FACULTAD DE INGENIERÍA, UNAM

LABORATORIO DE MICROCOMPUTADORASS

SEMESTRE 2023-2

GRUPO 11

PREVIO PRÁCTICA 8

PUERTOS PARALELOS E/S, PUERTO SERIE

NOMBRE DEL ALUMNO:

ARRIAGA MEJÍA JOSÉ CARLOS

PROFESOR

ING. ROMAN V. OSORIO COMPARAN

FECHA DE ENTREGA: **12 DE MAYO DE 2023** **CALIFICACION**

Objetivo

Realización de programas a través de programación en C y empleo del puerto serie para visualización y control.

```
#include <16f877.h> //biblioteca del micro
#fuses HS,NOPROTECT, //parámetros físicos - eléctricos del controlador
#use delay(clock=20000000) // 20 MHz establece el reloj a utilizar
// reserva la memoria donde esta el bootloader
#org 0x1F00, 0x1FFF void loader16F877(void) {}
// El código prende y apaga 1 bit de la salida portb cada segundo.
void main(){
    while(1){
        output_b(0x01); // '00000001' salida para portb
        delay_ms(1000); // retardo de 1s
        output_b(0x00); // '00000000' salida para portb
        delay_ms(1000); // retardo de 1s
    } //while
}//main
```

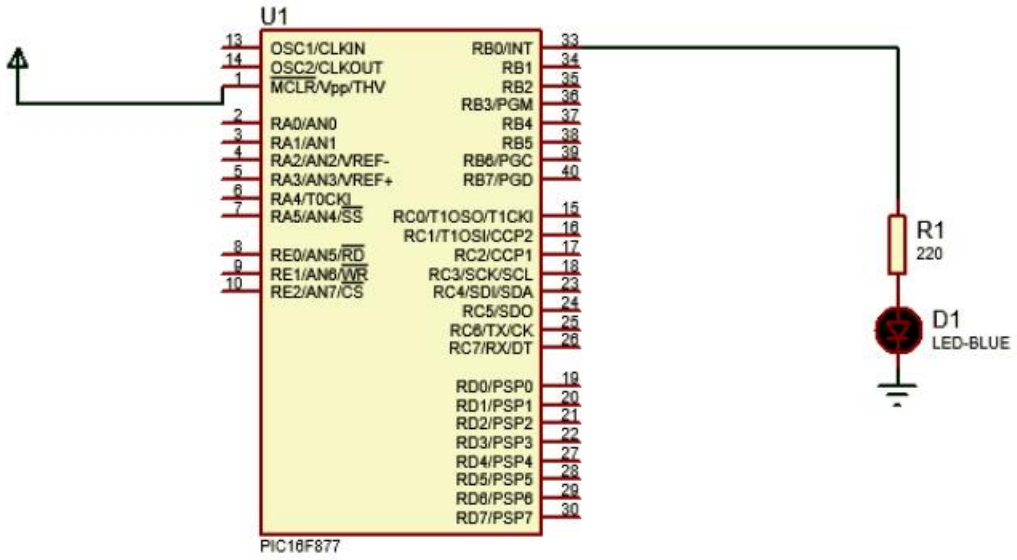

Figura 8.2 Circuito a implementar para la actividad 1

2.- Modificar el programa para que active y desactive todos los bits del puerto B.

