TD N° 3

Exercice 2 :

Soit un microprocesseur ayant un bus d'adresses de 16 bits, pouvant accéder à une donnée de 8 bits.

1. Quelle est la capacité d'adressage totale ?

Puisque le bus d'adresse supporte au maximum une adresse codée sur 16 bits, dans ce cas on ne peut accéder qu'à 2^{16} adresse mémoire même si la mémoire possède d'autres emplacements mémoires. 2^{16} = 65536

2. Quelle est la capacité de cette mémoire ?

Chaque emplacement mémoire est composé de 8 bits de stockage. Dans notre cas, on a 65 536 emplacements de 8 bits soit 65 536 Octet, ou bien 64 Kilo-Octet = 64 x 1024 Octet = 65 536 Octet.

3. Représenter la structure de cette mémoire (adresse de début et adresse de fin en hexadécimal).

@ de début : 0000h -------- > @ de fin : FFFFh

Exercice 3 :

Soit une mémoire de 64 Mo organisée en cases de 16 bits.

1. Déterminer le nombre de lignes d'adresses nécessaires.

64 Mo = 64 x 1024 x 1024 octet.

Si la mémoire est organisée par cases de 8 bits, on aura 67 108 864 cases.

Si la mémoire est organisée par cases de 16 bits, on aura (67 108 864 / 2) = 33 554 432 cases.

2. Dans le cas de cases de 16 bits, si l'adresse de début est $(0)_{10}$, déterminer l'adresse de fin en hexadécimal.

L'@ de fin est : 33 554 432 – 1 = $(33\ 554\ 431)_{10}$ = $(01FF\ FFFF)_{16}$

Exercice 4 :

1. Si l'intervalle des adresses d'une mémoire va de 0000H à FFFEH. Combien cette mémoire a de cases ?

Le nombre des cases mémoires est : (FFFEh – 0000h) + 1 = FFFFh = $(65535)_{10}$

2. Si une mémoire possède 5120 emplacements en mémoire. Donner l'intervalle (début et fin) de ces adresses exprimées en hexadécimal.

@ de début est 0000h. @ de fin est : 5120 – 1 = $(5119)_{10}$ = 13FFh

3. Si l'intervalle des adresses d'une mémoire va de 0531H à F20DH. Combien cette mémoire a de cases ?

Le nombre de cases est : (F20Dh - 0531h) + 1h = ECDEh = $(60638)_{10}$

Exercice 5 :

1. Donner les adresses physiques des mémoires telles que les adresses logiques sont comme suit :

3500:AB00 ; 1CD0:1111 ; 0022:FFFF

La règle est écrite comme suit : (Valeur Segment x 16) + Valeur Offset. Ceci est équivalent à un décalage à gauche en hexadécimal : (Valeur Segment x 10h) + Valeur Offset.

L'@ physique de 3500:AB00 est : 3500h x 10h + AB00h = 35000h + AB00h = 3FB00h = $(260864)_{10}$

L'@ physique de 1CD0:1111 est : 1CD0h x 10h + 1111h = 1CD00h + 1111h = 1DE11h = $(122385)_{10}$

L'@ physique de 3500:AB00 est : 0022h x 10h + FFFFh = 0220h + FFFFh = 1021Fh = $(66079)_{10}$

2. Proposer une (des) adresse(s) segment : offset pour les mémoires d'adresse physique : 10000h ; FFFFFh ; 00000h

Soit A une adresse physique. Tout couple (S, O) satisfaisant l'équation : A = S x 10h + O, peut être considéré comme un couple de valeurs Segment/Offset valide. Par exemple :

10000h = 1000h x 10h + 0000h ---- > Segment = 1000h ; Offset = 0000h

10000h = 500h x 10h + 5000h ---- > Segment = 500h ; Offset = 5000h

FFFFFh = FEEEh x 10h + 111Fh ---- > Segment = FEEEh ; Offset = 111Fh

FFFFFh = FEE0h x 10h + 11FFh ---- > Segment = FEE0h ; Offset = 11FFh

00000h = 0000h x 10h + 0000h ---- > Segment = 0000h ; Offset = 0000h (Pas d'autres possibilités)

TD N° 4

Exercice 1 :

