

Universidad Nacional Autónoma de México

Facultad de Ingeniería

Laboratorio de Microcomputadoras

**Práctica 01: Introducción a la programación del microcontrolador PIC16F877;
“Direccionamiento Directo”**

Profesor: Rubén Anaya García

Alumnos:

- Murrieta Villegas Alfonso
- Reza Chavarria, Sergio Gabriel
- Valdespino Mendieta Joaquín

Grupo: 4

Semestre: 2021-2

Práctica 01: Introducción a la programación del microcontrolador PIC16F877; “Direccionamiento Directo”

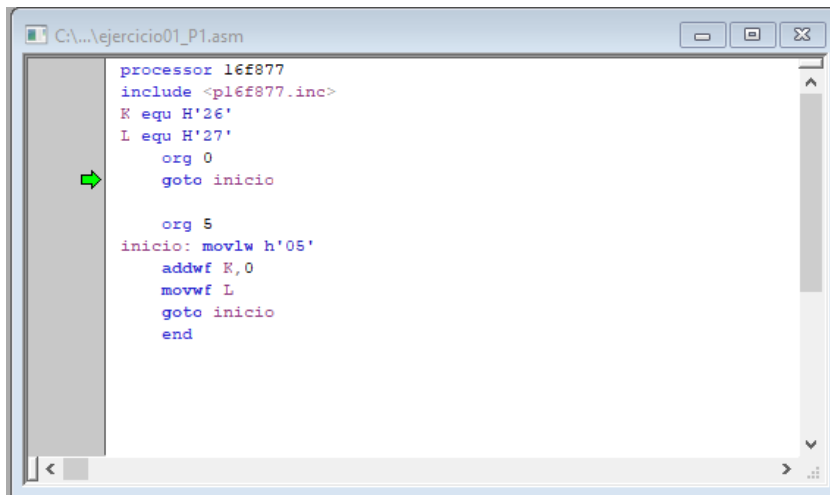
Objetivo

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

Desarrollo

Ejercicio 1

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



```
processor 16f877
include <pl6f877.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
```

Código 1: Ejercicio 1

Recomendaciones:

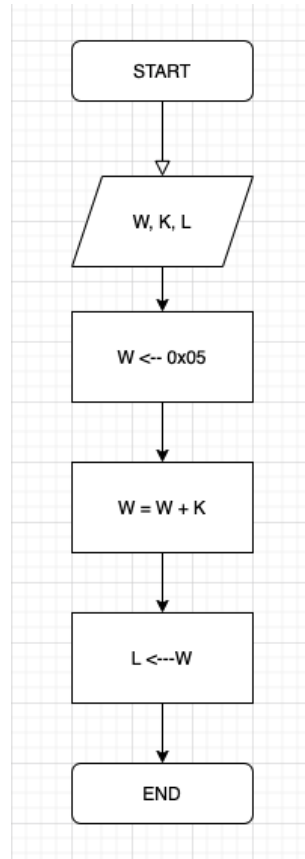
- a) Las etiquetas y la definición de variables en la primera columna.
- b) Las instrucciones deben iniciar a partir de la segunda columna.
- c) Las instrucciones y directivas pueden ser en mayúsculas o minúsculas de manera indistintas, no así las variables.

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

Descripción

El programa consiste en la suma de un número, guardado en la dirección de memoria 0x26 (K), con el valor de 05, guardado en el registro W de PIC16F877. El resultado de la suma se guardará en la dirección 0x27 (L).

Diagrama de flujo del algoritmo



Ejecución

Para el manejo de las memorias se utilizó el Archivo de Registros para modificar los valores de las direcciones que utilizaremos, en este caso el H'26'.

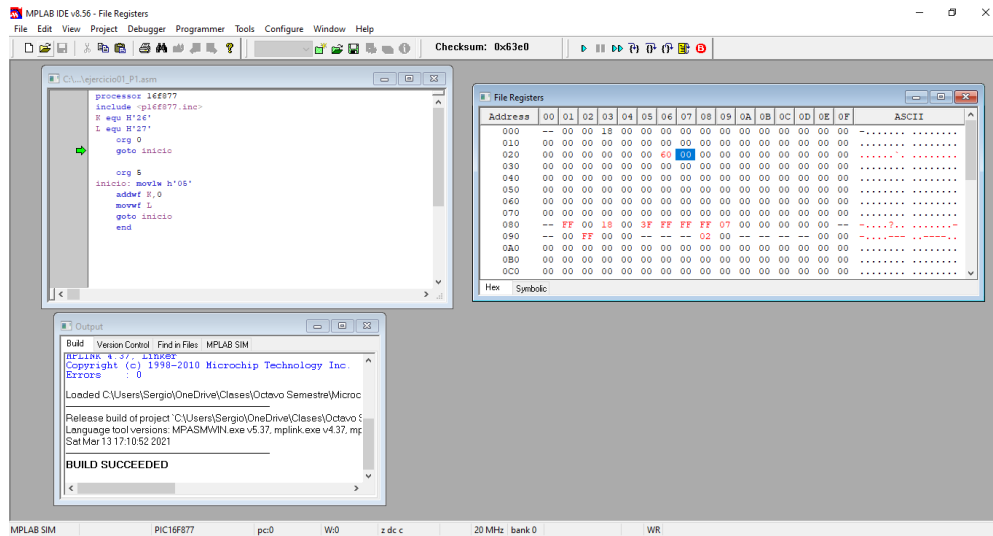
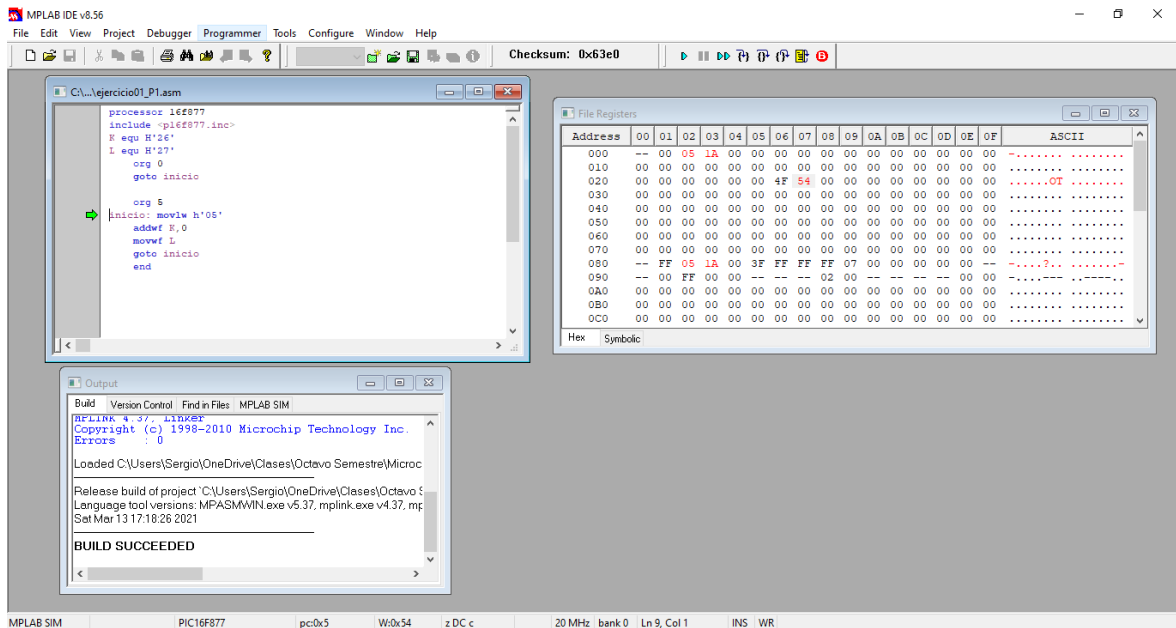


