

Universidad Nacional Autónoma de México

Facultad de Ingeniería

Laboratorio de Microcomputadoras

Práctica 02: Programación en ensamblador direccionamiento indirecto

Profesor: Rubén Anaya García

Alumnos:

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

Grupo: 4

Semestre: 2021-2

Práctica 02: Programación en ensamblador direccionamiento indirecto

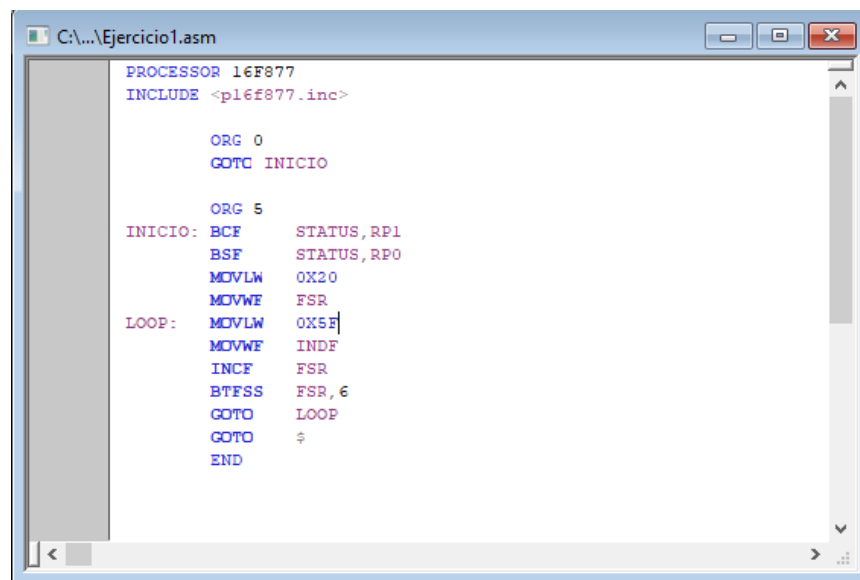
Objetivo

Analizar la programación en lenguaje ensamblador. Realizar algoritmos en lenguaje ensamblador empleando direccionamiento indirecto.

Desarrollo

Actividad 1

Escribir, comentar y ejecutar la simulación del siguiente programa:



```
PROCESSOR 16F877
INCLUDE <pl6f877.inc>

ORG 0
GOTO INICIO

ORG 5
INICIO: BCF     STATUS,RP1
        BSF     STATUS,RP0
        MOVLW   0X20
        MOVWF   FSR
LOOP:   MOVLW   0X5F
        MOVWF   INDF
        INCF    FSR
        BTFSS   FSR,6
        GOTO    LOOP
        GOTO    $
        END
```

Código 1: Ejercicio 1

Descripción

A partir del direccionamiento indirecto se declara el apuntador estar en la dirección 0x20. El valor que se moverá a este registro será el valor de 0x5F. Al finalizar el guardado del valor 5F, el valor del FSR incrementará en 1. El proceso de asignar 5F se realizará de manera repetitiva hasta que el valor de FSR llegue al valor de 0x40. Esto se revisará con revisar el bit en la posición 6 del valor de FSR.

