

# PRACTICA 1

## Subrutinas y pasaje de parámetros

*Objetivos: Comprender la utilidad de las subrutinas y la comunicación con el programa principal a través de una pila. Escribir programas en el lenguaje assembly del simulador MSX88. Ejecutarlos y verificar los resultados, analizando el flujo de información entre los distintos componentes del sistema.*

- 1) \* **Repaso de uso de la pila** Si el registro SP vale 8000h al comenzar el programa, indicar el valor del registro SP **luego de ejecutar** cada una de las instrucciones de la tabla, **en el orden en que aparecen**. Indicar, de la misma forma, los valores de los registros AX y BX.

	Instrucción	Valor del registro SP	AX	BX
1	mov ax,5			
2	mov bx,3			
3	push ax			
4	push ax			
5	push bx			
6	pop bx			
7	pop bx			
8	pop ax			

- 2) \* **Llamadas a subrutinas y la pila** Si el registro SP vale 8000h al comenzar el programa, indicar el valor del registro SP **luego** de ejecutar cada instrucción. Considerar que el programa comienza a ejecutarse con el IP en la dirección 2000h, es decir que la primera instrucción que se ejecuta es la de la línea 5 (push ax).

Nota: Las sentencias ORG y END no son instrucciones sino indicaciones al compilador, por lo tanto no se ejecutan.

#	Instrucción	Valor del registro SP
1	org 3000h	-----
2	rutina: mov bx,3	
3	ret	
4	org 2000h	-----
5	push ax	
6	call rutina	
7	pop bx	
8	hlt	
9	end	-----

- 3) \* **Llamadas a subrutinas y dirección de retorno**

- Si el registro SP vale 8000h al comenzar el programa, **indicar el contenido de la pila luego** de ejecutar cada instrucción. Si el contenido es desconocido/basura, indicarlo con el símbolo ?. Considerar que el programa comienza a ejecutarse con el IP en la dirección 2000h, es decir que la primera instrucción que se ejecuta es la

de la línea 5 (call rut). Se provee la ubicación de las instrucciones en memoria, para poder determinar la dirección de retorno de la rutina.

Nota: Las sentencias ORG y END no son instrucciones sino indicaciones al compilador, por lo tanto no se ejecutan ni tienen ubicación en memoria.

2. Explicar detalladamente:

- Las acciones que tienen lugar al ejecutarse la instrucción call rut
- Las acciones que tienen lugar al ejecutarse la instrucción ret

```

org 3000h
rut: mov bx,3      ; Dirección 3000h
    ret           ; Dirección 3002h

```

```

org 2000h
call rut      ; Dirección 2000h
add cx,5      ; Dirección 2002h
call rut      ; Dirección 2004h
hlt           ; Dirección 2006h
end

```

4) \* **Tipos de Pasajes de Parámetros** Indicar con un tilde, para los siguientes ejemplos, si el pasaje del parámetro es por registro o pila, y por valor o referencia;

	Código	Registro	Pila	Valor	Referencia
a)	mov ax,5 call subrutina				
b)	mov dx, offset A call subrutina				
c)	mov bx, 5 push bx call subrutina pop bx				
d)	mov cx, offset A push cx call subrutina pop cx				
e)	mov dl, 5 call subrutina				
f)	call subrutina mov A, dx				

5) **Cálculo de A+B-C. Pasaje de parámetros a través de registros.**

En este ejercicio, programarás tus primeras subrutinas. Las subrutinas recibirán tres parámetros A, B y C, y realizarán un cálculo muy simple, A+B-C, cuyo resultado deben retornar. Si bien en general no tendría sentido escribir una subrutina para una cuenta tan simple que puede implementarse con dos instrucciones, esta simplificación permite concentrarse en los aspectos del pasaje de parámetros.

