M16

Universidad Nacional Autónoma de México

Facultad de Ingeniería

Semestre 2023-2

Laboratorio de Microcomputadoras

Práctica 4:

Puertos Paralelos E/S

Profesor:

M.I. Ruben Anaya García

Alumnos:	Número de cuenta:
Barreiro Valdez Alejandro	317520888
Zepeda Baeza Jessica	317520747

Grupo Laboratorio: 4

Grupo Teoría: 1

Fecha de realización: 21 de marzo de 2023

Fecha de entrega: 28 de marzo de 2023

Desarrollo

Para esta práctica se buscó crear programas en ensamblador que hicieran uso de los puertos paralelos del microcontrolador PIC16F877, configurándolos como entradas o salidas. El microcontrolador contiene 5 puertos de diferente tamaño: A (6 bits), B (8 bits), C (8 bits), D (8 bits) y E (3 bits), que se acceden por el nombre PORTX. Sin embargo, para definirlos como entrada y salida es necesario cambiar el valor de los registros TRISX ubicados en el banco 1, haciéndolo 0 para indicar que es una salida y 1 para una entrada. Además, otra configuración inicial se realiza para los puertos A y E, donde se indica en el registro ADCON1 (en el banco 1) que dichos puertos serán utilizados como digitales al asignarle el valor de 06H.

Para esta práctica se utilizaron los puertos A y B como entrada y salida respectivamente ya que el puerto A está conectado a un Dip Switch y el puerto B a un LED por bit. Los programas desarrollados fueron:

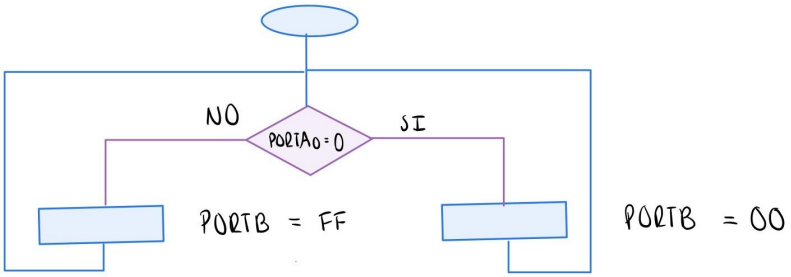
1. Un programa que utiliza dos puertos paralelos del microcontrolador configurados como entrada y salida respectivamente. El bit más significativo de la entrada se muestra en la salida.
2. Un programa que utiliza dos puertos paralelos: uno de entrada y uno de salida. La entrada indica la acción que se realizará en la salida como se muestra en la tabla:

DATO **PUERTO A**	ACCION **PUERTO B**	*Ejecución*
0x00	Todos los leds apagados	*00000000*
0x01	Todos los leds encendidos	*11111111*
0x02	Corrimiento del bit más significativo hacia la derecha	*10000000* *01000000* *00100000* …….. *00000001*
0x03	Corrimiento del bit menos significativo hacia la izquierda	*00000001* *00000010* *00000100* …….. *10000000*
0x04	Corrimiento del bit más significativo hacia la derecha y a la izquierda	*10000000* *01000000* …….. *00000001* *00000010* …….. *10000000*
0x05	Apagar y encender todos los bits.	*00000000* *11111111*