1. Créer huit variables de type octet, nommées B0 à B7 et contenant les valeurs décimales 1, 255, -1, 'e', les valeurs hexadécimale 32, ff et les valeurs binaires 1, 11111111.

```
B0 DB 1d                                          B4 DB 032h
B1 DB 255d                                        B5 DB 0FFh
B2 DB -1d   ; (ou bien en Hex avec Cà2 : B2 DB FFh )   B6 DB 1b
B3 DB 'e'   ; (ou bien en code ASCI B3 DB 65 )    B7 DB 11111111b
```

2. Créer les mêmes variables de type mot que vous nommerez D0, D1, … D7

```
D0 DW 1d                                          D4 DW 032h
D1 DW 255d                                        D5 DW 0FFh
D2 DW -1d ;(ou bien en Hex avec Cà2 : D2 DB FFFFh)     D6 DW 1b
D3 DW 'e'  ;(ou bien en code ASCI : D3 DW 65h)    D7 DW 11111111b
```

3. Créer ensuite un tableau de 10 octets qui contient lui aussi ces valeurs. Faites de même avec un tableau de 10 mots.

```
TableauB DB 1d, 255d, -1d, 'e', 032h, 0FFh, 1b, 11111111b
```

4. Créer ensuite un tableau de 152 octets contenant la valeur décimale 111 et un tableau de 150 mots contenant la valeur décimale 43981.

```
TableauB DB 152 DUP(111)

TableauD DW 150 DUP(43981)
```

Exercice 2 :

Préciser la location OFFSET de chaque donnée dans le segment de donnée suivant :

```
DONNEE SEGMENT

d1 DB 55H                    ; Offset = 0 = 0000h

d2 DW 2560H                  ; Offset = 1 = 0001h

m1 DW 02                     ; Offset = 3 = 0003h

m2 DB "Tel : 25607080"; Offset = 5 = 0005h

Tab1 DB 6 DUP (253)          ; Offset = 19 = 0013h

Tab2 DW 12 DUP (?)           ; Offset = 25 = 0019h

DONNEE ENDS
```

Exercice 3 :

Indiquer le contenu de chacun des registres suivants : BL, BH, et BX après l'exécution de cette instruction : Mov BX,0D74Eh

BL et BH sont deux registres de 8 bits chacun, les deux forment le registre BX qui est de taille 16 bits. Dans notre cas BX = D74E et donc, BH contient les bits de poids le plus fort D7, et BL contient les bits de poids le plus faible 4E.

BX 16 bits	
BH = D7	BL = 4E
8 bits	8 bits

Exercice 4 :

Montrer les contenus de la destination dans chacun des cas suivants (avec SI=2000h et l'adresse DS:12DAh=3C01h)

Mov AX, 0C21h ; AX reçoit 0C21h

Mov DI, 12DAh ; DI reçoit 12DAh

Mov BX, DI ; BX reçoit la valeur de DI soit 12DAh

Mov [SI], AX ; L'adresse mémoire DS:2000h reçoit la valeur de AL (21) et 2001h reçoit la valeur de AH (21)

Mov BX, [DI] ; BX (16 bits) reçoit la valeur contenue dans l'adresse mémoire DS:12DAh (BL = 01h, 8 bits) et
; DS:12DBh (BH = 3Ch, 8 bits)

Exercice 5 :

Dans le cas où les registres ont les valeurs suivantes :

AX = 56EFh, BX = 100Ch, CX = 256Ch, DX = 002Eh, DS = 0B9Ch, SS = 0C40h, CS = 0B99h, SI = 0021h, DI = 2042h, BP = 5400h

L'adresse physique (ou effective) est calculée comme suit : Adresse Physique = (Valeur Segment x 16) + Valeur Offset. Ceci est calculé facilement en hexadécimal via le décalage à gauche de la valeur hexadécimale du segment (équivalent à la multiplication par 16) et en ajoutant la valeur de l'offset.