- Escribir un programa que dados los valores etiquetados como A, B y C y almacenados en la memoria de datos, calcule A+B-C y guarde el resultado en la memoria con etiqueta D, **sin utilizar subrutinas**.
- Escribir un programa como en a) pero ahora el cálculo y el almacenamiento del resultado debe realizarse en una subrutina llamada calculo, sin recibir ni devolver parámetros, es decir, utilizando A, B, C y D como variables globales. Si bien esta técnica no está recomendada, en ejercicio sirve para ver sus diferencias con el uso de parámetros.
- Volver a escribir el programa, pero ahora con una subrutina que reciba A, B y C por valor a través de los registros AX, BX y CX, calcule AX+BX-CX, y devuelva el resultado por valor en el registro DX. El programa principal debe llamar a la subrutina y luego guardar el resultado en la memoria con etiqueta D
- Si tuviera que realizar el cálculo dos veces con números distintos, por ejemplo, unos guardados en variables A1, B1, C1 y otros guardados en variables A2, B2, C2, ¿podrían reutilizarse las subrutinas del inciso b) sin modificarse? ¿y las del inciso c)?

```

;Memoria de Datos
    org 1000h
    A   DW 5h
    B   DW 6h
    C   DW 2h
    D   DW ?
;Memoria de programa
    org 2000h
    ...    ; COMPLETAR
end

```

#### 6) \* Multiplicación de números sin signo. Pasaje de parámetros a través de registros.

El simulador no posee una instrucción para multiplicar números. Escribir un programa para multiplicar los números NUM1 y NUM2, y guardar el resultado en la variable RES

- Sin hacer llamados a subrutinas, resolviendo el problema desde el programa principal;
- Llamando a una subrutina MUL para efectuar la operación, pasando los parámetros por **valor** desde el programa principal a través de **registros** y devolviendo el resultado a través de un **registro** por **valor**.
- Llamando a una subrutina MUL, pasando los parámetros por **referencia** desde el programa principal a través de registros, y devolviendo el resultado a través de un **registro** por **valor**.

#### 7) Multiplicación de números sin signo. Pasaje de parámetros a través de Pila.

El programa de abajo utiliza una subrutina para multiplicar dos números, pasando los parámetros por valor para NUM1 y NUM2, y por referencia (RES), en ambos casos a través de la **pila**. Analizar su contenido y contestar.

- ¿Cuál es el modo de direccionamiento de la instrucción MOV AX, [BX]? ¿Qué se copia en el registro AX en este caso?
- ¿Qué función cumple el registro temporal **ri** que aparece al ejecutarse una instrucción como la anterior?
- ¿Qué se guarda en AX al ejecutarse MOV AX, OFFSET RES?
- ¿Cómo se pasa la variable RES a la pila, por valor o por referencia? ¿Qué ventaja tiene esto?
- ¿Cómo trabajan las instrucciones PUSH y POP?

#### Observaciones:

- Los contenidos de los registros AX, BX, CX y DX antes y después de ejecutarse la subrutina son iguales, dado que al comienzo se almacenan en la pila para poder utilizarlos sin perder la información que contenían antes del llamado. Al finalizar la subrutina, los contenidos de estos registros son restablecidos desde la pila.
- El programa sólo puede aplicarse al producto de dos números mayores que cero.

<pre> MUL:  ORG 3000H ; Subrutina MUL       PUSH BX   ; preservar registros       PUSH CX       PUSH AX       PUSH DX       MOV BX, SP       ADD BX, 12 ; corrección por el IP(2),                   ; RES(2) y los 4 registros(8)       MOV CX, [BX] ; cx = num2       ADD BX, 2    ; bx apunta a num1       MOV AX, [BX] ; ax = num1       SUB BX, 4    ; bx apunta a la dir de                   ; resultado       MOV BX, [BX] ; guardo       MOV DX, 0 SUMA: ADD DX, AX       DEC CX       JNZ SUMA       MOV [BX], DX ; guardar resultado       POP DX      ; restaurar registros       POP AX       POP CX       POP BX       RET </pre>	<pre>       ORG 1000H ; Memoria de datos       NUM1 DW 5H       NUM2 DW 3H       RES  DW ?        ORG 2000H; Prog principal       ; parámetros       MOV AX, NUM1       PUSH AX       MOV AX, NUM2       PUSH AX       MOV AX, OFFSET RES       PUSH AX       CALL MUL       ; desapilar parámetros       POP AX       POP AX       POP AX       HLT       END </pre>
---	---

Analizar el siguiente diagrama de la pila y verificarlo con el simulador:

DIRECCIÓN DE MEMORIA	CONTENIDO		
7FF0H	DL	7FF8H	IP RET. L
	DH		IP RET. H
7FF2H	AL	7FFAH	DIR. RES L
	AH		DIR. RES H
7FF4H	CL	7FFCH	NUM2 L
	CH		NUM2 H
7FF6H	BL	7FFE H	NUM1 L
	BH		NUM1 H
		8000H	—