Ejecución

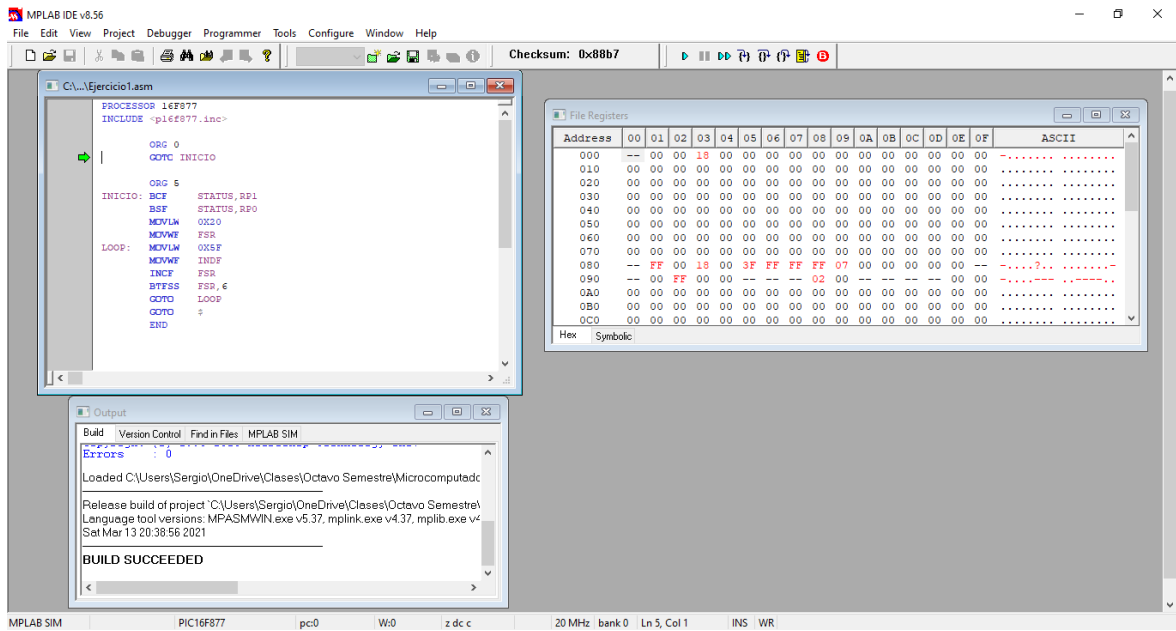


Imagen 1: Estado inicial de ejecución del programa

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	0D	38	21	00	00	00	00	00	00	00	00	00	00	00	-.8!..
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
020	5F	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	0D	38	21	3F	FF	FF	FF	07	00	00	00	00	00	--	-.8!?.
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 2: Inicio de Asignación de 5F en 0x20

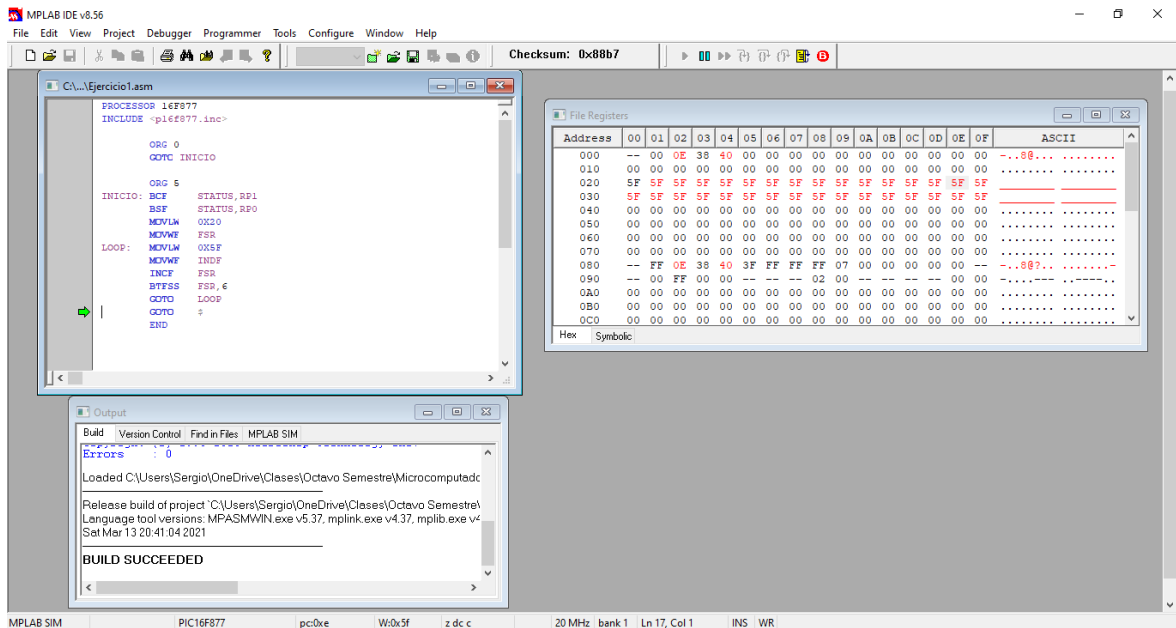


Imagen 3: Finalización y asignación de F5 de las direcciones 0X20 a 0x3F

Actividad 2

Elaborar un programa que encuentre el número menor, de un conjunto de datos ubicados entre las localidades de memoria 0x20 a 0X3F; mostrar el valor en la dirección 40H.

