Universidad Nacional Autónoma de México

Facultad de Ingeniería

Semestre 2023-2

Laboratorio de Microcomputadoras

Práctica 8:

Programación en C

Puertos Paralelos E/S, Puerto Serie

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: 24 de mayo de 2023

Fecha de entrega: 30 de mayo de 2023

Desarrollo

Para esta práctica se utilizará el lenguaje de programación C para programar el microcontrolador PIC16F877A. Para esta práctica se utilizó el programa de PIC C Compiler. Esta aplicación cuenta con un IDE con todas las facilidades para programar con las bibliotecas de este microcontrolador y generar el archivo .hex a partir del código fuente. Además se tiene una sección de ayuda donde se enlista una documentación sobre las diferentes funciones que se pueden utilizar para programar el microcontrolador utilizando C. Esta herramienta provee una alternativa poderosa a programar en ensamblador ya que es un lenguaje de un nivel más alto que ensamblador. Los programas que se desarrollaron sirvieron para probar los puertos de entrada y salida y el puerto en serie.

Los programas desarrollados fueron:

1. Probar y comentar un código proporcionado por la práctica.
2. Generar un código que prenda y apague los LEDs del puerto B.
3. Probar y comentar un código que utiliza como entrada el puerto A.
4. Probar y comentar un código que utiliza la recepción de datos.
5. Probar y comentar un código que utiliza el display de cristal líquido.
6. Un programa que utiliza el puerto en serie para encender LEDs de cierta manera a partir de una variable de control.
7. Un programa que muestre en un display de cristal líquido el número de veces que se ha presionado un botón.

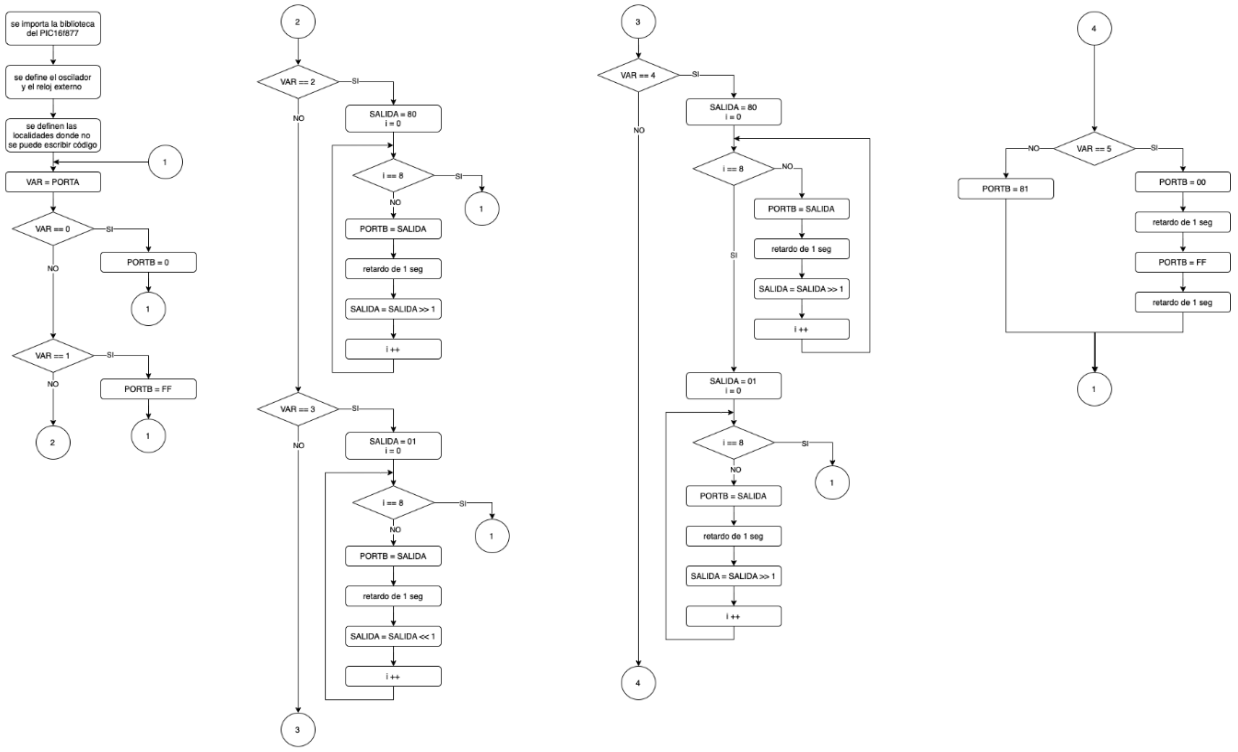
Algoritmos

Ejercicio 2

Para el ejercicio 2 se tomó como base el código proporcionado por el ejercicio 1. Lo único que se modificó fueron los valores asignados al Puerto B. De manera que primero se le asigna un 00 para apagar todos los leds y después del retardo de 1 segundo, se le asigna FF para prender todos los leds. Esto se muestra en la carta ASM.

