PRACTICA 1

Subrutinas y pasaje de parámetros

Explicación

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 VonSim. 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	5	?
2	mov bx,3	8000H	5	3
3	push ax	7FFEH	5	3
4	push ax	7FFCH	5	3
5	push bx	7FFAH	5	3
6	pop bx	7FFCH	5	3
7	pop bx	7FFEH	5	3
8	pop ax	8000H	5	3

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**

1. 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:
   a) Las acciones que tienen lugar al ejecutarse la instrucción call rut
   b) 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.

a) 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**.
b) 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.
c) 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
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
a) Sin hacer llamados a subrutinas, resolviendo el problema desde el programa principal;
b) 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**.
c) 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.

a) ¿Cuál es el modo de direccionamiento de la instrucción MOV AX, [BX]? ¿Qué se copia en el registro AX en este caso?
b) ¿Qué función cumple el registro temporal **ri** que aparece al ejecutarse una instrucción como la anterior?
c) ¿Qué se guarda en AX al ejecutarse MOV AX, OFFSET RES?
d) ¿Cómo se pasa la variable RES a la pila, por valor o por referencia? ¿Qué ventaja tiene esto?
e) ¿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.

```
    ORG 3000H ; Subrutina MUL
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
```

```
     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
```

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

DIRECCIÓN DE MEMORIA	CONTENIDO
7FF0H	DL
	DH
7FF2H	AL
	AH
7FF4H	CL
	CH
7FF6H	BL
	BH

7FF8H	IP RET. L
	IP RET. H
7FFAH	DIR. RES L
	DIR. RES H
7FFCH	NUM2 L
	NUM2 H
7FFEH	NUM1 L
	NUM1 H
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_VOC de 'contar1#!' 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_CAR de 'abbcde!' y 'b' debe retornar 2, mientras que CONTAR_CAR de 'abbcde!' 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 **1**0010100 se obtiene 00101001**1**, y al rotar a la izquierda **0**1101011 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):

```
   01101011
+  01101011
   11010110 (CARRY=0)
```

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.

```
   10010100
+  10010100
   00101000  (CARRY=1)
+  00000001
   00101001
```

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 **10**010100, se obtiene el byte 01010100.

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 10**010100** 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
MUL:  CMP AX, 0
        JZ  FIN
        ADD CX, AX
        DEC AX
        CALL MUL
FIN:  RET
```

```
        ORG 2000H
MOV CX, 0
        MOV AX, 3
        CALL MUL
        HLT
END
```

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.