Ejemplo:

Dirección	Dato
20	FF
21	FE
22	FD
23	FC
24	FB
25	FA
26	89
27	88
28	87
29	86
2A	85
2B	84
2C	83
2D	82
2E	81
2F	80

Dirección	Dato
30	6B
31	69
32	68
33	67
34	40
35	41
36	42
37	43
38	44
39	45
3A	46
3B	47
3C	48
3D	49
3E	90
3F	01

Dirección	Dato
40	01

Imagen 4: Ejemplo de ejercicio 2

Descripción.

Para la comparación entre si un valor es menor que otro se realizará una resta. Cuando se realiza una resta se aplica el complemento a 1 y complemento a 2 para generar la

operación. Si el número al que se le aplica los complementos generarán un bit de acarreo significa que ese número es el menor, si no es así es el mayor.

Ejemplo:

- 9F-FE

$$9F = 1001 \ 1111; FE = 1111 \ 1110$$

Complemento a 1 de FE = 0000 0001; Complemento a 2 de FE = 0000 0010

$$9F - FE = 1001 \ 1111 + 0000 \ 0010 = 1010 \ 0001$$

- FE-9F

$$9F = 1001 \ 1111; FE = 1111 \ 1110$$

Complemento a 1 de 9F = 0110 0000; Complemento a 2 de 9F = 0110 0001

$$FE - 9F = 1111 \ 1110 + 0110 \ 0001 = \mathbf{1} \ 0101 \ 1111$$

Ya con esto contemplado se describirá el procedimiento para obtener el número menor.

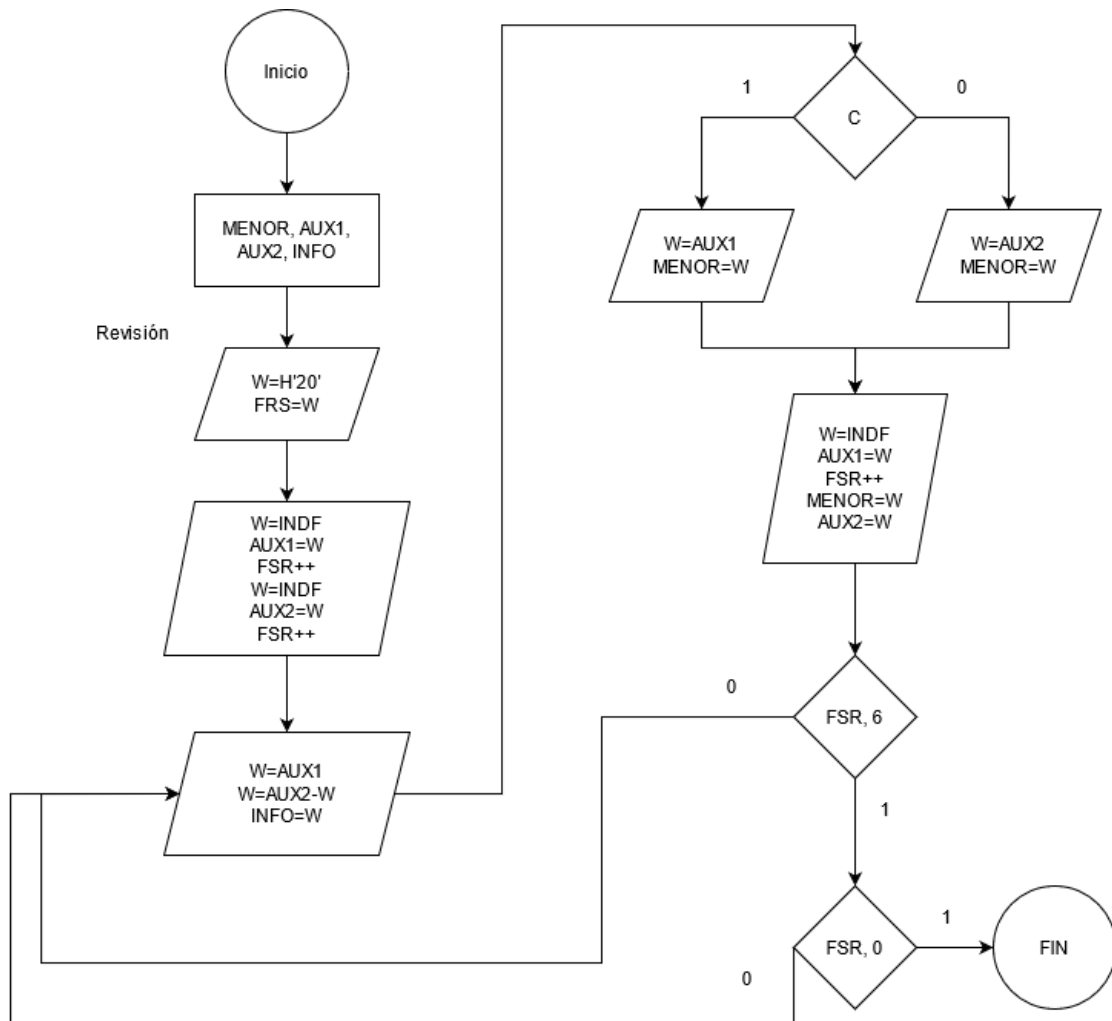
Se declararon 4 variables, MENOR que obtendrá el valor más pequeño durante el procedimiento, variables auxiliares para realizar las restas y la información del resultado de la resta.

