Facultad de ingeniería

Materia: Laboratorio de Microcomputadoras

REPORTE DE LA PRÁCTICA 2

Título: Programación en ensamblador direccionamiento indirecto.

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: 9 de marzo de 2025

Semestre: 2025-2

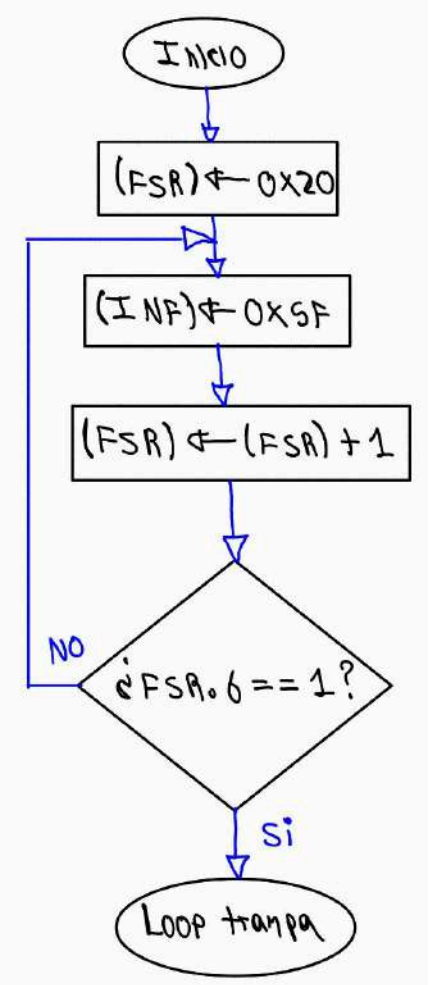
Objetivo: Analizar la programación en lenguaje ensamblador. Realizar algoritmos en lenguaje ensamblador empleando direccionamiento indirecto.

Ejercicios

Ejercicio 1: Escribir, comentar y ejecutar la simulación del siguiente programa:

```
PROCESSOR 16f877
INCLUDE <p16f877.inc>
        ORG 0 ; vector de reset
        GOTO INICIO

        ORG 5
INICIO:
        ;BCF STATUS,RP1
        ;BSF STATUS,RP0
        MOVLW 0X20; W<- 0X20
        MOVWF FSR ;(FSR) <- W
LOOP:
        MOVLW 0X5F ; W<- 0X5F
        MOVWF INDF ; (INDF) <- W
        INCF FSR ; (FSR) <- (FSR) + 1
        BTFSS FSR,6 ; ¿YA ACABE?  ¿FSR.6 == 1?
        GOTO LOOP ; NO, VE A LOOP
        GOTO $  ; SI, YA TERMINE: ETIQUETA GOTO

        END
```

Diagrama de flujo:



Funcionamiento de la solución:

Código:

```
INCLUDE <p16f877.inc>
    ORG 0 ; vector de reset
    GOTO INICIO

    ORG 5
INICIO:
    ;BCF STATUS,RP1
    ;BSF STATUS,RP0
    MOVLW 0X20; W<- 0X20
    MOVWF FSR ;(FSR) <- W
LOOP:
    MOVLW 0X5F ; W<- 0X5F
    MOVWF INDF ; (INDF) <- W
    INCF FSR ; (FSR) <- (FSR) + 1
    BTFSS FSR,6 ; ¿YA ACABE?  ¿FSR.6 == 1?
    GOTO LOOP ; NO, VE A LOOP
    GOTO $  ; SI, YA TERMINE: ETIQUETA GOTO

    END
```

secuencia 1:

File Registers

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
000	--	00	07	18	20	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	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

secuencia 2:

File Registers

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
000	--	00	0B	18	30	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	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F
030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

secuencia 3:

File Registers

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
000	--	00	0C	08	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	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F
030	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F	5F

El código realiza la carga del número “0x5F” desde la localidad 0x20 hasta la localidad 0x30, esto se logra ya que se cumple la condición de que prenda el bit 6 que carga la dirección “FSR”.