Imagen 1: Programa 1 y File RegisterS

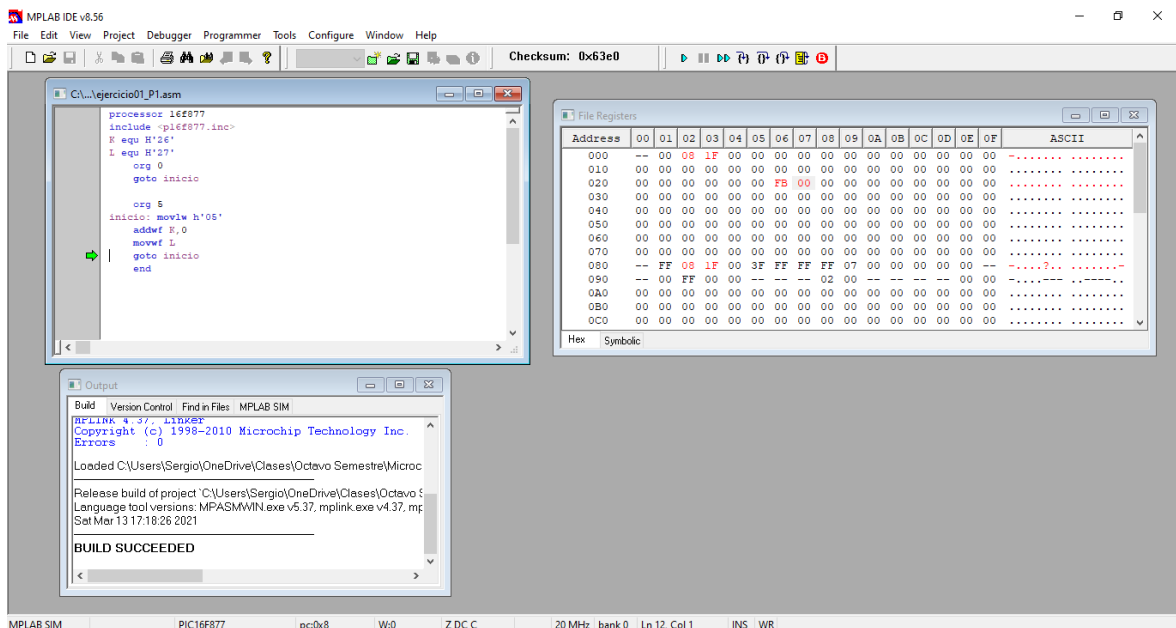
A continuación, se mostrarán algunas pruebas de la ejecución.

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	07	18	00	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	00	00	00	00	60	65	00	00	00	00	00	00	00	00e.....
030	00	00	00	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	07	18	00	3F	FF	FF	FF	07	00	00	00	00	00	--?.....
090	--	00	FF	00	00	--	--	--	02	00	--	--	--	--	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

Pruebas 1: Suma de 60 en dirección H'26 con 5 en registro W



Pruebas 2: Suma de 4F con 5.

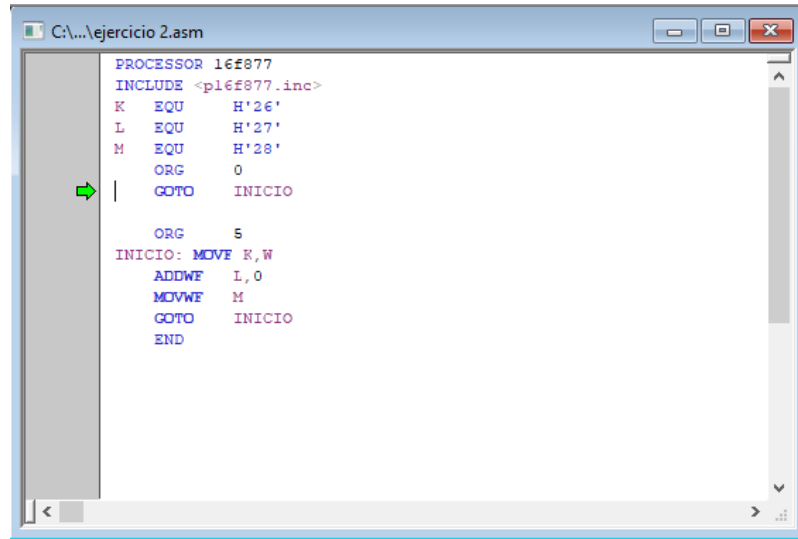


Pruebas 3: Suma de FB con 5

En la última ejecución se maneja el bit de acarreo y el bit cero para especificar el cambio de banderas. Esto hay que tomarlo en cuenta para diferentes algoritmos que veremos en el curso. Estas banderas se encuentran en la parte inferior de MPLab.

