**C and PIC24 Assembly Language Programming**

**5th Laboratory Report for ECE 383**

**Microcomputers**

**Submitted by**

**Shomari Thomas**

**11672867**

**Yichen Huang**

**11906882**

**The University of Alabama**

**Tuscaloosa, Alabama 35487**

**February 19, 2020**

**Abstract**

Lab 5 was an introduction to translating basic C language code to PIC24 assembly language instructions. In order to become more familiar with this practice, we executed programs using the C language as well as their equivalent programs in PIC24 assembly language. During this lab, we implemented basic C arithmetic operations, converted PIC24 assembly language to C program language, converted C program language to PIC24 assembly language, and implemented use of PIC24 hardware.

For task 1, we used the C program provided in the lab document to create multiple C and assembly language projects implementing arithmetic operations. In task 2, we used MPLAB to create a project containing a C program that meets the specifications provided in the lab document. For task 3, we created a PIC24 assembly language program that is the equivalent of the C program from task 2. Task 4 consisted of implementing the provided PIC24 assembly language program in the lab document, writing and implementing the equivalent of the C program language, and downloading the program onto the PIC24 hardware.

In Lab 5, we became familiar with translation between C program language and PIC24 assembly language and verified the success of our programs using live results and the accuracy of the response from the PIC24 hardware after successful download of a C program equivalent to an assembly language program.