1. + 2. Quels sont les modes d'adressage correspondants à chacune des instructions ? Calculer l'adresse physique de la mémoire où l'opérande est sauvegardé, ainsi que le contenu des locations mémoires dans chacune des instructions suivantes :

a. Mov [SI], AL ; L'@ physique = B9C0h + 0021h = B9E1h. Le contenu de l'@ B9E1h = EFh
; Mode d'adressage Indirecte Index

b. Mov [SI+1], AH ; L'@ physique = B9C0h + (0021h + 0001h) = B9E2h. Le contenu de l'@ B9E2h = 56h
; Mode d'adressage Indirecte Index + déplacement

c. Mov [SI], AX ; L'@ physique = B9C0h + 0021h = B9E1h. Le contenu de l'@ B9E1h = EFh
; Mode d'adressage Indirecte Index

d. Mov [DI+100], CS ; L'@ physique = B9C0h + (2042h + 64h) = DA66h. Le contenu de l'@ DA66h = 99h
; Mode d'adressage Indirecte Index + déplacement

e. Mov [BX], DX ; L'@ physique = B9C0h + 100Ch = C9CCh. Le contenu de l'@ C9CCh = 2Eh
; Mode d'adressage Indirecte Base + déplacement

f. Mov [SI+BX+81h], CX ; L'@ physique = B9C0h + 0021h + 100Ch + 81h = CA6Eh. Le contenu de l'@ = 6Ch
; Mode d'adressage Indirecte Base avec Index + déplacement

g. Mov [BP+150h], AX ; L'@ physique = C400h + (5400h + 150h) = 11950h. Le contenu de l'@ = EFh
; Mode d'adressage Indirecte Base + déplacement

h. Mov [BP][DI+75h], BX ; L'@ physique = C400h + (5400h + 2042 + 75h) = 1206Fh. Le contenu de l'@ = 0Ch
; Mode d'adressage Indirecte Base avec Index + déplacement

Exercice 6 :

Donner le rôle de chaque instruction suivante et le contenu des registres/mémoires correspondant :

a. Mov AL,2 ; Le registre AL reçoit comme valeur 2

b. Mov AL, [25] ; Le registre AL reçoit comme valeur le contenu de l'adresse DS:0019h

c. Mov [B4F], AX ; avec AX = B4F, l'@ DS :0B4F reçoit la valeur 0B4Fh

d. Mov AX, [B4F] ; Le registre AX reçoit la valeur contenue dans l'@ DS:0B4F

e. Mov AL, [DI] ; Le registre AL reçoit comme valeur le contenu de l'adresse DS:DI

f. Mov BP,05 ; Le registre BP reçoit la valeur 5

g. Mov [BP],05 ; L'@ SS:BP reçoit la valeur 5

h. Mov AL, DS : [BX] ; Le registre AL reçoit la valeur contenue dans l'@ DS:[BX]

i. Mov AL, DS : [DI] ; Le registre AL reçoit la valeur contenue dans l'@ DS:[DI]

j. Mov DS : [DI], AL ; L'@ DS:DI recoit la valeur contenue dans le registre AL

k. Mov AL, SS : [DI] ; Le registre AL reçoit la valeur contenue dans l'@ DS:[BX]

Exercice 7 :

On considère le contenu des registres suivants :

DS = 14B3h, CS = 8700h, SS = 5ACFh, AX = 87A5h, BX = 0054h, SI = 0008h, BP = 07C2h, DI = 0C87h, IP = 5ED7h

1. Donner le(s) mode(s) d'adressage possible(s) de chaque instruction

   a. Mov BP, BX ; Mode d'adressage Registre Directe

   b. Mov [SI], BP ; Mode d'adressage Indirecte Index

   c. Mov [57Eh], 0C5E2h ; Mode d'adressage Immédiat

   d. Mov [BP][DI+0Ah], DS ; Mode d'adressage Indirecte Base avec Index + déplacement

   e. Mov [IP], IP ; Mode d'adressage Indirecte

Les combinaisons d'adressage possibles pour l'opération Mov :

Mov registre, immédiat

Mov registre, registre ; a

Mov registre, mémoire

Mov mémoire, immédiat ; c

Mov mémoire, registre ; b, d et e

2- Donner le contenu de la mémoire ou du registre pour le cas des instructions n° : b, c et e.

b. Mov [SI], BP ; Le contenu de l'@ 14B3 :0008 reçoit la valeur C2h

c. Mov [57Eh], 0C5E2h ; Le contenu de l'@ 14B3 : 057E reçoit la valeur E2h

e. Mov [IP], IP ; Le contenu de l'@ 8700 : 5ED7h reçoit la valeur D7h

3- Calculer l'adresse physique de la case mémoire où l'opérande est sauvegardé pour les instructions b, d et e.

b. Mov [SI], BP ; L'@ physique = 14B30h + 0008h = 14B30h

d. Mov [BP][DI+0Ah], DS ; L'@ physique = 5ACF0h + (07C2h + 0C87h + 000Ah) = 5C143h

e. Mov [IP], IP ; L'@ physique = 87000h + 5ED7h = 8CED7h

TD N° 5

Exercice 1 :

1. Définir dans le segment de donnée deux cases mémoires, en mots, A et B.

A DW ?

