**Trabajo Práctico N° 3:**

**Entrada/Salida.**

**Ejercicio 1: 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.*

**(a)** *Escribir un programa que encienda las luces con el patrón 11000011, o sea sólo las primeras y las últimas dos luces deben prenderse y el resto deben apagarse.*

PB EQU 31h

CB EQU 33h

org 1000h

PATRON DB 11000011b

org 2000h

mov al, 0

out CB, al

mov al, PATRON

out PB, al

int 0

end

**(b)** *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”. Sólo 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.*

PA EQU 30h

CA EQU 32h

org 1000h

MSJ1 DB “Llave prendida”

FIN1 DB ?

MSJ2 DB “Llave apagada”

FIN2 DB ?

org 2000h

mov al, 0FFh

out CA, al

in al, PA

and al, 80h

cmp al, 0

jz APAGADA

mov bx, offset MSJ1

mov al, offset FIN1 - offset MSJ1

jmp FIN

APAGADA: mov bx, offset MSJ2

mov al, offset FIN2 - offset MSJ2

FIN: int 7

int 0

end

**(c)** *Escribir un programa que permita 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 sólo la primera llave está encendida, entonces, sólo la primera luz se debe quedar encendida.*

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

POLL: in al, PA

out PB, al

jmp POLL

int 0

end

**(d)** *Escribir un programa que implemente un encendido y apagado sincronizado de las luces. Un contador, que inicializa en cero, se incrementa en uno una vez por segundo. Por cada incremento, se muestra a través de las luces, prendiendo sólo aquellas luces donde el valor de las llaves es 1. Entonces, primero, se enciende sólo la luz de más a la derecha, correspondiente al patrón 00000001. Luego, se continúa con los patrones 00000010, 00000011 y así sucesivamente. El programa termina al llegar al patrón 11111111.*

CONT EQU 10h

COMP EQU 11h

EOI EQU 20h

IMR EQU 21h

INT1 EQU 25h

PB EQU 31h

CB EQU 33h

N\_CLK EQU 10

org 40

IP\_CLK DW RUT\_CLK

org 3000h

RUT\_CLK: push ax

mov ax, cx

out PB, ax

inc cx

cmp cx, 256

jnz SEGUIR

mov al, 0FFh

out IMR, al

jmp FIN

SEGUIR: mov al, 0

out CONT, al

FIN: mov al, EOI

out EOI, al

pop ax

iret

org 2000h

cli

mov al, 0FDH

out IMR, al

mov al, N\_CLK

out INT1, al

mov al, 0

out CONT, al

mov al, 10

out COMP, al

mov al, 0

out CB, al

mov cx, 0

sti

LAZO: cmp cx, 256

jnz LAZO

int 0

end

**(e)** *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. Cada luz, debe estar encendida durante un segundo. El programa nunca termina.*

Opción 1:

PB EQU 31h

CB EQU 33h

org 1000h

PATRON DB 0,1,2,4,8,16,32,64,128

org 2000h

mov bx, offset PATRON

mov al, 0

out CB, al

CRECER: inc bx

mov al, [bx]

out PB, al

cmp byte ptr [bx], 128

jnz CRECER

DECRECER: dec bx

mov al, [bx]

out PB, al

cmp byte ptr [bx], 1

jnz DECRECER

jmp CRECER

int 0

end

Opción 2:

CONT EQU 10h

COMP EQU 11h

EOI EQU 20h

IMR EQU 21h

INT1 EQU 25h

PB EQU 31h

CB EQU 33h

N\_CLK EQU 10

org 40

IP\_CLK DW RUT\_CLK

org 1000h

PATRON DB 0,1,2,4,8,16,32,64,128

org 3000h

RUT\_CLK: push ax

mov al, 0

out CONT, al

mov al, EOI

out EOI, al

pop ax

iret

org 2000h

cli

mov al, 0FDh

out IMR, al

mov al, N\_CLK

out INT1, al

mov al, 0

out CONT, al

mov al, 10

out COMP, al

mov al, 0

out CB, al

mov bx, offset PATRON

sti

CRECER: cli

inc bx

mov al, [bx]

out PB, al

sti

cmp byte ptr [bx], 128

jnz CRECER

DECRECER: cli

dec bx

mov al, [bx]

out PB, al

sti

cmp byte ptr [bx], 1

jnz DECRECER

jmp CRECER

int 0

end

**Ejercicio 2: Uso de la impresora a través del 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. Presionar F5 para mostrar la salida en papel. El papel se puede blanquear ingresando el comando BI.*

