Universidad Nacional Autónoma de México

Facultad de Ingeniería

Laboratorio de Microcomputadoras

Práctica No.8: Programación en C.

Puertos Paralelos E/S, Puerto Serie

Profesor: Rubén Anaya García

Alumnos:

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

Grupo: 4

Semestre: 2021-2

Práctica 08: Programación en C. Puertos Paralelos E/S, Puerto Serie

Objetivo

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

Desarrollo

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

Código

Ejercicio1_P8.c

```
#include <16f877.h>
#fuses HS,NOPROTECT,
#use delay(clock=20000000)
#org 0x1F00, 0x1FFF void loader16F877(void) {}
void main(){
   while(1){
      output_b(0x01);   //Salida de 0x01 por puerto B
      delay_ms(1000);   //Retardo de 1 seg
      output_b(0x00);   //Salida de 0x01 por puerto B
      delay_ms(1000);   //Retardo de 1 seg
   } //while
}//main
```

Código 1: Encendido y apagado de led en puerto B

Diagrama de Flujo

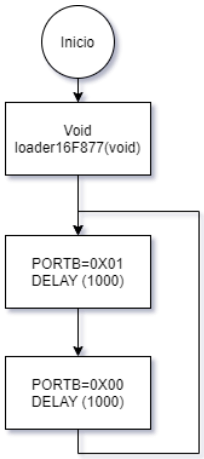
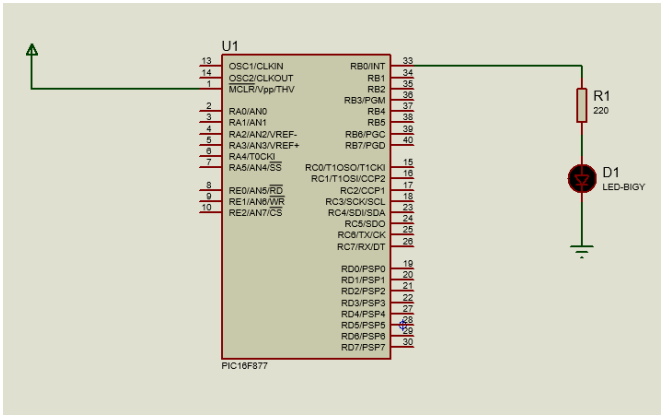
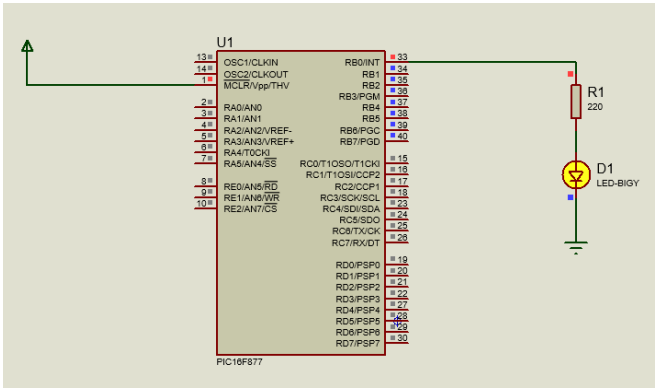


Diagrama 1: Ejercicio 01

Ejecución



Simulación 1: Estado Base de la simulación del código 01



Simulación 2: Encendido del bit

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

Código

Ejercicio2_P8.c

```
1  #include <16f877.h>
2  #fuses HS,NOPROTECT,
3  #use delay(clock=20000000)
4  #org 0x1F00, 0x1FFF void loader16F877(void) {}
5  void main(){
6     while(1){
7        output_b(0xFF);   //Salida de 0xFF por puerto B
8        delay_ms(1000);   //Retardo de 1 seg
9        output_b(0x00);   //Salida de 0x01 por puerto B
10       delay_ms(1000);   //Retardo de 1 seg
11    } //while
12 }//main
```

