



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	8000h	0005h	BASURA
2	mov bx,3	8000h	0005h	0003h
3	push ax	7FFEh	0005H	0003H
4	push ax	7FFCh	0005H	0003H
5	push bx	7FFAh	0005H	0003H
6	pop bx	7FFCh	0005H	0003H
7	pop bx	7FFEh	0005H	0005H
8	pop ax	8000H	0005H	0005H

- 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	7FFCh
3	ret	7FFEh
4	org 2000h	-----
5	push ax	7FFEh
6	call rutina	7FFCh
7	pop bx	8000H
8	hlt	8000H
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 2000 h
add cx,5 ; Dirección 2002 h
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)	<code>mov ax,5</code> <code>call subrutina</code>	x		x	
b)	<code>mov dx, offset A</code> <code>call subrutina</code>	x			x
c)	<code>mov bx, 5</code> <code>push bx</code> <code>call subrutina</code> <code>pop bx</code>		x	x	
d)	<code>mov cx, offset A</code> <code>push cx</code> <code>call subrutina</code> <code>pop cx</code>		x		x
e)	<code>mov dl, 5</code> <code>call subrutina</code>	x		x	
f)	<code>call subrutina</code> <code>mov A, dx</code>	x		x	

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)?



```
c)org 1000h
a db 5
b db 15
c db 10
d db 0
org 3000h
calculo: mov ax, a
mov bx, b
mov cx, c
add ax,bx
sub ax,cx
mov dx,ax
mov d,dx
ret
```

```
org 2000h
call calculo
hlt
end
```

me parece que la 5b no se podría reutilizar así tal cual porque se cargan los registros desde las variables directamente dentro de la subrutina. pero el 5c) sí, desde el programa principal podés cargar lo que necesitas y después llamás la subrutina y usa esos registros





```
;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 **val**
- 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**. (los hizo el ruso en el primer video, este y el otro)

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? es indirecto por registro, copia el valor a la que la dirección contenida en bx esta apuntando
- ¿Qué función cumple el registro temporal **ri** que aparece al ejecutarse una instrucción como la anterior? guarda temporalmente el índice de la variable a utilizar antes de que sea llevada al registro
- ¿Qué se se guarda en AX al ejecutarse MOV AX, OFFSET RES? guarda la dirección en memoria de res, en este caso es 1004h
- ¿Cómo se pasa la variable RES a la pila, por valor o por referencia? ¿Qué ventaja tiene esto? se para por referencia, tiene utilidad para no utilizar registros demas
- ¿Cómo trabajan las instrucciones PUSH y POP? al hacer un push los datos se guardan en la pila y el punto de la pila se incrementa en 2, el pop desapila la pila, es decir copia los datos que tenia al registro indicado y decrementa el valor del puntero de la pila en 2

```

    ORG 3000H ; Subrutina MUL
    MOV BX, 12 ; preservar registros
    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
    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

    ORG 1000H ; Memoria de datos
    NUM1 DW 5H
    NUM2 DW 6H
    RES DW ?

    ORG 2000H ; Prog principal
    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

```

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



6 b) org 1000h num1 db 2 num2 db 5 res db ?	6 c) org 1000h num1 db 2 num2 db 5 res db ?
ORG 3000h mul: mov cx,0 loop: add cx,bx dec ax jnz loop ret org 2000h mov ax,num1 mov bx,num2 CALL mul hlt end	ORG 3000h mul: mov cx,0;cx=0 mov bx,sp; bx=pila add bx,2;bx apunta a num 2 mov dx, [bx] ;dx=num2 add bx,2; bx apunta a num 1 mov ax, [bx];ax=num1 loop: add cx,dx; agrego dx(num2) a cx(0) dec ax ; decremento ax (2-1) jnz loop; si eso no me dio cero, continuo ret org 2000h mov ax,num1;5 mov bx,num2;2 push ax;pila -2 push bx;pila -2 CALL mul;pila -2 mov res,cx; muevo el resultado que esta en cx a res pop ax; restauro los valores de la pila pop bx; restauro los valores de la pila y de los registros hlt end

8 b) ORG 1000H
TAB1 DB "AbCd"
 DB 00H
RES DW ?
ORG 3000H
CONTAR_MIN: MOV CX,0; lleva cx a cer
SALTO: MOV DL, [BX]; lo que apunta la direccion de bx la lleva a dl
CMP DL, 0; compara dl con 0
JZ FIN; si es cero, termina
INC BX; incrementa bx;
CMP DL, 97 ; se fija si es letra determinada
JS SALTO; si es, salta
CMP DL, 122; se fija si es letra determinada
JNS SALTO; si no es, salta de nuevo
INC CX; aumenta cx
JMP SALTO; hace un salto incondicional a salto
FIN: RET; termina la subrutina
RET
ORG 2000H
MOV BX, OFFSET TAB1; le pasa la direccion del string
CALL CONTAR_MIN; llama a la subrutina
MOV RES, CX; mueve cuantas coincidencias hubo a cx
HLT
END



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

8) Subrutinas para realizar operaciones con cadenas de caracteres

- a) 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)

- b) 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.

- c) * 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

- d) * 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_MIN de 'aBcDE1#!' debe retornar 2

- e) 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_MIN de 'abbcdel!' y 'b' debe retornar 2, mientras que CONTAR_MIN de 'abbcdel!' y 'z' debe retornar 0.

- f) 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

- a) 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.

11) 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?
suma consecutivamente los numeros, por ejemplo si entra un 5 hace= $5+4+3+2+1$
- b) ¿Cuál será el valor final de CX? en este caso es 6
- c) Dibujar las posiciones de memoria de la pila, anotando qué valores va tomando: por cada call se apila a la proxima direccion que tiene que ir, por ejemplo luego del primero queda 200D y luego dentro de la subrutina se llama tres veces mas al call y queda 3012 tres veces
- d) ¿Cuál será la limitación para determinar el valor más grande que se le puede pasar a la subrutina a través de AX? que tiene solo 16 bits ax

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



```
10) org 1000h
num1 dw 9999h
num2 dw 9fffh
```

```
org 3000h
swap: mov bx,sp
add bx,2
mov ax, [bx]
mov bx,ax
mov cx,[bx]
```

```
mov bx,sp
add bx,4
mov dx,[bx]
mov bx,dx
mov bx,[bx]
```

```
push bx
mov bx,dx
mov [bx],cx
pop cx
mov bx,ax
mov [bx],cx
ret
```

```
org 2000h
mov ax,99h
push ax
push bx
push cx
push dx
mov ax, offset num1
push ax
mov ax,offset num2
push ax
call swap
pop ax
pop ax
pop dx
pop cx
pop bx
pop ax
hlt
end
```