

4)

```

ORG 1000H ( Dirección donde comienza la memoria de datos)
NUM0 DB 0CAH (Nombre_Variable Tipo_Dato Valor_Inicial)
NUM1 DB 0
NUM2 DW ?
NUM3 DW 0ABCDH
NUM4 DW ?

ORG 2000H ( Dirección donde comienza la memoria de programa )
MOV BL,NUM0 ; Carga en BL el valor que guarda NUM0, posición de memoria 1000H
               ;  $\Rightarrow$  BL:=CA , mdd Directo
MOV BH,0FFH ; Carga en BH el valor FF  $\Rightarrow$  BH:= FF, mdd Inmediato
MOV CH,BL ; Carga en CH el valor de BL  $\Rightarrow$  CH:=BL , mdd por Registro
MOV AX,BX ; Carga en AX el valor de BX  $\Rightarrow$  AX:=BX , mdd por Registro
MOV NUM1,AL ; Almacena en NUM1 el valor de AL; NUM1 ocupa el lugar de
               ; memoria 1001H; contenido de 1001H:=AL, mdd Directo
MOV NUM2,1234H ; Almacena en NUM2 el valor 1234H, NUM2 ocupa las
               ; posiciones de memoria 1002H y 1003H, mdd Inmediato
MOV BX,OFFSET NUM3 ; Usando la directiva OFFSET se almacena en BX el valor
               ;de la dirección de NUM3 (1004H) y NO su contenido (ABCDH),
               ; mdd inmediato
MOV DL,[BX] ; Carga en DL el valor almacenado en la posición de memoria
               ; apuntada por BX, mdd Indirecto por registro
MOV AX,[BX] ; Carga en AX el valor almacenado en la posición de memoria
               ; apuntada por BX y la siguiente, pues AX es de 16 bits
MOV BX,1006H; Carga en BX el valor 1006H, mdd Inmediato
MOV WORD PTR[BX],1006H; PTR es el operador “puntero” que indica que [BX]
               ;apunta a un dato tipo WORD

HLT
END

```

1b)

INSTRUCCION	MODO DE DIRECCIONA.	DESTINO
MOV BL,NUM0	DIRECTO	BL:=CAH
MOV BH,0FFH	INMEDIATO	BH:=FFH
MOV CH,BL	POR REGISTRO	CH:=CAH
MOV AX,BX	POR REGISTRO	AX:=FFCAH
MOV NUM1,AL	DIRECTO	1001H:=CAH
MOV NUM2,1234H	INMEDIATO	1002H:=34H1003H:=12H
MOV BX,OFFSET NUM3	INMEDIATO	BX:=1004H
MOV DL,[BX]	INDIRECTO POR REG	DL:=CDH
MOV AX,[BX]	INDIRECTO POR REG	AX:=ABCDH
MOV BX,1006H	INMEDIATO	BX:=1006H
MOV WORDPTR[BX],1006H	INDIRECTO POR REG	1006H:=06H1007H:=10H

5)

```
ORG 1000H
NUM0 DB 80H           1000H := 80H
NUM1 DB 200           1001H := C8H
NUM2 DB -1            1002H := FFH
BYTE0 DB 01111111B    1003H := 7FH
BYTE1 DB 10101010B     1004H := AAH
```

```
ORG 2000H
MOV AL,NUM0 ; AL:= 80H
ADD AL,AL ; AL:=AL + AL = 80H + 80H = 00H  $\Rightarrow$  ZNVC=1011
INC NUM1 ; NUM1:=NUM1 + 1 = C8 + 1 = C9
MOV BH,NUM1; BH:= C9H
MOV BL,BH ; BL:= C9H
DEC BL; BL:=C9 - 1 = C8H
SUB BL,BH; BL:=BL - BH= C8 - C9 = FFH ( -1 )  $\Rightarrow$  ZNVC=0101
MOV CH,BYTE1; CH:= AAH
AND CH,BYTE0 ; CH:= AA AND 7F = 2AH
NOT BYTE0 ; Contenido de 1003H := 80H
OR CH,BYTE0; CH:= 2AH OR 80H = AAH
XOR CH,11111111B CH:= AAH XOR FFH = 55H
HLT
END
```