B DW ?

2. Ecrire un programme permettant d'échanger les contenus de A et B.

ORG 100h	Directive

Mov AX, A
Mov DX, B
Mov B, AX
Mov A, DX
	Instructions

A DW 10 B DW 100	Déclarations

Exercice 2 :

1. Ecrire un programme qui fait la somme de deux variables de type mot (précédemment déclarées) et qui sauvegarde le résultat dans une troisième variable de type mot. Faites des tests avec des valeurs positives et négatives.

ORG 100h	Directive

Mov AX, A
Add AX, B
Mov C, AX
RET
	Instructions

A DW 10 B DW -100 C DW 0	Déclarations

Valeur de C = FFA6h = -90 (Complément à deux en mode 16 bits)

2. Ecrire un programme qui fait la somme de deux variables de type octet (précédemment déclarées) et qui sauvegarde le résultat dans une troisième variable de type mot.

ORG 100h	Directive

Xor AX, AX ; Equivalent à : Mov AX, 0
Mov AL, A
Add AL, B
Mov C, AX ; On ne peut pas passer AL (8 bits)
RET
	Instructions

A DB 10 B DB -100 C DW 0	Déclarations

3. Ecrire un programme qui fait la somme de deux variables de type mot (précédemment déclarées) et qui sauvegarde le résultat dans une variable de type octet si cela est possible ou dans une variable de type mot si le résultat ne tient pas dans une variable de type octet. Faire des tests avec les valeurs signées.

On ne peut pas affecter une variable de type mot à un type octet. On sauvegarde le résultat de la somme dans une variable de type mot. Voir la réponse de la question 1. On peut résoudre l'exercice en utilisant les branchements (Série 6).

Exercice 3 :

Tracer le programme ci-dessous en indiquant à chaque fois la valeur des indicateurs CF et OF

a.
```
Mov AL, 64h      ; CF = 0 et OF = 0
Mov BL, 2        ; CF = 0 et OF = 0
Mul BL           ; CF = 0 et OF = 0
```

b.
```
Mov AL, 64h      ; CF = 0 et OF = 0
Mov CL, 3        ; CF = 0 et OF = 0
Mul CL           ; CF = 1 et OF = 1
```

CF = OF = 0 quand AH (dans le cas de 8 bit) ou DX (dans le cas de 16 bit) est égal à zéro.

Exercice 4 :

Dans un registre de 8 bits, effectuer des opérations sur des nombres signés en donnant leurs résultats et en positionnant les indicateurs :

C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0
1	0	1	1	0	0	0	0
	1	0	1	1	0	0	0	0
+								
	1	0	1	1	1	1	0	0
=	0	1	1	0	1	1	0	0

SF = 0 CF = C_7 = 1

ZF = 0 OF = C_6 *xor* C_7 = 1

C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0
0	1	0	0	0	0	0	0
	0	1	0	1	0	0	0	0
+								
	0	1	1	0	0	0	0	0
=	1	0	1	1	0	0	0	0

SF = 1 CF = C_7 = 0

ZF = 0 OF = C_6 xor C_7 = 1

C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0
1	1	1	1	0	0	0	0
	1	1	1	1	0	0	0	0
+								
	0	0	0	1	0	0	0	0
=	0	0	0	0	0	0	0	0

SF = 0 CF = C_7 = 1

ZF = 1 OF = C_6 xor C_7 = 0

Exercice 5 :

Donner le contenu de AL et l'état des indicateurs ZF, CF, SF et OF après l'exécution des programmes suivants :

- ```
  MOV AL, 79h
  ADD AL, 30h
  ; Le contenu de AL est A9h, ZF = 0, CF = 0, SF = 1 et OF = 1
  ```

- ```
  MOV AL, A0h
  ADD AL, A0h
  ; Le contenu de AL est 140h, ZF = 0, CF = 1, SF = 0 et OF = 1
  ```

- MOV AL, 0h
  SUB AL, 1h
  ; Le contenu de AL est FFh, ZF = 0, CF = 1, SF = 1 et OF = 0

Exercice 6 :

