

3-a)

```
ORG 1000H
NUM1 DW 9234H }
      DW 1111H } NUM1 = 11119234H
NUM2 DW 9234H }
      DW 2222H } NUM1 = 22229234H
```

```
ORG 2000H
MOV AX, NUM1 ; AX:=9234H Parte baja de NUM1
MOV CX, NUM2 ; CX:=9234H Parte baja de NUM2
MOV DX, NUM1+2; DX:=1111H Parte alta de NUM1
MOV BX, NUM2+2; CX:=2222H Parte alta de NUM2
ADD AX, CX ; AX:=AX+BX Suma parte baja
ADC DX, BX ; DX:=DX+BX+Carry ( Carry generado por la suma de la parte baja )
HLT
END
```

El resultado de la suma queda almacenado en DX AX. ( 32 bits ). Aquí y en el resto del ej. consideramos que la parte alta de la suma no genera Carry.

3-b)

```
ORG 1000H
NUM1 DW 9234H
      DW 1111H
NUM2 DW 9234H
      DW 2222H
```

```
ORG 3000H
SUM32: ADD AX, CX } Subrutina que suma y deja el resultado en DX AX.
      ADC DX, BX }
      RET
```

```
ORG 2000H
MOV AX, NUM1 }
MOV DX, NUM1+2 } Pasaje de parámetros en los registros para que los use la
MOV CX, NUM2 } subrutina. En este caso se pasan los números ó sea por valor
MOV BX, NUM2+2 }
```

```
CALL SUM32
HLT
END
```

El programa principal es responsable por cargar en los registros los valores adecuados (parámetros). La subrutina utiliza estos valores directamente (AX, DX, CX, BX).

3-c)

```
ORG 1000H
NUM1 DW 9234H
      DW 1111H
NUM2 DW 9234H
      DW 2222H

ORG 3000H
SUM32: MOV BX,AX      ; BX:=1000H dirección de NUM1
        MOV AX,[BX]   ; AX:=9234H parte baja de NUM1
        ADD BX,2       ; BX:=1002H apunta parte alta NUM1
        MOV DX,[BX]   ; DX:=1111H parte alta NUM1
        MOV BX,CX      ; BX:=1004H dirección de NUM2
        MOV CX,[BX]   ; CX:=9234H parte baja de NUM2
        ADD BX,2       ; BX:=1006H apunta parte alta NUM2
        ADD AX,CX      ; AX:=9234H+9234H=2468H y generó Carry
        ADC DX,[BX]   ; DX:=1111H+2222H+1=3334H
        RET

ORG 2000H
MOV AX,OFFSET NUM1 ; AX:=1000H ,Dirección NUM1
MOV CX,OFFSET NUM2 ; CX:=1004H ,Dirección NUM2
CALL SUM32
HLT
END
```

} Pasaje por referencia, se pasan las direcciones de los N°s.

3-d) Por valor por la pila

```
ORG 1000H
NUM1 DW 9234H
      DW 1111H
NUM2 DW 9234H
      DW 2222H

ORG 3000H
SUM32: PUSH BX ; Salva Viejo BX=0000H en 7FF5H y 7FF4H
        MOV BX,SP ; BX=SP=7FF4H
        ADD BX,10; BX=7FFEh apunta a parte baja NUM1
        MOV AX,[BX] ; AX:=9234H
        SUB BX,2 ; BX:=7FFCh apunta a parte alta NUM1
        MOV DX,[BX] ; DX:=1111H
        SUB BX,2 ; BX:=7FFAh apunta a parte baja NUM2
        MOV CX,[BX] ; CX:=9234H
        SUB BX,2 ; BX:=7FF8H apunta parte alta NUM2
        ADD AX,CX ; AX:=9234H+9234
        ADC DX,[BX] ; DX:=1111H+2222H+1
```

