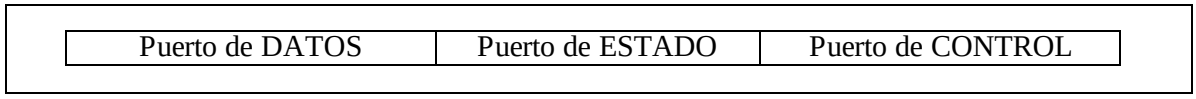
Periféricos Avanzados
Práctica 3. Programación del puerto paralelo.

Fundamento teórico.

El puerto paralelo de un PC es una plataforma barata y una potente para implementar proyectos que deban hacerse con el control de periféricos.

- Asignación de puertos:

El puerto de impresora consta de tres direcciones de puertos , una correspondiente al puerto de datos, otra al puerto de estado y otra al puerto de control. Estas direcciones se encuentran en orden secuencial, lo que quiere decir que si el puerto de datos está por ejemplo en la dirección 0x0378, el puerto de estado correspondiente estará en 0x0379 y el de control en 0x037A. Por lo tanto la abstracción de un puerto paralelo ha de ser el conjunto de estos tres puertos:



Puerto de impresora (puerto paralelo)

Para identificar definitivamente qué direcciones de puertos tiene nuestro(s) puerto(s) podemos leer de la dirección de BIOS 0040:0008 (*0x00000408*).

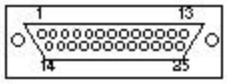
NOTA:

Recuerden que lo que contiene la dirección de BIOS *0x00000408* son direcciones de memoria, es más, son direcciones de memoria de :

M[0x00000408] : dirección base de LPT1 .
M[0x00000408]+1: dirección base de LPT2., etc ...

Por tanto, se ha de seleccionar el tipo de datos adecuado a la hora de establecer el puntero a esa zona de memoria para leer su contenido.

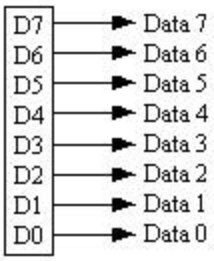
- Descripción de las salidas:

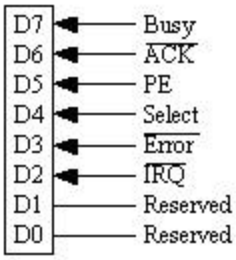


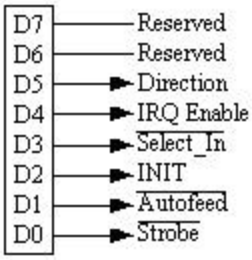
Conector DB-25 macho.
A la derecha se muestra la asignación de bits.

1	Strobe
2	Data 0
3	Data 1
4	Data 2
5	Data 3
6	Data 4
7	Data 5
8	Data 6
9	Data 7
10	ACK
11	Busy
12	Paper Empty
13	Select
14	Auto Feed
15	Error
16	Initialize Printer
17	Select Input

A continuación se muestra la asignación de los puertos :







Puerto de datos	Puerto de estado	Puerto de control
Relación posicional del byte del puerto de datos. Todos son de salida. Todas las salidas siguen una lógica de nivel alto (1 = 5v), escribir un uno en un bit supone activar a 5v la patilla correspondiente.	Algunos de los bits en este puerto siguen la lógica invertida, lo que supone que sacar un 1 lógico por el puerto hace que se vea un 0 lógico. Esto da un poco más de complejidad al puerto paralelo, pero se soluciona invirtiendo los bits que usan lógica negativa con la función XOR – ver ejemplo-	Para hacer que la impresora tome el dato que hemos escrito en el puerto de datos es necesario poner Strobe a cero momentáneamente (1 invertido). Pero para sacar bits por el puerto de datos si estos no tienen como destino a una impresora no tiene por que usarse strobe .