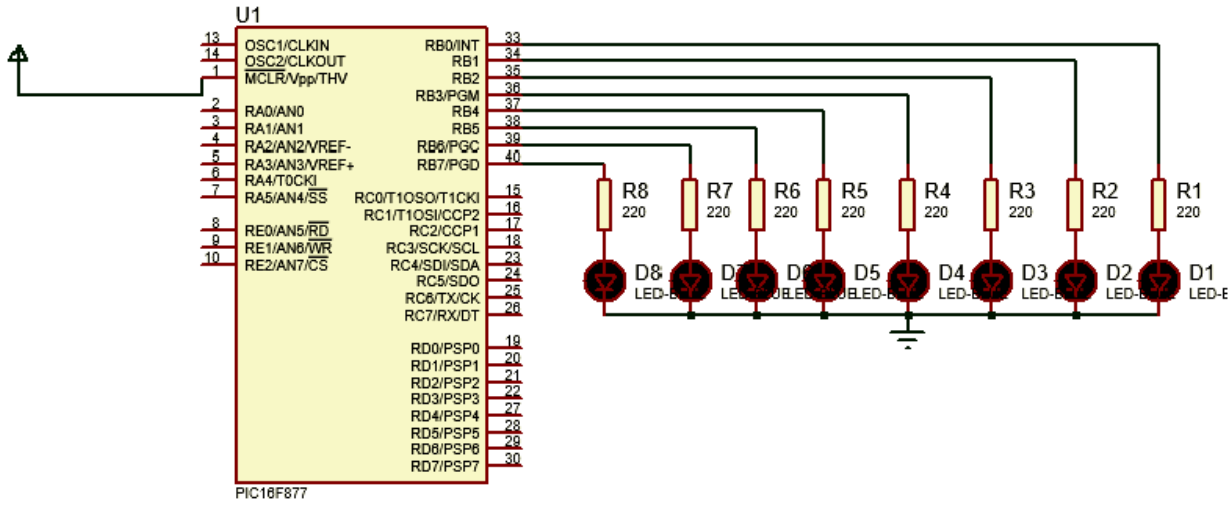

Figura 8.3 Circuito a implementar para la actividad 2

EJ1.c EJ2.C

```c
#include <16f877.h> //biblioteca del micro
#fuses HS,NOPROTECT, //parámetros físicos - eléctricos del controlador
#use delay(clock=20000000) // 20 MHz establece el reloj a utilizar
// reserva la memoria donde esta el bootloader
#org 0x1F00, 0x1FFF void loader16F877(void) {}
// El código prende y apaga 8 bits de la salida portb cada segundo.
void main(){
   while(1){
      output_b(0xff); // '11111111' salida para portb
      delay_ms(1000); // retardo de 1s
      output_b(0x00); // '00000000' salida para portb
      delay_ms(1000); // retardo de 1s
   } //while
}//main

```

3.- Escribir, comentar, compilar el siguiente programa usando el ambiente del PIC C Compiler y comprobar el funcionamiento.

EJ1.c | EJ2.C | EJ3.C

```c
#include <16f877.h> //biblioteca del micro
#fuses HS,NOPROTECT, //parámetros físicos - eléctricos del controlador
#use delay(clock=20000000) // 20 MHz establece el reloj a utilizar
// reserva la memoria donde esta el bootloader
#org 0x1F00, 0x1FFF void loader16F877(void) {}

int var1; // variable global

// el programa asigna la entrada PORTA a la salida PORTB
void main(){
   while(1){
      var1=input_a(); //lee la entrada del puerto A y almacena en var1
      output_b(var1); //asigna var1 a salida del puerto B
   }//while
}//main
```

