Série de TD N°5

Exercice 1 : (Les instructions de branchement)

1. Ecrivez un code assembleur qui permet de calculer la somme des chiffres de 1 à 9.

2. Ecrivez un code assembleur qui permet d'effectuer un saut si la somme de deux nombres est supérieure à 10.
3. Ecrivez un code assembleur qui permet d'effectuer un saut si un nombre est pair.

Exercice 2 : (Boucles et conditions en assembleur)

1. On veut additionner deux nombres signés N1 et N2 se trouvant respectivement aux offsets 1100H et 1101H. Le résultat est rangé à l'offset 1102H s'il est positif, à l'offset 1103H s'il est négatif et à l'offset 1104H s'il est nul. Donnez l'organigramme qui présente le problème donnée et le programme en langage assembleur associé

2. Ecrire le code (uniquement les suites d'instructions) Assembleur pour faire la somme de dix (10) éléments d'un tableau. Le premier étant rangé à la case mémoire ayant l'adresse "0F68H" ? (chaque nombre est sur un octet et la somme peut dépassée 255)

Exercice 3 : (Appel des procédures, adressage segmenté et gestion de la pile)

- **Dans le programme principal :**
  1. Ecrivez un programme en utilisant les 02 instructions PUSH et POP pour la gestion d'une pile de 4 éléments.
- **Dans le sous programme:**
  2. Ecrivez le programme Assembleur 8086 contenant les deux procédures suivantes :
  - ✓ L'affichage des éléments d'une pile
  - ✓ L'addition d'un nombre quelconque pour chaque élément

Exercice 4: (Gestion des entrées/sorties)

Expliquez les interruptions 21h les plus utilisés suivantes en donnant des exemples avec les instructions de l'assembleur 8086 :

- La fonction 09
- La fonction 02
- La fonction 01
- La fonction 0AH

Correction Série de TD N°5

Exercice 1 : 1. Ecrivez un code assembleur qui permet de calculer la somme des chiffres de 1 à 9.

1. Ecrivez un code assembleur qui permet de calculer la somme des chiffres de 1 à 9.

L'une des solutions pour additionner les chiffres de 1 _a 9 pourrait être la suivante:

```
MOV AX,0
MOV BX,1
MOV CX,10
BOUCLE: ADD AX, BX     ; résultat de l'addition dans AX 1,2, 3……..9
INC BX
CMP BX, 10             comparaison jusqu'à indicateur TF achevé « boucle LOOPNE il faut une condition »
LOOPNE BOUCLE      ; do    while
```

1. Ecrivez un code assembleur qui permet d'effectuer un saut si la somme de deux nombres est supérieure à 10.

Le code assembleur pour effectuer un saut si la somme de deux nombres est supérieure _a 10 est le suivant:

```
 MOV AX,#nombre1
MOV BX,#nombre2
ADD AX, BX
CMP AX, 10  ; résultat dans AX
JG saut     ;  jump ou saut si  valeur de AX elle est supérieur à 10
   saut: …
```

2. Ecrivez un code assembleur qui permet d'effectuer un saut si un nombre est pair.

Pour vérifier si un nombre est pair, nous devons le diviser sur 2 et vérifier le reste de la division. Si le reste est 0 le nombre est pair sinon, il est impair.

Le code assembleur pour faire ca est le suivant:

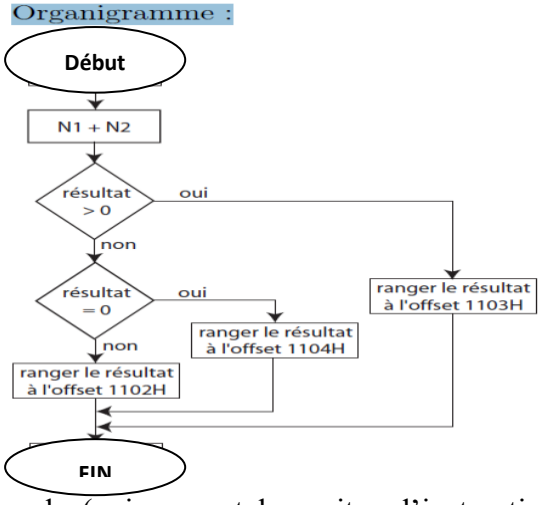
```
MOV AX, #nombre    ; le contenu du rY + la valeur 5.
MOV BX, 2          ; 2  présenter sur un octet le reste dans AH
DIV  BX            ; diviser AX sur 2. Le reste est stocker dans AH.
CMP AH, 0
JE SAUT      ; effectuer un saut si  le reste de division égal à zero « est nb paire ».
…  ;reste du code.
```

- **DIV :** Si **l'opérande est un octet**, alors **on récupère le quotient** dans le registre **AL** et **le reste dans le registre AH**.
  Si **l'opérande est un mot**, alors **on récupère le quotient** dans le registre **AX** et **le reste dans le registre DX**.

