Facultad de ingeniería

Materia: Laboratorio de Microcomputadoras

REPORTE DE LA PRÁCTICA 1

Título: Introducción a la programación del microcontrolador PIC16F877; “Direccionamiento Directo”

Integrantes:

- Martínez Pérez Brian Erik - 319049792
- Nuñez Rodas Abraham Enrique - 114003546
- Vicenteño Maldonado Jennifer Michel - 317207251

Profesor: Moises Melendez Reyes

Grupo: 1

Fecha de Entrega: 2 de marzo de 2025

Semestre: 2025-2

Objetivo: Familiarizar al alumno en el conocimiento del ensamblador, el simulador, el conjunto de instrucciones de un microcontrolador y ejecutar programas en tiempo de simulación.

Ejercicios

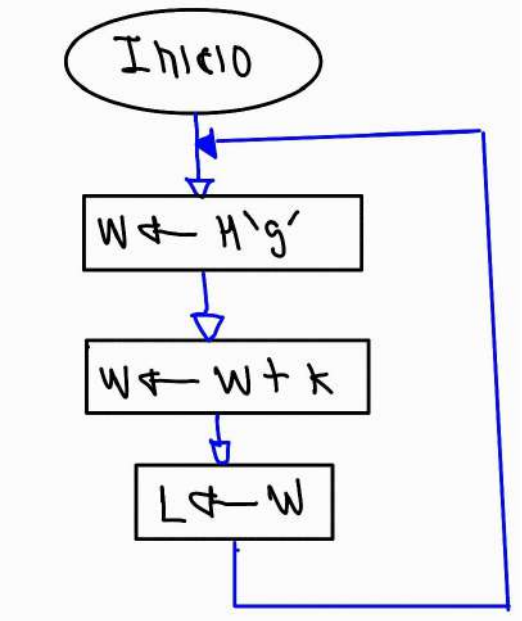
Ejercicio 1: Siguiendo las indicaciones previas, escribir el siguiente programa, ensamblar y simular el funcionamiento de este.

```
        PROCESSOR 16f877
        INCLUDE <p16f877.inc>

K       equ     H'26'
L       equ     H'27'

                ORG   0
                GOTO  INICIO

                ORG 5
INICIO:         MOVLW H'05'
                ADDWF  K,0
                MOVWF  L
                GOTO  INICIO
                END
```

Ingresar un dato de 8 bits a la dirección reservada a la variable K y ejecutar la simulación del programa utilizando diferentes valores.

K	L
66	**6B**

Diagrama de flujo:

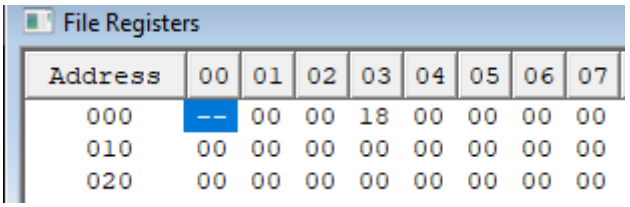


Funcionamiento de la solución:

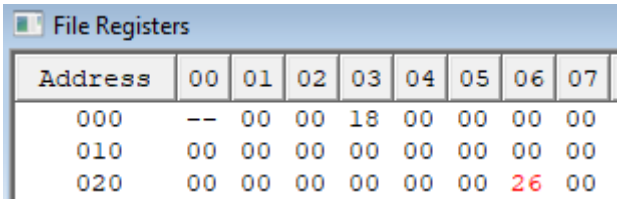
Código:

```
PROCESSOR 16f877
INCLUDE <p16f877.inc>
K equ H'26'
L equ H'27'
    ORG 0
    GOTO INICIO ; SALTO

    ORG 5
INICIO:
    MOVLW H'05'
    ADDWF K,0
    MOVWF L
    GOTO INICIO
    END
```

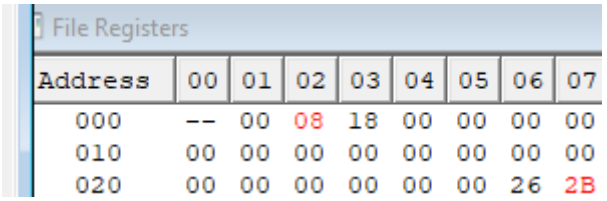
prueba 1 - secuencia 1:



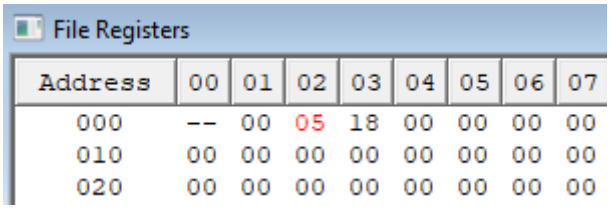
prueba 1 - secuencia 2:



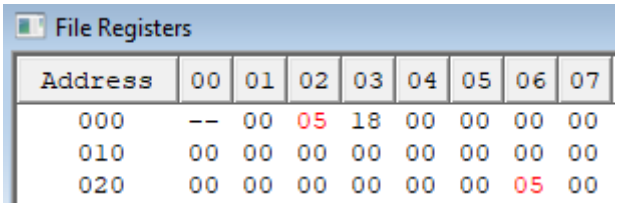
prueba 1 - secuencia 3:



prueba 2 - secuencia 1:



prueba 2 - secuencia 2:



prueba 2 - secuencia 3:

File Registers

Address	00	01	02	03	04	05	06	07
000	--	00	05	18	00	00	00	00
010	00	00	00	00	00	00	00	00
020	00	00	00	00	00	00	05	0A

En este programa se realiza una suma de un valor constante en este primer caso “5” al contenido de la variable K, que es “26” y almacena el resultado en L, obteniendo “2B”.

Análisis: Para este primer ejercicio se realizó un pequeño algoritmo donde se implementa la suma de dos números un primer operando constante y otro operando variable que colocamos en memoria, al final el resultado se colocará en la variable “L”.

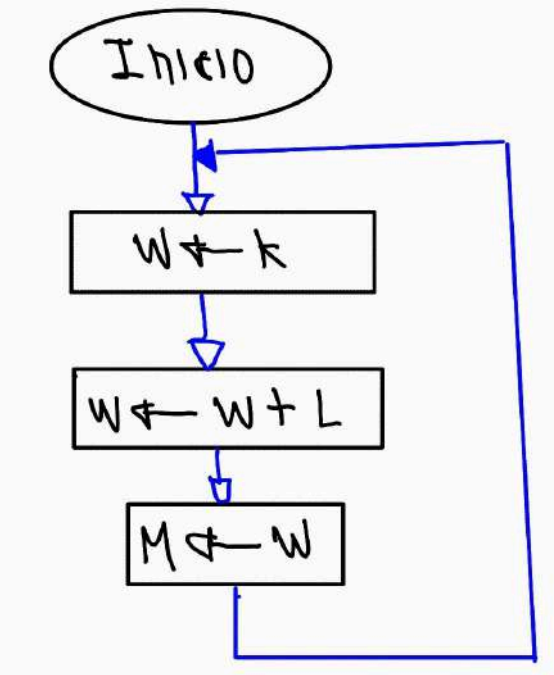
Ejercicio 2: Escribir, ensamblar y ejecutar el siguiente programa:

```
        PROCESSOR 16f877
        INCLUDE <p16f877.inc>

K       equ     H'26'
L       equ     H'27'
M       equ     H'28'

                ORG  0
                GOTO INICIO

                ORG 5
INICIO:         MOVF K,W
                ADDWF L,0
                MOVWF M
                GOTO INICIO
                END
```

a. Comentar e indicar que hace el programa
b. Realizar la ejecución con diferentes valores en K y L
c. Revisar el valor que se genera en la bandera C

K	L	M
50	55	**A5**

Diagrama de flujo:



Funcionamiento de la solución:

Código:

```
processor 16f877
include <p16f877.inc>

K equ H'26'
L equ H'27'
M equ H'28'
     ORG 0
     GOTO INICIO
     ORG 5
INICIO:
     MOVF K,W
     ADDWF L,0
     MOVWF M
     GOTO INICIO
     END
```

prueba 1 - secuencia 1:

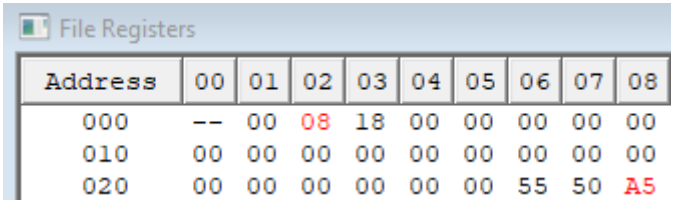
Address	00	01	02	03	04	05	06	07	08
000	--	00	05	18	00	00	00	00	00
010	00	00	00	00	00	00	00	00	00
020	00	00	00	00	00	00	00	00	00

prueba 1 - secuencia 2:

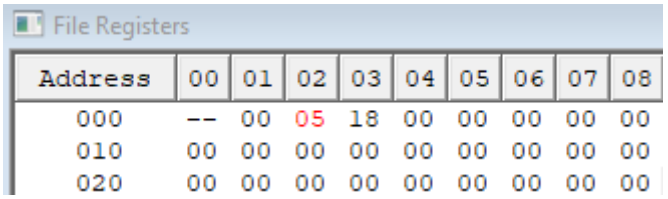
File Registers

Address	00	01	02	03	04	05	06	07	08
000	--	00	05	18	00	00	00	00	00
010	00	00	00	00	00	00	00	00	00
020	00	00	00	00	00	00	55	50	00