Ejercicio 2

Escribir, ensamblar y ejecutar el siguiente programa:



```
PROCESSOR 16f877
INCLUDE <pic16f877.inc>
K EQU H'26'
L EQU H'27'
M EQU H'28'
ORG 0
GOTO INICIO

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

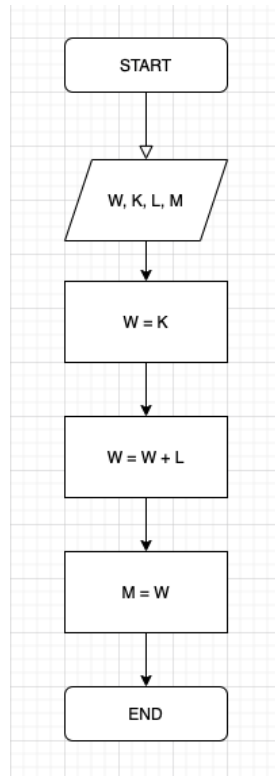
Código 2: Programa 2

- 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

Descripción

El programa consta de la suma de 2 números guardados en 2 localidades de memoria, estos siendo la dirección H'26' y H'27' (K y L). Para operar el dato de K se guarda en el registro W, y W opera con el dato en L. Después de realizar la suma, el resultado guardado en el registro w, se guarda en el registro H'28' (M).

Diagrama de flujo del algoritmo



Ejecución

A continuación, se realizarán algunas ejecuciones.

Ejecución 1: En la primera ejecución, se guardarán os siguientes datos. En K tendrá el valor de 05 y L 06.

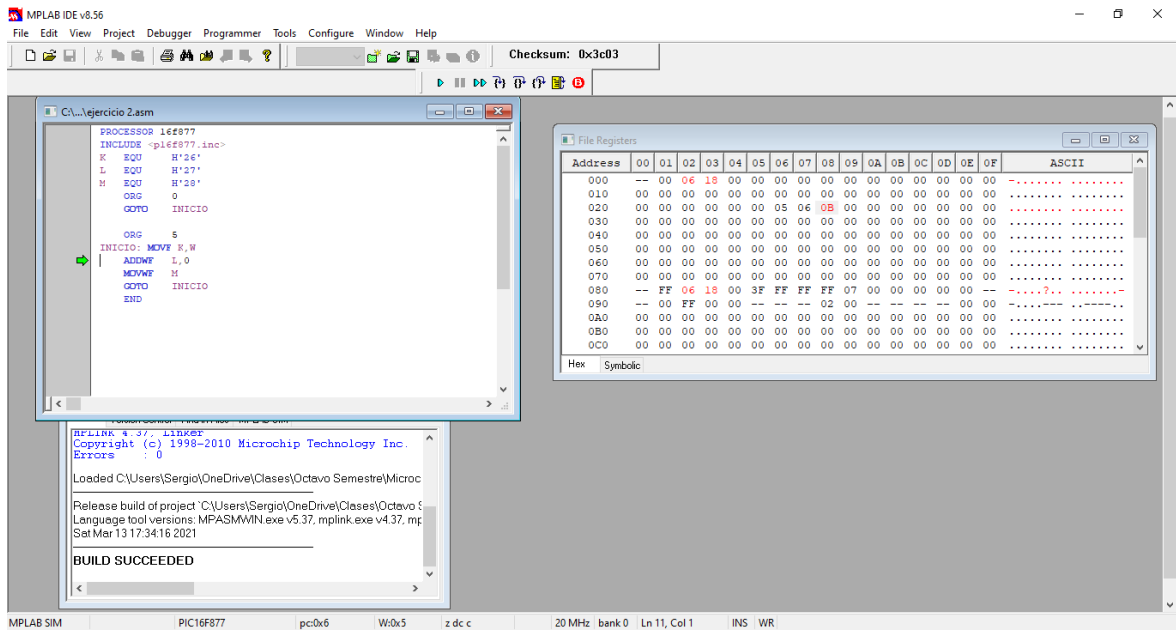


Imagen 2: Programa 2

The screenshot shows the File Registers window with the following data:

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	06	18	00	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	00	00	00	00	05	06	0B	00	00	00	00	00	00	00
030	00	00	00	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	18	00	3F	FF	FF	FF	07	00	00	00	00	--	--?
090	--	00	FF	00	00	--	--	--	02	00	--	--	--	--	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

Imagen 3: Resultado de la ejecución, resultado de 0B en dirección H'27' (M)

En la siguiente operación no se presentan los casos de un resultado de 0, no hay ni acarreo ni acarreo medio. Esto se puede revisar en la parte inferior de MPLab. Al estar en minúsculas están en 0, mayúsculas en 1.

Ejecución 2: Se presentan los valores de 2A en K y 06 en L.

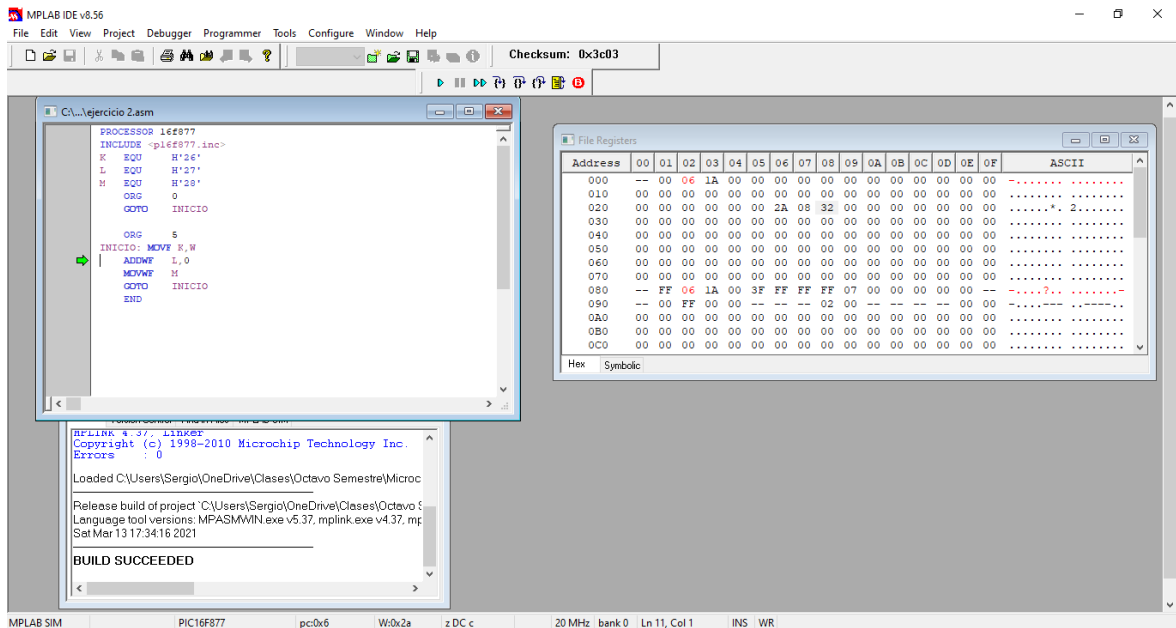


Imagen 4: Ejecución 2 $K=20B_6$ y $L=06$.

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	05	1A	00	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	00	00	00	00	2A	08	32	00	00	00	00	00	00	00*. 2.....
030	00	00	00	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	1A	00	3F	FF	FF	FF	07	00	00	00	00	--	--	-----?.....
090	--	00	FF	00	--	--	--	--	02	00	--	--	--	--	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

Imagen 5: Resultado de operación en M, valor 32.

La única bandera arriba es la de medio acarreo, esta representa un bit que sobrepasa al primer bit del número. La suma de A y 8 genera el bit medio de acarreo y este pasa a la suma de la segunda posición del número.

Ejecución 3: Los valores son K igual a D5 y L igual a B9.

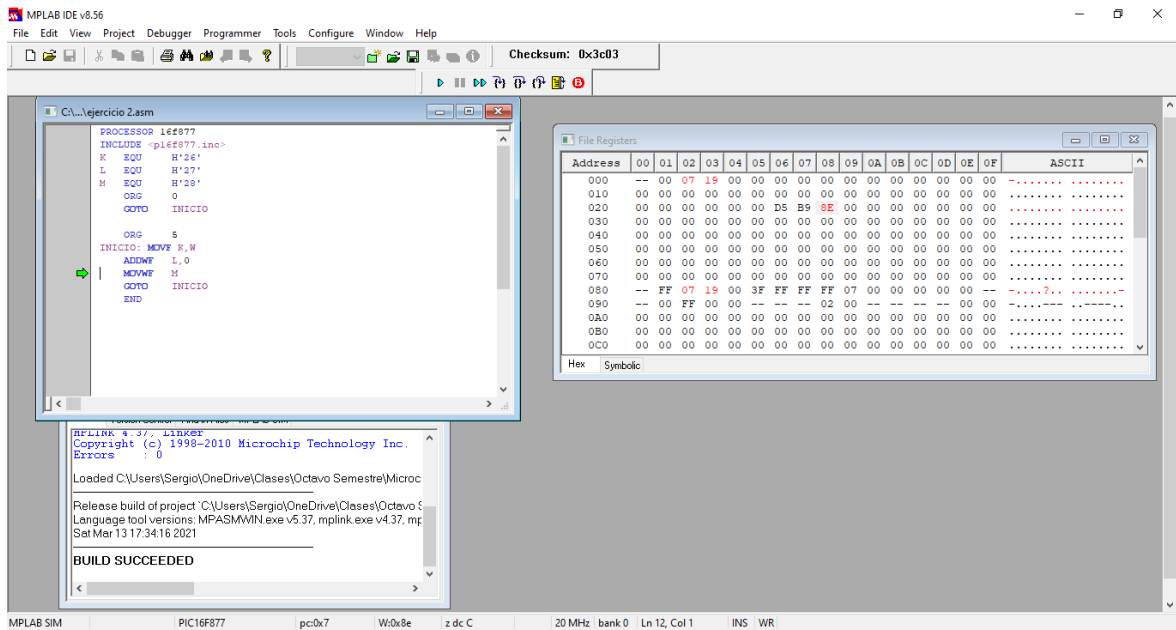


Imagen 6: Ejecución K=D5 Y L=B9

File Registers																	
Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	07	19	00	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	00	00	00	00	D5	B9	8E	00	00	00	00	00	00	00
030	00	00	00	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	07	19	00	3F	FF	FF	FF	07	00	00	00	00	00	--?
090	--	00	FF	00	--	--	--	--	02	00	--	--	--	--	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

Imagen 7: Resultado de 8E en dirección H'28' (M)

La operación presenta levantada la bandera de acarreo, esto sucede por sobrepasar el valor en el segundo bit del número ($D+B=18$). Las otras banderas se presentan abajo.

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.

Descripción

El siguiente programa consiste en utilizar el registro de estatus. En el File Registers se puede consultar en qué dirección de memoria se encuentra, la dirección de STATUS está en H'03'. El programa consiste en el uso de estas 4 direcciones y del registro de estatus.

El valor de J se moverá a W y este se sumará con K, el resultado se guardará en W. El resultado de la operación se guardará en el registro R1. El valor de C1 se resetea antes de revisar la bandera C. El valor de la bandera de acarreo se revisará, si en la dirección H'03', el bit en la posición 0, está arriba se guarda el valor 01 en C1, si no es así termina el programa.

Direcciones utilizadas:

- J: H'26'
- K: H'27'
- C1: H'28'
- R1: H'29'
- STATUS: H'03'

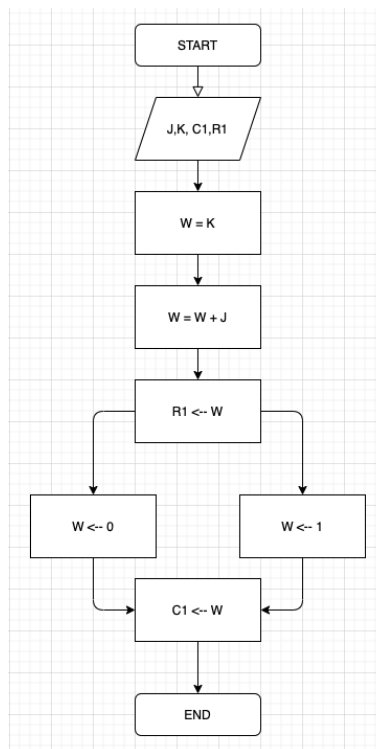
```

