An Intro. to Assembly Instructions

EE 310/EE310L - Mcirocontrller - Spring 2023

SONOMA STATE UNIVERSITY ELECTRICAL ENGINEERING

NAME______Ryan Chan_______ DATE: ____2/23/2025____

Assignment # 3

https://docs.google.com/document/d/1C3rr_gcK2FCCMlxZd8K1UOKlxvd7T93F/edit?usp=sharing&ouid=111229422470150013614&rtpof=true&sd=true

1. Assignment Overview

In this experiment we start learning about the architecture of PIC18F and are introduced to some basic assembly instructions.

2. Learning Objectives

By the end of this lab, you will

- Understand PIC architecture
- Basic PIC assembly instructions

3. Review

Consider reviewing the following before you start this assignment (must be logged on SSU to open the links):

- Review the assembly instruction command list.
- Review Mazidi- Chapter 2
- Full datasheet for the 40PIN PDIP PIC18F46K42 https://ww1.microchip.com/downloads/en/DeviceDoc/PIC18(L)F26-27-45-46-47-55-56-57K42-Data-Sheet-40001919G.pdf

4. Materials

You need the following to complete this assignment:

- PIC Simulator - Download OshoSoftware PIC Simulator. This is free software but you can only open it 30 times. The duration of each session is 1 hour. NOTE: The simulator works only on PC. You can either use the machines in the lab or use Virtualization for MacOS. We will be using the simulator for only the first two weeks of the course.

5. Background Information

In the previous assignment, we looked at the WREG register of the PIC. We demonstrate the use of one of the most widely used registers of the PIC with simple instructions such as MOVE. In this assignment, we learn more about assembly instructions for PIC18F. We look at some widely used Assembly language directives, pseudocode, and data types related to the PIC.

6. Assignment

Complete the following steps.

6.1 Examine this code

Cafully, examine the code below. Note that it ends with END. Type the code in your simulator. When you compile the code, refer to the program memory section. Clearly review the way the code has been placed in the program memory. In case there are any syntax errors in the program below, fix them and make sure the program compiles. Answer the following questions.

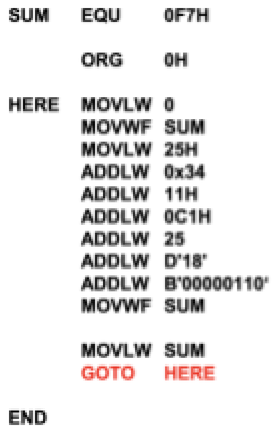
1- The command GOTO HERE, what does HERE refer to?

HERE refers to the command MOVLW 0.

2- When the program completes the first loop and reaches GOTO command, what would be the value of WREG?

Value of WREG is 0xF7.

3- Complete the table below, as the program goes through the first 5 loops.



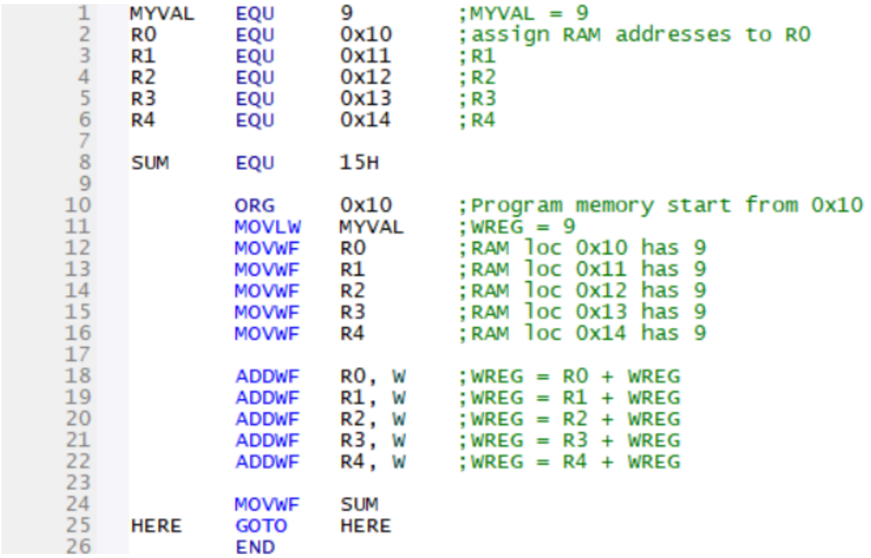
```
SUM     EQU     0F7H

        ORG     0H

HERE    MOVLW   0
        MOVWF   SUM
        MOVLW   25H
        ADDLW   0x34
        ADDLW   11H
        ADDLW   0C1H
        ADDLW   25
        ADDLW   D'18'
        ADDLW   B'00000110'
        MOVWF   SUM

        MOVLW   SUM
        GOTO    HERE

END
```

Loop Number	Value in W	Value in REG(F7)
1	F7	5C
2	F7	5C
3	F7	5C
4	F7	5C
5	F7	5C

Sample Code	Complete the table.

