PRÁCTICA 1 – Algunas resoluciones

Subrutinas y pasaje de parámetros

Ejercicio 1

	Instrucción	**Valor del registro SP**	**AX**	**BX**
1	mov ax,5	8000	5	?
2	mov bx,3	8000	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	5
8	pop ax	8000	5	5

Ejercicio 2

#	**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	

Ejercicio 3

La siguiente tabla indica el contenido de las direcciones 7FFFh y 7FFEh de la pila, que son las únicas que se utilizan en todo el programa y además el valor del registro SP, luego de la ejecución de cada instrucción. Cada columna indica una instrucción ejecutada. Algunas instrucciones se repiten, ya sea porque están dos veces en el programa (como `call rut`) y otras porque se *ejecutan* dos veces, como las instrucciones que están dentro de `rut`.

	Instrucción ejecutada							
Pila	call rut	mov bx,3	ret	add cx,5	call rut	mov bx,3	ret	hlt
7FFEh	02	02	?	?	06	06	?	?
7FFFh	20	20	?	?	20	20	?	?
SP	7FFEh	7FFEh	8000h	8000h	7FFEh	7FFEh	8000h	8000h

Ejercicio 4

	Código	Registro	Pila	Valor	Referencia
a	`mov ax,5` `call subrutina`	SI		SI	
b	`mov dx, offset A` `call subrutina`	SI			SI

c	`mov bx, 5`
	`push bx`
	`call subrutina`
	`pop bx`
			SI	SI	

d	`mov cx, offset A`
	`push cx`
	`call subrutina`
	`pop cx`
			SI		SI

e	`mov dl, 5` `call subrutina`	SI		SI	
f	`call subrutina` `mov A, dx`	SI		SI	

Ejercicio 5a

```
  ; Memoria de Datos
     org 1000h
  A  DW  5h
  B  DW  6h
  C  DW  2h
  D  DW  ?
  ; Memoria de programa
     org 2000h
     mov ax, A
     add ax, B
     sub ax, C
     mov D, ax
     hlt
     end
```

Ejercicio 5b

```
  ; Memoria de Datos
     org 1000h
  A  DW  5h
  B  DW  6h
  C  DW  2h
  D  DW  ?
     org 3000h
calculo: mov ax, A
         add ax,B
         sub ax,C
         mov D, ax
         ret
  ; Memoria de programa
     org  2000h
     call calculo
     hlt
     end
```

Ejercicio 5c

```
  ; Memoria de Datos
```

```
     org 1000h
  A  DW  5h
  B  DW  6h
  C  DW  2h
  D  DW  ?
     org 3000h
  ; Recibe en ax, bx y cx tres valores A, B y C
  ; Devuelve en dx el cálculo A+B-C
calculo: mov dx,ax
         add dx,bx
         sub dx,cx
         ret
     org 2000h       ; programa principal
     mov ax, A
     mov bx, B
     mov cx, C
     call calculo
     mov D, dx
     hlt
     end
```

Ejercicio 6

A)

```
      ; Memoria de Datos
      ORG 1000H
NUM1  DB    5H
NUM2  DB    3H
RES   DW    ?
      ; Memoria de Instrucciones
      ORG 2000H
      MOV  DX, 0
      MOV  AL, NUM1
      CMP  AL, 0
      JZ   FIN
      MOV  AH, 0
      MOV  CL, NUM2
LOOP: CMP  CL, 0
      JZ   FIN
      ADD  DX, AX
      DEC  CL
      JMP  LOOP
FIN:  MOV  RES, DX
      HLT
      END
```

B)

```
      ; Memoria de Datos
      ORG 1000H
NUM1  DB   5H
NUM2  DB   3H
RES   DW   ?
      ; Memoria de Instrucciones
      ORG 3000H   ; Subrutina MUL
MUL:  MOV  DX, 0
      CMP  CL, 0
      JZ   FIN
      MOV  AH, 0
LAZO: ADD  DX, AX
      DEC  CL
      JNZ  LAZO
FIN:  RET

      ORG 2000H   ; Programa
principal
      MOV  AL, NUM1
      MOV  CL, NUM2
      CALL MUL
      MOV  RES, DX
      HLT
      END
```