C:\...ejercicio 3.asm

PROCESSOR 16f877
INCLUDE <p16f877.inc>
J equ H'26'
K equ H'27'
C1 equ H'28'
R1 equ H'29'
ORG 0
GOTO INICIO
ORG 5
INICIO: MOVF J,W ;Dato en J mover a W.
        ADDWF K,0 ;Sumar W con K
        MOVWF R1 ;Mover de E a R1
        CLRF C1 ;Limpiar C1
        BTFS H'03',0 ;Banderas en dirección H'03'
        ;Si está levantado el bit en posición 0.
        GOTO FIN ;Si no se cumple ir a fin
        MOVLW H'01' ; si se cumple mover el valor 01 a W
        MOVWF C1 ;Mover W a C1
FIN: GOTO INICIO
END

```

Diagrama de flujo del algoritmo



Código 3: Programa 3, suma con bit de acarreo.

Pruebas

Prueba 1: Los valores de J y K serán A5 y A2 respectivamente. La suma generará un acarreo presentará en C1 el valor de 01 y en R1 el valor de 47.

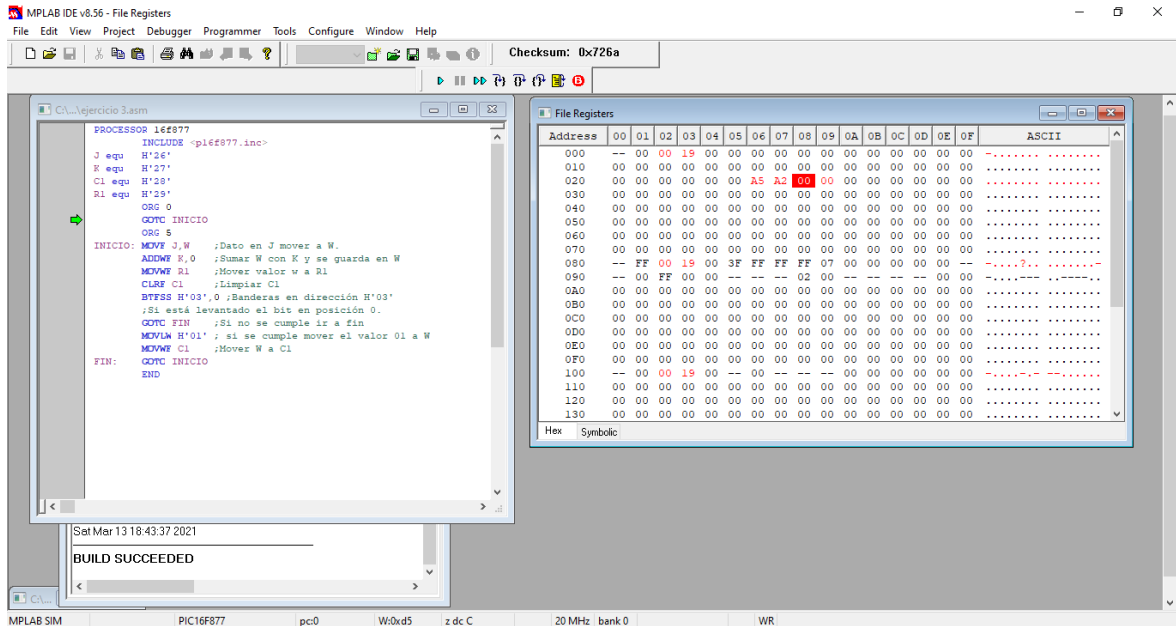


Imagen 8: Estado Inicial del programa

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	00	19	00	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	00	00	00	00	A5	A2	00	00	00	00	00	00	00	00
030	00	00	00	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	00	19	00	3F	FF	FF	FF	07	00	00	00	00	00	--?
090	--	00	FF	00	00	--	--	--	02	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	00	19	00	--	00	--	--	--	00	00	00	00	00	00
110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
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

Imagen 9: Estado inicial de la Ejecución

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	0D	1D	00	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	00	00	00	00	A5	A2	01	47	00	00	00	00	00	00G.....
030	00	00	00	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	1D	00	3F	FF	FF	FF	07	00	00	00	00	00	--?.....
090	--	00	FF	00	00	--	--	--	02	00	--	--	--	--	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	1D	00	--	--	--	--	--	00	00	00	00	00	00
110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
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

Imagen 10: Estado final de la Ejecución

Prueba 2: Suma de K=3B y J=83. El resultado de la suma 3B+83=BF no generará bit de acarreo, por lo que C1 tiene el valor de 00.

MPLAB IDE v8.56

File Edit View Project Debugger Programmer Tools Configure Window Help

Checksum: 0x726a

C:\...ejercicio 3.asm