Análisis: utilizar el registro “FSR” permite acceder a múltiples direcciones usando un solo registro sin necesidad de modificar manualmente cada una. En otras palabras, este código es dinámico, ya que podría almacenar valores en cualquier rango de memoria, sin cambiar las instrucciones.

Ejercicio 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.

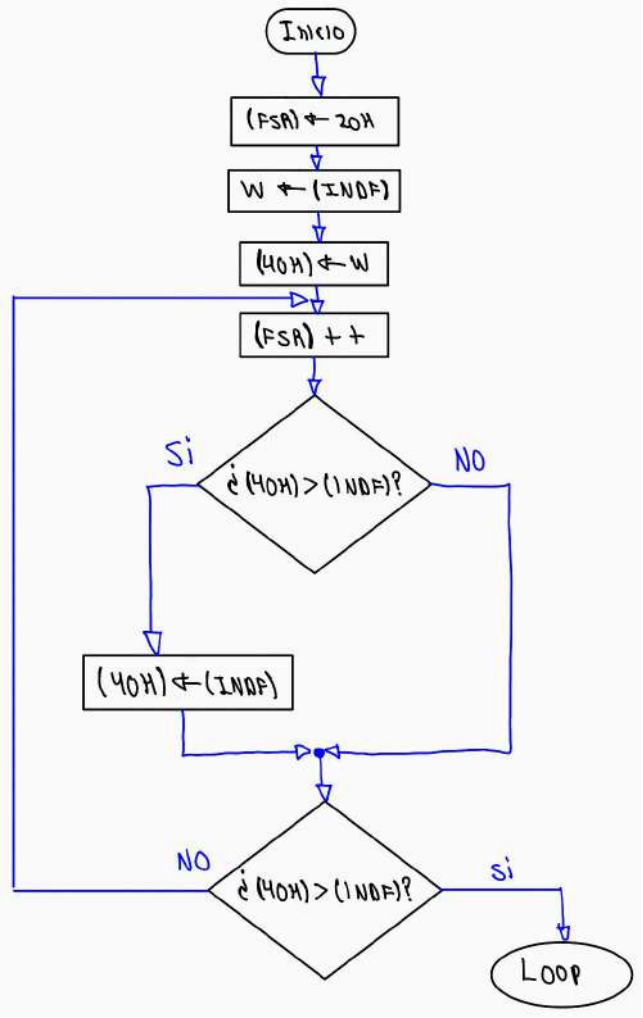
Ejemplo:

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

Diagrama de flujo:



Funcionamiento de la solución:

Código:

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

ORG 0
GOTO INICIO
ORG 5

INICIO:
MOVLW 0X20; W<- 0X20
MOVWF FSR ;(FSR) <- W
MOVF INDF,W
MOVWF 0X40

LOOP:
INCF FSR, 1
MOVF INDF, W
SUBWF 0X40, W
BTFSS STATUS, C
GOTO MENOR
GOTO MAYORIGUAL
    MAYORIGUAL:
    MOVF INDF, W
    MOVWF 0X40
    BTFSS FSR, 6
    GOTO LOOP