1. Écrire un programme qui calcule x $\times 2^n$, où x et n sont deux variables positives sur 16 bits, stockées aux adresses 130h et 132h. Le résultat sera rangé dans 134h. On utilisera une instruction de décalage bit à bit.

```
ORG 100h
… ; Initialisation des cases mémoires avec x et n
Mov AX, [130h]
Mov CL, [132]          ; [132h] doit être de type octet !.
SHL AX, CL
Mov [134h], AX
RET
```

2. A l'aide de l'instruction SHR, écrire un programme qui divise par 8 la valeur contenue à l'adresse 0130h, et qui range le résultat en 0131h.

```
ORG 100h
… ; Initialisation de la case mémoire avec x
Mov AL, [130h]
Mov CL, 3
SHR AL, CL             ; ou bien directement : SHR AX, 3
Mov [131h], AL
RET
```

Exercice 7 :

1. Écrire un programme qui calcule $r = x \times y$, où x, y et r sont des variables naturelles de

a. 16 bits, rangées respectivement en 130h, 132h et 134h

```
…
Mov AX, [130h]
Mov CX, [132h]
Mul CX
Mov [134h], AX
Mov [136h], DX
```

b. 8 bits, rangées respectivement en 80h, 81h et 82h

```
…
Mov AL, [80h]

Mov CL, [81h]

Mul CL

Mov [82h], AL

Mov [83h], AH
```

2. Écrire un programme qui calcule *x / y*, *où x*, *y* sont des variables naturelles :

a. *x* est 16 bits et *y* est de 8 bits, rangées respectivement en 130h, 132h et le résultat à partir de 134h

```
…
Mov AX, [130h]

Mov CL, [132h]

Div CL

Mov [134h], AL

Mov [135h], AH
```

b. *x* est de 32 bits et *y* est de 16 bits, rangées respectivement en 80h, 90h et le résultat à partir de 82h

```
…
Mov AX, [80h]

Mov DX, [82h]

Mov CX, [90h]

Div CX

Mov [82h], AX

Mov [84h], DX
```

TD N° 6

Exercice 1 :
Convertissez les algorithmes ci-dessous en assembleur

1.If - then - else

If (AX > BX) {

Max = AX

} else {

Max = BX

}

2. La boucle for

BX = 5

AX = 2

For (CX = 0 ; CX < BX ; CX++) {

AX += CX

}

3. La boucle While

AX = 0

CX = 0

While (AX < 10) {

CX += 2 x AX

AX++
}

4. La boucle Repeat

BX = 5

Repeat 10 {

BX += BX

}

1.If - then - else

```
…
TestSI :
    CMP AX, BX
    JG MaxCestAX
   Mov Max, BX
   JMP FinSi
     MaxCestAX:
     Mov Max, AX
 FinSi :
…
```

2. La boucle for

```
…
Mov BX, 5
Mov AX, 2
Mov CX, 0
BoucleFor
   CMP CX, BX
   JNLE FinBoucleFor
   Add AX, CX
   Inc CX
   JMP BoucleFor
FinBoucleFor :
…
```

3. La boucle While

```
…
Mov AX, 0
Mov CX, 0
BoucleWhile :
   CMP AX, 10
   JNLE FinBoucleWhile
   Mov BX, AX
   Mov DX, 2
   Mul DX
   Add CX, AX
   Mov AX, BX
   Inc AX
   JMP BoucleWhile
FinBoucleWhile :
…
```

4. La boucle Repeat

```
…
Mov BX,5
Mov CX, 10
BoucleRepeat :
   Add BX, BX
LOOP BoucleRepeat
…
```

Exercice 2 :

Ecrire un programme, en langage assembleur 8086, qui permet de compter les nombres nuls dans un tableau d'octets mémoire de longueur 100 et débutant à l'adresse [200h], le résultat sera placé à l'adresse [400h].

```
; Initialisation du tableau
Mov BX, 200h
Mov CX, 50
InitialiserAun:
  Mov [BX], 1
  Inc BX
LOOP InitialiserAun
Mov CX, 20
InitialiserAzero:
  Mov [BX], 0
  Inc BX
LOOP InitialiserAzero
Mov CX, 30
InitialiserAdeux:
  Mov [BX], 2
  Inc BX
LOOP InitialiserAdeux      ; Suite ---->
```

```
; Compteur des zéros.
Mov AX, 0
Mov BX, 200h
Mov CX, 100
Boucle:
  CMP [BX], 0
  JNE TestFaux
  Inc AX
  TestFaux:
  Inc BX
LOOP Boucle
Mov [400h], AX
RET
```

