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
                ; ⟹ BL:=CA , mdd Directo
  MOV  BH,0FFH ; Carga en BH el valor FF  ⟹  BH:= FF,  mdd Inmediato
  MOV  CH,BL ; Carga en CH el valor de BL  ⟹  CH:=BL , mdd  por Registro
  MOV  AX,BX ; Carga en AX el valor de BX  ⟹  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)
  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		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  ⟹  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 )  ⟹  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 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.

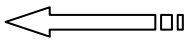
8)

```
          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
```

```
        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
```