

TD N° 6

Exercice 1 :

Convertissez les algorithmes ci-dessous en assembleur

1. If - then - else

If (AX > BX) {

Max = AX

} else {

Max = BX

}

3. La boucle While

AX = 0

CX = 0

While (AX < 10) {

CX += 2 x AX

AX++

}

2. La boucle for

BX = 5

AX = 2

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

AX += CX

}

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 :

...

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 :

...

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 :

...

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	; Compteur des zéros.
Mov BX, 200h	Mov AX, 0
Mov CX, 50	Mov BX, 200h
InitialiserAun:	Mov CX, 100
Mov [BX], 1	Boucle:
Inc BX	CMP [BX], 0
LOOP InitialiserAun	JNE TestFaux
Mov CX, 20	Inc AX
InitialiserAzero:	TestFaux:
Mov [BX], 0	Inc BX
Inc BX	LOOP Boucle
LOOP InitialiserAzero	Mov [400h], AX
Mov CX, 30	RET
InitialiserAdeux:	
Mov [BX], 2	
Inc BX	
LOOP InitialiserAdeux	

; Suite ---->

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

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

```

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 où 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
    RET

```

Exercice 4 :

1. Ecrire le programme qui calcule la somme des 11 premiers entiers ($0 + 1 + 2 + \dots + 10 + 11$). 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.

<pre> ; 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 ----> </pre>	<pre> Mov R, AL RET ; Déclaration N DB 11 R DB 0 </pre>
---	--

2. Même question mais en utilisant l'instruction LOOP

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

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)

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	; Chercher Min et MAX
Mov BX, 0B000h	Mov CX, 1024
Mov CX, 1000	Mov BX, 0B000h
InitialiserAzero:	Mov AX, -32768
Mov AX, -3	Mov DX, 32767
Mov [BX], AX	Boucle:
Inc BX	CMP [BX], AX
Inc BX	JLE TestMin
LOOP InitialiserAzero	Mov AX, [BX]
Mov CX, 20	TestMin:
InitialiserAcentvingt:	CMP [BX], DX
Mov AX, -120	JNLE IterationSuivante
Mov [BX], AX	Mov DX, [BX]
Inc BX	IterationSuivante:
Inc BX	Inc BX
LOOP InitialiserAcentvingt	Inc BX
Mov CX, 4	LOOP Boucle
InitialiserAcent:	Mov [100h], AX
Mov AX, 100	Mov [104h], DX
Mov [BX], AX	RET
Inc BX	
Inc BX	
LOOP InitialiserAcent	

; Suite ---->

Exercice 7 :

- Pour un segment de pile de 256 octets : Quelles sont les offsets du sommet de la pile :
 - Dans le cas où la pile est vide (hexadécimale) :
Le sommet de la pile vide pointe sur l'@ : FFFh
 - Dans le cas où la pile contient 30 éléments :
FFFF - 003C = FFC2 : Le sommet de la pile décrémente à chaque empilement de deux octets, soit $30 \times 2 = 60 = 3C$
- Ecrire un programme pour : (Le programme ci-dessous rassemble les questions a, b, c, d)
 - 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. FFFh - 0042h = 0BCh : Le sommet de la pile décrémente à chaque empilement de deux octets, soit $33 \times 2 = 66 = 42h$

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

ORG 100h	BoucleChercherOccurance:
Mov SI, 0	CMP DX, [Tab2 + DI]
Mov CX, N	JNE elementSuivant
Mov AX, 0	Push DX
Mov DX, 2	Inc AX
BoucleRemplireA2:	JMP FinBoucleChercherOccurance
Add AX, DX	elementSuivant:
Mov [Tab1 + SI], AX	Inc DI
Inc SI	Inc DI
Inc SI	CMP DI, D
LOOP BoucleRemplireA2	JE FinBoucleChercherOccurance
Mov SI, 0	JMP BoucleChercherOccurance
Mov CX, N	FinBoucleChercherOccurance:
Mov AX, 0	Inc SI
Mov DX, 3	Inc SI
BoucleRemplireA3:	LOOP BoucleEmpilerCommun
Add AX, DX	Mov BX, 0
Mov [Tab2 + SI], AX	Mov CX, AX
Inc SI	DepilerEtAdditionner:
Inc SI	pop DX
LOOP BoucleRemplireA3	Add BX, DX
Mov SI, 0	LOOP DepilerEtAdditionner
Mov AX, 0	Mov [0FADh], BX
Mov DX, 0	RET
Mov CX, N	; Déclaration
BoucleEmpilerCommun:	N DW 100
Mov DI, 0	D DW 200
Mov DX, [Tab1 + SI]	Tab1 DW N DUP(?)
; Suite ----->	Tab2 DW N DUP(?)