Código 2: Apagado y encendido de leds en puerto B

Diagrama de Flujo

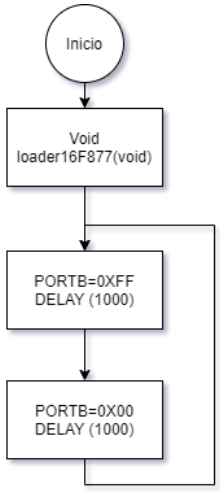
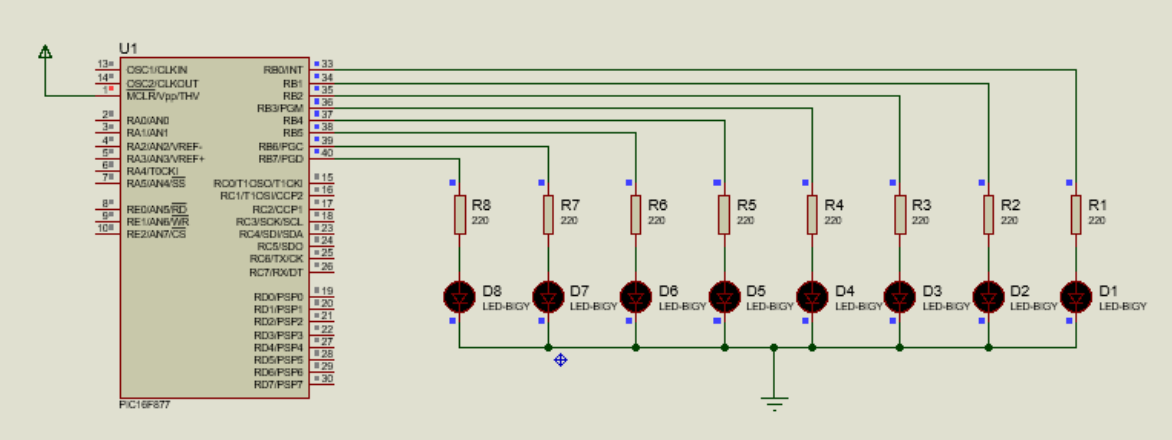
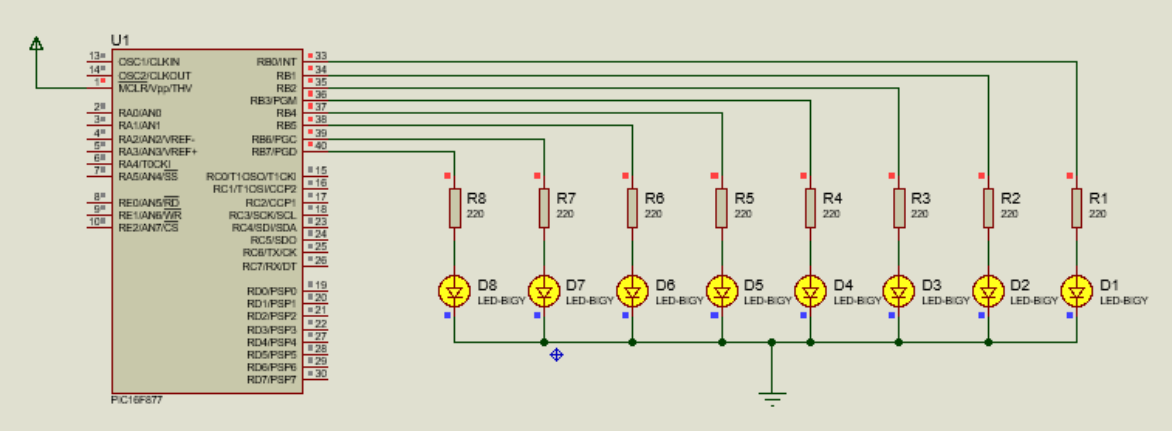


Diagrama 2: Ejercicio 02

Ejecución



Simulación 3: Estado base de la simulación del código 02



Simulación 4: Encendido de bits

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

Código

Ejercicio3_P8.c

```
1  #include <16f877.h>
2  #fuses HS,NOPROTECT,
3  #use delay(clock=20000000)
4  #org 0x1F00, 0x1FFF void loader16F877(void) {}
5  int var1;
6  void main(){
7     while(1){
8        var1=input_a();//Obtener la información de entrada y guardar en var1
9        output_b(var1);//Salida de var1
10    }//while
11 }//main
```

Código 3: Salida del puerto B de los datos de entrada del puerto a

Diagrama de Flujo

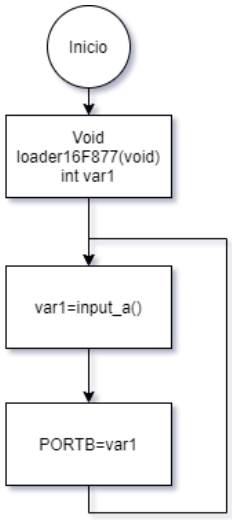
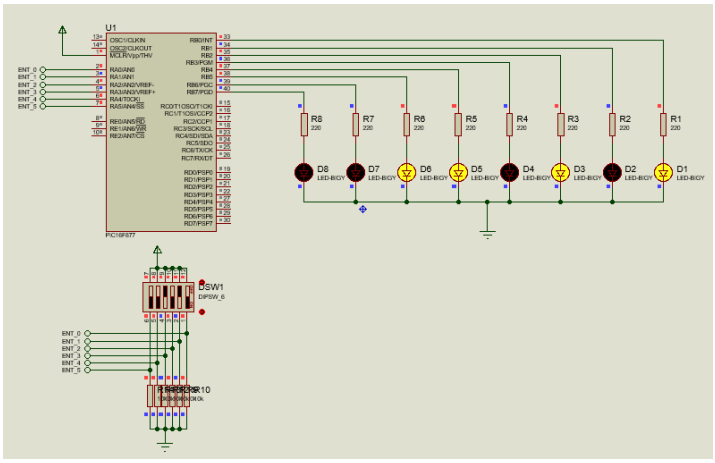
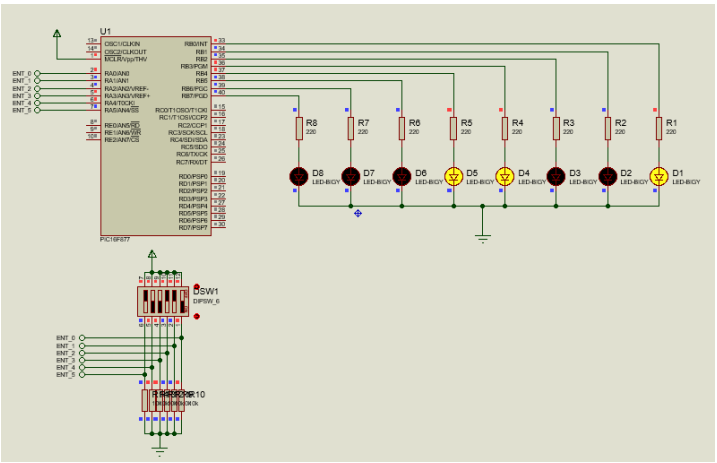


