**C and PIC24 Assembly Language Programming**

**5th Laboratory Report for ECE 383**

**Microcomputers**

**Submitted by**

**Yichen Huang**

**11906882**

**Shomari Thomas**

**11672867**

**The University of Alabama**

**Tuscaloosa, Alabama 35487**

**February 19, 2020**

Table of Contents

[Abstract 3](#_Toc33108853)

[Introduction 4](#_Toc33108854)

[Prelab 4](#_Toc33108855)

[Task 1- Basic C arithmetic operations 4](#_Toc33108856)

[Task 2 – C Program “check\_val” 5](#_Toc33108857)

[Procedure/Results 5](#_Toc33108858)

[Task 3 – Assembly Language Program check\_val 5](#_Toc33108859)

[Task 4 – Assembly to C Example 6](#_Toc33108860)

[Conclusion 6](#_Toc33108861)

[Appendixes 7](#_Toc33108862)

[Appendix A – Task 1 7](#_Toc33108863)

[Source Code – “task1.c” 7](#_Toc33108864)

[Figure (1 - 6) 8](#_Toc33108865)

[Appendix B – Task 2 12](#_Toc33108866)

[Source code – “task2.c” 12](#_Toc33108867)

[Figure (7 - 18) 12](#_Toc33108868)

[Appendix C – Task 3 21](#_Toc33108869)

[Source code – “task3.s” 21](#_Toc33108870)

[Figure (19 - 30) 24](#_Toc33108871)

[Appendix D – Task 4 32](#_Toc33108872)

[Source code – “task4.c” 32](#_Toc33108873)

[Figure (31 - 42) 33](#_Toc33108874)

# 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.

# Prelab

## 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. The source code and the pictures are in [Appendix A](#_Appendix_A_–).

## Task 2 – C Program “check\_val”

In Task 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 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. The source code and the pictures of the windows are in [Appendix B](#_Appendix_B_–).

# Procedure/Results

## 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 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 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. The source code and pictures are in [Appendix C](#_Appendix_C_–).

## 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. The source code and pictures are in [Appendix D](#_Appendix_D_–).

# 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.

# Appendixes

## Appendix A – Task 1

### Source Code – “task1.c”

#include "pic24\_all.h"

uint16 u16\_a, u16\_b, u16\_c, u16\_d;

uint8 u8\_x, u8\_y, u8\_z;

void main(void) {

u8\_x=0xFF;

u8\_y=0x01;

u16\_a = 0xFFFF;

u16\_b = 0x0001;

u8\_z=u8\_x+u8\_y;

u16\_d=(uint16) u8\_x + (uint16) u8\_y;

u16\_c=u16\_a+u16\_b;

}

### Figure (1 - 6)

A screenshot of a social media post

Description automatically generated

Figure 1 Output Window

A screenshot of a social media post

Description automatically generated

Figure 2 task1.c

A screenshot of a cell phone

Description automatically generated

Figure 3 Watch Window

A screenshot of a cell phone

Description automatically generated

Figure 4 File Registers

A screenshot of a cell phone

Description automatically generated

Figure 5 Special Function Registers

A screenshot of a cell phone

Description automatically generated

Figure 6 Program Memory

## Appendix B – Task 2

### Source code – “task2.c”

#include "pic24\_all.h"

uint16 check\_val = 0;

uint8 ones\_count = 0, first\_one = 0, count = 0, set = 0;

void main (void) {

for(check\_val = 0xF508;check\_val != 0; check\_val = check\_val / 2) {

if (check\_val & 0x0001 == 0x0001) {

ones\_count++;

if(set == 0) {

first\_one = count;

set = 1;

}

}

count++;

}

}

### Figure (7 - 18)

Simulated (7 - 12)

A screenshot of a social media post

Description automatically generated

Figure 7 Output Window

A screenshot of a social media post

Description automatically generated

Figure 8 task2.c

A screenshot of a cell phone

Description automatically generated

Figure 9 Watch Window

A screenshot of a cell phone

Description automatically generated

Figure 10 File Registers

A screenshot of a cell phone

Description automatically generated

Figure 11 Program Memory

A screenshot of a cell phone

Description automatically generated

Figure 12 Special Function Register

On PIC24 Microcontroller (13 - 18)