C)

```
      ORG 3000H   ; Subrutina MUL
MUL:  MOV  DX, 0
      ; obtener operandos
      ; desde la memoria
      MOV  BX, AX
      MOV  AX, [BX]
      MOV  BX, CX
      MOV  CX, [BX]
      ; comprobar que CX > 0
      CMP  CX, 0
      JZ   FIN
LAZO: ADD  DX, AX
      DEC  CX
      JNZ  LAZO
FIN:  RET
```

```
; Memoria de datos
     ORG 1000H
NUM1  DW  5H
NUM2  DW  3H
RES   DW  ?

     ORG 2000H    ; Programa
     MOV  AX, OFFSET NUM1
     MOV  CX, OFFSET NUM2
     CALL MUL
     MOV  RES, DX
     HLT
     END
```

Ejercicio 8a

```
          ORG 1000H
  CAD     DB  "EXCELENTE"
          DB  00H
          ORG 3000H
LONGITUD: MOV DX, 0         ;contador
  LOOP:   MOV AH, [BX]
          CMP AH, 00H
          JZ  FIN
          INC DX
          INC BX
          JMP LOOP
  FIN:    RET

         ORG 2000h
         MOV BX, offset CAD
         CALL LONGITUD
         HLT
         END
```

Ejercicio 8c. Esta resolución sólo aplica para el caso en que las vocales sean mayúsculas.

```
          ORG 1000H
resultado DB ?    ; cambiar este valor y ver que queda en la variable resultado
   CHAR   DB "E"

  ; Recibe el caracter a verificar por AH
  ; Devuelve el resultado en AL
          ORG 3000H
ES_VOCAL: MOV AL, 0FFH
          CMP AH, 41H       ; A
          JZ  FIN
          CMP AH, 45H       ; E
          JZ FIN
          CMP AH, 49H       ; I
          JZ FIN
          CMP AH, 4FH       ; O
          JZ FIN
          CMP AH, 55H       ; U
          JZ FIN
          MOV AL, 00H
  FIN:    RET

          ORG 2000h
          MOV AH, CHAR
          CALL ES_VOCAL
          MOV resultado, AL
          HLT
          END
```

Ejercicio 8d

```
     ; Recibe el caracter a verificar por AH
     ; Devuelve el resultado en AL
          ORG 4000H
ES_VOCAL: MOV AL, 0FFH
          CMP AH, 41H       ;A
          JZ FIN
          CMP AH, 45H       ;E
          JZ FIN
          CMP AH, 49H       ;I
          JZ FIN
          CMP AH, 4FH       ;O
          JZ FIN
          CMP AH, 55H       ;U
```

```
            JZ FIN
            MOV AL, 00H
   FIN:     RET

            ORG 1000H
   CAD      DB  "EXCELENTE"
   CERO     DB  0
 resultado DW  ?

   ; Recibe en BX la dirección de la cadena
   ; retorna en CX la cantidad de vocales
            ORG 3000H
   VOCALES: MOV CX, 0         ; cantidad de vocales
   LOOP:    MOV AH, BYTE PTR [BX] ; pongo en AX el caracter correspondiente a [BX]
            CMP AH, 0   ; si llegue al valor 0 (fin de cadena)
            JZ  fin_vocales   ; retorno
            CALL ES_VOCAL
            CMP AL, 0FFH      ; si no son iguales, no es vocal
            JNZ NOES
            INC CX            ; incremento vocales
   NOES:    INC BX            ; me muevo por la cadena
            JMP LOOP          ; verifico el próximo char
fin_vocales: RET

            ORG 2000h
            MOV BX, offset CAD
            CALL VOCALES
            MOV resultado, CX
            HLT
            END
```

Ejercicio 9a

```
   ; Recibe el caracter a rotar en AH
   ; Devuelve el resultado también en AH
            ORG 3000H
 ROTARIZQ:  ADD AH, AH
            ADC  AH, 0
            RET
            ORG 1000H
      b     DB  27H           ; (00100111) en binario
            ORG 2000H
            MOV AH, b         ; AH = 00100111
  ; Realizamos una rotación
            CALL ROTARIZQ     ; AH = 01001110
  ; Realizamos una segunda rotación
            CALL ROTARIZQ     ; AH = 10011100
            HLT
            END
```