    GOTO $

    MENOR:
    BTFSS FSR,6
    GOTO LOOP
    GOTO $

    END
```

prueba 1 - secuencia 1:

File Registers

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
000	--	00	00	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	0B	2C	7D	1E	6F	10	DC	1F	50	FC	AD	2F	D8	7F	0F	81
030	BB	90	F4	CF	1A	DA	25	4E	A6	57	87	E1	08	20	9C	8C
040	00	00	00	00	00	00	00	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	09	0A	0B	0C	0D	0E	0F
000	--	00	13	0B	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	0B	2C	7D	1E	6F	10	DC	1F	50	FC	AD	2F	D8	7F	0F	81
030	BB	90	F4	CF	1A	DA	25	4E	A6	57	87	E1	08	20	9C	8C
040	08	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

prueba 2 - secuencia 1:

File Registers

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
000	--	00	0C	1B	21	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	A7	33	C6	5B	E0	F1	94	1C	5C	7D	2C	E0	AE	F2	CD	38
030	2F	12	C2	98	95	8B	91	B9	C8	7A	76	3B	F1	2E	0F	8F
040	A7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

prueba 2 - secuencia 2:

File Registers

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
000	--	00	13	0B	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	A7	33	C6	5B	E0	F1	94	1C	5C	7D	2C	E0	AE	F2	CD	38
030	2F	12	C2	98	95	8B	91	B9	C8	7A	76	3B	F1	2E	0F	8F
040	0F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Análisis: El código funciona de manera adecuada al recorrer la memoria correctamente y almacenar el valor mínimo, cuando FSR llega a `0x40`, el programa se detiene y el menor valor queda almacenado en esa dirección, El uso de direccionamiento indirecto permite optimizar el código y hacer más dinámico el acceso a los datos en la memoria, Este código no interactúa directamente con periféricos como puertos de entrada/salida o temporizadores, ya que trabaja únicamente con la memoria RAM interna del PIC

Ejercicio 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:

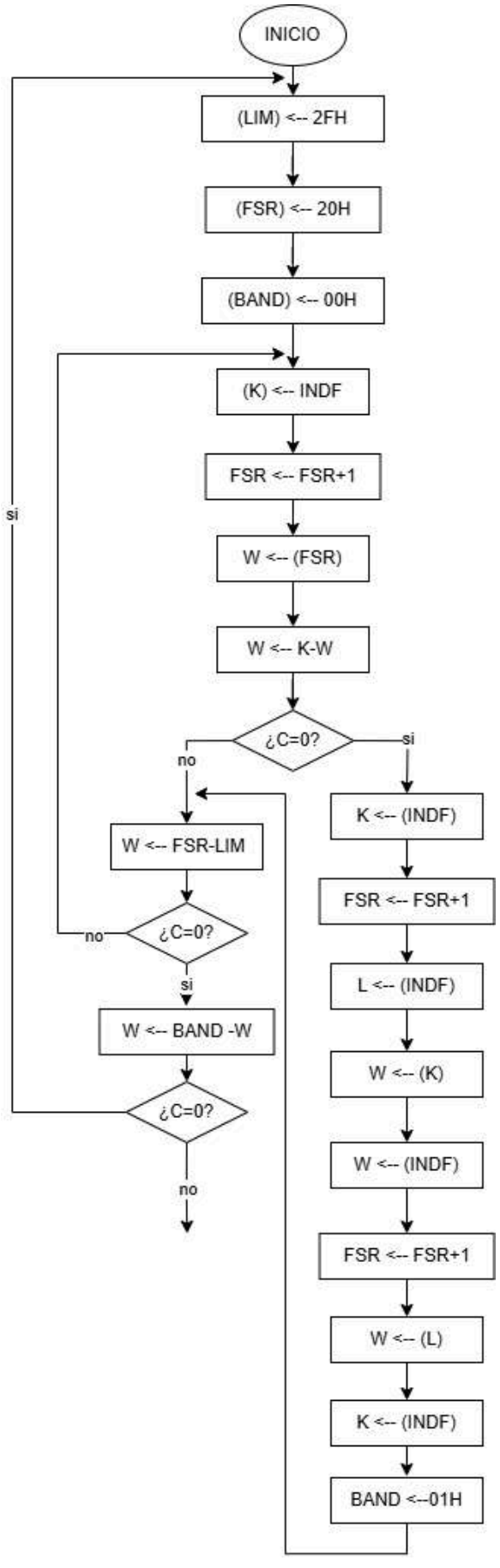
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