Diagrama 3: Código 3

Ejecución



Simulación 5: Ejemplo 1 del código 03



Simulación 6: Ejemplo 2 del código 03

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

Código

Ejercicio4_P8.c

```c
#include <16f877.h>
#fuses HS,NOPROTECT,
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
//Configuración de la comunicación serial con la terminal
#org 0x1F00, 0x1FFF void loader16F877(void) {}
void main(){
   while(1){
      output_b(0xff); //Salida por Puerto B
      printf(" Todos los bits encendidos \n\r");
      //Impresión de texto a la terminal
      delay_ms(1000);//Redardo
      output_b(0x00); //Salida por puerto B
      printf(" Todos los leds apagados \n\r");
      //Impresión de texto a la terminal
      delay_ms(1000);//Retardo
   }//while
}//main
```

Código 4: Encendido y apagado de leds con mensajes en terminal

Diagrama de Flujo

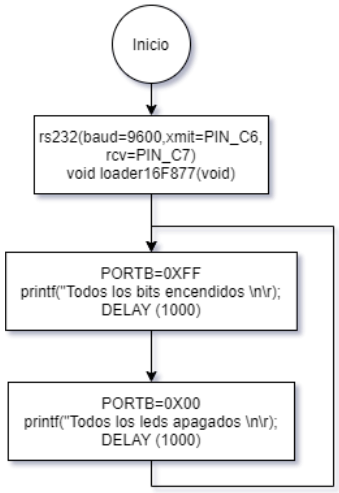
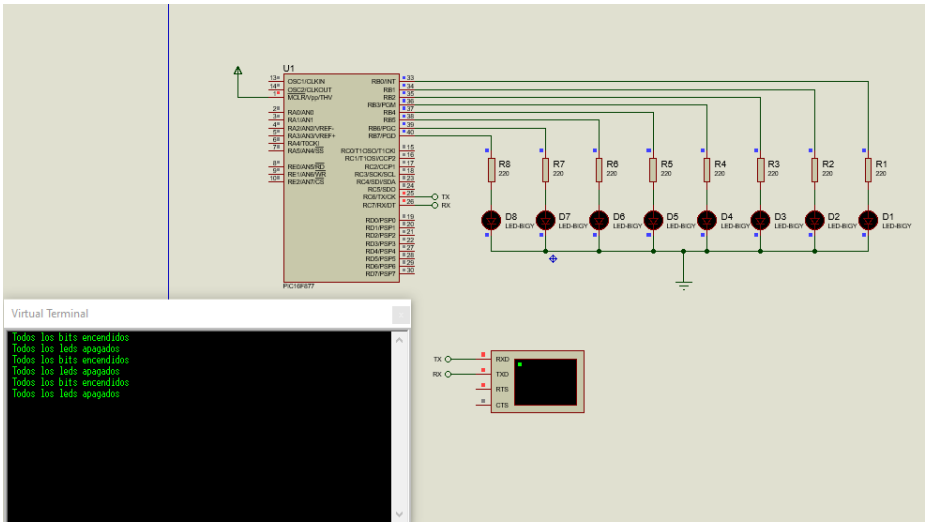
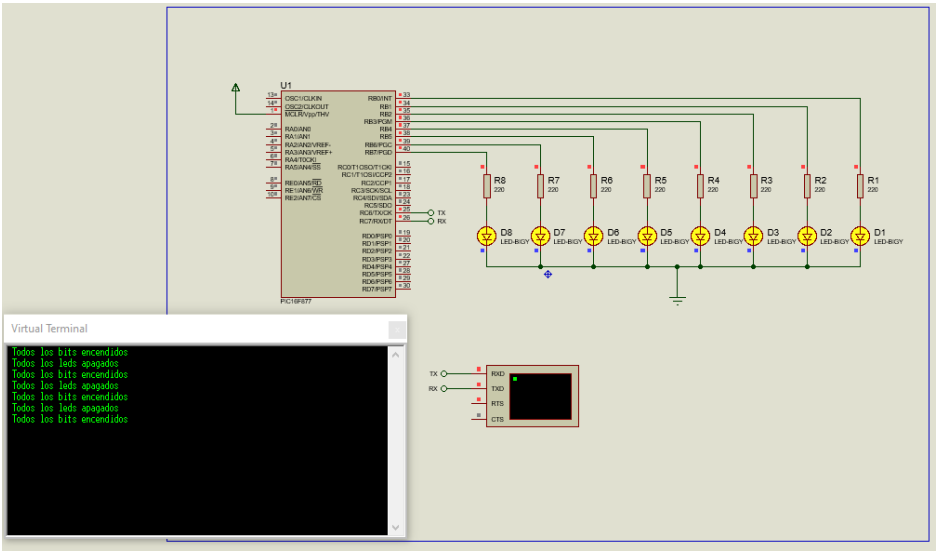