Exercice 3 :
Soit un ensemble de programmes ayant le même segment de données.

```
Data Segment
Tab1 DB 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
Tab2 DB 16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1
Tab3 DB 16,13,10,8,1,15,3,2,5,4,8,9,7,14,11,12
resTab1 DB 16 DUP (0)
resTab2 DB 16 DUP (0)
resTab3 DB 16 DUP (0)
ends
```

Ecrivez les programmes suivants :

1. Le tableau resTab1 contient le tableau Tab1 ou l'on a ajouté 1 aux nombres impairs

```
ORG 100h
Mov SI, 0
Mov CL, 16
Mov BL, 2
Boucle:
    Mov AL, [Tab1 + SI]
    Div BL
    CMP AH, 0
    JE CestUnNombrePair
        Mov AL, [Tab1 + SI]      ; Suite ---->
        Inc AL
        Mov [resTab1 + SI], AL
        JMP IterationSuivante
    CestUnNombrePair:
        Mov AL, [Tab1 + SI]
        Mov [resTab1 + SI], AL
    IterationSuivante:
        Inc SI
LOOP Boucle
RET
```

2. Le tableau resTab2 contient le tableau Tab2 où on a enlevé 10 aux nombres. Si un nombre est négatif, on stockera 0 à sa place.

```
ORG 100h
Mov SI, 0
Mov CL, 16
Boucle:
    Mov AL, [Tab2 + SI]
    Sub AL, 10
    CMP AL, 0
    JNLE Positif      ; Suite ---->
        Mov [resTab2 + SI], 0
        JMP IterationSuivante
    Positif:
        Mov [resTab2 + SI], AL
    IterationSuivante:
        Inc SI
LOOP Boucle
RET
```

3. Le tableau resTab3 contient le tableau Tab3 ou l'on a ajouté 120 aux nombres pairs. Si un nombre de resTab3 est supérieur à 127 on met 127.

```
Mov SI, 0
Mov CL, 16
Mov BL, 2
Boucle:
    Mov AL, [Tab3 + SI]
    Div BL
    CMP AH, 0
    JE CestUnNombrePair
        Mov AL, [Tab3 + SI]
        Mov [resTab3 + SI], AL
        JMP IterationSuivante
    CestUnNombrePair:      ; Suite ---->
        Mov AL, [Tab3 + SI]
        Add AL, 120
        CMP AL, 0
        JLE PlusGrandque127
        Mov [resTab3 + SI], AL
        JMP IterationSuivante
    PlusGrandque127:
        Mov [resTab3 + SI], 127
    IterationSuivante:
        Inc SI
LOOP Boucle
RE
```

Exercice 4 :

1. Ecrire le programme qui calcule la somme des 11 premiers entiers (0 + 1 + 2…+ 10 + 11). On utilisera pour cela les instructions MOV, CMP, JNE, ADD, DEC ou INC… On utilisera une variable R pour stocker le résultat et une variable N pour stocker le nombre 11.

```
; Calcul de la Somme de 0 à N
Mov AL,0
Mov CL, N
Boucle:
  CMP CL, 0
  JE FinBoucle
  Add AL, CL
  Dec CL
  JMP Boucle
FinBoucle:          ; Suite ---->
Mov R, AL
RET
; Déclaration
N DB 11
R DB 0
```

2. Même question mais en utilisant l'instruction LOOP

```
; Calcul de la Somme de 0 à N
Mov AL,0
Mov CL, N
Boucle:
  Add AL, CL
LOOP Boucle         ; Suite ---->
Mov R, AL
RET
; Déclaration
N DB 11
R DB 0
```

Exercice 5 :

Donner la valeur de AX et de CX à la fin de l'exécution des programmes suivants :

1.
```
MOV AX, 0
MOV CX, 5
Boucle :
        INC AX
        LOOP Boucle
```

CX = 0 ; AX = 5

2.
```
MOV CX,5
DeBuT :
        INC CX
        LOOP DeBuT
```

CX = entre 5 et 6 (Boucle infinie) ; AX = Garde sa valeur initiale (N'intervient pas dans le programme)