Primero se cargarán los primeros 2 números del bloque de direcciones a las variables auxiliares. Esto se realizará a partir del incremento que tendrá FSR para cambiar de direcciones. Después de obtener los valores auxiliares, el aux1 se cargará al registro W. Una vez cargado el valor se realizará la operación AUX2-W. En el caso que W sea menor, este generará un bit de acarreo, por el contrario, si no genera aux2 será menor.

Dependiendo el resultado, el valor de la variable auxiliar será guardado en la dirección de la variable MENOR.

Una vez obtenido se cargará el siguiente valor apuntado por FSR a aux1 y el valor guardado en MENOR será cargado a aux2. Después de la carga se revisará si FSR llega a tener el valor de 0x40, esto revisando el bit 6, si no es así repetirá el procedimiento de la resta, si es así terminará el programa.

Diagrama de Flujo



Código y Ejecución

```
PROCESSOR 16f077
INCLUDE <pl6f077.inc>
MENOR EQU H'40'
AUX1 EQU H'41'
AUX2 EQU H'42'
INFO EQU H'43'
ORG 0
GOTO INICIO
ORG 5
INICIO: MOVLW H'20'
        MOVWF FSR
REVISION:
        MOVF INDF,W; Carga el valor de FSR en W
        MOVWF AUX1; Carga el valor de w en aux1
        INCF FSR; incrementa el apuntador FSR
        MOVF INDF,W; Carga el valor de FSR en W
        MOVWF AUX2; Carga el valor de w en aux2
        INCF FSR; incrementa el valor de FSR
RESTA:
        MOVF AUX1,W; Mueve el dato de aux1 a W
        SUBWF AUX2,W; Del valor de W le restas AUX2, lo pasa a W
        MOVWF INFO; Información de la resta
        BTFSS STATUS,0; Si c==1 aux1 es menor, si c==0 aux2 es menor
        GOTO CASOAUX2; Si es menor aux2 ir a CASOAUX2
        GOTO CASOAUX1; Si es menor aux1 ir a CASOAUX1
CASOAUX1:
        MOVF AUX1,W; Carga el valor de aux1 a W
        MOVWF MENOR; Carga el valor de W a Menor
        GOTO REVISION2; Dirigir a REVISION2
CASOAUX2:
        MOVF AUX2,W; Carga el valor de aux2 a W
        MOVWF MENOR; Carga el valor de W a Menor
        GOTO REVISION2; Dirigir a REVISION2

REVISION2:
        MOVF INDF,W; Carga el valor de FSR en W
        MOVWF AUX1; Carga el valor W1 en aux1
        INCF FSR; incrementa el FSR
        MOVF MENOR,W; Carga el valor de Menor en E
        MOVWF AUX2; carga el valor de W en aux2
        BTFSS FSR,6; Si llega a 0x40zz
        GOTO RESTA; si FSR,6==0
        GOTO FIN
FIN:
        BTFSC FSR,0; Si llega a 0x41
        GOTO $
        GOTO RESTA; si FSR,0==1

END
```

Código 2: Obtención de número menor entre direcciones 0x20 a 0x3F.