POP BX ; Carga en BX el valor que tenía antes de llamar al procedimiento  
RET

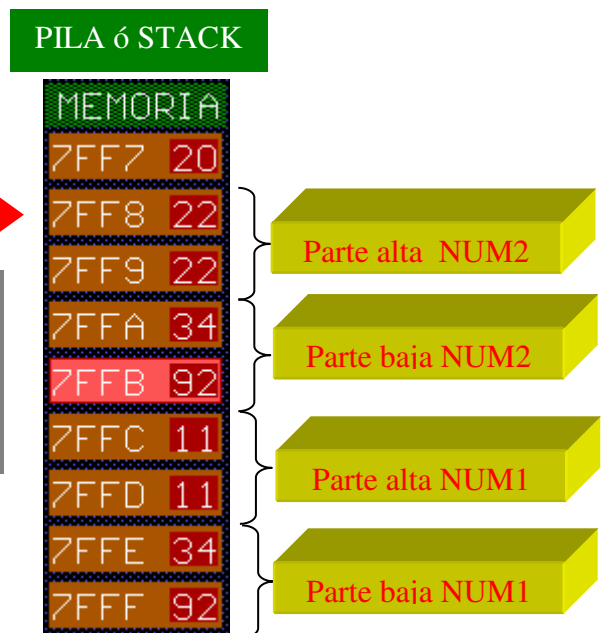
ORG 2000H

MOV AX,NUM1 ; AX:=9234H parte baja NUM1  
PUSH AX ; Apila parte baja de NUM1 en 7FFFH y 7FFE  
MOV AX,NUM1+2 ; AX:=1111H parte alta de NUM1  
PUSH AX ; Apila parte alta de NUM1 en 7FFDH y 7FFCH  
MOV AX,NUM2 ; AX:=9234H parte baja de NUM2  
PUSH AX ; Apila parte baja de NUM2 en 7FFBH y 7FFAH  
MOV AX,NUM2+2 ; AX:=2222H parte alta de NUM2  
PUSH AX ; Apila parte alta de NUM2 en 7FF9H y 7FF8H  
CALL SUM32  
HLT

Pasaje de parámetros  
por valor en la pila.

La figura muestra la pila y la posición de SP  
justo antes de llamar a la subrutina SUM32.  
Recordar que el simulador inicializa SP=8000H y  
primero se decrementa antes de “apilar” el primer  
dato.

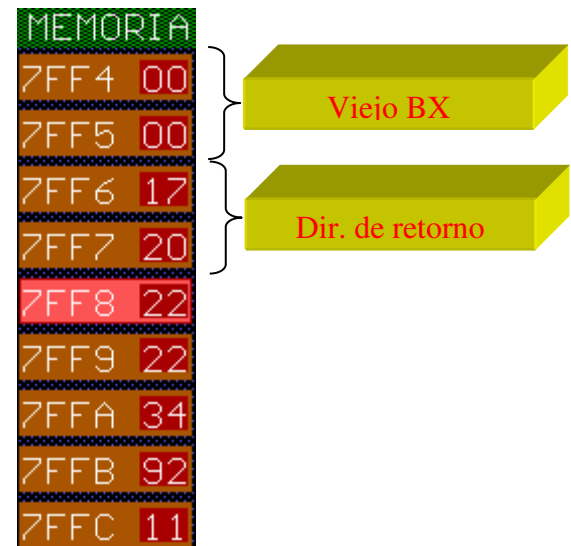
SP →



END

BX=SP →

La figura muestra la pila y la posición de SP luego  
de “entrar” en SUM32 y ejecutar PUSH BX. La  
dirección de retorno es 2017H dirección de HLT,  
instrucción debajo de CALL SUM32. Aquí se hace  
BX=SP.