Diagrama 4: Código 04

Ejecución



Simulación 7: Ejecución del código 05



Simulación 8: Ejecución del código 05

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

Código

Ejercicio5_P8.c

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#define use_portb_lcd true //Definir puerto de uso para el LCD
#include <lcd.c>//Biblioteca para el manejo de LCD
void main() {
   lcd_init();//Inicialización LCD
   while( TRUE ) {
      lcd_gotoxy(1,1);//Posición LCD Renglon 1, Columna 1
      printf(lcd_putc," UNAM \n ");//Impresión de Mensaje
      lcd_gotoxy(1,2);//Posición LCD Renglon 2, Columna 1
      printf(lcd_putc," FI \n ");//Impresión de Mensaje
      delay_ms(300);//Retardo
   }
}
```

Código 5: Manejo de display LCD

Diagrama de Flujo

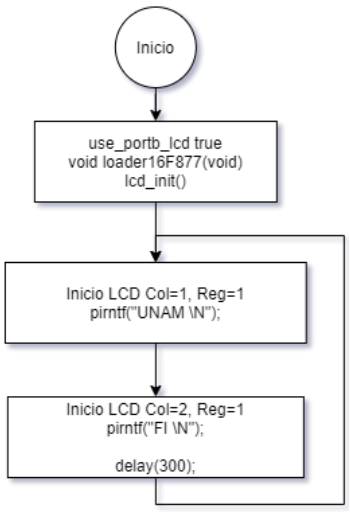
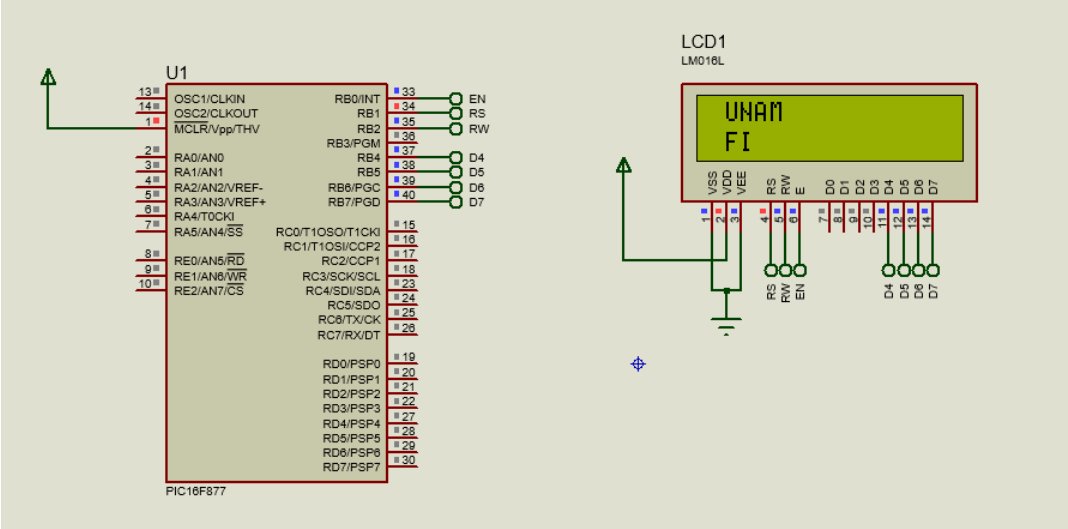


Diagrama 5: Código 05

Ejecución



Simulación 9: Impresión en LCD

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.

Código

Ejercicio6_P8.c

