Universidad Nacional Autónoma de México

Facultad de Ingeniería

Laboratorio de Microcomputadoras

Nombre del Profesor:

Amaranto de Jesús Dávila Jauregui

Semestre 2023-2

Grupo 3

Práctica 1

Nombre del alumno:

- Castelan Ramos Carlos
- Corona Nava Pedro Jair

Fecha de entrega: 13/03/2023

Práctica 1

Introducción.

Algunas de las características más importantes que tiene el microcontrolador son:

- 8K de memoria FLASH
- 368 bytes de memoria RAM
- 255 bytes de memoria EEPROM
- 35 instrucciones
- 5 puertos paralelos (A, B, C, D, E)
- Convertidor Analógico Digital
- Comunicación Serie Asíncrona
- Comunicación Serie Síncrona (paralela, I2C)
- Tres módulos temporizadores
- Dos módulos CCP que pueden operar como Comparación, Captura o PWM
- 14 posibles fuentes de interrupción

Los registros disponibles para el programador son:

W	Registro de trabajo W
PC	Registro Contador de Programa
STATUS	Registro de banderas

Tanto los registros PC y STATUS están ubicados en localidades de memoria RAM, dentro de los bancos en los que se divide los 368 bytes de memoria de datos, como se muestra en la Figura 1.1. Por otro lado, el registro STATUS, además de indicar el estado de lo que ocurrió en la última operación, se dispone de banderas que permiten seleccionar el banco de memoria RAM donde se desea acceder.

RP1	RP0	BANCO	UBICACIÓN
0	0	0	00H-7FH
0	1	1	80H-FFH
1	0	2	100H-17FH
1	1	3	180H-1FFH

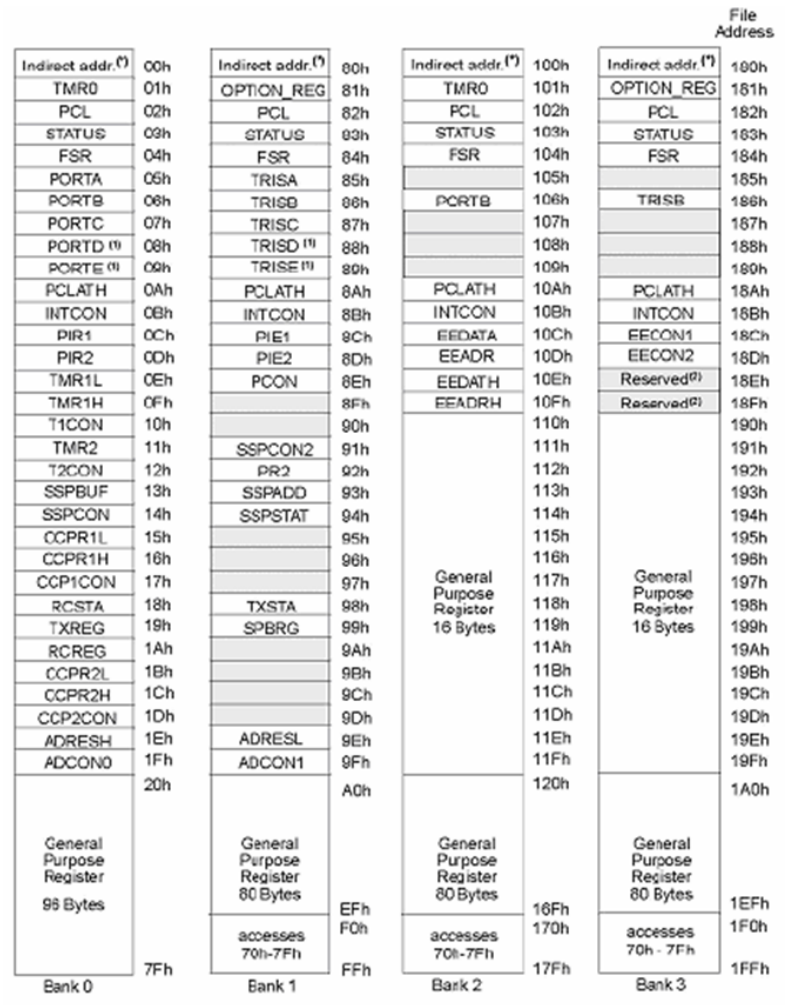


Figura 1.1: Mapa de memoria de datos.

