EXERCICE EMU8086

1) Initialisation d'un tableau T de n nombres non signés, tel que n est saisi en entrée et chaque nombre est représenté sur 2 octets (les nombres saisis doivent être en décimal). Le nombre d'éléments n est supérieur ou égal à 100 et inférieur ou égal à 200 Exemple : 4556 26398, etc.

2) Affichage des éléments du tableau, après lecture, en Hexadécimal

3) Définir le maximum des éléments du tableau et son affichage en décimal

4) Définir le minimum des éléments du tableau et son affichage en hexadécimal

5) Générer un nouveau tableau NewT, dont les éléments sont le produit des éléments de T par le minimum défini dans la partie1, divisé par 16 comme suit : NewT[i] = T[i]*(minimum/16)

6) Affichage du tableau NewT, en binaire

Code :

```
data segment
    newline   db 10,13,'$'
    txt1      db ' / en hexa : $'
    txt2      db ' / en binaire : $'
       newtabtxt db 10,13,'Les elements du nouveau tableau
:',10,13,'$'
    maxtxt    db 10,13,'Le Max en decimal est : $'
    mintxt    db 10,13,'Le Min en hexadecimal est : $'
    tailletxt db 13,10,'Entrer la taille du tableau : $'
    valtxt    db 13,10,'Entrez une valeur : $'
    vals      db 5 dup ?
    n         dw ?
    max       dw ?
    min       dw ?
    tab       dw n dup ?
    newtab    dw n dup ?
ends

stack segment
    dw 128 dup(0)
ends

code segment
    mov  ax,data
    mov  ds,ax
    mov  ax,stack
    mov  ss,ax

    re:
    mov  ax,00h
    mov  cx,00h

    mov  ah,09h
    mov  dl,offset tailletxt
```

```
int  21h

call StringNumber
mov  cl,vals[1]
mov  di,02h
mov  ax,00h
call String2Number
mov  n,ax
push n
;cmp  n,200
;jg   re
;cmp  n,100
;jl   re
mov  cx,n
mov  si,00h

tabi:
mov  ah,09h
mov  dl,offset valtxt
int  21h
push cx

call StringNumber
mov  cl,vals[1]
mov  di,02h
mov  ax,00h

call String2Number
mov  tab[si],ax

mov  ah,09h
mov  dl,offset txt1
int  21h
mov  ax,tab[si]

call AffHexa
pop  cx
add  si,02h
loop tabi

pop  n
mov  cx,n
call MaxMin

mov  ah,09h
mov  dl,offset maxtxt
int  21h
mov  ax,max
call AffDec

mov  ah,09h
mov  dl,offset mintxt
int  21h
mov  ax,min
push ax
call AffHexa
```

```
    pop  ax
    mov  bx,10h
    mov  dx,00h
    div  bx

    push ax
    mov  si,00h
    mov  cx,n
    mov  ah,09h
    mov  dl,offset newtabtxt
    int  21h
    pop  ax

    newt:
    push ax
    push cx
    mov  di,00h
    mov  cx,n
    vv:
    push tab[di]
    add  di,02h
    loop vv
    push di
    mov  bx,tab[si]
    mul  bx
    mov  newtab[si],ax
    mov  bx,newtab[si]
    push bx
    call AffDec
    pop  bx
    mov  ah,09h
    mov  dl,offset txt2
    int  21h
    call AffBin
    mov  ah,09h
    mov  dl,offset newline
    int  21h
    mov  cx,n
    pop  di
    v:
    sub  di,02h
    pop  tab[di]
    loop v
    add  si,02h
    pop  cx
    pop  ax
    loop newt

ends
HLT

StringNumber proc
    mov  dx,offset vals
    mov  ah,0Ah
    int  21h
```

```
    ret
StringNumber endp

String2Number proc
    q:
    mov  dx,10
    mul  dx
    mov  dl,vals[di]
    sub  dl,30h
    add  ax,dx
    inc  di
    loop q
    ret
String2Number endp

MaxMin proc
    mov  si,00h
    w:
    cmp  cx,n
    jl   h
    mov  ax,tab[0]
    mov  min,ax
    mov  max,ax
    jmp  z
    h:
    mov  ax,min
    cmp  tab[si],ax
    jl   valmin
    u:
    mov  ax,max
    cmp  tab[si],ax
    jg   valmax
    jmp  z
    valmin:
    mov  ax,tab[si]
    mov  min,ax
    jmp  u
    valmax:
    mov  ax,tab[si]
    mov  max,ax
    z:
    add  si,02h
    loop w
    ret
MaxMin endp

AffDec proc
    mov  bx,0Ah
    mov  cx,00h
    empiler:
    mov  dx,00h
    div  bx
    add  dx,30h
    push dx
    inc  cx
    cmp  ax,00h
```

```
    jne  empiler
    depiler:
    mov  ah,02h
    pop  dx
    int  21h
    loop depiler
    ret
AffDec endp

AffHexa proc
    mov  bx,10h
    mov  cx,00h
    emp:
    mov  dx,00h
    div  bx
    cmp  dx,09h
    jg   hex
    add  dx,30h
    push dx
    jmp  i
    hex:
    cmp  dx,0Ah
    je   a
    cmp  dx,0Bh
    je   b
    cmp  dx,0Ch
    je   c
    cmp  dx,0Dh
    je   d
    cmp  dx,0Eh
    je   e
    cmp  dx,0Fh
    je   f
    jmp  i
    a:
    mov  dx,41h
    push dx
    jmp  i
    b:
    mov  dx,42h
    push dx
    jmp  i
    c:
    mov  dx,43h
    push dx
    jmp  i
    d:
    mov  dx,44h
    push dx
    jmp  i
    e:
    mov  dx,45h
    push dx
    jmp  i
    f:
    mov  dx,46h
```

```
    push dx
    i:
    inc  cx
    cmp  ax,00h
    jne  emp
    dep:
    mov  ah,02h
    pop  dx
    int  21h
    loop dep
    ret
AffHexa endp

AffBin proc
    mov  cx,10h
    print:
    mov  ah,02h
    mov  dl,'0'
    test bx,1000000000000000b
    jz   zero
    mov  dl,'1'
    zero:
    int  21h
    shl  bx, 1
    loop print
    ret
AffBin endp
```