UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

LABORATORIO DE MICROCOMPUTADORAS

PRÁCTICA 8:

PUERTIO SERIE Y PROGRAMACIÓN EN C

GRUPO: 12

PROFESORA: M.I LOURDES ANGÉLICA QUIÑONES JUAREZ

ALUMNO: CHÁVEZ DELGADO JORGE LUIS

N° DE CUENTA: 312217493

FECHA DE ASIGNACIÓN: *07/04/17*

FECHA DE ENTREGA: 18/*04/17*

PRÁCTICA 7: Puerto Serie SCI (ASÍNCRONO)

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

Ejercicio 1:

En este ejercicio aprendimos las configuraciones básicas en lenguaje C para programar el pic, en este caso fue encender un led como se puede ver en el código.

Código:

```
#include <16f877.h>
#fuses HS,NOPROTECT, //Indicamos que trabajaremos a alta frecuencia
#use delay(clock=20000000) //Frecuencia de oscilaciòn de acuerdo al cristal ensamblado
#org 0x1F00, 0x1FFF void loader16F877(void) {}
void main(){
//De aquì para arriba es la plantilla para C compiler
while(1){
      output_b(0x01); //En ensamblador debimos configurar los registros TRIS para usar el puerto B
      delay_ms(1000); //Enn ensamblador se debiò crear una rutina con el tiempo de cada instrucciòn
      output_b(0x00); //En ensamblador hubieramos mandado 0 al puerto B
      delay_ms(1000); //Retardo 1 seg
}//while
}//main
```

Ejercicio 2:

Para este ejercicio repetimos el anterior pero modificando la sentencia que envía el dato al puerto B para que en lugar de que fuese un led encendido, se encendieran los 8 leds y se apagaran con un retardo de 1 segundo.

Código:

```
#include <16f877.h>
#fuses HS,NOPROTECT, //Indicamos que trabajaremos a alta frecuencia
```

```
#use delay(clock=20000000) //Frecuencia de oscilaciòn de acuerdo al cristal ensamblado

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

void main(){

//De aquì para arriba es la plantilla para C compiler

while(1){

 output_b(0xFF); //En ensamblador debimos configurar los registros TRIS para usar el puerto B, SENTENCIA MODIFICADA

 delay_ms(1000); //Enn ensamblador se debiò crear una rutina con el tiempo de cada instrucciòn

 output_b(0x00); //En ensamblador hubieramos mandado 0 al puerto B

 delay_ms(1000); //Retardo 1 seg

}//while

}//main
```

Ejercicio 3:

En este ejercicio vimos la instrucción para leer el puerto A y mandar lo que se leyó en el mismo al puerto B.

Código:

```
#include <16f877.h>

#fuses HS,NOPROTECT,

#use delay(clock=20000000)

#org 0x1F00, 0x1FFF void loader16F877(void) {} //for the 8k 16F876/7

int var1; //Se inicializa una variable, en ensamblador seria var1  EQU y una direcciòn disponible

void main(){

while(1){  //En ensamblador crearíamos una subrutina loop y ahí las demás instrucciones

var1=input_a(); //En ensamblador hubieramos configurado el puerto A como entrada

output_b(var1); //Lo que se lee en el puerto A se manda directamente al puerto B, para usar el puerto A debimos configurarlo como entrada con TRISA

}//while
```

```
}//
```

Ejercicio 4:

Este ejercicio realiza lo mismo que el número 2, con la diferencia de que aquí mandamos un mensaje a la terminal por medio de comunicación asíncrona. Los mensajes enviados son: “Todos los bits apagados” cuando estos estan apagados, y “Todos los bits encendidos” cuando todos los leds estan prendidos.

Código:

```
#include <16f877.h>

#fuses HS,NOPROTECT,

#use delay(clock=20000000)

#use rs232(baud=38400, xmit=PIN_C6, rcv=PIN_C7) // (xmit=pinc_c6 ) = tx
, (rcv=PIN_C7)= Rx

#org 0x1F00, 0x1FFF void loader16F877(void) {} //for the 8k 16F876/7

void main(){

while(1){

 output_b(0xff); //En ensamblador hubieramos pasado a w un valor
hexadecimal h'ff' y después moverlo al puerto B previamente
configurado

 printf(" Todos los bits encendidos \n\r"); //No mandamos a terminal
mensajes pero sí enviamos a través de la comunicación asíncrona del pic

  delay_ms(1000); // Sacabamos el tiempo por instrucción y lo
multiplicabamos para obtener un retardo de cierto tiempo

 output_b(0x00);//En ensamblador hubieramos pasado a w un valor
hexadecimal h'00' y después moverlo al puerto B previamente
configurado

 printf(" Todos los leds apagados \n\r"); // No mandamos a terminal
mensajes pero sí enviamos a través de la comunicación asíncrona del pic

  delay_ms(1000); //Sacabamos el tiempo por instrucción y lo
multiplicabamos para obtener un retardo de cierto tiempo

}//while

}//main
```

Ejercicio 5:

Este ejercicio nos costó un poco más de trabajo, ya que debíamos aplicar lo que vimos en la práctica 3, que era hacer un seleccionador de casos y realizar las acciones correspondientes cuando se recibiera cierta opcion en el puerto A.

