Ordinateurs et Systèmes d'Exploitation
Exercices – VIII

1. La suite de Fibonacci $F_0, F_1, F_2, \ldots$ est définie ainsi :
   $F_0 = 0$
   $F_1 = 1$
   $F_2 = F_0 + F_1 (= 1)$
   $F_3 = F_1 + F_2 (= 2)$
   $F_4 = F_2 + F_3 (= 3)$
   ...
   $F_{i+2} = F_i + F_{i+1}$
   ...
   Les premiers termes de la suite sont donc 0 1 1 2 3 5 8 13 21 34 55 ...
   Chaque terme est la somme des deux précédents.
   Écrire un programme assembleur 80x86 calculant F_n, le $(n+1)$-ième nombre de la suite de Fibonacci. La valeur de n est initialement contenue dans le registre CX. La valeur de F_n devra se trouver dans le registre AX après l'exécution du programme.
   Par exemple, si CX contient initialement la valeur 10, le registre AX contiendra après l'exécution du programme la valeur 55 ($= F_{10}$).
   Le résultat final et les résultats intermédiaires peuvent être supposés représentables sur 16 bits.

   Solution Après avoir traité les cas particuliers CX=0 et CX=1, le calcul est effectué par une boucle. Après i itérations de celle-ci, le registre AX contient F_{i-1} et BX contient F_{i-2}.

```
            SECTION .text
            cmp cx, 1
            ja suite
            mov ax, cx
            ret
suite:      mov bx, 0
            mov ax, 1
            dec cx
boucle:     add bx, ax
            xchg bx, ax
            loop boucle
            ret
```

2. Écrire un programme assembleur 80x86 détectant la présence de 3 "1" consécutifs dans un tableau d'octets.
   Le registre BX pointe initialement vers ce tableau d'octets. Le nombre d'octets contenus dans le tableau est donné dans le registre CX et est supposé non nul.
   Le programme doit renvoyer la valeur 1 dans le registre AX s'il existe au moins trois octets consécutifs égaux à 1 dans le tableau donné. Sinon le registre AX doit contenir la valeur 0 à la fin du programme.

Solution On va utiliser le registre DX comme un compteur, initialisé à 0. Il est mis à jour après la lecture de chaque octet du tableau : si la valeur de l'octet est 1, on augmente DX d'une unité (si DX vaut alors 3, on quitte notre routine), sinon on réinitialise DX à 0.

```
          SECTION .text
          mov ax, 0
          mov dx, 0
boucle:   cmp byte [bx], 1
          jne reset
          inc dx
          cmp dx, 3
          jne suite
          mov ax, 1
          ret
reset:    mov dx, 0
suite:    inc bx
          loop boucle
          ret
```

3. Un palindrome est un mot qui se lit de façon identique de gauche à droite et de droite à gauche. Par exemple, le mot "RADAR" est un palindrome.

   Écrire un programme assembleur 80x86 capable de reconnaitre un palindrome. Le mot à tester est représenté par une séquence de caractères codés en ASCII et placés dans des octets consécutifs du segment de données. L'adresse du premier octet de ce mot est initialement contenue dans le registre BX et la longueur du mot est donnée dans CX.

   Le but du programme demandé est de placer dans le registre AL la valeur 1 si le mot pointé par BX est un palindrome et 0 dans le cas contraire.

   Solution Dans une boucle, on va comparer les caractères pointés par BX+SI et BX+DI. Avant la boucle, BX+SI donnera l'offset du premier caractère et BX+DI l'offset du dernier caractère du mot à tester. Après chaque itération, on incrémentera SI et on décrémentera DI. La boucle se terminera lorsque l'on constatera une différence entre les caractères comparés (AL=0) ou bien lorsque SI sera devenu supérieur ou égal à DI (AL=1).

```
          SECTION .text
          mov al, 1
          mov si, 0
          mov di, cx
          dec di
boucle:   cmp si, di
          jge fin_1
          mov ah, [bx + si]
          cmp ah, [bx + di]
          jne fin_0
          inc si
          dec di
          jmp boucle
fin_0:    mov al, 0
fin_1:    ret
```