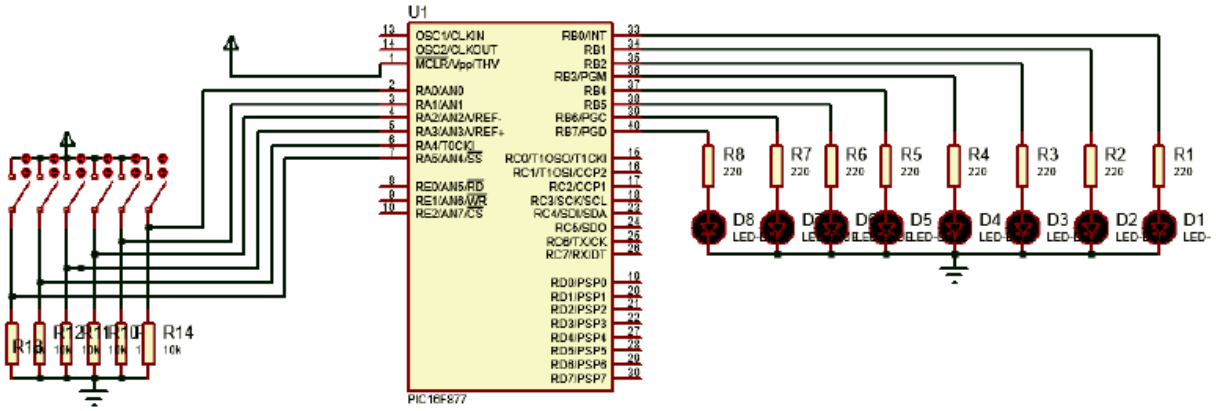

Figura 8.4 Circuito a implementar para la actividad 3

4.- Escribir, comentar, compilar, el siguiente programa usando el ambiente del PIC C Compiler y comprobar el funcionamiento.

EJ1.c | EJ2.C | EJ3.C | EJ4.c

```c
#include <16f877.h> //biblioteca del micro
#fuses HS,NOPROTECT, //parámetros físicos - eléctricos del controlador
#use delay(clock=20000000) // 20 MHz establece el reloj a utilizar
//utiliza estándar rs232 con la configuración:
//transmite por portC.6 y recibe por portC.7 con 9600 bauds
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
// reserva la memoria donde esta el bootloader
#org 0x1F00, 0x1FFF void loader16F877(void) {}
// el programa controla 8 bits de salida e imprime en terminal mensaje
void main(){
   while(1){
      output_b(0xff); // salida llena de 8 bits
      printf(" Todos los bits encendidos \n\r"); //imprime en consola
      delay_ms(1000); // retardo 1s
      output_b(0x00); // salida limpia de 8 bits
      printf(" Todos los leds apagados \n\r"); //imprime en consola
      delay_ms(1000); // retardo 1s
   }//while
}//main
```