La llamada gama baja y media de PIC's a la que pertenece el PIC16F877 tiene el siguiente conjunto de instrucciones, Mostrados en la Figura 1.2:

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 1fff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff		1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 1fff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00 0000 0000 1000		
SLEEP	-	Go into standby mode	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z	

Figura 1.2: Conjunto de Instrucciones del PIC 16F877

Por otro lado, en cuanto al entorno de desarrollo, se emplea MPLAB el cual es uno de los llamados Ambientes de Desarrollo Integrado IDE, que permite escribir, ensamblar y simular un programa, e incluso usando cierto hardware, se puede simular en circuito y programar al microcontrolador. Al iniciarse, se tiene algo similar a la siguiente interfaz (Figura 1.3):

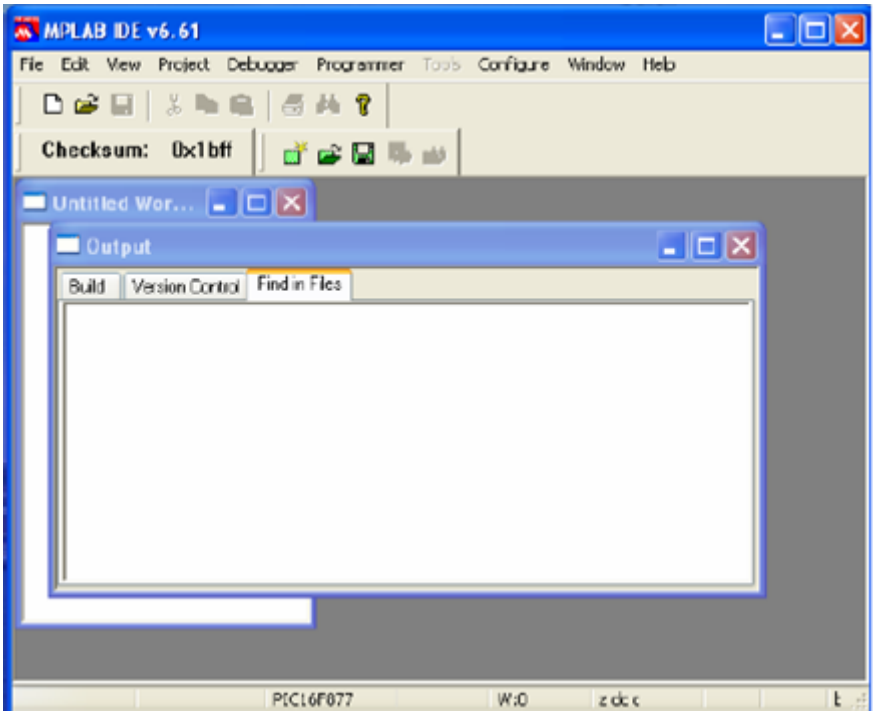


Figura 1.3: Entorno de MPLAB

Objetivos.