### 8) Subrutinas para realizar operaciones con cadenas de caracteres

- Escribir una subrutina LONGITUD que cuente el número de caracteres de una cadena de caracteres terminada en cero (00H) almacenada en la memoria. La cadena se pasa a la subrutina por referencia vía registro, y el resultado se retorna por valor también a través de un registro.  
Ejemplo: la longitud de 'abcd'00h es 4 (el 00h final no cuenta)
- Escribir una subrutina CONTAR\_MIN que cuente el número de letras minúsculas de la 'a' a la 'z' de una cadena de caracteres terminada en cero almacenada en la memoria. La cadena se pasa a la subrutina por referencia vía registro, y el resultado se retorna por valor también a través de un registro.  
Ejemplo: CONTAR\_MIN de 'aBcDE1#!' debe retornar 2.
- \* Escriba la subrutina ES\_VOCAL, que determina si un carácter es vocal o no, ya sea mayúscula o minúscula. La rutina debe recibir el carácter por valor vía registro, y debe retornar, también vía registro, el valor 0FFH si el carácter es una vocal, o 00H en caso contrario.  
Ejemplos: ES\_VOCAL de 'a' o 'A' debe retornar 0FFh y ES\_VOCAL de 'b' o de '4' debe retornar 00h
- \* Usando la subrutina anterior escribir la subrutina CONTAR\_VOC, que recibe una cadena terminada en cero por referencia a través de un registro, y devuelve, en un registro, la cantidad de vocales que tiene esa cadena.  
Ejemplo: CONTAR\_VOC de 'contar1#!' debe retornar 2
- Escriba la subrutina CONTAR\_CAR que cuenta la cantidad de veces que aparece un carácter dado en una cadena terminada en cero. El carácter a buscar se debe pasar por valor mientras que la cadena a analizar por referencia, ambos a través de la pila.  
Ejemplo: CONTAR\_CAR de 'abbcd!' y 'b' debe retornar 2, mientras que CONTAR\_CAR de 'abbcd!' y 'z' debe retornar 0.
- Escriba la subrutina REEMPLAZAR\_CAR que reciba dos caracteres (ORIGINAL y REEMPLAZO) por valor a través de la pila, y una cadena terminada en cero también a través de la pila. La subrutina debe reemplazar el carácter ORIGINAL por el carácter REEMPLAZO.

### 9) Subrutinas para realizar rotaciones

- Escribir una subrutina ROTARIZQ que haga **una** rotación hacia la izquierda de 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, sino que se modifica directamente la memoria.

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):

$$\begin{array}{r}
 01101011 \\
 + \quad 01101011 \\
 \hline
 11010110 \text{ (CARRY=0)}
 \end{array}$$

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.

$$\begin{array}{r}
 10010100 \\
 + \quad 10010100 \\
 \hline
 00101000 \text{ (CARRY=1)} \\
 + \quad 00000001 \\
 \hline
 00101001
 \end{array}$$

b) 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**.

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

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.

d) Escriba la subrutina ROTARDER del ejercicio anterior, pero sin usar la subrutina ROTARIZ. Compare qué ventajas tiene cada una de las soluciones.

**10) SWAP** Escribir una subrutina SWAP que intercambie dos datos de 16 bits almacenados en memoria. Los parámetros deben ser pasados por referencia desde el programa principal a través de la pila.

Para hacer este ejercicio, tener en cuenta que los parámetros que se pasan por la pila son las *direcciones* de memoria, por lo tanto para acceder a los datos a intercambiar se requieren accesos indirectos, además de los que ya se deben realizar para acceder a los parámetros de la pila.

### 11) Subrutinas de cálculo

a) Escriba la subrutina DIV que calcule el resultado de la división entre 2 números positivos. Dichos números deben pasarse por valor desde el programa principal a la subrutina a través de la pila. El resultado debe devolverse también a través de la pila por valor.

b) \* Escriba la subrutina RESTO que calcule el resto de la división entre 2 números positivos. Dichos números deben pasarse por valor desde el programa principal a la subrutina a través de registros. El resultado debe devolverse también a través de un registro por referencia.

c) Escribir un programa que calcule la suma de dos números de 32 bits almacenados en la memoria sin hacer llamados a subrutinas, resolviendo el problema desde el programa principal.

d) Escribir un programa que calcule la suma de dos números de 32 bits almacenados en la memoria llamando a una subrutina SUM32, que reciba los parámetros de entrada por valor a través de la pila, y devuelva el resultado también por referencia a través de la pila.

