

## **TP03 : Manipulation de la Pile et des Interruptions sous EMU8086**

### **Exercice 1 :**

Soit la chaîne de caractère suivante :

« Filières d'ingénieur GTR et GInfo »

1. Programme qui permet d'afficher cette chaîne de caractère est le suivant :

```
Pile Segment Para Stack
    DB 256 DUP (0)
Pile ends

Donnees Segment Para
    Chaine DB "Filières d'ingénieur GTR et GInfo"
Donnees ends

Code Segment Para
    Assume CS : Code
    Assume DS : Donnees
    Assume SS : Pile

Debut:  Mov AX, Donnees
        Mov DS, AX
        LEA DX, Chaine      ; ⇔  Mov Dx, offset Chaine
        Mov AH, 9
        Int 21h

        Mov AH, 4Ch
        INT 21h

Code Ends
End Debut
```

2. Programme qui permet d'introduire une chaîne de caractère via le clavier et l'afficher à l'envers :

```

Pile Segment Para Stack
    DB 256 DUP (0)
Pile ends

Donnees Segment Para
    msg1 DB "Saisir une chaîne de caractère pour arrêter entrez : $ "
    msg2 DB "Inverse de la chaîne saisie est : $ "
Donnees ends

Code Segment Para
    Assume CS : Code
    Assume DS : Donnees
    Assume SS : Pile

Debut:  Mov AX, Donnees      } ; Initialiser le segment de données
        Mov DS, AX
        LEA DX, msg1
        Mov AH, 9            } ; Affichage du msg1
        Int 21h
        Call saisie
        Jmp Fin

Saisie:  Mov CL, 00
        Mov AH, 1            } ; Lire le caractère entré, le résultat est
        Int 21h              } ; stocké dans AL

        Cmp AL, 58           ; 58 est le code ASCII du caractère ':'
        JE Affiche
        Mov AH, 00h
        Pusch AX
        Inc CL
        Jmp Saisie
```

```

Affiche: Mov AH,2
          Mov DL, 10
          Int 21h
          } ; Passer à une nouvelle ligne

          Lea DX, msg2
          Mov AH, 9
          Int 21h
          } ; Afficher le msg2

Boucle: POP AX
        Mov DL, AL
        Mov AH, 2
        Int 21h
        Dec CL
        Cmp CL, 00
        JNE Boucle
Ret

Fin :    Mov AH, 4Ch
        INT 21h
        } ; Pour stopper le programme

Code Ends
End Debut

```

## Exercice 2 :

Le programme qui permet de lire les caractères frappés au clavier et les affichent à l'écran tant que l'utilisateur n'appuie pas sur « ; »

```

; Déclaration du segment de pile
; Déclaration du segment de données

Code Segment Para
    Assume CS : Code
    Assume DS : Donnees
    Assume SS : Pile

Debut: Mov AX, Donnees
        Mov DS, AX
        } ; Initialisation du segment de données DS

```

```

Boucle: Mov AH, 1          } ; Lire le caractère entré, le résultat est
        Int 21h           } ; stocké dans AL

        Cmp AL, 59         ; 59 est le code ASCII du caractère ' ;'
        JE  Fin
        Call Affiche
        Jmp Boucle

Affiche: Mov DH, 00
        Mov DL, AL
        Mov AH, 02h       } ; Affichage du caractère entré
        Int 21h
        Ret

Fin:     Mov AH, 4Ch       } ; Arrêt du programme
        Int21h

Code Ends
End Debut

```

### Exercice 3 :

#### 1. Programme qui permet d'afficher un nombre entier (**Ex : le nombre entier 118**)

```

Pile Segment Para Stack
        DB 256 DUP (0)
Pile ends

Donnees Segment Para
        Centaines DB ?
        Dizaines DB ?
        Unites DB ?
Donnees ends

Code Segment Para
        Assume CS : Code
        Assume DS : Donnees
        Assume SS : Pile
Debut:  Mov AX, Donnees
        Mov DS, AX
        Mov ax, 118

```

```

Mov BL,100
Div BL           ; AL ← AX/BL et reste → AH
Mov Centaines, AL

Mov AL, AH
Mov AH, 00h
Mov BL, 10
Div BL           ; AL ← AX/BL et reste → AH
Mov Dizaines, AL
Mov Unites, AH

Add Centaines, 30h
Mov DL, Centaines
Mov AH, 2h
Int 21h          } ; Affichage des Centaines

Add Dizaines, 30h
Mov DL, Dizaines
Mov AH, 2h
Int 21h          } ; Affichage des Dizaines

Add Unites, 30h
Mov DL, Unites
Mov AH, 2h
Int 21h          } ; Affichage des Unites

Mov AH, 4Ch
Int 21h

Code Ends
End Debut