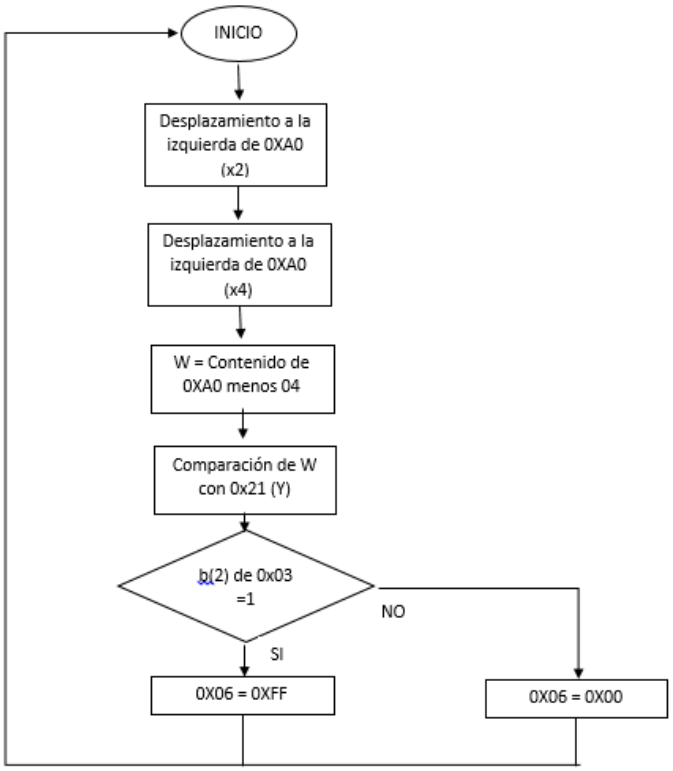
- Que el estudiante reconozca la programación en lenguaje ensamblador de microcomputadoras mediante la programación de tareas sencillas.
- Que el estudiante se familiarice con el uso del set de instrucciones de la microcomputadora PIC16F886 mediante el diseño de programas sencillos y la validación de éstos en el simulador básico de MPLAB.

Desarrollo.

Ejercicios.

1. Diseñar y simular un programa que verifique si un punto (x,y) pertenece a la recta y = 4x2 - 4. La variable x se encuentra en la localidad 0xA0, la variable y en la localidad 0x21 de la memoria dato. Si el punto (x,y) pertenece a la recta escribir 0xFF en la localidad 0x06, si no pertenece escribir 0x00.

Diagrama de flujo:



Breve explicación

Para realizar la multiplicación solicitada en la ecuación de la recta se hizo uso del desplazamiento hacia la izquierda (debido a que se va a multiplicar por una potencia de 2), para verificar el correcto funcionamiento fue necesario poner el carry en 0 antes y después de cada desplazamiento.

```
BCF 0X03, 0X00    ; PONER EN 0 EL CARRY
RLF 0X20          ; DESPLAZAMIENTO A LA IZQUIERDA DEL VALOR DE X (MULTIPLICARLO POR 2)
BCF 0X03, 0X00    ; PONER EN 0 EL CARRY
RLF 0X20          ; DESPLAZAMIENTO A LA IZQUIERDA DEL VALOR DE X (MULTIPLICARLO POR 4)
```

Posteriormente se realizó la resta presente en la ecuación de la recta, para ello se cargó el valor 04 en W para facilitar su uso.

```
MOVLW 0X04          ; CARGA LITERAL 0X04 EN W
SUBWF 0X20,0        ; RESTA DE 4X - 4 Y OBTENEMOS EL VALOR DE Y
```

Al realizar esta operación encontramos el valor de y a partir de una X dada. Con este dato obtenido procedemos a comparar dicho dato con el valor de Y que se encuentra en la localidad 0X21. Si son iguales significa que las coordenadas pertenecen a la recta y además el resultado será 0 por lo tanto se pondrá un 1 en la bandera Z del STATUS, así que verificamos dicha condición y nos movemos a una etiqueta u otra según sea el caso.

```
SUBWF 0X21, 0      ; COMPARACION DE AMBAS Y
BTFSS 0X03, 0X02   ; VERIFICAR EL BIT DE LA BANDERA Z DEL STATUS SI 1 PC=PC+1
```