Ejercicio 6

Para el ejercicio 6 se realizó la siguiente carta ASM:



Primero es necesario importar la biblioteca del pic, definir el reloj externo y la sección de memoria donde no se puede escribir código. Después se guarda la entrada del Puerto A en una variable que se ingresa a un switch para comparar su valor y realizar distintas acciones en el Puerto B. En caso de ser 0, el Puerto B también se vuelve cero. En caso de que la variable sea 1, el Puerto B se vuelve FF para encenderlo todo.

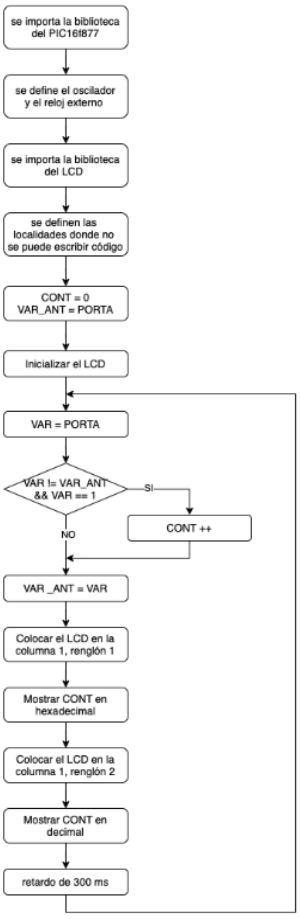
En caso de que la variable sea 2, se le asigna el valor 80 a una variable ‘SALIDA’ y mientras i no sea 8, al Puerto B se le asigna el valor de ‘SALIDA’ , se llama un retardo, se hace un corrimiento a la derecha de bits en la variable ‘SALIDA’ y se incrementa i. De forma que se genera un corrimiento del led encendido en el Puerto B a la derecha. En caso de que la variable sea 3, se hace lo mismo con la diferencia de inicializar ‘SALIDA’ con 01 y hacer el corrimiento a la izquierda.

Para el caso en el que la variable es 4, se hace el procedimiento en conjunto de la variable igual a 2 e igual a 3. De manera que se ve como un corrimiento a la derecha seguido por un corrimiento a la izquierda. Por último, cuando la variable es 5, se le asigna 00 al Puerto B para ver los leds apagados, se realiza un retardo de 1 segundo, se le asigna FF al Puerto B para ver los leds encendidos y se vuelve realizar un retardo de 1 segundo.

En caso de que la variable no coincida con ningún valor del 0 al 5, se le asigna 81 al Puerto B para ver el primer y último led encendidos. Después de realizar la respectiva acción de cada caso, vuelve a obtener el valor en el Puerto A para realizar otra acción.

Ejercicio 7

Para el último ejercicio se realizó la siguiente carta ASM:



De igual forma, primero se importa la biblioteca del microcontrolador, se define el reloj externo y el oscilador, se importa la biblioteca que maneja el LCD y se indican las localidades donde no se puede escribir código. Después se inicializa un contador en 0 y una variable ‘VAR_ANT’ con el valor del Puerto A. Lo siguiente es inicializar el LCD.

Después se guarda el valor del Puerto A en otra variable ‘VAR’ de manera que se compara si VAR_ANT y VAR son diferentes para saber si se movió el Switch. Debido a que sólamente se deben contar las veces que se prende el Switch y no que se apaga, también se verifica si la VAR (el valor actual del Puerto A) es igual a 1. De cumplir con ambas condiciones, se incrementa en 1 el contador. Después, haya cumplido o no la condición, se actualiza el valor de VAR_ANT asignándole el de VAR. Por último, se ubica el LCD en el renglón 1 y columna 1 para mostrar el valor del contador en hexadecimal y luego se ubica

en el renglón 2 y columna 1 para ahora mostrarlo en decimal. Se realiza un retardo de 300 ms y se repite el proceso desde que se le asigna a VAR el valor del Puerto A.