**(a)** *Escribir un programa para imprimir la letra “A” utilizando la impresora a través de la PIO.*

PA EQU 30h

PB EQU 31h

CA EQU 32h

CB EQU 33h

org 1000h

CHAR DB “A”

org 3000h

PIO: push ax

mov al, 1

out CA, al

mov al, 0

out CB, al

pop ax

ret

org 4000h

STROBE0: push ax

in al, PA

and al, 11111101b

out PA, al

pop ax

ret

org 5000h

STROBE1: push ax

in al, PA

or al, 00000010b

out PA, al

pop ax

ret

org 6000h

POLL: push ax

in al, PA

and al, 1

jnz POLL

pop ax

ret

org 2000h

call PIO

call STROBE0

call POLL

mov al, CHAR

out PB, al

call STROBE1

nop

nop

nop

nop

nop

int 0

end

**(b)** *Escribir un programa para imprimir el mensaje “ORGANIZACIÓN Y ARQUITECTURA DE COMPUTADORAS” utilizando la impresora a través de la PIO.*

PA EQU 30h

PB EQU 31h

CA EQU 32h

CB EQU 33h

org 1000h

MSJ DB “ORGANIZACIÓN Y ARQUITECTURA DE COMPUTADORAS”

FIN DB ?

org 3000h

PIO: push ax

mov al, 1

out CA, al

mov al, 0

out CB, al

pop ax

ret

org 4000h

STROBE0: push ax

in al, PA

and al, 11111101b

out PA, al

pop ax

ret

org 5000h

STROBE1: push ax

in al, PA

or al, 00000010b

out PA, al

pop ax

ret

org 6000h

POLL: push ax

in al, PA

and al, 1

jnz POLL

pop ax

ret

org 2000h

call PIO

call STROBE0

mov bx, offset MSJ

mov cl, offset FIN - offset MSJ

LAZO: call POLL

mov al, [bx]

out PB, al

call STROBE1

call STROBE0

inc bx

dec cl

jnz LAZO

int 0

end

**(c)** *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.*

PA EQU 30h

PB EQU 31h

CA EQU 32h

CB EQU 33h

org 1000h

CHAR DB ?

CHARS DB 5

org 3000h

PIO: push ax

mov al, 1

out CA, al

mov al, 0

out CB, al

pop ax

ret

org 4000h

STROBE0: push ax

in al, PA

and al, 11111101b

out PA, al

pop ax

ret

org 5000h

STROBE1: push ax

in al, PA

or al, 00000010b

out PA, al

pop ax

ret

org 6000h

POLL: push ax

in al, PA

and al, 1

jnz POLL

pop ax

ret

org 2000h

call PIO

call STROBE0

mov bx, offset CHAR

mov cl, CHARS

LAZO: int 6

call POLL

mov al, [bx]

out PB, al

call STROBE1

call STROBE0

dec cl

jnz LAZO

int 0

end

**(d)** *Escribir un programa que solicite ingresar caracteres por teclado y que, recién al presionar la tecla F10, los envíe a la impresora a través de la PIO. No es necesario mostrar los caracteres en la pantalla.*

EOI EQU 20h

IMR EQUI 21h

INT0 EQU 24h

PA EQU 30h

PB EQU 31h

CA EQU 32h

CB EQU 33h

N\_F10 EQU 10

org 40

ID\_F10 DW RUT\_F10

org 1000h

CADENA DB ?

org 3000h

RUT\_F10: push ax

mov ch, 1

mov al, 0FFh

out IMR, al

mov al, EOI

out EOI, al

pop ax

iret

org 4000h

PIO: push ax

mov al, 1

out CA, al

mov al, 0

out CB, al

pop ax

ret

org 4500h

STROBE0: push ax

in al, PA

and al, 11111101b

out PA, al

pop ax

ret

org 5000h

STROBE1: push ax

in al, PA

or al, 00000010b

out PA, al

pop ax

ret

org 5500h

POLL: push ax

in al, PA

and al, 1

jnz POLL

pop ax

ret

org 6000h

PIC: push ax

mov al, 0FEh

out IMR, al

mov al, N\_F10

out INT0, al

pop ax

ret

org 2000h

cli

call PIO

call STROBE0

call PIC

sti

mov bx, offset CADENA

mov cl, 0

mov ch, 0

LAZO1: int 6

inc bx

inc cl

cmp ch, 1

jnz LAZO1

mov bx, offset CADENA

LAZO2: call POLL

mov al, [bx]

out PB, al

call STROBE1

call STROBE0

inc bx

dec cl

jnz LAZO2

int 0

end

**Ejercicio 3: Uso de la impresora a través del HAND-SHAKE.**

*Ejecutar los programas configurando el simulador VonSim con los dispositivos “Impresora (Handshake)”.*