Como ejemplo sirva este fragmento que tiene como destino el mandar un byte al puerto de datos y otro al de control :

```
val1 = 0x81;   /* 1000 0001 */      /* Activar bits 7 y 0*/
outportb(puertoDatos, val1);

val2 = 0x08;   /* 0000 1000 */
outportb(puertoControl, VAL2^0x0b);

/* SELECT_IN = 1, INIT = 0, /AUTO_FEED = 0, /STROBE = 0 */
```

- Descripción de las entradas :

Sólo tiene sentido si se usa el puerto paralelo para comunicarse con una impresora. Si SELECT tiene un 1 indica que la impresora está online , un 1 en Busy o en PE (Paper Empty) indica que la impresora o está ocupada o se ha quedado sin papel. Un cero en /ACK indica que la impresora ha recibido algo y un 0 en ERROR indica que la impresora se haya en una situación de error. Esta información está disponible leyendo los 5 bytes mas significativos del puerto de estado. Es decir, en condiciones normales el estado significativo del puerto se consigue leyendo del puerto de estado los *cinco* bits más significativos, los demás no interesa leerlos.

NOTA: Aunque Busy debe seguir lógica positiva, los diseñadores del hardware de las impresoras han invertido el bit asociado a Busy usando hardware por tanto para activar BUSY es necesario mandar un 0 o un 1 invertido.

Ejemplo xor

¿Cómo se invierte un determinado bit?

En c hay operadores lógicos a nivel de bit como ya vimos en la práctica 2 (ratón) , uno de ellos es ^ que es el que se usa para realizar el xor entre dos bits.

variable = ((inportb(puertoEstado)^0x80) >> 3);

a	b	a^b
0	0	0
0	1	1
1	0	1
1	1	0

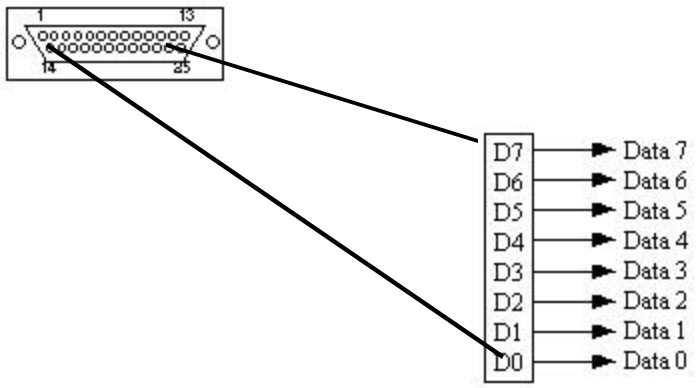
***Explica con un ejemplo qué hace la instrucción* " variable = ((inportb(puertoEstado)^0x80) >> 3)"**

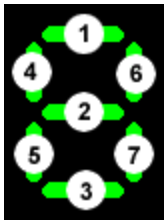
¿Por qué se desplaza tres posiciones a la derecha el resultado de la operación?. Explica la respuesta

Realización práctica.

Obligatorio

Implementar un contador BCD-7 segmentos a través del puerto paralelo.
Representación gráfica:





El puerto paralelo se conecta al bus que se proporciona en clase, este bus tiene 9 salidas, 8 correspondientes a datos y una correspondiente a tierra.

* Es importante notar que entre la salida del bus y la entrada del display se necesitan acoplar resistencias.

El display toma como entrada en cada segmento una salida del bus de tal forma que si en esa salida del bus hay 5 voltios (un 1 logico) el segmento se activa.

Pasos a seguir para realizar la práctica:

1.- Conocer que dirección tienen los puertos.
2.- Probar a escribir / leer de los mismos.
3.- Implementar una función en la que dado un numero entre 0 y 9 de cómo resultado un byte en el que se representan los segmentos que se activarán, por ejemplo:

Deseo que en el led aparezca un 0 -> debo activar los segmentos 1,3,4,5,6 y 7, por tanto tendré que dar como salida al puerto de datos el byte : 01111101 .

Se pueden comprobar todas las interacciones que hagáis con el puerto paralelo usando el programa monitor.zip, con este programa (tras instalarlo) podréis comprobar que salida se le está mandando al puerto paralelo sin necesidad del material de clase.

Opcional

La parte opcional de la práctica consiste en interrumpir un programa a través de una interrupción de puerto paralelo, es decir, se debe forzar una interrupción en el puerto paralelo.
El programa debe mostrar con un contador el numero de veces que ocurre la interrupción.

Pista: ACK = 0 y activación del BIT 4 puerto de control

Detalles para la elaboración de esta parte.