3-d) Por referencia por la pila

```
ORG 1000H
NUM1 DW 9234H
      DW 1111H
NUM2 DW 9234H
      DW 2222H
```

```
ORG 3000H
SUM32: PUSH BX    ; Salva Viejo BX:=0000H en 7FF9H y 7FF8H
      MOV BX,SP   ; BX:=SP=7FF8H
      ADD BX,6    ; BX:=7FF8H+0006H=7FFEh apunta a dir NUM1 en la pila
      MOV AX,[BX] ; AX:=1000H dir NUM1
      SUB BX,2    ; BX:=7FFCH apunta a dir NUM2 en la pila
      MOV CX,[BX] ; CX:=1004H dir NUM2
      MOV BX,AX   ; BX=AX=1000H
      MOV AX,[BX] ; AX:=9234H parte baja NUM1
      ADD BX,2    ; BX:=1000H+0002H=1002H dir parte alta NUM1
      MOV DX,[BX] ; DX:=1111H parte alta NUM1
      MOV BX,CX   ; BX:=1004H dir NUM2
      MOV CX,[BX]
      ADD BX,2    ; BX:=1004H+0002H=1006H dir parte alta NUM2
      ADD AX,CX
      ADC DX,[BX] ; DX:=1111H+2222H+1=3334H
      POP BX     ; Recupera viejo BX
      RET
```

```
ORG 2000H
MOV AX,OFFSET NUM1 ; AX:=1000H dirección de NUM1
PUSH AX
MOV AX,OFFSET NUM2 ; AX:=1004H dirección de NUM2
PUSH AX
CALL SUM32
HLT
END
```

} Pasaje de parámetros por  
referencia en la pila.



4)

```

ORG 1000H
DATO DB 193      ; Número a rotar
ROTA DB 4        ; Veces a rotar
    
```

```

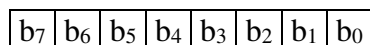
ORG 3000H
ROTARIZ: ADD AL,AL ; Suma
          JNC SIGO  ; Si no hay carry salta
          ADC AL,0   ; Hay carry entonces hay que sumarlo
SIGO:     DEC CL     ; Decremento el número de veces a rotar
          JNZ ROTARIZ ; Sigo hasta que CL:=0
          RET
    
```

```

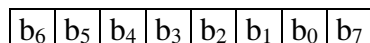
ORG 2000H
MOV AL,DATO ; AL:=193
MOV CL,ROTA ; CL:=4
CALL ROTARIZ
HLT
END
    
```

Parámetros pasados por registros.

Expliquemos el mecanismo que usamos para rotar hacia izquierda



Rotar un lugar a izquierda significa que el contenido de la celda de memoria ó registro será



Este desplazamiento a izquierda se puede interpretar como multiplicar por 2, o sea sumar al número consigo mismo. Veamos un ej.

Arquitectura de Computadoras – Fac. de Informática – Prof. Jorge Runco  
Curso 2019 – TP N° 1 : Subrutinas y pasaje de parámetros

$$\begin{array}{r} + \quad 00001001 \\ \quad 00001001 \\ \hline \quad 00010010 \end{array}$$

Acá el mecanismo funcionó bien. Veamos otro ej.

$$\begin{array}{r} + \quad 10001000 \\ \quad 10001000 \\ \hline 1 \leftarrow 00010000 \end{array}$$

No funcionó porque se generó carry. El resultado correcto se obtiene sumando al resultado parcial este carry, entonces :

$$\begin{array}{r} + \quad 00010000 \\ \quad \quad 1 \\ \hline \quad 00010001 \end{array}$$

Esto es lo que hace la subrutina, suma el número consigo mismo y verifica si hay carry para sumárselo al resultado parcial.