```
#include <16f877.h>   //Incluye la librería del microprocesador
#fuses HS,NOPROTECT,
#use delay(clock=20000000) //Frec. de oscilación 20Mhz
//Configura y activa el puerto SERIAL
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#org 0x1F00, 0x1FFF void loader16F877(void) {}
int cont;   //Contador
char opt;   //Caracter de opciones
int val;    //Información de salida
void main(){
   while(1){
      opt=getchar();//Obtener caracter de la terminal
      switch(opt){
         case '0':                  //Apagador de los bits
            printf("0) Todos los bits apagados\n\r");
            output_b(0x00);
         break;
         case '1':                  //Encender los bits
            printf("1) Todos los bits encendidos\n\r");
            output_b(0xFF);
         break;
         case '2':                 //Caso del corrimiento a la derecha
            printf("2) Corrimiento a la derecha\n\r");
            val=0x80;
            for(cont=0x00; cont<0x08; cont++)      //Ciclo del corrimiento
            {
               output_b(val);
               rotate_right(&val,1);
               delay_ms(500);
            }
         break;
         case '3':                  //Caso del corrimiento a la izquerda
            printf("3) Corrimiento a la izquierda\n\r");
            val=0x01;
            for(cont=0x00; cont<0x08; cont++)      //Ciclo para el corrimiento
            {
               output_b(val);
               rotate_left(&val,1);
               delay_ms(500);
            }
         break;
         case '4':                                 //Corrimiento de ambos lados
            printf("4) Derecha a Izquierda\n\r");
            val=0x80;
            for(cont=0x00; cont<0x07; cont++)      //Corr. a la derecha
            {
               output_b(val);
               rotate_right(&val,1);
               delay_ms(500);
            }
            for(cont=0x00; cont<0x08; cont++)      //Corr. a la izqu
            {
               output_b(val);
               rotate_left(&val,1);
               delay_ms(500);
            }
         break;
         case '5':
            printf("5) Encendido y apagado\n\r");
            output_b(val);
            output_b(0x00);                       //Apagado
            delay_ms(500);
            output_b(0xff);                        //Encendido
            delay_ms(500);
         break;
         default:
            printf("Dato fuera del rango\n\r");    //Si se ingresa un caracter
            //diferente, muestra el rango
         break;
      }
   }
}
```

Código 6: Ejercicio 6 de salidas diferentes por puerto b

Diagrama de Flujo

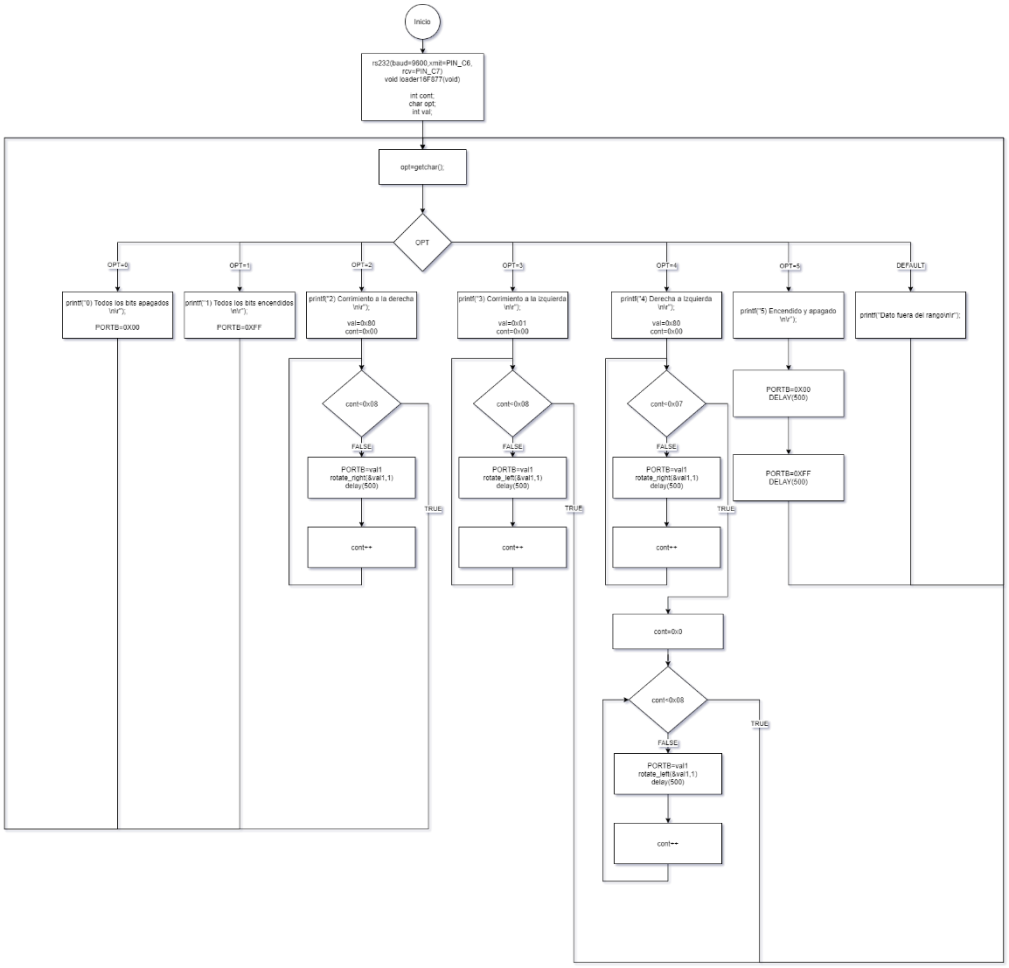
