**Trabajo Práctico N° 2:**

**Interrupciones.**

**Ejercicio 1: Escritura de datos en la pantalla de comandos.**

*Implementar un programa en el lenguaje Assembler del simulador MSX88 que muestre, 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 INFORMÁTICA - ”

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

**Ejercicio 2.**

*Escribir un programa que muestre, en pantalla, todos los caracteres disponibles en el simulador MSX88, comenzando con el caracter cuyo código es el número 01h.*

org 1000h

CHAR DW 01h

org 2000h

mov al, 1

mov bx, offset CHAR

LAZO: int 7

inc CHAR

cmp CHAR, 256

jnz LAZO

int 0

end

**Ejercicio 3.**

*Escribir un programa que muestre, en pantalla, las letras del abecedario, sin espacios, intercalando mayúsculas y minúsculas (AaBB…), sin incluir texto en la memoria de datos del programa. Tener en cuenta que el código de “A” es 41h, el de “a” es 61h y que el resto de los códigos son correlativos según el abecedario.*

org 1000h

MAY DB 41h

MIN DB 61h

org 2000h

mov bx, offset MAY

mov al, 2

LAZO: int 7

inc MAY

inc MIN

cmp MAY, 5Bh

jnz LAZO

int 0

end

**Ejercicio 4: Lectura de datos desde el teclado.**

*Escribir un programa que solicite el ingreso de un número (de un dígito) por teclado e, inmediatamente, lo muestre en la pantalla de comandos, haciendo uso de las interrupciones por software INT 6 e INT 7.*

org 1000h

MSJ DB “INGRESAR UN NÚMERO (de un dígito): ”

FIN DB ?

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

*Responder brevemente:*

**(a)** *Con referencia a la interrupción INT 7, ¿qué se almacena en los registros BX y AL?*

En los registros BX y AL, se almacena la dirección de memoria del caracter inicial de MSJ y el tamaño del mensaje, respectivamente.

**(b)** *Con referencia a la interrupción INT 6, ¿qué se almacena en BX?*

En BX, se almacena la dirección de memoria de NUM.

**(c)** *En el programa anterior, ¿qué hace la segunda interrupción INT 7? ¿qué queda almacenado en el registro CL?*

La segunda interrupción INT 7 lo que hace es imprimir el número (de un dígito) que se ingresó por teclado. Lo que queda almacenado en registro CL es el código ASCII correspondiente al número (como caracter) ingresado por teclado.

**Ejercicio 5.**

*Modificar el programa anterior agregando una subrutina llamada ES\_NUM que verifique si el caracter ingresado es, realmente, un número. De no serlo, el programa debe mostrar el mensaje “CARACTER NO VÁLIDO”. La subrutina debe recibir el código del caracter por referencia desde el programa principal y debe devolver, vía registro, el valor 0FFh, en caso de tratarse de un número, o el valor 00h, en caso contrario. Tener en cuenta que el código del “0” es 30h y el del “9” es 39h.*

org 1000h

MSJ1 DB “INGRESAR UN NÚMERO (de un dígito): ”

FIN1 DB ?

MSJ2 DB “CARACTER NO VÁLIDO”

FIN2 DB ?

NUM DB ?

org 3000h

ES\_NUM: mov ah, 0FFh

cmp byte ptr [bx], 30h

js ERROR

cmp byte ptr [bx], 3Ah

jns ERROR

jmp FIN

ERROR: mov ah, 00h

mov bx, offset MSJ2

mov al, offset FIN2 - offset MSJ2

int 7

FIN: ret

org 2000h

mov bx, offset MSJ1

mov al, offset FIN1 - offset MSJ1

int 7

mov bx, offset NUM

int 6

call ES\_NUM

mov al, 1

int 7

int 0

end

**Ejercicio 6.**

*Escribir un programa que solicite el ingreso de un número (de un dígito) por teclado y muestre, en pantalla, dicho número expresado en letras. Luego, que solicite el ingreso de otro y así sucesivamente. Se debe finalizar la ejecución al ingresarse, en dos vueltas consecutivas, el número cero.*

org 1000h

CERO DB “CERO ”

DB “UNO ”

DB “DOS ”

DB “TRES ”

DB “CUATRO”

DB “CINCO ”

DB “SEIS ”

DB “SIETE ”

DB “OCHO ”

DB “NUEVE ”

MSJ DB “INGRESAR UN NÚMERO (de un dígito): ”

FIN DB ?

