PRÁCTICA 2

Interrupciones

Objetivos: Comprender la utilidad de las interrupciones por software y por hardware y el funcionamiento del Controlador de Interrupciones Programable (PIC). Escribir programas en el lenguaje assembler del simulador MSX88. Ejecutarlos y verificar los resultados, analizando el flujo de información entre los distintos componentes del microprocesador.

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 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
```

2)ORG 1000H

MSJ DB 01h
FIN DB ?
ORG 2000H
MOV BX, offset MSJ
mov al, 1
INT 7
sigo: add byte ptr [bx],1
mov al, 1
int 7
cmp byte ptr [bx], 0FFH
jnz sigo

INT 0
END

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.

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.

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      "INGRESE UN NUMERO
:" FIN    DB       ?

        ORG 1500
     H 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 bx se almacena la direccion de memoria de la variable NUM y en Al se guarda la cantidad de datos a imprimir

b) Con referencia a la interrupción INT 6, ¿qué se almacena en BX? Se almacena la direccion donde se va a guardar e l dato ingresado

c) En el programa anterior, ¿qué hace la segunda interrupción INT 7? ¿qué queda almacenado en el registro CL?La segud a interrupcion 7 imprime el pantalla el numero ingresado y en CL queda guardado el codigo ascii del digito ingresado

5) Modificar el programa anterior agregando una subrutina llamada ES_NUM que verifique si el caracter ingresado es

```
ORG 1000H
MSJ   DB      "caracter no valido"
FIN   DB      ?
ORG 1500H
NUM   DB      ?

ORG 2000H


MOV BX, OFFSET NUM
INT 6
cmp byte ptr [bx], 30h
js invalido
cmp  byte ptr [bx] ,39h
jns invalido
jmp valido
invalido: MOV BX, OFFSET MSJ
MOV AL, OFFSET FIN-OFFSET MSJ
INT 7
jmp fin
valido: MOV AL, 1
INT 7
MOV CL, NUM
fin: INT 0
END
```

```
ORG 1000H
CERO DB "CERO " ; Todos los nombres tienen 6 caracteres
para
DB "UNO " ; facilitar posicionarnos al imprimir el nombre del
numero
DB "DOS "
DB "TRES "
DB "CUATRO"
DB "CINCO "
DB "SEIS "
DB "SIETE "
DB "OCHO "
DB "NUEVE "
MSJ DB "INGRESE UN NUMERO:"
FIN DB ?
ORG 1500H
NUM DB ?
ORG 2000H
MOV CL, 0 ; Contador de veces que ingresa el valor 0 de
forma consecutiva
OTRO: MOV BX, OFFSET MSJ
MOV AL, OFFSET FIN-OFFSET MSJ
INT 7 ; Imprimo mensaje en pantalla pidiendo el ingreso de
un numero
MOV BX, OFFSET NUM
INT 6 ; Leo un caracter y queda guardado en NUM
CMP NUM, 30H
JNZ NO_CERO
INC CL ; Si vino un valor 0, incremento el contador
JMP SEGUIR
NO_CERO:MOV CL, 0 ; Como no vino un valor 0,
reinicializo CL
SEGUIR: MOV BX, OFFSET CERO ; La direccion BASE
sera la del primer mensaje ("CERO")
; Luego se posicionara al inicio del mensaje adecuado
MOV AL, 6 ; Se va a imprimir 6 caracteres, todos tienen el
mismo largo
LOOP: CMP NUM, 30H
JZ IMPRIME ; Si es el valor adecuado, imprimo en pantalla
el nombre del numero
ADD BX, 6 ; Si no es el valor adecuado, me posiciono en el
siguiente nombre
DEC NUM ; Al llegar NUM a 0 estara posicionado en el
nombre que corresponde
JMP LOOP
IMPRIME:INT 7
CMP CL, 2
JNZ OTRO ; Si no se ingreso dos veces seguidas el numero
0, sigue procesando
INT 0 ; Se ingreso dos veces seguidas 0, por lo que el
programa termina
END
```

realmente un número. De no serlo, el programa debe mostrar el mensaje "CARACTER NO VALIDO". 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.

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.

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.

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.

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".

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 2000H
          CLI
          MOV AL, 0FEH
          OUT PIC+1, AL ; PIC: registro IMR
          MOV AL, N_F10
          OUT PIC+4, AL ; PIC: registro INT0
          MOV DX, 0
          STI
LAZO:     JMP LAZO

          ORG 3000H
RUT_F10:  PUSH AX
          INC DX
          MOV AL, EOI
          OUT EOI, AL  ; PIC: registro EOI
          POP AX
          IRET

          END
```

```
ORG 1000H
MSJ  DB    "caracter no valido"
FIN  DB    ?
ORG 1500H
NUM  DB    ?
NUM2 DB    ?

ORG 2000H


MOV BX, OFFSET NUM
INT 6
mov ax, word ptr [bx]
MOV BX, OFFSET NUM2
INT 6
sub ax, word ptr [bx]
js negativo
jmp positivo
negativo: mov bx, ax
add al,1
int 7
positivo: mov bx, ax
add al,1
int 7

fin: INT 0
END
```

Explicar detalladamente:

a) La función de los registros del PIC: ISR, IRR, IMR, INT0-INT7, EOI. Indicar la dirección de cada uno.
b) Cuáles de estos registros son programables y cómo trabaja la instrucción OUT.
c) Qué hacen y para qué se usan las instrucciones CLI y STI.

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.

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.

