







# PRÁCTICA 2




## *E/S mediante Consulta de Estado. Ejercicios Integradores.*

 **Objetivos** Comprender la comunicación entre el microprocesador y los periféricos externos (luces , llaves  e impresora ). Configurar la interfaz de entrada/salida (PIO ) y el dispositivo de handshaking (HAND-SHAKE .

### Resumen de dispositivos básicos de entrada/salida del VonSim:

Dirección	Dispositivo	Nombre	Función
30h	PIO	PA	Registro de datos del puerto paralelo de comunicación A. Tiene 8 conectores programables, cada uno puede enviar o recibir un bit.
31h	PIO	PB	Registro de datos del puerto paralelo de comunicación B. Tiene 8 conectores programables, cada uno puede enviar o recibir un bit. Funciona de forma independiente a PA.
32h	PIO	CA	Registro de configuración del puerto paralelo de comunicación A (PA). Tiene 8 bits, para configurar el sentido de cada uno de los conectores correspondientes. Un valor de 0 indica que el conector es de salida, y un bit de 1 indica que es de entrada.
33h	PIO	CB	Idem CA pero para el puerto PB
40h	Handshake	Dato	Registro de datos del puerto paralelo Dato para comunicarse con la impresora. Sirve para enviar caracteres a imprimir a dicho dispositivo.
41h	Handshake	Estado	Registro de estado y configuración. Tiene 8 bits, pero solo los bits 0, 1 y 7 se utilizan para interactuar con la impresora. El formato del registro es IXXXXXSB, donde I es el bit que configura al Handshake en modo consulta de estados o interrupciones. S y B indican el valor de la señales de Strobe y Busy, respectivamente, de la impresora.

**Resumen de conexión del VonSim** Cada conexionado conecta distintos dispositivos al PIO y/o al HANDSHAKE

Conexionado	Conexión
PIO  Luces/Llaves	<ul style="list-style-type: none"> <li>• Puerto <b>PA</b> del PIO conectado a las 8 Llaves (dispositivo de entrada)</li> <li>• Puerto <b>PB</b> del PIO conectado a las 8 Luces (dispositivo de salida)</li> </ul>
PIO  Impresora	<ul style="list-style-type: none"> <li>• Puerto <b>PB</b> del PIO al puerto de datos de 8 bits de la impresora (salida)</li> <li>• Puerto <b>PA</b> del PIO:               <ul style="list-style-type: none"> <li>○ El <b>bit 0</b> a la señal busy de la impresora (entrada)</li> <li>○ El <b>bit 1</b> a la señal strobe de la impresora (salida)</li> </ul> </li> </ul>
HANDSHAKE  Impresora	<ul style="list-style-type: none"> <li>• Puerto <b>Dato</b> del Handshake al puerto de datos de 8 bits de la impresora (salida)</li> <li>• Puerto <b>Estado</b> del Handshake:               <ul style="list-style-type: none"> <li>○ El bit 0 a la señal busy de la impresora (entrada)</li> <li>○ El bit 1 a la señal strobe de la impresora (salida)</li> </ul> </li> </ul>

## Parte 1: Entrada/Salida con Luces y Llaves mediante el PIO

### 1) Instrucciones de entrada y salida con Luces y Llaves ★

Los siguientes programas interactúan mediante instrucciones IN y OUT con las luces y llaves a través del PIO. Completar las instrucciones faltantes, e indicar que hace el programa en cada caso.

<b>PB EQU 31h</b> <b>CB EQU 33h</b>  <b>ORG 2000h</b> <u>mov al, 0</u> out CB, al mov al, 0Fh out PB, al int 0 end	<b>PA EQU 30h</b> <b>CA EQU 32h</b> <b>ORG 1000h</b> msj db "Apagadas" <b>ORG 2000h</b> mov al, 0FFh <u>OUT CA,AL</u> in al, PA cmp al, 0 jnz <b>fin</b> mov al, 8 mov bx, offset msj int 7 <b>fin:</b> int 0 end	<b>CB EQU 33h</b> <b>PB EQU 31h</b> <b>CA EQU 32h</b> <b>PA EQU 30h</b>  <b>ORG 2000h</b> <u>MOV AL, 0FFh</u> <u>OUT CA,AL</u> <u>MOV AL, 0</u> out CB, al in al, PA out PB, al int 0 end
-----------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

### 2) Uso de las luces y las llaves a través del PIO. ★★

Ejecutar los programas con el simulador VonSim utilizando los dispositivos "Llaves y Luces" que conectan las llaves al puerto PA del PIO y a las luces al puerto PB.