Programas comentados

Ejercicio 1

```
1   #include <16f877.h>          //Biblioteca para PIC16F877
2   #fuses HS,NOPROTECT,
3   #use delay(clock=20000000)  //Configurando el reloj
4   #org 0x1F00, 0x1FFF void loader16F877(void) {}
5
6   void main(){
7       while(1){
8           output_b(0x01); //Puerto B = 0000 0001
9           delay_ms(1000); //Retardo de 1000ms
10          output_b(0x00); //Puerto B = 0000 0000
11          delay_ms(1000); //Retardo de 1000ms
12      } //while
13  }//main
```

Ejercicio 2

```
1   #include <16f877.h>          //Biblioteca para PIC16F877
2   #fuses HS,NOPROTECT,
3   #use delay(clock=20000000)  //Configurando el reloj
4   #org 0x1F00, 0x1FFF void loader16F877(void) {}877(void) {}
5
6   void main(){
7       while(1){
8           output_b(0xFF); //Puerto B = 1111 1111
9           delay_ms(1000); //Retardo de 1000ms
10          output_b(0x00); //Puerto B = 0000 0000
11          delay_ms(1000); //Retardo de 1000ms
12      } //while
13  }//main
```

Ejercicio 3

```
#include <16f877.h>          //Biblioteca para PIC16F877
#fuses HS,NOPROTECT,
#use delay(clock=20000000)  //Configurando el reloj
#org 0x1F00, 0x1FFF void loader16F877(void) {}

int var1;                //Variable para guardar la entrada

void main(){
    while(1){
        var1=input_a(); //var1 = Puerto
        output_b(var1); //Puerto B = var1
    }//while
}//main
```

Ejercicio 4

```
#include <16f877.h>                                 //Biblioteca de PIC16F877A
#fuses HS,NOPROTECT,
#use delay(clock=20000000)                          //Configurando el reloj
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)      //Configurando Baud y recepción
#org 0x1F00, 0x1FFF void loader16F877(void) {}

void main(){
    while(1){
        output_b(0xff);                             //Puerto B = 1111 1111
        printf(" Todos los bits encendidos \n\r");  //Impresión en terminal
        delay_ms(1000);                             //Retardo de 1000ms
        output_b(0x00);                             //Puerto B = 0000 0000
        printf(" Todos los leds apagados \n\r");    //Impresión en terminal
        delay_ms(1000);                             //Retardo de 1000ms
    }//while
}//main
```

Ejercicio 5

```
#include <16F877.h>                     //Biblioteca PIC16F877A
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)              //Configurando el reloj
#include <lcd.c>                        //Biblioteca de LCD
#org 0x1F00, 0x1FFF void loader16F877(void) {}

void main() {
    lcd_init();                         //Incializa el display de crisital líquido
    while( TRUE ) {
        lcd_gotoxy(1,1);                //Colocarse en columna 1, renglón 1
        printf(lcd_putc," UNAM \n ");   //Imprimir en LCD
        lcd_gotoxy(1,2);                //Colocarse en columna 1, renglón 2
        printf(lcd_putc," FI \n ");     //Imprimir en LCD
        delay_ms(300);                  //Retardo de 300ms
    }
}
```

Ejercicio 6

```c
#include <16F877.h>                    //Biblioteca PIC16F877A
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)             //Configurando el reloj
#org 0x1F00, 0x1FFF void loader16F877(void) {}

/*
var1: variable de entrada
salida: variable para rotar los bits
i: contador
*/
int var1, salida, i;

void main(){
    while(1){
        var1=input_a();                //Guardar entrada de Puerto A en var1
        switch(var1){                  //Switch con var1
        case 0:                        //Cuando var1 es 0
            output_b(0x00);            //Puerto B = 0000 0000
            break;
        case 1:                        //Cuando var1 es 1
            output_b(0xFF);            //Puerto B = 1111 1111
            break;
        case 2:                        //Cuando var1 es 2
            salida = 0x80;             //salida = 1000 0000
            for(i=0; i<8; i++){        //8 iteraciones
                output_b(salida);      //Puerto B = 1000 0000
                delay_ms(1000);        //Retardo de 1000ms
                salida = salida >> 1;  //Rotar a la derecha
            }
            break;
        case 3:                        //Cuando var1 es 3
            salida = 0x01;             //salida = 0000 0001
            for(i=0; i<8; i++){        //8 iteraciones
                output_b(salida);      //Puerto B = 0000 0001
                delay_ms(1000);        //Retardo de 1000ms
                salida = salida << 1;  //Rotar a la izquierda
            }
            break;
        case 4:                        //Cuando var1 es 4
            salida = 0x80;             //salida = 1000 0000
            for(i=0; i<8; i++){        //8 iteraciones
                output_b(salida);      //Puerto B = 1000 0000
                delay_ms(1000);        //Retardo de 1000ms
                salida = salida >> 1;  //Rotar a la derecha
            }
            salida = 0x01;             //salida = 0000 0001
            for(i=0; i<8; i++){        //8 iteraciones
                output_b(salida);      //Puerto B = 0000 0001
                delay_ms(1000);        //Retardo de 1000ms
                salida = salida << 1;  //Rotar a la izquierda
            }
            break;
        case 5:                        //Cuando var1 es 5
            output_b(0xFF);            //Puerto B = 1111 1111
            delay_ms(1000);            //Retardo de 1000ms
            output_b(0x00);            //Puerto B = 0000 0000
            delay_ms(1000);            //Retardo de 1000ms
            break;
        default:                       //Cuando se tiene otro valor
            output_b(0x81);            //Puerto B = 1000 0001
        }
    }//while
}//main
```