Se almacenaron números aleatorios en el rango de 0x20 a 0x39. El resultado del número menor se guardará en 0x40.

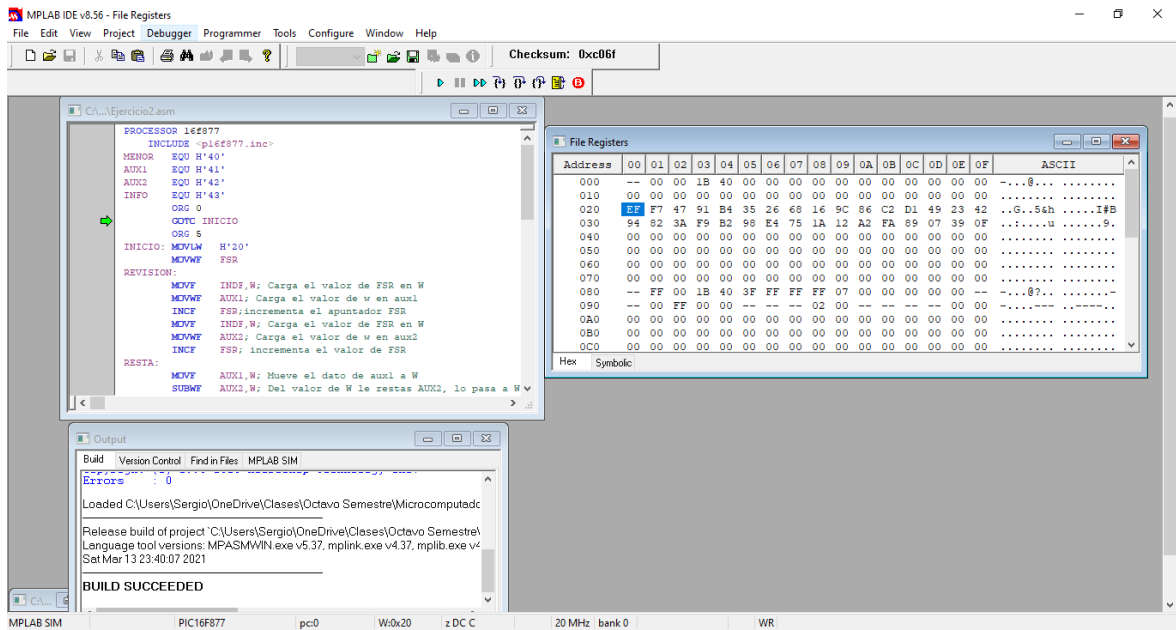


Imagen 5: Estado inicial de la ejecución

File Registers																	
Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	00	00	1B	40	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	EF	F7	47	91	B4	35	26	68	16	9C	86	C2	D1	49	23	42	..G..5&h....I#B
030	94	82	3A	F9	B2	98	E4	75	1A	12	A2	FA	89	07	39	0Fu.....9.
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	1B	40	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

Hex Symbolic

Imagen 6: Estado inicial de File Registers

File Registers																	ASCII
Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
000	--	00	20	18	40	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	EF	F7	47	91	B4	35	26	68	16	9C	86	C2	D1	49	23	42	..G..5&hI#B
030	94	82	3A	F9	B2	98	E4	75	1A	12	A2	FA	89	07	39	0Fu9.
040	07	0F	07	CE	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	20	18	40	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

Imagen 7: Estado final de la ejecución, elemento menor =07

File Registers																	ASCII
Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
000	--	00	22	0B	41	00	00	00	00	00	00	00	00	00	00	00	-.".A... ..
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
020	F4	79	CD	9E	6F	1D	E3	8F	47	D9	22	BD	39	C8	42	D2	.y..o... G.".9.B.
030	DF	6F	CB	03	24	57	2B	DA	FB	FA	05	39	51	E4	CD	02	.o..\$W+. ...9Q...
040	02	02	02	01	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	22	0B	41	3F	FF	FF	FF	07	00	00	00	00	00	--	-.".A?.. ..
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

Ilustración 1: Último valor menor

Actividad 3

Desarrollar el algoritmo y el programa que ordene de manera ascendente un conjunto de datos ubicados en el banco 0 del registro 0X20 al 0X2F.