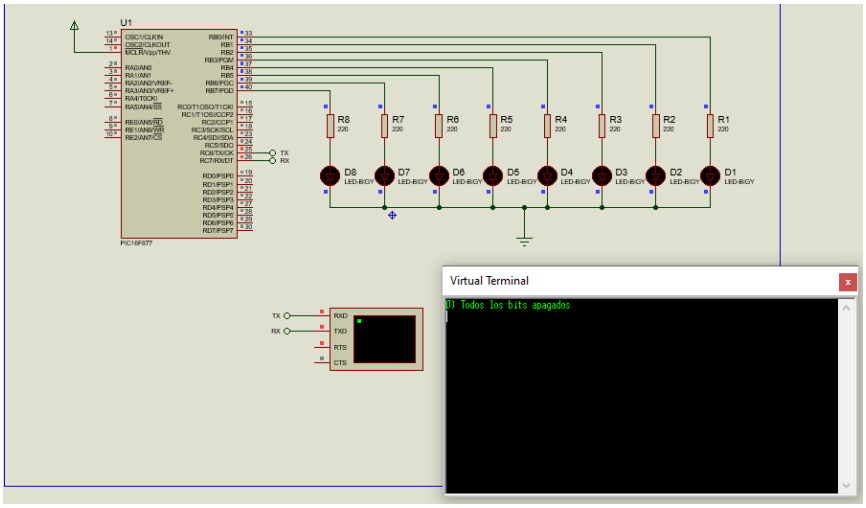
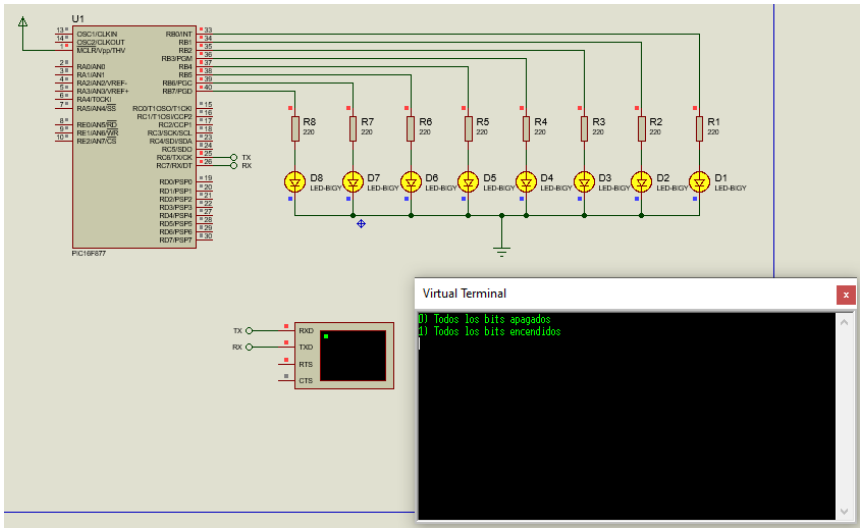


Diagrama 6: Código 06

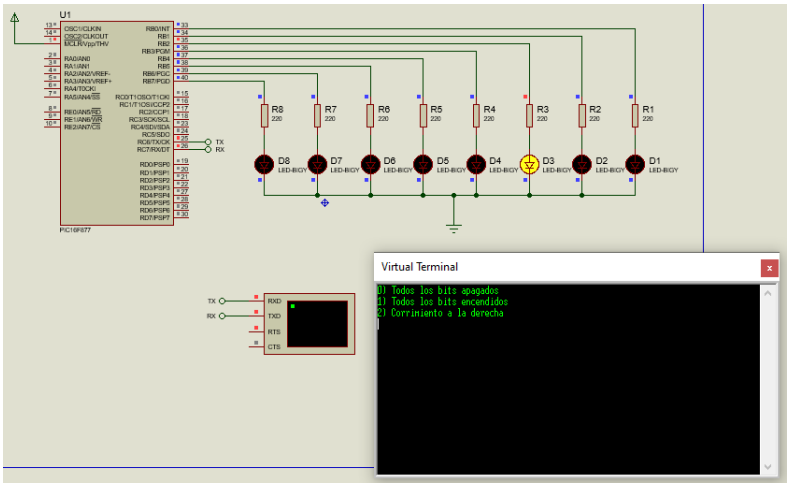
Ejecución



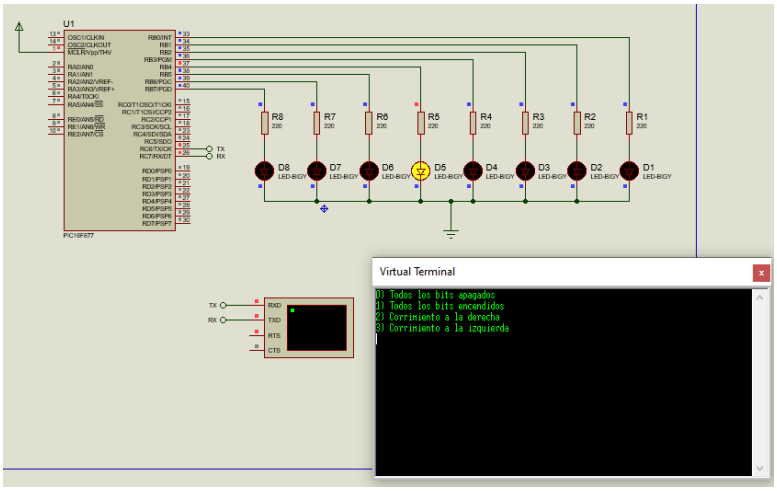
Simulación 10: Caso 0 código 06



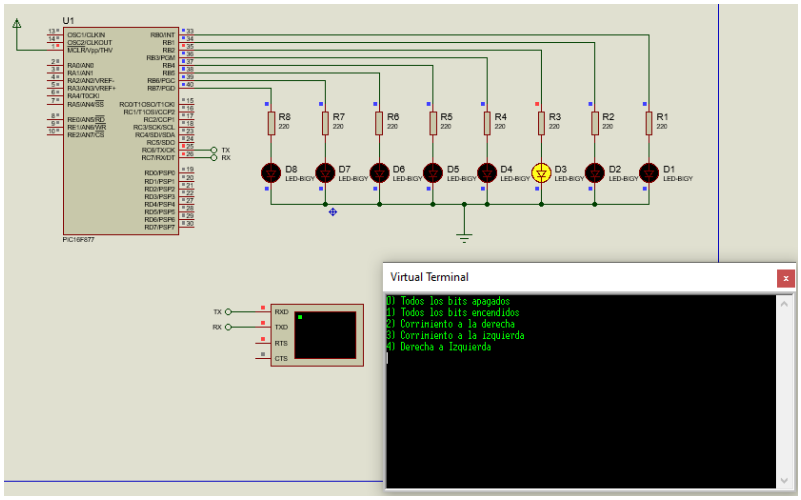
Simulación 11: Caso 1 código 06



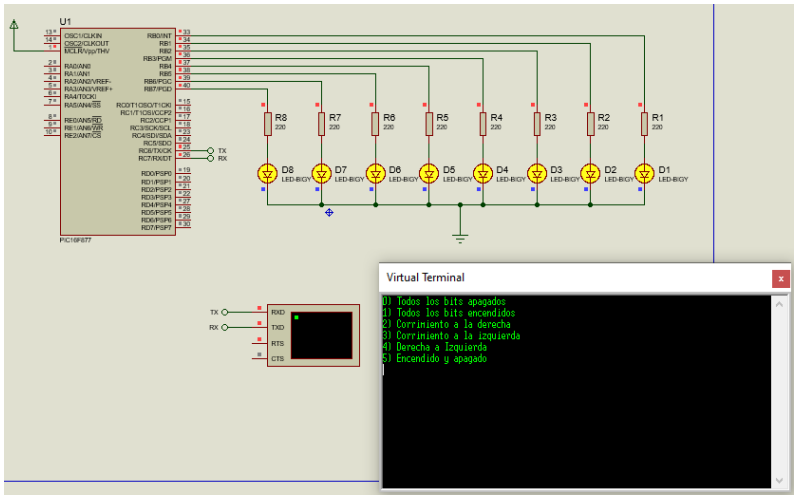
Simulación 12: Caso 2 código 06



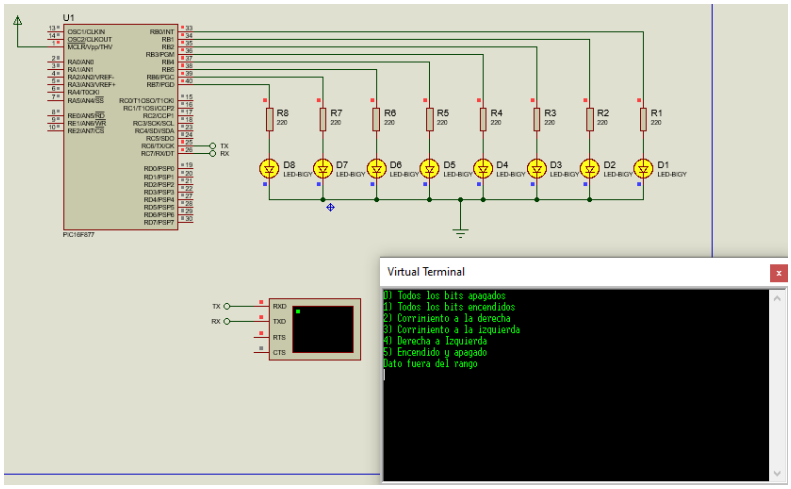
Simulación 13: Caso 3 código 06



Simulación 14: Caso 4 código 06



Simulación 15: Caso 5 código 06



Simulación 16: Caso por defecto código 06

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

Código

Ejercicio7_P8.c

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#define use_portb_lcd true
#include <lcd.c>//Manejo del LCD

#org 0x1F00, 0x1FFF void loader16F877(void) {}

long push_button=0;
void main() {
   lcd_init();//Inicialización LCD
   set_tris_a(0xFF);//A como entrada
   while(1) {
      if(input(PIN_A0)==0){//Caso del boton oprimido
         push_button++;    //Sumar 1 a las veces presionado

      }
      lcd_gotoxy(5,1);//Posición LCD Renglon 1, Columna 1
      printf(lcd_putc,"%04ld\n",push_button);//Impresión de Mensaje
      lcd_gotoxy(5,2);//Posición LCD Renglon 2, Columna 1
      printf(lcd_putc,"%04lX\n",push_button);//Impresión de Mensaje
      delay_ms(300);//Retardo
   }
}
```