Algoritmos

Ejercicio 1

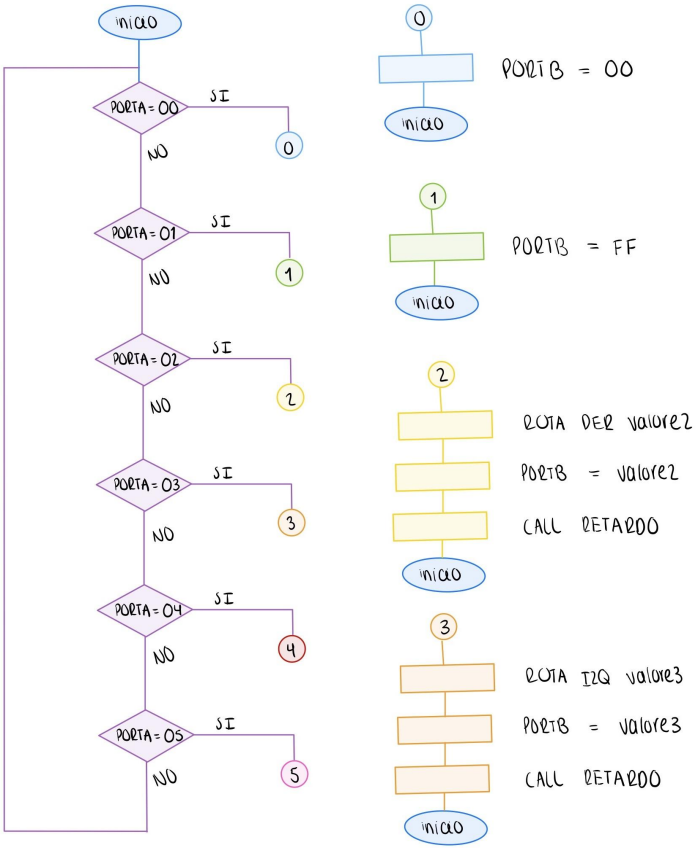
Para el Ejercicio 1 se realizó la siguiente carta ASM:

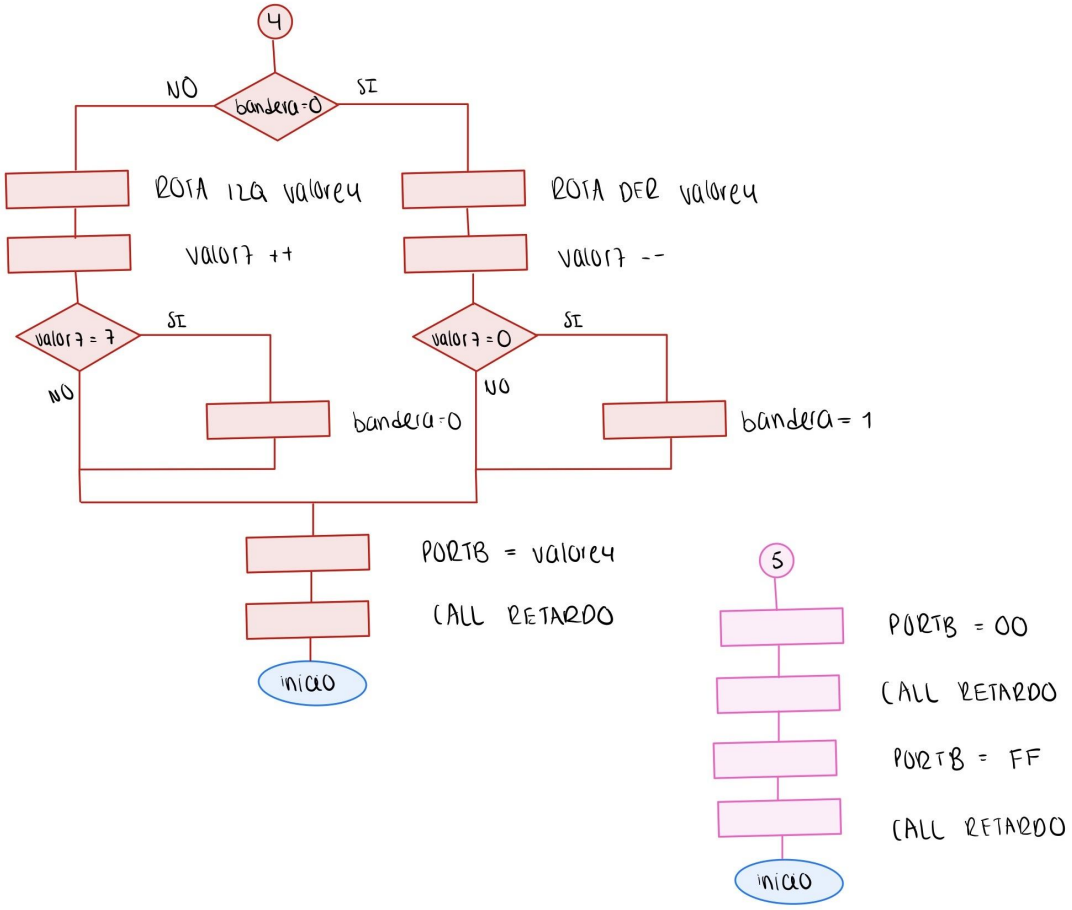


El programa consiste en un ciclo infinito donde se verifica el valor del bit menos significativo del puerto A. Si el valor es 0, el puerto B obtiene el valor de 00 que se ve como ningún LED prendido. Si el valor es 1, el puerto B obtiene un valor de FF que se ve como todos los LEDS encendidos.

