Escribir un programa en assembler para recorrer una tabla de bytes que comienza en TABLA y cuya longitud se encuentra en la dirección [DS:100h] el programa deberá poner a partir de la dirección [DS:500h] en forma ;consecutiva aquellos valores pares superiores al promedio de los números de bit7=0 y bit2=1.

```
.model small

.stack 100 ; en el caso de no poner nada guardaba unos 1024 por la dudas

.data
    tabla db ?  ;defino tipo de dato, son bytes

.code

    mov ax, @data ; estas son combinaciones necesarias para que vea donde
deja el SO
    mov ds, ax    ; los datos, sino el micro no los ve, es decir se equivoca


    mov di,ds:[0100h] ;cargo la longitud del arreglo
    xor bx,bx
otro:
    mov al,tabla[bx]
    mov ah,al
    and ah, 84h   ;enmascaro para guardarme el bit 7 y bit 2, es resto los
                  ;fuerzo a "0"

    cmp ah,04h    ;pregunto por la condición del problema bit7=0 y bit2=1
    jne sigo
    xor ah,ah     ;lo hago cero "ah" xq voy a usar un registro de 16bits para
que me alcance
    inc cx        ;como acumulador de la suma dx, sin hacer lio con el valor
ah,
    add dx, ax    ;cx es el contador de los que cumplen con la condición
sigo:
    inc bx
    cmp bx,di
    jl otro   ;salta por menor xq el arreglo se recorre desde "0"

mov ax,bx ;acá podría preguntar si es "0"para saber divido o no (no lo puse)
div cx  ;el resultado queda en "al" promedio

;ahora realizo una nueva lectura del arreglo para copiar los valores mayores
;al promedio a partir de la dirección [DS:500H]


    mov si,500h  ; cargo la dirección de destino, "di" aun tiene la cantidad
de valores
    xor bx,bx ;inicializo el índice del arreglo, es un direccionamiento base
leo:
    mov ah, tabla [bx]
    cmp ah,al ;en "al" aun esta el promedio
    jle incremento
    mov [si],ah
    inc si
incremento:
    inc bx
```

```
    cmp bx, di
    jl leo

int 20h  ;termino el programa

end
```

Escribir un programa en assembler para recorrer una tabla de bytes que comienza en ds:100h y termina en ds:2E1h, aquellos valores impares quedaran inalterados y los otros cambiarán su contenido de la siguiente forma: antes = b7b6b5b4b3b2b1b0 después=b3b2b1b0 b5b6b4b7

```
.model small

.stack 100 ; en el caso de no poner nada guardaba unos 1024 por la dudas

.data
    ;no se usa esta sección xq ya tenemos las direcciones del arreglo

.code

;    mov ax, @data ;no hace falta
;    mov ds, ax


    mov di,100 ;cargo dirección de inicio
 otro:
    mov al,[di]
    mov ah,al
    and ah,01h   ;enmascaro para guardarme el bit0 el resto los fuerzo a "0"
    cmp ah,01h    ;pregunto por la condición del problema bit0=1
    je sigo

    call modificar
    mov [di], al
sigo:
    inc di
    cmp di,2E1h ;dirección de fin
    jle otro   ;salta mientras no lea todo

int 20h  ;termino el programa


modificar proc
    mov ah,al
    and ah,0Fh ;me quedo con la parte menos significativa del byte
    and al,0F0h ;me quedo con la parte más significativa del byte


    shl ah,4 ;desplazo la parte menos significativa 4 lugares a la izquierda
             ;queda b3b2b1b0 0000

    rcl al,1
    adc dl,00 ;pongo el bit7 como bit0

    rcl al,1
    adc dh,00
    shl dh,02
```

```
    add dl,dh ;en dl ya tengo 0000 0b60b7

    xor dh,dh
    rcl al,1
    adc dh,00
    shl dh,03
    add dl,dh ; en dl tengo 0000 b5b60b7

    xor dh,dh
    rcl al,1
    adc dh,00
    shl dh,1
    add dl,dh ;en dl queda 0000 b5b6b4b7

    add ah,dl ;en ah queda lo pedido b3b2b1b0 b5b6b4b7


    ret
modificar endp

end
```

Mostrar un conjunto de caracteres que están ubicados en un arreglos llamado “tabla”, dar más de una forma de hacerlo.

```
.model small

.stack 100 ; en el caso de no poner nada guardaba unos 1024 por la dudas

.data
    tabla db 48h,"o",6Ch,01100001b,115,"$"  ;defino tipo de dato, son bytes

.code

    mov ax, @data ; estas son combinaciones necesarias para que vea donde
deja el SO
    mov ds, ax    ; los datos, sino el micro no los ve, es decir se equivoca



    xor bx,bx
    mov ah,2
uno:

    mov dl,tabla[bx]       ;carga datos con direccionamiento base
    int 21h                ;bx actua de indice comenzando en "0"
    inc bx
    cmp bx,5
    jl uno


    mov dl,0Dh
    int 21h
    mov dl,0Ah
    int 21h
```

```
    lea si, tabla            ;carga la direccion efectiva de tabla
dos:
    mov dl,[si]
    int 21h
    inc si
    cmp si,5
    jl dos


    mov dl,0Dh
    int 21h
    mov dl,0Ah
    int 21h

    mov si, offset tabla       ;con la direccion de comienzo
tres:
    mov dl,[si]                ;usando el modificador offset
    int 21h
    inc si
    cmp si,5
    jl tres

    mov dl,0Dh
    int 21h
    mov dl,0Ah
    int 21h

;cuatro
    mov ah,9
    mov dx, offset tabla          ;como string
    int 21h

    mov ah,1        ;para poder ver las salidas, toco cualquier
    int 21h         ;tecla y termina

    int 20h

end
```