Código 7: Impresion de conteo en decimal y hexadecimal

Diagrama de Flujo

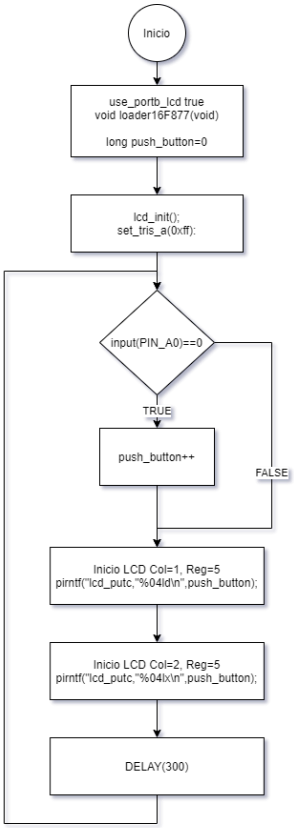
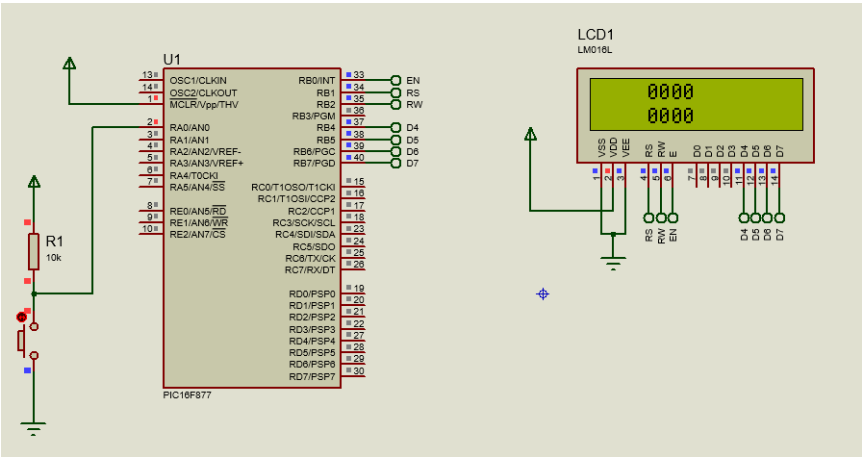
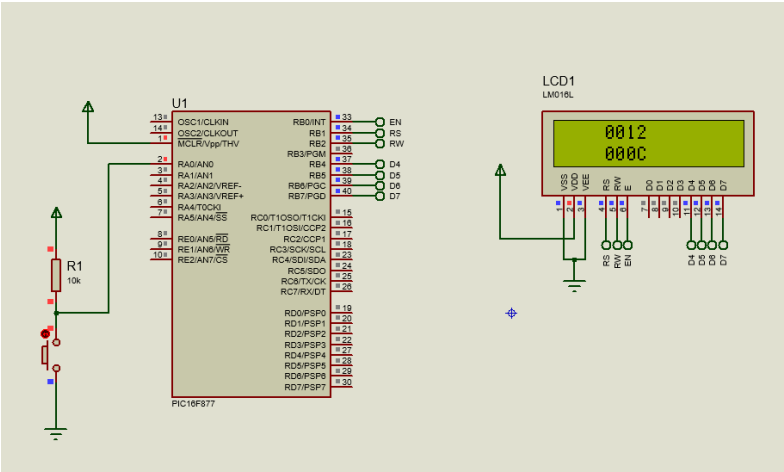


Diagrama 7: Código 07

Ejecución



Simulación 17: Caso base, código 07



Simulación 18: Conteo e impresión en display LCD

Conclusiones

Murrieta Villegas Alfonso

Uno de los conceptos más importantes en Computación es el denominado "Capas de Abstracción", pues estas son la forma en la que podemos observar en qué nivel estamos haciendo algo en la computadora, puede ser desde algo a nivel de hardware (Bajo nivel) mediante ensamblador o directamente trabajar con señales o en dado caso aspectos de alto nivel como software de aplicación como puede ser un editor de fotos.

Precisamente a lo largo de las primeras 7 prácticas aprendimos a trabajar a bajo nivel con lenguaje ensamblador para conocer como optimizar, interactuar y sobre todo trabajar con elementos o componentes concretos. En la presente práctica, aplicamos los mismos conceptos, pero empleando un lenguaje de programación de más alto nivel.

La reflexión es sobre todo que si bien las cosas se vuelven más fáciles de realizar también debemos considerar que para llegar a emplear estas herramientas o estar en esta capa de abstracción en algún momento se trabajó en una capa inferior.

Reza Chavarria Sergio Gabriel

A partir de los conceptos aprendidos y del manejo del lenguaje ensamblador en las prácticas anteriores implica el hecho de conocer a profundidad el fruncimiento de las instrucciones, registros y manejo de los componentes internos de los microcontroladores.

Con el manejo de un lenguaje alto nivel, como lo es C, apoya a la facilidad en la creación de proyectos teniendo el mismo resultado en comparación del manejo del programa en lenguaje ensamblador.

Valdespino Mendieta Joaquin

En la presente practica pudimos analizar y aplicar conceptos respecto al lenguaje de bajo nivel, en este caso ensamblador, profundizando su aplicación, a traes del manejo de entidades propias del microcontrolador, memoria o registros, operaciones, logica interna,

etc., pero visto desde otro nivel (uno superior), aplicándolo a un lenguaje de alto nivel como es el lenguaje C, comúnmente usado en arduinos por dar un ejemplo. Por otro lado, debido al feedback que tenemos respecto al lenguaje, tuvimos la facilidad de programación de estas entidades y la lógica para realizar una determinada tarea o tareas, presentadas en estos ejercicios.