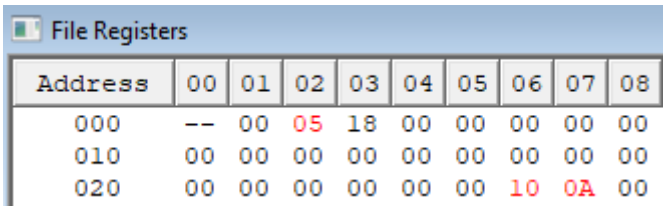
prueba 1 - secuencia 3:



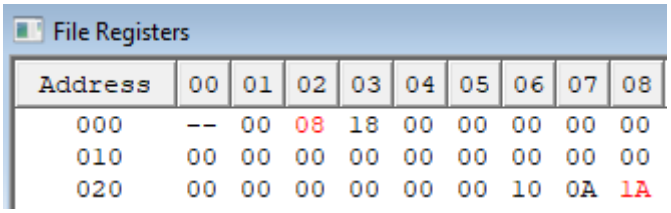
prueba 2 - secuencia 1:



prueba 2 - secuencia 2:



prueba 2 - secuencia 3:



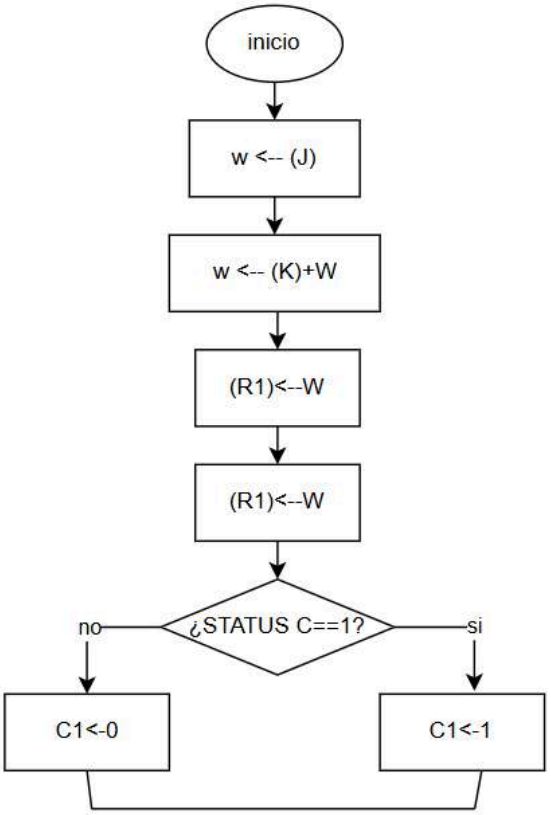
Este código suma el contenido de “K”, que guarda 55 y “L”, que guarda 50, y almacena el resultado en “M”, al final obteniendo como resultado A5.

Análisis: Para este segundo ejercicio se realizó un algoritmo donde se implementa la suma de dos números un primer operando variable y otro operando también variable que colocamos en memoria al momento de ejecución, al final el resultado se colocará en la variable “L”.

Ejercicio 3: Modificar el programa anterior, para que ahora los datos que operará se encuentren en las localidades J y K respectivamente, el resultado almacenarlo en otras direcciones, reservadas para C1 y R1; C1 representa el valor de la bandera de acarreo y R1 el resultado.
Un ejemplo de datos y del resultado de la suma es:

J	K	C1	R1
FF	FF	01	FE

Diagrama de flujo:

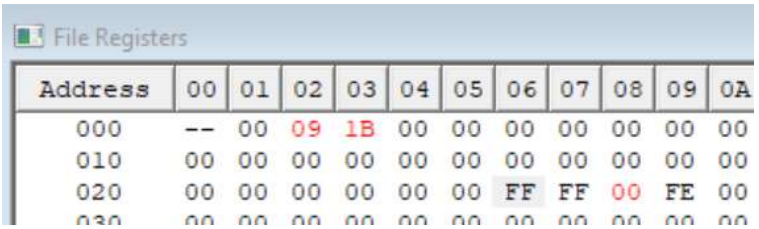


Funcionamiento de la solución:

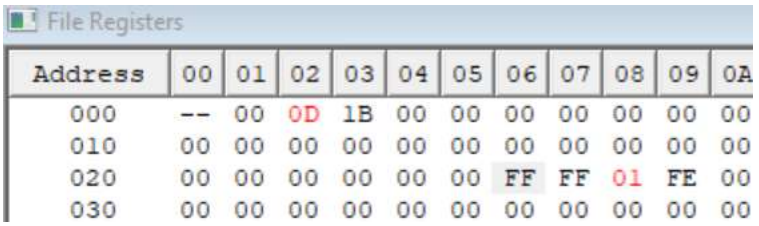
Código

```
processor 16f877
include <p16f877.inc>

    J equ H'26'
    K equ H'27'
    CL equ H'28'
    R1 equ H'29'
    ORG 0
    GOTO INICIO
    ORG 5
INICIO:
    MOVF J,0     ;W<-(J)
    ADDWF K,0    ;W<- (K)+W
    MOVWF R1     ;(R1)<-W
    MOVWF R1     ;(R1)<-W
    BTFSS STATUS, C ;¿STATUS C==1?
    GOTO CERO       ;C=0
UNO:
    MOVLW 0X01   ;W<-1
    MOVWF CL     ;C1<-W
CERO:
    CLRF CL

END
```

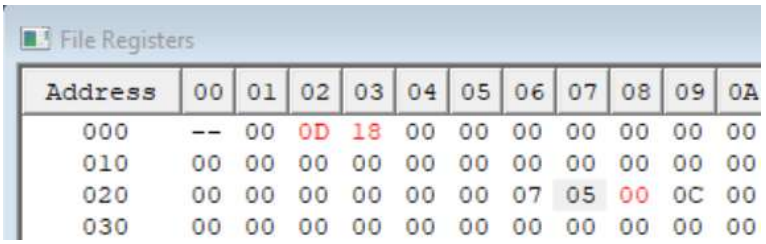
prueba 1 - secuencia 1



prueba 1-secuencia 2



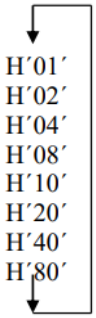
prueba 2 - secuencia 1



Análisis:

Este código coloca en W el valor de J, después le suma K y lo almacena en el R1, después revisa el estado de la bandera C y coloca C1 en 1 si el valor es igual a 1, si no coloca 0.

Ejercicio 4: Realice un programa que ejecute la siguiente secuencia, misma que deberá ver en la dirección de memoria (registro) de su elección.
Secuencia:

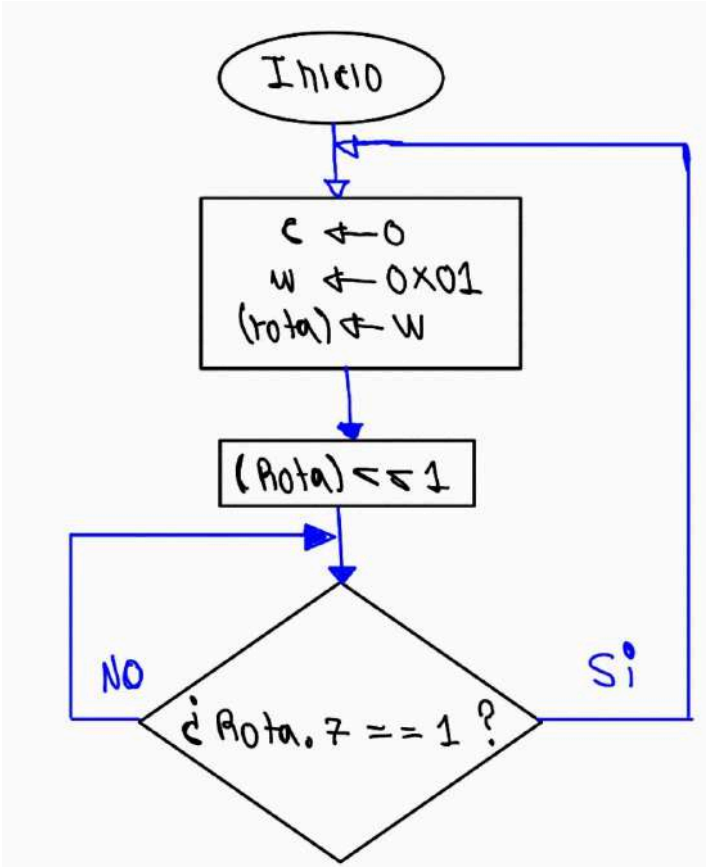


Donde H´01´ indica que el dato está dado en hexadecimal. En caso de seleccionar el registro cuya dirección es 0X20

DIR	20	20	20	20	20	20	20	20
DATO	**01**	**02**	**04**	**08**	**10**	**20**	**40**	**80**

Nota: La secuencia indicada, deberá mostrarse en una misma dirección de memoria.