A screenshot of a cell phone

Description automatically generated

Figure 13 Output Window

A screenshot of a social media post

Description automatically generated

Figure 14 task2.c

A screenshot of a cell phone

Description automatically generated

Figure 15 Watch Window

A screenshot of a computer

Description automatically generated

Figure 16 File Registers

A screenshot of a social media post

Description automatically generated

Figure 17 Program Memory

A screenshot of a cell phone

Description automatically generated

Figure 18 Special Function Registers

## Appendix C – Task 3

### Source code – “task3.s”

.include "p24Hxxxx.inc"

.global \_\_reset

.bss

check\_val: .space 2

ones\_count: .space 1

first\_one: .space 1

count: .space 1

set: .space 1

.text

\_\_reset:

mov #\_\_SP\_init, w15

mov #\_\_SPLIM\_init,W0

mov W0, SPLIM

;-----------------------C code--------------------------------

;uint16 check\_val = 0;

;uint8 ones\_count = 0, first\_one = 0, count = 0, set = 0;

;void main (void) {

; for(check\_val = 0xFFFF;check\_val != 0; check\_val = check\_val / 2) {

; if (check\_val & 0x0001 == 0x0001) {

; ones\_count++;

; if(set == 0) {

; first\_one = count;

; set = 1;

; }

; }

; count++;

; }

;}

;----------------------code start here----------------------

; check\_val = 0xF508

mov #0xF508, w0

mov wreg, check\_val

;ones\_count = 0

clr.b ones\_count

;first\_one = 0

clr.b first\_one

;count = 0

clr.b count

;set = 0

clr.b set

top:

mov check\_val w0;

bra Z, end\_loop;

and #0x0001, w0;

bra Z, end\_if1;

inc.b ones\_count

mov.b set, wreg

bra NZ, end\_if2

mov.b count, wreg

mov.b wreg, first\_one

mov.b #0x01, w0

mov.b wreg, set

end\_if2:

end\_if1:

inc.b count

lsr check\_val

bra top

end\_loop:

done: goto done

.end

### Figure (19 - 30)

Simulated (19 - 24)

A screenshot of a social media post

Description automatically generated

Figure 19 Output Window

A screenshot of a social media post

Description automatically generated

Figure 20 task3.s

A screenshot of a cell phone

Description automatically generated

Figure 21 watch window

A screenshot of a computer

Description automatically generated

Figure 22 File Register

A screenshot of a cell phone

Description automatically generated

Figure 23 Special Function Registers

A screenshot of a cell phone

Description automatically generated

Figure 24 Program Memory

On PIC24 Microcontroller (25 - 30)

A screenshot of a cell phone

Description automatically generated

Figure 25 Output Window

A screenshot of a computer

Description automatically generated

Figure 26 task3.s

A screenshot of a computer

Description automatically generated

Figure 27 Watch Window

A screenshot of a computer

Description automatically generated

Figure 28 File Registers

A screenshot of a cell phone

Description automatically generated

Figure 29 Special Function Registers

A screenshot of a social media post

Description automatically generated

Figure 30 Program Memory

## Appendix D – Task 4

### Source code – “task4.c”

#include "pic24\_all.h"

uint16 x,y;

uint8 count;

void main(void) {

x = 1;

y = 3;

for(count = 3; count>0; count--) {

if((x-y) == 0) {

y++;

}

if (x < y) {

x = x + 2;

}

}

}

### Figure (31 - 42)

Simulated (31 - 36)

A screenshot of a cell phone

Description automatically generated

Figure 31 Output Window

A screenshot of a cell phone

Description automatically generated

Figure 32 task4.c

A screenshot of a social media post

Description automatically generated

Figure 33 watch window

A screenshot of a cell phone

Description automatically generated

Figure 34 File Registers

A screenshot of a cell phone

Description automatically generated

Figure 35 Special Function Register

A screenshot of a social media post

Description automatically generated

Figure 36 Program Memory

On PIC24 Microcontroller (37 - 42)

A screenshot of a social media post

Description automatically generated

Figure 37 Output Window

A screenshot of a cell phone

Description automatically generated

Figure 38 task4.c

A screenshot of a cell phone

Description automatically generated

Figure 39 Watch Window

A screenshot of a computer

Description automatically generated

Figure 40 File Registers

A screenshot of a cell phone

Description automatically generated

Figure 41 Special Function Registers

A screenshot of a social media post

Description automatically generated

Figure 42 Program Memory