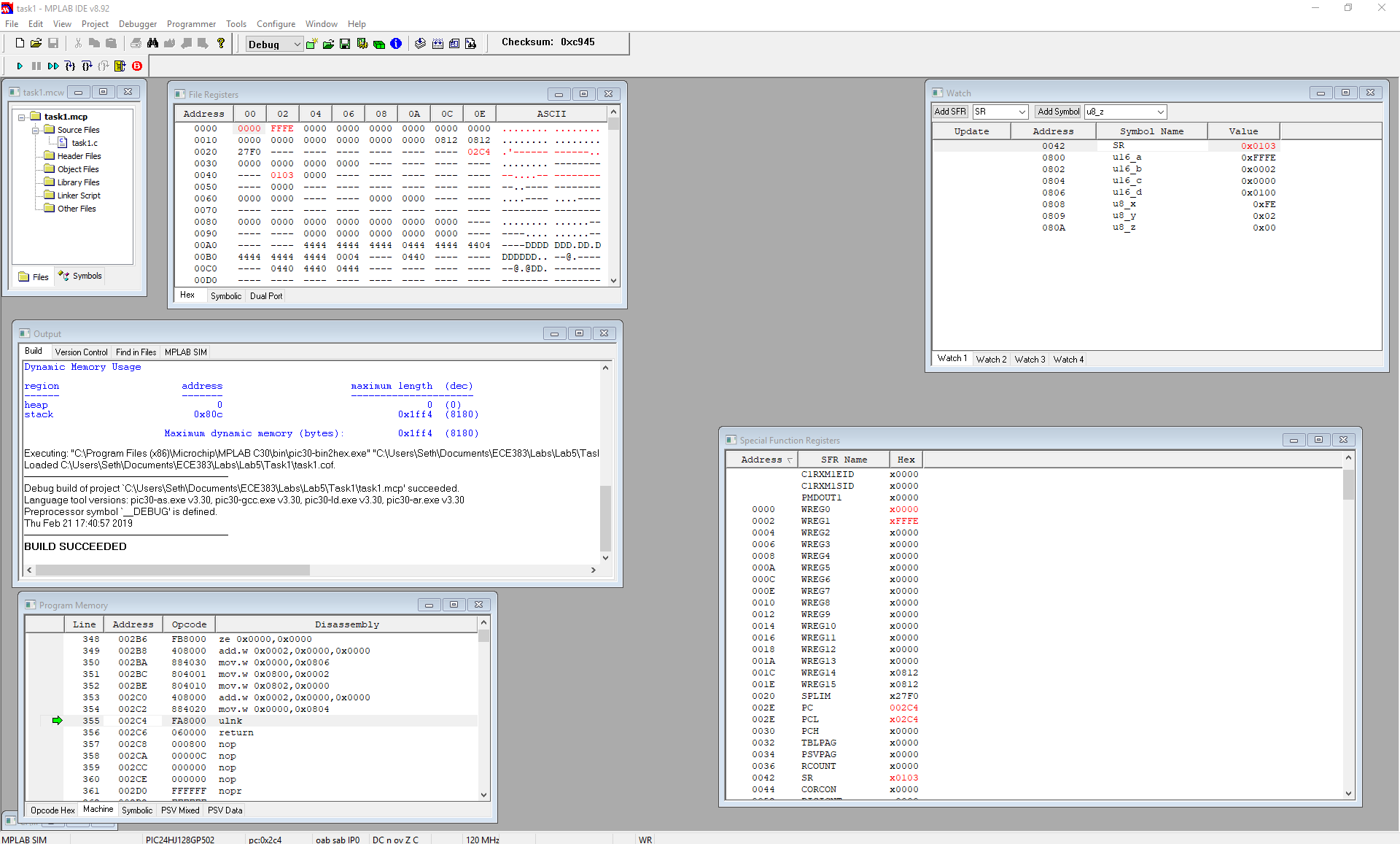
**Introduction**

In Lab 5 we were introduced to translating basic PIC24 assembly language into C programming language and C programming language to PIC24 assembly language in the MPLAB Integrated Development Environment (IDE). The MPLAB Integrated Development Environment (IDE) was used as a helpful tool to simulate both the assembly language programs and the C language programs and provide the results in the form of the final state of the programs, data memory, and watch windows. Additionally, a C language program was downloaded onto a PIC24 device, giving a physical example of the capabilities of the C language program and its application in PIC24 hardware use. This lab exemplified the similarities between the capabilities of C language and PIC24 assembly language via simulation, and showed how we can use the C language, PIC24 assembly language, and MPLAB Integrated Development Environment together to solve problems.