Ejemplo:

20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
0F	0E	0D	0C	0B	0A	09	08	07	06	05	04	03	02	01	FF

Solución:

20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	FF

Imagen 8: Ejemplo de ejercicio 3

Descripción

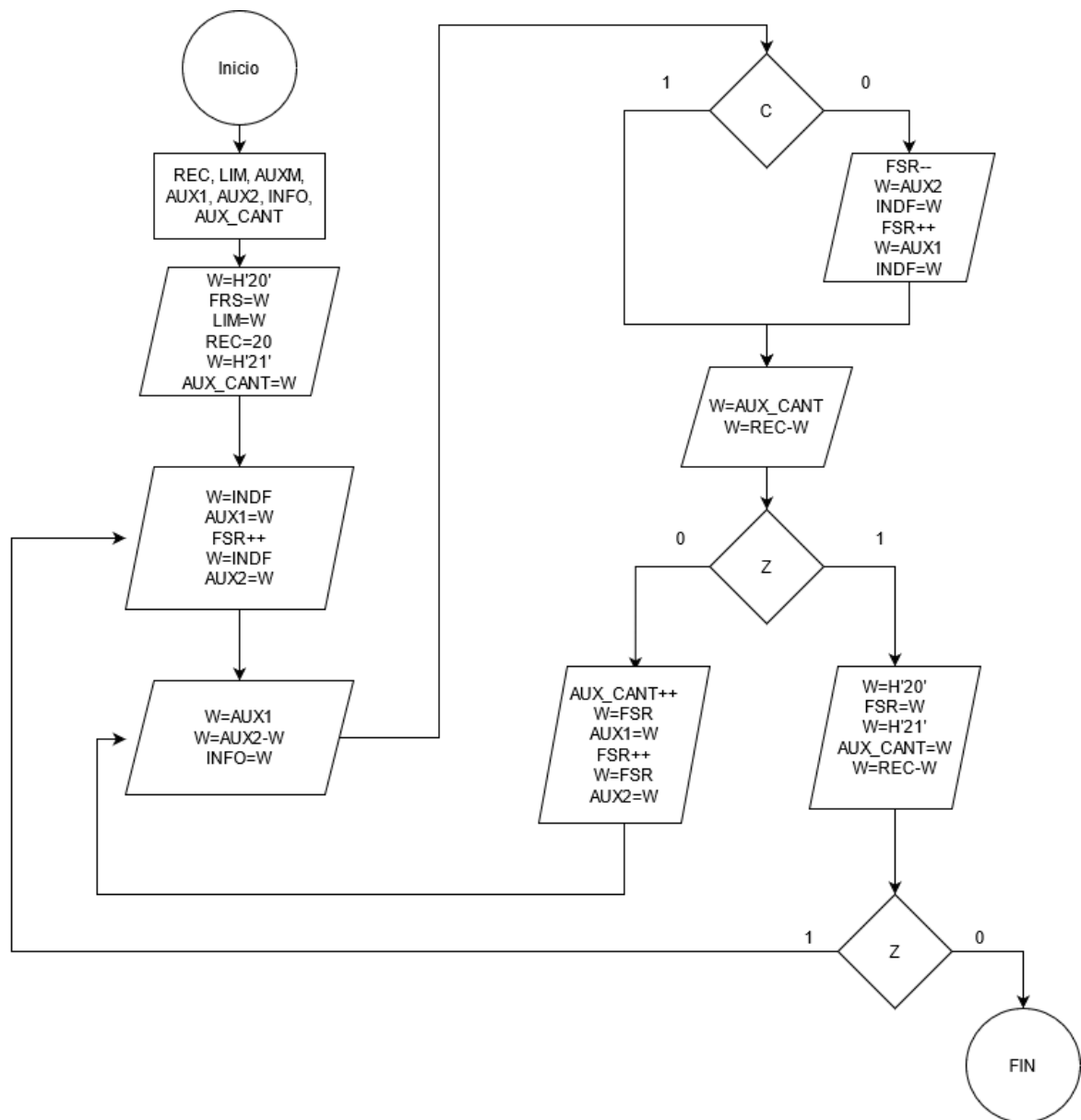
A partir del algoritmo de ordenamiento Bubble Sort, el número más grande se va recorriendo hasta el final de la lista y así sucesivamente con todos los números, se desarrolló el algoritmo en lenguaje ensamblador.