Ejercicio 9b (asumimos que está disponible la subrutina ROTARIZQ definida anteriormente)

```
  ; Recibe el caracter a rotar en AH
  ; Recibe la cantidad de posiciones en BH
  ; Devuelve el resultado también en AH
            ORG 4000H
ROTARIZQ_N: CMP BH, 0   ; mientras BH>0
            JZ  FIN     ; si BH=0, entonces finalizar la subrut.
            CALL ROTARIZQ
            DEC BH
            JMP ROTARIZQ_N    ; aprovecho la etiqueta de la subrutina
                              ; para hacer el salto
  FIN:      RET

            ORG 1000H
      b     DB  27H           ; (00100111) en binario
```

```
          ORG 2000H
          MOV AH, b
  ; Realizamos una rotación de 2 posiciones a la izquierda
          MOV BH, 2
          CALL ROTARIZQ_N   ; AH = 10011100 (C9H)
          HLT
          END
```

Ejercicio 9c (asumimos que está disponible la subrutina ROTARIZQ_N definida anteriormente)

```
    ; Utiliza los mismos registros que ROTARIZQ_N
    ; Recibe en BH la cantidad de posiciones
          ORG 5000H
ROTARDER_N: MOV CH, 8
          SUB CH, BH        ; cantidad de bytes que debo rotar hacia la izq.
          MOV BH, CH        ; vuelvo a copiar en BH
      ; ROTARIZQ usará el valor almacenado en BH para rotar.
          CALL ROTARIZQ_N
          RET

          ORG 1000H
     b    DB 27h            ; (00100111) en binario

          ORG 2000H
          MOV AH, b
   ; Realizamos una rotación de 6 posiciones a la derecha
          MOV BH, 2
          CALL ROTARDER_N   ; AH = 10011100 (C9H)
          HLT
          END
```

Ejercicio 10

```
    ; Recibe las direcciones de dos celdas de memoria a intercambiar M1 y M2
    ; a través de la pila
          ORG 3000H
    SWAP: PUSH BX           ; preservo los 3 registros
          PUSH AX
          PUSH DX
    ; OBTENER EL VALOR DE M2 en CX
          MOV BX, SP
          ADD BX, 8         ; apunto al segundo parámetro
    ; 8=6+2: 6=3*2 son de los push; los otros 2 por la dir de retorno
          MOV BX, [BX]      ; BX tiene la DIR de M2
          MOV CX, [BX]      ; CX tiene el valor de M2
    ; OBTENER EL VALOR DE M1 en DX
          MOV BX, SP
          ADD BX, 10        ; apunto al primer parámetro
    ; 10=6+2+2: 6=3*2 son de los push; 2 por la dir de retorno, y 2 por M2
          MOV BX, [BX]      ; BX tiene la DIR de M1
          MOV DX, [BX]      ; DX tiene el valor de M1
    ; PONER EL VALOR DE M1 (DX) en M2
          MOV BX, SP
          ADD BX, 8         ; apunto al segundo parámetro
          MOV BX, [BX]      ; BX tiene la DIR de M2
          MOV [BX], DX      ; Asigno el valor de M1 en la dir de M2
    ; PONER EL VALOR DE M2 (CX) en M1
          MOV BX, SP
          ADD BX, 10        ; apunto al primer parámetro
          MOV BX, [BX]      ; BX tiene la DIR de M2
          MOV [BX], CX      ; Asigno el valor de M2 en la dir de M1
    ; restauro los 3 registros
          POP DX
          POP AX
          POP BX
```

```
         RET

         ORG 1000H
   val1  DW  1234H
   val2  DW  5678H

         ORG 2000H
         MOV AX, offset val1
         PUSH AX
         MOV AX, offset val2
         PUSH AX
         CALL SWAP
; verificar que se hayan intercambiado los valores entre val1 y val2
         HLT
         END
```

Ejercicio 11b

```
         ORG 1000H
   num1  DB  6H
   num2  DB  4H

; subrutina resto
; Recibe dos números en los registros CH y CL
; Retorna el resto de la división entera (sin coma) de CH/CL
; Por ejemplo el resto de 6/4 es 2
         ORG 3000H
resto:  MOV AL, 0          ; inicializo el resto en 0
         MOV DH, 0         ; inicializo el cociente de la división
         CMP CH, 0         ; CH tiene NUM2
         JZ  FIN
         CMP CL, 0         ; CL tiene NUM1
         JZ  FIN
DIV:     SUB CL, CH
         JS RES            ; si resultado negativo, voy a calcular el resto
         INC DH            ; sumo al cociente
         JMP DIV
RES:     ADD CL, CH        ; sumo de vuelta CH para determinar el resto
         MOV AL, CL        ; devuelvo el resto en AX
FIN:     RET

         ORG 2000H
         MOV CL, num1
         MOV CH, num2
         CALL resto
         HLT
         END
```