**Procedure/Results**

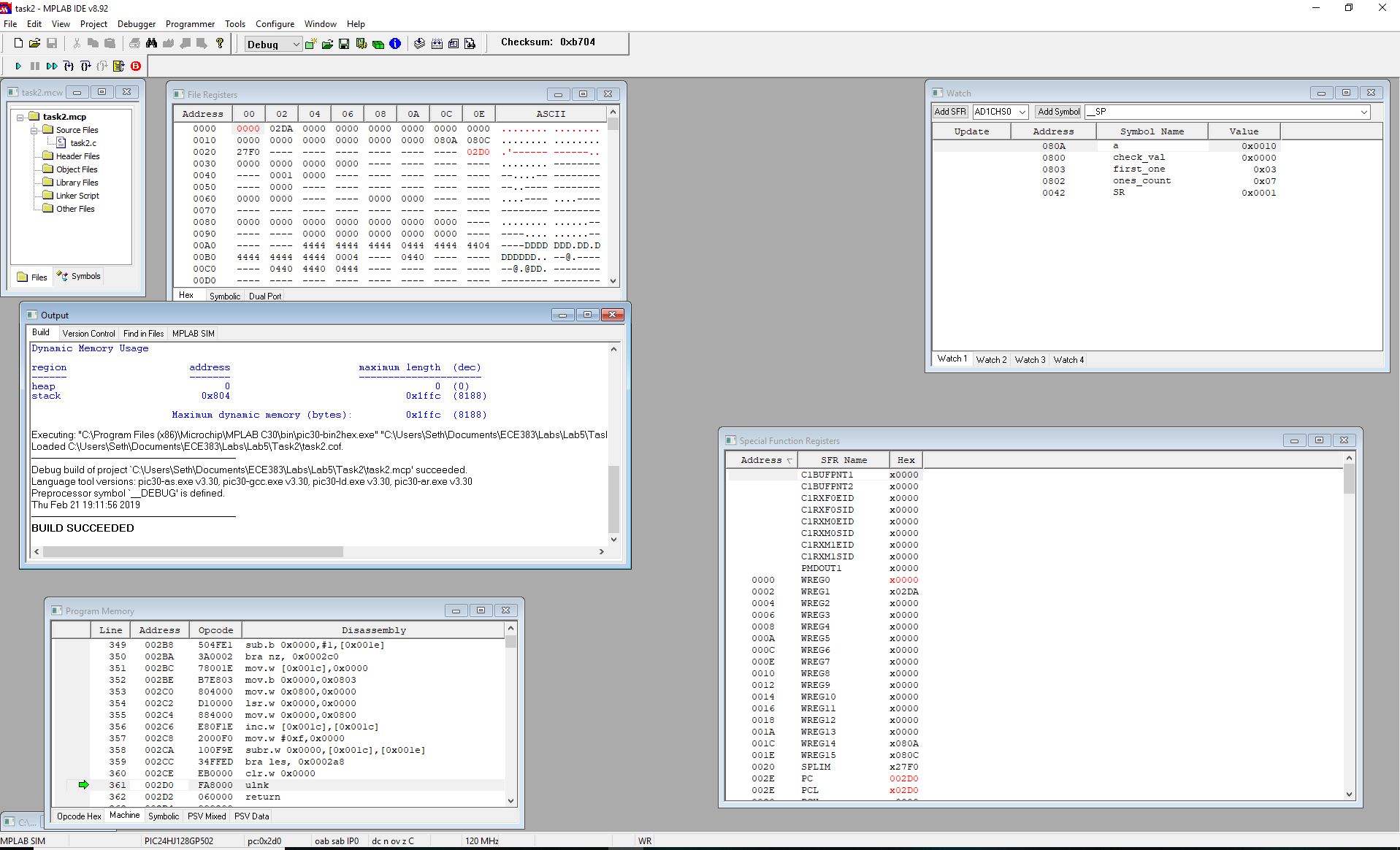
Task 1- Basic C arithmetic operations  
 For Exercise 1 of Lab 5, we first created a new project and new project file for use with the PIC24HJ128GP502 device in the MPLAB IDE. In the project we entered the C program provided in the lab document. After compiling the program, we watched the program memory, beginning in 0x200, data memory located in 0x800 of the file registers, and special function registers. Lastly, we opened the watch window, and after adding the SFR symbol ran 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. We were then able to see the value of the result and the value of the sign/negative (N), carry (C), zero (Z), and overflow (V) flags for all three of the arithmetic operations. Pictured below in Figure (1) is a screenshot of the final state of the program, data memory, and the watch window.