```

PROCESSOR 16F877
INCLUDE <pic877.inc>
J equ H'26'
K equ H'27'
C1 equ H'28'
R1 equ H'29'

ORG 0
GOTO INICIO
ORG 5
INICIO: MOVF J,W ;Dato en J mover a W.
        ADDWF K,0 ;Sumar W con K y se guarda en W
        MOVWF R1 ;Mover valor w a R1
        CLRF C1 ;Limpiar C1
        BTFSF H'03',0 ;Bandera en dirección H'03'
        ;Si está levantado el bit en posición 0.
        GOTO FIN ;Si no se cumple ir a fin
        MOVWF H'01' ; si se cumple mover el valor 01 a W
        MOVWF C1 ;Mover W a C1
FIN: GOTO INICIO
END
  
```

File Registers

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	0D	1D	00	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	00	00	00	00	3B	84	00	00	00	00	00	00	00	00
030	00	00	00	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	1D	00	3F	FF	FF	FF	07	00	00	00	00	--?.....	
090	--	00	FF	00	00	--	--	--	02	00	--	--	--	--	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	1D	00	--	--	--	--	--	00	00	00	00	00	00
110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
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

Hex Symbolic

Set Mar 13 18:51:13 2021

BUILD SUCCEEDED

MPLAB SIM PIC16F877 pc:0 W:0x1 Z dc C 20 MHz bank 0 Ln 8, Col 1 INS WR

Imagen 11: Estado Inicial de la ejecución

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	00	1D	00	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	00	00	00	00	3B	84	00	00	00	00	00	00	00	00?
030	00	00	00	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	00	1D	00	3F	FF	FF	FF	07	00	00	00	00	00	--?
090	--	00	FF	00	00	--	--	--	02	00	--	--	--	--	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	00	1D	00	--	00	--	--	--	00	00	00	00	00	00
110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
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

Imagen 12: Esto inicial de File Register

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	05	1C	00	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	00	00	00	00	3B	84	00	BF	00	00	00	00	00	00?
030	00	00	00	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	05	1C	00	3F	FF	FF	FF	07	00	00	00	00	00	--?
090	--	00	FF	00	00	--	--	--	02	00	--	--	--	--	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	05	1C	00	--	00	--	--	--	00	00	00	00	00	00
110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
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

Imagen 13: Estado final del File Register

Ejercicio 4

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

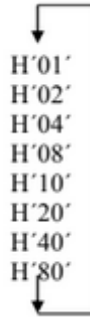


Ilustración 1: Secuencia del ejercicio 4 a realizar

Descripción

La secuencia proporcionada, si es en número binario, se puede apreciar que se trata de un corrimiento de bits. En este caso al tener solo un bit se generan los valores de la secuencia. El desplazamiento es a la izquierda, así para aumentar el valor del registro.

El programa consiste en asignar en 2 secciones, la sección de inicio asigna el valor de H'01' en W y lo mueve a la dirección de variable Rota (H'20'). La segunda sección consiste en realizar el recorrido a la izquierda del número. Revisa si el último bit está encendido, si es así dirigirse a Sección de Inicio, si no repite el proceso.