Diagrama de flujo:



Funcionamiento de la solución:

Código:

```
processor 16f877
include <p16f877.inc>

DIR EQU H'20'
    ORG 0
    GOTO INICIO
    ORG 5
    BCF STATUS, C
INICIO:
    MOVLW 1
    MOVWF DIR
CICLO:
    RLF DIR,0
    MOVWF DIR
    BTFSS DIR, 7
    GOTO CICLO
    GOTO INICIO
END
```

secuencia 1:

File Registers

Address	00	01	02
000	--	00	04
010	00	00	00
020	01	00	00

secuencia 2:

File Registers

Address	00	01	02
000	--	00	06
010	00	00	00
020	02	00	00

secuencia 3:

File Registers

Address	00	01	02
000	--	00	06
010	00	00	00
020	04	00	00

secuencia 4:

File Registers

Address	00	01	02
000	--	00	06
010	00	00	00
020	08	00	00

secuencia 5:

File Registers

Address	00	01	02
000	--	00	06
010	00	00	00
020	10	00	00

secuencia 6:

File Registers

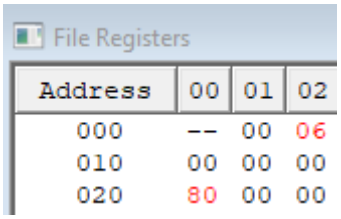
Address	00	01	02
000	--	00	06
010	00	00	00
020	20	00	00

secuencia 7:

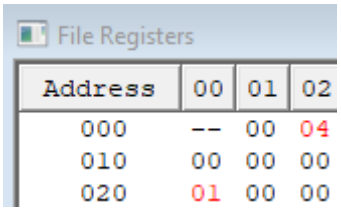
File Registers

Address	00	01	02
000	--	00	06
010	00	00	00
020	40	00	00

secuencia 8:



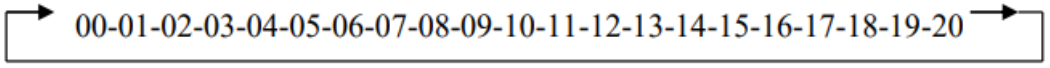
secuencia 9:



Comenzamos definiendo “DIR” la cual guarda la dirección de memoria donde se ejecutaron los cambios de número tal como se muestra en la secuencia. limpiamos la bandera “C” de acarreo, para no modificar el número deseado, inmediatamente colocamos el numero 1 en “DIR”, el cual ocupa la localidad H“020”, posteriormente realizamos el corrimiento hacia la izquierda, este proceso se realizará hasta que en el bit más significativo se encuentre el 1, para verificar esta instrucción utilizamos la instrucción “BTFSS DIR, 7”, si esta condición se cumple, se reinicia el proceso con la instrucción “GOTO CICLO“.

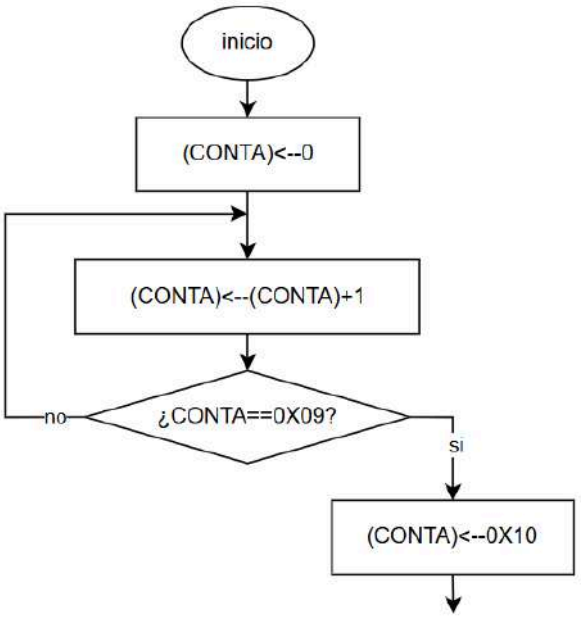
Análisis: Para este ejercicio se tomó en cuenta cómo obtener los números que se piden en la secuencia, para ello supimos que se obtenían sólo manteniendo el número 1 en el bit más significativo para un número de 8 bits, además de que al inicio se tenía que forzar a tener la bandera de acarreo apagada para evitar que se modificarán los números deseados.

Ejercicio 5: Desarrollar un programa que presente la cuenta en numeración decimal en la localidad de memoria de su elección, como se indica a continuación



Nota: La secuencia indicada, deberá mostrarse en una misma dirección de memoria, tal como fue realizado en el ejercicio anterior.