Si pertenece se carga 0XFF en 0X06, de lo contrario se carga un 0X00.

```
PERTENECE MOVLW 0XFF   ; CARGA LITERAL 0XFF
          MOVWF 0X06   ; MOVEMOS LA LITERAL A 0X06
          GOTO INICIO        ; IR A LA ETIQUETA INICIO
NO_PERTENECE MOVLW 0X00        ; CARGA LITERAL 0X00
             MOVWF 0X06        ; MOVEMOS LA LITERAL A 0X06
             GOTO INICIO       ; IR A LA ETIQUETA INICIO
END
```

Código:

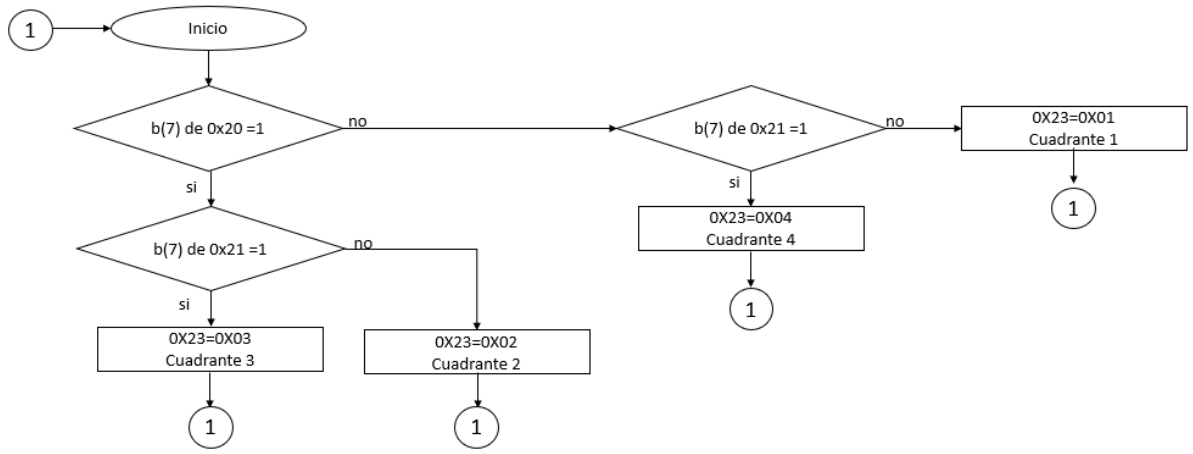
```
processor PIC16F877A  ; procesador a utilizar
org 0X00          ; Vector de reset
GOTO 0X05          ;Envia PC a la direccion 05
org 0X05          ; Inicio del programa
INICIO            ; Etiqueta INICIO
        BCF 0X03, 0X06    ; BIT CLEAR EN EL REG. DE LA BANDERA PARA ESTAR EN EL BANCO 1
        BSF 0X03, 0X05    ; BIT CLEAR EN EL REG. DE LA BANDERA PARA ESTAR EN EL BANCO 1

        BCF 0X03, 0X00    ; PONER EN 0 EL CARRY
        RLF 0X20        ; DESPLAZAMIENTO A LA IZQUIERDA DEL VALOR DE X (MULTIPLICARLO POR 2)
        BCF 0X03, 0X00    ; PONER EN 0 EL CARRY
        RLF 0X20        ; DESPLAZAMIENTO A LA IZQUIERDA DEL VALOR DE X (MULTIPLICARLO POR 4)

        MOVLW 0X04      ; CARGA LITERAL 0X04 EN W
        SUBWF 0X20,0      ; RESTA DE 4X - 4 Y OBTENEMOS EL VALOR DE Y

   BCF 0X03, 0X06    ; BIT CLEAR EN EL REG. DE LA BANDERA PARA ESTAR EN EL BANCO 0
        BCF 0X03, 0X05    ; BIT CLEAR EN EL REG. DE LA BANDERA PARA ESTAR EN EL BANCO 0
        SUBWF 0X21, 0     ; COMPARACION DE AMBAS Y
        BTFSS 0X03, 0X02  ; VERIFICAR EL BIT DE LA BANDERA Z DEL STATUS SI 1 PC=PC+1
        GOTO NO_PERTENECE ; NOS SALTAMOS A LAS INSTRUCCIONES PARA EL CASO DE QUE NO
PERTENEZCA A LA RECTA
     GOTO PERTENECE      ; NOS DIRIGIMOS A LAS INSTRUCCIONES CUANDO LAS COORDENADAS
PERTENECEN A LA RECTA
PERTENECE MOVLW 0XFF  ; CARGA LITERAL 0XFF
       MOVWF 0X06  ; MOVEMOS LA LITERAL A 0X06
```

```
                GOTO INICIO      ; IR A LA ETIQUETA INICIO
NO_PERTENECE MOVLW 0X00       ; CARGA LITERAL 0X00
        MOVWF 0X06       ; MOVEMOS LA LITERAL A 0X06
                GOTO INICIO      ; IR A LA ETIQUETA INICIO
END
```