5)

```
ORG 1000H
FRASE DB "AzAytA",0
```

```
ORG 3000H
CONCAR: CMP BYTE PTR [BX],0 ; Verifica si llegó al final
        JZ FIN             ; Salta si alcanzó el 0 en el string y termina
        INC CX              ; Cuenta los caracteres
        INC BX              ; Incrementa BX para apuntar al siguiente caracter
        JMP CONCAR
FIN:    RET
```

```
ORG 2000H
MOV BX,OFFSET FRASE
MOV CX,0
CALL CONCAR
HLT
END
```

6)

```

PARAMETRO1    ORG 1000H
                dw 1234H
PARAMETRO2    dw 5abcH

                ORG 3000H
SWAP :         MOV BX, SP
                ADD BX, 4
                MOV CX, [BX]
                SUB BX, 2
                MOV DX, [BX]
                MOV BX, CX
                MOV AX, [BX]
                PUSH AX
                MOV BX, DX
                MOV AX, [BX]
                MOV BX, CX
                MOV [BX], AX
                POP AX
                MOV BX, DX
                MOV [BX], AX
                RET

                ORG 2000H
                MOV AX, OFFSET PARAMETRO1
                PUSH AX
                MOV AX, OFFSET PARAMETRO2
                PUSH AX
                CALL SWAP
                HLT
                END
```

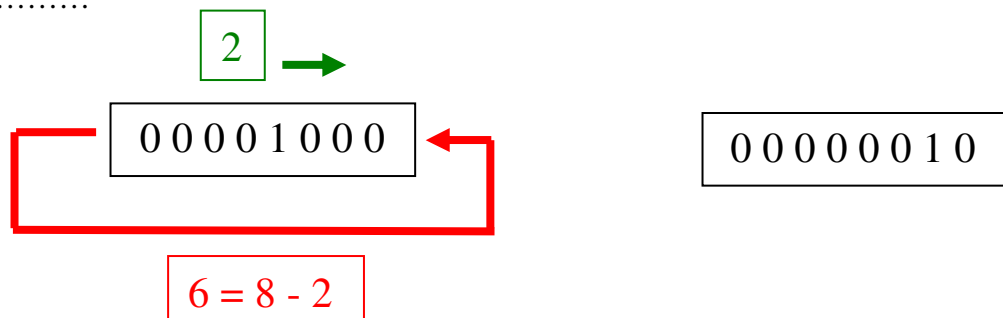
7)

```
ORG 1000H
FRASE DB "AzAytA",0
CARAC DB "A"

ORG 3000H
NCHAR: PUSH BX
        MOV BX,SP
        ADD BX,4
        MOV AX,[BX]
        ADD BX,2
        MOV DX,[BX]
        MOV BX,DX
SIGO:   CMP BYTE PTR [BX],0
        JZ FIN
        CMP AL,[BX]
        JNZ NO_SUMO
        INC CX
        MOV [BX], "X"
NO_SUMO: INC BX
        JMP SIGO
FIN:     POP BX
        RET

ORG 2000H
MOV CX,0000H
MOV BX,OFFSET FRASE
PUSH BX
MOV AH,0
MOV AL,CARAC
PUSH AX
CALL NCHAR
HLT
END
```

8) Rotar 1 vez a derecha es = rotar 7 veces a izquierda, rotar 2 veces a derecha es = rotar 6 veces a izquierda y así.....





```

                                ORG 1000H
DATO      DB    193
ROTADE    DB     3

                                ORG 3000H
ROTARIZ:  ADD AL,AL ; Suma
          JNC SIGO   ; Si no hay carry salta
          ADC AL,0    ; Hay carry entonces hay que sumarlo
SIGO:     DEC CL     ; Decremento el número de veces a rotar
          JNZ ROTARIZ ; Sigo hasta que CL:=0
          RET

                                ORG 4000H
ROTADER:  MOV AH, 8
          SUB AH, CL
          MOV CL, AH
          CALL ROTARIZ
          RET

                                ORG 2000H
          MOV AL, DATO
          MOV CL, ROTADE
          CALL ROTADER
          HLT
          END
```

9)

Va la explicación : vamos a rotar a derecha el 7 = 00000111 tres posiciones y debe quedar 11100000.