0F	**0E**	**0D**	**0C**	**0B**	**FF**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**0A**	**01**

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**

a. Comprobar el funcionamiento de su programa con distintos conjuntos de datos.

Diagrama de flujo:

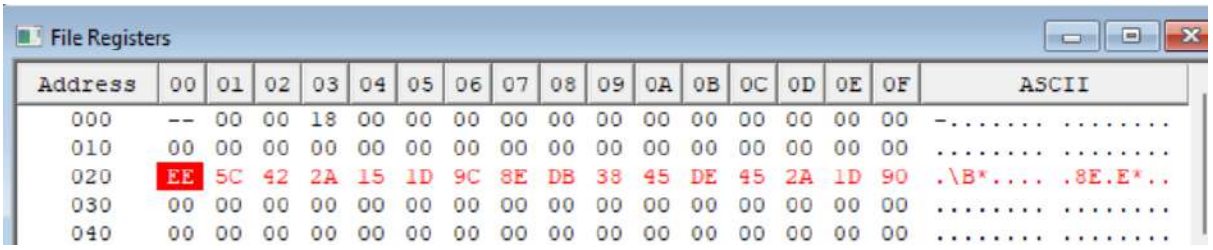


Funcionamiento de la solución:

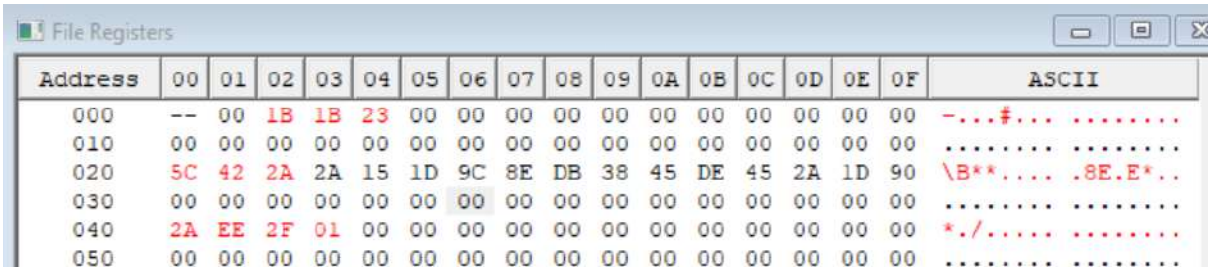
Código:

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

K   equ H'40'   ;aux1
L   equ H'41'   ;aux2
LIM equ H'42'   ;aux limite
BAND equ H'43'  ;bandera orden

    ORG 0
    GOTO INICIO
    ORG 5

INICIO:  MOVLW H'2F'
         MOVWF LIM      ; (LIM)<-- W límite de la lista
         MOVLW H'20'    ;  W<-- primer número
         MOVWF FSR      ; (FSR)<--W
         MOVLW H'00'
         MOVWF BAND     ; (BAND)<--W '0'

NUM1:    MOVF INDF, W   ; W<--INDF
         MOVWF K        ; (K)<--W

NUM2: INCF FSR          ; FSR<--FSR+1
      MOVF INDF, W      ; W<--INDF

COMPARA: SUBWF K, W         ; W<--(K-W)
         BTFSC STATUS, C    ;¿C=0?
         GOTO ORDENA        ; C=0
         GOTO FINAL     ; c=1

ORDENA: MOVF INDF, W    ; W<--INDF
        MOVWF K         ; (K)<--W
        DECF FSR        ; FSR<--FSR+1
        MOVF INDF, W    ; W<--INDF
        MOVWF L         ; L<--W
        MOVF K, W       ; W<--K
        MOVWF INDF

        INCF FSR        ; FSR<--FSR+1
        MOVF L, W       ; W<--L
        MOVWF INDF

        MOVF INDF, W    ; W<-- INDF
        MOVWF K         ; K<--W

        MOVLW H'01'
        MOVWF BAND      ; BAND<--'1'

FINAL:
        ;comprueba que llegue al final de la lista
        MOVF LIM, W         ; W<--LIM
        SUBWF FSR, W        ; W<--(FSR-LIM)
        BTFSS STATUS, C     ; ¿C=0?
        GOTO NUM1           ; c=1

        ;revisa que la lista este ordenada
        MOVLW H'01'         ; C=0
        SUBWF BAND, W       ; W <-- (BAND - W)
        BTFSS STATUS, C     ; ¿C=0?
        GOTO $              ; C=1
        GOTO INICIO         ; C=0

        END
```