Exercice 2 :

1. On veut additionner deux nombres signés N1 et N2 se trouvant respectivement aux offsets 1100H et 1101H. Le résultat est rangé à l'offset 1102H s'il est positif, à l'offset 1103H s'il est négatif et à l'offset 1104H s'il est nul. Donnez l'organigramme qui présente le problème donnée et le programme en langage assembleur associé

Organigramme :



Programme :

```
                mov     al,[1100H]
                add     al,[1101H]
                js      negatif
                jz      nul
                mov     [1102H],al
                jmp     fin
negatif :       mov     [1103H],al
                jmp     fin
nul :           mov     [1104H],al
fin :           hlt
```

2. Ecrire le code (uniquement les suites d'instructions) Assembleur pour faire la somme de dix (10) éléments d'un tableau. Le premier étant rangé à la case mémoire ayant l'adresse "0F68H" ? (chaque nombre est sur un octet et la somme peut dépassée 255)

```
mov bx, 0F68H    ;BX contient l'adresse du 1er élément
mov cx, 10       ;CX (compteur) contient le nombre d'élément du tableau
```

```
mov ax, 0        ;AX contient la somme et initialisé à 0
mov dx, 0        ;initialiser DX à zéro, le sous-registre DL va contenir
                 ;l'élément lu de la mémoire, DH restera toujours zéro et sera
                 ;utile pour l'addition des 2 registres AX et DX

repeat:
     mov dl, [bx]     ;pour lire 1 seul octet de la mémoire
     add ax, dx       ;DH étant à zéro faire la somme de 2 registre 16bits
     inc bx           ;incrémenter BX pour lire l'elt suivant du tableau
     loop repeat
```

Exercice 3 : (Appel des procédures, adressage segmenté et gestion de la pile)

Dans le programme principal :

3. Ecrivez un programme en utilisant les 02 instructions PUSH et POP pour la gestion d'une pile de 4 éléments.

Dans le sous programme:

4. Ecrivez le programme Assembleur 8086 contenant les deux procédures suivantes :

- ✓ L'affichage des éléments d'une pile
- ✓ L'addition d'un nombre quelconque pour chaque élément

Solution :

```
Data segment
        N1  dw  3      ; déclaration des 4 éléments
        N2 dw 4
        N3 dw 5
        N4 dw 6
ends
stack segment
        dw 128  dup (0)     ;  déclararion de la pile ou chaque element est de 16 bits
ends
start:
 mov   ax, data
mov  ds, ax                 ; stocker les données dans le registre segment données ds pour les empliler
mov   ax , stack
mov    ss, ax                ; stocker la pile dans le registre de segment ss
push    N1
push    N2                  ; empiler les éléments dans la pile
push    N3
push    N4
call    fct_affichage               ; Appel de la fonction d'affichage
call  fct_addition                  ; Appel de la fonction d'addition
fct _affichage   proc  near    ;  nom_fonction    type « procédure »    near «  appel dans le même programme »
mov   cx, 4      ; initialiser le compteur de la boucle avec 4 itérations ( 4 éléments)
boucle:
      mov    ah, 02h      ; Affichage (en code ASCII)
       POP  dx        ; stocker les données  empilées dans le registre général  dx
       Int    21h
       Loop  boucle
Ret    ; retour au prgm principal
Endp
```

```
call fct_addition    proc near    ; nom_fonction   type « procédure »   near « appel dans le même programme »
mov   cx, 4      ; initialiser le compteur de la boucle avec 4 itérations ( 4 éléments)
boucle:
     ADD    dx, 10      ; Ajouter 10 pour chaque élément
     mov    ah, 02h      ; Affichage (en code ASCII)
     POP  dx         ; stocker les données  empilées dans le registre général  dx
      Int    21h
      Loop  boucle
Ret    ; retour au prgm principal
Endp

Mov   ax, 4cOOh              ; Vider le registre AX à la fin du programme
Int 21h                      ; fin de programme
ends
```

Exercice 4: (Gestion des entrées/sorties)

Expliquez les interruptions 21h les plus utilisés suivantes en donnant des exemples avec les instructions de l'assembleur 8086 :

- La fonction 09
- La fonction 02
- La fonction 01
- La fonction 0AH

Solution

- La fonction 09 : Affiche le contenu de registre Dx et arrete l'affichage jusqu'à quand le caractère $, sera trouvé (signe arrêt d'affichage)
- Exemple : CH DB ' HELLO', $; CH « label »

```
Mov  dx, offset   CH
MOV   AH, 09   ; activer l'interruption
Int  21h
```

- La fonction 02 : Affiche le contenu de registre Dl « un seul caracter ou nb »

Exemple :
```
mov  DL, 'j'
Mov  AH, 02          ; afficher:  j   affiche code ASCII
Int  21h
```

- La fonction 01 : Affiche le premier caractère taper sur clavier, qu'est stocker dans le registre AL
  Exemple : mov AH, 01 ; AL= 3

  Int 21h

- La fonction 0AH : Affiche les caractères taper sur clavier nbrs fixés par l'utilisateur. La saisie s'arrête quand on tape la touche de retour, le caractère CR (13) est mémorisé avec la chaîne
  Exemple : ORG 100h

  Data :
  Nb_max DB 6 DUP(FF)

  Mov AH, 0AH

  Mov DX, offset *Nb_max*

  Int 21h

Explications:

0001	0002	0003	0004	0005	0006	
FF	FF	FF	FF	FF	FF	:Avant
6	A	B	C	CR	FF	:Après