NUM DB ?

org 2000h

mov cl, 0

OTRO: mov bx, offset MSJ

mov al, offset FIN - offset MSJ

int 7

mov bx, offset NUM

int 6

cmp NUM, 30h

jnz NO\_CERO

inc cl

jmp SEGUIR

NO\_CERO: mov cl, 0

SEGUIR: mov bx, offset CERO

mov al, 6

LAZO: cmp NUM, 30h

jz IMPRIME

add bx, 6

dec NUM

jmp LAZO

IMPRIMIR: int 7

cmp cl, 2

jnz OTRO

int 0

end

**Ejercicio 7.**

*Escribir un programa que efectúe la suma de dos números (de un dígito cada uno) ingresados por teclado y muestre el resultado en la pantalla de comandos. Recordar que el código de cada caracter ingresado no coincide con el número que representa y que el resultado puede necesitar ser expresado con 2 dígitos.*

org 1000h

MSJ1 DB “INGRESAR UN NÚMERO (de un dígito): ”

FIN1 DB ?

MSJ2 DB 10, “INGRESAR OTRO NÚMERO (de un dígito): ”

FIN2 DB ?

MSJ3 DB 10, “RESULTADO DE LA SUMA DE AMBOS NÚMEROS INGRESADOS: ”

RES\_D DB “0”

RES\_U DB ?

FIN3 DB ?

NUM1 DB ?

NUM2 DB ?

org 2000h

mov bx, offset MSJ1

mov al, offset FIN1 - offset MSJ1

int 7

mov bx, offset NUM1

int 6

mov al, 1

int 7

mov bx, offset MSJ2

mov al, offset FIN2 - offset MSJ2

int 7

mov bx, offset NUM2

int 6

mov al, 1

int 7

mov al, NUM1

sub al, 30h

add al, NUM2

cmp al, 3Ah

js UNIDAD

sub al, 10

inc RES\_D

UNIDAD: mov RES\_U, al

mov bx, offset MSJ3

mov al, offset FIN3 - offset MSJ3

int 7

int 0

end

**Ejercicio 8.**

*Escribir un programa que efectúe la resta de dos números (de un dígito cada uno) ingresados por teclado y muestre el resultado en la pantalla de comandos. Antes de visualizarlo, el programa debe verificar si el resultado es positivo o negativo y anteponer, al valor, el signo correspondiente.*

org 1000h

MSJ1 DB “NUM1: “

FIN1 DB ?

MSJ2 DB 10, “NUM2: ”

FIN2 DB ?

MSJ3 DB 10, “RESTA: ”

SIGNO DB “+”

RES DB ?

FIN3 DB ?

NUM1 DB ?

NUM2 DB ?

org 2000h

mov bx, offset MSJ1

mov al, offset FIN1 - offset MSJ1

int 7

mov bx, offset NUM1

int 6

mov al, 1

int 7

mov bx, offset MSJ2

mov al, offset FIN2 - offset MSJ2

int 7

mov bx, offset NUM2

int 6

mov al, 1

int 7

mov al, NUM1

mov ah, NUM2

cmp al, ah

js NEGATIVO

sub al, ah

mov RES, al

jmp FIN

NEGATIVO: sub ah, al

mov RES, ah

FIN: add RES, 30h

mov bx, offset MSJ3

mov al, offset FIN3 - offset MSJ3

int 7

int 0

end

**Ejercicio 9.**

*Escribir un programa que aguarde el ingreso de una clave 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 el mensaje “Acceso denegado”.*

org 1000h

CLAVE\_PRE DB “1234”

MSJ1 DB “INGRESAR UNA CLAVE (de cuatro caracteres): ”

FIN1 DB ?

MSJ2 DB 10, “ACCESO PERMITIDO”

FIN2 DB ?

MSJ3 DB 10, “ACCESO DENEGADO”

FIN3 DB ?

CLAVE DB ?,?,?,?

org 2000h

mov bx, offset MSJ1

mov al, offset FIN1 - offset MSJ1

int 7

mov bx, offset CLAVE

mov ah, 4

LAZO1: int 6

mov al, 1

int 7

inc bx

dec ah

cmp ah, 0

jnz LAZO1

mov al, CLAVE

mov cx, 0

LAZO2: mov bx, offset CLAVE\_PRE

add bx, cx

mov dl, [bx]

mov bx, offset CLAVE

add bx, cx

mov dh, [bx]

cmp dl, dh

jnz DENEGADO

inc cx

cmp cx, 4

jnz LAZO2

mov bx, offset MSJ2

mov al, offset FIN2 - offset MSJ2

int 7

jmp FIN

DENEGADO: mov bx, offset MSJ3

mov al, offset FIN3 - offset MSJ3

int 7

FIN: int 0

end

**Ejercicio 10: Interrupción por hardware (Tecla F10).**

*Escribir un programa que, mientras ejecuta un lazo infinito, cuente el número de veces que se presiona la tecla F10 y acumule este valor en el registro DX.*