2. Escribir y simular un programa que indique el cuadrante del plano cartesiano en el cual se encuentra el punto (x, y) dado por el contenido de las localidades 0x20 y 0x21 respectivamente. El cuadrante se debe indicar en la localidad 0x23. Considerar que los datos se encuentran representados en complemento a 2.

- Por ejemplo si el punto se encuentra en el tercer cuadrante se deberá escribir un 0x03 en la localidad 0x23.

Diagrama de flujo:



Breve explicación:

Para este código, hicimos uso del complemento a dos, donde si el bit más significativo tenía un uno, entonces el valor de este era negativo, es por eso que en base a pruebas de bit, condicionamos los valores asignados para x,y y dimos su correspondiente asignación de valor de cuadrante en la dirección de memoria 0X23.

Código:

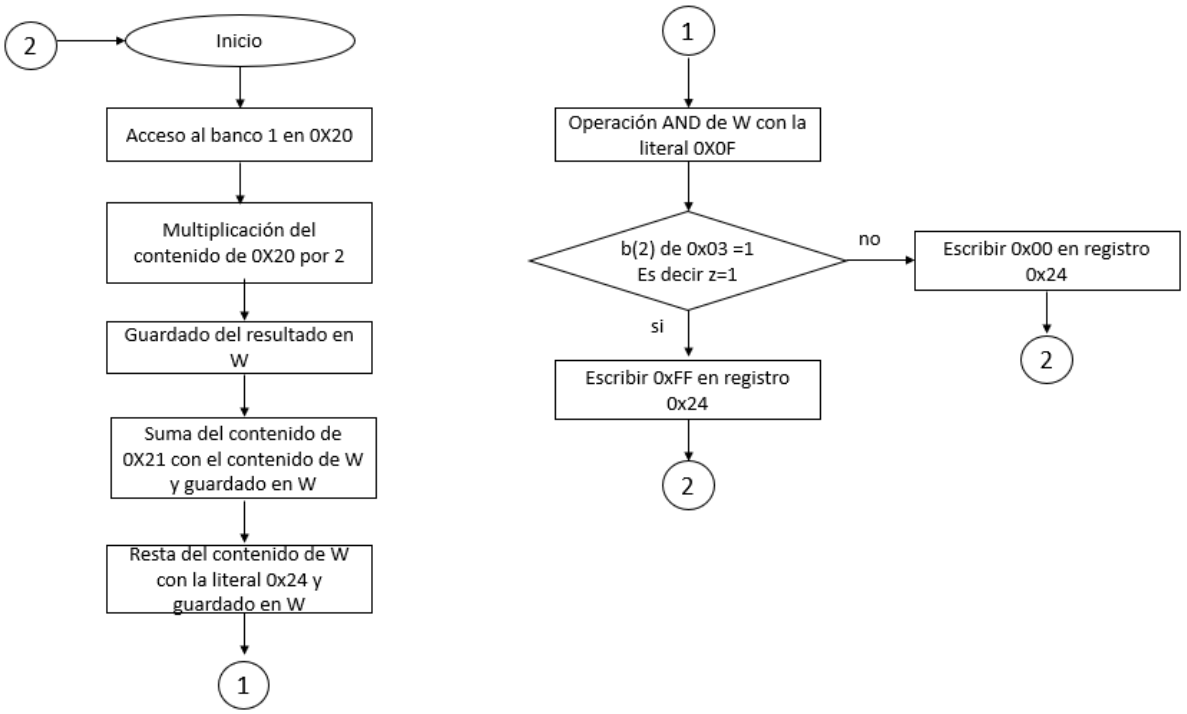
```
processor PIC16F877A  ; procesador a utilizar
org 0X00          ; Vector de reset
GOTO 0X05         ;Envia PC a la direccion 05
org 0X05          ; Inicio del programa
INICIO            ; ETIQUETA INICIO
        BCF 0X03, 0X05   ; BIT CLEAR EN EL REG. DE LA BANDERA PARA ESTAR EN EL BANCO 0
        BCF 0X03, 0X06   ; BIT CLEAR EN EL REG. DE LA BANDERA PARA ESTAR EN EL BANCO 0
        BTFSS 0X20, 0X07  ; VERIFICAR EL BIT 7 DE LA LOCALIDAD 0X20 SI 1 PC=PC+1
```

```
        GOTO UNO_CUATRO   ; ETIQUETA PARA EL CUANDRANTE 1 O 4
