PRÁCTICA 2

Interrupciones por Software.
E/S mediante Consulta de Estado.

Objetivos *Comprender la comunicación entre el microprocesador y los periféricos externos (luces, microconmutadores e impresora). Configurar la interfaz de entrada/salida (PIO) y el el dispositivo de handshaking (HAND-SHAKE). Escribir programas en el lenguaje assembler del simulador VonSim. Ejecutarlos y verificar los resultados, analizando el flujo de información entre los distintos componentes del microprocesador.*

Parte 1: Entrada/Salida con Interrupciones por Software

Resumen de interrupciones por software del VonSim

Instrucción	Operación
INT 0	Detiene el programa.
INT 3	Pausa la ejecución del simulador, y lo pone en modo de *depuración* para poder inspeccionar el programa
INT 6	Lee un carácter de teclado. Para eso, previamente se debe guarda en BX la dirección de memoria en donde se almacenará el carácter leído. Luego de ejecutar INT 6, el carácter leído se encontrará en esa dirección de memoria.
INT 7	Imprime un string en pantalla. Previo a llamar a INT 7, se debe guardar en BX la dirección de comienzo del string a mostrar, y en AL la longitud o cantidad de carácteres del string.

1) **Mostrar mensajes en la pantalla de comandos**. El siguiente programa del lenguaje assembler del simulador VonSim muestra en la pantalla de comandos un mensaje previamente almacenado en memoria de datos, aplicando la interrupción por software INT 7.

```
        ORG 1000H
MSJ     DB  "ARQUITECTURA DE COMPUTADORAS-"
        DB  "FACULTAD DE INFORMATICA-"
        DB  55H
        DB  4EH
        DB  4CH
        DB  50H
FIN     DB  ?

        ORG 2000H
        MOV BX, OFFSET MSJ
        MOV AL, OFFSET FIN-OFFSET MSJ
        INT 7
        INT 0
        END
```

a) Ejecutar en el simulador ¿qué imprime?
b) Con referencia a la interrupción INT 7, ¿qué se almacena en los registros BX y AL?

2) **Lectura de datos desde el teclado**.
El siguiente programa solicita el ingreso de un número (de un dígito) por teclado e inmediatamente lo muestra en la pantalla de comandos, haciendo uso de las interrupciones por software INT 6 e INT 7.

```
        ORG 1000H
MSJ     DB      "INGRESE UN NUMERO:"
FIN     DB      ?

        ORG 1500H
NUM     DB      ?

        ORG 2000H
        MOV BX, OFFSET MSJ
        MOV AL, OFFSET FIN-OFFSET MSJ
        INT 7
        MOV BX, OFFSET NUM
        INT 6
        MOV AL, 1
        INT 7
        MOV CL, NUM
        INT 0
        END
```

a) Con referencia a la interrupción INT 6, ¿qué se almacena en BX?
b) En el programa anterior, ¿qué hace la segunda interrupción INT 7?
c) ¿Qué valor queda almacenado en el registro CL?

3) **Mostrar caracteres individuales**
a) Escribir un programa que muestre en pantalla todos los caracteres del ASCII extendido disponibles en el simulador VonSim, comenzando con el carácter cuyo código es el número 1 y terminando con el 255.
b) Escribir un programa que muestre en pantalla las letras mayúsculas ("A" a la "Z").
c) Escribir un programa que muestre en pantalla los dígitos ("0" al "9")
d) Modificar el ejercicio c) para que cada dígito se muestre en una línea separada.
Pista: El código ASCII del carácter de *nueva línea* es el 10, comúnmente llamado "\n" o LF ("line feed" por sus siglas en inglés y porque se usaba en impresoras donde había que "alimentar" una nueva línea)
e) Modificar el inciso b) para mostrar las mayúsculas junto con las minúsculas intercaladas (AaBb…)

4) **Impresión y conteo.**
A) Escribir una subrutina que reciba un número **X** del 0 al 9 y muestre el mensaje "**X** elefantes se balanceaban…". Utilizar la subrutina para escribir un programa que imprima el mensaje variando **X** desde 0 a 9.
B) Modificar la subrutina anterior para permitir números desde 0 al 99.
Pista: Modificar la subrutina para recibir dos números **X** e **Y**, de forma tal que el número 25 se representa con **X=2** e **Y=5.** Tener en cuenta que números de dos dígitos son representados por dos caracteres.