**(a)** *Escribir un programa que imprima “INGENIERÍA E INFORMÁTICA” en la impresora a través del HAND-SHAKE. La comunicación se establece por consulta de estado (polling). ¿Qué diferencias se encuentran con el Ejercicio 2b?*

DATO EQU 40h

ESTADO EQU 41h

org 1000h

MSJ DB “INGENIERÍA E INFORMÁTICA”

FIN DB ?

org 2000h

mov bx, offset MSJ

mov cl, offset FIN - offset MSJ

POLL: in al, ESTADO

and al, 1

jnz POLL

mov al, [bx]

out DATO, al

inc bx

dec cl

jnz POLL

int 0

end

Las diferencias que se encuentran con el Ejercicio 2b son que no es necesario configurar el PIO ni tampoco es necesario configurar las señales de strobe.

**(b)** *¿Cuál es la ventaja en utilizar el HAND-SHAKE con respecto al PIO para comunicarse con la impresora? Sacando eso de lado, ¿qué ventajas tiene el PIO, en general, con respecto al HAND-SHAKE?*

La ventaja en utilizar el HAND-SHAKE con respecto al PIO para comunicarse con la impresora es que manda señal de strobe automáticamente. Sacando eso de lado, las ventajas que tiene el PIO, en general, con respecto al HAND-SHAKE, es que sirve para comunicarse con otros dispositivos.

**(c)** *Escribir un programa que imprime “UNIVERSIDAD NACIONAL DE LA PLATA” en la impresora a través del HAND-SHAKE. La comunicación se establece por interrupciones emitidas desde el HAND-SHAKE cada vez que la impresora se desocupa.*

EOI EQU 20h

IMR EQU 21h

INT2 EQU 26h

DATO EQU 40h

ESTADO EQU 41h

N\_HSK EQU 10

org 40

IP\_HSK DW RUT\_HSK

org 1000h

MSJ DB “UNIVERSIDAD NACIONAL DE LA PLATA”

FIN\_MSJ DB ?

org 3000h

RUT\_HSK: push ax

mov al, [bx]

out DATO, al

inc bx

dec cl

cmp cl, 0

jnz FIN

mov al, 0FFh

out IMR, al

in al, ESTADO

and al, 0111111b

out ESTADO, al

FIN: mov al, EOI

out EOI, al

pop ax

iret

org 2000h

cli

mov al, 0FBh

out IMR, al

mov al, N\_HSK

out INT2, al

in al, ESTADO

or al, 10000000b

out ESTADO, al

mov bx, offset MSJ

mov cl, offset FIN\_MSJ - offset MSJ

sti

LAZO: cmp cl, 0

jnz LAZO

int 0

end

**(d)** *Escribir un programa que solicite el ingreso de cinco caracteres por teclado y los almacene en memoria. Una vez ingresados, que los envíe a la impresora a través del HAND-SHAKE, en primer lugar tal cual fueron ingresados y, a continuación, en sentido inverso. Utilizar el HAND-SHAKE en modo consulta de estado. ¿Qué diferencias se encuentran con el Ejercicio 2c?*

DATO EQU 40h

ESTADO EQU 41h

org 1000h

CHARS DB 5

CADENA DB ?,?,?,?,?

FIN DB ?

org 2000h

mov bx, offset CADENA

mov cl, CHARS

LAZO: int 6

inc bx

dec cl

cmp cl, 0

jnz LAZO

mov bx, offset CADENA

mov cl, offset FIN - offset CADENA

POLL1: in al, ESTADO

and al, 1

jnz POLL1

mov al, [bx]

out DATO, al

inc bx

dec cl

jnz POLL1

mov bx, offset CADENA+4

mov cl, offset FIN - offset CADENA

POLL2: in al, ESTADO

and al, 1

jnz POLL2

mov al, [bx]

out DATO, al

dec bx

dec cl

jnz POLL2

int 0

end

Las diferencias que se encuentran con el Ejercicio 2c son que los cinco caracteres son enviados a la impresora todos a la vez y no de a uno por vez.

**(e)** *Idem (d), pero, ahora, utilizar el HAND-SHAKE en modo interrupciones.*

EOI EQU 20h

IMR EQU 21h

INT2 EQU 26h

DATO EQU 40h

ESTADO EQU 41h

N\_HSK EQU 10

org 40

IP\_HSK DW RUT\_HSK

org 1000h

CADENA DB ?,?,?,?,?

