Práctica No. 5b

• Sección de Entrada/Salida

Objetivo: Acceso a los puertos genéricos del dispositivo programable PPI-8255 que se encuentra **emulado** en la tarjeta T-Juino.

Material: - Programa ensamblador y encadenador (TASM/TLINK)

- Tarjeta T-Juino.

Equipo: - Computadora Personal

Teoría: * * * INVESTIGACION: INTERFAZ DE PUERTOS PARALELO 8255 * * *

Desarrollo:

Parte A: Programación y uso de puertos del **PPI8255.**

1) Crear archivo **Prac5b.c** que contenga a partir del listado 1

Listado 1

```
#define BYTE unsigned char
#define WORD unsigned int

#define PA     0x40
#define PB     0x41
#define PC     0x42
#define RCtr   0x43
#define PTOs_all_out 0x80

char dato;

void main( void ){
    puts("Practica 5b\n\r");          /* imprimir mensaje           */
    outportb(RCtr, PTOs_all_out);     /* inicializar 8255           */
    outportb(PA,0x55);                /* presentar 55h en el Pto A  */
    while(1){
        dato = getch();               /* leer tecla                 */
        outportb(PB,dato);            /* presentar tecla en PB      */
        printBin(dato);
        puts("\n\r");
    }
}

/* función para lectura de puertos usando ensamblador in-line     */
void outportb( WORD port, BYTE dato){
   asm  mov dx, port
   asm  mov al, dato
   asm  out dx, al
}

/* función simple para desplegar un byte en formato binario       */
void printBin( BYTE dato ){
    BYTE msk=0x80;
    do{
        putchar( (dato & msk) ? '1' : '0' );
        msk>>=1;
    }while( msk );
}
```

2) Realizar proceso para ejecutar el programa en T-Juino.

3) La siguiente función llamada **SetBitPort** manipula la información de un puerto dado para activar un determinado bit. Es decir mediante ella se puede **activar** (hacer uno) un bit del puerto. El número del bit esta en el rango de 0 a 7 siendo el bit 7 es más significativo.

Listado 2

```
void SetBitPort(WORD Puerto, BYTE num_bit)
{

   BYTE mask=0x01;                    /* mascara inicial                  */
   BYTE temp;                         /* dato auxiliar                    */

    temp = inportb( Puerto );         /* leer dato del puerto             */
    mask = mask << num_bit;           /* ajustar mascara según num_bit    */
    temp = temp | mask;               /* aplicar mascara con operador OR  */
    outportb( Puerto , temp );        /* presentar resultado en el puerto */

}
```

Listado 2 – Simplificado

```
void SetBitPort(WORD Puerto, BYTE num_bit)
{

    outportb( Puerto , inportb( Puerto )|( 0x01 << num_bit ) );

}
```

4) Diseñe las siguientes funciones para manipulación de bit de puertos .

Nota: *Es necesario implementar las funciones básica de E/S **inportb( )** y **outportb( )** en lenguaje ensamblador en archivo ASM independiente -- no usar funciones in-line.*

a) Función **ClrBitPort** la cual **borrar** un bit; es decir hace cero el bit de la posición num_bit del puerto dado por el parámetro Puerto.

```
void ClrBitPort( WORD Puerto, BYTE num_bit )
```

Función **NotBitPort** la cual **invierte** el bit de la posición num_bit del puerto dado por el parámetro Puerto.

```
void NotBitPort( WORD Puerto, BYTE num_bit )
```

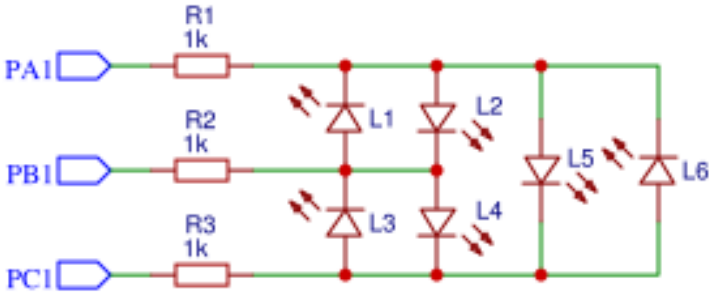
b) Función **TstBitPort** la cual retorna el estado del bit de la posición num_bit del puerto dado por el parámetro Puerto. Si el bit del puerto está en 0 lógico entonces la función retorna valor cero (0) de otra forma retorna valor uno (1).

```
BYTE TstBitPort ( WORD Puerto, BYTE num_bit )
```

5) Verifique el funcionamientos de las funciones del punto anterior realizando los siguientes pasos:

   a. Capturar 8 bits mediante ***TstBitPort*** en **PC4**, dando tiempo para que el usuario pueda alterar el valor de **PC4** (por ejemplo, detener el programa mediante un *getchar()*), y mostrar los bits capturados en la terminal,

   b. Mostrar los 6-bits menos significativos sobre el siguiente arreglo de LEDs:



Respetando L1 como el LSB, y L6 como el MSB.

Conclusiones y Comentarios.

Bibliografía