```

**NB: Centaines, Dizaines et Unités sont des variables.**

2. Programme qui permet d'afficher la valeur **52612** en binaire [**52612** ⇔ **1100110110000100b**]

```
; Déclaration du segment de pile
; Déclaration du segment de données
Code Segment Para
    Assume CS : Code
    Assume DS : Donnees
    Assume SS : Pile

Debut:  Mov AX, Donnees
        Mov DS, AX
        Mov CX, 16
        Mov BX, 52612

Boucle: Cmp CX, 00
        JLE Fin
        ROL BX, 1
        Jnc Etiq1
        Mov DL, '1'
        Jmp Affiche

Etiq1:  Mov DL, '0'
Affiche: Mov AH, 02h
        Int 21h
        Dec CX
        Jmp Boucle

Fin:    Mov AH, 4Ch
        INT 21h

Code Ends
End Debut
```

#### Exercice 4 :

Programme qui permet de calculer le maximum de deux nombres et l'affiche sur l'écran :

Pile Segment Para Stack

DB 256 DUP (0)

Pile ends

Donnees Segment Para

Nb1 DW 120h

Nb2 DW 253h

Max DW ?

Centaines DB ?

Dizaines DB ?

Unites DB ?

Donnees ends

Code Segment Para

Assume CS : Code

Assume DS : Donnees

Assume SS : Pile

Debut: Mov AX, Donnees

Mov DS, AX

Mov AX, Nb1

Cmp AX, Nb2

JG Sup

JL Inf

Jmp Fin

Sup: Mov Max, AX

Call Affiche

Jmp Fin

Inf: Mov BX, Nb2

Mov Max, Bx

Call Affiche

Jmp Fin

Affiche:

; C'est le programme de l'exercice 3 (question1). Il faut juste remplacer  
; le nombre **118** par **Max**

**Ret**

Fin: Mov AH, 4Ch

INT 21h

Code Ends

End Debut

### Exercice 5 :

Programme qui permet de trouver le nombre des '0' dans un mot « **Ex : 0011101001011110** » :

; Déclaration du segment de pile

; Déclaration du segment de données

Code Segment Para

Assume CS : Code

Assume DS : Donnees

Assume SS : Pile

Debut: Mov AX, Donnees

Mov DS, AX

Mov CX, 16

Mov BX, 0011101001011110b

Boucle: ROL BX, 1

Jnc Etiq1

Jmp Etiq2

Etiq1: Inc **Nzero** ; **Nzero** est une variable déclarée dans le segment de données

Etiq2: Dec CX

Cmp CX, 00

Jne Boucle

Call Afficher ; **Afficher** est le sous-programme qui permet d'afficher  
; un nombre entier (**Voir Exercice 3, question 1**)



```
Fin:      Mov AH, 4Ch
          INT 21h
Code Ends
End Debut
```

### Exercice 6 :

Un programme qui permet de calculer le **PGDC** (plus grand diviseur commun) de deux nombres **30** et **20**. Les deux valeurs sont transmises au sous-programme appelé via la pile. Le résultat est stocké dans le registre BX.

Pile Segment Para Stack

DB 256 DUP (0)

Pile ends

Donnees Segment Para

Nb1 DW 20

Nb2 DW 30

PGDC DW ?

Centaines DB ?

Dizaines DB ?

Unites DB ?

Donnees ends

Code Segment Para

Assume CS : Code

Assume DS : Donnees

Assume SS : Pile

Debut: Mov AX, Donnees

Mov DS, AX

Push Nb1

Push Nb2

Call Sousprog

Mov **PGDC**, BX

Call Afficher

Jmp Fin

```

Sousprog: Mov BP, SP
          Mov AX, [BP + 4]
          Mov CX, [BP + 2]
          Cmp AX,CX
          Je Etiqu1

Etiqu2:   Cmp AX,CX
          JG Etiqu3
          Xchg AX, CX

Etiqu3:   Sub AX, CX
          Cmp AX,CX
          Je Etiqu1
          Jmp etiqu2

Etiqu1:   Mov BX, AX
          Ret

Fin:      Mov AH, 4Ch
          INT 21h

Code Ends
End Debut

```

Afficher :

- ; C'est le programme de l'exercice 3 (question1). Il faut juste remplacer
- ; le nombre **118** par **PGDC**