Ordinateurs et Systèmes d'Exploitation
Exercices – IX

1. Écrire un programme assembleur 80x86 capable de mesurer la plus longue série d'octets consécutifs identiques dans une suite d'octets donnée. Le registre AX pointe initialement vers cette suite d'octets dont la longueur est donnée dans CX. La longueur de la plus longue série sera placée dans le registre DX en fin d'exécution.

   Solution On parcourt les octets de la suite en conservant dans le registre AX la longueur de la série actuelle d'octets consécutifs identiques. La valeur des octets de cette série est contenue dans le registre BL. Le registre DX est mis à jour, si cela est nécessaire, lors d'un changement de valeur dans la suite et aussi en fin d'exécution.

```
          SECTION .text
          cmp cx, 1
          ja init
          mov dx, cx
          ret
init:     mov bp, ax
          mov dx, 0
          mov ax, 1
          dec cx
          mov bl, [bp]
boucle:   inc bp
          cmp bl, [bp]
          jne max
          inc ax
          jmp suite
max:      cmp dx, ax
          jae reset
          mov dx, ax
reset:    mov ax, 1
          mov bl, [bp]
suite:    loop boucle
          cmp dx, ax
          jae fin
          mov dx, ax
fin:      ret
```

2. Écrire un programme de tri en assembleur 80x86. Le registre BX pointe initialement vers une suite d'octets représentant des nombres entiers signés (un nombre par octet). La longueur de cette suite est donnée dans CX et est non nulle. En fin d'exécution, les octets de la suite doivent être rangés par ordre croissant (l'efficacité n'étant pas une priorité dans le cadre de ce problème, un nombre d'opérations effectuées quadratique en fonction du nombre d'octets à traiter est considéré acceptable).

   Solution La routine commence par déplacer en tête de la suite l'octet de plus faible valeur. Ensuite on augmente BX d'une unité pour pointer vers le second octet de la suite originale, on diminue CX d'une unité pour refléter la taille de la nouvelle suite et on recommence l'opération sur la sous-suite.

```
            SECTION .text
tri:        mov al, [bx]
            mov si, 1
boucle:     cmp si, cx
            jae next_bx
            cmp al, [bx + si]
            jle next_si
            xchg al, [bx + si]
            mov [bx], al
next_si:    inc si
            jmp boucle
next_bx:    inc bx
            loop tri
            ret
```

3. Une entreprise propose un prix différent pour son produit suivant la quantité demandée par le client : les 100 premiers produits commandés par un client coûtent toujours 25 euros alors que les suivants ne coûtent plus que 20 euros. Cette entreprise aimerait connaître la somme totale qu'elle va recevoir de ses clients en fonction de leurs commandes.

   On demande d'écrire un programme assembleur 80x86 effectuant ce calcul. Le registre BX pointe initialement vers un tableau d'octets qui représentent chacun le nombre de produits commandés par un client (en représentation binaire non signée). Le nombre de commandes est donné dans le registre CX.

   Le programme doit renvoyer la somme totale due par les clients dans le registre AX.

   Exemple : Si le tableau contient les trois valeurs 50, 220 et 123, la somme totale est $(50 * 25) + (100 * 25 + 120 * 20) + (100 * 25 + 23 * 20) = 9110$.

   Solution Après avoir traité le cas où CX est nul, on met à zéro le registre DX qui servira d'accumulateur. On parcourt alors les différents octets, on les compare au nombre 100 et on ajoute à DX le montant, calculé différemment suivant le résulat de la comparaison. En fin d'exécution, on n'oublie pas de placer le contenu de DX dans AX, comme demandé.

```
            SECTION .text
            cmp cx, 0
            jne init
            mov ax, 0
            ret
init:       mov dx, 0
boucle:     mov al, [bx]
            cmp al, 100
            ja beaucoup
            mul byte 25
            jmp suite
beaucoup:   add dx, 2500
            sub al, 100
            mul byte 20
suite:      add dx, ax
            inc bx
            loop boucle
            mov ax, dx
            ret
```