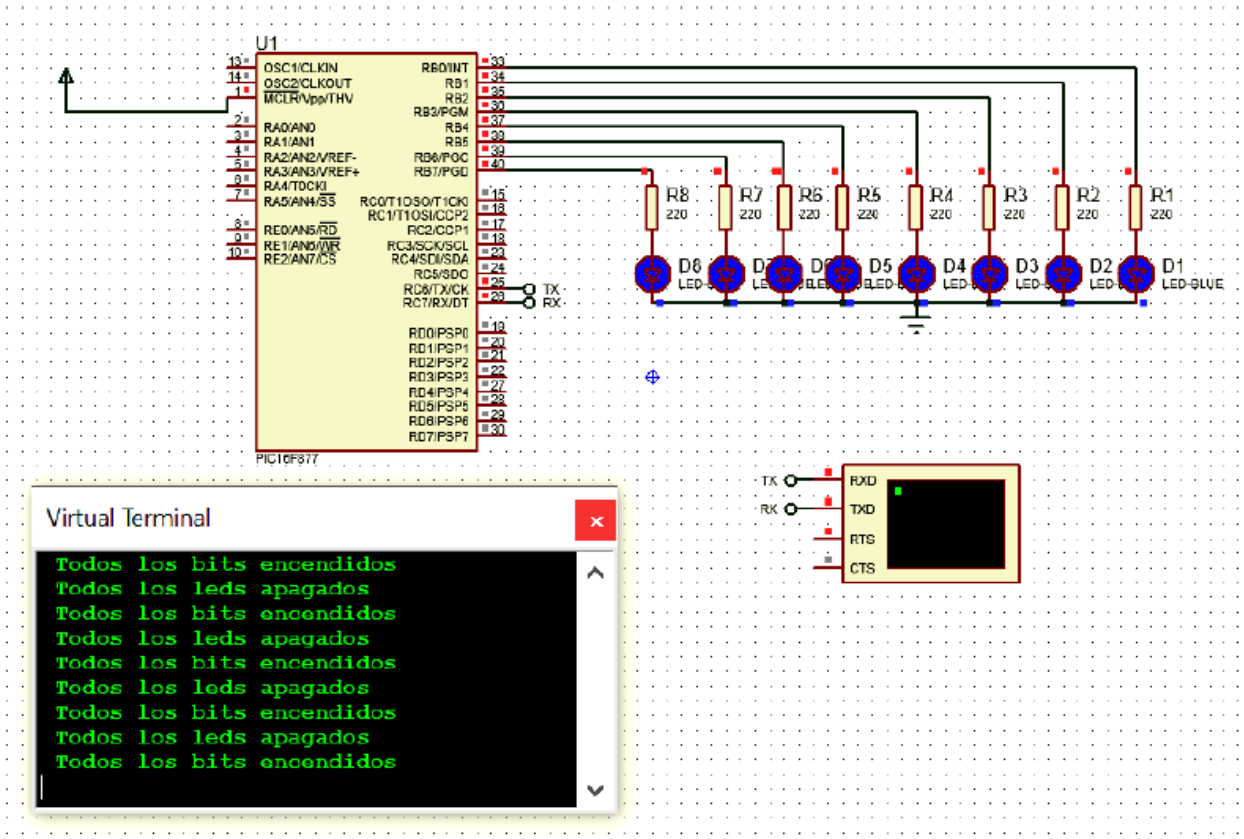

Figura 8.5 Circuito a implementar para la actividad 4

5.- Escribir, comentar, compilar, el siguiente programa usando el ambiente del PIC C Compiler y comprobar el funcionamiento.

Nota: La biblioteca lcd.c asigna las terminales para uso del LCD, en la plataforma usada se ha conectado al puerto D; también permite usar el puerto B para las señales de control; en este caso agregar previo a incluir la librería #define use_portb_lcd true.

EJ1.c | EJ2.C | EJ3.C | EJ4.c | EJ5.C

```
#include <16f877.h> //biblioteca del micro
#fuses HS,NOPROTECT, //parámetros físicos - eléctricos del controlador
#use delay(clock=20000000) // 20 MHz establece el reloj a utilizar
#define use_portd_lcd true // establece PORTD como conexión del display LCD
#include <lcd.c> // biblioteca para control de display LCD.
// el programa escribe la frase UNAM FI en display LCD
void main() {
lcd_init(); //inicialización del display
   while( TRUE ) {
      lcd_gotoxy(1,1); //posiciona cursor en (1,1) inicio del primero
      printf(lcd_putc," UNAM \n "); //escribe los chars en el LCD
      lcd_gotoxy(1,2); //posiciona cursor en (1,2) inicio del segundo
      printf(lcd_putc," FI \n "); //escribe los chars en el LCD
      delay_ms(300); //retardo de 300ms
   } //end while
}//end main
```