- Patrón de Luces Fijo** Escribir un programa que encienda las luces con el patrón 11000011, o sea, solo las primeras y las últimas dos luces deben prenderse, y el resto deben apagarse.
- Verificar Llave** Escribir un programa que verifique si la llave de más a la izquierda está prendida. Si es así, mostrar en pantalla el mensaje "Llave prendida", y de lo contrario mostrar "Llave apagada". Solo importa el valor de la llave de más a la izquierda (bit más significativo). Recordar que las llaves se manejan con las teclas 0-7.
- Control de luces mediante llaves** Escribir un programa que permite encender y apagar las luces mediante las llaves. El programa no deberá terminar nunca, y continuamente revisar el estado de las llaves, y actualizar de forma consecuente el estado de las luces. La actualización se realiza simplemente prendiendo la luz *i*, si la llave *i* correspondiente está encendida (valor 1), y apagándola en caso contrario. Por ejemplo, si solo la primera llave está encendida, entonces solo la primera luz se debe quedar encendida.

# PARTE 1

2 a .-

```
PB EQU 31h
CB EQU 33H
ORG 1000H ; MEMORIA DE DATOS
PATRONCIS db 11000011b
ORG 2000H
MOV AL, 0
out CB, al
MOV AL, PATRONCIS
OUT PB,AL
int 0
end
```

2 b.-

```
PA EQU 30h
CA EQU 32H
ORG 1000H
KEY DB 10000000b
MSJ DB "Llave Prendida"
FINMSJ DB ?
MSJAP DB "Llave Apagada"
FINMSJAP DB ?
ORG 2000H
MOV AL, 0FFH
out CA, al
IN AL,PA
AND AL, KEY
JZ APAGADA
mov al, offset finmsj - offset msj
mov bx, offset msj
int 7
JMP FIN
APAGADA:
mov al, offset finmsjap - offset msjap
mov bx, offset msjap
int 7
FIN:
int 0
end
```

3 c.-

```
PA EQU 30H
PB EQU 31H
CA EQU 32H
CB EQU 33H
ORG 2000H
MOV AL,0FFH
OUT CA,AL
MOV AL, 0
OUT CB,AL
REPEAT:
    IN AL,PA
    OUT PB,AL
    JMP REPEAT
INT 0
END
```

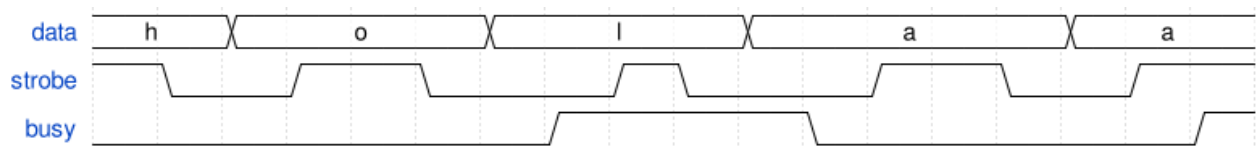
## Parte 2: Entrada/Salida con la Impresora mediante el PIO y el HANDSHAKE

### 1) Protocolo de comunicación Centronics para la impresora ★

El protocolo Centronics sirve para indicarle a la impresora cuando se está enviando un carácter a imprimir. El protocolo utiliza 3 señales: **busy**, **strobe** y **data**. Para enviar un carácter a la impresora, se debe esperar que la misma esté libre (**busy** = 0), luego poner el carácter en la señal de **data**, y finalmente indicar que hay un carácter para imprimir mediante la generación de un **flanco ascendente** en la señal de **strobe**. Un flanco ascendente sucede cuando la señal cambia de 0 a 1.

La impresora sólo imprime cuando está disponible (**busy** es 0) y recibe una señal de flanco ascendente en la línea de **strobe**. Si no se cumple alguna de estas condiciones, la impresora ignora la línea de **data**.

El siguiente gráfico muestra un ejemplo de las señales.