12) Analizar el funcionamiento de la siguiente subrutina y su programa principal:

ORG 3000H	ORG 2000H
MUL: CMP AX, 0	MOV CX, 0
JZ FIN	MOV AX, 3
ADD CX, AX	CALL MUL
DEC AX	HLT
CALL MUL	END
FIN: RET	

a) ¿Qué hace la subrutina?

b) ¿Cuál será el valor final de CX?

c) Dibujar las posiciones de memoria de la pila, anotando qué valores va tomando

d) ¿Cuál será la limitación para determinar el valor más grande que se le puede pasar a la subrutina a través de AX?

**Nota:** Los ejercicios marcados con \* tienen una solución propuesta.

# 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) 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 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

```

Responder brevemente:

- Con referencia a la interrupción INT 7, ¿qué se almacena en los registros BX y AL?
- Con referencia a la interrupción INT 6, ¿qué se almacena en BX?
- En el programa anterior, ¿qué hace la segunda interrupción INT 7? ¿qué queda almacenado en el registro CL?

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 “CHARACTER NO VALIDO”. La subrutina debe recibir el código del carácter 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 carácter 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 INTO
        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
```

Explicar detalladamente:

- La función de los registros del PIC: ISR, IRR, IMR, INT0-INT7, EOI. Indicar la dirección de cada uno.
- Cuáles de estos registros son programables y cómo trabaja la instrucción OUT.
- 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.

```

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



# PRACTICA 3

## Entrada/Salida

*Objetivos: Comprender la comunicación entre el microprocesador y los periféricos externos (luces, microconmutadores e impresora). Configurar la interfaz de entrada/salida (PIO), el dispositivo de handshaking (HAND-SHAKE) y el dispositivo de comunicación serie (USART) para el intercambio de información entre el microprocesador y el mundo exterior. Escribir programas en el lenguaje assembly del simulador VonSim y el MSX88. Ejecutarlos y verificar los resultados, analizando el flujo de información entre los distintos componentes del sistema.*

**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, 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.

**2) 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.

**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 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 2b?

- 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?
  - 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.
  - 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 encuentra con el ejercicio 2c?
  - e) Idem d), pero ahora utilizar el HAND-SHAKE en modo **interrupciones**.
- 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**.

**Nota: Los ejercicios marcados con \* tienen una solución propuesta.**

## Anexo DMA

*Objetivos: Comprender el funcionamiento del Controlador de Acceso Directo a Memoria (CDMA) incluido en el simulador MSX88. Configurarlos para la transferencia de datos memoria-memoria y memoria-periférico en modo bloque y bajo demanda. Escribir programas en el lenguaje assembly del simulador MSX88. Ejecutarlos y verificar los resultados, analizando el flujo de información entre los distintos componentes del sistema*

### 1- 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.

PIC	EQU 20H	ORG 2000H
DMA	EQU 50H	CLI
N_DMA	EQU 20	MOV AL, N_DMA
		OUT PIC+7, AL ; reg INT3 de PIC
	ORG 80	MOV AX, OFFSET
		MSJ
IP_DMA	DW RUT_DMA	OUT DMA, AL ; dir comienzo ..
		MOV AL, AH ; del bloque ..
	ORG 1000H	OUT DMA+1, AL ; a transferir
MSJ	DB "FACULTAD DE"	MOV AX, OFFSET FIN-OFFSET MSJ
	DB " INFORMATICA"	OUT DMA+2, AL ; cantidad ..
FIN	DB ?	MOV AL, AH ; a ..
NCHAR	DB ?	OUT DMA+3, AL ; transferir
		MOV AX, OFFSET COPIA
	ORG 1500H	OUT DMA+4, AL ; dir destino ..
COPIA	DB ?	MOV AL, AH ; del ..
		OUT DMA+5, AL ; bloque
; rutina	aten interrupción del CDMA	MOV AL, 0AH ; CDMA en transfer..
	ORG 3000H	OUT DMA+6, AL ; mem-mem por bloque
RUT_DMA:	MOV AL, 0FFH ;inhabilita..	MOV AL, 0F7H
	OUT PIC+1, AL ;interrupc de PIC	OUT PIC+1, AL ; habilita INT3
	MOV BX, OFFSET COPIA	STI
	MOV AL, NCHAR	MOV BX, OFFSET MSJ
	INT 7	MOV AL, OFFSET FIN-OFFSET MSJ
	MOV AL, 20H	MOV NCHAR, AL
	OUT PIC, AL ; EOI	INT 7 ; mensaje original
	IRET	MOV AL, 7H
		OUT DMA+7, AL ; arranque Transfer
		INT 0
		END

### Cuestionario:

- Analizar minuciosamente cada línea del programa anterior.
- Explicar qué función cumple cada registro del CDMA e indicar su dirección.
- Describir el significado de los bits del registro CTRL.
- ¿Qué diferencia hay entre transferencia de datos por bloque y bajo demanda?
- ¿Cómo se le indica al CDMA desde el programa que debe arrancar la transferencia de datos?
- ¿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?
- 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.
- ¿Qué sucede con los registros RF, CONT y RD del CDMA después de transferido un byte?
- ¿Qué evento hace que el CDMA emita una interrupción y a través de qué línea de control lo hace?
- ¿Cómo se configura el PIC para atender la interrupción del CDMA?
- ¿Qué hace la rutina de interrupción del CDMA del programa anterior?

## 2- 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.