FIN\_CADENA DB ?

org 3000h

RUT\_HSK: push ax

mov al, [bx]

out DATO, al

dec ch

cmp ch, 6

js DESC

inc bx

jmp SEGUIR

DESC: cmp ch, 5

jz SEGUIR

dec bx

SEGUIR: dec cl

cmp cl, 0

jnz FIN

mov al, 0FFh

out IMR, al

in al, ESTADO

and al, 0111111b

out ESTADO, al

FIN: mov al, EOI

out EOI, al

pop ax

iret

org 2000h

cli

mov al, 0FBh

out IMR, al

mov al, N\_HSK

out INT2, al

in al, ESTADO

or al, 10000000b

out ESTADO, al

mov bx, offset CADENA

mov cl, offset FIN\_CADENA - offset CADENA

LAZO1: int 6

inc bx

dec cl

cmp cl, 0

jnz LAZO1

mov bx, offset CADENA

mov cl, 10

mov ch, 10

sti

LAZO2: cmp cl, 0

jnz LAZO2

int 0

end

**Ejercicio 4: Uso de la impresora a través del dispositivo USART por consulta de estado.**

*Ejecutar utilizando el simulador MSX88 (versión antigua del VonSim) en configuración P1 C4 y utilizar el comando PI que corresponda en cada caso (ver uso de Comando PI en el simulador).*

**(a)** *Escribir un programa que imprima el carácter “A” en la impresora a través de la USART usando el protocolo DTR . La comunicación es por consulta de estado.*

**(b)** *Escribir un programa que imprima la cadena “USART DTR POLLING” en la impresora a través de la USART usando el protocolo DTR. La comunicación es por consulta de estado.*

**(c)** *Escribir un programa que imprima la cadena “USART XON/XOFF POLLING” en la impresora a través de la USART usando el protocolo XON/XOFF realizando la comunicación entre CPU y USART por consulta de estado.*

**Ejercicio 5: DMA (Transferencia de datos memoria-memoria).**

*Programa que copia una cadena de caracteres almacenada a partir de la dirección 1000H en otra parte de la memoria, utilizando el CDMA en modo de transferencia por bloque. La cadena original se debe mostrar en la pantalla de comandos antes de la transferencia. Una vez finalizada, se debe visualizar en la pantalla la cadena copiada para verificar el resultado de la operación. Ejecutar el programa en la configuración P1 C3.*

**(a)** *Analizar, minuciosamente, cada línea del programa anterior.*

**(b)** *Explicar qué función cumple cada registro del CDMA e indicar su dirección.*

**(c)** *Describir el significado de los bits del registro CTRL.*

**(d)** *¿Qué diferencia hay entre transferencia de datos por bloque y bajo demanda?*

**(e)** *¿Cómo se le indica al CDMA desde el programa que debe arrancar la transferencia de datos?*

**(f)** *¿Qué le indica el CDMA a la CPU a través de la línea hrq? ¿Qué significa la respuesta que le envía la CPU a través de la línea hlda?*

**(g)** *Explicar, detalladamente, cada paso de la operación de transferencia de un byte desde una celda a otra de la memoria. Verificar que, en esta operación, intervienen el bus de direcciones, el bus de datos y las líneas mrd y mwr.*

**(h)** *¿Qué sucede con los registros RF, CONT y RD del CDMA después de transferido un byte?*

**(i)** *¿Qué evento hace que el CDMA emita una interrupción y a través de qué línea de control lo hace?*

**(j)** *¿Cómo se configura el PIC para atender la interrupción del CDMA?*

**(k)** *¿Qué hace la rutina de interrupción del CDMA del programa anterior?*

**Ejercicio 6: DMA (Transferencia de datos memoria-periférico).**

*Programa que transfiere datos desde la memoria hacia la impresora sin intervención de la CPU, utilizando el CDMA en modo de transferencia bajo demanda.*

**(a)** *Analizar, minuciosamente, cada línea del programa anterior.*

**(b)** *¿Qué debe suceder para que el HAND-SHAKE emita una interrupción al CDMA?*

**(c)** *¿Cómo demanda el periférico, en este caso el HAND-SHAKE, la transferencia de datos desde memoria? ¿A través de qué líneas se comunican con el CDMA ante cada pedido?*

**(d)** *Explicar, detalladamente, cada paso de la operación de transferencia de un byte desde una celda de memoria hacia el HAND-SHAKE y la impresora.*

**(e)** *¿Qué evento hace que el CDMA emita una interrupción al PIC?*

**(f)** *¿Cuándo finaliza la ejecución del LAZO?*

**Ejercicio 7: Configuración del CDMA.**

*Indicar cómo configurar el registro Control del CDMA para las siguientes transferencias:*

**(a)** *Transferencia Memoria Memoria, por robo de ciclo.*

**(b)** *Transferencia Periférico Memoria, por ráfagas.*

**(c)** *Transferencia Memoria Periférico, por robo de ciclo.*