- Indica cuáles de las 3 señales son de entrada o de salida.
- Si bien la señal de **data** envía el string "holaa", las señales de **strobe** y **busy** que se envían no permiten que se impriman todos los caracteres. Indicar qué imprime la impresora. ¿Cuál es la razón por la cuál algunos caracteres no se imprimen?
- El diseño de la impresora tiene 2 señales de sincronización. No obstante, se podría pensar que sólo con la señal **busy** alcanza. ¿Qué sucedería al querer imprimir "aa" si no existiera la señal de **strobe**?

**Nota:** para simplificar, la línea de **data** de la impresora se muestra directamente con el carácter a enviar. No obstante, en la realidad **data** consta de 8 líneas de 1 bit que representan el código ascii del carácter.

### 2) Verificación del bit busy ★

El siguiente programa tiene como objetivo verificar el bit de busy a través del PIO e imprimir "Ocupada" si la impresora está ocupada y "Libre" de lo contrario.

- Complete el código para que el programa funcione correctamente.
- Modifique el código para que el programa no imprima "Ocupada". En lugar de eso, el programa debe esperar a que el bit de busy sea 0 usando consulta de estado. Cuando eso suceda, imprimir "Libre" y terminar el programa.

<b>PA EQU 30h</b>	
<b>CA EQU 32h</b>	<b>jnz ocupada</b>
<b>ORG 1000h</b>	
<b>si db "Ocupada"</b>	<b>mov al, 5</b>
<b>no db "Libre"</b>	<b>int 7</b>
	<b>jmp fin</b>
<b>ORG 2000H</b>	<b>ocupada: mov bx, offset si</b>
	<b>mov al, 7</b>
<b>out CA, al</b>	
<b>in al, PA</b>	<b>fin: int 0</b>
	<b>end</b>


### 3) Subrutina para el envío del carácter y la señal de Strobe ★★


El envío de la señal de strobe se puede modularizar en una subrutina para ser reutilizado en distintas ocasiones. Implementar una subrutina `flanco_ascendente` que envía el flanco ascendente (un 0 y luego un 1) a través del **strobe**. Asumir que el PIO ya está configurado correctamente para comunicarse con la impresora.


### 4) Uso de la impresora a través de la PIO ★★


Ejecutar los programas configurando el simulador VonSim con los dispositivos "Impresora (PIO)". En esta configuración, el puerto de datos de la impresora se conecta al puerto PB del PIO, y los bits de busy y strobe de la misma se conectan a los bits 0 y 1 respectivamente del puerto PA.

- a) **Imprimir letra fija** Escribir un programa para imprimir la letra “A” utilizando la impresora a través de la PIO. Recordar que el programa deberá esperar hasta que el bit de busy esté apagado, luego enviar el carácter, y luego enviar el flanco ascendente a través de la señal de strobe. Modularizar el programa en 4 subrutinas:

 **ini\_pio**: Inicializa el PIO configurando los registros CA y CB según corresponde a strobe, busy y puerto de datos

 **poll**: Consulta el bit busy de la impresora, e itera hasta esté en 0. Cuando está en 0 la subrutina retorna sin devolver ningún valor

 **flanco\_ascendente**: Igual que la subrutina implementada en el ejercicio anterior

 **imprimir\_caracter**: Recibe un carácter a imprimir en AL, y utilizando **poll** y **flanco\_ascendente**, realiza todos los pasos necesarios para enviar a la impresora el carácter.

- b) **Imprimir mensaje fijo** Escribir un programa para imprimir el mensaje “ORGANIZACION Y ARQUITECTURA DE COMPUTADORAS” utilizando la impresora a través de la PIO. Utilizar la subrutina **imprimir\_caracter** del inciso anterior.

- c) **Imprimir mensaje leído** Escribir un programa que solicita el ingreso de cinco caracteres por teclado y los envía de a uno por vez a la impresora a través de la PIO a medida que se van ingresando. No es necesario mostrar los caracteres en la pantalla. Utilizar la subrutina **imprimir\_caracter** del inciso anterior.