6)

```
ORG 1000H
INI DB 0
FIN DB 15

ORG 2000H
MOV AL,INI ; AL:= 0
MOV AH,FIN ; AH:= 15
SUMA: INC AL ; AL:=AL + 1
      CMP AL,AH ; AL – AH y afecta los flags. Compara AL y AH.
      JNZ SUMA ; El salto se ejecuta siempre que su condición sea verdadera.
              ; En este caso “Salta Si No Es Cero” y se chequea el flag de Z
              ; para ver si la resta anterior dio cero, o sea AL=AH

      HLT
      END
```

El lazo se ejecuta 15 veces hasta que AL:= 15 y la resta da cero. La comparación es una resta que no guarda el resultado pero modifica los flags. Cuando AL llega a 15 AL – AH da cero y la condición del salto es falsa y el programa termina.

JS

Si se reemplaza por la instrucción “Salta Si Hay Signo”, AL siempre es < que AH, entonces será verdadera la condición del lazo, es decir el resultado será negativo hasta que AL=15 y el resultado de la resta sea cero y la condición del salto sea falsa, entonces terminará el programa con AL =15.

JZ

Si se reemplaza por la instrucción “Salta Si Es Cero”, entonces la primera vez AL=1 y la resta con AH=15 no da cero, entonces sólo se ejecutará una vez el lazo (instrucciones antes de CMP) y terminará con AL=1.

JMP

Si se reemplaza por la instrucción “Salto Incondicional”, no se chequea ninguna condición entonces el programa no terminará nunca.

7) Consideramos números en BSS

if (A < B) then C=A else C=B

```
        CMP AL, BL
        JS salto1 ; si hay signo (es negativo) se cumple AL<BL
        MOV CL, BL ; instrucción del else
        JMP salto2
salto1 : MOV CL, AL ; instrucción del then
salto2 : (sigue el programa)
```

if (A ≤ B) then C=A else C=B

```
        CMP AL, BL
        JZ salto1 ; si es cero el resultado (Z=1) se cumple AL=BL
        JS salto1 ; si no es cero y si hay signo (es negativo) se cumple AL<BL
        MOV CL, BL ; instrucción del else
        JMP salto2
salto1 : MOV CL, AL ; instrucción del then
salto2 : (sigue el programa)
```

if (A = B) then C=A else C=B

```
        CMP AL, BL
        JZ salto1 ; si es cero el resultado (Z=1) se cumple AL=BL
        MOV CL, BL ; instrucción del else
        JMP salto2
salto1 : MOV CL, AL ; instrucción del then
salto2 : (sigue el programa)
```

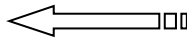
8)

```
ORG 1000H
TABLA DB 2,4,6,8,10,12,14,16,18,20 ; Dir desde 1000H hasta 1009H
FIN    DB ? ; Dir 100AH
TOTAL DB ? ;Dir 100BH
MAX    DB 13 ; Dir 100CH

ORG 2000H
MOV AL,0
MOV CL, OFFSET FIN - OFFSET TABLA ; Cantidad de números en TABLA
MOV BX, OFFSET TABLA ; Dir de comienzo de TABLA
SUMA:  ADD AL,[BX] ; AL:= AL + contenido de la dir apuntada por BX
        INC BX ; Apunto al elemento que sigue
        DEC CL ; Decremento la cuenta , cuando llego a cero no hay más elementos
        JNZ SUMA ; Voy a la etiqueta SUMA hasta que CL llegue a 0
        HLT
        END
```

Para que almacene en TOTAL, la suma quedó almacenada em AL

```
MOV TOTAL, AL
```



9)

```
ORG 1000H
TABLA DB 2,4,6,8,10,12,14,16,18,20 ; Dir desde 1000H hasta 1009H
FIN    DB ? ; Dir 100AH
TOTAL DB ? ;Dir 100BH
MAX    DB 13 ; Dir 100CH

ORG 2000H
MOV AL, MAX ; Carga el valor a comparar
MOV CH,0 ; Cuenta la cantidad de elementos iguales y menores a MAX
MOV CL, OFFSET FIN - OFFSET TABLA ; Cantidad de números en TABLA
MOV BX, OFFSET TABLA ; Dir de comienzo de TABLA
SIGO:  CMP AL, [BX] ; Compara AL con cada elemento de TABLA
        JNZ MENOR ; Si la comparación no dio 0 (no son iguales) hay que ver si es menor
        INC CH ; Acá cuenta si son iguales
        JMP NOMENOR ; Si pasó por acá hay que ir al final
MENOR: JS NOMENOR ; Se fija si es menor
        INC CH ; Cuenta si es menor
NOMENOR: INC BX ; Para todos los casos apunta al que sigue
        DEC CL ; Decrementa la cantidad de elementos, cuando llega a cero termina.
```

```

JNZ SIGO
MOV TOTAL, CH
HLT
END