Se inicializan los apuntadores y el límite de la lista. Se realiza una revisión inicial a los 2 primeros valores. Se realiza una resta entre estos, dependiendo el comportamiento del bit de acarreo, si el valor del bit de acarreo es 0 se tiene que realizar un intercambio en las posiciones de los números, si el valor de acarreo es 1 continúa revisando.

Se revisa si el contador de la cantidad de valores checados es igual al límite de la lista, si es así se realizará un reseteo a los valores, tanto del FSR, el límite y el contador, por el contrario, se realizará una revisión al siguiente valor de la lista a apuntar y se realizará la comparación por medio de la resta.

En la sección de reseteo se revisará si el contador y el límite son iguales, si es así disminuye el límite, en caso contrario terminará el algoritmo.

Diagrama de Flujo



Código y Ejecución

```
PROCESSOR 16F877
#include <P16F877.INC>

REC      EQU H'40'; Informe de limite
LIM      EQU H'2F'; Limite de Rango
AUXM     EQU H'40'
AUX1     EQU H'41'
AUX2     EQU H'42'
INFO     EQU H'43'; Info resta
AUX_CANT EQU H'44'; Contador de iteraciones

ORG 0
GOTO     INICIO
ORG 5

INICIO:  ;Inicialización de variables
        MOVLW  H'20'
        MOVWF  FSR
        MOVLW  LIM
        MOVWF  REC
        MOVLW  H'21'
        MOVWF  AUX_CANT

REVISION:
        MOVF   INDF,W; Carga el valor de FSR en W
        MOVWF  AUX1; Carga el valor de w en aux1
        INCF   FSR;incrementa el apuntador FSR
        MOVF   INDF,W; Carga el valor de FSR en W
        MOVWF  AUX2; Carga el valor de w en aux2

RESTA:   ;Revision de numero menor
        MOVF   AUX1,W; Mueve el dato de aux1 a W
        SUBWF  AUX2,W; Del valor de AUX2 le restas W, lo pasa a W
        MOVWF  INFO; Información de la resta
        BTFSS  STATUS,0; Si c==1 revisa, si c==0 intercambia
        GOTO   INTERCAMBIO; Intercambia
        GOTO   REVISION2; Sigue revisando
-----
```

```

INTERCAMBIO:
    ;Intercambio entre valores
    DECF    FSR
    MOVF    AUX2,W
    MOVWF   INDF ;Menor a la izquierda
    INCF    FSR
    MOVF    AUX1,W
    MOVWF   INDF ;Mayor a la derecha

REVISION2:
    ;Revision si el contador y el limite son iguales
    MOVF    AUX_CANT,W
    SUBWF   REC,W
    BTFS    STATUS,Z; Si son iguales
    GOTC    RESETEO; Reinicia el ciclo
    GOTC    REV3; Sigue revisando

REV3:
    INCF    AUX_CANT
    MOVF    INDF,W; Carga el valor de FSR en W
    MOVWF   AUX1; Carga el valor de w en aux1
    INCF    FSR;incrementa el apuntador FSR
    MOVF    INDF,W; Carga el valor de FSR en W
    MOVWF   AUX2; Carga el valor de w en aux2
    GOTC    RESTA; Compara siguientes numeros

RESETEO:
    ; Reinicia el ciclo
    MOVLW   H'20'
    MOVWF   FSR
    MOVLW   H'21'
    MOVWF   AUX_CANT
    SUBWF   REC,W; Limite - aux_cant
    BTFS    STATUS,Z;Si es cero
    GOTC    ;Termina orden
    DECF    REC; Reduce limite
    GOTC    REVISION;Revisar valores
END

