. Watch Window

Appendix E – Manual Scrips

Manual Computation(26 - 27)



Wo = 6882
i = 6882

i = 6883 (6882 + 1)

Wo = 6883
J = 6883

J = 6883 - 1 = 6882

wo = 6883
wo = i+J = 6883 + 6882
K = 13765

Wo = #0X6882 or 26754
1sp = #0X6882 or 26754

wo = #0x1190 or 4496
msp = 4496

wo.1sp = 100
aa = 100

wo.1sb = 22
b = 22

wo = 4496
wo = 1sp+msp 26754 + 4496 = 31250
sum = 31250

wo.1sb = 100
w1 = 100
wo1sb = 22
wo = 22

Wo = 100+ 22 = 122
sum = 122+ 31250 = 31372

wo = #0XDEAD or
xx = 57005
wo = #0XBEEF
yy = 48879

wo = #0x52
i = 82
wo = #0x44
j = 68
wo = #0x5A
K = 90

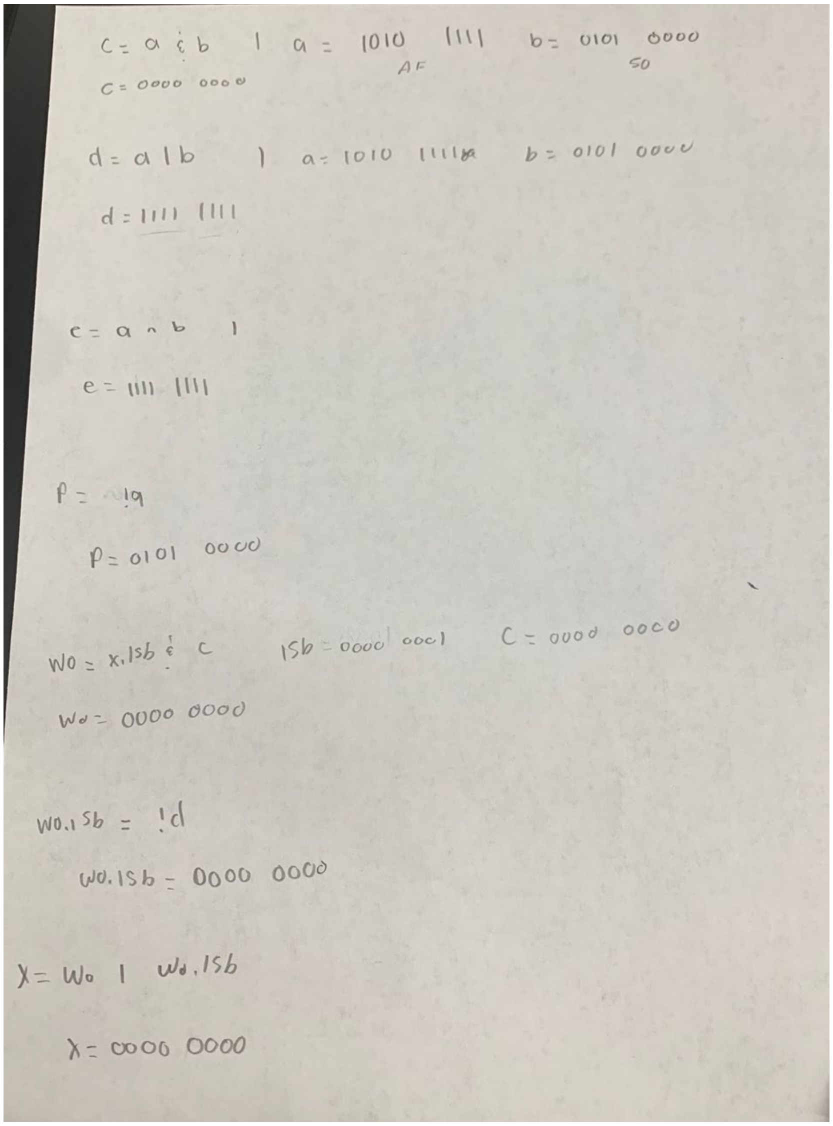
wo = K+i 90+82 = 172
l = 172

wo = J - (K+i) 68 - (90+82) = 56
m = J - l 68 -172 = -152

wo.1sb = 152
W1 = 152
wo = 48879 = yy
wo = xx - yy = 8126 (57003 - 48879)
wo = xx-yy-m = 7974 (57005 - 48879 - 152)