**Figure (1)**



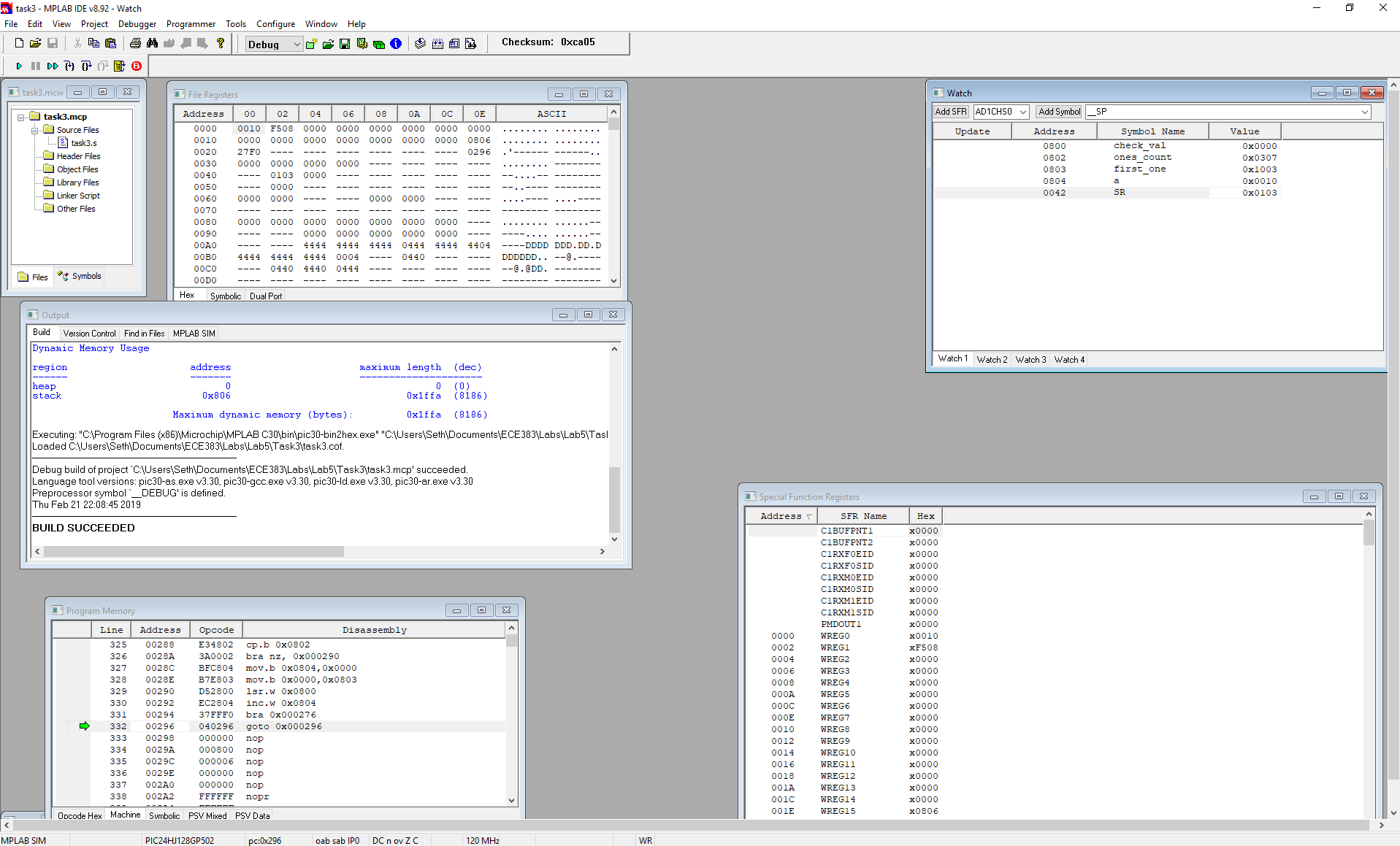
Task 2 – C Program “check\_val”  
 In Exercise 2, we created an MPLAB project and wrote a C language program containing three specialized variables. The variable check\_val counts the number of one bit in a 16-bit unsigned integer named. For the check\_val variable, the program also determines which is the first bit set. The variable ones\_count is an 8-bit unsigned variable in which the the count value should be stored. The variable first\_one is an 8-bit unsigned variable and the location where the first bit set should be stored. Once the C program was successfully compiled, we downloaded the program onto the PIC24 hardware successfully. Pictured below in Figure (2.1) is a screenshot of the final state of the program, data memory, and the watch window. Figure (2.2) shows our C language program with the aforementioned variables.