5) **Uso de la impresora a través del HAND-SHAKE.** ★★

El HANDSHAKE es un dispositivo diseñado específicamente para interactuar con la impresora mediante el protocolo centronics. Por este motivo, no requiere enviar la señal de flanco ascendente manualmente; en lugar de eso, al escribir un valor en su registro **DATA**, el mismo HANDSHAKE manda el flanco ascendente. Ejecutar los programas configurando el simulador VonSim con los dispositivos “Impresora (Handshake)”

- a) Escribir un programa que imprime “INGENIERIA E INFORMATICA” en la impresora a través del HAND-SHAKE. La comunicación se establece por **consulta de estado** (polling). Para ello, implementar la subrutina **imprimir\_caracter\_hand** que funcione de forma análoga a **imprimir\_caracter** pero con el handshake en lugar del PIO.
- b) ¿Qué diferencias encuentra con el ejercicio anterior? ¿Cuál es la ventaja en utilizar el HAND-SHAKE con respecto al PIO para comunicarse con la impresora? ¿Esta ventaja también valdría para otros dispositivos, como las luces?

## Parte 3: Entrada/Salida con dispositivos genéricos a través del PIO

### 1) PIO con dispositivos genéricos ★★ ★

Si bien el PIO solo permite conectarse a las luces, interruptores y la impresora en este simulador, al ser un dispositivo programable podríamos utilizarlo para interactuar con nuevos dispositivos que no cuenten con un hardware especializado para interactuar con ellos. En los siguientes incisos, te proponemos escribir programas que usen nuevos dispositivos ficticios, pero con un protocolo de comunicación definido. **Nota:** los dispositivos nuevos no están implementados en el simulador, pero podés probar el funcionamiento del programa utilizando las luces y las llaves haciendo de cuenta que sirven como entrada y salida del dispositivo.

- a) Escribir un programa que, utilizando el puerto PB del PIO, envíe la cadena de caracteres “UNLP” a un dispositivo nuevo. Este dispositivo debe recibir la cadena de a un carácter a la vez. Para distinguir entre caracteres, el dispositivo necesita que antes de cada carácter nuevo se envíe el código ASCII 0. Además, para indicar que se finalizó el envío, debe mandarse el código ASCII 255.

**Ejemplo:** Para transmitir la cadena “UNLP”, debe enviarse: 0, “U”, 0, “N”, 0, “L”, 0, “P”, 255

- b) Escribir un programa que reciba una cadena de caracteres de un dispositivo nuevo conectado a los puertos PA y PB. Este dispositivo envía la cadena de a un carácter a la vez. Para que el dispositivo sepa cuándo la CPU está lista para recibir un carácter, la CPU deberá enviar el valor FF al dispositivo a través del puerto PB. Luego, la CPU deberá leer desde el puerto PA, y volver a enviar el valor FF al dispositivo. La transmisión termina cuando se recibe el código ASCII 0.


**Ejemplo** para recibir la cadena “ASD”: CPU envía el FF por PB → CPU recibe “A” por PA → CPU envía el FF por PB → CPU recibe “S” por PA → CPU envía el FF por PB → CPU recibe “D” por PA → CPU envía el FF por PB → CPU recibe 0 por PA

## Parte 4: Ejercicios integradores o tipo parcial

### 1) Juego de Luces con Rotaciones ★★★★★

Escribir un programa que encienda una luz a la vez, de las ocho conectadas al puerto paralelo del microprocesador a través de la PIO, en el siguiente orden de bits: 0-1-2-3-4-5-6-7-6-5-4-3-2-1-0-1-2-3-4-5-6-7-6-5-4-3-2-1-0-1-..., es decir, 00000001, 00000010, 00000100, etc. El programa nunca termina. Para ello, deberá utilizar las subrutinas **ROTAR\_IZQ** y **ROTAR\_DER\_N**, que le permitirán rotar el bit de estado de las luces y generar el juego correspondiente.

Las rotaciones son operaciones que, aunque no parezca, tienen muchas utilidades, como dividir o multiplicar por dos de forma rápida, o manipular los bits de un byte, pero no hay una instrucción que las implemente directamente.

 **ROTARIZQ:** Escribir una subrutina ROTARIZQ que aplique **una** rotación hacia la izquierda a los bits de un byte almacenado en la memoria. Dicho byte debe pasarse por valor desde el programa principal a la subrutina a través de registros y por referencia. No hay valor de retorno, se modifica directamente la celda de memoria referenciada.

**Pista:** Una rotación a izquierda de un byte se obtiene moviendo cada bit a la izquierda, salvo por el último que se mueve a la primera posición. Por ejemplo al rotar a la izquierda el byte 10010100 se obtiene 00101001, y al rotar a la izquierda 01101011 se obtiene 11010110.