*PIC EQU 20H*

*EOI EQU 20H*

*N\_F10 EQU 10*

*ORG 40*

*IP\_F10 DW RUT\_F10*

*ORG 3000H*

*RUT\_F10: PUSH AX*

*INC DX*

*MOV AL, EOI*

*OUT EOI, AL*

*POP AX*

*IRET*

*ORG 2000H*

*CLI*

*MOV AL, 0FEH*

*OUT PIC+1, AL*

*MOV AL, N\_F10*

*OUT PIC+4, AL*

*MOV DX, 0*

*STI*

*LAZO: JMP LAZO*

*END*

*Explicar detalladamente:*

**(a)** *La función de los registros del PIC: ISR, IRR, IMR, INT0-INT7, EOI. Indicar la dirección de cada uno.*

La función de los registros del PIC es:

* ISR (23h): Indicar la interrupciones en ejecución.
* IRR (22h): Indicar la interrupciones pedidas.
* IMR (21h): Indicar la interrupciones habilitadas.
* INT0-INT7 (24h-31h): Indicar el ID de interrupción de cada dispositivo.
* EOI (20h): Indicar la finalización de la interrupción.

**(b)** *Cuáles de estos registros son programables y cómo trabaja la instrucción OUT.*

De estos registros, son programables el IMR, INT0-INT7 y el EOI. La instrucción OUT trabaja moviendo contenido del registro AL al PIC.

**(c)** *Qué hacen y para qué se usan las instrucciones CLI y STI.*

Las instrucciones CLI y STI se usan para deshabilitar y habilitar interrupciones, respectivamente.

**Ejercicio 11.**

*Escribir un programa que permita seleccionar una letra del abecedario al azar. El código de la letra debe generarse en un registro que incremente su valor desde el código de A hasta el de Z continuamente. La letra debe quedar seleccionada al presionarse la tecla F10 y debe mostrarse, de inmediato, en la pantalla de comandos.*

EOI EQU 20h

IMR EQU 21h

INT0 EQU 24h

N\_F10 EQU 10

org 40

IP\_F10 DW RUT\_F10

org 1000h

LETRA DB ?

org 3000h

RUT\_F10: push ax

push bx

inc cl

mov LETRA, ah

mov bx, offset LETRA

mov al, 1

int 7

mov al, EOI

out EOI, al

pop bx

pop ax

iret

org 4000h

ABCDE: mov ah, 65

LAZO: cmp cl, 1

jz FIN

inc ah

cmp ah, 90

jnz LAZO

jmp ABCDE

FIN: ret

org 2000h

cli

mov al, 0FEh

out IMR, al

mov al, N\_F10

out INT0, al

mov cl, 0

sti

call ABCDE

int 0

end

**Ejercicio 12: Interrupción por hardware (Timer).**

*Implementar, a través de un programa, un reloj segundero que muestre, en pantalla, los segundos transcurridos (00-59 seg.) desde el inicio de la ejecución.*

*TIMER EQU 10H*

*PIC EQU 20H*

*EOI EQU 20H*

*N\_CLK EQU 10*

*ORG 40*

*IP\_CLK DW RUT\_CLK*

*ORG 1000H*

*SEG DB 30H*

*DB 30H*

*FIN DB ?*

*ORG 3000H*

*RUT\_CLK: PUSH AX*

*INC SEG+1*

*CMP SEG+1, 3AH*

*JNZ RESET*

*MOV SEG+1, 30H*

*INC SEG*

*CMP SEG, 36H*

*JNZ RESET*

*MOV SEG, 30H*

*RESET: INT 7*

*MOV AL, 0*

*OUT TIMER, AL*

*MOV AL, EOI*

*OUT PIC, AL*

*POP AX*

*IRET*

*ORG 2000H*

*CLI*

*MOV AL, 0FDH*

*OUT PIC+1, AL*

*MOV AL, N\_CLK*

*OUT PIC+5, AL*

*MOV AL, 10*

*OUT TIMER+1, AL*

*MOV AL, 0*

*OUT TIMER, AL*

*MOV BX, OFFSET SEG*

*MOV AL, OFFSET FIN - OFFSET SEG*

*STI*

*LAZO: JMP LAZO*

*END*

*Explicar detalladamente:*

**(a)** *Cómo funciona el TIMER y cuándo emite una interrupción a la CPU.*

El Timer es otro dispositivo de ES como el F10. Se utiliza como un reloj despertador para la CPU. Se configura para contar una cantidad determinada de segundos y, cuando finaliza la cuenta, emite una interrupción. El Timer tiene dos registros, CONT (registro contador) y COMP (registro de comparación), con direcciones de la memoria de ES 10h y 11h, respectivamente.