**Figure (2.1)**

****

**Figure (2.2) - APPENDIX**

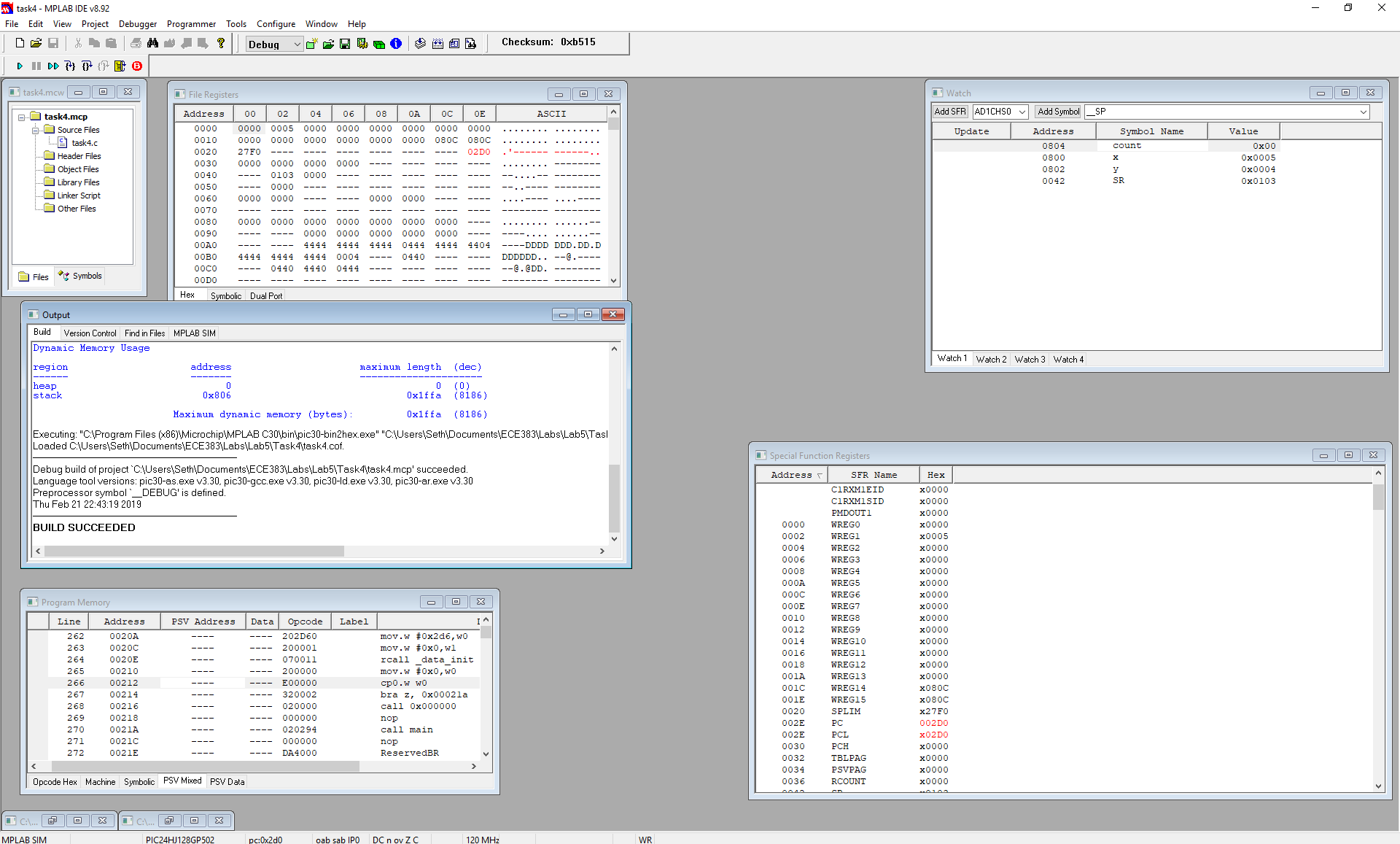
Task 3 – Assembly Language Program check\_val  
 In Exercise 3, we created an MPLAB project and wrote an assembly language program  
equivalent to the C language program in task 2. Just as in task 2, the three variables described in the lab document followed their given requirement. The variable check\_val counts the number of one bits in a 16-bit unsigned integer named. For the check\_val variable, the program also determines which is the first bit set. The variable ones\_count is an 8-bit unsigned variable in which the the count value should be stored. The variable first\_one is an 8-bit unsigned variable and the location where the first bit set should be stored. Once the assembly language program was successfully compiled, we downloaded the program onto the PIC24 hardware successfully. Pictured below in Figure (3.1) is a screenshot of the final state of the program, data memory, and the watch window. Figure (3.2) shows our C language program with the aforementioned variables.

**Figure (3.1)**

**Figure (3.2) - APPENDIX**

Task 4 – Assembly to C Example  
 In the 4th and final task of lab 5, we created an assembly project using the assembly language program provided in the lab document. Next, we executed the given program to obtain its results. We then wrote a C language program equivalent to the assembly language program provided in the lab document. Following a successful compile of the program, we downloaded our C language program onto the PIC24 device. Pictured below in Figure (4.1) is a screenshot of the final state of the program, data memory, and the watch window. Figure (4.2) shows our C language program.

**Figure (4.1)**

****

**Figure (4.2) - APPENDIX**

**Conclusion**

We are now confident in our ability to translate basic PIC24 assembly language into C  
programming language and C programming language to PIC24 assembly language, and simulate our results in the MPLAB Integrated Development Environment (IDE). In addition, we had success in downloading the C language programs and assembly language programs onto the PIC24 device in tasks 2, 3, and 4. The results of the simulations showed the equivalence of the C language programs and assembly language programs when translated. This lab provided an introduction C language and assembly language translation and allows for us to be confident in their use for programming the PIC24 device thereafter.

**Appendix**

**Task 2: Figure (2.2)**

#include "pic24\_all.h"  
//Declare Variables  
uint16 check\_val;  
uint8 ones\_count;  
uint8 first\_one;  
int main( void){  
//Set Variable Values  
check\_val = 0xF508;  
ones\_count = 0;  
first\_one = 0;  
int a; // for loop variable with C99  
//for loop to shift right and check ones values  
for ( a = 0; a<16; a++){  
 if (check\_val & 1){ //Compare check\_val to 0x0001 with bitwise and  
 ones\_count++; //increment ones\_count  
 if(ones\_count == 1) //check to see if this is first one  
 first\_one = a; // if this is first one, set first\_one to a  
 }  
 check\_val = check\_val >> 1; //Shift check\_val to the right  
}  
   
 return 0;  
 }

**Task 3: Figure(3.2)**

.include "p24Hxxxx.inc"  
  
 .global \_\_reset ;The label for the first line of code.   
  