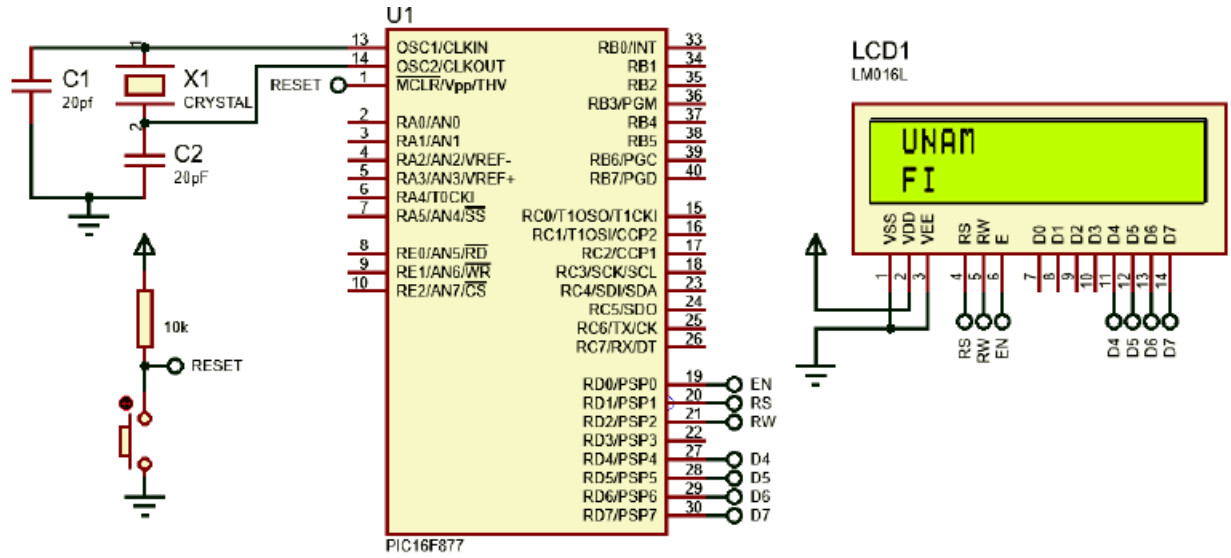

Figura 8.6 Circuito a implementar para la actividad 5

6.- Realizar un programa empleando el compilador de C, para ejecutar las acciones mostradas en la siguiente tabla, estas son controladas a través del puerto serie; usar retardos de ½ segundos.

DATO	ACCION Puerto B	*Ejecución*
0	Todos los bits apagados	*00000000*
1	Todos los bits encendidos	*11111111*
2	Corrimiento del bit más significativo hacia la derecha	*10000000* …….. *00000001*
3	Corrimiento del bit menos significativo hacia la izquierda	*00000001* …….. *10000000*
4	Corrimiento del bit más significativo hacia la derecha y a la izquierda	*10000000* …….. *00000001* …….. *10000000*
5	Apagar y encender todos los bits.	*00000000* *11111111*

Tabla 8.1 Control a través del puerto serie

EJ1.c | EJ2.C | EJ3.C | EJ4.c | EJ5.C | EJ6.c

```
1  #include <16f877.h> //biblioteca del micro
2  #fuses HS,NOPROTECT, //parámetros físicos - eléctricos del controlador
3  #use delay(clock=20000000) // 20 MHz establece el reloj a utilizar
4  // utiliza estándar rs232 con la configuración:
5  // transmite por portC.6 y recibe por portC.7 con 9600 bauds
6  #use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
7  // reserva la memoria donde esta el bootloader
8  #org 0x1F00, 0x1FFF void loader16F877(void) {}
9  char dato; // variable globar tipo caracter 8 bits
10
11 void correrDerecha(){ // corrimiento derecha
12    output_b(0x80);
13    delay_ms(500);
14    output_b(0x40);
15    delay_ms(500);
16    output_b(0x20);
17    delay_ms(500);
18    output_b(0x10);
19    delay_ms(500);
20    output_b(0x08);
21    delay_ms(500);
22    output_b(0x04);
23    delay_ms(500);
24    output_b(0x02);
25    delay_ms(500);
26    output_b(0x01);
27    delay_ms(500);
28 }
```

```
29  void correrIzquierda(){ // corrimiento izquierda
30      output_b(0x01);
31      delay_ms(500);
32      output_b(0x02);
33      delay_ms(500);
34      output_b(0x04);
35      delay_ms(500);
36      output_b(0x08);
37      delay_ms(500);
38      output_b(0x10);
39      delay_ms(500);
40      output_b(0x20);
41      delay_ms(500);
42      output_b(0x40);
43      delay_ms(500);
44      output_b(0x80);
45      delay_ms(500);
46  }
47  void correrMixto(){ // corrimiento mixto
48      correrDerecha(); // llama corrimiento derecha
49      correrIzquierda(); // llama corrimiento izquierda
50  }
51  // el programa realiza diversas acciones dependiendo de la entrada
52  void main(){
53      while(1){
54      scanf("%c", &dato); // recibe y almacena dato de entrada
55      switch(dato) { // evalua el caso y realiza la acción
56          case '0':
57          output_b(0x00); // todos los bits apagados
58          delay_ms(500);
59          break;
60          case '1':
61          output_b(0xff); // todos los bits encendidos
62          delay_ms(500);
63          break;
```

```
64          case '2':
65          correrDerecha(); // Corrimeinto derecha
66          output_b(0x00);
67          delay_ms(500);
68          break;
69          case '3':
70          correrIzquierda(); // Corrimiento izquierda
71          output_b(0x00);
72          delay_ms(500);
73          break;
74          case '4':
75          correrMixto(); // ida y vuelta
76          output_b(0x00);
77          delay_ms(500);
78          break;
79          case '5':
80          output_b(0xff); // apagar y encender
81          delay_ms(500);
82          output_b(0x00);
83          delay_ms(500);
84          break;
85          default:
86          break;
87       }// switch
88       }//while
89  }//main
90
```