```

Código 3: Código del algoritmo de ordenamiento

Pruebas

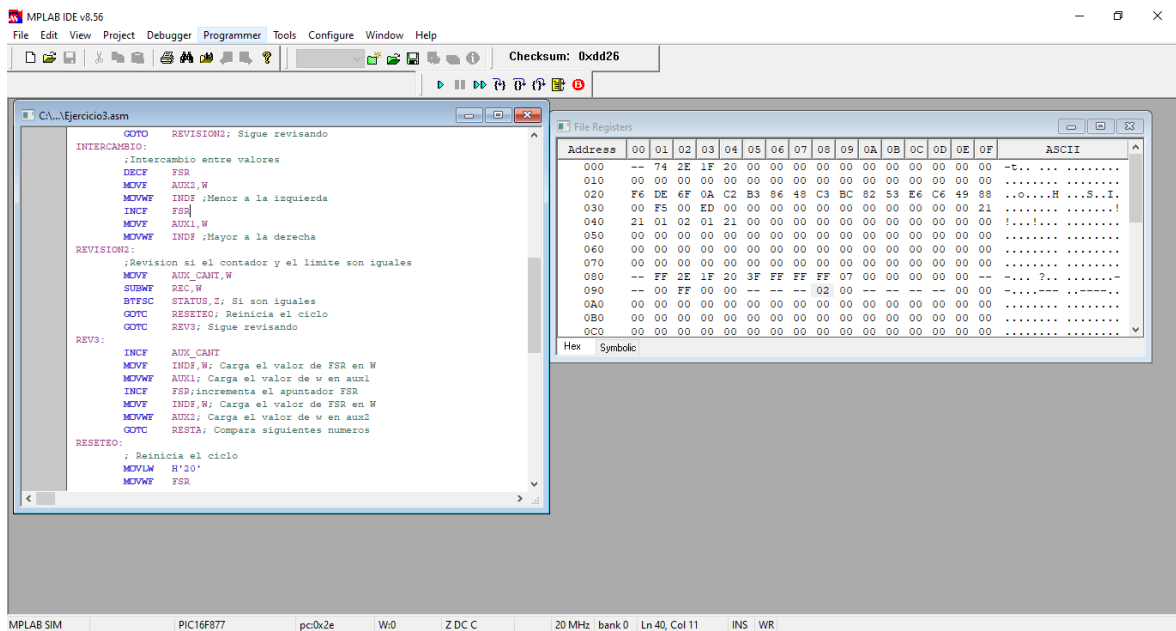


Imagen 9: Estado inicial del programa

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	74	2E	1F	20	00	00	00	00	00	00	00	00	00	00	00	-t.. ...
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
020	F6	DE	6F	0A	C2	B3	86	48	C3	BC	82	53	E6	C6	49	88	..o....H ...S..I.
030	00	F5	00	ED	00	00	00	00	00	00	00	00	00	00	00	21!
040	21	01	02	01	21	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	2E	1F	20	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

Imagen 10: Estado inicial FileRegister, prueba 1

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	74	00	1B	20	00	00	00	00	00	00	00	00	00	00	00	-t.. ...
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
020	0A	48	49	53	6F	82	86	88	B3	BC	C2	C3	C6	DE	E6	F6	.HISo...
030	00	F5	00	ED	00	00	00	00	00	00	00	00	00	00	00	21!
040	2F	E6	F6	10	21	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	1B	20	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

Imagen 11: Ordenamiento finalizado

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	--	74	2E	0F	20	00	00	00	00	00	00	00	00	00	00	00	-t.. ...
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
020	E6	BB	50	88	6B	CE	6C	6F	44	DC	B9	5F	2A	47	61	ED	..P.k.lo D..*Ga.
030	00	F5	00	ED	00	00	00	00	00	00	00	00	00	00	00	21!
040	21	20	26	06	21	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	2E	0F	20	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

Imagen 12: Prueba 2

datos y memoria, dándonos la posibilidad de generar programas con un flujo de datos que nos permita implementar lógicas que con otros direccionamientos serían más extensos, también hay que tener en cuenta que hay que ser cuidadosos con el manejo de este tipo de instrucciones en este modo de direccionamiento como es el caso de FSR y INDF.