8)org 1000h

```
msj db  "ingrese un numero"
fin db   ?
msj2 db  "ingrese otro numero"
fin2 db   ?
num1 db   ?
num2 db   ?
resul db   0
nega db   2dh

org 2000h
mov ax, 0000h
mov bx, offset msj
mov al, offset fin - offset msj
int 7
mov bx, offset num1
int 6
mov bx, offset msj2
mov al, offset fin2 - offset msj2
int 7
mov bx, offset num2
int 6
mov al, num1
sub al, num2
mov cl, al
mov ah, al
and ah, 10000000b
jnz negativo
add al, 30h
jmp positivo
negativo:  mov ah,al
mov bx,offset nega
mov al, 1
int 7
mov cl, 255
sub cl, ah
add cl, 1
mov al, cl
add al, 30h
positivo: mov resul, al
mov bx, offset resul
mov al, 1
int 7
int 0
end
```

```
ORG 1000H
MSJ DB "INGRESE UN NUMERO:"
FIN DB ?
ORG 1500H
NUM1 DB ?
NUM2 DB ?
RES_D DB "0" ; Decena del resultado.
RES_U DB ? ; Unidad del resultado.
; Por ej. si se suma "6" + "7", la decena del resultado sera "1" y la unidad "3"
ORG 2000H
MOV BX, OFFSET MSJ
MOV AL, OFFSET FIN-OFFSET MSJ
INT 7 ; Imprimo mensaje en pantalla pidiendo el ingreso de un numero
MOV BX, OFFSET NUM1
INT 6 ; Leo un caracter y queda guardado en NUM1
MOV BX, OFFSET MSJ
INT 7 ; Imprimo mensaje en pantalla pidiendo el ingreso de un numero
MOV BX, OFFSET NUM2
INT 6 ; Leo un caracter y queda guardado en NUM2
MOV AL, NUM2 ; Copio el segundo caracter leido en AL
SUB AL, 30H ; Le resto 30H, para quedarme con el valor del numero
ADD AL, NUM1 ; Le sumo el primer caracter leido
CMP AL, 3AH ; Si quedo un valor entre 30H y 39H, la suma no supero 9
; Entonces la unidad esta lista
; Y la decena tambien, ya que comienza con valor "0"
JS NUM_OK
SUB AL, 10 ; Si quedo un valor mayor a 39H
; entonces se le resta 10 para obtener la unidad
INC RES_D ; Se suma 1 a la decena (pasa de ser el caracter "0" a "1"
NUM_OK: MOV RES_U, AL ; Copio el valor de la unidad a RES_U
MOV BX, OFFSET RES_D ; A partir de la dir. de RES_D, se imprime 2 caracteres
MOV AL, 2
INT 7
INT 0
END
```

```
org  1000h

msj db  "ingrese un numero"
fin db   ?
msj2 db  "ingrese otro numero"
fin2 db   ?
num1 db   ?
num2 db   ?
resul db   ?
nega db   2DH

org 2000h
mov ax, 0000h
mov bx, offset msj
mov al, offset fin - offset msj
int 7
mov bx, offset num1
int 6
mov bx, offset msj2
mov al, offset fin2 - offset msj2
int 7
mov bx, offset num2
int 6
mov al, num1
sub al, num2
mov ah, al
and ah, 10000000b
;MOV AH, 2BH
js negativo
add al, 30h
jmp positivo
negativo: mov bx,offset nega
mov al, 1
int 7
not al
;add al, 30h
mov resul, al
add resul,1
mov resul, al
mov bx, offset resul
mov al, 1
int 7
jmp FIN
positivo: mov resul, al
add resul,1
mov resul, al
mov bx, offset resul
mov al, 1
int 7
FIN: int 0
end
```

```
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       ; PIC: registro IMR
          MOV AL, N_CLK
          OUT PIC+5, AL       ; PIC: registro INT1
          MOV AL, 1
          OUT TIMER+1, AL     ; TIMER: registro COMP
          MOV AL, 0
          OUT TIMER, AL       ; TIMER: registro CONT
          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.
b) La función que cumplen sus registros, la dirección de cada uno y cómo se programan.

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

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.

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.

Nota: Los ejercicios marcados poseen una resolución propuesta

10)

```
EOI      EQU   20H
PIC      EQU   20H
IMR      EQU   21H
INT0     EQU   24H
F10    EQU    8
ORG  1000H
MOSTRAR    DB     "A"

ORG  32
DW    INT_F10
ORG  3000H
INT_F10:     INT 7
MOV AL, EOI
OUT PIC, AL
IRET

ORG  2000H
CLI
MOV AH, MOSTRAR
MOV AL, 0FEH
OUT IMR, AL
MOV AL, F10
OUT INT0, AL
MOV BX, OFFSET MOSTRAR
MOV AL,1
STI
SIGO:   INC AH
MOV MOSTRAR, AH
CMP AH, 5AH
JNZ  SIGO
HLT
END
```

12)

```
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
SEG1     DB 30H
ESPACIO  DB 13
FIN      DB ?
ORG 3000H
RUT_CLK: PUSH AX
INC SEG1
CMP SEG1, 3AH
JNZ RESET
MOV SEG1, 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 ; PIC: registro
MOV AL, N_CLK
OUT PIC+5, AL       ; PIC: registro
MOV AL, 1
OUT TIMER+1, AL     ; TIMER: registro
MOV AL, 0
OUT TIMER, AL       ; TIMER: registro
MOV AL, OFFSET FIN-OFFSET SEG
MOV BX, OFFSET SEG
STI
LAZO:    JMP LAZO
END
```