7.- Realizar un programa que muestre en un Display de Cristal Líquido, la cantidad de veces que se ha presionado un interruptor, el cual esta conectado a la terminal A0.
El despliegue a mostrar es:
a. Primer línea y 5 columna; la cuenta en decimal
b. Segunda línea y 5 columna; la cuenta en hexadecimal

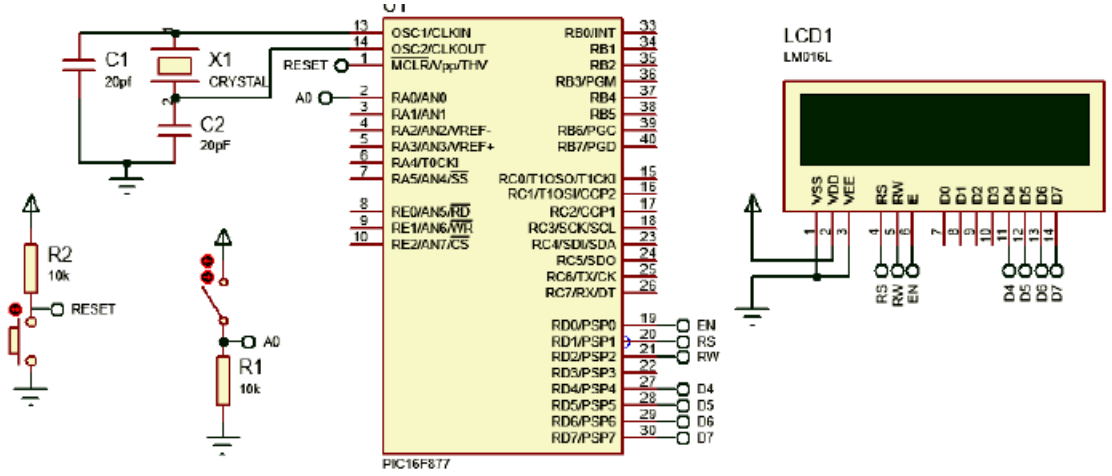

Figura 8.7 Circuito; actividad 7

EJ1.c | EJ2.C | EJ3.C | EJ4.c | EJ5.C | EJ6.c | EJ7.C

```c
#include <16F877.h> //biblioteca del micro
#fuses HS,NOWDT,NOPROTECT,NOLVP //parametros físicos-eléctricos controlador
#use delay(clock=20000000) // 20 MHz establece el reloj a utilizar
#define use_portd_lcd true // establece PORTD como conexión del display LCD
#include <lcd.c> //biblioteca para control de display LCD.

// Variables globales
int conteo; // lleva la cuenta de veces que se activa señal
int dato; // lectura de señal
int ante; // señal anterior (no contar varias veces una misma activación)

// el programa escribe la cuenta de activaciones en decimal y hexadecimal
// en el display LCD.
void main() {
   lcd_init(); //inicialización del display
   conteo=0;
   while( TRUE ) {
      ante=dato;
      dato=input_state(PIN_A0);
      if(dato==1 && dato!=ante){ // detecta solo cambio por subida
         conteo++;
      }
      lcd_gotoxy(5,1); //posiciona cursor en (5,1) inicio del primero
      printf(lcd_putc," %d\n", conteo); //escribe los chars en el LCD
      lcd_gotoxy(5,2); //posiciona cursor en (5,2) inicio del segundo
      printf(lcd_putc," %x\n", conteo); //escribe los chars en el LCD
      delay_ms(100); //retardo de 300ms
   } //end while
}//end main
```