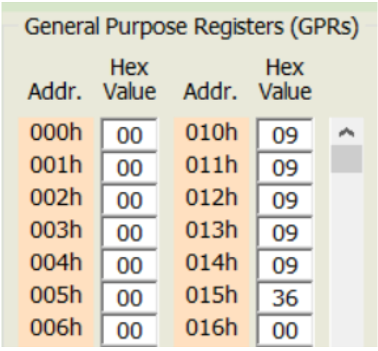
6.2 Find the SUM

Using the provided example code below, write a program to place a value, say MYVAL=9, into REG 0x10-0x14, then add all 5 registers together and then place the result in a register called SUM in location 0x15. Your program must start from location 0x10 in the program memory. When the program ends, use GOTO to stop the program. Take a snapshot of your result in register SUM. NOTE: Check your program and make sure it works for any value assigned to MYVAL.



```
MYVAL    EQU     9        ;MYVAL = 9
R0       EQU     0x10     ;assign RAM addresses to R0
R1       EQU     0x11     ;R1
R2       EQU     0x12     ;R2
R3       EQU     0x13     ;R3
R4       EQU     0x14     ;R4

SUM      EQU     15H

         ORG     0x10     ;Program memory start from 0x10
         MOVLW   MYVAL    ;WREG = 9
         MOVWF   R0       ;RAM loc 0x10 has 9
         MOVWF   R1       ;RAM loc 0x11 has 9
         MOVWF   R2       ;RAM loc 0x12 has 9
         MOVWF   R3       ;RAM loc 0x13 has 9
         MOVWF   R4       ;RAM loc 0x14 has 9

         ADDWF   R0, W    ;WREG = R0 + WREG
         ADDWF   R1, W    ;WREG = R1 + WREG
         ADDWF   R2, W    ;WREG = R2 + WREG
         ADDWF   R3, W    ;WREG = R3 + WREG
         ADDWF   R4, W    ;WREG = R4 + WREG

         MOVWF   SUM
HERE     GOTO    HERE
         END
```

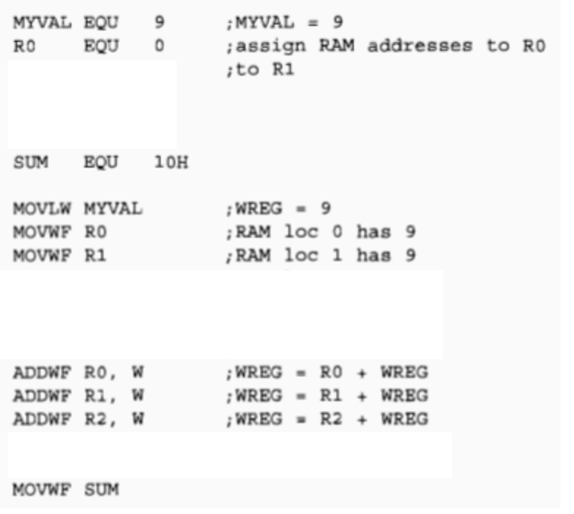
Snapshot of your code



General Purpose Registers (GPRs)

	Hex		Hex
Addr.	Value	Addr.	Value
000h	00	010h	09
001h	00	011h	09
002h	00	012h	09
003h	00	013h	09
004h	00	014h	09
005h	00	015h	36
006h	00	016h	00