 .bss ;uninitialized data section  
;;These start at location 0x0800 because 0-0x07FF reserved for SFRs  
check\_val: .space 2 ;Allocating space (in bytes) to variable.  
ones\_count: .space 1 ;Allocating space (in bytes) to variable.  
first\_one: .space 1 ;Allocating space (in bytes) to variable.  
a: .space 1 ;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  
;#include "pic24\_all.h"  
;  
;//Declare Variables  
;uint16 check\_val;  
;uint8 ones\_count;  
;uint8 first\_one;  
;  
;int main( void){  
;//Set Variable Values  
;check\_val = 0xF508;  
;ones\_count = 0;  
;first\_one = 0;  
;int a; // for loop variable with C99  
;  
;//for loop to shift right and check ones values  
;for ( a = 0; a<16; a++){  
;   
; if (check\_val & 1){ //Compare check\_val to 0x0001 with bitwise and  
; ones\_count++; //increment ones\_count  
; if(ones\_count == 1) //check to see if this is first one  
; first\_one = a; // if this is first one, set first\_one to a  
; }  
; check\_val = check\_val >> 1; //Shift check\_val to the right  
;}  
   
; return 0;  
; }  
  
  
  
  
;Set Variable Values  
  
;check\_val = 0xF508;  
mov #0xF508, w0 ; w0 = 0xF598  
mov wreg, check\_val ; check\_val = w0  
mov w0, w1 ; w1 = w0 (Saving value for later)  
  
;ones\_count = 0;  
mov #0, w0 ; w0 = 0  
mov.b wreg, ones\_count ; ones\_count = w0  
  
;first\_one = 0;  
mov.b wreg, first\_one ; first\_one = w0 (w0 is still = 0)  
  
;int a; // for loop variable with C99  
mov.b wreg, a ; a = w0 (w0 is still = 0)  
  
;//for loop to shift right and check ones values  
;for ( a = 0; a<16; a++){  
loop\_start:  
 mov #16, w0 ; w0 = 16  
 cp.b a ; compare a-w0  
 bra GEU, loop\_end ; If a > w0, goto loop\_end  
  
; if (check\_val & 1){ //Compare check\_val to 0x0001 with bitwise and  
 mov #1, w0 ; w0 = 1  
 and check\_val, wreg ; w0 = checkval & w0  
 cp w0, #1 ; compare w0 to 1 (w0-1)  
 bra NZ, END\_IF\_1 ; If comparison gives a zero,then we have a 1, otherwise go to END\_IF\_1  
  
; ones\_count++; //increment ones\_count  
 inc ones\_count ; ones\_count = ones\_count + 1  
  
; if(ones\_count == 1) //check to see if this is first one  
 mov #1, w0 ; w0 = 1  
 cp.b ones\_count ; ones\_count - w0 comparison  
 bra NZ, END\_IF\_2   
  
; first\_one = a; // if this is first one, set first\_one to a  
 mov.b a, wreg  
 mov.b wreg, first\_one  
  
 END\_IF\_2:  
; }  
END\_IF\_1:  
  
; check\_val = check\_val >> 1; //Shift check\_val to the right  
 lsr check\_val   
  
 inc a ; increment a   
 bra loop\_start ; return to top of loop  
;}  
loop\_end:   
; return 0;  
; }  
  
done:  
 goto done ;Place holder for last line of executed code  
  
.end ;End of program code in this file

**Task 4: Figure(4.2)**

#include "pic24\_all.h"  
  
uint16 x, y;  
uint8 count;  
  
int main(void){  
//mov #0x3, w0  
//mov.b wreg, count  
count = 3;  
  
//mov #0x1, w1  
//mov w1, x  
x = 1;  
  
//mov #0x3, w2  
//mov w2, y  
y = 3;  
  
//top:  
//cp0.b count  
//bra z, done  
for(count = 3; count > 0; count--){  
  
//cp w1, w2  
//bra nz, next  
//inc w2, w2  
//mov w2, y   
 if(x == y){  
 y++;  
 }  
  
//next:  
//cp w1, w2  
//bra GEU, next2  
//add #0x2, w1  
//mov w1, x  
 if( x < y){  
 x += 2;  
 }  
//dec.b count  
//bra top  
}  
  
  
  
  
return 0;  
}