```

PIC      EQU 20H
HAND     EQU 40H
DMA      EQU 50H
N_DMA    EQU 20

IP_DMA   DW  RUT_DMA

        ORG 1000H
MSJ      DB  " INFORMATICA"
FIN      DB  ?
FLAG     DB  0

; rutina atención interrupción del CDMA
RUT_DMA: ORG 3000H
        MOV AL, 0          ;inhabilita..
        OUT HAND+1, AL    ;interrup de HAND
        MOV FLAG, 1
        MOV AL, 0FFH      ;inhabilita..
        OUT PIC+1, AL     ;interrup de PIC
        MOV AL, 20H
        OUT PIC, AL       ; EOI
        IRET

        ORG 2000H
        CLI
        MOV AL, N_DMA
        OUT PIC+7, AL     ; reg INT3 de PIC
        MOV AX, OFFSET MSJ
        OUT DMA, AL       ; dir comienzo ..
        MOV AL, AH        ; del bloque ..
        OUT DMA+1, AL     ; a transferir
        MOV AX, OFFSET FIN-OFFSET MSJ
        OUT DMA+2, AL     ; cantidad ..
        MOV AL, AH        ; a ..
        OUT DMA+3, AL     ; transferir
        MOV AL, 4         ; inicialización ..
        OUT DMA+6, AL     ; de control DMA
        MOV AL, 0F7H
        OUT PIC+1, AL     ; habilita INT3
        OUT DMA+7, AL     ; arranque Transfer
        MOV AL, 80H
        OUT HAND+1, AL    ; interrup de HAND
        STI

        LAZO: CMP FLAG, 1
              JNZ LAZO
              INT 0
              END

```

### Cuestionario:

- Analizar minuciosamente cada línea del programa anterior.
- ¿Qué debe suceder para que el HAND-SHAKE emita una interrupción al CDMA?
- ¿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?
- Explicar detalladamente cada paso de la operación de transferencia de un byte desde una celda de memoria hacia el HAND-SHAKE y la impresora.
- ¿Qué evento hace que el CDMA emita una interrupción al PIC?
- ¿Cuándo finaliza la ejecución del LAZO?

**3. \* Configuración del CDMA.** Indique cómo configurar el registro **Control** del CDMA para las siguientes transferencias:

- Transferencia Memoria → Memoria, por robo de ciclo
- Transferencia Periférico → Memoria, por ráfagas
- Transferencia Memoria → Periférico, por robo de ciclo

# **PRACTICA 3**

## **Anexo USART**

- 1) **Uso de la impresora a través del dispositivo USART por consulta de estado.** Ejecutar utilizando el simulador MSX88 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**.

**Nota:** Los ejercicios marcados con \* tienen una solución propuesta.