Código:

```
#include <16f877.h>

#fuses HS,NOPROTECT,

#use delay(clock=20000000)

#use rs232(baud=38400, xmit=PIN_C6, rcv=PIN_C7) // (xmit=pinc_c6 ) = tx
, (rcv=PIN_C7)= Rx

#org 0x1F00, 0x1FFF void loader16F877(void) {} //for the 8k 16F876/7

char opcion;

int i=0;

void main(){

while(1){  //Mandamos nuestro menú de opciones a la terminal

printf("Indica la opciòn: \n"); //Solicitamos la opción del usuario de
acuerdo a lo indicado

printf("0.-Apaga los leds\n");

printf("1.-Prende los leds\n");

printf("2.-Corrimiento a la derecha\n");

printf("3.-Corrimiento a la izquierda\n");

printf("4.-Corrimiento hacia ambos lados\n");

printf("5.-Prende y apaga todos los leds");

printf("salir");

opcion=getch(); //En ensamblador hubieramos configurado el puerto A
para recibir el dato de la opcion a realizar

putc(opcion);

switch(opcion){  // Usamos una estructura de c (SWITCH) la cual en
ensamblador se realizaría con un xorlw verificando la bandera Z

  case '0':

    output_b(0x00); //En ensamblador hubieramos mandado 0 al puerto B

    break;
```

```
  case '1':
    output_b(0xFF); //En ensamblador hubieramos mandado ff al puerto
B
    break;
  case '2': // Corrimiento a la derecha
    int valor =128;
    int aux=0;
    while(valor>0){
      output_b(valor); //Pasamos la variable al puerto
       delay_ms(1000); //En ensamblador se debiò crear una rutina con el
tiempo de cada instrucciòn
      aux=valor/2; dividimos
      valor=valor-aux; //Restamos el valor actual en decimal
      if(valor<1) // Validamos que haya llegado a uno
        break;
    }
    break;
  case '3': //Corrimiento a la izquierda
       int valor2 =1; // inializamos valor2, en ensamblador seria valor 2
equ h'XX' ->Direccion
    int aux2=0; // Inicializamos auxiliar
     while(valor2<=128){  //Ciclo hasta que el valor mas significativo de
os leds
      output_b(valor2); //Scanos al pueto b lo de valor2
       delay_ms(1000); //Enn ensamblador se debiò crear una rutina con
el tiempo de cada instrucciòn
      aux2=valor2*2; //Multiplicamos por dos para el recorrimiento


    }


  break;
```

```
  case '4':
  int var2;
        var2=0x80;//Inicilizamo valor 2 con valor hexadecimal
        output_b(var2); //Var2 lo sacmos por el puerto b
        delay_ms(1000);  // Retardo de un segundo
        do{
                var2=var2/2;// Reducimos el valor a la mitad para el
soguiente led
          output_b(var2);  // El var2 pasa al puerto b
          delay_ms(1000); // Retardo de un segundo
        }while (var2!=1);  //condicion de paro para  valor sea difente de 1
        var2=0x01;
        output_b(var2);
        delay_ms(1000); //Retardo de un segundo
        do{
          var2+=var2; // Aumenta valor de var2
          output_b(var2); // El var2 pasa al puerto b
          delay_ms(1000); //retardo de un segundo
        }while (var2!=0x80);   // Hasta llegar al bit menos significativos
  break;
  case '5':
    int conta=0;
         while(conta<5){   //En ensamblador hubieramos realizado una
comparación contra un bit
    output_b(0xff); //poner unos en todos los bits del puerto b
    printf(" Todos los bits encendidos \n\r"); //impresion de pantalla
    delay_ms(1000); // retardo fe un segundo
       output_b(0x00);//En ensamblador hubieramos mandado un 00 al
puerto B
    printf(" Todos los leds apagados \n\r"); // impresion de pantalla
    delay_ms(1000); // retardo de un segundo
```

```
    conta++; //En ensamblador hubieramos utilizado la instrucción INCF
    }//while
    break;
}
}
}//main
```

Conclusiones:

Principalmente se puede concluir que se cumplió el objetivo de la práctica, pues aprendimos a programar el pic con lenguaje C y aplicar las sentencias de control para recibir datos y ejecutar instrucciones. También cabe mencionar que aunque puede parecer sencilla la programación con lenguaje C fue de mucha utilidad comenzar con ensamblador ya que así comprendemos lo que esta realizando cada instrucción de C e imaginamos como lo realizaríamos en ensamblador, además de que nos ayuda a mejorar la lógica de nuestros programas. Una gran ventaja que veo sobre programar en lenguaje C es el tiempo que al realizar los programas; si en ensamblador te tardabas 40 minutos , con C puedes tardarte a lo mucho 15. Pero nunca hay que olvidar que la tarea del compilador es facilitarnos la traducción a ensamblador para posteriormente cargar el programa al pic.