Ejercicio 2

Para el Ejercicio 2 se realizó la siguiente carta ASM:





Para el ejercicio 2 se tiene un flujo principal donde se verifica qué número está guardado en el Puerto A, empieza verificando si es 0. En caso de que no lo sea verifica si es 1 y así sucesivamente. Cuando encuentra el número que guarda el puerto A, salta a la respectiva función; cuando acaba la función regresa al inicio del programa a verificar el siguiente número.

La función del estado 0 asigna puros ceros al Puerto B mientras la función del estado 1 asigna puros unos al Puerto B. La función 2 guarda un valor en una localidad de memoria (que se inicializa con 10000000) y al entrar al bloque de la función se hace una rotación a la derecha en esa localidad. Una vez rotado, se pasa el valor al puerto B y se llama una subrutina de retardo. Lo mismo sucede con la función 3 pero la rotación se hace a la izquierda con otro valor guardado en memoria que se inicializa con (00000001).

La cuarta función analiza el valor de una localidad de memoria llamada bandera, que indicará hacia dónde hacer el corrimiento: 0 es derecha y 1 es izquierda. Si la bandera es 0, se hace la rotación a la derecha y se decrementa un contador inicializado con 7. De esta forma se hará la rotación hasta que el contador llegue a 0 y cuando esto pase, se cambia el valor de la bandera a 1 para ahora moverse a la izquierda. El mismo procedimiento se hace al rotar a la izquierda con la diferencia que el contador aumenta hasta llegar a 7 y una vez

que llega vuelve a cambiar la bandera a 0. Al final de actualizar el contador y la rotación, se pasa el valor de la localidad al puerto B y se llama a la subrutina de retardo.
La función 5 le asigna el valor de 00 al puerto B, llama un retardo y luego le asigna un valor de FF para volver a llamar un retardo.

Programas comentados

Ejercicio 1

```
PROCESSOR 16F877
INCLUDE <P16F877.INC>

    ORG 0 ;Vector de reset
    GOTO INICIO

    ORG 5
INICIO:
    CLRF PORTA      ; Limpia PORTA
    BSF STATUS,RP0  ; Cambia a banco 1
    BCF STATUS,RP1
    MOVLW 06H       ;Define puertos A y E como digitales
    MOVWF ADCON1
    MOVLW H'3F'     ;Configura puerto A como entrada
    MOVWF TRISA
    CLRF TRISB      ;Configura puerto B como salida
    BCF STATUS,RP0  ;Cambia al banco
LOOP:               ;Inicia Loop
    BTFSC PORTA,0   ;Checa el primer bit del puerto A
    GOTO CERO       ;Si es cero va a CERO
    MOVLW 0XFF      ;Si no es cero:
    MOVWF PORTB     ;puertoB = FF
    GOTO LOOP       ;Regresa al Loop
CERO:
    CLRF PORTB      ;puertoB = 00
    GOTO LOOP       ;Regresa al Loop
    END
```