5) **Contador de vocales**
A. Escribir un programa que lea una cadena de 10 caracteres. Una vez leído completamente, el string se muestra en pantalla, y también se debe mostrar la cantidad de vocales que tiene el string. Para implementar este programa, utilizar una subrutina **leer_str_n** que recibe por parámetro la dirección donde se guardan los caracteres y la cantidad de caracteres a leer. Usar también la subrutina **contar_voc** de la práctica anterior.
B. Idem A) pero ahora la longitud de la cadena no se conoce de antemano; se leen caracteres hasta que llega el caracter "." (punto). Para ello, implementar la subrutina **leer_str_car**, que recibe el carácter de fin por parámetro y la dirección de comienzo donde se almacenarán los caracteres.

6) **Lectura de dígitos con verificación** Escribir un programa que lea un vector de 20 números de un solo dígito, y los almacene en un vector. Luego, se calcula el máximo de los números, y se imprime en pantalla. Los números deben almacenarse como valores entre 0 y 9, y no como los códigos ASCII de los caracteres que les corresponde. Para ello, implementar las siguientes subrutinas:
es_digito: recibe un código ASCII y devuelve 1 si el mismo pertenece a un dígito ('0' al '9) y 0 de lo contrario
leer_digito: lee caracteres con int 6 hasta que se ingresa un carácter dígito. Antes de leer por primera vez, debe

mostrarse el mensaje “Ingresar un dígito del 0 al 9”. Cada vez que se ingresa un carácter no dígito, debe mostrarse el mensaje “Carácter inválido: solo dígitos del 0 al 9”. Cuando se ingresa finalmente un carácter dígito, debe devolverse por registro.

leer_vector: recibe la dirección de comienzo de un vector y un número N, lee N números de un dígito y los guarda en el vector.

max_num: recibe dos números en los registros DL y DH, y devuelve el mayor en el registro DL.

max_vector: recibe la dirección de comienzo de un vector y la cantidad de elementos, y devuelve el valor máximo del mismo.

7) **Contraseña** Escribir un programa que aguarde el ingreso de una contraseña de cuatro caracteres por teclado sin visualizarla en pantalla. En caso de coincidir con una clave predefinida (y guardada en memoria) que muestre el mensaje "Acceso permitido"; caso contrario mostrar el mensaje "Acceso denegado", y volver a pedir que se ingrese una contraseña. Al 5to intento fallido, debe mostrarse el mensaje “Acceso BLOQUEADO” y terminar el programa.

Parte 2: Entrada/Salida con PIO y HANDSHAKE

Resumen de dispositivos básicos de entrada/salida del VonSim:

Dirección	Dispositivo	Nombre	Función
30h	PIO	PA	Puerto multipropósito A
31h	PIO	PB	Puerto multipropósito B
32h	PIO	CA	Configuración del Puerto A (Entrada o salida)
33h	PIO	CB	Configuración del Puerto B (Entrada o salida)
40h	Handshake	Dato	Registro de datos de la interfaz con la impresora
41h	Handshake	Estado	Registro de estado de la interfaz con la impresora

8) **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, solo las primeras y las últimas dos luces deben prenderse, y el resto deben apagarse.

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”. 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.

c) 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.

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 solo aquellas luces donde el valor de las llaves es 1. Entonces, primero se enciende solo 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.

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.

9) 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. 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.
b) Escribir un programa para imprimir el mensaje “ORGANIZACION Y ARQUITECTURA DE COMPUTADORAS” utilizando la impresora a través de la PIO.
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.
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.

10) **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 imprime “INGENIERIA E INFORMATICA” en la impresora a través del HAND-SHAKE. La comunicación se establece por **consulta de estado** (polling). ¿Qué diferencias encuentra con el ejercicio anterior?
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?

11) **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 otros dispositivos. En los siguientes incisos, te proponemos escribir programas que usen 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.

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. Los caracteres se deben enviar al dispositivo uno por uno, esperando 10 segundos entre cada envío.
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 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 FF por P