```
PROCESSOR 16F877
#include <P16F877A.INC>

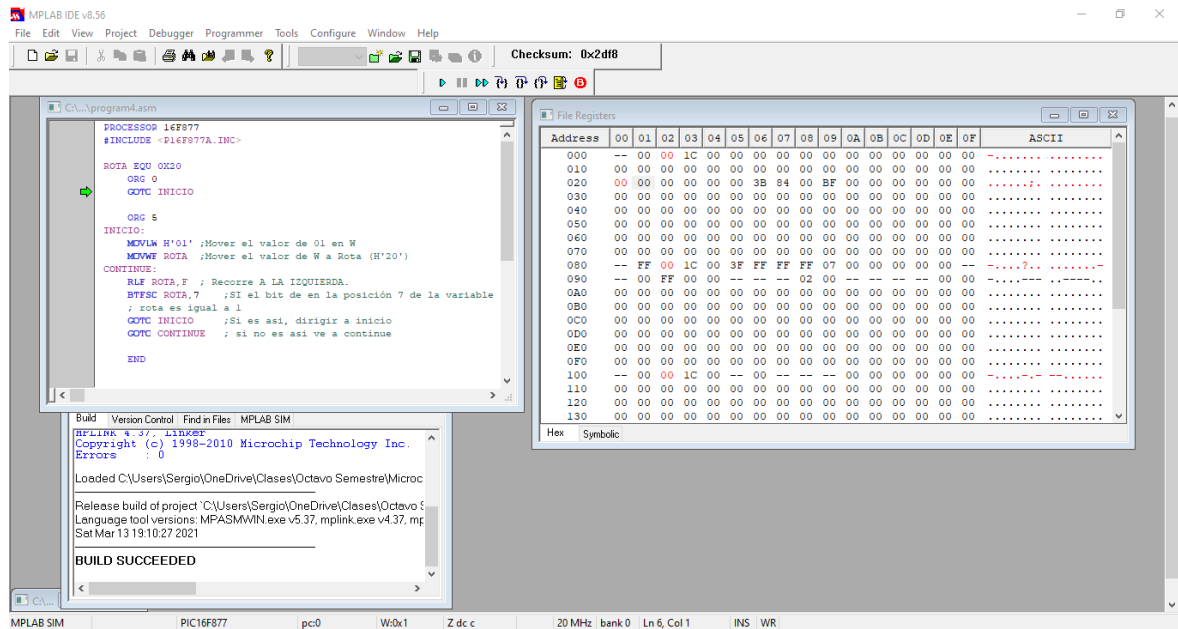
ROTA EQU 0X20
ORG 0
GOTO INICIO

ORG 5
INICIO:
    MOVLW H'01'; Mover el valor de 01 en W
    MOVWF ROTA ; Mover el valor de W a Rota (H'20')
CONTINUE:
    RLF ROTA,F ; Recorre A LA IZQUIERDA.
    BTFSC ROTA,7 ; SI el bit de en la posición 7 de la variable
    ; rota es igual a 1
    GOTO INICIO ; Si es así, dirigir a inicio
    GOTO CONTINUE ; si no es así ve a continue

END
```

Código 4: Generador de secuencia del ejercicio 4

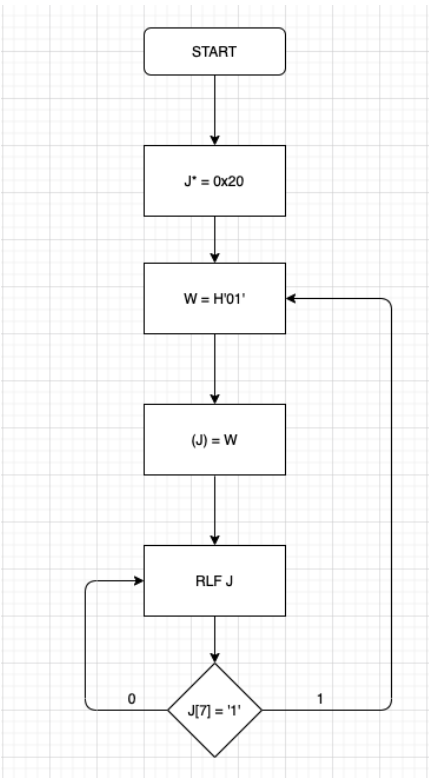
Prueba



Código 5: Estado inicial de la ejecución.

A continuación, se anexarán los 8 estados de la secuencia en la siguiente tabla. La dirección en donde se encuentra la secuencia es en H'20'

Diagrama de flujo del algoritmo



Valor de Ejecución
secuencia

H'01'

File Registers																	ASCII	
Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F		
000	--	00	07	1C	00	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	01	00	00	00	00	00	3B	84	00	BF	00	00	00	00	00	00
030	00	00	00	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	07	1C	00	3F	FF	FF	07	00	00	00	00	00	--	--?
090	--	00	FF	00	00	--	02	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	07	1C	00	--	00	--	--	--	00	00	00	00	00	00
110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
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

Hex Symbolic

H'02'

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	0A	1C	00	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	02	00	00	00	00	00	00	3B	84	00	BF	00	00	00	00	00
030	00	00	00	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	0A	1C	00	3F	FF	FF	FF	07	00	00	00	00	--	--?
090	--	00	FF	00	00	00	--	--	02	--	--	--	--	--	--	--
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	0A	1C	00	--	--	--	--	00	00	00	00	00	00	00
110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
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

Hex Symbolic

H'04'

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Comment
000	--	00	07	1C	00	00	00	00	00	00	00	00	00	00	00	00	ASCII
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	...
020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	...
030	00	00	00	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	07	1C	00	3F	FF	FF	FF	07	00	00	00	00	--	--	...
090	--	00	FF	00	00	--	--	--	02	00	--	--	--	--	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	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	...
110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	...
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	...

Hex Symbolic

The screenshot shows the 'File Registers' window in the Proteus IDE. The window contains a table with 16-bit registers (Addresses 000 to 130) and their corresponding values. The 'ASCII' column shows the character representation of the register values. The 'Hex' and 'Symbolic' columns are visible at the bottom.

Address	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	0A	1C	00	00	00	00	00	00	00	00	00	00	00	
001	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
020	08	00	00	00	00	3B	84	00	BF	00	00	00	00	00	00
030	00	00	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
050	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
070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
080	--	FF	0A	1C	00	3F	FF	FF	07	00	00	00	00	--	--?
090	--	FF	00	00	--	--	02	00	--	--	--	--	--	00	00?
0A0	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
0C0	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
0E0	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
100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
120	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

Hex Symbolic

H'10'

[illegible]

H'20'

H'40'

File Registers

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	0A	1C	00	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	40	00	00	00	00	00	3B	84	00	BF	00	00	00	00	00	00	8.....
030	00	00	00	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	0A	1C	00	3F	FF	FF	FF	07	00	00	00	00	00	--	-----
090	--	00	FF	00	--	--	--	02	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	0A	1C	00	--	--	--	--	--	00	00	00	00	00	00	-----
110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	-----
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	-----

Hex Symbolic

H'80'

The screenshot displays the 'File Registers' window from a debugger. It features a table with columns for 'Address', 'Hex', and 'Symbolic'. The 'Address' column ranges from 000 to 130. The 'Hex' column contains hexadecimal values, some of which are highlighted in red (e.g., 08, 80, 3B, BF, FF). The 'Symbolic' column shows the corresponding ASCII characters, such as dots, question marks, and dashes. A vertical scrollbar on the right indicates that the data extends beyond what is currently visible.

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	08	1C	00	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	80	00	00	00	00	00	00	3B	84	00	BF	00	00	00	00	00?
030	00	00	00	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	08	1C	00	3F	FF	FF	FF	07	00	00	00	00	00	--	-...?..-
090	--	00	FF	00	00	--	--	--	02	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	08	1C	00	--	--	--	--	00	00	00	00	00	00	00	-...-..
110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
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

At the bottom left, there are two tabs: 'Hex' and 'Symbolic', both of which appear to be selected or active.

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.

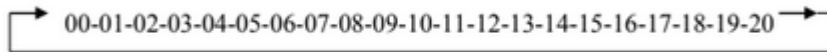


Ilustración 2: Secuencia del ejercicio 5

Descripción

El programa se dividirá en 2 secciones, los valores con el segundo bit en 0 y en 1. Se realizará el incremento del valor del registro hasta los valores de 09 y 19. Se realizará una resta para comparar el valor actual del registro con los valores mencionados, si se da un resultado de 0 en la resta se realizará la asignación del valor, en hexadecimal, del siguiente número en “formato decimal” (10 y 20 respectivamente). Si no es así, en ambos casos se seguirá incrementando el valor del registro.

El bloque de 0-10 al terminar continuará con el bloque de 11-20. Al finalizar el segundo bloque se reiniciará el ciclo.