xx = wo = 7974

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



c = a & b | a = 1010 1111 b = 0101 0000
AF 50
c = 0000 0000

d = a | b | a = 1010 1111 b = 0101 0000
d = 1111 1111

e = a ^ b |
e = 1111 1111

p = !a
p = 0101 0000

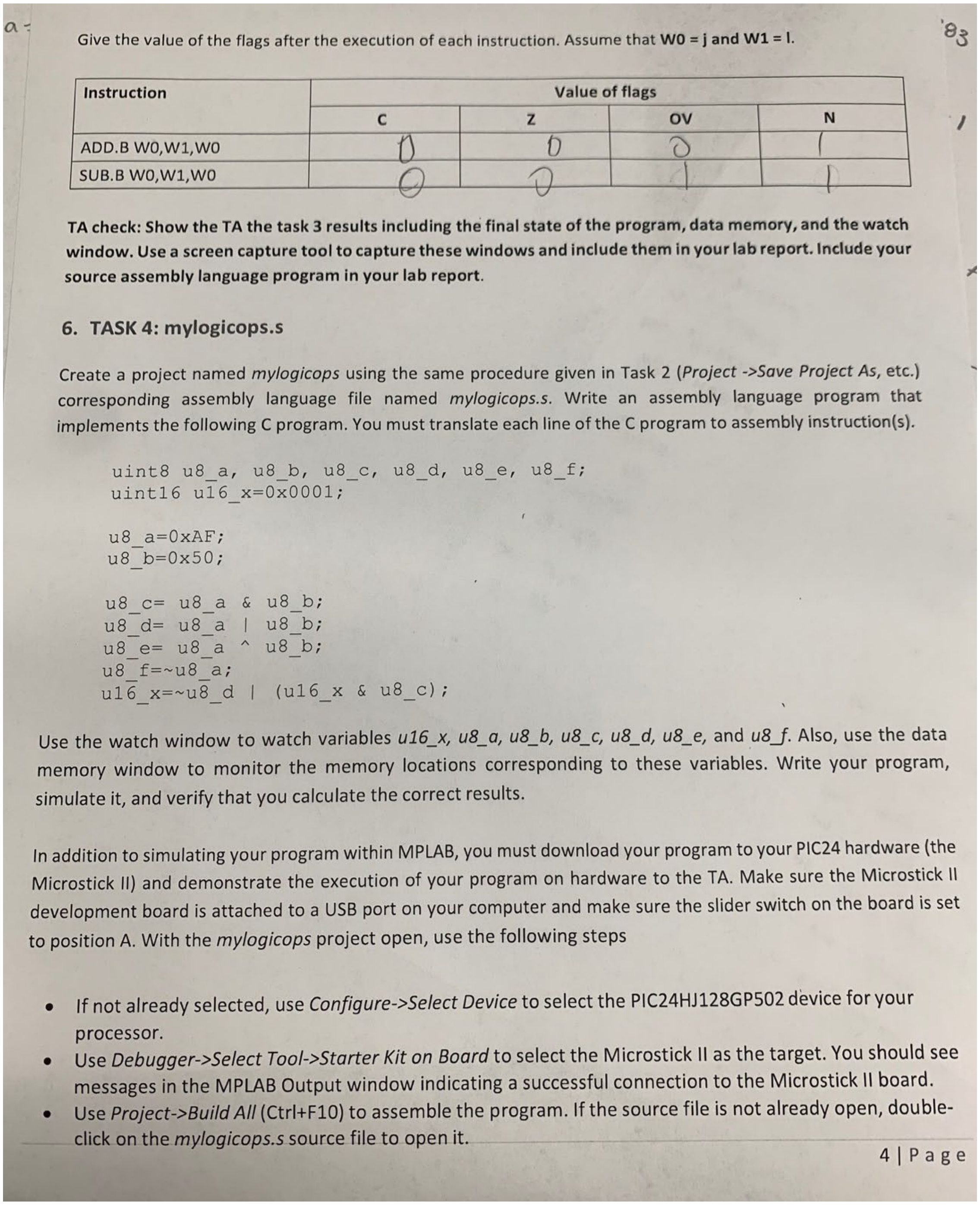
w0 = x.lsb & c   lsb = 0000 0001   c = 0000 0000
w0 = 0000 0000

w0.lsb = !d
w0.lsb = 0000 0000

x = w0 | w0.lsb
x = 0000 0000

Figure 27. Manual Scrip 2 (Task4)

Flag Table in Task-3(28)



83

Give the value of the flags after the execution of each instruction. Assume that **W0 = j and W1 = l.**

Instruction	Value of flags			
	C	Z	OV	N
ADD.B W0,W1,W0	0	0	0	1
SUB.B W0,W1,W0	0	0	1	1

TA check: Show the TA the task 3 results including the final state of the program, data memory, and the watch window. Use a screen capture tool to capture these windows and include them in your lab report. Include your source assembly language program in your lab report.

6. TASK 4: mylogicops.s

Create a project named *mylogicops* using the same procedure given in Task 2 (*Project ->Save Project As*, etc.) corresponding assembly language file named *mylogicops.s.* Write an assembly language program that implements the following C program. You must translate each line of the C program to assembly instruction(s).

```
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);
```

Use the watch window to watch variables *u16_x, u8_a, u8_b, u8_c, u8_d, u8_e,* and *u8_f.* Also, use the data memory window to monitor the memory locations corresponding to these variables. Write your program, simulate it, and verify that you calculate the correct results.

In addition to simulating your program within MPLAB, you must download your program to your PIC24 hardware (the Microstick II) and demonstrate the execution of your program on hardware to the TA. Make sure the Microstick II development board is attached to a USB port on your computer and make sure the slider switch on the board is set to position A. With the *mylogicops* project open, use the following steps

- If not already selected, use *Configure->Select Device* to select the PIC24HJ128GP502 device for your processor.
- Use *Debugger->Select Tool->Starter Kit on Board* to select the Microstick II as the target. You should see messages in the MPLAB Output window indicating a successful connection to the Microstick II board.
- Use *Project->Build All* (Ctrl+F10) to assemble the program. If the source file is not already open, double-click on the *mylogicops.s* source file to open it.

4 | Page

Figure 28. Task3 Flag Form