Para rotar a la izquierda un byte, se puede multiplicar el número por 2, o lo que es lo mismo sumarlo a sí mismo. Por ejemplo (verificar):

```

01101011
+ 01101011
11010110 (CARRY=0)


```

Entonces, la instrucción add ah, ah permite hacer una rotación a izquierda. No obstante, también hay que tener en cuenta que si el bit más significativo es un 1, el carry debe llevarse al bit menos significativo, es decir, se le debe sumar 1 al resultado de la primera suma.

```

10010100
+ 10010100
00101000 (CARRY=1)
+ 00000001
00101001

```

 **ROTARIZQ\_N:** Usando la subrutina ROTARIZQ del ejercicio anterior, escriba una subrutina ROTARIZQ\_N que realice N rotaciones a la izquierda de un byte. La forma de pasaje de parámetros es la misma, pero se agrega el parámetro N que se recibe por valor y registro. Por ejemplo, al rotar a la izquierda 2 veces el byte 10010100, se obtiene el byte 01010010.

 **ROTARDER\_N:** \* Usando la subrutina ROTARIZQ\_N del ejercicio anterior, escriba una subrutina ROTARDER\_N que sea similar pero que realice N rotaciones hacia la **derecha**.

**Pista:** Una rotación a derecha de N posiciones, para un byte con 8 bits, se obtiene rotando a la izquierda 8 - N posiciones. Por ejemplo, al rotar a la derecha 6 veces el byte 10010100 se obtiene el byte 01010010, que es equivalente a la rotación a la izquierda de 2 posiciones del ejemplo anterior.

### 2) CriptoLlaves (Llaves, Luces): ★★★★★

Escriba un programa de VonSim que permita jugar al CriptoLlaves. El usuario debe adivinar un patrón secreto de 8 bits que está almacenado en la memoria del programa. Para ello, debe manipular las llaves hasta que el patrón de bits de las llaves sea exactamente igual al del patrón secreto. Como ayuda para el usuario, si el estado de una llave acierta al bit correspondiente, el programa debe prender el led correspondiente. Por ejemplo, si el patrón es 0100 0101 y las llaves están en el estado 1110 0100, deben prenderse las luces de los bits 1, 2, 3, 4 y 6.



Como no acertó a todos los bits, el usuario no ha adivinado el patrón y debe continuar jugando. El programa termina cuando el usuario acierta todos los bits del patrón, mostrando en pantalla el mensaje "GANASTE".

### 3) Llaves y mensajes ★★

- a) Escribir un programa que continuamente verifique el estado de las llaves. Si están prendidas la primera y la última llave, y el resto están apagadas (patrón 10000001), se debe mostrar en pantalla el mensaje "ACTIVADO". En caso contrario, no se debe mostrar nada.
- b) Modificar a) para que el mensaje se imprima una sola vez cada vez que detecte ese patrón de bits. Por ejemplo, si el programa lee la siguiente secuencia de patrones:  
00010101 → 10010000 → **10000001** → **10000001** → **10000001** → 10010001 → **10000001** → **10000001** → 10010101 → 01110001  
Entonces solo deberá imprimir "ACTIVADO" dos veces.

### 4) Luces, llaves y opciones ★★★ Escribir un programa que deberá utilizar las luces y llaves. El programa revisa el estado de las llaves, y evalúa estos tres casos:

- A. Verificar si todas llaves están apagadas. Si es así, mostrar en pantalla el mensaje "Fin de programa" y finalizar el mismo. Caso contrario, hacer tanto B como C.
- B. Actualizar las luces a su estado opuesto. Por ejemplo, si las llaves están en el estado "00011010" las luces tendrán el estado "11100101".
- C. Si la primera llave (la del bit menos significativo) está prendida, mostrar en pantalla el mensaje "Arquitectura de Computadoras: ACTIVADA".

El programa sólo termina en el caso A. En los casos B y C, debe volver a revisar el estado de las llaves y evaluar los 3 casos otra vez.

Las funciones "A", "B" y "C" deben implementarse utilizando tres subrutinas independientes. La subrutina A debe devolver 1 si hay que finalizar el programa y 0 de lo contrario.