        BTFSS 0X21, 0X07  ; VERIFICAR EL BIT 7 DE LA LOCALIDAD 0X21 SI 1 PC=PC+1
        GOTO DOS      ; IR A LA ETIQUETA DOS PARA MOVER EL VALOR EN LA DIRECCION SOLICITADA
        MOVLW 0X03        ; CARGA LITERAL 0X03
        MOVWF 0X23        ; MUEBVE LA LITERAL A 0X23
GOTO INICIO          ; IR A LA ETIQUETA INICIO
DOS MOVLW 0X02       ; CARGA LITERAL 0X02
   MOVWF 0X23        ; MUEBVE LA LITERAL A 0X23
   GOTO INICIO       ; IR A LA ETIQUETA INICIO
UNO_CUATRO BTFSS 0X21, 0X07 ; VERIFICAR EL BIT 7 DE LA LOCALIDAD 0X21 SI 1 PC=PC+1
                    GOTO UNO   ; IR A LA ETIQUETA UNO
                    MOVLW 0X04 ; CARGA LITERAL 0X04
                    MOVWF 0X23 ; MUEBVE LA LITERAL A 0X23
                    GOTO INICIO ; IR A LA ETIQUETA INICIO
                    UNO MOVLW 0X01  ; CARGA LITERAL 0X01
                       MOVWF 0X23  ; MUEBVE LA LITERAL A 0X23
                       GOTO INICIO ; IR A LA ETIQUETA INICIO
END                ; FIN DEL PROGRAMA
```

3. Escribir y simular un programa que realice la siguiente operación: (0x02*[0x120]+[0x21]-0x24) &0x0F

- Si el resultado es cero escribir 0xFF en la localidad 0x24, si es diferente de cero escribir 0x00 en la localidad 0x24.
- Los valores entre [ ] representan el contenido de una localidad de memoria dato.
- Los valores que no están entre [ ] representan una literal.

Diagrama de flujo:



Código:

Para el desarrollo de este programa fue necesario primero colocar nuestra bandera de carry en cero, posteriormente hicimos un cambio de banco de memoria para acceder al registro 20 y realizar una multiplicación a través de un corrimiento por la izquierda, esta es la razón de que nuestro carry fuera un cero, enseguida almacenamos el valor en W y regresamos al banco de memoria donde realizamos una suma de W con el contenido del registro 0X21 y guardamos el resultado en el mismo registro, enseguida cargamos la literal 0x24 en W para poder realizar una resta con el contenido de 0X21 para que entonces se almacena en W y finalmente realizamos una operación AND con W y la literal 0X24, para poder guardar el resultado correspondiente en la dirección 0X24 del banco 1 ocupamos una prueba de bit de la localidad 0X03 con la bandera Z, recordando que si Z=0 significa que el resultado es diferente de cero, en cambio si Z=1 entonces el resultado es igual a cero.

```
processor PIC16f877A ;procesador a utilizar
org 0x00            ;vector de reset
Goto0x05                    ;envia pc a la direccion 05
org 0x05            ;origen del programa
```

```
INICIO
        BCF 0X03,0X05;MODIFICAMOS RP0
        BCF 0X03,0X05;MODIFICAMOS RP1
        CLRW ;LIMPIAMOS W
        BCF 0X03, 0 ;COLOCAMOS CARRY EN CERO

        ;ENTRO AL BANCO 1
        BSF 0X03, 0X06;MODIFICAMOS RP0
        BCF 0X03, 0X05;MODIFICAMOS RP1
        RLF 0X20 ;MULTIPLICANDO POR DOS
        CLRW;LIMPIAMOS W
        MOVF 0X20, 0 ;PASAMOS A W

        ;REGRESO AL BANCO 0
        BCF 0X03, 0X06;MODIFICAMOS RP0
        BCF 0X03, 0X05;MODIFICAMOS RP1

        ADDWF 0X21, 1; SUMAMOS W CON EL REGISTRO Y LO GUARDO EN EL REGISTRO
        MOVLW 0X24     ;CARGA LA LITERAL 0X24
        SUBWF 0X21, 0 ;RESTO W CON 0X21 Y GUARDO EN W
        ANDLW 0X0F ;A W OPERO LA LITERAL CON AND Y GUARDO EN W
        ;VERIFICAMOS EL RESULTADO
        BTFSS 0X03,2 ;STATUS - BIT DE BANDERA Z
        GOTO Z_UNO ; Significa que el resultado es cero
        GOTO Z_CERO ; Significa que el resultado es dif de cero

Z_UNO   MOVLW 0XFF     ;CARGA LA LITERAL 0XFF
                MOVWF 0X24 ;CARGA LA DIRECCIÓN
                GOTO INICIO ;VA ALA ETIQUETA INICIO
Z_CERO          MOVLW 0X00     ;CARGA LA LITERAL 0X00
                MOVWF 0X24;CARGA LA DIRECCIÓN
                GOTO INICIO;VA ALA ETIQUETA INICIO
END
```

Liga de videos de los códigos: https://drive.google.com/drive/folders/1iMLBunpag2rMo_h1iFO4_27Y5T9okoFq?usp=share_link

Conclusiones:

- Carlos Castelan Ramos: Para esta práctica, se logró comprender el uso y funcionamiento del set de instrucciones del PIC, así como el funcionamiento del mismo a través de sus bancos de memoria y sus registros reservados para partes principales como el STATUS. Considero que el nivel de los ejercicios fue un poco avanzado, sin embargo, en base a consultas al profesor de teoría y laboratorio se lograron llevar a cabo correctamente. Finalmente el IDE MPLAB, fue una herramienta de gran ayuda, por lo que se mejoró la habilidad del uso del mismo, es entonces que digo que se cumplieron con éxito los objetivos planteados.
- Corona Nava Pedro Jair: La realización de esta práctica fue muy provechosa ya que por fin pudimos aplicar los conocimientos teóricos que estuvimos adquiriendo en las sesiones previas, sobretodo el hecho de emplear las instrucciones para el microcontrolador y hacer programas funcionales con ellas como si se tratara de alguno de los lenguajes de programación con los que estamos más familiarizados. Además entendimos un poco más como funciona el IDE con el que vamos a trabajar la mayor parte de prácticas que es MPLAB, lo cual es muy importante ya que así nos vamos familiarizando con él de forma rápida y sencilla. Por todas estas razones puedo mencionar que se cumplieron con los objetivos propuestos y que la práctica se concluyó de manera satisfactoria en tiempo y forma