```

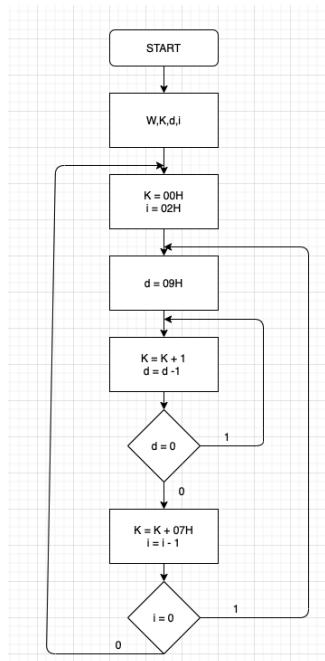
C:\...\program5.asm
PROCESSOR 16F877
#include <P16F877A.INC>

CONTA EQU 0X20
ORG 0
GOTO INICIO
ORG 5
INICIO:
    CLRF CONTA    ;Limpia CONTA
CONTINUAL:
    INCF CONTA,F  ;Incrementa en 1 CONTA
    MOVF CONTA,W  ;Mueve Cona a W.
    SUBLW 0X09    ;Resta 09 a W.
    BTFSS STATUS,Z ;Si el STATUS Z==1
    GOTO CONTINUAL ;NO, ir a continua.
    MOVLW 0X10    ;SI, Mover el valor de 10 a W
    MOVWF CONTA   ;Mover el valor de W a CONTA
CONTINUA2:
    INCF CONTA,F  ;Incrementa en 1 CONTA
    MOVF CONTA,W  ;Mueve Cona a W.
    SUBLW 0X19    ;Resta 19 a W
    BTFSS STATUS,Z ;Si STATUS Z==1
    GOTO CONTINUA2 ;NO, Ir a Continua2
    MOVLW 0X20    ;SI, Mover el valor de 10 a W
    MOVWF CONTA   ;Mover el valor de W a Cona
    GOTO INICIO   ;Reinicio de conteo.