**(b)** *La función que cumplen sus registros, la dirección de cada uno y cómo se programan.*

La función de los registros del Timer es:

* CONT (10h): Se incrementa, automáticamente, una vez por segundo, para contar tiempo transcurrido.
* COMP (11h): Contiene el tiempo límite del Timer. Cuando CONT vale igual que COMP, se dispara la interrupción.

**Ejercicio 13.**

*Modificar el programa anterior para que también cuente minutos (00:00-59:59), pero que actualice la visualización en pantalla cada 10 segundos.*

CONT EQU 10h

COMP EQU 11h

EOI EQU 20h

IMR EQU 21h

INT1 EQU 25h

N\_CLK EQU 10

org 40

IP\_CLK DW RUT\_CLK

org 1000h

MIN\_D DB 30h

MIN\_U DB 30h, 58

SEG\_D DB 30h

SEG\_U DB 30h, 32

FIN DB ?

org 3000h

RUT\_CLK: push ax

inc SEG\_D

cmp SEG\_D, 36h

jnz IMPRIMIR

mov SEG\_D, 30h

inc MIN\_U

cmp MIN\_U, 3Ah

jnz IMPRIMIR

mov MIN\_U, 30h

inc MIN\_D

cmp MIN\_D, 36h

jnz IMPRIMIR

mov MIN\_D, 30h

IMPRIMIR: int 7

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 bx, offset MIN\_D

mov al, offset FIN - offset MIN\_D

sti

LAZO: jmp LAZO

int 0

end

**Ejercicio 14.**

*Implementar un reloj similar al utilizado en los partidos de básquet, que arranque y detenga su marcha al presionar sucesivas veces la tecla F10 y que finalice el conteo al alcanzar los 30 segundos.*

CONT EQU 10h

COMP EQU 11h

EOI EQU 20h

IMR EQU 21h

INT0 EQU 24h

INT1 EQU 25h

N\_CLK EQU 10

N\_F10 EQU 20

org 40

IP\_CLK DW RUT\_CLK

org 80

IP\_F10 DW RUT\_F10

org 1000h

SEG\_D DB 30h

SEG\_U DB 30h, 32

FIN DB ?

org 3000h

RUT\_CLK: push ax

inc SEG\_U

cmp SEG\_U, 3Ah

jnz IMPRIMIR

mov SEG\_U, 30h

inc SEG\_D

cmp SEG\_D, 33h

jnz IMPRIMIR

mov cl, 1

mov al, 0FFh

out IMR, al

IMPRIMIR: int 7

mov al, 0

out CONT, al

mov al, EOI

out EOI, al

pop ax

iret

org 4000h

RUT\_F10: push ax

in al, IMR

xor al, 00000010b

out IMR, al

mov al, EOI

out EOI, al

pop ax

iret

org 2000h

cli

mov al, 0FCh

out IMR, al

mov al, N\_F10

out INT0, al

mov al, N\_CLK

out INT1, al

mov al, 0

out CONT, al

mov al, 10

out COMP, al

mov bx, offset SEG\_D

mov al, offset FIN - offset SEG\_D

mov cl, 0

sti

LAZO: cmp cl, 0

jz LAZO

int 0

end

**Ejercicio 15.**

*Escribir un programa que implemente un conteo regresivo a partir de un valor ingresado desde el teclado. El conteo debe comenzar al presionarse la tecla F10. El tiempo transcurrido debe mostrarse en pantalla, actualizándose el valor cada segundo.*

CONT EQU 10h

COMP EQU 11h

EOI EQU 20h

IMR EQU 21h

INT0 EQU 24h

INT1 EQU 25h

N\_CLK EQU 10

N\_F10 EQU 20

org 40

IP\_CLK DW RUT\_CLK

org 80

IP\_F10 DW RUT\_F10

org 1000h

NUM DB ?, 32

FIN\_NUM DB ?

org 3000h

RUT\_CLK: push ax

int 7

dec cl

cmp cl, 30h

jns SEGUIR

mov al, 0FFh

out IMR, al

jmp FIN

SEGUIR: mov NUM, cl

mov al, 0

out CONT, al

FIN: mov al, EOI

out EOI, al

pop ax

iret

org 4000h

RUT\_F10: push ax

in al, IMR

xor al, 00000010b

out IMR, al

mov al, EOI

out EOI, al

pop ax

iret

org 2000h

cli

mov al, 0FEh

out IMR, al

mov al, N\_F10

out INT0, al

mov al, N\_CLK

out INT1, al

mov al, 0

out CONT, al

mov al, 10

out COMP, al

mov bx, offset NUM

int 6

mov bx, offset NUM

mov al, offset FIN\_NUM - offset NUM

mov cl, NUM

sti

LAZO: cmp cl, 30h

jns LAZO

int 0

end