Ejercicio 2

```
PROCESSOR 16F877
INCLUDE <P16F877.INC>

valor1 equ h'21'        ;Variable para el retardo
valor2 equ h'22'        ;Variable para el retardo
valor3 equ h'23'        ;Variable para el retardo
cte1 equ 20h            ;Constante para el retardo
cte2 equ 50h            ;Constante para el retardo
cte3 equ 60h            ;Constante para el retardo
estado0 equ 0x00        ;Valor estado 0
estado1 equ 0x01        ;Valor estado 1
estado2 equ 0x02        ;Valor estado 2
estado3 equ 0x03        ;Valor estado 3
estado4 equ 0x04        ;Valor estado 4
estado5 equ 0x05        ;Valor estado 5
banderarota equ h'24'   ;Bandera rotación: 0=der, 1=izq
valore2 equ h'25'       ;Variable para el estado 2
valore3 equ h'26'       ;Variable para el estado 3
valore4 equ h'27'       ;Variable de salida estado 4
valor7 equ h'28'        ;Contador para estado 4

    ORG 0               ;Vector de reset
    GOTO INICIO

    ORG 5
INICIO:
    CLRF PORTA          ;Limpia PORTA
    BSF STATUS,RP0      ;Cambia a banco 1
    BCF STATUS,RP1
    MOVLW 06H           ;Define puertos A y E como digitales
    MOVWF ADCON1
    MOVLW H'3F'         ;Configura puerto A como entrada
    MOVWF TRISA
    CLRF TRISB          ;Configura puerto B como salida
    BCF STATUS,RP0      ;Cambia al banco 00
    BCF STATUS,C        ;Limpia el carry
    CLRF banderarota    ;Limpia la bandera de rotación
```

```
    MOVLW 0X80
    MOVWF valore2       ;valore2 = 80
    MOVWF valore4       ;valore4 = 80
    MOVLW 0X01
    MOVWF valore3       ;valore3 = 01
    MOVLW 0X07
    MOVWF valor7        ;valor7 = 07


LOOP:                   ;Determinar en qué estado se está
    MOVLW estado0
    XORWF PORTA,W       ;Se compara puertoA == estado0
    BTFSC STATUS,Z      ;Saltar si no son iguales
    GOTO ECERO          ;Ir a ECERO si son iguales

    MOVLW estado1
    XORWF PORTA,W       ;Se compara puertoA == estado1
    BTFSC STATUS,Z      ;Saltar si no son iguales
    GOTO EUNO           ;Ir a EUNO si son iguales

    MOVLW estado2
    XORWF PORTA,W       ;Se compara puertoA == estado2
    BTFSC STATUS,Z      ;Saltar si no son iguales
    GOTO EDOS           ;Ir a EDOS si son iguales

    MOVLW estado3
    XORWF PORTA,W       ;Se compara puertoA == estado3
    BTFSC STATUS,Z      ;Saltar si no son iguales
    GOTO ETRES          ;;Ir a ETRES si son iguales

    MOVLW estado4
    XORWF PORTA,W       ;Se compara puertoA == estado4
    BTFSC STATUS,Z      ;Saltar si no son iguales
    GOTO ECUATRO        ;Ir a ECUATRO si son iguales

    MOVLW estado5
    XORWF PORTA,W       ;Se compara puertoA == estado5

    BTFSC STATUS,Z      ;Saltar si no son iguales
    GOTO ECINCO         ;Ir a ECINCO si son iguales
    GOTO LOOP           ;Regresar al LOOP

ECERO:                  ;Estado 0
    CLRF PORTB          ;Se apaga la salida
    GOTO LOOP           ;Se revisa la entrada

EUNO:                   ;Estado 1
    MOVLW 0XFF
    MOVWF PORTB         ;puertoB = FF (se prende salida)
    GOTO LOOP           ;Se revisa la entrada

EDOS:                   ;Estado 2
    RRF valore2         ;Rotar a la derecha valore2
    MOVF valore2,W
    MOVWF PORTB         ;puertoB = valore2
    CALL RETARDO        ;Se llama el retardo
    GOTO LOOP           ;Se revisa la entrada

ETRES:                  ;Estado 3
    RLF valore3         ;Rotar a la izquierda valore3
    MOVF valore3,W
    MOVWF PORTB         ;puertoB = valore3
    CALL RETARDO        ;Se llama al retardo
    GOTO LOOP           ;Se revisa la entrada

ECUATRO:                ;Estado 4
    BTFSS banderarota,0 ;Checa el valor de la bandera de rotación
    GOTO ROTADER        ;Si es cero rota a la derecha
    GOTO ROTAIZQ        ;Si es uno rota a la izquierda
ROTADER:                ;Rotación a la derecha
    RRF valore4         ;Rota a la derecha valore4
    DECF valor7         ;Decrementa el contador
    CLRW                ;Limpia W
    XORWF valor7,W      ;Checar si el contador es cero
    BTFSS STATUS,Z
```

```
    GOTO FINAL          ;Si no es cero se va directo a FINAL
    BSF banderarota,0   ;Si es cero cambia la bandera para rotar izq
    GOTO FINAL          ;Ir a FINAL
ROTAIZQ:                ;Rotación a la izquierda
    RLF valore4         ;Rota a la izquierda los valores de valore4
    INCF valor7         ;Incrementar el contador
    MOVLW 0X07          ;W = 07
    XORWF valor7,W      ;Checar si el contador es 7
    BTFSS STATUS,Z
    GOTO FINAL          ;Si no es 7 se va directo a FINAL
    BCF banderarota,0   ;Si es 7 se cambia la bandera para rotar der
    GOTO FINAL          ;Ir a FINAL
FINAL:
    MOVF valore4,W
    MOVWF PORTB         ;puertoB = valore4
    CALL RETARDO        ;Llamar al retardo
    GOTO LOOP           ;Se revisa la entrada

ECINCO:                 ;Estado 5
    CLRW
    MOVWF PORTB         ;puertoB = 00
    CALL RETARDO        ;Se llama al retardo
    MOVLW 0XFF
    MOVWF PORTB         ;puertoB = FF
    CALL RETARDO        ;Se llama al retardo
    GOTO LOOP           ;Se revisa la entrada


 RETARDO                ;Subrutinan retardo
     MOVLW cte1
     MOVWF valor1       ;valor1 = cte1
 tres
     MOVLW cte2
     MOVWF valor2       ;valor2 = cte2
 dos
     MOVLW cte3
     MOVWF valor3       ;valor3 = cte3
 uno
     DECFSZ valor3      ;decrementar valor 3
     GOTO uno           ;ir a uno
     DECFSZ valor2      ;decrementar valor 2
     GOTO dos           ;ir a dos
     DECFSZ valor1      ;decrementar valor 1
     GOTO tres          ;ir a tres
     RETURN             ;fin subrutina

     END
```