Ce document a été Préparé par Mr. Mohamed Amine EL MAJDOULI. Veuillez envoyer vos questions à propos de ce corrigé à : elmajdouli@ieee.org

Exercice 6 :

Ecrire la suite des instructions qui permet de trouver les valeurs MAX et MIN d'un tableau de 1024 entiers signés rangés à partir de l'adresse B000H. On rangera la valeur MAX à l'adresse 0100h et la valeur MIN à l'adresse 0104h.

```
; Initialisation
Mov BX, 0B000h
Mov CX, 1000
InitialiserAzero:
  Mov AX, -3
  Mov [BX], AX
  Inc BX
  Inc BX
LOOP InitialiserAzero
Mov CX, 20
InitialiserAcentvingt:
  Mov AX, -120
  Mov [BX], AX
  Inc BX
  Inc BX
  LOOP InitialiserAcentvingt
Mov CX, 4
InitialiserAcent:
  Mov AX, 100
  Mov [BX], AX
  Inc BX
  Inc BX
LOOP InitialiserAcent          ; Suite ---->
```

```
; Chercher Min et MAX
Mov CX, 1024
Mov BX, 0B000h
Mov AX, -32768
Mov DX, 32767
Boucle:
  CMP [BX], AX
  JLE TestMin
  Mov AX, [BX]
  TestMin:
   CMP [BX], DX
   JNLE IterationSuivante
   Mov DX, [BX]
  IterationSuivante:
  Inc BX
  Inc BX
LOOP Boucle
Mov [100h], AX
Mov [104h], DX
RET
```

Exercice 7 :

1. Pour un segment de pile de 256 octets : Quelles sont les offsets du sommet de la pile :
   a. Dans le cas où la pile est vide (hexadécimale) :

   Le sommet de la pile vide pointe sur l'@ : FFFEh

   b. Dans le cas où la pile contient 30 éléments :

   FFFE - 003C= FFC2 : Le sommet de la pile décrémente à chaque empilement de deux octets, soit 30x2=60=3Ch

2. Ecrire un programme pour : (Le programme ci-dessous rassemble les questions a, b, c, d)
   e. Après l'empilement des éléments dans la pile, quelle est l'adresse de son sommet ?
   Le nombre de valeurs communs entre les deux tableaux est 33. FFFEh – 0042h = 0BCh : Le sommet de la pile décrémente à chaque empilement de deux octets, soit 33 x 2 = 66 = 42h

a. Déclarer deux tableaux de 100 mots, tab1 et tab2.
b. Remplir le premier avec les 100 premiers nombres positifs multiple de 2 et le second avec les 100 premiers nombres positifs multiple de 3.
c. Empiler dans la pile les éléments qui sont communs aux deux tableaux
d. Additionner les éléments de la pile en les dépilant et stocker le résultat à l'adresse hexadécimale 0FADh.

```
ORG 100h
Mov SI, 0
Mov CX, N
Mov AX, 0
Mov DX, 2
BoucleRemplireA2:
 Add AX, DX
 Mov [Tab1 + SI], AX
 Inc SI
 Inc SI
LOOP BoucleRemplireA2
Mov SI, 0
Mov CX, N
Mov AX, 0
Mov DX, 3
BoucleRemplireA3:
 Add AX, DX
 Mov [Tab2 + SI], AX
 Inc SI
 Inc SI
LOOP BoucleRemplireA3
Mov SI, 0
Mov AX, 0
Mov DX, 0
Mov CX, N
BoucleEmpilerCommun:
 Mov DI, 0
 Mov DX, [Tab1 + SI]
;  Suite ---------->
```

```
BoucleChercherOccurance:
 CMP DX, [Tab2 + DI]
 JNE elementSuivant
 Push DX
 Inc AX
 JMP FinBoucleChercherOccurance
 elementSuivant:
  Inc DI
  Inc DI
  CMP DI, D
  JE FinBoucleChercherOccurance
  JMP BoucleChercherOccurance
FinBoucleChercherOccurance:
  Inc SI
  Inc SI
LOOP BoucleEmpilerCommun
Mov BX, 0
Mov CX, AX
DepilerEtAdditionner:
pop DX
Add BX, DX
LOOP DepilerEtAdditionner
Mov [0FADh], BX
RET
; Déclaration
N DW 100
D DW 200
Tab1 DW N DUP(?)
Tab2 DW N DUP(?)
```