```

10)

```

ORG 2000H
MOV AX, 1
MOV BX, 1000H ; BX tiene el comienzo de la tabla
CARGA : MOV [BX], AX
ADD BX, 2 ; Incrementa de a 2 porque cada elemento tiene 16 bits (2 bytes)
ADD AX, AX ; Multiplica por 2
CMP AX, 200 ; Compara el contenido de AX con 200. Mientras exista signo (resultado
              negativo) salta a CARGA y sigue. Cuando AX >=200 no hay signo y no
              se ejecuta el salto
JS CARGA
HLT
END

```

11)

```

ORG 1000H
DIR DW 3000H
MAX DW 52

ORG 2000H
MOV BX, DIR
MOV [BX], 0 ; Primer elemento del arreglo 0 x 5
ADD BX, 2 ; Apunta al próximo elemento (cada elemento=2bytes)
MOV AX, 0
SIGO : INC AX ; En AX almaceno los números 1,2,.....MAX
MOV DX, 0 ; Almacenamos en DX=AX . 5 (Inicializamos en 0)
MOV CL, 5 ;
LAZO:  CMP CL, 0 ; Multiplica por 5 mediante sumas repetidas (ej. 2 x 5
              JZ FIN ; es sumar 5 veces 2.
              ADD DX, AX ; En DX se almacena el producto.
              DEC CL
              JMP LAZO
FIN:   MOV [BX], DX ; Almacena en memoria (arreglo) el múltiplo de 5
ADD BX, 2 ; Apunta al siguiente elemento del arreglo
CMP AX, MAX ; Para saber si llegué al último número a multiplicar
JNZ SIGO ; (MAX).
HLT
END

```

12)

```
ORG 1000H
X      DB 17
TABLA DB 100 DUP (0) ; Reserva 100 bytes en memoria
        ; con valor inicial 0

ORG 2000H
MOV BX, OFFSET TABLA ; BX apunta al comienzo de TABLA
MOV AL, X
ARRIBA: CMP AL, 0 ; para saber si llegué a 0
        JZ FIN
SIGO:   MOV AH, AL ; trabajo sobre AH así el valor de AL no se ve
        ; afectado
        AND AH, 00000001b ; El último bit me dice si es par o impar
        JZ PAR
        ADD AL, 5 ; es impar suma 5
        JMP AFUERA
PAR:    SUB AL, 7 ; es par resta 7
AFUERA: MOV [BX], AL ; almacena el nuevo valor en el arreglo
        INC BX ; apunta al próximo elemento del arreglo
        JMP ARRIBA
FIN:    HLT
        END
```

13)

```
ORG 1000H
FRASE DB "Organización y la Computación"
FIN   DB ?

ORG 2000H
MOV BX, OFFSET FRASE; BX=1000H
MOV AL, "a"
MOV AH, "c"
MOV CL, OFFSET FIN - OFFSET FRASE
ARRIBA: MOV CH, [BX]
SIGO1:  CMP CH, AL
        JNZ SIGO ; el caracter no es una "a"
        DEC CL ; sigo por acá, es una "a" y un caracter menos
        JZ FINAL ; si es el último terminé
        INC BX ; sino apunto al que sigue
        MOV CH, [BX] ; cargo el que sigue
        CMP CH, AH ; me fijo si es una "c"
        JNZ SIGO1 ; sino es "c" no cuento y me tengo que fijar si es "a"
        INC DL ; si es una "c" cuento 1 más
SIGO:   INC BX ; apunto al que sigue
        DEC CL
        JNZ ARRIBA ; sino es el último sigo arriba
FINAL:  HLT
```