Conclusiones y/o comentarios

Zepeda Baeza Jessica:

A pesar de trabajar con puertos como salidas desde la práctica anterior, está práctica permitió tanto trabajar con puertos como entradas (en forma de Dip Switch) como con puertos de salida (en forma de LED). Además se aprendió la configuración de los puertos para trabajarlos de forma digital lo que permiten seleccionar entre diferentes opciones de salida como se vio en los ejercicios realizados. Ambos ejercicios permitieron simular como un menú donde dependiendo la entrada se obtiene una salida. Estas salidas fueron sencillas de incluir debido a las prácticas anteriores porque el prendido y apagado de LEDS así como las rotaciones ya se habían programado. Al igual se hicieron uso de herramientas previamente vistas como el uso de XOR y banderas para comprobar si hay números iguales, el uso de subrutinas de retardo y el cambio de bancos para configurar los puertos.

Barreiro Valdez Alejandro:

En esta práctica se pudo trabajar con puertos de entrada y de salida de manera paralela. La salida ya se había utilizado, pero en esta práctica se agregó una entrada y se realizaron funcionalidades a partir de esa entrada. Los puertos se trabajaron de manera digital para poder interpretar los datos de la mejor manera. En el primer ejercicio se buscó prender o apagar los LEDs a partir de la entrada. Este fue el primer acercamiento de cómo configurar cada uno de los puertos y cómo estos pueden funcionar tomando la entrada para generar una salida. El segundo ejercicio fue un poco más complejo ya que se tenían varias entradas y varias salidas. Algunas de las salidas ya habían sido programadas por lo que no hubo tanto problema, pero la que generó más dificultades fue la de rotar hacia la izquierda y la derecha. Se utilizó una bandera, el valor de salida y un contador para esta salida. Se utilizó el conocimiento de prácticas anteriores para aplicarlo a un sistema mínimo donde se programaron puertos paralelos de entrada y de salida.