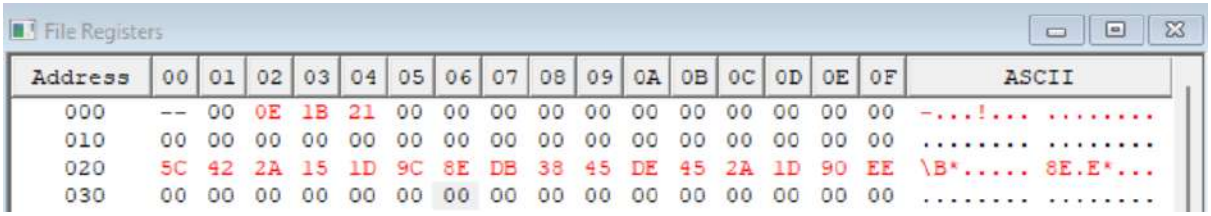
Prueba 1 - Secuencia 1



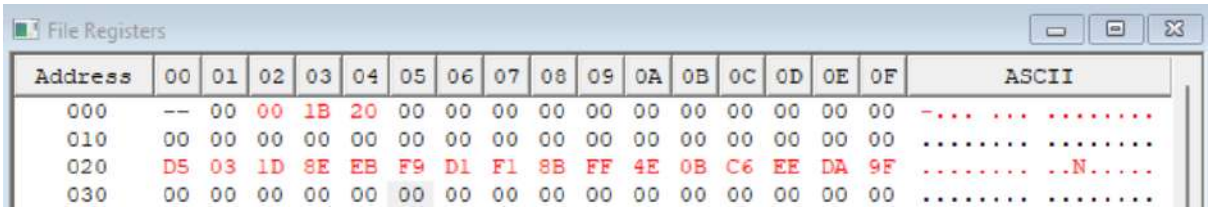
Prueba 1 - Secuencia 2



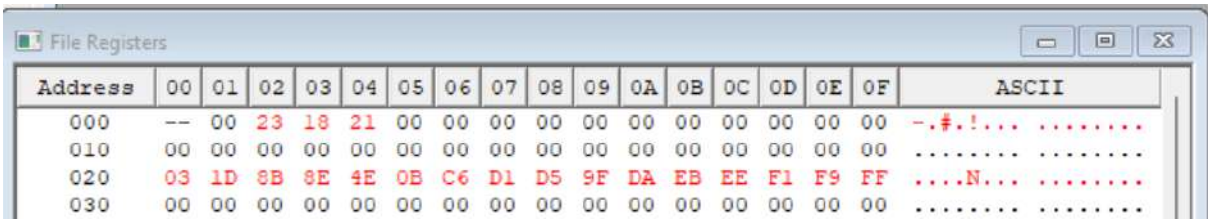
Prueba 1 - Secuencia 3



Prueba 2 - Secuencia 1



Prueba 2 - Secuencia 2



Análisis:
El código implementa bubble sort, va comparando pares de números almacenados en k y L, después realiza el intercambio si es necesario, repite el proceso hasta que estén ordenados completamente. Utiliza la variable Lim para definir el límite de la lista de números, y usa BAND para indicar si ha terminado.

Conclusiones:

Martínez Pérez Brian Erik

En esta práctica logramos crear algoritmos que tenían como propósito principal comprender el uso de instrucciones con modo de direccionamiento indirecto, para ello tuvimos que comprender que la misma instrucción hace la referencia a otra dirección en memoria donde obtendremos el dato deseado, con ayuda de este tipo de direccionamiento podemos crear algoritmos más dinámicos, y evitar realizar modificaciones exageradas a nuestros códigos.

Nuñez Rodas Abraham Enrique

Vicenteño Maldonado Jennifer Michel
Para esta práctica utilizamos direccionamiento indirecto, para realizar algoritmos de búsqueda y ordenamiento, también utilizamos bucles , banderas y variables auxiliares. La utilización de direccionamiento indirecto nos permite recorrer listas o matrices, también es útil al momento de manejar un buffer y arreglos.

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 3Elnu

Programación de microcontroladores y electrónica. (2021, 17 noviembre). Como configurar los puertos del PIC16F887 como entradas/salidas digitales, usando MIKROC. [Vídeo]. YouTube. https://www.youtube.com/watch?v=6U4i8d8Wugs

Jorge APC. (2024b, abril 21). PIC16F887 miKroC PRO Configuración de entradas y salidas [Vídeo]. YouTube. https://www.youtube.com/watch?v=vKTNDlhwt1c