Snapshot of your results showing register 0x10-0x16

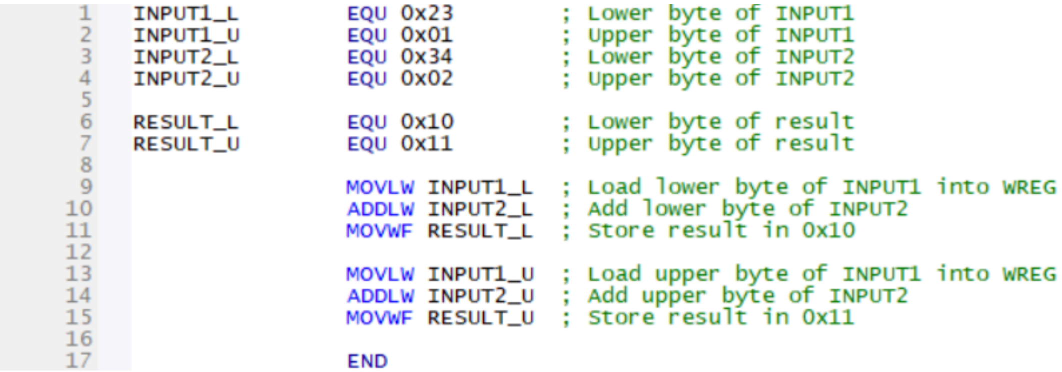


```
MYVAL  EQU    9       ;MYVAL = 9
R0     EQU    0       ;assign RAM addresses to R0
                      ;to R1



SUM    EQU    10H

MOVLW  MYVAL          ;WREG = 9
MOVWF  R0             ;RAM loc 0 has 9
MOVWF  R1             ;RAM loc 1 has 9



ADDWF  R0, W          ;WREG = R0 + WREG
ADDWF  R1, W          ;WREG = R1 + WREG
ADDWF  R2, W          ;WREG = R2 + WREG


MOVWF  SUM
```

Sample code.

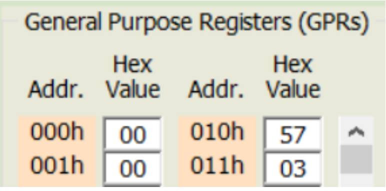
6.3 Adding two 16-bit values with no carry

Assume INPUT1 = 0x123, INPUT2 = 0x234. Add the two inputs and place the results in registers 0x10 (lower byte) and 0x11 (upper byte) - Show your code and results in the appropriate registers. HINT: In this case, you need to add the lower bytes together and place the results in register 0x10. Then you need to add the upper bytes together and place the results in register 0x10. Note that in this case, the assumption is that the sum of the lower bytes will never generate a CARRY.



```
INPUT1_L        EQU 0x23        ; Lower byte of INPUT1
INPUT1_U        EQU 0x01        ; Upper byte of INPUT1
INPUT2_L        EQU 0x34        ; Lower byte of INPUT2
INPUT2_U        EQU 0x02        ; Upper byte of INPUT2

RESULT_L        EQU 0x10        ; Lower byte of result
RESULT_U        EQU 0x11        ; Upper byte of result

                MOVLW INPUT1_L  ; Load lower byte of INPUT1 into WREG
                ADDLW INPUT2_L  ; Add lower byte of INPUT2
                MOVWF RESULT_L  ; Store result in 0x10

                MOVLW INPUT1_U  ; Load upper byte of INPUT1 into WREG
                ADDLW INPUT2_U  ; Add upper byte of INPUT2
                MOVWF RESULT_U  ; Store result in 0x11

                END
```

Snapshot of your code



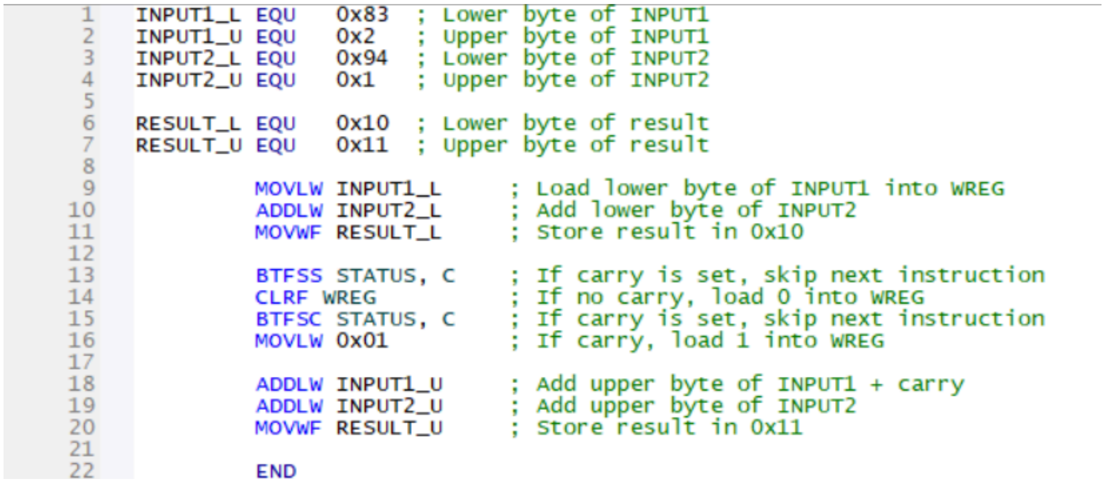
General Purpose Registers (GPRs)