Rotar a derecha una posición es como dividir por 2. Para dividir por 2 lo hago por definición, ej  $7-2=5$ ;  $5-2=3$ ;  $3-2=1$  acá paro cuando el resultado es 1 ó 0. Cuento las veces que resté 2, en este caso 3 y es el resultado de la división. El resultado de la última resta es 1 cuando el número es impar ó 0 cuando el número es par. Después de una división por 2, el 7=00000111 se transformó en

3=00000011, cuando el resto es 1 hay que poner el bit más significativo en 1 (rota el 1 de la derecha) o sea le sumo a 00000011 + 10000000 = 10000011 Ahora éste es el 7 rotado una vez. Esto lo tengo que hacer las veces que quiera rotar el número.

```
ORG 1000H
NUM      DB  7 ; Número a rotar
VECES    DB  3 ; Cantidad de rotaciones
```

```
ORG 2000H
MOV AH, NUM
MOV CL, VECES
CALL ROTADER
HLT
END
```

```
ORG 3000H
ROTADER : MOV DL, 0
          SIGH : SUB AH, 2 ; Hago las restas sucesivas
              INC DL      ; Cuento cuantas veces resté 2
              CMP AH, 2    ; Comparo para saber cuando la cuenta da 1 ó 0
              JNC SIGH     ; Mientras no da 1 ó 0 sigo restando
              MOV AH, DL   ; Cargo el resultado de la división en AH
              JZ  NADA     ; Me fijo si el resto es 1 ó 0
              ADD AH, 80H   ; Si es 1 lo sumo adelante
          NADA : DEC CL    ; Veces a rotar
              JNZ ROTADER
              RET
```

A ver si se entendió.

00000111=7 así arranco

00000011=3 luego de dividir por 2, pero todavía no está completo falta un 1 adelante que debía rotar, 00000011+10000000=10000011 ahora está completa la rotación de una sola posición. Si el resto es 0 este paso no hay que hacerlo.

La segunda división por 2 daría 01000001, igual que antes le sumamos 10000000 y queda 11000001 y queda completa.

La tercera división por 2 daría 01100000 y finalmente 11100000 que es rotar el 7, tres posiciones a derecha.

10)

```
ORG 1000H
DIVIDENDO DB 127
DIVISOR DB 10

RESTO:
ORG 3000H
SUB AH, CL
CMP AH, CL
JNC RESTO
RET

ORG 2000H
MOV AH, DIVIDENDO
MOV CL, DIVISOR
CALL RESTO
HLT
END
```

11)

```
ORG 3000H
ES_VOCAL: CMP AL, 41H ; ASCII A = 41H
JZ FIN
CMP AL, 45H ; Letra E
JZ FIN
CMP AL, 49H
JZ FIN
.....
CMP AL, 6FH ; ASCII o = 6FH
JZ FIN
CMP AL, 75H
JZ FIN
MOV AL, 00H
JMP FINAL
FIN: MOV AL, 0FFH
FINAL: RET
```

12)

```
ORG 4000H
VOCALES: CMP BYTE PTR [BX], 0
JZ FIN1
MOV AL, [BX]
CALL ES_VOCAL
CMP AL, 0
JZ NO_ES_VOC
INC CX
NO_ES_VOC: INC BX
JMP VOCALES
```

Arquitectura de Computadoras – Fac. de Informática – Prof. Jorge Runco  
Curso 2019 – TP N° 1 : Subrutinas y pasaje de parámetros  
FIN1: RET

El programa completo sería :

```
                ORG 1000H
FRASE          DB  "ABBEuMMno",0

                ORG 3000H
ES_VOCAL: ..... (EJ. 11)

                ORG 4000H
VOCALES: ..... (EJ.12)

                ORG 2000H
                MOV BX, OFFSET FRASE
                MOV CX, 0
                CALL VOCALES
                HLT
                END
```