Diagrama de flujo:



Funcionamiento de la solución:

```
processor 16f877
include <p16f877.inc>

CONT     equ H'30'  ; Define la variable CONT en la dirección 0x30

        org 0
        goto begin  ; Salta a la etiqueta 'begin'
        org 5

begin:
        clrf CONT        ; Inicializa CONT en 0

incl:
        incf CONT, 1     ; Incrementa CONT en 1
        movlw H'09'      ; Carga 0x09 en W
        xorwf CONT, 0    ; Compara CONT con 0x09
        btfss STATUS, Z  ; Si CONT es 0x09, salta la siguiente instrucción
        goto incl        ; Si no es 0x09, repite el ciclo
        movlw H'10'      ; Carga 0x10 en W
        movwf CONT       ; Asigna 0x10 a CONT

inc2:
        incf CONT, 1     ; Incrementa CONT en 1
        movlw H'19'      ; Carga 0x19 en W
        xorwf CONT, 0    ; Compara CONT con 0x19
        btfss STATUS, Z  ; Si CONT es 0x19, salta la siguiente instrucción
        goto inc2        ; Si no es 0x19, repite el ciclo
        movlw H'20'      ; Carga 0x20 en W
        movwf CONT       ; Asigna 0x20 a CONT
        goto begin       ; Reinicia el proceso

end
```

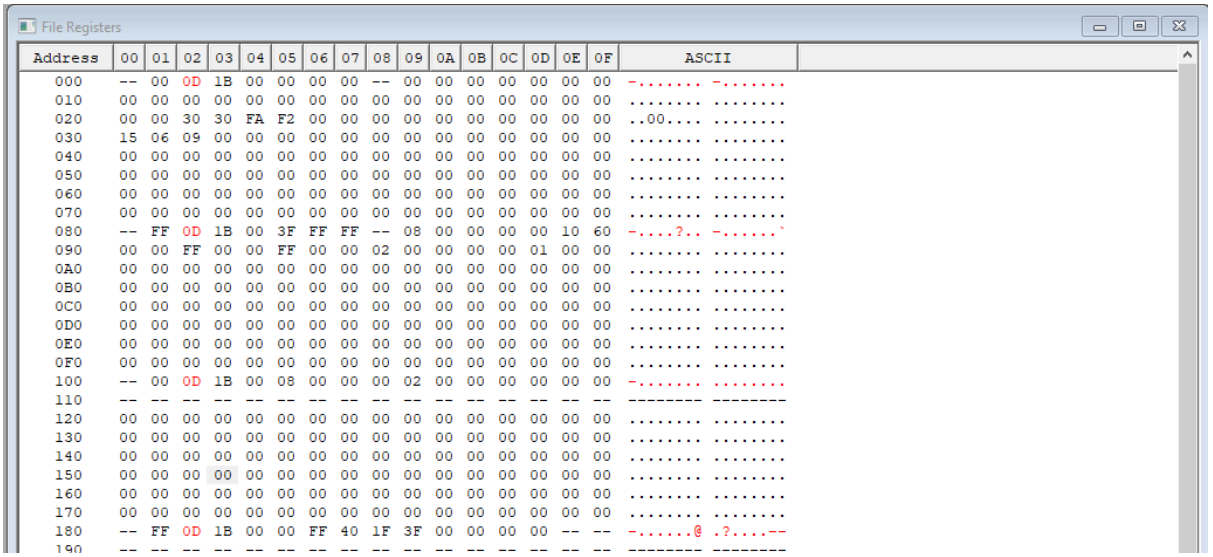
Secuencias:

File Registers

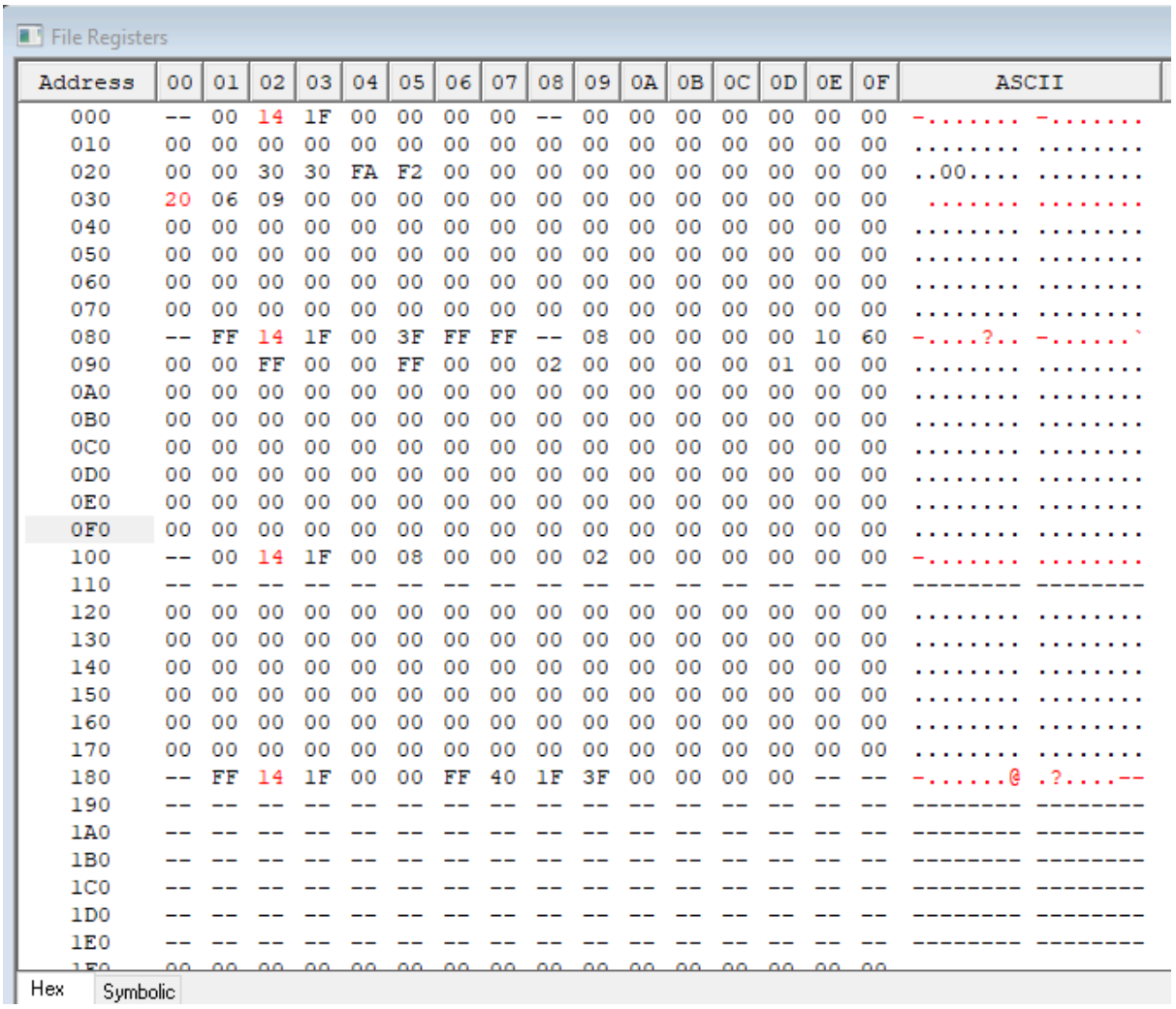
Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	06	1F	00	00	00	00	--	00	00	00	00	00	00	00	-....... -.......
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
020	00	00	30	30	FA	F2	00	00	00	00	00	00	00	00	00	00	..00.... ....
030	00	06	09	00	00	00	00	00	00	00	00	00	00	00	00	00	....
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
080	--	FF	06	1F	00	3F	FF	FF	--	08	00	00	00	00	10	60	-....?.. -.....`
090	00	00	FF	00	00	FF	00	00	02	00	00	00	00	01	00	00	....
0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
100	--	00	06	1F	00	08	00	00	00	02	00	00	00	00	00	00	-....... ....
110	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	-------- --------
120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
180	--	FF	06	1F	00	00	FF	40	1F	3F	00	00	00	00	--	--	-......@ .?....--
190	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	-------- --------

File Registers

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	0D	1B	00	00	00	00	--	00	00	00	00	00	00	00	-....... -.......
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
020	00	00	30	30	FA	F2	00	00	00	00	00	00	00	00	00	00	..00.... ....
030	11	06	09	00	00	00	00	00	00	00	00	00	00	00	00	00	....
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
080	--	FF	0D	1B	00	3F	FF	FF	--	08	00	00	00	00	10	60	-....?.. -.....`
090	00	00	FF	00	00	FF	00	00	02	00	00	00	00	01	00	00	....
0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
100	--	00	0D	1B	00	08	00	00	00	02	00	00	00	00	00	00	-....... ....
110	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	-------- --------
120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
180	--	FF	0D	1B	00	00	FF	40	1F	3F	00	00	00	00	--	--	-......@ .?....--
190	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	-------- --------



Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	0D	1B	00	00	00	00	--	00	00	00	00	00	00	00	-....... -.......
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
020	00	00	30	30	FA	F2	00	00	00	00	00	00	00	00	00	00	..00.... ....
030	15	06	09	00	00	00	00	00	00	00	00	00	00	00	00	00	....
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
080	--	FF	0D	1B	00	3F	FF	FF	--	08	00	00	00	00	10	60	-....?.. -......`
090	00	00	FF	00	00	FF	00	00	02	00	00	00	00	01	00	00	....
0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
100	--	00	0D	1B	00	08	00	00	00	02	00	00	00	00	00	00	-....... ....
110	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	-------- --------
120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
180	--	FF	0D	1B	00	00	FF	40	1F	3F	00	00	00	00	--	--	-......@ .?....--
190	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	-------- --------



Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	14	1F	00	00	00	00	--	00	00	00	00	00	00	00	-....... -.......
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
020	00	00	30	30	FA	F2	00	00	00	00	00	00	00	00	00	00	..00.... ....
030	20	06	09	00	00	00	00	00	00	00	00	00	00	00	00	00	....
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
080	--	FF	14	1F	00	3F	FF	FF	--	08	00	00	00	00	10	60	-....?.. -......`
090	00	00	FF	00	00	FF	00	00	02	00	00	00	00	01	00	00	....
0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
100	--	00	14	1F	00	08	00	00	00	02	00	00	00	00	00	00	-....... ....
110	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	-------- --------
120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
150	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
180	--	FF	14	1F	00	00	FF	40	1F	3F	00	00	00	00	--	--	-......@ .?....--
190	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	-------- --------
1A0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	-------- --------
1B0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	-------- --------
1C0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	-------- --------
1D0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	-------- --------
1E0	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	-------- --------
1F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

Análisis:

Comenzamos definiendo “CONT” como la dirección de memoria donde se almacenará el valor que se incrementará en el programa. Inicialmente, se borra el contenido de “CONT” para asegurarnos de que comienza desde cero.

A continuación, se inicia un primer ciclo de incremento. En este ciclo, “CONT” se incrementa en uno hasta alcanzar el valor hexadecimal **H'09'**. Para verificar si se ha alcanzado este valor, se utiliza la instrucción **XORWF CONT, 0**, la cual compara el contenido de “CONT” con **H'09'**. Si el resultado de la comparación es cero, el bit Z en el registro STATUS se activa, lo que indica que “CONT” ha alcanzado el valor deseado. En este caso, el programa sale del ciclo y asigna el valor **H'10'** a “CONT”.

Después de esta primera etapa, se inicia un segundo ciclo de incremento. Este ciclo sigue la misma lógica que el anterior, pero esta vez se incrementa “CONT” hasta alcanzar **H'19'**. Una vez que se cumple esta condición, se asigna el valor **H'20'** a “CONT”.

Finalmente, el programa vuelve a empezar desde el inicio con la instrucción **GOTO BEGIN**, repitiendo indefinidamente el proceso.

Análisis:

Para este ejercicio se establecieron dos ciclos de incremento de la variable “CONT” hasta alcanzar valores específicos. Se utilizó la operación lógica XOR para verificar si “CONT” había alcanzado el valor esperado, aprovechando la activación del bit Z en el registro STATUS como condición de salida. El proceso se repite de manera indefinida, asegurando que el programa continúe ejecutándose sin interrupciones.

Conclusiones:

Martínez Pérez Brian Erik

En esta práctica pude adquirir un conocimiento más profundo sobre el conjunto de instrucciones del microcontrolador, comprendiendo cómo manipular registros, realizar operaciones aritméticas y controlar el flujo del programa mediante saltos y condiciones.

Nuñez Rodas Abraham Enrique

A lo largo de esta práctica, adquirimos un mejor entendimiento sobre la programación en ensamblador para el microcontrolador PIC16F877, aplicando instrucciones fundamentales como **MOV, ADD, GOTO, ORG y EQU**, así como el uso de registros y banderas para manipular datos y controlar el flujo del programa.

Mediante la ejecución de los ejercicios, reforzamos la importancia de las instrucciones de comparación y salto condicional, como **BTFSS**, que nos permitió

evaluar el estado de la bandera Z y realizar operaciones en función del resultado. Además, analizamos el impacto del direccionamiento directo y la forma en que los valores almacenados en la memoria pueden ser modificados y utilizados en operaciones aritméticas y lógicas.

Vicenteño Maldonado Jennifer Michel
Durante la práctica implementamos las instrucciones que vimos antes cómo MOV, ADD, GOTO, ORG, EQU, utilizamos registros, banderas, así como la instrucción BTFSS para revisar el estado de la bandera de acarreo y realizamos modificaciones a los programas vistos durante la primera sesión basándose en diagramas de flujo colocados por el profesor. También utilizamos saltos condicionados con el valor de la bandera C.

Referencias:

del PIC, 2. 1-La Familia. (s/f). 2.- Descripción General del PIC16F877. Edu.ar. Recuperado el 10 de febrero de 2025, de https://exa.unne.edu.ar/ingenieria/sysistemas/public_html/Archi_pdf/HojaDatos/Microcontrolador es/PIC16F877.pdf

(S/f). Newark.com. Recuperado el 10 de febrero de 2025, de https://mexico.newark.com/microchip/pic16f877a-i-p/microcontroller-mcu-8-bitpic16/dp/69K7640?srsltid=AfmBOorWLTceQMTppGk0OMGmmjB6UpIiiw55U28F2qBZH2pUYET 3EInu