Addr.	Hex Value	Addr.	Hex Value
000h	00	010h	57
001h	00	011h	03

Snapshot of your result

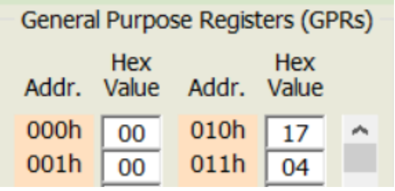
6.4 Adding two 16-bit values with carry

Assume Assume INPUT1 = 0x283, INPUT2=0x194. Add the two inputs and place the results in registers 0x10 (lower byte) and 0x11 (upper byte) - Show your code and results in the appropriate registers. Make sure your program works for any unsigned INPUTs.



```
INPUT1_L EQU   0x83  ; Lower byte of INPUT1
INPUT1_U EQU   0x2   ; Upper byte of INPUT1
INPUT2_L EQU   0x94  ; Lower byte of INPUT2
INPUT2_U EQU   0x1   ; Upper byte of INPUT2

RESULT_L EQU   0x10  ; Lower byte of result
RESULT_U EQU   0x11  ; Upper byte of result

         MOVLW INPUT1_L      ; Load lower byte of INPUT1 into WREG
         ADDLW INPUT2_L      ; Add lower byte of INPUT2
         MOVWF RESULT_L      ; Store result in 0x10

         BTFSS STATUS, C     ; If carry is set, skip next instruction
         CLRF WREG           ; If no carry, load 0 into WREG
         BTFSC STATUS, C     ; If carry is set, skip next instruction
         MOVLW 0x01          ; If carry, load 1 into WREG

         ADDLW INPUT1_U      ; Add upper byte of INPUT1 + carry
         ADDLW INPUT2_U      ; Add upper byte of INPUT2
         MOVWF RESULT_U      ; Store result in 0x11

         END
```

Snapshot of your code



General Purpose Registers (GPRs)

Addr.	Hex Value	Addr.	Hex Value
000h	00	010h	17
001h	00	011h	04

Snapshot of your result

6.5 Answer the following questions

Make sure your answers are clearly visible.

Review the datasheet for PIC18(L)F46K42 ;
(https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/DataSheets/40001861B.pdf)

Part 1 - Referring to Table 1 (don't forget the units)

1. How much Program flash memory is available?
   64 KB
2. How much Data EEPROM is available?
   1024 B
3. How many IO Pins does it have?
   36 I/O Pins
4. Does it have a DAC?
   Yes
5. Does it have an Analog to Digital converter?
   Yes

6. How many comparators does it have?
   2

Part II – Referring to Figure 4

1. How many bits does port E have?
   3 bits
2. How many bits does port D have?
   8 bits

Part III

1. Referring to Table 4, ANA0 is one of the inputs to the ADC. Which pin is ANA0 for a 40 – PIN PDIP?
   PIN 2
2. Referring to Table 4, MCLR is the reset pin. Which pin is MCLR for a 40 – PIN PDIP?
   PIN 1
3. Referring to Table 4, how many GRND pins (also noted as VDD) is available to a 40 – PIN PDIP? What are the pins?
   2 GRND pins, pin 11 and pin 32
4. Referring to Table 4, how many VCC pins (also noted as VSS) are available to a 40 – PIN PDIP? What are the pins?
   2 VCC pins, pin 12 and pin 31

7. Survey Questions

Answer the following questions, please:

Survey question	Response
On a scale of 1-10 how did you like this exercise? (10 is the best, 1 is the worst)	6
On a scale of 1-10 how much did you learn as a result of completing this exercise? (10 = plenty; 1=very little)	4
How many hours did you spend completing this exercise?	5

8. References

[1] Complete Electronics Self-Teaching Guide with Projects | Earl Boysen, Harry Kybett. ISBN: 978-1-118-28232-8 July 2012