Ejercicio 7

```c
#include <16F877.h>                    //Biblioteca PIC16F877A
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)             //Configurando el reloj
#include <lcd.c>                       //Biblioteca LCD
#org 0x1F00, 0x1FFF void loader16F877(void) {}

int var, var_ant, cont;
/*
var: Nueva entrada
var_ant: Entrada anterior
cont: Contador de cambios
*/

void main(){
   cont = 0;                           //Inicializar conteo en 0
   var_ant = input_a();                //Entrada = Puerto A

   lcd_init();                         //Inicializar LCD
   while(1){
      var = input_a();                 //Nueva entrada = Puerto A
      if (var != var_ant && var == 1){ //Si son entradas diferentes y es 1
         cont ++;                      //Se aumenta la cuenta
         }
      var_ant = var;                   //Se actualiza la variable anterior

      lcd_gotoxy(1,1);                 //Colocarse en columna 1, renglón 1
      printf(lcd_putc,"%X",cont);      //Imprimir cuenta en hexadecimal
      lcd_gotoxy(1,2);                 //Colocarse en columna 1, renglón 2
      printf(lcd_putc,"%u",cont);      //Imprimir cuenta en decimal
      delay_ms(300);                   //Retardo de 300ms

   }//while
}//main
```

Conclusiones y/o comentarios

Zepeda Baeza Jessica:

En esta práctica se aprendió a manejar el microcontrolador utilizando el lenguaje de programación C y un nuevo entorno de desarrollo: PIC C Compiler. En primer lugar se aprendió cómo manejar el IDE e identificar sus diferentes ventanas así como también la forma de crear nuevos proyectos y compilarlos y de buscar las funciones que se pueden implementar. Después se observaron las bibliotecas e instrucciones que son necesarias para manejar el PIC16F877 y hacer sus configuraciones iniciales como el oscilador, el reloj externo, los puertos de transmisión y recepción, el baud rate, etc. Por último se realizaron ejercicios sencillos que ya se habían implementado anteriormente pero en lenguaje ensamblador como el corrimiento de bits, apagar y prender un bit, mostrar un dato en el LCD, utilizar la terminal de la computadora para transmitir un dato, etc. Se pudo entender que es mucho más sencillo en términos de código utilizar el lenguaje C debido a que existen funciones que ya implementan todo un procedimiento como lcd_init() (inicializa el LCD),

output_b() (manda un dato como salida), delay_ms() (hace retardos del tiempo indicado), printf() (transmite un dato ya sea al LCD o a la terminal), etc. La desventaja de esto es que probablemente se utiliza mucho más código y que hay configuraciones que ya no quedan al alcance del programador.

Barreiro Valdez Alejandro:

En esta práctica se aprendieron a utilizar funciones básicas para el uso de la biblioteca del PIC16F877A en el lenguaje de programación de C. Se comprendió cómo se puede utilizar el IDE que viene con C Compiler y cómo se compila en esta aplicación. Entre las funciones más importantes que se aprendieron están las de configurar los valores de los puertos de salida, configurar el tiempo de retardo, configurar la salida del display de cristal líquido y recibir datos a través de un puerto. Se observaron todas las bibliotecas que son necesarias para cada una de las acciones que se realizan. Cada uno tiene su manera de trabajar como la del LCD que requiere una inicialización de este componente. Por último, se pudo entender cuáles son las ventajas y cuáles son las desventajas de trabajar utilizando C. Entre las ventajas están que se tienen programas más fáciles de entender y con un flujo más convencional y reducido. Entre las desventajas están que no se puede controlar el código ensamblador que se genera y hay muchas configuraciones que no se pueden controlar. Con este programa se pudo entender cómo funciona C en el proceso de generar programar para el microcontrolador PIC16F877A.