END
  
```

Código 6: Programa de secuencia de 00-20 en representación decimal con formato Hexadecimal

Diagrama de flujo del algoritmo



Pruebas

A continuación, se anexarán las capturas del recorrido que realiza el incremento de la variable, en este caso de H'20'.

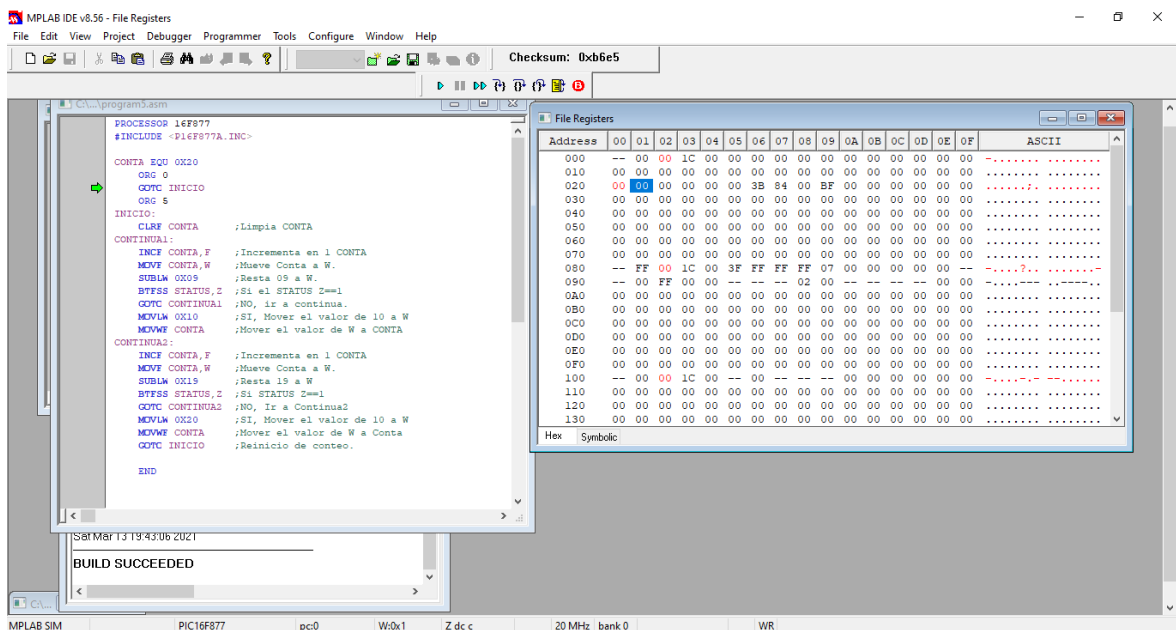


Imagen 14: Estado inicial del programa

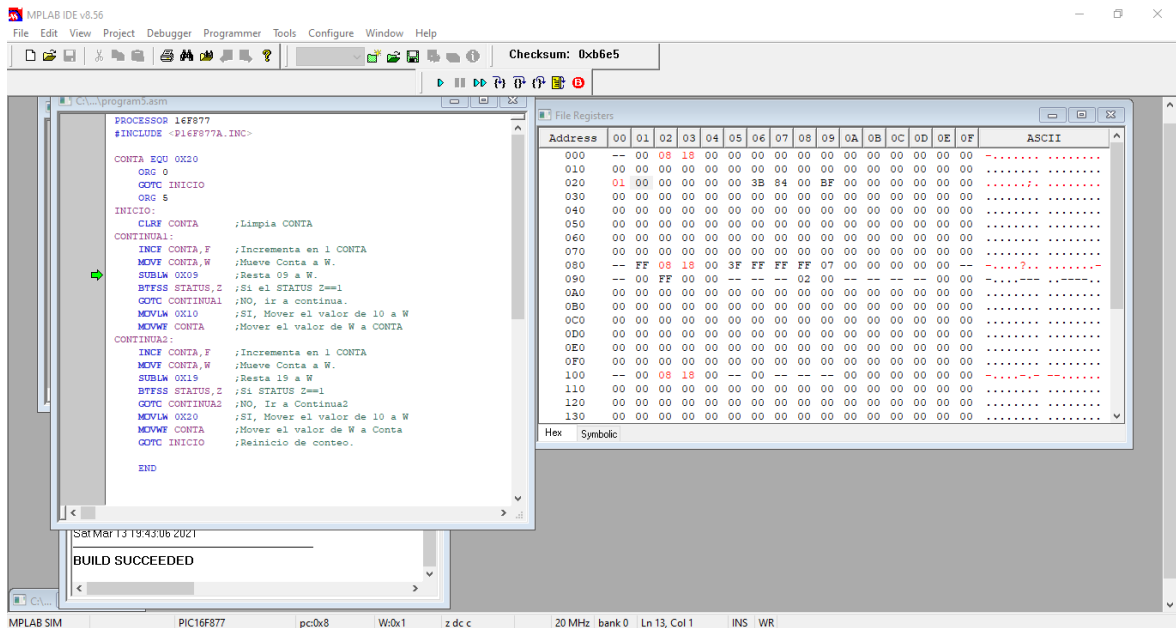


Imagen 15: Ejecución del primer bloque de la secuencia.

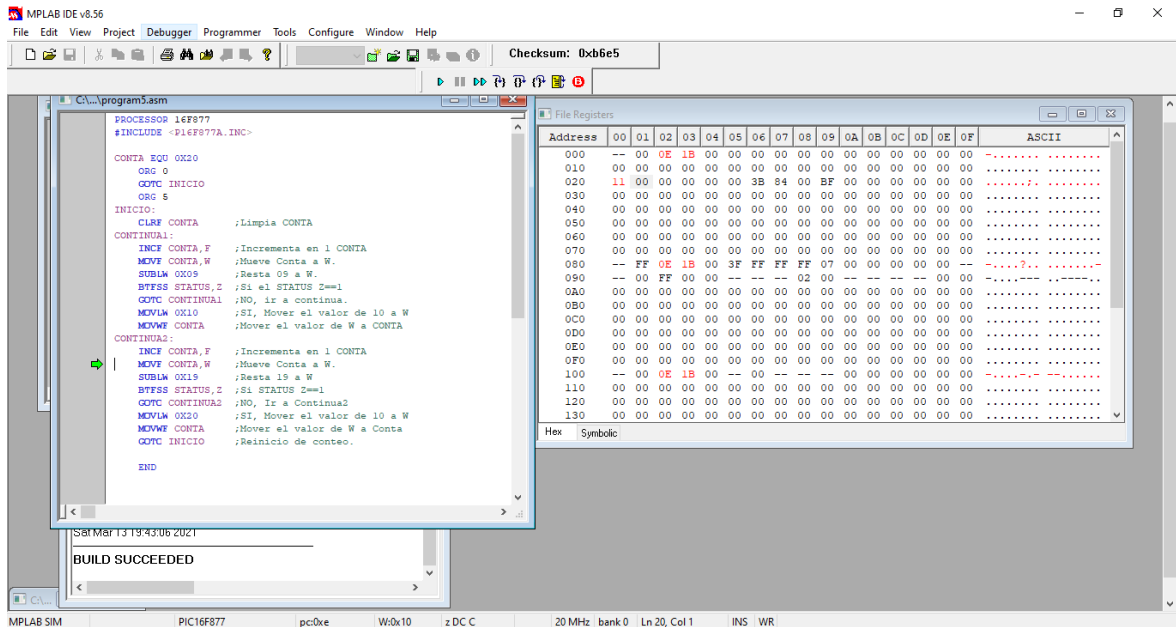


Imagen 16: Ejecución del segundo bloque de la secuencia

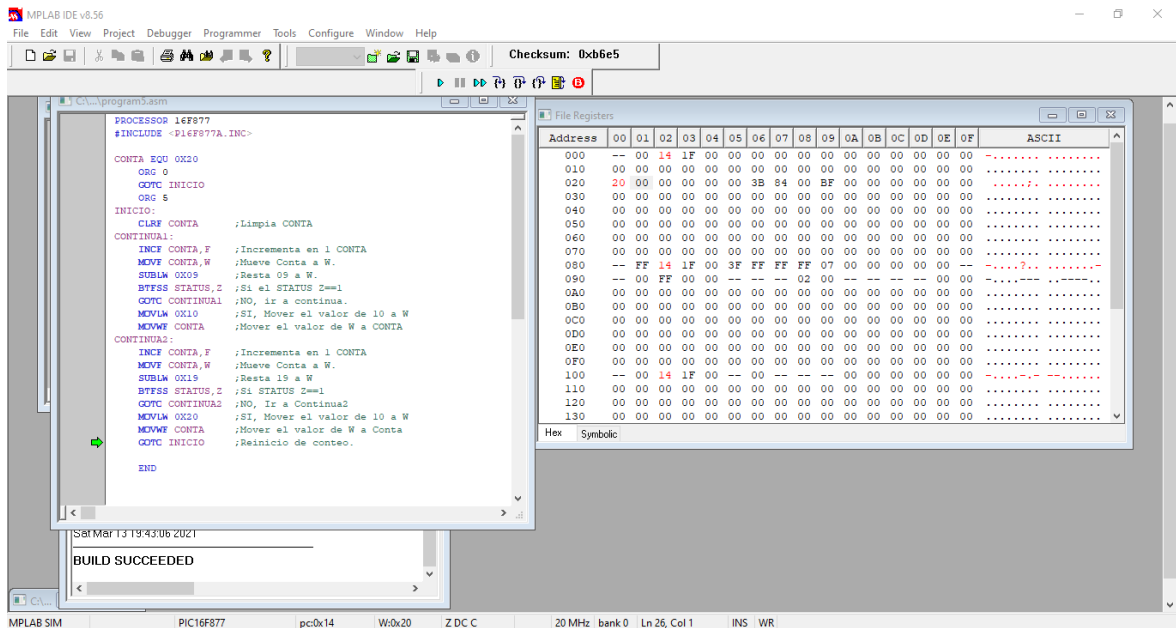


Imagen 17: Llegada al último valor de la secuencia

Conclusiones

Murrieta Villegas Alfonso

En la presente práctica repasamos conceptos de materias previas como son Diseño Digital VLSI y sobre todo estructura y programación de computadoras, desde aspectos tan básicos como conocer los nemónicos o instrucciones en “assembly” de un microprocesador hasta aspectos más complejos de entender como el cómo es que se manejan los registros dentro de nuestro pic además del flujo de las instrucciones de control mediante banderas e incluso las diferencias que sufren con el tipo de direccionamiento o modo en la que se trabajará.

Por último, pero no menos importante, aprendimos a utilizar nuestra herramienta de desarrollo y simulación MPLAB que sin duda nos proporcionará de manera visual tanto en los registros como en la lectura de cada instrucción la forma en que trabaja nuestro microprocesador además de poder darnos herramientas para depurar e incluso analizar a detenimiento nuestros códigos.

Reza Chavarria Sergio Gabriel

A partir de la realización de los ejercicios se pudo revisar los conceptos básicos de las instrucciones que puede realizar el un microprocesador, como lo fueron en las operaciones orientadas a registros, manejo de bits, instrucciones de control, los registros y tipos de banderas que se manejan. El ambiente del simulador de MPLAB nos proporciona la visualización del comportamiento que tiene la memoria a la hora de su ejecución, esto nos permite en la comprensión, de manera visual, de las acciones que realiza el programa en la memoria.

Valdespino Mendieta Joaquin

En la presente practica se pudieron relacionar los conceptos y retomar los conocimientos de la materia relacionada a estructura y programacion de computadoras, con el desarrollo de estos ejercicios pudimos observar y comprender el funcionamiento de las instrucciones, registros y el uso de banderas, ademas del flujo del programa en este tipo de microprocesador, por otra parte pudimos conocer y manejar los inicios de MPLAB como IDE de desarrollo para esta materia.