La intención de esta parte práctica es la de aprender a usar interrupciones hardware a través del puerto paralelo.

Una interrupción hardware es el medio con el que se detienen las operaciones software que se estén realizando por que en algún lugar ha ocurrido un evento hardware, y estas operaciones que se estaban realizando se detienen para realizarse otras tareas software que manejen la excepción que ha ocurrido. Cuando se ha ejecutado la rutina que se tenia preparada para cuando ocurriese esta excepción, se reanudan las operaciones previas a la interrupción.

Tabla de manejadores de interrupciones:

Cuando ocurre una interrupción, el PC debe saber dónde ir para ejecutar la rutina de tratamiento de esa interrupción. Esta tabla alberga 256 tipos de interrupciones (herencia del 8088) cada tipo de interrupción tiene cuatro bytes en una tabla que comienza en la dirección de memoria 0x00000000. Asi por tanto, la INT 0 utiliza las localizaciones de memoria 0x00000000.....0x00000003, la interrupción hardware 9 INT9 comienza en la 0x00000024, por tanto, estos 1024 bytes (256 x 4 bytes) es lo que se conoce como tabla del vector de interrupciones. Estos cuatro bytes forman la dirección a la que debe saltar el control cuando ocurra una interrupcion.

IBM reservó ocho interrupciones hardware, empezando en INT 0x08 , para futuras expansiones, y coloquialmente se denominan **IRQ0.....IRQ7** así IRQ es INT8, IRQ1 es INT9 ..... .

Para ver qué direcciones tienen asignadas IRQ0.....IRQ7 ,como ejercicio , se puede usar debug de msdos.

Teclead en la consola msdos: debug (intro).
Luego:

D 0000:0020 20

Recordad que la dirección de almacenamiento para la INT8 (IRQ0) era 0x0020

Resultado :

B3 10 3B 0B 73 2C 3B 0B-57 00 70 03 8B 3B 3B 0B
ED 3B 3B 0B AC 3A 3B 0B-B7 00 70 03 F4 06 70 00

Lo marcado en rojo se corresponde con la direccion de IRQ0 que es :
0B3B:10B3, por tanto para IRQ7 (azul) sería : 0070:06F4.

Por tanto, sabemos que cuando ocurre una interrupción IRQ7 se está produciendo un INT 0x0f.

Modificar el manejador del vector de interrupciones

Se usará lo mismo que en prácticas anteriores, la funcion getvect para obtener la dirección del manejador antiguo y setvect para establecer un nuevo manejador, en este caso .....:

Vectorviejo = getvect(0x0f); y recordad que para establecer un manejador vuestro se usaba:

Setvect(0x0f, vuestro_manejador);

Enmascaramiento de interrupciones:

Si se enmascara una interrupción , le estamos indicando al pc que la ignore, por tanto debemos establecer la mascara para asegurarnos que IRQ7 está activa.

El puerto 0x21 es el puerto que se asocia con la mascara de interrupciones, asi que por ser un puerto podemos activar las interrupciones via IRQ7 con lo siguiente:

Mascara=inportb(0x21) & ~0x80;
Con esta linea obtenemos la mascara actual, ponemos a 1 el bit 7 y dejamos a los demas bits intactos.

Outportb(0x21,Mascara);
Con esta linea estamos activando la mascara.

A partir de ahora siempre que se produzca una interrupcion IRQ7 se saltará a "vuestro_manejador". Dentro del manejador se puede hacer lo que se desee, pero siempre hay que terminar con el ya conocido EOI en el PIC (ver práctica de teclado): outportb(0x20,0x20);

Otro consejo es que antes de terminar el programa se salga restableciendolo todo:

```
mascara=inportb(0x21) | 0x80;
outportb(0x21, mascara);

setvect(0x0f, vectorviejo);
```

NOTA: En nuestro caso podemos activar el IRQ con el bit 4 del puerto de control de la siguiente manera :

outportb(inportb(puertoControl) | 0x10);

Antes de salir del programa , para desactivar el IRQ que hemos habilitado se debe hacer:

outportb(CONTROL, inportb(CONTROL) & ~0x10);

NOTA : en dos.h existen funciones como **enable()** y **disable()** que habilitan/ignoran las interrupciones.